

# Universidad Nacional de la Matanza

Departamento:

Ingeniería e Investigaciones Tecnológicas

Cátedra:

## Fundamentos de TIC's

(Tecnologías de la Información y la Comunicación)

JEFE DE CÁTEDRA:

Mg Artemisa Trigueros

### UNIDAD NRO. 3

#### **Introducción al Análisis y Síntesis de Estructuras Lógicas**

COLABORACIÓN:

DOCENTES DE LA CÁTEDRA

CICLO LECTIVO:

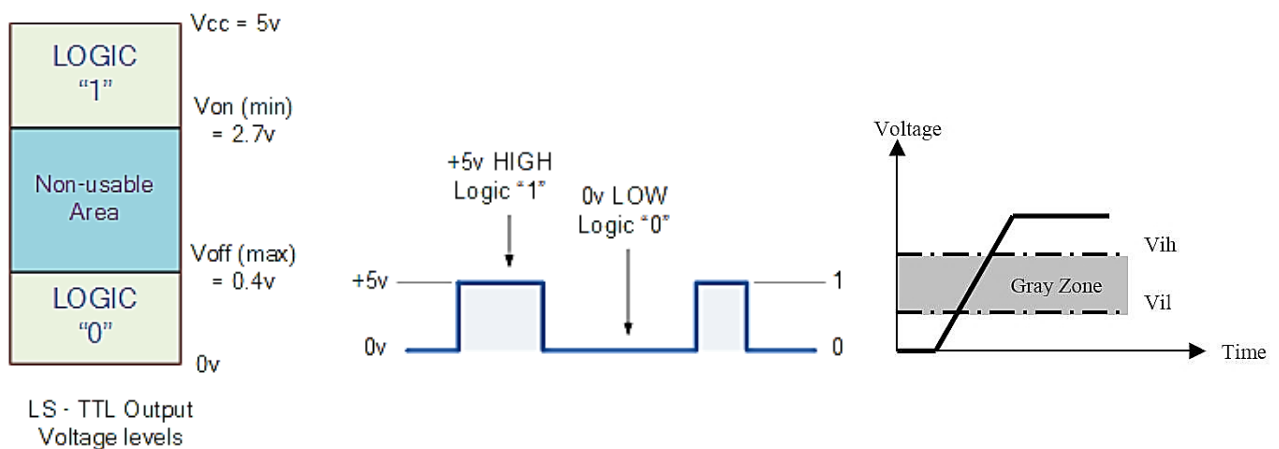
**2020**

## Fundamentos de TICs • Unidad 3

### Introducción al Análisis y Síntesis de Estructuras Lógicas

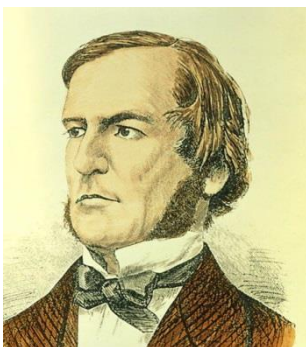
#### 1. Introducción

Habitualmente se utiliza el término *digital* para hacer referencia a la representación de la información de modo binario (en dos estados). Los sistemas digitales están constituidos por una combinación de dispositivos capaces de generar, transmitir, procesar y almacenar información representada en forma digital. Utilizan una lógica de *dos estados* que se representan por dos niveles de tensión eléctrica: alto (high) y bajo (low). Tales estados se simbolizan con unos y ceros, con el propósito de facilitar las aplicaciones lógicas y el desarrollo aritmético.



**Figura 1.** Estados lógicos, ejemplos.

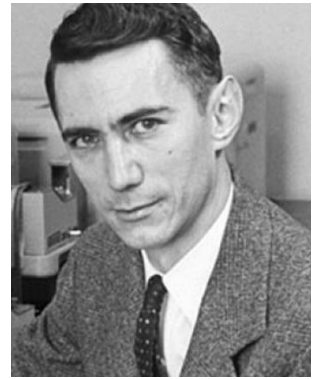
Para el análisis y la síntesis de los sistemas digitales se requiere un álgebra binaria. En 1854, George Boole desarrolló y publicó un sistema de reglas que permitían expresar, manipular y simplificar problemas lógicos por procedimientos matemáticos; y cuyos argumentos admiten dos estados (verdadero o falso). La lógica proposicional y la teoría de conjuntos tienen propiedades similares que se utilizan para definir la estructura del Álgebra de Boole. Su trabajo fue ampliado y perfeccionado por otros matemáticos, como Augustus De Morgan; sin embargo, en aquel entonces, parecía no tener utilidad práctica. En 1938, Claude Shannon demostró que el álgebra booleana se podía utilizar en el análisis y la síntesis de los circuitos digitales.



George Boole (1815–1864) nació en Inglaterra, en una familia de escasos recursos. Su carácter amable y modesto inspiró a todos sus amigos la estima más profunda. Trabajó durante toda su vida para sostener a su familia.

Pese a que recibió una medalla de la Royal Society y el título de Doctor Honoris Causa en Derecho por la Universidad de Dublín, no solicitó ni recibió los beneficios a los que sus descubrimientos le darían derecho. En 1849 fue nombrado Profesor de Matemática del Queen's College, en Cork (Irlanda), donde trabajó como docente el resto de su vida. En 1864 enfermó gravemente tras mojarse bajo la lluvia mientras caminaba hasta el aula donde daba clase. Murió poco después de pulmonía. Tenía 49 años. Como inventor del Álgebra de Boole, es considerado uno de los fundadores de las Ciencias de la Computación.

Desde joven, Claude Shannon (1916 – 2001) demostró una inclinación hacia la mecánica y la tecnología. Se destacaba en las asignaturas de ciencias. En 1936 obtuvo los títulos de Ingeniero Electricista y Matemático en la Universidad de Michigan, su ciudad natal. Su interés por la matemática y la ingeniería continuó durante toda su vida. En 1936 aceptó el puesto de asistente de investigación en el Departamento de Ingeniería Eléctrica en el Instituto Tecnológico de Massachusetts (MIT). En su tesis de maestría demostró cómo el álgebra booleana se podía utilizar en el análisis y la síntesis de la conmutación y los circuitos digitales. Estudió la eficacia de los diferentes métodos de transmisión de la información a través de medios sólidos (cables) como aéreos, por medio de ondas electromagnéticas.



Debido a una enfermedad, Augustus De Morgan (1806 – 1871) perdió la visión de uno de sus ojos poco después de nacer. Su padre falleció cuando él tenía diez años. La señora De Morgan residió en diversos lugares en Inglaterra, y su hijo recibió su educación primaria en varias modestas escuelas locales. A pesar de las dificultades, fue un estudiante destacado. A los 22 años, Augustus fue nombrado Profesor de Matemática en la Universidad de Londres. En clase, invitaba a sus estudiantes a sentarse a su lado y explicarle sus resultados; el razonamiento y la comprensión eran mucho más importantes para él que la destreza meramente analítica en la resolución de problemas. Además de su Tratado de Lógica Formal, publicó Elementos de Aritmética en 1830. Fue un apasionado del álgebra y puso de manifiesto la existencia de otras álgebras distintas de la ordinaria. Fue elegido miembro de la Real Sociedad Astronómica; sin embargo, nunca aceptó asociarse

a la Royal Society y también rechazó otros honores. Una de sus mayores contribuciones fue la introducción de las leyes lógicas que llevan su nombre, fundamentales en el desarrollo de las Ciencias de Computación.

El principio de funcionamiento de las computadoras está fundamentado en la aplicación de los circuitos de conmutación, o *circuitos lógicos*. Se denominan así los circuitos cuyos componentes realizan operaciones equivalentes a las que indican los operadores lógicos y las funciones del Álgebra de Boole. Un *conmutador* es un dispositivo capaz de encontrarse en sólo dos estados, abierto o cerrado, que pueden denotarse con 0 ó con 1 respectivamente.

Los sistemas digitales *combinacionales* son aquellos cuyas salidas sólo dependen del estado de sus entradas en un momento determinado. En ellos no existe la noción de memoria, debido a que las salidas no dependen de los estados previos de las entradas. Los sistemas digitales *secuenciales*, en cambio, son aquellos cuyas salidas dependen de estados previos, además del estado de sus entradas en un instante dado. En estos sistemas tiene origen el concepto de memoria, que almacena la información de la historia pasada del sistema.

## 2. Fundamentos de la lógica conmutacional

Hemos mencionado que el principio de funcionamiento de las computadoras está basado en la aplicación de los circuitos de conmutación, o *circuitos lógicos*. Se denominan así los circuitos cuyos componentes realizan operaciones equivalentes a las que indican los operadores lógicos y las funciones del Álgebra de Boole. Un *conmutador* es un dispositivo capaz de encontrarse en sólo dos estados, abierto o cerrado, que pueden denotarse con 0 ó con 1 respectivamente. Un interruptor eléctrico simple es un ejemplo de conmutador. La base 2 – para la elección de un sistema de numeración – era la más adecuada desde el punto de vista de la confiabilidad y el costo. Por esta razón los sistemas digitales<sup>1</sup> trabajan con elementos binarios (sólo pueden adoptar

<sup>1</sup> Aunque en los ejemplos señalados el término digital se ha relacionado siempre con dispositivos binarios, ello no significa que los términos *digital* y *binario* sean sinónimos. Por ejemplo, en el código Morse se utilizan cinco

dos valores). Se verá a continuación el enfoque desde el punto de vista de la lógica de circuitos eléctricos y juntamente con éste, se presentarán las compuertas lógicas básicas.

## 2.1. Parámetros eléctricos fundamentales

Algunos esquemas mecánicos, circuitos eléctricos, electrónicos y otros sistemas físicos, nos permiten resolver problemas lógicos. En la tabla siguiente se describen algunos parámetros eléctricos básicos.

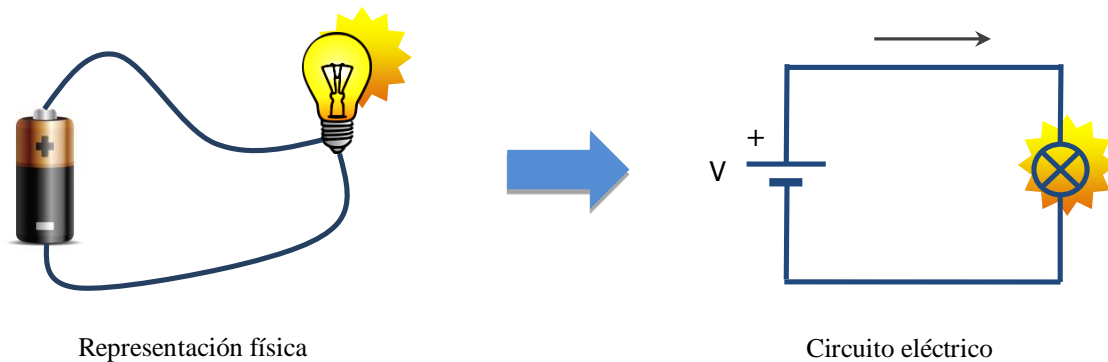
MAGNITUD (SÍMBOLO)	CARACTERÍSTICAS	UNIDAD
Tensión eléctrica (V)	<ul style="list-style-type: none"> <li>También llamada “diferencia de potencial” eléctrico entre dos puntos del espacio en la cercanía de una carga eléctrica, o fuerza electromotriz (fem).</li> <li>Es una magnitud física que impulsa a las cargas eléctricas a desplazarse.</li> <li>Producida por un generador eléctrico en un circuito.</li> </ul>	Volt (V)
Corriente eléctrica (I)	<ul style="list-style-type: none"> <li>Consiste en el movimiento de cargas eléctricas en el tiempo.</li> <li>Entregada por el generador eléctrico en un circuito cerrado.</li> <li>Corriente continua: producida por un generador de tensión constante.</li> <li>Corriente alterna: producida por un generador de tensión que cambia alternativamente de polaridad. Es la que suministran los generadores que se utilizan en los hogares para iluminación y electrodomésticos.</li> </ul>	Ampere (A)
Energía eléctrica (E)	<ul style="list-style-type: none"> <li>Es la capacidad de producir trabajo que tienen la tensión y la corriente en un circuito.</li> <li>En corriente continua, resulta del producto de la tensión por la corriente por el tiempo.</li> </ul>	Kilowatt hora (Kwh) o Joule (J)
Potencia eléctrica (P)	<ul style="list-style-type: none"> <li>Es la energía en la unidad de tiempo, es decir, resulta de dividir la energía por el tiempo.</li> <li>En corriente continua, resulta del producto de la tensión por la corriente.</li> </ul>	Watt (W)
Resistencia Eléctrica (R)	<ul style="list-style-type: none"> <li>Es la oposición al paso de la corriente.</li> <li>En un circuito, resulta del cociente entre tensión y corriente.</li> <li>Según la resistencia de los materiales al paso de la corriente, se clasifican en conductores, aisladores y semiconductores.</li> <li>Un material conductor no se opone al paso de la corriente eléctrica, es de baja resistencia. Ej.: Cobre (Cu).</li> <li>Un material aislante se opone al paso de la corriente eléctrica, tiene un elevado valor de resistencia. Ej.: Diamante (C).</li> <li>Un material semiconductor se comporta como aislador en bajas temperaturas y como conductor a temperatura ambiente (23 °C). Ej.: Silicio (Si).</li> </ul>	Ohm( $\Omega$ )

**Tabla 1.** Parámetros eléctricos fundamentales.

Una lámpara, un motor o un calefactor son ejemplos usuales de resistencias eléctricas. Un circuito eléctrico simple puede ayudar a comprender estos conceptos<sup>2</sup>.

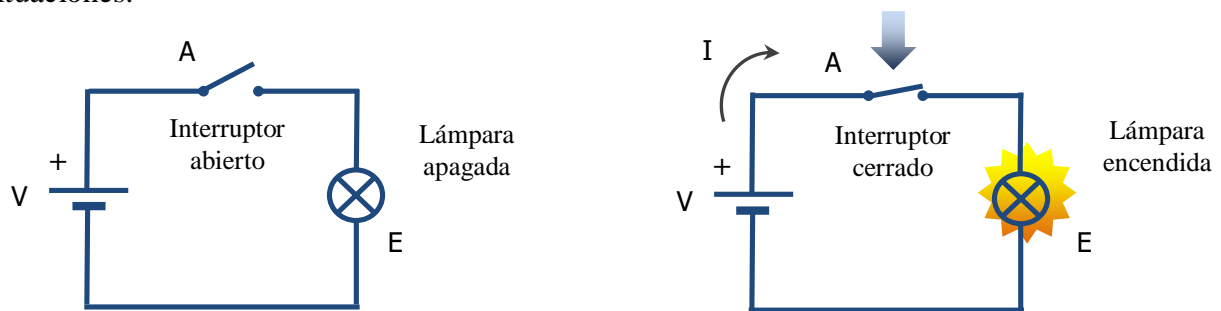
estados digitales: punto, raya, espacio corto (entre letras), espacio medio (entre palabras) y espacio largo (entre frases).

<sup>2</sup> Benítez, Guillermo. Álgebra Conmutacional y Compuertas Lógicas. 2011, Pág. 6.



**Figura 2.** Circuito eléctrico simple.

Para encender o apagar la lámpara, podría emplearse una lámina metálica con un pequeño pliegue que deja al circuito normalmente desconectado<sup>3</sup> o *abierto* en un extremo. Al ejercer presión sobre ella, el extremo abierto se pone en contacto, el circuito se *cierra* y la corriente eléctrica puede circular. Al dejar de ejercer presión, la lámina metálica se levanta debido a su elasticidad y la corriente se interrumpe nuevamente. A continuación se representan estas situaciones:



**Figura 3.** Posiciones posibles de un interruptor.

El proceso de permitir o no el paso de la corriente eléctrica, por ejemplo cambiando de una posición a otra el interruptor, se denomina **conmutación**<sup>4</sup>.

De la observación de la figura anterior:

- En el circuito de la izquierda, “NO” se ejerce presión sobre el interruptor “A”, por lo tanto, la lámpara “NO” se enciende.
- En el circuito de la derecha, “SÍ” se ejerce presión sobre el interruptor “A” y la lámpara “SÍ” se enciende.

Que la lámpara encienda o no, depende de la posición del interruptor “A”. En consecuencia, *el encendido de la lámpara es una función de la variable “A”*. A partir de este análisis se pueden construir las siguientes tablas.

<sup>3</sup> Se denomina interruptor de corriente “Normal Abierto”, o sintéticamente “NA”.

<sup>4</sup> Obsérvese el comportamiento discreto (digital) del interruptor: conduce o no la corriente eléctrica. Idealmente, entre esos estados no hay otros estados intermedios. Benítez, Guillermo. Op. Cit.

Interruptor "A" presionado	Lámpara Encendida	A	E	A	E	A	E
No	No	No	No	F	F	0	0
Si	Si	Si	Si	V	V	1	1

**Tabla(s) 2.** Diferentes modos de representación del comportamiento del circuito de la Fig. 3. Todas son equivalentes.

En las tablas anteriores, se han realizado las convenciones siguientes:

- **No** presionar el interruptor "A" se representa con "F" (falso) o con 0 (cero); y **Sí** presionar el interruptor "A" se simboliza con "V" (verdadero) o con 1 (uno).
- La lámpara **No** encendida se indica con "F" (falso) o con 0 (cero); y la lámpara **Sí** encendida se representa con "V" (verdadero) o con 1 (uno).

Esta convención se denomina *lógica positiva*. Una relación opuesta (por ejemplo, representar la lámpara apagada con 1 y la lámpara encendida con 0), significará una *lógica negativa*<sup>5</sup>.

### 3. Lógica proposicional

Las ciencias experimentales se basan en la observación de la naturaleza y emplean un razonamiento de tipo inductivo para poder formular teorías generales: de las observaciones particulares se extraen conclusiones generales que permiten predecir resultados de futuros experimentos. La matemática sigue un modelo deductivo de razonamiento. A partir de una colección de verdades (axiomas) se obtienen, mediante reglas correctas de deducción, más hechos verdaderos (teoremas). El objeto de la *lógica* es estudiar las condiciones generales de validez de estas deducciones o demostraciones.

La lógica proposicional es la más antigua y simple de las formas de lógica. En la actualidad tiene importancia por su aplicación en los circuitos lógicos utilizados en electrónica y en informática. La lógica y la programación también guardan una relación mutua, porque esta última se basa en el estudio matemático y lógico de los *lenguajes*.

A partir de una representación sencilla del lenguaje, la lógica proposicional permite interpretar y manipular sentencias sobre el mundo que nos rodea. Es posible razonar a través de un mecanismo que primero evalúa proposiciones simples y luego proposiciones compuestas, formuladas mediante el uso de conectivos lógicos o proposicionales; por ejemplo "y" ( $\wedge$ ), "o" ( $\vee$ ). Este mecanismo determina la veracidad de una proposición compuesta analizando los valores de verdad asignados a las proposiciones simples que la conforman.



*El estudio de la lógica se inicia con Euclides y Aristóteles (s. III a.C.), quienes la consideraban como el fundamento necesario para todo conocimiento.*

<sup>5</sup> Como se ha mencionado en párrafos anteriores, en electrónica digital, a un determinado nivel de tensión se lo llama estado alto (high) o "1" lógico; y a otro, estado bajo (low) o "0" lógico. Supóngase que las señales eléctricas con que trabaja un sistema digital son 0 V y 5 V. En lógica positiva, 5 V será el estado alto o "1" lógico, mientras que 0 V será el estado bajo o "0" lógico. En lógica negativa ocurre todo lo contrario, es decir, se representa al estado "1" con los niveles más bajos de tensión y al "0" con los niveles más altos. En nuestro Curso trabajaremos con lógica positiva.

***Toda oración declarativa susceptible de ser verdadera o falsa, es una proposición.***

¿Cuáles de los siguientes ejemplos pueden ser proposiciones?

- 1.– Lisboa es capital de Portugal.
- 2.– “6” es un número impar.
- 3.– Pablo aprobó Matemática.
- 4.– ¿Quién viene?
- 5.– Abra las ventanas.

Los tres primeros ejemplos son oraciones declarativas, de las cuales tiene sentido decir que son verdaderas o falsas; mientras que los casos (4) y (5) presentan una pregunta y una orden. De estas últimas, no puede decirse si son verdaderas o falsas.

Según sea una proposición verdadera o falsa, se dice que *su valor de verdad* es verdadero o falso. En el ejemplo (1) el valor de verdad de la proposición dada es verdadero (V), mientras que en el ejemplo (2) su valor de verdad es falso (F). Respecto del ejemplo (3), no se dispone de información suficiente para determinar su valor de verdad; pero sólo existen dos posibilidades: es verdadero o es falso. Las proposiciones se designan con letras minúsculas  $p, q, r, s$ .

*Ejemplos:*

- $p$ : Todo triángulo equilátero es también isósceles.  
 $q$ : El oxígeno es sólido a temperatura ambiente.

Las proposiciones que tienen un predicado único, como las anteriores, se denominan *simples*. Al combinar dos o más proposiciones simples, se obtienen nuevas proposiciones llamadas *compuestas*. Esta combinación se realiza por medio de operadores llamados *conectivos lógicos*. Una propiedad fundamental es que la verdad o falsedad de una proposición compuesta dependerá de la verdad o falsedad de las proposiciones simples que la componen y de la forma en que éstas se combinan.

*Ejemplos:*

- $p$ : 3 es un número primo **y** 2 es un número par.  
 $q$ : Pablo se comunica por teléfono **o** por mail.  
 $r$ : **Si** Natalia es tucumana, **entonces** Natalia es argentina.  
 $s$ : **Si** el agua está en estado sólido (hielo), **entonces** su temperatura es menor o igual a 0° Celsius<sup>6</sup>.

En la Tabla 3 se muestran los conectivos lógicos, las operaciones asociadas y sus significados. En nuestra asignatura estudiaremos exclusivamente las cuatro primeras.

---

<sup>6</sup> La determinación del valor de verdad de las proposiciones es a veces una cuestión extra lógica, que pertenece a otro campo del conocimiento. En el caso de la proposición (r), seguramente un experto indicaría que es válida sólo si la presión a la que se encuentra el agua es la normal.

Conectivo	Operación lógica	Notación	Significado
$\sim$	Negación	$\sim p$	<b>no</b> $p$ , o no es cierto que $p$
$\wedge$	Conjunción	$p \wedge q$	$p$ <b>y</b> $q$
$\vee$	Disyunción	$p \vee q$	$p$ <b>o</b> $q$ (en sentido incluyente)
$\underline{\vee}$	Disyunción exclusiva	$p \underline{\vee} q$	$p$ <b>ó</b> $q$ (en sentido excluyente)
$\Rightarrow$	Implicación <sup>7</sup>	$p \Rightarrow q$	<b>si</b> $p$ <b>entonces</b> $q$ , o $p$ <b>implica</b> $q$
$\Leftrightarrow$	Doble implicación <sup>8</sup>	$p \Leftrightarrow q$	$p$ <b>si y sólo si</b> $q$

Tabla 3. Conectivos lógicos.

