TECANA AMERICAN UNIVERSITY

Bachelor of Science in Computer Science

INFORME II

Teoría de conmutación de circuitos y diseño lógico

Por la presente, doy fe que soy la única autora de esta investigación, y que su contenido es consecuencia de mi trabajo académico.

Eugenia Bahit

Buenos Aires, 22 de Diciembre de 2020

ÍNDICE DE CAPÍTULOS

CAPÍTULOS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

INTRODUCCIÓN Y OBJETIVOS

La manipulación de la información en un ordenador es posible gracias a operaciones realizadas a nivel de hardware, mediante unos bloques denominados *puertas lógicas.* Estos, consisten en unos circuitos electrónicos que producen señales binarias y las procesan mediante operaciones lógicas, que responden a expresiones algebraicas cuya base se encuentra en un tipo de álgebra concreta denominada *álgebra booleana*. La simplificación de esta es lo que facilita el *diseño* de los *circuitos digitales* sobre los que se construye un ordenador. Todos estos aspectos son abarcados en la *Teoría de Conmutación de Circuitos y Diseño Lógico*, que se pretende explicar a lo largo de este trabajo.

El objetivo general es comprender qué son, en términos concretos, las operaciones que se ejecutan en los circuitos electrónicos y cómo estas se llevan a cabo. Para ello, se agrupan los temas que componen la teoría de conmutación, en tres capítulos: *Álgebra booleana*, cuyo objetivo pretende responder a cuáles son las bases teóricas sobre las que dichas operaciones se fundamentan; *Puertas lógicas y diseño digital*, que pretende explicar los mecanismos de aplicación del álgebra booleana, empleados para producir resultados concretos; y finalmente, *circuitos lógico digitales*, donde se pretende dar respuesta a cómo el álgebra booleana y el diseño digital se interconectan, para finalmente producir los circuitos digitales que llevan a cabo el total de las operaciones en un ordenador.

Observaciones: excepto donde se indique lo contrario, todo el contenido de este trabajo, incluidas las tablas y gráficos, son de elaboración propia.

CAPÍTULO I

ÁLGEBRA BOOLEANA

El *álgebra booleana* es un tipo de álgebra diseñada para manejar variables binarias y operaciones lógicas. Su conocimiento es la base para el tratamiento de la información al nivel más bajo posible. Su estudio permite comprender el nexo entre la física —a través de los componentes electrónicos que la implementan— y los sistemas informáticos. En este capítulo se hace un repaso exhaustivo sobre las bases matemáticas teóricas del álgebra booleana para llegar a su aplicación en el diseño lógico, a fin de permitir así una comprensión más concreta de los temas que serán abarcados en el siguiente capítulo.

Álgebra booleana como teoría matemática

Desde la perspectiva de su estructura abstracta, puede definirse el álgebra boolena de un conjunto B no vacío formado por dos operaciones binarias + y *, una operación unaria $'$, y dos elementos distintivos 0 y 1, si para cualquier variable $a,b,c \in B$ se cumplen todas las leyes descritas a continuación (Huntington, 1904).

Leyes del álgebra booleana

LEY CONMUTATIVA. Las operaciones + y * son conmutativas:

$$a+b=b+a \qquad a*b=b*a$$

LEY DE IDENTIDAD. 0 y 1 son dos identidades para las operaciones + y * respectivamente:

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

LEY DISTRIBUTIVA. Cada operación binaria es distributiva sobre la otra:

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

LEY DE COMPLEMENTO. Para cada $a \in B$ existe un elemento $a' \in B$:

$a+a'=1 \qquad a*a'=0$

Teoremas del álgebra booleana

De todo lo anterior, se obtienen además, los siguientes teoremas:

TEOREMA 1. Principio de dualidad. Cualquier teorema del álgebra booleana es válido si + se intercambia con * y 0 con 1 a lo largo del teorema.

TEOREMA 2. Las siguientes leyes se sostienen en un álgebra booleana B :

Idempotencia:

$\