### 3.1. Operaciones Proposicionales

#### • Negación

Dada una proposición  $p$ , se denomina *negación* de  $p$  a otra proposición denotada por  $\sim p$  (se lee "no  $p$ ") a la que se le asigna el valor de verdad opuesto al de  $p$ . Esta ley que define la negación lógica, o simplemente negación, puede presentarse en forma resumida utilizando una tabla de doble entrada denominada **tabla de verdad**. La negación es una operación unitaria, pues a partir de una proposición se obtiene otra (que es su opuesta). La tabla de verdad de la negación es:

$p$	$\sim p$
$F$	$V$
$V$	$F$

*Ejemplos:*

$p$ : Las gaviotas son mamíferos.

$\sim p$ : Las gaviotas no son mamíferos.

#### • Conjunción

Es la proposición compuesta que se obtiene uniendo dos proposiciones simples mediante la conjunción "y". Simbólicamente, la conjunción se indica  $p \wedge q$ . Su valor de verdad está dado por la tabla siguiente.

$p$	$q$	$p \wedge q$
$F$	$F$	$F$
$F$	$V$	$F$
$V$	$F$	$F$
$V$	$V$	$V$

De la tabla de verdad resulta que la conjunción  $p \wedge q$  es verdadera solamente en el caso de que ambas  $p$  y  $q$  sean verdaderas simultáneamente. Dicho de otro modo, la conjunción es falsa cuando al menos una de las proposiciones que la forman es falsa<sup>9</sup>.

<sup>7</sup> En algunos textos se denomina *Condicional*.

<sup>8</sup> También llamada *Bicondicional*.



*Ejemplo:*

$p$ : Para aprobar la materia, hay que tener un 75% de asistencia.

$q$ : Para aprobar la materia, hay que aprobar los parciales.

$p \wedge q$ : Para aprobar la materia, hay que tener un 75% de asistencia y aprobar los parciales.

### • Disyunción

Es la proposición compuesta que se obtiene uniendo dos proposiciones simples mediante el conectivo “o” en sentido *incluyente*, es decir; uno u otro o *ambos*. Simbólicamente, la disyunción se indica  $p \vee q$ . Su valor de verdad está dado por la tabla siguiente.

$p$	$q$	$p \vee q$
$F$	$F$	$F$
$F$	$V$	$V$
$V$	$F$	$V$
$V$	$V$	$V$

De la tabla de verdad resulta que la disyunción  $p \vee q$  es falsa solamente en el caso de que ambas  $p$  y  $q$  sean falsas simultáneamente<sup>10</sup>. En otras palabras, con que al menos una de las proposiciones sea verdadera, la disyunción será verdadera.

*Ejemplo:*

$p$ : En Excel puede modificarse el alto de las celdas.

$q$ : En Excel puede modificarse el ancho de las celdas.

$p \vee q$ : En Excel pueden modificarse el alto o el ancho de las celdas<sup>11</sup>.

### • Disyunción exclusiva<sup>12</sup>

Es la proposición compuesta que se obtiene uniendo dos proposiciones simples mediante el conectivo “ó” en sentido *excluyente*, es decir; uno u otro, pero *no ambos*. Simbólicamente, la disyunción exclusiva se indica  $p \underline{\vee} q$ . Su valor de verdad está dado por la tabla siguiente.

$p$	$q$	$p \underline{\vee} q$
$F$	$F$	$F$
$F$	$V$	$V$
$V$	$F$	$V$
$V$	$V$	$F$

De la tabla de verdad resulta que la disyunción exclusiva  $p \underline{\vee} q$  es verdadera solamente cuando  $p$  y  $q$  tienen *diferente valor de verdad*.

*Ejemplos:*

<sup>9</sup> Como ejercitación, con la lectura de los ejemplos, determinar a cuál de las combinaciones de la tabla de verdad correspondería cada caso.

<sup>10</sup> También suele decirse que la disyunción es verdadera cuando *al menos una* de las proposiciones es verdadera.

<sup>11</sup> O ambas dimensiones.

<sup>12</sup> También llamada Disyunción Excluyente.

$p$ : El número 2 es par.

$q$ : El número 2 es impar.

$p \vee q$ : El número 2 es par ó impar<sup>13</sup>.

$r$ : A las 21 hs. veré El Hombre Araña, en la Sala 2.

$s$ : A las 21 hs. veré Leyendas de Pasión, en la Sala 4.

$r \vee s$ : A las 21 hs. veré El Hombre Araña ó Leyendas de Pasión<sup>14</sup>.

Según algunos autores, puede pensarse que una disyunción exclusiva es verdadera cuando efectivamente *se puede elegir entre dos alternativas diferentes*.

### • Implicación

Se llama implicación o condicional de  $p$  con la proposición  $q$ , a la proposición de la forma  $p \Rightarrow q$  obtenida anteponiendo a la primera la palabra “si” y uniendo ambas mediante la palabra “entonces”. Las proposiciones  $p$  y  $q$  se llaman, respectivamente, *antecedente* y *consecuente*<sup>15</sup>. El valor de verdad de la implicación se muestra en la tabla que sigue<sup>16</sup>.

$p$	$q$	$p \Rightarrow q$
$F$	$F$	$V$
$F$	$V$	$V$
$V$	$F$	$F$
$V$	$V$	$V$

De la tabla resulta que la implicación  $p \Rightarrow q$  es falsa sólo si el antecedente es verdadero y el consecuente falso.

En otras disciplinas, la proposición condicional puede emplearse como elemento de decisión para el control de un proceso. En esos contextos, la forma “*si p entonces q*” incluye una condición  $p$  y una orden de control  $q$ . El condicional dice: “*cuando se cumpla la condición p, debe ejecutarse la acción q*”. Por ejemplo:

- ▶ Si hay luz roja, deténgase.
- ▶ Si la temperatura del fluido supera los 100°C, entonces abra la válvula de salida de vapor.
- ▶ Si desea guardar los cambios, presione Aceptar.

### • Doble Implicación

Doble implicación de las proposiciones  $p$  y  $q$  es la proposición  $p \Leftrightarrow q$ , cuya lectura es “ $p$  si y sólo si  $q$ ”. Presenta la siguiente tabla de verdad.

<sup>13</sup> En este caso, “ó” tiene el sentido excluyente; puesto que 2, o bien es par ó bien es impar, pero no ambos.

<sup>14</sup> No es posible ver las dos películas al mismo tiempo.

<sup>15</sup> Nótese que en las operaciones lógicas anteriores, el orden de las proposiciones podía ser indistinto. En el caso de la implicación no sucede así: el orden tiene importancia.

<sup>16</sup> En el caso de la implicación existen expresiones sinónimas, como pueden ser: *si p, q; p implica q; q si p; q cuando p*, etc.

$p$	$q$	$p \Leftrightarrow q$
$F$	$F$	$V$
$F$	$V$	$F$
$V$	$F$	$F$
$V$	$V$	$V$

De la tabla resulta que la doble implicación  $p \Leftrightarrow q$  sólo es verdadera si *ambas proposiciones tienen el mismo valor de verdad*.

*Ejemplo:*

$p$ : T es un triángulo equilátero.

$q$ : T tiene tres lados iguales.

$p \Leftrightarrow q$ : T es equilátero si y sólo si T tiene tres lados iguales.

• **No hemos de profundizar, en nuestra asignatura, el estudio de la implicación y de la doble implicación. Se han explicado brevemente, no obstante, a título informativo y en razón de sus aplicaciones en la mayoría de los campos del conocimiento.**

### 3.2. Proposiciones Equivalentes

Son *proposiciones equivalentes* las que presentan **la misma tabla de verdad**. Por ejemplo, la doble implicación puede definirse como la conjunción de una implicación y su recíproca. De este modo, la tabla de verdad de  $p \Leftrightarrow q$  puede obtenerse mediante la tabla de  $(p \Rightarrow q) \wedge (q \Rightarrow p)$ , como se muestra en el ejemplo que sigue:

$p$	$q$	$p \Rightarrow q$	$q \Rightarrow p$	$(p \Rightarrow q) \wedge (q \Rightarrow p)$
$F$	$F$	$V$	$V$	$V$
$F$	$V$	$V$	$F$	$F$
$V$	$F$	$F$	$V$	$F$
$V$	$V$	$V$	$V$	$V$

**Tabla 4.** Ejemplo de equivalencia de proposiciones.

La noción de operaciones lógicas equivalentes tiene particular importancia en el estudio y las aplicaciones del Álgebra de Boole.

## 4. Álgebra conmutacional y compuertas lógicas básicas

La compuerta lógica es el bloque de construcción básico de los sistemas digitales. Las compuertas lógicas operan con números binarios; por ello suelen llamarse compuertas lógicas binarias. Todas las tensiones utilizadas en las compuertas lógicas son ALTA ó BAJA. A los fines de nuestro estudio, una tensión ALTA significa un “1” lógico o binario; mientras que una tensión BAJA significa un “0” lógico o binario.

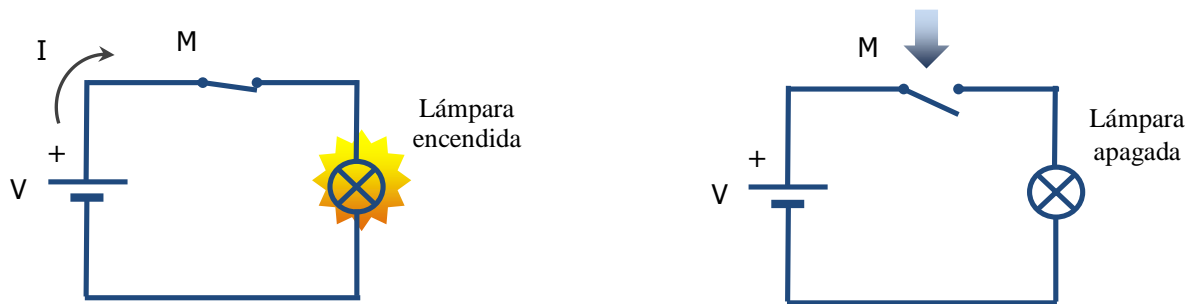
Las compuertas lógicas son circuitos electrónicos y están construidas generalmente a partir de transistores. Estos circuitos responden solamente a tensiones ALTAS (llamadas “1”) o BAJAS (llamadas “0”).

Todos los sistemas digitales se construyen utilizando tres compuertas lógicas básicas: las compuertas NOT, AND y OR.

El empleo de las propiedades de los interruptores eléctricos a la lógica de proceso es el concepto básico que subyace en todos los sistemas electrónicos modernos en los equipos digitales.

#### 4.1. Compuerta NOT

Si se utiliza un interruptor de mecanismo inverso al de la Fig. 3, al ejercer presión sobre él, la lámpara se apaga.

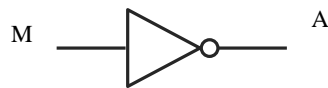


**Figura 4.** Posiciones posibles de un interruptor (en este caso, normalmente cerrado)

Este interruptor “M” presenta un funcionamiento opuesto al interruptor “A” de la Fig. 3.

Esto puede escribirse:  $M = \bar{A}$ .

La compuerta lógica que se utiliza para invertir una variable se denomina NOT, y su representación es la que sigue:



$M = \bar{A}$	A
0	1
1	0

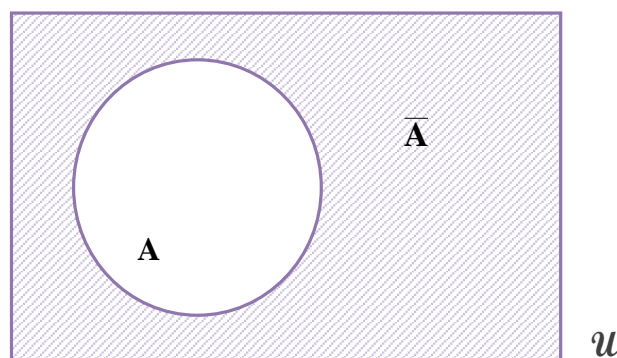
**Figura 5.** Compuerta NOT (inversora) y su tabla de verdad.

En lógica proposicional, la compuerta NOT corresponde a la *negación* (“no”).

En teoría de conjuntos, el elemento equivalente es el *complemento*.

El complemento de un conjunto A es el conjunto formado por los elementos de U que no pertenecen a A. En símbolos:

$$\bar{A} = \{x \in U / x \notin A\} \quad x \in \bar{A} \Leftrightarrow x \notin A$$



**Figura 6.** El conjunto A y su complemento,  $\bar{A}$ .

En particular, se tiene que el complemento del conjunto vacío es el conjunto universal; y el complemento del conjunto universal es el conjunto vacío.

En el Álgebra de Boole (se estudiará más adelante) se asocia con el elemento opuesto.

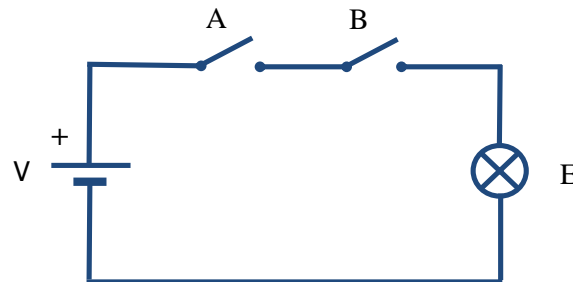
Los inversores, por motivos de sencillez, también pueden indicarse por medio de un pequeño círculo entre la variable y el resto del esquema. Pueden encontrarse a la entrada o a la salida de otras compuertas lógicas. Por ejemplo, asociándolos a las compuertas AND y OR que se verán a continuación.



**Figura 7.** El pequeño círculo equivale a una negación. El circuito (b) es equivalente al (a). El circuito (d) es equivalente al (c).

## 4.2. Compuerta AND

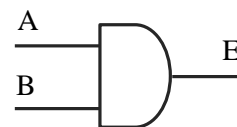
Sea el circuito siguiente:



**Figura 8.** Interruptores en serie.

En el circuito de la Fig. 8, para que la lámpara encienda, los interruptores A y B deben estar presionados al mismo tiempo. Es decir, se deben presionar A y B. Entonces, se podrá realizar la siguiente tabla de verdad:

A	B	E
0	0	0
0	1	0
1	0	0
1	1	1



**Figura 9.** Tabla de verdad de la compuerta AND y su representación.

La tabla de verdad se interpreta diciendo: para que la salida “E” sea 1, “A y B” deben ser 1.

Existen muchas situaciones en las cuales, para que un suceso determinado ocurra, deben cumplirse simultáneamente ciertas condiciones. Por ejemplo, para operar con un cajero automático, hay que introducir la tarjeta magnética y digitar la clave. Para accionar algunas máquinas industriales que pueden ser peligrosas; el operario debe presionar *al mismo tiempo dos botones*; esto asegura que sus manos no quedarán libres – es decir, “metidas en el lugar equivocado” – en el instante del accionamiento.



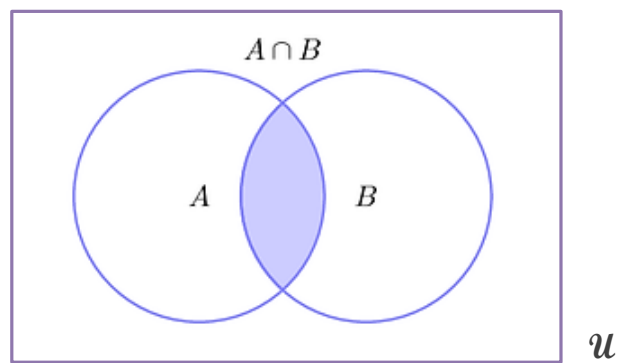
**Figura 10.** Accionamiento de una guillotina para cortar resmas de papel.

En lógica proposicional, la compuerta AND corresponde a la *conjunción* (“y”).

En teoría de conjuntos, la operación equivalente es la *intersección*.

La intersección de dos conjuntos A y B es el conjunto formado por los elementos que pertenecen a A y a B. La intersección de conjuntos se denota con el símbolo  $\cap$ . De manera que, simbólicamente:

$$A \cap B = \{x \in U / x \in A \wedge x \in B\}$$



**Figura 11.** Intersección de conjuntos (área sombreada).

En el Álgebra de Boole, la tabla de verdad coincide con la operación denominada “*producto lógico*”. La conjunción “y” se reemplaza por el producto lógico, representado con un punto.

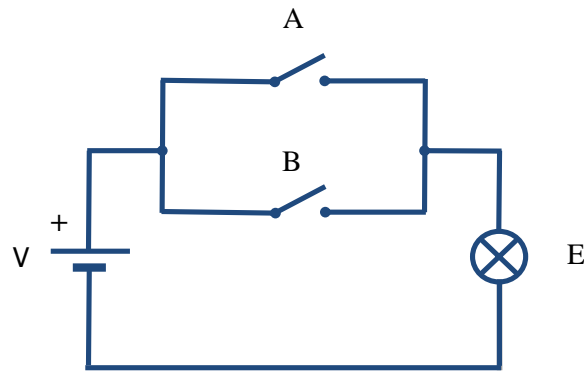
De esta forma, puede escribirse su tabla de verdad de la siguiente manera:

<i>A</i>	<i>B</i>	<i>E = A . B</i>
0	0	0
0	1	0
1	0	0
1	1	1

**Tabla 5.** Tabla de verdad del producto lógico ( . ) del Álgebra de Boole.

### 4.3. Compuerta OR

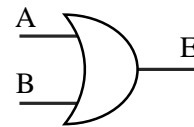
Sea el circuito siguiente:



**Figura 12.** Interruptores en paralelo.

En el circuito de la Fig. 12, a diferencia del caso anterior, para que la lámpara encienda, debe estar presionado el interruptor A **o** debe estarlo el interruptor B; o ambos. Siempre que *al menos uno* de los interruptores esté presionado, la lámpara se enciende. La tabla de verdad será, en este caso:

A	B	E
0	0	0
0	1	1
1	0	1
1	1	1



**Figura 13.** Tabla de verdad de la compuerta OR y su representación.

Un ejemplo podría ser una alarma que debe ser activada desde cualquiera de dos lugares diferentes. Los interruptores, en este caso, pueden estar asociados a sensores de movimiento. Entonces, presionando (activando) uno cualquiera o ambos interruptores, debe accionarse la alarma. Otro ejemplo es la activación de la luz interior del automóvil, que se enciende cuando se abre una puerta, o la otra, o ambas.



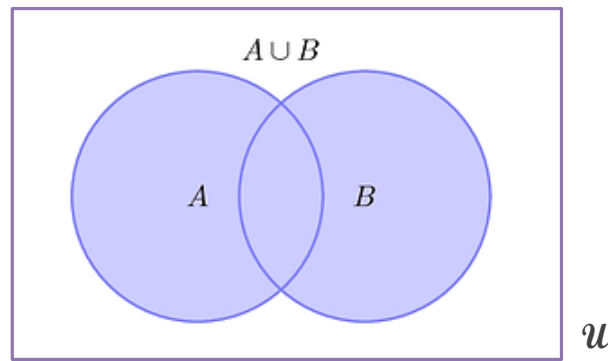
**Figura 14.** Accionamiento de la luz interior de un automóvil por la apertura de puertas.

En lógica proposicional, la compuerta OR corresponde a la disyunción (“**o**”).

En teoría de conjuntos, la operación equivalente es la *unión*.

La unión de dos conjuntos A y B es el conjunto formado por los elementos que pertenecen a A **o** pertenecen a B (o ambas cosas a la vez). La unión de conjuntos se denota con el símbolo  $\cup$ . De manera que, simbólicamente:

$$A \cup B = \{x \in U / x \in A \vee x \in B\}$$



**Figura 15.** Unión de conjuntos (área sombreada).

En otras palabras, la unión de dos conjuntos  $A \cup B$ , es el conjunto que contiene todos los elementos de ambos conjuntos. En el Álgebra de Boole, la tabla de verdad coincide con la operación denominada “*suma lógica*”. La disyunción “*o*” se reemplaza por la suma lógica, representada con un signo “+”. De esta forma, puede escribirse su tabla de verdad de la siguiente manera:

$A$	$B$	$E = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

**Tabla 6.** Tabla de verdad de la suma lógica (+) del Álgebra de Boole.

En nuestro idioma, la disyunción “*o*” presenta ambigüedad respecto de su significado lógico. Si decimos: “Esto es blanco o negro”, es uno u otro; pero no simultáneamente, se excluye la condición de simultaneidad (disyunción exclusiva o excluyente).

En cambio, en la expresión: “Espero tu respuesta, por mail o por teléfono”, implícitamente se acepta la situación de simultaneidad. Cualquiera de los medios (mail o teléfono) o los dos al mismo tiempo, serán válidos. En este caso, la “*o*” es inclusiva. Este último es el sentido en que lo empleamos en este caso y el que corresponde a la unión de conjuntos.

La disyunción exclusiva se estudiará más adelante.



**Figura 16.** Disyunciones exclusiva e inclusiva.



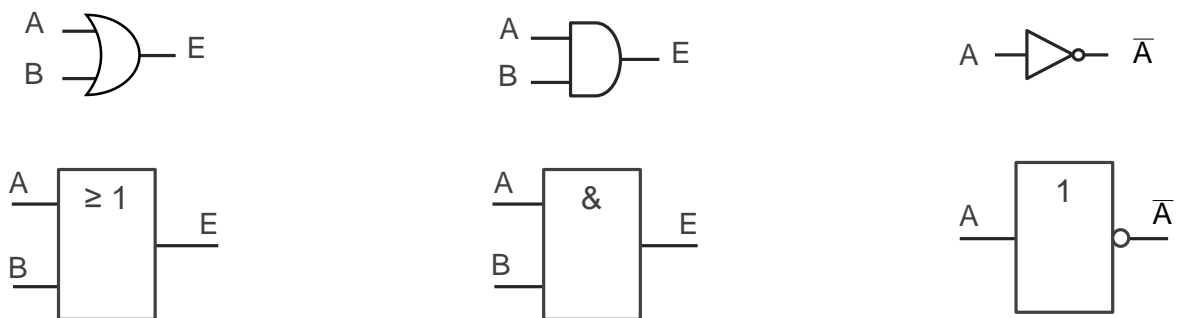
## 5. Comparación entre enfoques lógicos

De acuerdo con lo expuesto, podemos elaborar la siguiente tabla con la relación entre el Álgebra de Boole, la teoría de conjuntos, el cálculo proposicional, conmutacional (circuitos eléctricos) y los circuitos con compuertas lógicas.

<b>Álgebra de Boole (binaria)</b>	<b>Teoría de Conjuntos</b>	<b>Cálculo Proposicional</b>	<b>Conmutación (positiva)</b>	<b>Circuitos Lógicos</b>
Trabaja con Elementos	Trabaja con Conjuntos	Trabaja con Proposiciones	Trabaja con Acción, señal	Trabaja con Variables
Suma lógica (+)	Unión ( $\cup$ )	Disyunción ( $\vee$ ), "o"	Circuito paralelo	OR 
Producto lógico ( $\cdot$ )	Intersección ( $\cap$ )	Conjunción ( $\wedge$ ), "y"	Circuito serie	AND 
Elemento opuesto	Complemento	Negación (No)	Inversor	NOT 
Neutro de la suma	Conjunto Vacío ( $\emptyset$ )	Falsedad (F)	No a la acción	0
Neutro del producto	Conj. Universal ( $\mathcal{U}$ )	Certeza (V)	Sí a la acción	1

**Tabla 7.** Comparación entre enfoques o sistemas de estudio de la lógica.

En cuanto a la simbología, conviene aclarar que la representación tradicional que se utiliza aquí, es la más empleada en la bibliografía. No obstante, manuales de datos técnicos y otras publicaciones más avanzadas, utilizan la simbología normalizada (ISO) que se menciona a continuación. Esta última facilita la representación de bloques funcionales de circuitos integrados digitales y el diseño asistido por computadora (CAD).



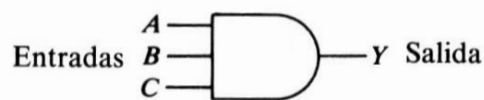
**Figura 17.** En la representación normalizada de compuertas lógicas se dibujan rectángulos a los que se coloca un símbolo que indica de qué compuerta se trata.

## 6. Aplicaciones de compuertas lógicas

### 6.1. Compuertas con más de dos variables de entrada (ejemplos)

$$A \cdot B \cdot C = Y$$

(a) Expresión booleana de tres variables



(b) Símbolo de una puerta AND de tres entradas

Entradas			Salida
$C$	$B$	$A$	$Y$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(c) Tabla de verdad con tres variables

$$A + B + C = Y$$

(a) Expresión booleana de tres variables

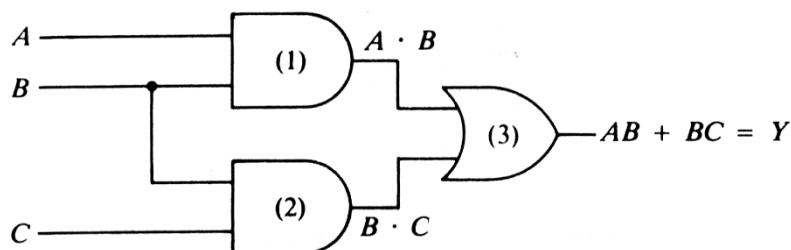


(b) Símbolo de una puerta OR de tres entradas

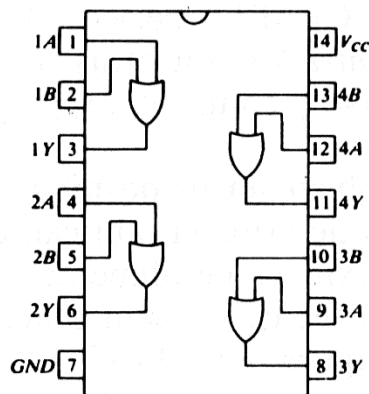
Entradas			Salida
$C$	$B$	$A$	$Y$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(c) Tabla de verdad con tres variables

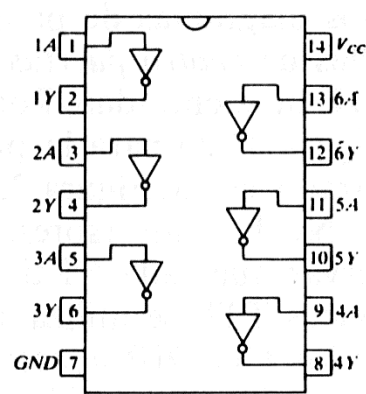
### 6.2. Ejemplo de asociación de compuertas lógicas



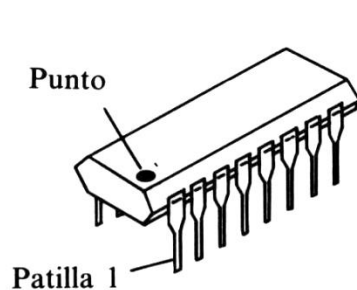
### 6.3. Ejemplos de utilización de compuertas lógicas en circuitos integrados



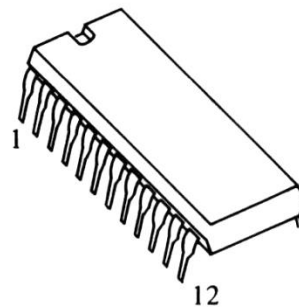
(a) Diagrama de patillas de un CI 7432



(b) Diagrama de patillas de un CI 7404



(a) Circuito integrado DIP de 16 patillas



(b) Circuito integrado DIP de 24 patillas

## 7. Álgebra de Boole

### 7.1. Postulados del Álgebra de Boole o Postulados de Huntington

Empleando los postulados del Álgebra de Boole, se demostrarán más adelante algunas de las leyes o teoremas necesarios para el desarrollo de circuitos lógicos. La habilidad en el empleo del álgebra permite avanzar en la comprensión de las aplicaciones y más tarde redundará en una reducción de costos de los sistemas digitales que, a partir de estas leyes, pueden ser simplificados.

#### • Definición del Álgebra de Boole<sup>17</sup>

Se definen los elementos del álgebra, sus relaciones y las dos operaciones admitidas, suma lógica y producto lógico. La definición se puede enunciar diciendo: Existe el conjunto "C" y existen los elementos (a, b,...) pertenecientes a "C", tal que el elemento "a" guarda relación de equivalencia con el elemento "b" y existe la operación (a + b) cuyo resultado pertenece a "C" y existe la operación (a . b) cuyo resultado pertenece a "C".

<sup>17</sup> Significados de la notación:  $\exists$  = existe,  $\wedge$  = y,  $\in$  = pertenece,  $/$  = tal que,  $\forall$  = para todo,  $\Rightarrow$  = implica.

$$1) \exists C \wedge \exists (a, b, \dots) \in C / a R b \wedge \exists (a + b) \in C \wedge \exists (a \cdot b) \in C$$

Recordemos que  $a$  y  $b$  están en relación de equivalencia “ $a R b$ ”, si se cumplen los principios de identidad “ $a R a$ ”, simetría “ $a R b \Rightarrow b R a$ ” y transitividad “ $a R b \wedge b R c \Rightarrow a R c$ ”.

• *Conmutatividad*

$$2 a) \forall (a, b) \in C / a + b = b + a$$

*Conmutativa de la suma*

$$2 b) \forall (a, b) \in C / a \cdot b = b \cdot a$$

*Conmutativa del producto*

• *Asociatividad*

$$3 a) \forall (a, b, c) \in C / (a + b) + c = a + (b + c)$$

*Asociativa de la suma*

$$3 b) \forall (a, b, c) \in C / (a \cdot b) \cdot c = a \cdot (b \cdot c)$$

*Asociativa del producto*

• *Distributividad*

$$4 a) \forall (a, b, c) \in C / (a + b) \cdot c = a \cdot c + b \cdot c$$

*Distributiva de la suma*

$$4 b) \forall (a, b, c) \in C / (a \cdot b) + c = (a + c) \cdot (b + c)$$

*Distributiva del producto*

• *Existencia del elemento neutro*

$$5 a) \forall a \in C \exists N_1 / a + N_1 = a$$

*Neutro de la suma*

$$5 b) \forall a \in C \exists N_2 / a \cdot N_2 = a$$

*Neutro del producto*

• *Existencia del elemento opuesto*

$$6 a) \forall a \in C \exists \bar{a} / a + \bar{a} = N_2$$

*Opuesto de la suma*

$$6 b) \forall a \in C \exists \bar{a} / a \cdot \bar{a} = N_1$$

*Opuesto del producto*

## 7.2. Teoremas del Álgebra de Boole<sup>18</sup>

• *Teorema I) Dualidad<sup>19</sup>:*

Cada identidad deducida a partir de los postulados del álgebra de Boole, permanece válida si las operaciones “+” y “.” y los elementos “0” y “1” se intercambian entre sí.

Se deduce de la simetría de los postulados con respecto a las dos operaciones: suma lógica y producto lógico; y a los dos elementos neutros:  $N_1$  (el “0” lógico, neutro de la suma) y  $N_2$  (el “1” lógico, neutro del producto). Por otra parte, se pueden escribir nuevamente los postulados realizando el cambio propuesto por el Teorema de Dualidad y comprobar que vuelve a obtener los postulados originales, cambiados de orden.

• *Teorema II)<sup>20</sup>*  $a + 1 = 1$

*Demostración:*

<sup>18</sup> Para facilitar su ordenamiento y estudio, identificamos los Postulados con números arábigos y los Teoremas con números romanos.

<sup>19</sup> El Teorema de Dualidad no tiene una fórmula asociada. Se trata del concepto de su significado y de su aplicación a cualquier expresión lógica.

<sup>20</sup> El Teorema II no tiene un nombre específico.

$$1 = a + \bar{a} = a + \bar{a} \cdot 1 = (a + \bar{a}) \cdot (a + 1) = 1 \cdot (a + 1) = a + 1$$

Hemos utilizando los postulados: el recíproco de 6a, 5b, 4b, 6a y 5b, respectivamente. Empleando el concepto de dualidad, se podrá comprobar el teorema: “ $a \cdot 0 = 0$ ”, aplicando los postulados: recíproco de 6b, 5a, 4a, 6b y 5a, respectivamente.

• *Teorema III) Unicidad:*  $a + a = a$

$$a = a + 0 = a + a \cdot \bar{a} = (a + a) \cdot (a + \bar{a}) = (a + a) \cdot 1 = a + a$$

Hemos utilizando los postulados: elemento neutro de la suma, producto con el elemento opuesto, distributividad, suma del elemento opuesto y elemento neutro del producto, respectivamente. También aquí podrá utilizarse el concepto de dualidad y demostrar: “ $a \cdot a = a$ ”.

• *Teorema IV) Absorción:*  $a + a \cdot b = a$

$$a = 1 \cdot a = (1 + b) \cdot a = (a \cdot 1 + b \cdot a) = a + a \cdot b$$

Hemos utilizando los postulados: elemento neutro del producto, Teorema “II” aplicado a la variable b, distributividad y producto con el elemento opuesto, respectivamente.

• *Teorema V) Doble negación:*  $\bar{\bar{a}} = a$

a) Sea:  $\bar{a} = x$  y por lo tanto,  $\bar{x} = a$

b) Según el postulado del elemento opuesto:  $x + \bar{x} = 1$  y  $x \cdot \bar{x} = 0$

c) Sustituyendo las variables:  $\bar{a} + a = 1$  y  $\bar{a} \cdot a = 0$

d) Aplicando el postulado de conmutatividad:  $\bar{a} + a = 1$  y  $\bar{a} \cdot a = 0$

e) Que por supuesto, satisfacen el postulado de existencia del elemento opuesto:

$$6a) \forall a \in C \exists \bar{a} / a + \bar{a} = N_2$$

*Opuesto de la suma*

$$6b) \forall a \in C \exists \bar{a} / a \cdot \bar{a} = N_1$$

*Opuesto del producto*

Donde:  $N_1 = 0$  y  $N_2 = 1$  y, por lo tanto, esa parte del postulado podría escribirse:

$$a + \bar{a} = 1 \text{ y } a \cdot \bar{a} = 0$$

Ya que por el postulado 1, todo elemento del Álgebra de Boole es equivalente a sí mismo, y por lo tanto  $\bar{\bar{a}} = a$ , considerando la deducción del punto d) anterior:

$$\bar{a} + a = 1 \text{ y } \bar{a} \cdot a = 0 \text{ y el postulado 6):}$$

$$a + \bar{a} = 1 \text{ y } a \cdot \bar{a} = 0 \text{ se deduce que } \bar{\bar{a}} = a \text{ como queríamos demostrar.}$$

• *Teorema VI) Teorema de De Morgan:*  $\overline{a + b} = \bar{a} \cdot \bar{b}$

a) Sea:  $a + b = u \Rightarrow \overline{a + b} = \bar{u}$

Si se cumple la igualdad presentada en el Teorema de De Morgan, entonces se cumplirá:

$$\bar{a} \cdot \bar{b} = \bar{u}$$

Lo demostraremos aplicando el postulado del elemento opuesto.

b) Consideremos dicho postulado, aplicado a una variable auxiliar “u”:

$$u + \bar{u} = 1 \quad \text{y} \quad u \cdot \bar{u} = 0$$

c) Sustituyendo las variables:

$$(a + b) + (\bar{a} \cdot \bar{b}) = 1 \quad \text{y} \quad (a + b) \cdot (\bar{a} \cdot \bar{b}) = 0$$

d) Aplicando el postulado de distributividad:

$$(a + b + \bar{a}) \cdot (a + b + \bar{b}) = 1 \quad \text{y} \quad (a \cdot \bar{a} \cdot \bar{b}) + (b \cdot \bar{a} \cdot \bar{b}) = 0$$

e) Ahora aplicamos el postulado de conmutatividad:

$$(a + \bar{a} + b) \cdot (a + b + \bar{b}) = 1 \quad \text{y} \quad (a \cdot \bar{a} \cdot \bar{b}) + (b \cdot \bar{b} \cdot \bar{a}) = 0$$

f) A continuación utilizamos el postulado del elemento opuesto:

$$(1 + b) \cdot (a + 1) = 1 \quad \text{y} \quad (0 \cdot \bar{b}) + (0 \cdot \bar{a}) = 0$$

g) Aplicando el Teorema II) aplicado a las variables “a” o “b”, según corresponda:

$$(1 + b) \cdot (a + 1) = 1 \Rightarrow 1 \cdot 1 = 1 \quad \text{y} \quad (0 \cdot \bar{b}) + (0 \cdot \bar{a}) = 0 \Rightarrow 0 + 0 = 0$$

que es lo que queríamos demostrar.

### 7.3. Funciones de un álgebra de Boole

Una función del álgebra de Boole es una variable binaria (es decir, que puede adoptar uno de dos valores: “0”, ó “1”), cuyo valor depende de una cierta combinación de valores relacionados por las operaciones suma lógica y producto lógico, y donde las variables que intervienen pueden presentarse en forma directa o por medio de su opuesto.

Una forma de nombrarla es:  $f(\dots, c, b, a)$ .

Por ejemplo:

$$f_{(b,a)} = a + \bar{b} \cdot \bar{a}$$

En este caso, la función  $f_{(b,a)}$  vale 1 cuando  $a=1$ , o cuando  $b=0$  y  $a=0$ . También puede decirse que  $f_{(b,a)}$  vale cero sólo si  $b=1$  y  $a=0$ .

Pueden representarse estas posibilidades en una tabla de verdad:

b	a	$f_{(b,a)} = a + \bar{b} \cdot \bar{a}$
0	0	$f_{(0,0)} = 1$
0	1	$f_{(0,1)} = 1$
1	0	$f_{(1,0)} = 0$
1	1	$f_{(1,1)} = 1$

**Tabla 8.** Ejemplo de confección de tabla de verdad para una función lógica.

O más brevemente:

b	a	f ( b, a )
0	0	1
0	1	1
1	0	0
1	1	1

**Tabla 9.** Tabla de verdad para una función lógica.

## 7.4. Términos canónicos y expresiones canónicas

Un término canónico es aquel en el cual intervienen *todas* las variables de la función con cualquier estado de complemento (directas o negadas).

Por ejemplo, en la función:

$$f(c, b, a) = a + \bar{b} \cdot \bar{a} + c \cdot b \cdot \bar{a} + c \cdot b + c \cdot \bar{b} \cdot a + \bar{b} \cdot a + c + c \cdot b \cdot a$$

son términos canónicos el tercero ( $c \cdot b \cdot \bar{a}$ ), el quinto ( $c \cdot \bar{b} \cdot a$ ) y el último ( $c \cdot b \cdot a$ ).

### 7.4.1. Minitérminos, Suma Canónica, Suma de productos o Primera Forma Normal<sup>21</sup>

Para una función booleana de  $n$  variables  $x_1, \dots, x_n$ , un producto lógico en el que cada una de las  $n$  variables aparece una sola vez (directa o negada) es llamado *minitérmino*. Es decir, un *minitérmino* es una expresión lógica de  $n$  variables consistente únicamente en el operador producto lógico (AND) y el operador complemento o negación (NOT).

Por ejemplo, son ejemplos de minitérminos para una función del Álgebra de Boole de tres variables,  $c$ ,  $b$  y  $a$ :

$$c \cdot b \cdot \bar{a} \quad c \cdot b \cdot a \quad c \cdot \bar{b} \cdot \bar{a} \quad \bar{c} \cdot \bar{b} \cdot \bar{a}$$

En algunos libros, se asigna a cada minitérmino (escribiendo las variables que lo componen en el mismo orden), un índice basado en el valor binario del minitérmino. Un término negado, como " $\bar{a}$ ", es considerado como el número binario 0 y el término directo " $a$ " es considerado como un 1. Por ejemplo, se asociaría el número 6 con  $c \cdot b \cdot \bar{a}$  ( $110_2$ ) y nombraríamos la expresión con el nombre  $m_6$ . Entonces, siguiendo con los ejemplos,  $m_0$  de tres variables es  $\bar{c} \cdot \bar{b} \cdot \bar{a}$  ( $000_2$ ) y  $m_7$  debería ser  $c \cdot b \cdot a$  ( $111_2$ ).

#### • Función equivalente expresada como suma de productos

Se puede observar que cada minitérmino sólo devuelve "verdadero" con una sola entrada de las posibles. Por ejemplo, el minitérmino 5,  $c \cdot \bar{b} \cdot a$  es verdadero solo cuando  $c$  y  $a$  son verdaderos y  $b$  es falso. Es decir, la entrada  $c = 1$ ,  $b = 0$ ,  $a = 1$ , da como resultado 1.

Si tenemos una tabla de verdad de una función lógica, es posible escribir la función como "suma de productos". Si se suman todos los minitérminos asociados a las filas en las que la función

<sup>21</sup> Existen dos tipos de términos canónicos: producto canónico y suma canónica. Se los conoce en algunos textos como: "términos mínimos" y "términos máximos", respectivamente.

El ordenamiento de las variables, en una forma u otra, no cambia a la función en sí. Es frecuente que cada autor utilice un mismo ordenamiento en toda su obra.

lógica vale 1, tendremos una ecuación que representa el comportamiento lógico especificado en la tabla de verdad. A dicha expresión se le conoce como Primera Forma Normal o suma de minitérminos. Por ejemplo, dada la tabla de verdad:

c	b	a	$f(c, b, a)$
0	0	0	1
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	0

**Tabla 10.** Tabla de verdad (ejemplo) para una función lógica de tres variables.

Es posible anotar las posiciones, como guía, los nombres de los minitérminos y la expresión lógica de cada uno de ellos, como se muestra en la tabla siguiente:

	c	b	a	$f(c, b, a)$	Minitérmino	Expresión
0	0	0	0	1	$m_0$	$\bar{c} \cdot \bar{b} \cdot \bar{a}$
1	0	0	1	0		
2	0	1	0	1	$m_2$	$\bar{c} \cdot b \cdot \bar{a}$
3	0	1	1	0		
4	1	0	0	0		
5	1	0	1	1	$m_5$	$c \cdot \bar{b} \cdot a$
6	1	1	0	0		
7	1	1	1	0		

**Tabla 11.** Minitérminos de una función lógica (ejemplo). Teniendo en cuenta que cada minitérmino sólo vale 1 para una combinación de las entradas, podemos asociar a cada fila de una tabla de verdad un único minitérmino que valdrá 1 sólo cuando las entradas tomen el valor de esa fila.

Observamos que las filas con resultado 1 son las de posiciones 0, 2 y 5. Entonces podremos escribir  $f$  como la suma de los minitérminos  $m_0$ ,  $m_2$  y  $m_5$ .

$$f(c, b, a) = \bar{c} \cdot \bar{b} \cdot \bar{a} + \bar{c} \cdot b \cdot \bar{a} + c \cdot \bar{b} \cdot a$$

Para verificar esto, tendremos que la tabla de verdad de la función, calculándola directamente, será la misma.

A menudo es conveniente usar una notación más simple para expresar una suma de minitérminos. Teniendo en cuenta que cada minitérmino está asociado a una fila de la tabla de verdad, basta con enumerar los números de las filas de dicha tabla en las que la función lógica vale 1. Así la suma de minitérminos anterior puede expresarse como:  $\Sigma_3(0, 2, 5)$ , donde “ $\Sigma$ ” indica “suma de minitérminos” o *sumatoria*, el subíndice “3” señala que se trata de una función de tres variables, y entre paréntesis se colocan las posiciones de la tabla de verdad en las que la función vale 1.



Es importante destacar que las formas canónicas NO son formas simplificadas o mínimas de la función; todo lo contrario, son funciones expandidas, que se utilizan para poner de manifiesto su comportamiento.

#### 7.4.2. Maxitérminos, Producto Canónico, Producto de sumas o Segunda Forma Normal

Un *maxitérmino* es una expresión lógica de  $n$  variables que consiste únicamente en la suma lógica y el operador complemento o negación. Los maxitérminos son una expresión dual de los minitérminos. En vez de usar operaciones AND utilizamos operaciones OR y procedemos de forma similar. Por ejemplo, los siguientes son maxitérminos:

$$(c + b + a) \quad (\bar{c} + \bar{b} + \bar{a}) \quad (c + b + \bar{a}) \quad (c + \bar{b} + \bar{a})$$

Para indexar maxitérminos lo haremos precisamente de la forma contraria a la que seguimos con los minitérminos. Se asigna a cada maxitérmino un índice basado en el complemento del número binario que representa (otra vez, asegurándonos que las variables se escriben en el mismo orden, usualmente alfabético decreciente:  $c, b, a$ ). También **puede numerarse nuevamente la tabla de verdad, pero esta vez, de abajo hacia arriba**. Y en este caso, **se buscan los ceros (0) de la función**. Así, por ejemplo, podemos asignar  $M_1$  (maxitérmino 1) al maxitérmino  $\bar{c} + \bar{b} + a$ . De forma similar,  $M_7$  de tres variables debería ser  $c + b + a$  y  $M_0$  es  $\bar{c} + \bar{b} + \bar{a}$ .

##### • Función equivalente expresada como producto de sumas

Se puede ver fácilmente que un maxitérmino sólo da como resultado un cero para una única entrada de la función lógica. Por ejemplo, el maxitérmino  $M_6$ ,  $c + b + \bar{a}$  es falso solo cuando  $c$  y  $b$  son falsos y  $a$  es verdadero. Por ello, la entrada  $c = 0, b = 0, a = 1$  da como resultado un cero. Si tenemos la tabla de verdad de una función lógica, es posible escribir la función como “producto de sumas”. Por ejemplo, dada la tabla de verdad:

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

**Tabla 12.** Tabla de verdad (ejemplo) para una función lógica de tres variables.

	c	b	a	$f(c, b, a)$	Minitérmino	Expresión
7	0	0	0	1		
6	0	0	1	0	$M_6$	$c + b + \bar{a}$
5	0	1	0	1		
4	0	1	1	0	$M_4$	$c + \bar{b} + \bar{a}$
3	1	0	0	0	$M_3$	$\bar{c} + b + a$
2	1	0	1	1		
1	1	1	0	0	$M_1$	$\bar{c} + \bar{b} + a$
0	1	1	1	0	$M_0$	$\bar{c} + \bar{b} + \bar{a}$

**Tabla 13.** Maxitérminos de una función lógica (ejemplo). Se numera de abajo hacia arriba (primera negación) y se toman en cuenta los ceros (segunda negación). Teniendo en cuenta que cada maxitérmino sólo vale 0 para una combinación de las entradas, podemos asociar a cada fila de una tabla de verdad un único maxitérmino que valdrá 0 sólo cuando las entradas tomen el valor de esa fila.

Observamos que las filas que tienen como salida un 0 son las de posiciones 0, 1, 3, 4 y 6 (numeradas desde abajo hacia arriba). Entonces podemos escribir  $f(c, b, a)$  como un producto de maxitérminos  $M_0, M_1, M_3, M_4$  y  $M_6$ .

$$f(c, b, a) = (\bar{c} + \bar{b} + \bar{a}) \cdot (\bar{c} + \bar{b} + a) \cdot (\bar{c} + b + a) \cdot (c + \bar{b} + \bar{a}) \cdot (c + b + \bar{a})$$

Si queremos verificar esto, tendremos que la tabla de verdad de la función, calculándola directamente, será la misma.

Una notación más simple para expresar un producto de maxitérminos, teniendo en cuenta que cada minitérmino está asociado a una fila de la tabla de verdad, consiste en enumerar los números de las filas de dicha tabla (colocados desde abajo hacia arriba) en las que la función lógica vale 0. Así, el producto de maxitérminos anterior puede expresarse como:  $\Pi_3(0, 1, 3, 4, 6)$ , donde “ $\Pi$ ” indica “producto de maxitérminos” o *productoria*, el subíndice “3” señala que se trata de una función de tres variables, y entre paréntesis se colocan las posiciones de la tabla de verdad (numeradas desde abajo hacia arriba) en las que la función vale 0.

Nuevamente conviene recordar que las formas canónicas NO son formas simplificadas o mínimas de la función; todo lo contrario, son funciones expandidas, que se utilizan para poner de manifiesto su comportamiento.

#### • Pasaje de minitérminos a maxitérminos

Complementado dos veces una suma de minitérminos se obtiene un producto de maxitérminos<sup>22</sup>. Se aplican los Teoremas de Doble Negación y De Morgan. Tomemos, como ejemplo sencillo, dos minitérminos:

$$c \cdot b \cdot \bar{a} + c \cdot \bar{b} \cdot \bar{a} \quad \text{se aplica Teorema de Doble Negación (Teorema V);}$$

$$\overline{c \cdot b \cdot \bar{a} + c \cdot \bar{b} \cdot \bar{a}} \quad \text{se aplica Teorema de De Morgan (Teorema VI);}$$

$$(\overline{c \cdot b \cdot \bar{a}}) \cdot (\overline{c \cdot \bar{b} \cdot \bar{a}}) \quad \text{nuevamente Teorema de De Morgan al interior de los términos;}$$

<sup>22</sup> El pasaje inverso también es válido y formalmente análogo en sus operaciones y desarrollo.

$$(\bar{c} + \bar{b} + a) \cdot (\bar{c} + b + a)$$

En otras palabras, hemos partido de los minitérminos 110 y 100 ( $m_6$  y  $m_4$ ) y hemos llegado a los maxitérminos 110 y 100 ( $M_1$  y  $M_3$ ). ¿Por qué?

El hecho de elegir los ceros supone aplicar una negación. Estamos eligiendo “los resultados que no están como 1 (uno)”. Y numerar la tabla de verdad de abajo hacia arriba significa aplicar la otra de las dos negaciones. Es decir, considerar el complemento a 7 de cada una de las posiciones. Puede notarse (observar respectivamente los subíndices) que  $M_1$  es el complemento a 7 de  $m_6$  (1 y 6 suman 7); y  $M_3$  es el complemento a 7 de  $m_4$  (3 y 4 suman 7).

De manera operativa, por ejemplo, partiendo de una expresión<sup>23</sup> de suma de minitérminos  $\Sigma_3$  (0, 2, 5), se anotan primero “los que faltan”, es decir: 1, 3, 4, 6, 7. Tenemos aquí la primera negación. Luego se calcula el complemento a 7 de cada uno de ellos: 6, 4, 3, 1, 0. Vemos en ello la otra negación. Por último, se ordenan de menos a mayor resultando la expresión<sup>24</sup> de producto de maxitérminos  $\Pi_3$  (0, 1, 3, 4, 6).

Como resumen de conceptos, puede recordarse:

- *Minitérmino*: término de producto donde aparecen todas las variables de la función con cualquier estado de complemento. Cada variable aparece complementada si su valor es 0 y sin complementar si es 1.
- *Maxitérmino*: término de suma donde aparecen todas las variables de la función con cualquier estado de complemento. Cada variable aparece complementada si su valor es 1 y sin complementar si es 0.
- *Suma canónica*: Expresión algebraica de una función lógica como la suma de los minitérminos que hacen 1 la función.
- *Producto canónico*: Expresión algebraica de una función lógica como el producto de los maxitérminos que hacen 0 la función.

Otro ejemplo aclarará la metodología para el pasaje de una expresión representada como minitérminos a su expresión equivalente como maxitérminos. Sea la función definida mediante la siguiente tabla:

<sup>23</sup> Ejemplo previo de minitérminos, ver páginas anteriores.

<sup>24</sup> Ejemplo previo de maxitérminos, ver páginas anteriores.

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

**Tabla 14.** Tabla de verdad, ejemplo.

Su expresión simbólica será:

$f = \Sigma_3 (2, 4, 5, 6, 7)$ , dada como suma de minitérminos.

Los términos que **no** pertenecen a la función son:

$$\bar{f} = \Sigma_3 (0, 1, 3)$$

El complemento de esto último es:

$$\bar{\bar{f}} = f = \Sigma_3 (0, 1, 3)$$

Luego, aplicando De Morgan será:

$$f = \Pi_3 (7, 6, 4)$$

Observemos que 7, 6, 4 son los complementos a 7 de 0, 1, 3 respectivamente.

Se ordenan de menor a mayor como:

$$f = \Pi_3 (4, 6, 7), \text{ expresada ya como producto de maxitérminos.}$$

En general y para facilitar el cálculo de los maxitérminos, el complemento a  $(2^n - 1)$  de los minitérminos que no pertenecen a la función, representa a los maxitérminos.

Como práctica sugerimos enunciar la expresión algebraica en sus dos formas.

Por ejemplo los maxitérminos en la función:

$$f = \Pi_3 (4, 6, 7)$$

permiten escribir la expresión como producto de sumas canónicas:

$$f_{(c, b, a)} = (c + \bar{b} + \bar{a}) \cdot (c + b + \bar{a}) \cdot (c + b + a)$$

## 7.5. Simplificación de funciones lógicas

A raíz de lo expuesto anteriormente, se infiere la necesidad de simplificar las funciones lógicas a fin de lograr circuitos más sencillos y eficientes. Existen varios métodos de simplificación.

Podremos clasificarlos en tabulares (gráficos) y numéricos. Los métodos gráficos como los de Veitch o Karnaugh permiten simplificar funciones de pocas variables.

Métodos numéricos como el de Quine McCluskey, en cambio, suministran una metodología apta para cualquier número de variables, aún mucho mayor ( $>4$ ).

La simplificación, en el caso de suma de productos, intenta reducir la cantidad de variables que intervienen en cada producto. A su vez baja la cantidad de productos. El análisis dual permite obtener ventajas análogas en el producto de sumas. Eso sí, resultan productos (o sumas) no canónicos.

### 7.5.1 Simplificación algebraica por aplicación de Postulados y Teoremas

Los métodos de simplificación algebraica se basan en la aplicación de los postulados y teoremas. Para hacerlo algebraicamente, primero se identifican y agrupan los términos donde las variables se repiten salvo una de ellas que, aparece en su forma directa en un término y negada en el otro. Entonces se aplica el recíproco del postulado N° IV. Luego aplicando el postulado N° VI se reemplaza la variable agrupada por el término neutro que corresponda y por último se aplica el postulado N° V.

### • Ejemplos

Dada la función:

$$f(c, b, a) = (c + b + \bar{a}) \cdot (c + b + a)$$

Se podrá agrupar:

$$f(c, b, a) = (c + b) + (a \cdot \bar{a}) = (c + b) + 0 = (c + b)$$

La variable que desaparece por efecto de la simplificación, tiene un peso tal en el ordenamiento elegido, que coincide con la diferencia entre los valores numéricos decimales asignados inicialmente a cada término canónico de la función.

Por ejemplo, dada la función expresada como suma de minitérminos:

$$f(c, b, a) = \bar{c} \cdot \bar{b} \cdot a + \bar{c} \cdot b \cdot a + c \cdot b \cdot \bar{a} + c \cdot b \cdot a$$

Aplicamos recíproca de la distributiva (postulado IV) entre los minitérminos 1 con el 3 (diferencia 2, se simplificará la variable b) y 6 con el 7 (diferencia 1, se simplificará la variable a).

$$f(c, b, a) = \bar{c} \cdot a \cdot (\bar{b} + b) + c \cdot b \cdot (\bar{a} + a)$$

Aplicamos ahora el postulado del elemento opuesto:

$$f(c, b, a) = \bar{c} \cdot a \cdot 1 + c \cdot b \cdot 1$$

Ahora bien, como “1” es el elemento neutro del producto, resulta:

$$f(c, b, a) = \bar{c} \cdot a + c \cdot b$$

### 7.5.2 Simplificación por el Método de Karnaugh

Maurice Karnaugh es un célebre físico, investigador y docente que nació en la ciudad de Nueva York en 1924. Trabajó en los laboratorios de telecomunicaciones de las Compañías IBM y Bell. En la década de 1950 desarrolló un método gráfico para simplificar funciones lógicas que se utiliza mundialmente y lleva su nombre.

#### • Los términos adyacentes y el mapa de Karnaugh

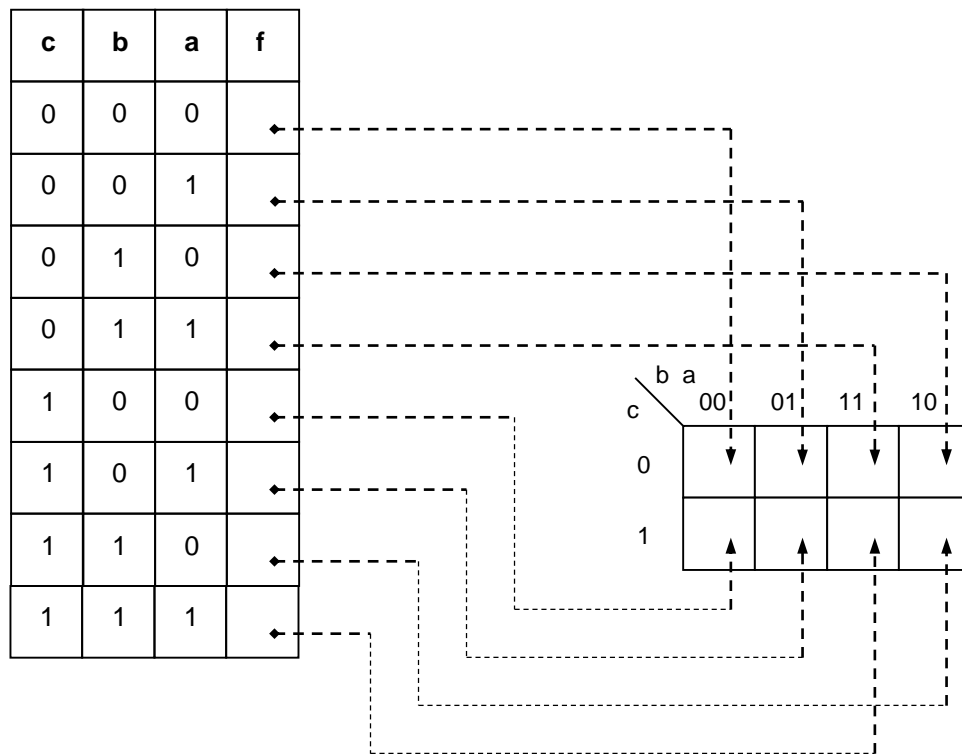
Se denominan *términos adyacentes* a los que difieren en el estado de una única variable lógica. Es decir, tienen las mismas variables con el mismo estado de complemento, excepto una. Por ejemplo:

$$a b c \quad \text{y} \quad a b \bar{c} \quad \text{son adyacentes.}$$

Los términos adyacentes pueden simplificarse fácilmente aplicando los Postulados del Álgebra de Boole:

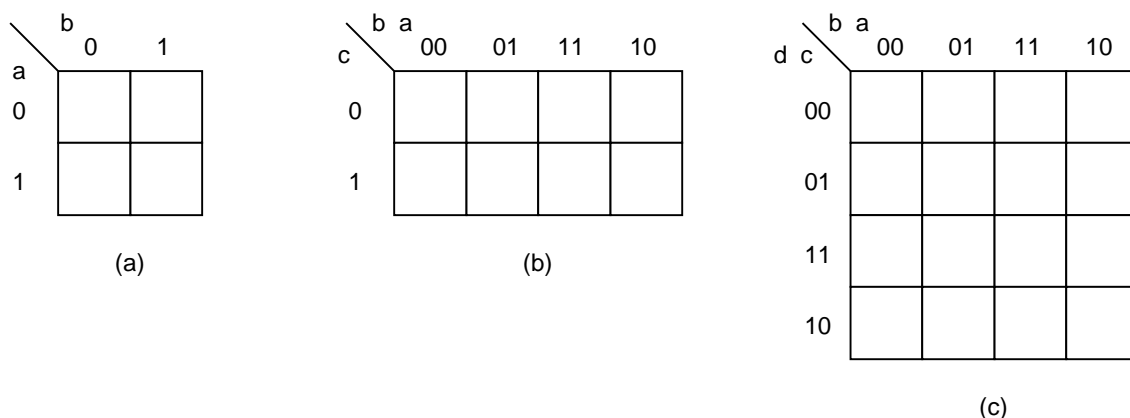
$$a b c + a b \bar{c} = a b (c + \bar{c}) = a b$$

El método de Karnaugh está basado en la búsqueda de términos adyacentes en la tabla de verdad. Para encontrarlos fácilmente, se dispone la tabla de verdad de tal forma que los valores de las variables de entrada vecinos resulten adyacentes. Esta tabla, así ordenada, recibe el nombre de diagrama o *mapa de Karnaugh*.



**Tabla 15.** La tabla de verdad y el mapa de Karnaugh.

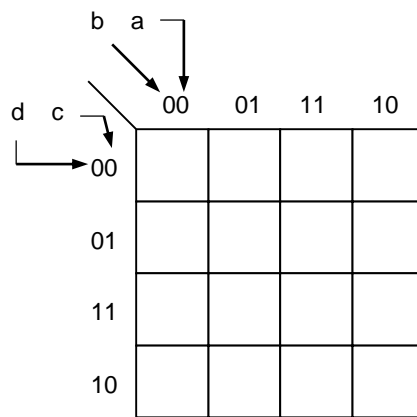
El método de Karnaugh se utiliza, habitualmente, para simplificar funciones lógicas de hasta cuatro variables. Los mapas tienen  $2^n$  cuadros, siendo  $n$  el número de variables que intervienen en la función. En la tabla anterior se ha tomado como ejemplo un diagrama para tres variables. El aspecto de los mapas de Karnaugh es el siguiente:



**Figura 20.** Mapas de Karnaugh para (a) dos, (b) tres y (c) cuatro variables lógicas.

En la construcción de los diagramas, es importante observar:

- Se ha trazado una línea diagonal en la esquina superior izquierda. Por encima y por debajo de ella se escriben los nombres de las variables que intervienen ( $a$ ,  $b$ ,  $c$ ,  $d$  según el mapa) en un orden determinado que, una vez establecido, debe respetarse.
- Para el mapa de cuatro variables, por ejemplo, los pares de ceros y unos que figuran en la parte superior del diagrama representan las combinaciones posibles de las variables  $b$  y  $a$ , en ese orden; mientras que los pares de ceros y unos del lado izquierdo indican las combinaciones de las variables  $d$  y  $c$ , también en ese orden.
- Leídos de izquierda a derecha y de arriba hacia abajo, los pares de ceros y unos que se han escrito en tercero y cuarto lugar presentan una inversión en el orden habitual binario (00, 01, **11**, **10**). Esta permutación responde a la necesidad de cambio de una sola variable lógica por cuadro adyacente (términos adyacentes) y constituye una de las cualidades fundamentales del diagrama de Karnaugh.

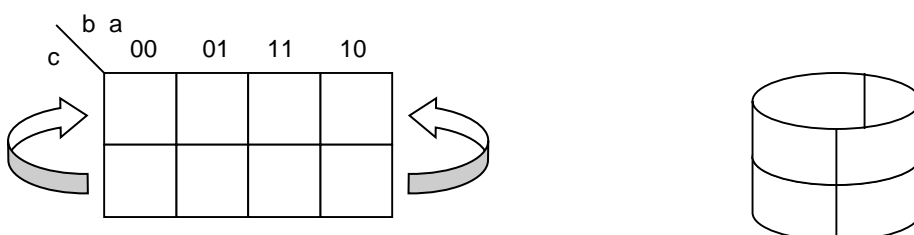


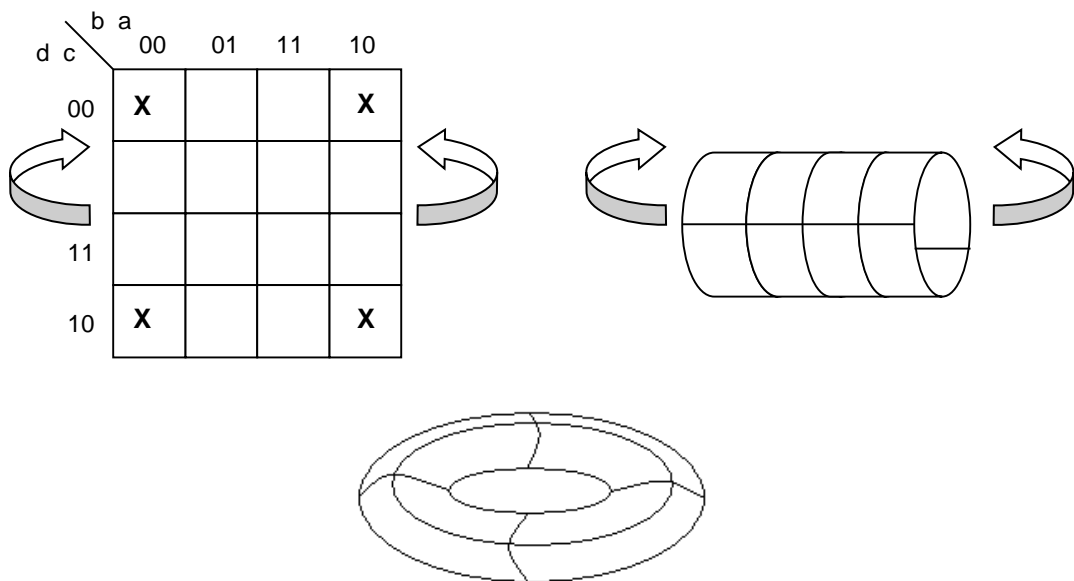
**Figura 21.** Importancia del orden y de la correspondencia de las variables en la construcción del mapa de Karnaugh.

#### • La adyacencia de los casilleros de los extremos

Otra propiedad notable del diagrama de Karnaugh es que los extremos corresponden, también, a términos adyacentes. Como si fueran casilleros vecinos. Es decir, no tenemos adyacencia gráfica en el plano de la página, pero existe adyacencia algebraica entre estos cuadros del diagrama. Esto se pone de manifiesto al comprobar, por ejemplo, que desde las posiciones “**10**” (situadas en el extremo derecho del mapa) a las posiciones “**00**” (ubicadas a la izquierda) sólo cambia el estado de una variable lógica, como puede apreciarse en la Fig. 22.

El mapa de Karnaugh para tres variables puede pensarse, entonces, como un cilindro; el de cuatro variables, como un anillo o toroide (Fig. 23). En este último, además de la adyacencia lateral, los casilleros situados en la parte superior son adyacentes con los de la línea inferior. Por este motivo existe, además, adyacencia algebraica entre las cuatro esquinas. Las figuras siguientes ayudan a ilustrar de manera visual estos conceptos.

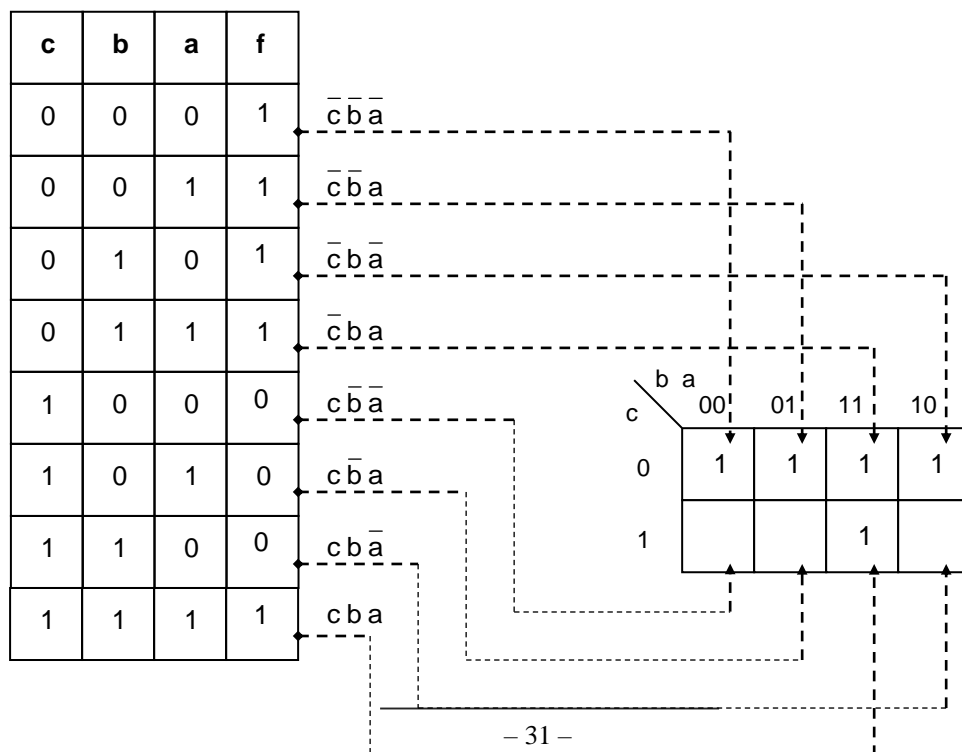


**Figura 22.** Casilleros adyacentes en el mapa de Karnaugh para tres variables.**Figura 23.** Casilleros adyacentes en el mapa de Karnaugh para cuatro variables. Los cuatro casilleros de las esquinas (señalados con **X**) también son adyacentes entre sí.

#### • Diagrama de Karnaugh para funciones de tres variables

El mapa de Karnaugh es, en realidad, una tabla de *minitérminos*. Consideremos la tabla de verdad de la Fig. 24(a). Como primer paso, puede escribirse la expresión booleana en la forma de minitérminos. Esta *suma de productos* es una expresión no simplificada.

Luego, se anotan en el mapa de Karnaugh los 1 (unos) de la función, en las posiciones previstas. Cada 1 corresponde a un grupo de variables ligadas por la operación de producto lógico (por ejemplo:  $c \cdot b \cdot a$ ).





Expresión no simplificada:  $f(c, b, a) = \bar{c} \cdot \bar{b} \cdot \bar{a} + \bar{c} \cdot \bar{b} \cdot a + \bar{c} \cdot b \cdot a + \bar{c} \cdot b \cdot \bar{a} + c \cdot b \cdot a$  (1)

**Figura 24(a).** Utilización de un diagrama de Karnaugh para tres variables.

A continuación se agrupan en el diagrama los conjuntos adyacentes de 1 (unos), formando lazos cuya cantidad de 1 (unos) sea igual a una potencia de 2. Es decir, pueden enlazarse ocho, cuatro o dos 1 (unos). Tomar un solo 1 (uno) también es válido, si es que se encuentra “solo”, sin otros 1 (unos) adyacentes. En este último caso, como se comprobará más adelante, estaremos tomando el minitérmino completo y no habrá simplificación.

En el diagrama de la Fig. 24(b) se realizan dos lazos o grupos. El lazo que aparece sombreado contiene cuatro 1 (unos) y el otro lazo, dos.

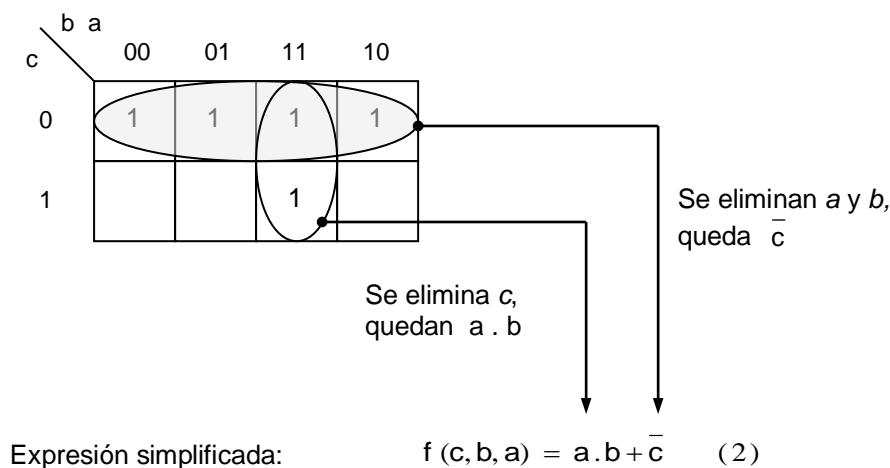
De los lazos, es importante destacar que deben contener la mayor cantidad de 1 (unos) que sea posible agrupar, con la condición – como se ha señalado – de que dicha cantidad sea una potencia de dos. Cada lazo, además, debe contener al menos un 1 (uno) que le sea propio, es decir, que no esté compartido con otro lazo.

Por último, se realiza la eliminación de variables o simplificación. *Cuando una variable y su complemento están en un lazo, esa variable se elimina.*

Observemos que el lazo sin sombrear de la Fig. 6(b) contiene los términos  $c = 0$  y  $c = 1$  (o lo que es lo mismo,  $\bar{c}$  y  $c$ ), por lo que puede eliminarse la variable  $c$ , quedando el término  $a \cdot b$  (señalado por los valores de la columna  $a = 1$  y  $b = 1$ ).

Por otra parte, el lazo sombreado contiene tanto  $a$  como  $\bar{a}$ , y tanto  $b$  como  $\bar{b}$ . De manera que pueden eliminarse las variables  $a$  y  $b$ , quedando sólo  $\bar{c}$  (señalado por la fila  $c = 0$ ).

Los términos obtenidos se vinculan mediante una suma lógica (OR).



**Figura 24(b).** Utilización de un diagrama de Karnaugh para tres variables (cont.)

Podemos comprobar que esta función simplificada es mucho más fácil de implementar, mediante compuertas lógicas, que la de la Fig. 24(a).

Naturalmente, las expresiones (1) y (2) son equivalentes, con idéntica tabla de verdad.

### • Diagrama de Karnaugh para funciones de cuatro variables

Los 1 (unos) en cualquier mapa de Karnaugh pueden anotarse desde la tabla de verdad de la función o bien, a partir de la expresión de la sumatoria que resume su primera forma canónica. Por ejemplo, sea la función  $f$  de cuatro variables, definida por sus minitérminos:

$$f(d, c, b, a) = \sum_4(0, 2, 3, 4, 7, 8, 9, 10, 11, 13, 15) \quad (3)$$

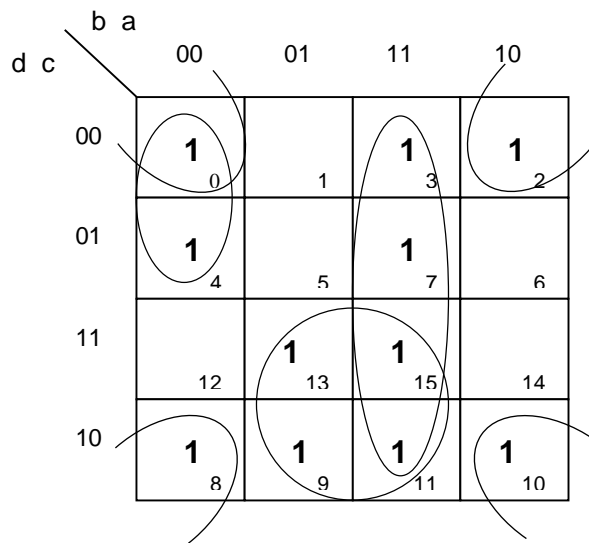
Es habitual numerar los casilleros del diagrama de Karnaugh con las posiciones decimales de la tabla de verdad, como guía. En el caso de una función lógica de cuatro variables, se numeran los casilleros desde el 0 (cero) hasta el 15 (quince), cuidando de guardar el orden mencionado en los párrafos anteriores. Luego, en los casilleros cuyo número aparece en la expresión de la sumatoria, se escribe un 1 (uno), como se muestra en la Fig. 25. Este proceder es análogo a transcribir los 1 (unos) desde la tabla de verdad.

Conviene recordar que, de trabajar con sólo tres variables, la numeración de los casilleros será desde el 0 (cero) hasta el 7 (siete), en el mismo orden que se observa en las dos primeras filas del diagrama siguiente.

		b a			
		00	01	11	10
d c	00	<b>1</b> 0		<b>1</b> 3	<b>1</b> 2
	01	<b>1</b> 4		<b>1</b> 7	
	11		<b>1</b> 13	<b>1</b> 15	
	10	<b>1</b> 8	<b>1</b> 9	<b>1</b> 11	<b>1</b> 10

**Figura 25.** Utilización de un diagrama de Karnaugh para cuatro variables.

Una vez anotados los 1 (unos) en el diagrama, se trazan los lazos respetando las condiciones conocidas: se agrupan éstos de a ocho, de a cuatro, de a dos o de a uno; los grupos deben ser lo más amplios posibles y **cada grupo debe contener al menos un 1 (uno) que le sea propio**. Al trabajar con cuatro variables, es más probable encontrar varias alternativas correctas de agrupamiento. Todas ellas serán equivalentes – con la misma tabla de verdad – aunque con expresiones algebraicas diferentes. Una alternativa para los grupos en la figura anterior es la que se muestra en la Fig. 26.



**Figura 26.** Grupos en un diagrama de Karnaugh para cuatro variables.

Una vez trazados los lazos, se puede leer el diagrama y escribir:

Del lazo 0-4 se obtiene:  $\bar{d} \cdot \bar{b} \cdot \bar{a}$

Del lazo 3-7-15-11:  $b \cdot a$

Del lazo 13-15-9-11:  $d \cdot a$

Del lazo que agrupa las cuatro esquinas 0-2-8-10:  $\bar{c} \cdot \bar{a}$

Luego, la función simplificada y expresada como suma de productos (minitérminos) será:

$$f(d, c, b, a) = \Sigma_4 (0, 2, 3, 4, 7, 8, 9, 10, 11, 13, 15) = \bar{d} \cdot \bar{b} \cdot \bar{a} + b \cdot a + d \cdot a + \bar{c} \cdot \bar{a} \quad (4)$$

Las expresiones (3) y (4) son, por supuesto, equivalentes.

Es interesante observar que en los grupos de cuatro variables se eliminan dos de ellas, en los grupos de dos; una. De existir un grupo de ocho, se eliminarían cuatro variables.

#### • Funciones simplificadas como producto de sumas

Aunque el método de Karnaugh se define como una tabla de minitérminos, su naturaleza simétrica permite aplicarlo para obtener directamente como resultado una función lógica simplificada expresada en la forma de producto de sumas.

Si en el diagrama de la Fig. 25 se anotan los ceros en los casilleros libres y se agrupan éstos respetando los mismos criterios de Karnaugh, se obtiene el mapa indicado en la Fig. 27.

	b a	00	01	11	10
d c	00	1 <sub>0</sub>	0 <sub>1</sub>	1 <sub>3</sub>	1 <sub>2</sub>
	01	1 <sub>4</sub>	0 <sub>5</sub>	1 <sub>7</sub>	0 <sub>6</sub>
	11	0 <sub>12</sub>	1 <sub>13</sub>	1 <sub>15</sub>	0 <sub>14</sub>
	10	1 <sub>8</sub>	1 <sub>9</sub>	1 <sub>11</sub>	1 <sub>10</sub>

**Figura 27.** Grupos de ceros para obtener una función simplificada como producto de sumas.

Sin embargo, al escribir los ceros en el diagrama, estamos representando – en realidad – los valores para los cuales la función se hace *falsa*. Esto motiva el cambio en el estado de complemento de las variables al realizar la lectura de los lazos sobre el diagrama.

Es decir, para el lazo 1-5 se elimina la variable  $c$  (a la izquierda del mapa puede observarse que  $c$  presenta valores diferentes para este lazo). Sin embargo, la variable  $d$  (que interviene con valores cero en este lazo) se representará como  $d$  sin negar, la variable  $b$  (que participa del lazo con valor cero) también se representará como  $b$  sin negar; mientras que la variable  $a$  se escribirá negada, porque su valor es 1 (uno) para este lazo. De acuerdo con lo anterior:

Del lazo 1-5 se elimina  $c$  y se obtiene:  $d + b + \bar{a}$

Del lazo 6-14 se elimina  $d$  y se obtiene:  $\bar{c} + \bar{b} + a$

Del lazo 12-14 se elimina  $b$  y se obtiene:  $\bar{d} + \bar{c} + a$

Luego, la función quedará simplificada como producto de sumas en la forma siguiente:

$$f(d, c, b, a) = d + b + \bar{a} \cdot \bar{c} + \bar{b} + a \cdot \bar{d} + \bar{c} + a \quad (5)$$

Por supuesto, la expresión (5) es una función lógica equivalente a las funciones (3) y (4), con idéntica tabla de verdad.

Existen autores que proponen otros modos similares de simplificar funciones para obtener como resultado una expresión de producto de sumas, aplicando – con sentido lógico válido – el método de Karnaugh.

#### • Ejemplo 1

Escribir la expresión booleana no simplificada, en forma de minitérminos, de la tabla de verdad siguiente y simplificarla aplicando el método de Karnaugh.

c	b	a	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

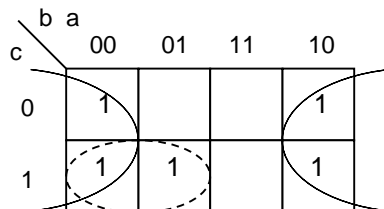
**Tabla 15.** Tabla de verdad para el Ejemplo 1.

### SOLUCIÓN

La función no simplificada, en forma de minterminos, es la siguiente:

$$f(c, b, a) = \bar{c} \cdot \bar{b} \cdot \bar{a} + \bar{c} \cdot b \cdot \bar{a} + c \cdot \bar{b} \cdot \bar{a} + c \cdot \bar{b} \cdot a + c \cdot b \cdot \bar{a}$$

Pueden trazarse el diagrama de Karnaugh y los lazos como se indica en la Fig. 28.



**Figura 28.** Diagrama de Karnaugh para el Ejemplo 1.

Es recomendable utilizar lápices de colores para dibujar e identificar los lazos. En la Fig. 28, el lazo en línea de trazos permite la eliminación de la variable  $a$  (que toma el valor directo y su complemento para este lazo, como puede verse en la parte superior del diagrama). Este término queda entonces  $c \cdot \bar{b}$ .

Luego, el lazo que agrupa los cuatro 1 (unos) de los extremos laterales del diagrama permite eliminar las variables  $c$  y  $b$ , con lo que el término queda simplemente  $\bar{a}$ .

La función simplificada, expresada como suma de productos, será entonces:

$$f(c, b, a) = c \cdot \bar{b} + \bar{a}$$

### • Ejemplo 2

Para la función lógica definida por:

$$f(c, b, a) = \sum_3(0, 1, 6)$$

Simplificarla aplicando el método de Karnaugh y expresar el resultado como:

- suma de productos.
- producto de sumas.

#### SOLUCIÓN

a) Para obtener la función simplificada y expresada como suma de productos, se puede comenzar dibujando el diagrama de Karnaugh habitual para una función de tres variables, anotando los 1 (unos) en las posiciones 0, 1 y 6 (por motivos de sencillez, en la figura no se numeran las celdas) y agrupándolos, como se muestra en la Fig. 30(a).

		b a			
c		00	01	11	10
	0	1	1		
	1				1

**Figura 30(a).** Diagrama de Karnaugh para el Ejemplo 3 (a).

La función simplificada, a partir de la lectura de los lazos o grupos del diagrama, puede escribirse como suma de productos en la forma siguiente:

$$f(c, b, a) = \bar{c} \cdot \bar{b} + c \cdot b \cdot \bar{a}$$

b) Para hallar la función simplificada y expresada como producto de sumas, se puede tomar el diagrama de Karnaugh anterior y anotar los 0 (ceros) en las posiciones restantes. En este caso, se agrupan los ceros respetando los mismos criterios de Karnaugh, como se muestra en la Fig. 30(b).

		b a			
c		00	01	11	10
	0	1	1	0	0
	1	0	0	0	1

**Figura 30(b).** Diagrama de Karnaugh para el Ejemplo 3 (b).

Recordando el cambio en el estado de complemento de las variables que intervienen en cada lazo o grupo de ceros, la función simplificada puede escribirse como producto de sumas en la forma siguiente:

$$f(c, b, a) = c + \bar{b} \cdot \bar{c} + b \cdot \bar{c} + \bar{a}$$

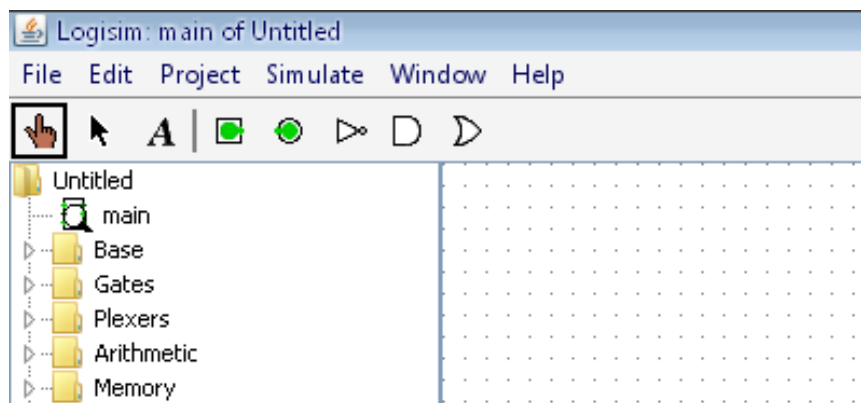
### • Simplificación de funciones con Logisim<sup>25</sup>

Logisim es una aplicación de freeware que permite diseñar circuitos lógicos y estudiar su comportamiento. Ofrece una amplia variedad de funciones a través de una interfaz gráfica muy sencilla. En este caso, sólo utilizaremos una parte del programa para la simplificación de una función lógica, con la visualización del Método de Karnaugh. Consideremos, como ejemplo, una función  $f(c, b, a)$  definida por la tabla de verdad siguiente.

c	b	a	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	1

**Tabla 16.** Tabla de verdad de la función  $f(c, b, a)$ .

La ventana principal de Logisim es la que se muestra en la Fig. 31.

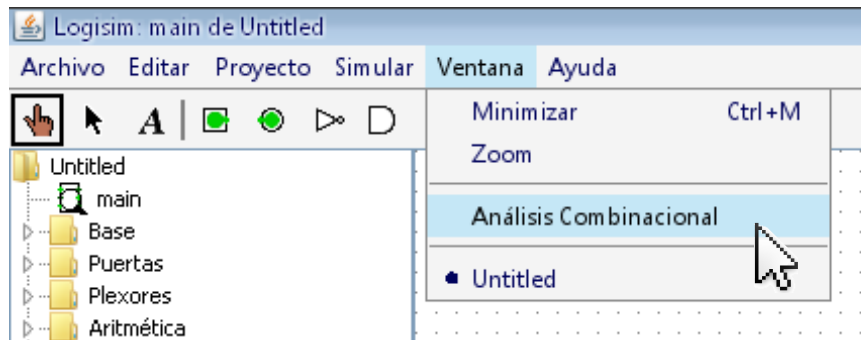


**Figura 31.** Ventana principal de Logisim.

Es posible seleccionar el idioma castellano en **File / Preferences... / International / Language / Spanish**. Se cierra este diálogo y aparece nuevamente la ventana principal.

A continuación, se selecciona el menú **Ventana / Análisis Combinacional**, como se muestra en la Fig. 32.

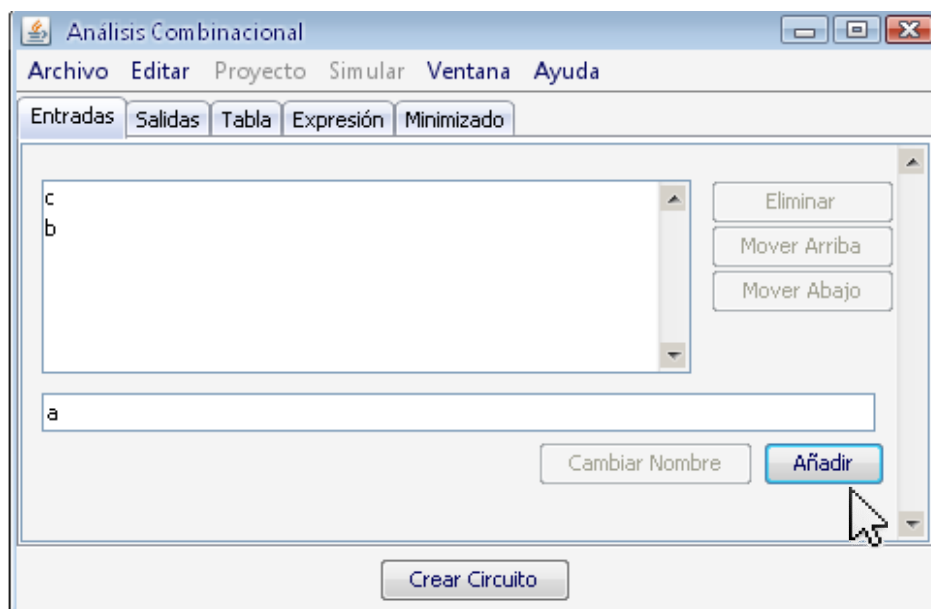
<sup>25</sup> © 2009, Carl Burch. Distributed under the GNU Public License, version 2.



**Figura 32.** Menú Ventana / Análisis Combinacional.

Al hacer clic en **Análisis Combinacional** aparece el cuadro de diálogo que se muestra en la Fig. 33. La primera pestaña se llama **Entradas**. El programa necesita que se definan las variables de entrada de la función que se desea estudiar o simplificar. Observemos nuevamente la tabla de verdad anterior. Las variables de entrada son **c**, **b**, **a**. En el campo inferior alargado del cuadro de diálogo se ingresan las variables, una por una y en orden. Se escribe **c** y se presiona el botón **Añadir** (la variable aparece en el campo superior), se escribe **b** y se presiona **Añadir**, y así sucesivamente.

Una vez ingresadas las variables de entrada (**c**, **b**, **a** en este caso), debe definirse la función de salida, que es **f**. Para ello, hacemos clic en la pestaña **Salidas** y definimos **f** con el mismo procedimiento anterior.



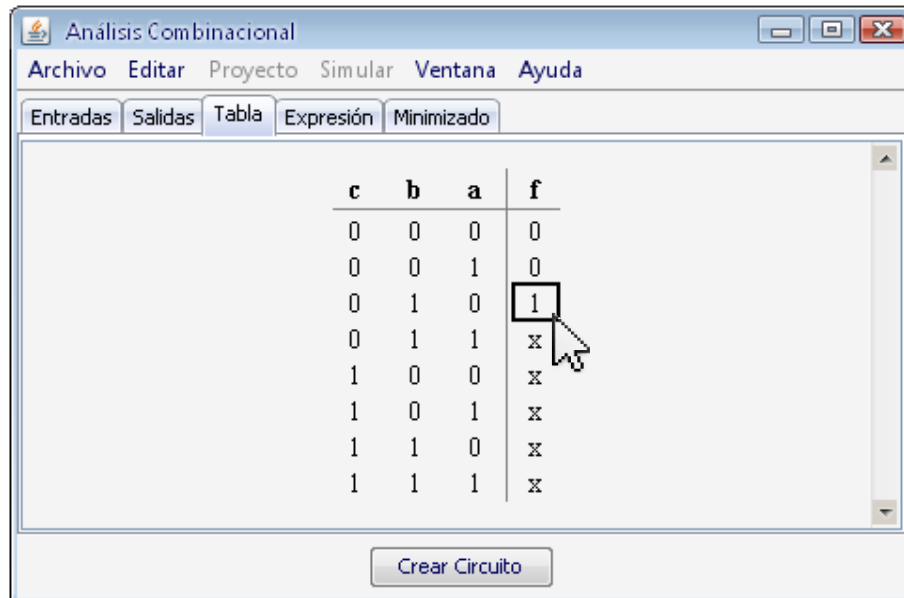
**Figura 33.** Ingreso de las variables de entrada.

Luego de definir las variables de entrada y salida, debemos indicar a Logisim cuál es la función lógica **f** que se desea representar o simplificar. Esto puede hacerse escribiendo algebraicamente la expresión de la función en la pestaña **Expresión** o ingresando los valores de la tabla de verdad en la pestaña **Tabla**. Cualquiera de las dos formas de hacerlo es igualmente válida. En este caso,



ingresaremos los valores de la tabla de verdad inicial. Para ello, hacemos clic en la pestaña **Tabla**.

La columna **f** contiene inicialmente posiciones señaladas con “x”. Haciendo varios clics sobre cada “x”, aparecen sucesivamente los valores “0”, “1” y nuevamente “x”. Con los clics necesarios, seleccionamos el valor de la función para cada línea de la tabla, como se muestra en la Fig. 34.



c	b	a	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	x
1	0	1	x
1	1	0	x
1	1	1	x

Crear Circuito

**Figura 34.** Ingreso de los valores de **f** en la tabla de verdad.

Una vez ingresados todos los valores de **f** en la tabla de verdad, la pestaña **Minimizado** nos mostrará automáticamente el Mapa de Karnaugh ya realizado y la función simplificada, como puede verse en la Fig. 35. Logisim puede ser útil, entonces, para la verificación de los ejercicios realizados. Sin embargo, es importante recordar que muchas funciones lógicas pueden admitir otras expresiones equivalentes, diferentes a las que propone el programa pero igualmente correctas. Lo mismo vale decir para los lazos del diagrama.



Salida: f

		b, a			
		00	01	11	10
c	0	0	0	1	1
	1	1	0	1	1

$b + c \bar{a}$

Fijar Como Expresión

Crear Circuito

**Figura 35.** Diagrama de Karnaugh y función simplificada.

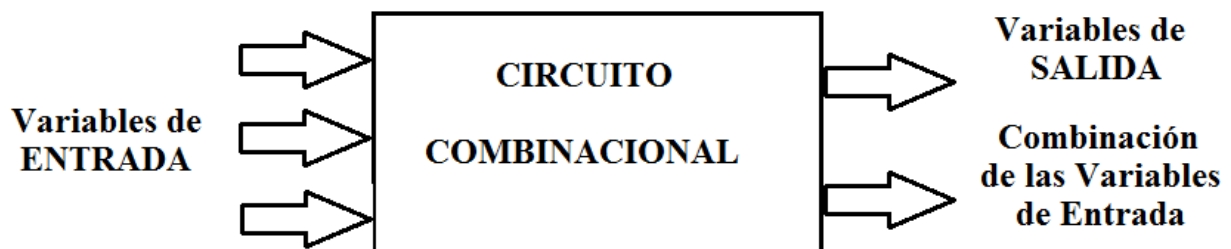
## 8. Implementación de funciones simples: Circuitos combinacionales

### 8.1. Introducción

Las compuertas lógicas tratadas con anterioridad, los postulados del Álgebra de Boole y los teoremas demostrados ofrecen la posibilidad de analizar y comprender algunos circuitos frecuentemente utilizados en sistemas automáticos de cómputo de datos y control digital. Por otra parte, la simplificación de funciones permite realizarlos de manera eficiente.

Desde el comienzo de la unidad 3, se han presentado distintos tipos de funciones, que sin embargo, comparten la misma característica:

Sus variables de SALIDA dependen exclusivamente de los valores de sus variables de ENTRADA. Este tipo de funciones, que se implementan por medio de circuitos, recibe el nombre de **CIRCUITOS COMBINACIONALES**, ya que su variables de SALIDA se obtienen exclusivamente como combinación de sus variables de ENTRADA.



Los circuitos combinacionales, además, proveen una representación gráfica tal que las aplicaciones lógicas y aritméticas resultan más claras.

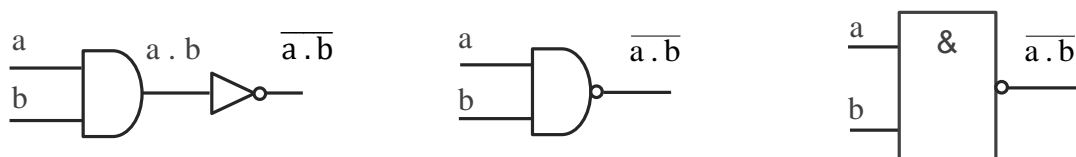
Las funcionalidades de los circuitos combinacionales que se exponen a continuación se emplean ampliamente en aplicaciones prácticas del hardware de sistemas digitales.

#### 8.1.1 Compuertas NAND y NOR

##### • Compuerta NAND

Debido a la simplicidad de diseño de circuitos digitales con compuertas del tipo NAND y NOR solamente, varias tecnologías como “lógica de transistor – transistor” (TTL) y otras, propiciaron la construcción de circuitos a partir de ellas.

La función NAND (NOT AND), corresponde a la negación del producto lógico. Es decir, se realiza el producto de las variables de entrada y luego se invierte la salida.



**Figura 36.** Compuerta NAND. Las tres representaciones son equivalentes (se utilizará la del centro).

Se puede decir que la salida de una compuerta NAND es 1 (uno) cuando alguna entrada es cero (detecta la presencia de al menos un cero). Desde otro punto de vista, la salida es cero si y solo si todas las entradas valen uno simultáneamente.

En una NAND, cuando una de las entradas es 1 (uno), la otra se invierte a la salida.

b	a	$f = \overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

**Tabla 17.** Tabla de verdad de la compuerta NAND.

Por ejemplo, si  $b = 1 \Rightarrow f = \overline{a}$

Asimismo, uniendo ambas entradas ( $a = b$ ), la salida será su negado  $f = \overline{a}$  o bien  $f = \overline{b}$ .

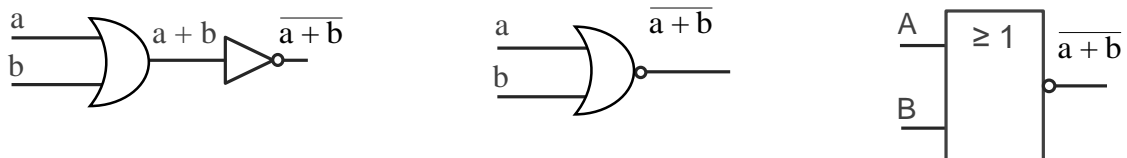
Si colocamos un inversor (negador) a la salida de una NAND obtendremos una AND.

Por otra parte, si negamos cada una de las entradas de una NAND, la función resultante corresponde a una OR. Esto surge de aplicar De Morgan.

#### • Compuerta NOR

La función NOR (NOT OR) corresponde a la negación de la suma lógica. Es decir, se realiza la suma de las variables de entrada y luego se invierte la salida.

Se puede decir que la salida de una compuerta NOR es 1 (uno) cuando todas las entradas son cero simultáneamente (detecta el valor cero en la entrada). Otro enfoque permite afirmar que la salida es cero si por lo menos una de las entradas vale uno.



**Figura 36.** Compuerta NOR. Las tres representaciones son equivalentes (se utilizará la del centro).

En una NOR, cuando una de las entradas es 0 (cero), la otra se invierte a la salida.

b	a	$f = \overline{a + b}$
0	0	1
0	1	1
1	0	1
1	1	0

#### 8.1.2 Compuerta XOR (Exclusive OR u “O Exclusiva”)

La compuerta “**O exclusiva**” entrega un uno a su salida cuando una de sus dos entradas está en uno, pero no simultáneamente: se excluye esta condición. En forma de función lógica, esto sería:

$$f = (b + a) \cdot b \cdot a$$

Esto difiere de la operación “O” (inclusiva) definida en el primer Postulado del Álgebra de Boole, en la cual la salida es uno cuando cualquiera de las entradas es uno, aún si todas tienen el valor uno al mismo tiempo.

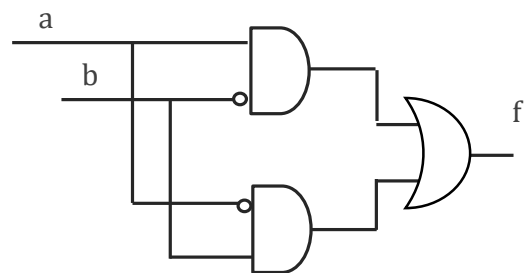
Otra forma de analizarla sería diciendo: la salida de una XOR es uno si una entrada no vale uno y la otra sí, o viceversa (la primera es uno y la segunda no). O bien, la salida de una XOR es uno cuando ambas entradas tienen valores lógicos distintos.

Ahora la expresión puede escribirse:

$$f = \bar{b} \cdot a + b \cdot \bar{a}$$

Las dos expresiones anteriores son equivalentes y poseen, por ello, la misma tabla de verdad.

b	a	$f = \bar{b} \cdot a + b \cdot \bar{a}$
0	0	1
0	1	1
1	0	1
1	1	0

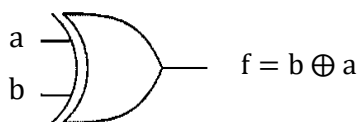


A la derecha se muestra una de las formas de implementar la función XOR mediante compuertas AND, OR e inversoras.

En la tabla de verdad de la función XOR se puede observar que la salida es uno cuando la cantidad de unos de entrada es “impar”. Por este motivo, se la emplea como *detector de paridad* (detecta paridad impar). Puede, además servir para *generar un bit de paridad par*. Su salida genera un uno cuando una combinación de entrada es impar y, agregando el bit generado a los bits de entrada, queda un conjunto de bit con paridad par<sup>26</sup>.

Otra característica importante es que cuando una de sus entradas es cero (por ejemplo,  $b = 0$ ); la salida es igual a la otra variable de entrada ( $f = a$ ). En cambio, cuando la primera es uno ( $b = 1$ ), la salida es el opuesto<sup>27</sup> de la segunda ( $f = \bar{a}$ ). Por este motivo se la puede utilizar como un inversor controlado.

Debido a su importancia y utilización extendida en muchas aplicaciones, a esta compuerta OR exclusiva se le ha asignado el siguiente símbolo y la notación de operación que se muestra:



$$f = b \oplus a \quad \text{se lee “b x-or a” o bien, “b or exclusiva a”}$$

Recordar:

$$f = \bar{b} \cdot a + b \cdot \bar{a} = b \oplus a$$

**Figura 37.** Compuerta XOR. Símbolo gráfico y notación operacional.

<sup>26</sup> Observar que las filas de la tabla anterior (sus tres columnas) tienen una cantidad par de unos, en todos los casos.

<sup>27</sup> Para razonar este comportamiento, se puede dividir la tabla a la mitad y observar, en la parte superior, que cuando la entrada “b” es “0”, la salida “f” toma los mismos valores que “a”, es decir, si  $a = 0$ ,  $f = 0$  y si  $a = 1$ ,  $f = 1$ . En la parte inferior de la tabla, cuando la entrada “b” es “1”, la salida “f” toma los valores opuestos a los de “a”, es decir si:  $a = 0$ ,  $f = 1$  y si  $a = 1$ ,  $f = 0$ .

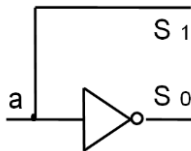
## 8.2. Circuitos multifunción

Un sistema que posee varias salidas y, por lo tanto, responde simultáneamente a varias funciones lógicas, se denomina multifunción.

### 8.2.1. Decodificadores

Un decodificador es un circuito combinacional de varias salidas (multifunción) que convierte la entrada binaria (de uno o más bit), en salidas correspondientes a productos canónicos (minitérminos). Cada salida tiene, por lo tanto, un solo uno. Este uno se ubica en el minitérmino que coincide con el valor asignado a la salida respectiva (por ejemplo, su equivalente en decimal).

El siguiente circuito lógico constituye un decodificador elemental. Está formado por una compuerta inversora y una simple línea (o podría darse por medio de una compuerta no inversora). Presenta una entrada “a” y dos salidas “S<sub>0</sub>” y “S<sub>1</sub>”. En la figura se muestra el esquema del decodificador elemental y su tabla de verdad.



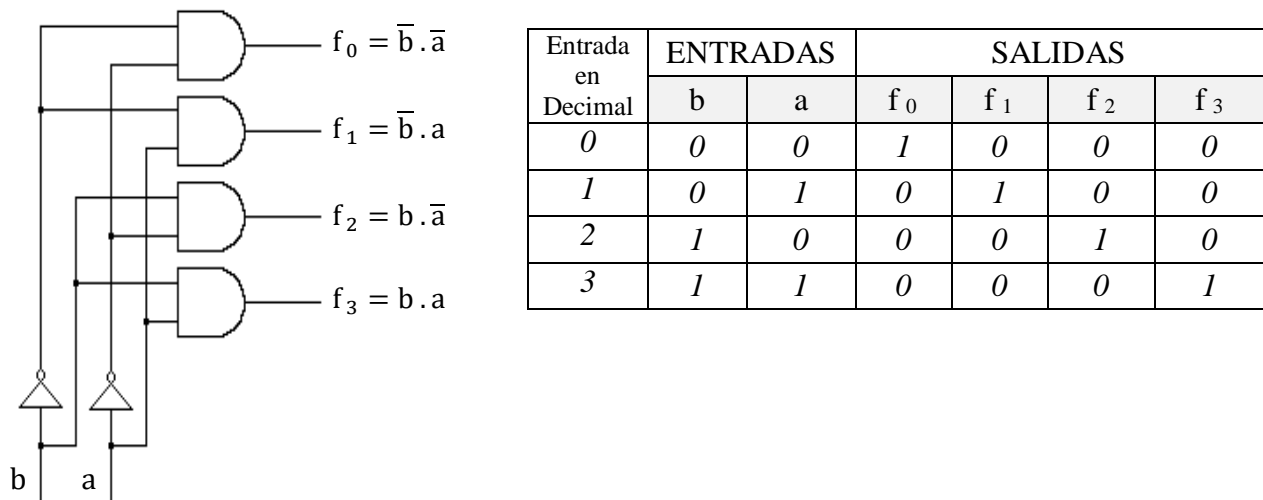
a	S <sub>0</sub>	S <sub>1</sub>
0	1	0
1	0	1

Las expresiones de las funciones lógicas serán: “S<sub>0</sub> =  $\bar{a}$ ” y “S<sub>1</sub> = a”.

Para entender las SALIDAS de un decodificador, se puede pensar en el visor ubicado sobre la puerta de la Planta Baja, de un ascensor. En dicho visor el ÚNICO número que se enciende es el del piso en el cual está el ascensor en ese momento y los demás están APAGADOS.



Un decodificador un poco más amplio es el siguiente. Se trata de un decodificador en el cual la entrada está formada por un código binario de dos bit (b, a), y cada salida se pone en uno SOLO SI su subíndice coincide con el código binario ingresado (decodificador binario a decimal).

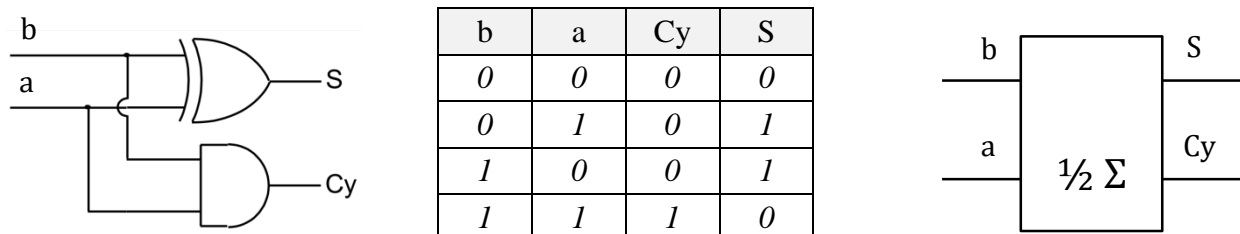


### 8.2.2. Suma aritmética

Un semi-sumador aritmético (en inglés, *half adder*), es también un circuito multifunción. Cuenta con entradas binarias de un bit “a” y “b” y las salidas: suma “S”, y arrastre “Cy” (o acarreo), de acuerdo al siguiente principio:

$$\begin{array}{r} + \quad a \\ \quad b \\ \hline C_y S \end{array}$$

La tabla de verdad se realiza colocando en ella los resultados de la suma binaria “Cy” y “S” para cada combinación de los valores de “b” y “a”. A continuación se muestra dicha tabla de verdad, el esquema lógico y las expresiones correspondientes a cada salida:



**Figura 38.** Semi-sumador. Circuito lógico, tabla de verdad y simbología abreviada.

Se puede observar que el arrastre o carry “Cy” coincide con el producto lógico:  $Cy = b \cdot a$ , mientras que la suma aritmética “S” coincide con la función O exclusiva:  $S = b \oplus a = \bar{b} \cdot a + b \cdot \bar{a}$ . Cuando necesitamos sumar números de más de un bit, por ejemplo la suma de dos números de 4 bits:  $A + B: a_3 a_2 a_1 a_0 + b_3 b_2 b_1 b_0$ , todas las columnas de la suma, con excepción de la primera, requerirán sumar tres bits.

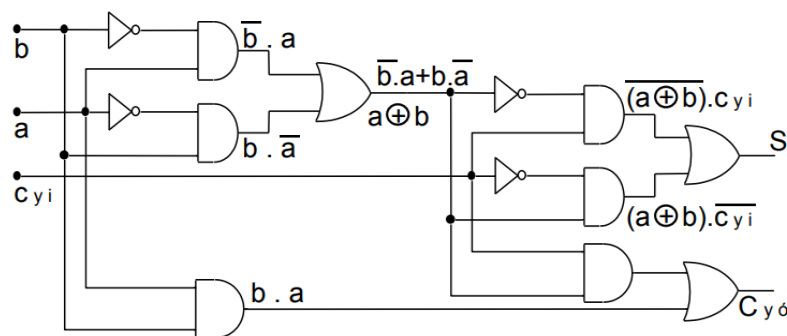
$$\begin{array}{rcccc}
 & \mathbf{c}_{Y3} & \mathbf{c}_{Y2} & \mathbf{c}_{Y1} & \mathbf{c}_{Y0} \\
 & \mathbf{a}_3 & \mathbf{a}_2 & \mathbf{a}_1 & \mathbf{a}_0 \\
 + & \mathbf{b}_3 & \mathbf{b}_2 & \mathbf{b}_1 & \mathbf{b}_0 \\
 \hline
 \mathbf{C}_{Y6} & \mathbf{S}_3 & \mathbf{S}_2 & \mathbf{S}_1 & \mathbf{S}_0
 \end{array}$$

Por ello, el sumador total (en inglés, *full adder*), tiene tres entradas binarias de un bit “a” y “b” y el arrastre de la suma anterior, que lo llamaremos de entrada “Cyi” y las salidas: suma “S”, y arrastre “Cyo” al cual nos referiremos como carry de salida.

La tabla de verdad se realiza colocando en ella los resultados de la suma binaria: “Cyo” y “S”, para cada combinación de los valores de “Cyi”, “b” y “a”.

ENTRADAS			SALIDAS	
Cyi	b	a	Cyo	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Implementar las funciones S y Cyo para esta tabla de verdad sería un poco complejo, como se observa en el siguiente circuito:



No obstante, tenemos aquí un notable ejemplo de la utilidad de las formas canónicas y de la aplicación de los postulados del Álgebra de Boole, para arribar a un circuito más simple.

Podemos analizar esta multifunción a partir de su tabla de verdad, considerando la suma de productos canónicos. Para la suma:  $S_{(C_{yi}, b, a)} = \Sigma_3 (1, 2, 4, 7)$ , de manera que:

$$S = \bar{C}_{yi} . \bar{b} . a + \bar{C}_{yi} . b . \bar{a} + C_{yi} . \bar{b} . \bar{a} + C_{yi} . b . a$$

Agrupando (postulado recíproco de la distributiva):

$$S = \bar{C}_{yi} . (\bar{b} . a + b . \bar{a}) + C_{yi} . (\bar{b} . \bar{a} + b . a)$$

Aplicando las definiciones de XOR y NOT XOR:

$$S = \bar{C}_{yi} . (b \oplus a) + C_{yi} . (\overline{b \oplus a})$$

que, si se observa detenidamente, es de la forma:  $\bar{m} . n + m . \bar{n}$ , por lo que puede aplicarse nuevamente la definición de XOR, obteniendo:

$$S = C_{yi} \oplus (b \oplus a)$$

Volviendo a analizar la tabla de verdad, ahora para el arrastre  $C_{yo}$ , se tiene:

$C_{y0} (C_{yi}, b, a) = \Sigma_3 (3, 5, 6, 7)$ , de manera que:

$$C_{y0} = \overline{C_{yi}} \cdot b \cdot a + C_{yi} \cdot \overline{b} \cdot a + C_{yi} \cdot b \cdot \overline{a} + C_{yi} \cdot b \cdot a$$

Agrupando (postulado recíproco de la distributiva):

$$C_{y0} = b \cdot a \cdot (\overline{C_{yi}} + C_{yi}) + C_{yi} \cdot (\overline{b} \cdot a + b \cdot \overline{a})$$

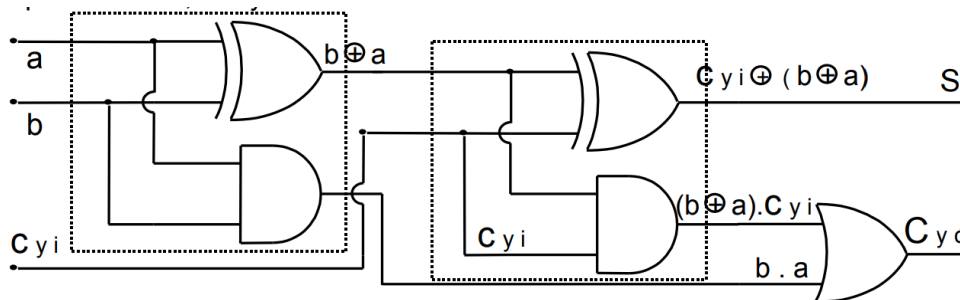
Aplicando el postulado del elemento opuesto de la suma  $a$ :  $\overline{C_{yi}} + C_{yi} = 1$ , y el del elemento neutro del producto  $a$ :  $b \cdot a \cdot (\overline{C_{yi}} + C_{yi}) = b \cdot a \cdot 1 = b \cdot a$ , se obtiene:

$$C_{y0} = b \cdot a + C_{yi} \cdot (\overline{b} \cdot a + b \cdot \overline{a})$$

Por definición de XOR:

$$C_{y0} = b \cdot a + C_{yi} \cdot (b \oplus a)$$

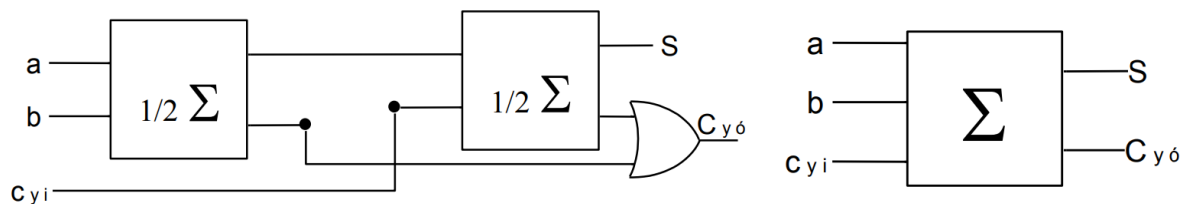
De esta manera, puede construirse el siguiente circuito con compuertas AND, OR y XOR:



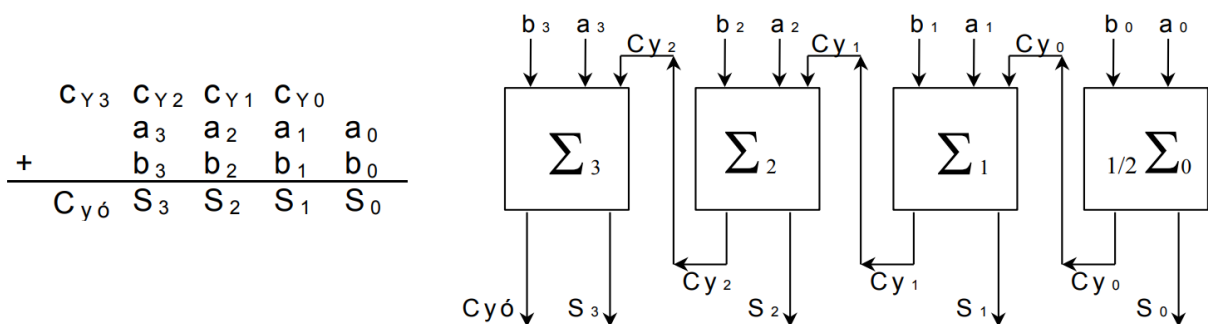
En este esquema se puede descubrir, por simple inspección, la presencia de dos semi-sumadores. El primero suma “a” con “b” y el segundo suma el resultado  $b \oplus a$  con  $C_{yi}$ .

El arrastre final, por su parte, se obtiene con la suma lógica de los arrastres parciales, es decir será 1 cuando cualquiera de los arrastres parciales sea 1.

El sumador total se puede representar gráficamente mediante semi sumadores, en forma de bloques. También, como un único bloque, como se aprecia en la figura siguiente:



Este último análisis se aproxima a la forma en la que frecuentemente realizamos la operación de suma de tres dígitos. Sumamos los primeros dos y ese resultado lo sumamos al tercero, reservando siempre el arrastre en cada paso. Para sumar números de varios bits, solemos aplicar la metodología tradicional, que transcribimos y la aplicamos al circuito esquemático simplificado:





### 8.2.3. Comparadores

Un comparador es un sistema que permite establecer si un número es mayor, igual o menor que otro. Es, asimismo, una multifunción. Básicamente podría razonarse para entradas binarias de un bit “A” y “B” (aunque podrían compararse binarios de “n” bit) y por ejemplo dos salidas: *igual* (“A = B”) y *mayor que* (“A > B”). La salida “A es igual a B” se pone en uno cuando las entradas A y B son iguales, mientras que la salida “A mayor a B” se pone en uno cuando el valor de A es mayor que el de B. Podría haber otra salida que informara cuando A es menor que B. La tabla de verdad de un comparador se puede deducir, entonces:

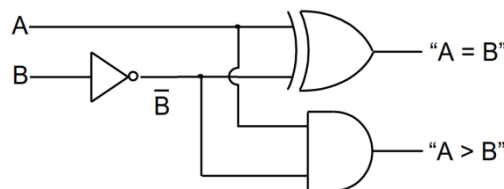
Igualdad (=)	
Si $A = B$	1
Si $A \neq B$	0

Mayor que (>)	
Si $A > B$	1
Si $A < B$ o $A = B$	0

B	A	A=B	A>B
0	0	1	0
0	1	0	1
1	0	0	0
1	1	1	0

$$A > B = A \cdot \bar{B}$$

$$A = B = \bar{A} \cdot \bar{B} + A \cdot B$$



Es fácil deducir la correspondencia entre las expresiones y el esquema lógico, aplicando Álgebra de Boole. El circuito del comparador nos muestra que contiene un semi - sumador que realiza la suma de las variables en sus entradas, y como la variable B está invertida (complemento a 1), se obtiene la resta mediante la suma del complemento a la base menos uno (de la variable B). Por otra parte este estudio permite reconocer que *una comparación no es otra cosa que una resta*. Al restar dos números, el resultado es cero si son iguales. Positivo si el primero es mayor que el segundo y negativo en el caso contrario.

Un concepto importante es el caso del comparador de igualdad:

B	A	A=B
0	0	1
0	1	0
1	0	0
1	1	1

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

Hemos reproducido a la derecha, la tabla de la XOR. Se concluye que el opuesto de la XOR es un comparador de igualdad. Por este motivo, a la XOR se la conoce también como comparador de desigualdad. Este efecto se consigue también negando una sola de las entradas (como se verifica en el circuito anterior, donde se hizo  $\bar{B}$ ).

Por lo cual, para la compuerta lógica XOR se cumple que:

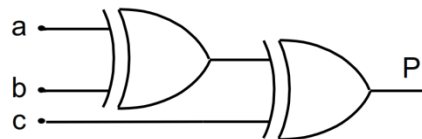
$$\overline{B \oplus A} = \bar{B} \oplus A = B \oplus \bar{A}$$

### 8.2.4. Generador (o detector) de bit de paridad

Un circuito generador de bit de paridad, permite obtener un bit de paridad par o impar. Agregándolo a un código para su transmisión, permitirá detectar los errores posibles de los sistemas teleinformáticos. Los detectores emplean el mismo concepto para verificar el código recibido. Se trata, en síntesis, de una aplicación de las XOR. Por simplicidad, supongamos un código de 3 bits, para el cual queremos generar un bit de paridad par en los unos. Esto es, la suma de los unos de un código (incluido el bit de paridad), debe ser par.

Nótese la tabla de verdad y compárese con la del sumador total (salida de suma S) anteriormente estudiado; como puede verse, son iguales.

c	b	a	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



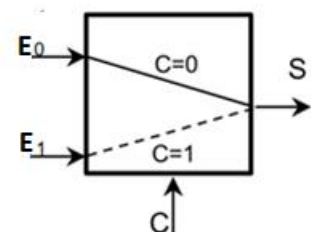
### 8.2.5. Multiplexores (MUX)

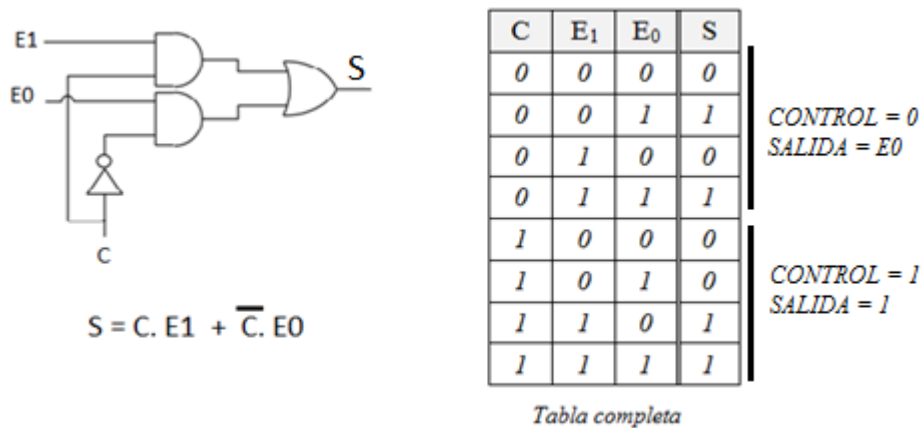
Un multiplexor es un sistema que permite seleccionar una de varias entradas de datos, mediante un conjunto de líneas de control. Así, por ejemplo, si tuviéramos varias computadoras y una sola impresora, podríamos seleccionar una de ellas para realizar tareas de impresión, mientras las demás continúan con otros trabajos. En muchas aplicaciones, aún analógicas, se emplea este concepto. Por ejemplo, en un equipo de audio, mediante un selector le indicamos al sistema aquello que queremos escuchar: radio o CD. Incluso seleccionamos radio AM o FM. De esta forma, el sistema electroacústico de salida (los parlantes), quedan conectados a la fuente (datos de entrada) que hemos seleccionado mediante los controles del panel frontal o el control remoto. Más específicamente, en un sistema digital, y para un sistema con dos posibles entradas de datos, la salida del MUX, en función de la entrada de control, es tal que coincide con la entrada  $E_0$  si el control se encuentra en estado “0” y será igual a  $E_1$  si la entrada de control se encuentra en “1”.



C	S
0	$E_0$
1	$E_1$

Tabla reducida



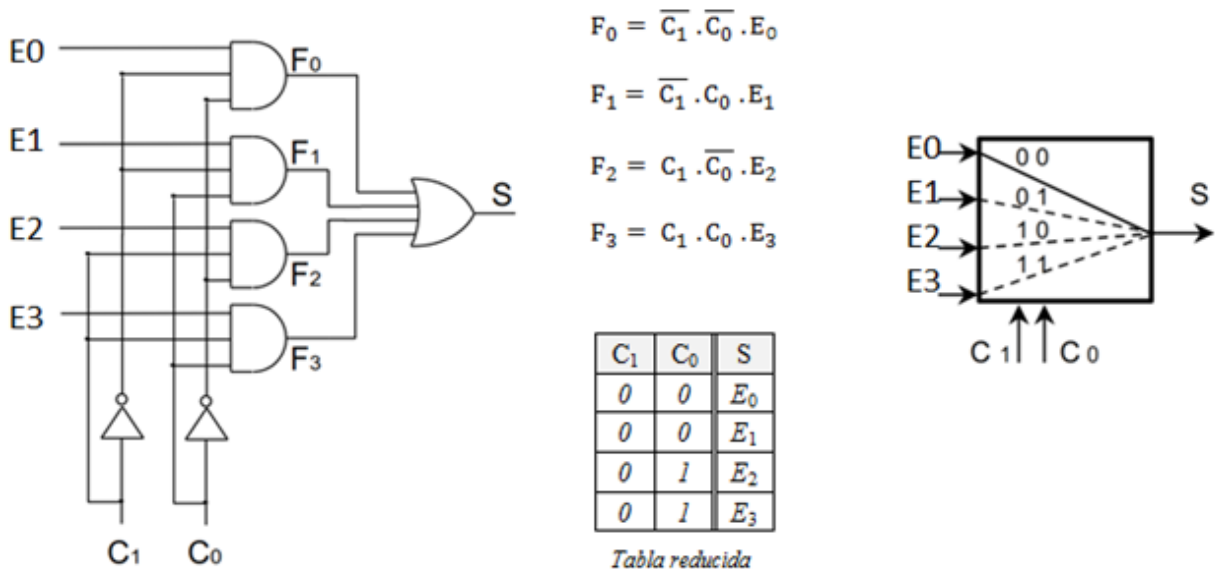


Un multiplexor, puede tener  $2^n$  entradas de datos para “n” terminales de control. Pero el MUX SIEMPRE TIENE UNA SOLA SALIDA.

Un MUX de dos entradas de control y 4 entradas de datos tiene el siguiente circuito, y tabla de verdad reducida:

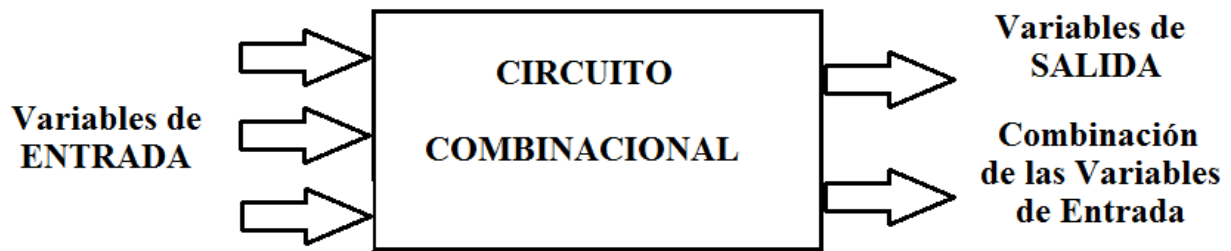
En este último caso, en total, disponemos de 6 entradas (cuatro de datos y dos de control). La tabla completa sería muy extensa, porque deberían escribirse las 64 combinaciones binarias correspondientes. De todas formas, la expresión booleana es fácil de deducir a partir de la tabla reducida:

$$S = \overline{C_1} \cdot \overline{C_0} \cdot E_0 + \overline{C_1} \cdot C_0 \cdot E_1 + C_1 \cdot \overline{C_0} \cdot E_2 + C_1 \cdot C_0 \cdot E_3$$

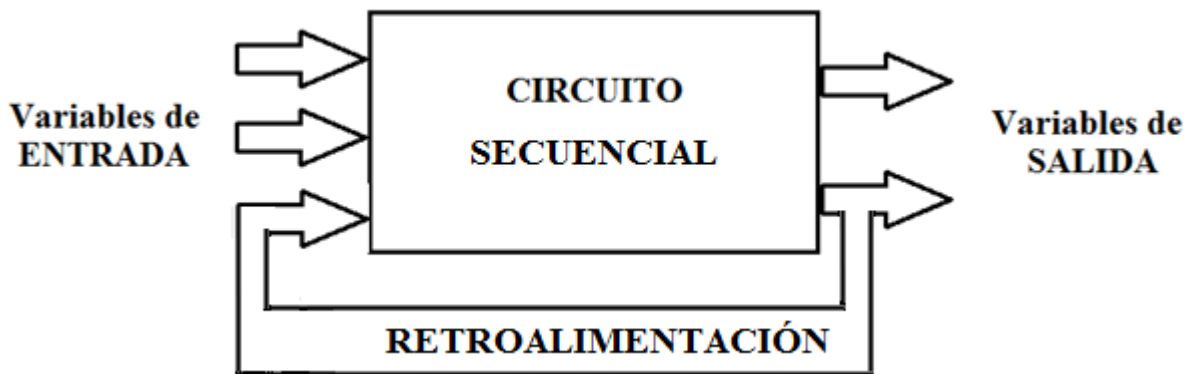


## 9. Circuitos secuenciales

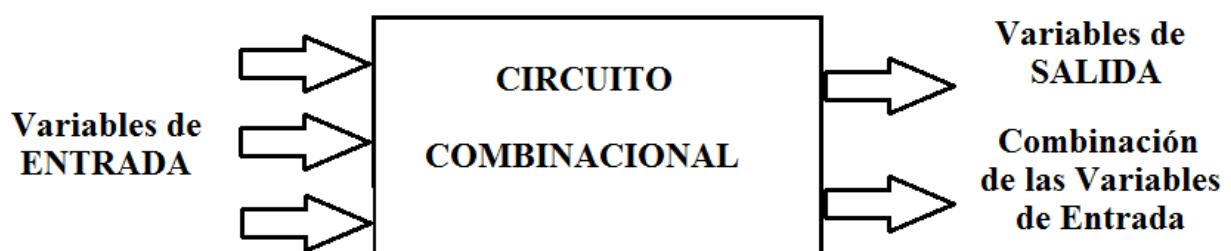
Como se explicó en apartados anteriores, un CIRCUITO COMBINACIONAL, es aquel en el cual sus variables de SALIDA se obtienen exclusivamente como combinación de sus variables de ENTRADA.



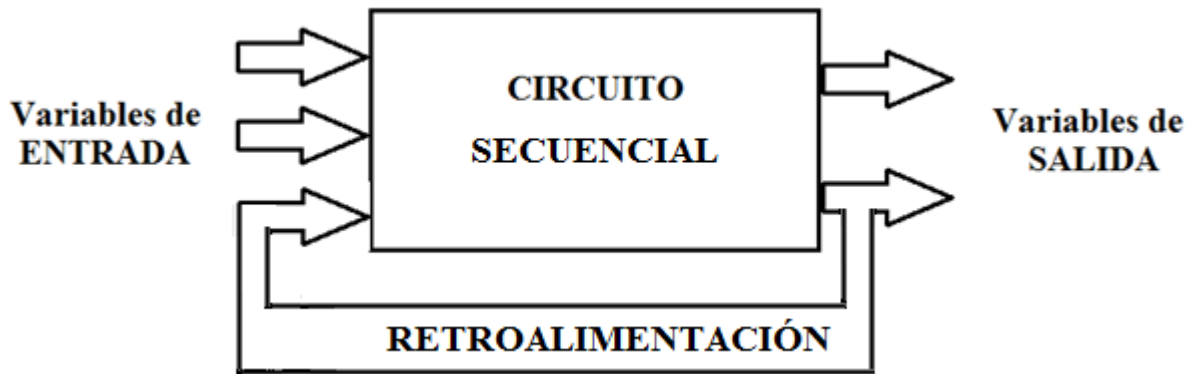
Sin embargo, existen otro tipo de circuitos, denominados CIRCUITOS SECUENCIALES en los cuales sus variables de Salida dependen de las variables de Entrada y también de la SECUENCIA de valores que fueron tomando las variables del sistema. Es decir, en este tipo de circuitos las variables de salida se convierten en variables de entrada en el instante siguiente. Este proceso se denomina “RETROALIMENTACIÓN” (feedback) y permite “recordar” el valor anterior de una variable, es decir MEMORIA!!!



Como se explicó en apartados anteriores, un CIRCUITO COMBINACIONAL, es aquel en el cual sus variables de SALIDA se obtienen exclusivamente como combinación de sus variables de ENTRADA.

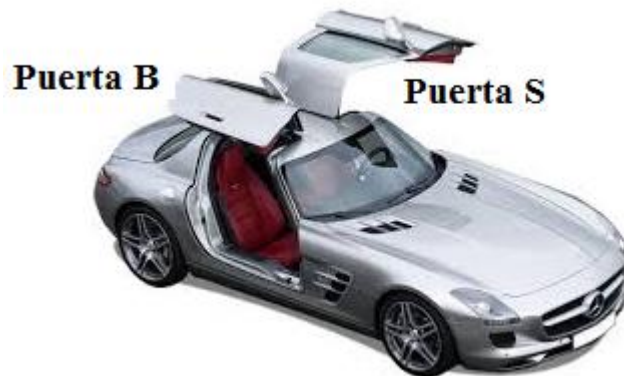


Sin embargo, existen otro tipo de circuitos, denominados CIRCUITOS SECUENCIALES en los cuales sus variables de Salida dependen de las variables de Entrada y también de la SECUENCIA de valores que fueron tomando las variables del sistema. Es decir, en este tipo de circuitos las variables de salida se convierten en variables de entrada en el instante siguiente. Este proceso se denomina “RETROALIMENTACIÓN” y permite “recordar” el valor anterior de una variable, es decir MEMORIA!!!



Analicemos estos conceptos desde un ejemplo sencillo<sup>28</sup>.

Un taller de Alarmas para Autos recibe el pedido de un cliente que desea instalar una alarma, Q, en un auto de 2 puertas, llamadas S y B.



Si alguna de las dos puertas, inclusive las dos al mismo tiempo, es abierta por un ladrón, la alarma debe sonar. Aprovechando los interruptores dispuestos para el encendido de la luz de cortesía, se puede realizar un sistema que haga sonar la alarma cuando se abra alguna puerta, o ambas.

Esto corresponde a la función lógica OR, de tantas variables de entrada como aperturas tenga el vehículo. Por ejemplo para dos variables de entrada “S” y “B” y suponiendo la convención: puerta cerrada “0”, puerta abierta “1”, alarma sonando “1”, quedaría una tabla como la siguiente:

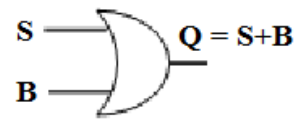
La situación se plantea en la tabla:

S Puerta ABIERTA	B Puerta ABIERTA	Q Alarma Suena
NO	NO	NO
NO	SI	SI
SI	NO	SI
SI	SI	SI

Plasmada en la siguiente Tabla de Verdad:

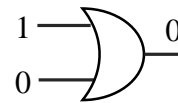
S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Que corresponde a la suma lógica (compuerta lógica OR).



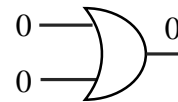
Ahora bien, al abrir la puerta, se activa la alarma:

S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



y el intruso podría salir corriendo cuando escuche el ruido... O cerrar la puerta:

S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



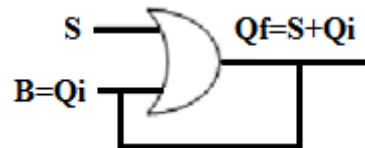
El inconveniente es que si la/s puerta/s se cierra/n, la alarma deja de sonar, permitiendo que el ladrón se robe el auto con toda comodidad.

El cliente queda muy disconforme...

Se recibe un nuevo cliente que solicita instalar una alarma para su auto de una sola puerta S.

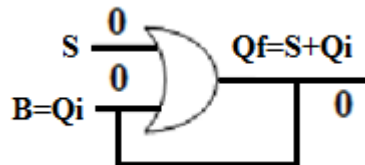


Como la instalación fallida del cliente anterior, dejó su enseñanza, se decidió mejorar el circuito. Por este motivo, se decidió que el sistema registre la maniobra y mantenga activa su salida (la alarma sonando) permanentemente, a partir de la primera apertura. El efecto deseado puede lograrse conectando la salida de la OR a una de sus entradas. A esta conexión se la denomina “realimentación”. En este caso, empleamos la entrada “B”. De este modo, cuando la salida tome el valor lógico “1”, la entrada adoptará ese mismo valor, manteniendo la salida en “1”. B y Q corresponden ahora a la misma conexión; podríamos decir simplemente Q (puesto que B = Q).



Analizamos el sistema a partir del estado inicial, en que la alarma se encuentra desactivada y la puerta cerrada (todo en “0”).

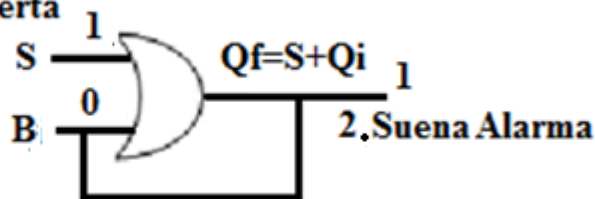
Si la puerta está cerrada ( $S=0$ ) y  $B=0$ , la alarma NO SUENA  $Q_f=0$ .



Al abrir la puerta, la entrada correspondiente (“S”) se pondrá en “1” provocando el cambio de la salida a “1” ( la alarma suena).

S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

**1. Puerta Abierta**



B y Q difieren momentáneamente durante este estado de transición. Inicialmente “B = 0” (que a partir de ahora lo llamaremos estado inicial  $Q_i$ ), luego “1” el estado final ( $Q_f$ ).

Al principio  $Q_i$  está en “0” y rápidamente cambia a “1”  $Q_f$ .

Cuando en S se coloca “1”, también transcurre un breve lapso (tiempo de demora) hasta que la salida Q adopta ese estado.

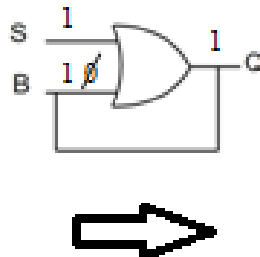
B y Q difieren momentáneamente durante este estado de transición. Inicialmente “B = 0” (que a partir de ahora lo llamaremos estado inicial  $Q_i$ ), luego “1” el estado final ( $Q_f$ ).

Al principio  $Q_i$  está en “0” y rápidamente cambia a “1”  $Q_f$ .

Cuando en S se coloca “1”, también transcurre un breve lapso (tiempo de demora) hasta que la salida Q adopta ese estado.

Este cambio de estado en la salida se reflejará inmediatamente a la entrada, pasando al estado resaltado en las tablas siguientes:

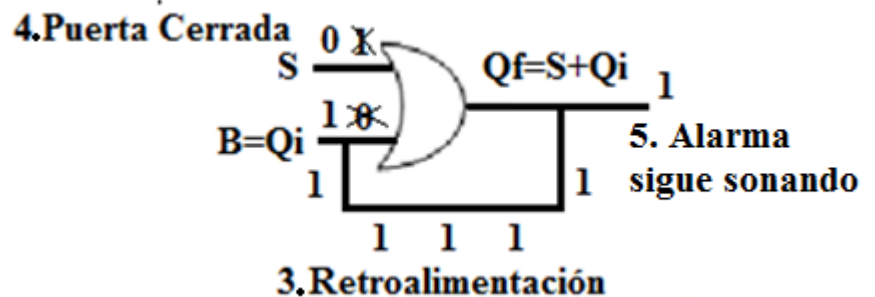
S	B ( $Q_i$ )	$Q_f$
0	0	0
0	1	1
1	0	1
1	1	1



S	B ( $Q_i$ )	$Q_f$
0	0	0
0	1	1
1	0	1
1	1	1

Pero si se realiza (3.) Retroalimentación y la puerta se cierra (4.)  $S=0$ , la alarma sigue sonando (5.).

S	B ( Q i )	Q f
0	0	0
0	1	1
1	0	1
1	1	1



La situación se plantea en las siguientes tablas:

Puerta S	B=Qi Estado Anterior RETROALIMENTACIÓN	Qf Estado Actual Alarma Suena
ABIERTA	NO	SI
CERRADA	SI	SIGUE SONANDO AUNQUE SE CIERRE LA PUERTA

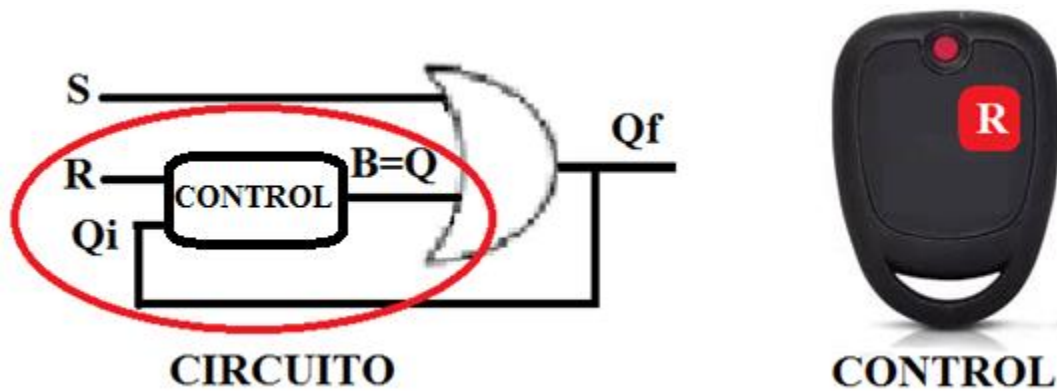
Puerta S	B=Qi RETROALIMENTACIÓN	Qf ALARMA
4. CERRADA	3. SE REALIMENTA	5. SUENA
1. ABIERTA	0	2. SUENA

... la salida seguirá en estado “1” pues la otra entrada ahora tiene un “1” ( $Q=1$ ). Éste es, precisamente, el concepto de *memoria*.

El sistema descrito proporciona una solución al problema de encender y mantener la alarma, pero tiene el inconveniente de que una vez activada, seguirá sonando indefinidamente  $Q_f=1$ . Para desactivarla, se podría agregar un circuito lógico en la realimentación, de manera de poder volver al estado inicial. Esto es, poner un cero en el punto indicado como “ $B=Q$ ” a la entrada de la compuerta OR. Podemos definir el funcionamiento de ese circuito lógico en la realimentación, diciendo que: a) debe presentar un “0” en los casos en que el usuario desee reiniciar el sistema ( $R=1$ , Reset) o b) cuando sin resetear ( $R=0$ ) la alarma no esté activada ( $Q=0$ ). Su tabla es la siguiente:

Para solucionarlo, se implementó un control, para desactivar la alarma por medio de un botón.





Ahora vamos a centrarnos en diseñar el CONTROL.

El botón R del CONTROL DEBE apagar la alarma si es que está sonando.

Es decir, al pulsar (poner en 1) el botón R, la alarma Qi que estaba sonando ( $Qi=1$ ), debe apagarse ( $Qf=0$ ).

R (Botón del Control)		Qi (Estado anterior de la alarma)		B (Estado actual de la alarma)	
0	NO PULSADO	0	NO SUENA	0	NO SUENA (mantiene el estado anterior)
0	NO PULSADO	1	SUENA	1	SUENA (mantiene el estado anterior)
1	PULSADO	0	NO SUENA	0	NO SUENA (sigue sin sonar)
1	PULSADO	1	SUENA	0	<b>ESTABA SONANDO Y SE APAGÓ</b>

Analizando la Tabla de Verdad resultante, se puede formular que el control, responde a la siguiente función, que posee un único minitérmino:

$$\text{Control (R, Qi)} = \overline{R} \cdot Qi$$

Que puede expresarse de la siguiente forma, utilizando doble negación:

$$\text{Control (R, Qi)} = \overline{\overline{R}} \cdot Qi$$

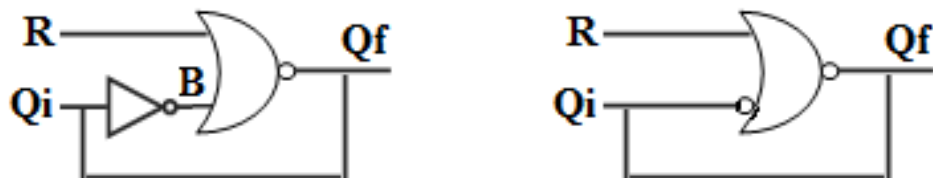
A continuación aplicamos el Teorema de De Morgan y la función queda expresada así:

$$\text{Control (R, Qi)} = \overline{\overline{R}} + Qi$$

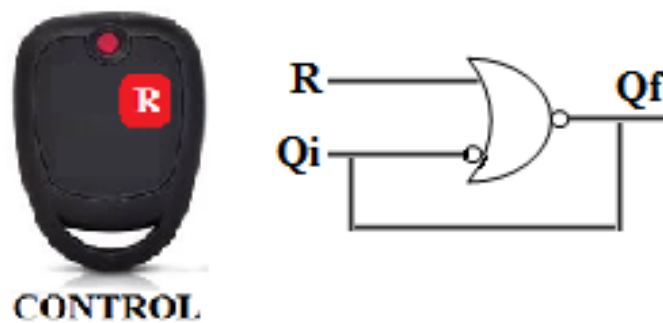
Y aplicando el Teorema de Doble Negación, resulta la siguiente expresión:

$$\text{Control (R, Qi)} = \overline{R + \overline{Qi}}$$

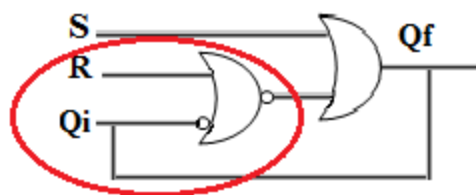
Que corresponde al siguiente circuito.



Por lo tanto se ha diseñado el circuito correspondiente al CONTROL.

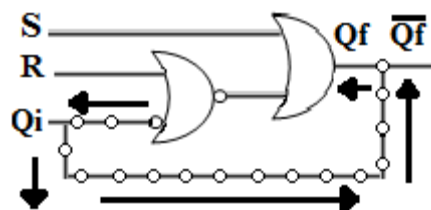


A continuación se ubicará dentro del circuito inicial.

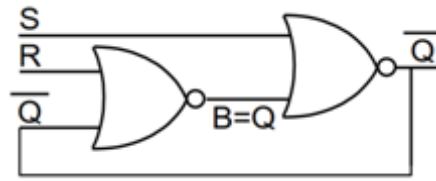


Se puede observar que Qi ingresa negada a la primera compuerta OR.

Corramos hacia atrás la compuerta NOT.



Quedando el siguiente circuito:



A continuación se muestra la tabla de verdad completa. En ella se presentan las dos variables de entrada al sistema R y S, la variable de estado interno B (Q) y la variable de salida Q.

Se observa en el esquema una salida  $\bar{Q}$ . La salida original Q se obtiene del punto B = Q. Los estados indicados con “X” (estado lógico no relevante), corresponden a la condición en que S y R valgan “1”, lo cual representa el absurdo de querer apagar la alarma cuando intentan robar el automóvil.

S	R	B ( Q i )	Q f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

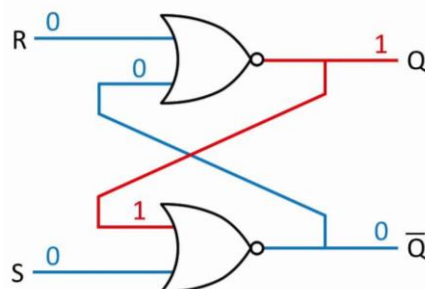
En esta tabla se presenta el caso de que pueden existir estados no definidos. En general, si una función presenta estados no definidos y se puede elegir libremente el estado lógico resultante, diremos que se tratará de una “función incompleta”.

Analicemos brevemente la tabla de verdad. En las dos primeras combinaciones (filas), el estado final de la salida Q f coincide con el estado anterior (inicial) Q i. Diremos que el circuito *retiene* (almacena o memoriza) el estado binario que tenía, sin alterarlo.

En las dos combinaciones siguientes, el estado final de la salida Q f es cero, independientemente del estado anterior Q i. Esta es la operación de Reset. Esto se logra gracias a que la entrada R se encuentra en uno y la S en cero. Las siguientes dos, Q f está en uno, sin importar el estado en el que se hallaba antes. En este caso lo llamamos Set. Para “setear” (poner en uno) la salida del circuito, debemos asegurar un uno en la entrada Set y un cero en el Reset.

Por ello, al circuito secuencial así formado se lo conoce como *biestable RS*, reset - set, ya que posee dos estados estables. También se lo llama flip flop RS. En inglés se los reconoce como *latch*, que significa *cerrojo*, porque es capaz de retener un estado. Por definición, los flip flops poseen dos salidas: Q y  $\bar{Q}$  que se obtuvieron aquí, de la salida de cada una de las NOR.

En la mayoría de los textos, la forma en la cual se lo encuentra dibujado es la siguiente:



El último par de combinaciones se prohíben, ya que además de lo expuesto en párrafos anteriores, en este caso impiden el cumplimiento de la definición del flip flop e incluso pueden producir efectos no deseados en su funcionamiento.

En el diseño por métodos tabulares, se coloca en los estados no definidos una letra “X”. Así el diseñador podrá elegir entre “0” y “1”, resultando un circuito más simple. En las siguientes figuras, se muestran las tablas de Karnaugh para la simplificación.

S \ R Q <sub>i</sub>	00		01	11	10
	0	1	2	3	4
0	0	1	0	0	
1	1	1	X	X	

S \ R Q <sub>i</sub>	00		01	11	10
	0	1	2	3	4
0	0	1	0	0	
1	1	1	X	X	

S \ R Q <sub>i</sub>	00		01	11	10
	0	1	2	3	4
0	0	1	0	0	
1	1	1	X	X	

En la segunda tabla, para simplificar por medio de maxitérminos, se ha supuesto que las “X” adoptan el valor cero. La expresión simplificada es:  $Q_f = (S + Q_i) \cdot \bar{R}$

Aplicando el teorema de doble negación y luego De Morgan:  $Q_f = \overline{\overline{(S + Q_i) \cdot \bar{R}}}$

$$Q_f = \overline{\overline{(S + Q_i)} + \bar{R}} \Rightarrow Q_f = \overline{\overline{(S + Q_i)} + R}$$

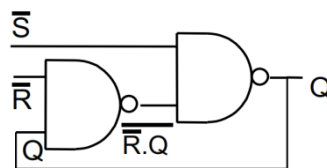
Con todas compuertas NOR. Esta última expresión coincide con el último circuito de la página anterior.

En la tercera tabla en cambio, para simplificar por medio de minitérminos, se ha supuesto que las “X” adoptan el valor uno. La expresión simplificada es:  $Q_f = S + \bar{R} \cdot Q_i$

Aplicando el teorema de doble negación y luego De Morgan:  $Q_f = \overline{\overline{S + \bar{R} \cdot Q_i}}$

$$Q_f = \overline{\bar{S} \cdot \bar{\bar{R} \cdot Q_i}}$$

Con todas compuertas NAND. Esta última expresión tiene el siguiente circuito:



Se observa que las entradas Set y Reset están invertidas. Esto implica que para poner en uno la salida, el  $\bar{S}$  debe estar en cero y  $\bar{R}$  en uno. Para poner en cero la salida,  $\bar{R}$  debe estar en cero y  $\bar{S}$  en uno. Si ambas entradas  $\bar{S}$  y  $\bar{R}$  están en uno, el sistema memoriza (mantiene la salida en su estado anterior). Si ambas estuvieran en cero, se dará el estado prohibido.

Después de este aprendizaje, resultará sintético y esclarecedor realizar la siguiente tabla de verdad reducida de un flip flop R S, donde se muestran las cuatro combinaciones posibles de las variables de entrada R y S y la función de salida final  $Q_f$ .

S	R	$Q_f$
0	0	$Q_i$

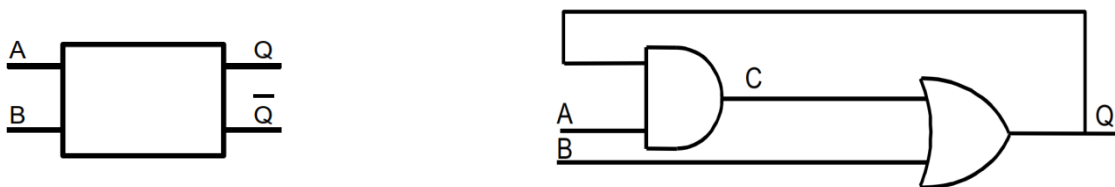
0	1	0
1	0	1
1	1	X

Observe que cuando S y R valen “0”, la salida  $Q_f$  mantiene el valor anterior de Q ( $Q_i$ ). Cuando  $S=1$  la salida pasa a “1” y cuando  $R=1$  la salida será “0”. Esto concuerda con la tabla completa. También queda claro el estado prohibido correspondiente a S y R simultáneamente en 1.

A esta altura de nuestro estudio, podremos reconocer las expresiones algebraicas y explicar el funcionamiento de otros circuitos posibles para un flip flop R S asincrónico<sup>29</sup>. A continuación, se propone un método de análisis.

Consideraremos las características del flip flop R S mediante las cuales: a) se logra poner en uno a la salida con una cierta combinación de las entradas; b) otra combinación logrará poner en cero la salida y c) una tercera combinación mantiene el estado que tenía (memoria de un bit).

Presentado el circuito, se deduce la expresión algebraica. Luego, por medio de las características mencionadas más arriba (o empleando la tabla de verdad característica del flip flop R S) buscamos reconocer las entradas Set y Reset. En general, veremos el esquema como un sistema con entradas y salidas:



Recorremos de izquierda a derecha el circuito y encontramos  $C = A \cdot Q$ . Continuamos hasta la salida,  $Q = B + C$ . Sustituyendo el valor encontrado para C, en la expresión de Q obtenemos:  $Q = B + (A \cdot Q)$ . Ahora bien, como sabemos, la variable de salida realimentada implica que en un momento inicial tendrá un cierto valor y luego un valor final. Ya en párrafos precedentes los hemos indicado con subíndices “i” y “f” respectivamente. Por lo tanto resultará:

$Q_f = B + (A \cdot Q_i)$ , que es la expresión algebraica que representa a este esquema.

Estamos en condiciones de reconocer la función de cada entrada, A y B.

Observamos que A está conectada a una compuerta AND (producto lógico) y sabemos por los postulados del Álgebra de Boole (más precisamente el “5b”) que el elemento neutro del producto es 1. La entrada B se encuentra conectada a una compuerta OR (suma lógica) y el elemento neutro de la suma es 0.

Un 1 en B pondrá en 1 a la salida, independientemente del estado del resto de las variables.

En resumen:  $B = 1 \Rightarrow Q = 1$ . Esto corresponde al Set. Podemos entonces hacer la equivalencia  $B = S$ .

Si B está en 0, la salida depende del producto  $A \cdot Q_i$ , pues:  $0 + A \cdot Q_i = A \cdot Q_i$

Si al mismo tiempo, A se encuentra en 1 (neutro del producto) resultará  $1 \cdot Q_i = Q_i$

Es decir, se conserva el estado anterior (memoria de un bit), con  $B = 0$  y  $A = 1$ .

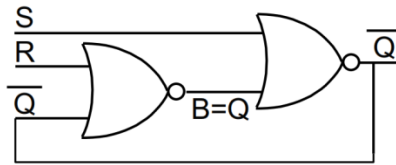
En cambio, si  $B = 0$  (neutro de la suma) y  $A = 0$ , la salida será cero  $Q_f = 0 + 0 \cdot Q_i = 0$

Sintéticamente:  $B = 0$  y  $A = 0 \Rightarrow Q = 0$ . Esto corresponde al Reset.

<sup>29</sup> Los circuitos secuenciales pueden ser síncronos o asíncronos. Los circuitos síncronos necesitan una entrada adicional (de clock o reloj) para sincronizar su funcionamiento.

Podemos entonces hacer la equivalencia  $A = R$ , ya que la operación de Reset se logra con  $A = 0$  lo cual corresponde al caso general  $R = 1$ .

Retomando el concepto del biestable RS (Flip Flop RS):



$S$	$R$	$Q_i$	$Q_f$	
0	0	0	0	Memoria $Q_f = Q_i$
0	0	1	1	
0	1	0	0	Reset $Q_f = 0$
0	1	1	0	
1	0	0	1	Set $Q_f = 1$
1	0	1	1	
1	1	0	X	Estados prohibidos
1	1	1	X	

En las dos primeras combinaciones, el estado final de la salida  $Q_f$  coincide con el estado anterior (inicial)  $Q_i$ . Diremos que el circuito retiene (almacena o memoriza) el estado binario que tenía, sin alterarlo.  $S$  y  $R$  en cero.

En las dos combinaciones siguientes, el estado final de la salida  $Q_f$  es cero independientemente del estado anterior  $Q_i$ . Esta es la operación de Reset. Esto se logra gracias a que la entrada  $R$  se encuentra en uno y un cero en  $S$ .

Las siguientes dos,  $Q_f$  está en uno, sin importar el estado en el que se hallaba antes.

En este caso lo llamamos Set. Para “setear” (poner en uno) la salida del circuito, debemos asegurar un uno en la entrada  $S$  y un cero en el  $R$ . Por ello, al circuito secuencial así formado se lo conoce como biestable  $R S$ , reset – set, ya que su estado posee dos condiciones estables. También se lo llama flip flop  $R S$ .

Por definición: los flip flop, poseen dos salidas:  $Q$  y  $\overline{Q}$  que se obtuvieron aquí, de la salida de cada una de las compuertas NOR. El último par de combinaciones, con  $S$  y  $R$  en uno, se prohíben ya que en este caso impiden el cumplimiento de la definición del flip flop e incluso pueden producir efectos no deseados en su funcionamiento.

**Índice**

1. Introducción	1
2. Fundamentos de la lógica Conmutacional	2
2.1. Parámetros eléctricos fundamentales	3
3. Lógica proposicional	5
3.1. Operaciones Proposicionales	7
3.2. Proposiciones Equivalentes	10
4. Álgebra conmutacional y compuertas lógicas básicas	10
4.1. Compuerta NOT	11
4.2. Compuerta AND	12
4.3. Compuerta OR	13
5. Comparación entre enfoques lógicos	16
6. Aplicaciones de compuertas lógicas	17
7. Álgebra de Boole	18
7.1. Postulados del Álgebra de Boole o de Huntington	18
7.2. Teoremas del Álgebra de Boole	19
7.3. Funciones de un Álgebra de Boole	21
7.4. Términos canónicos y expresiones canónicas	22
7.4.1. Minitérminos o Suma de productos	22
7.4.2. Maxitérminos o Producto de sumas	24
7.5. Simplificación de funciones lógicas	27
7.5.1 Simplificación algebraica por Postulados y Teoremas	27
7.5.2 Simplificación por el Método de Karnaugh	28
• Diagrama de Karnaugh para funciones de tres variables	31
• Diagrama de Karnaugh para funciones de cuatro variables	32
• Funciones simplificadas como producto de sumas	34
• Ejemplos	35
• Simplificación de funciones con Logisim	38
8. Implementación de funciones simples: Circuitos combinacionales	41
8.1. Introducción	41
8.1.1 Compuertas NAND y NOR	41
8.1.2 Compuerta XOR (Exclusive OR u “O Exclusiva”)	42
8.2. Circuitos multifunción	44
8.2.1. Decodificadores	44
8.2.2. Suma aritmética	45
8.2.3. Comparadores	48
8.2.4. Generador (o detector) de bit de paridad	49
8.2.5. Multiplexores (MUX)	49
9. Circuitos secuenciales	50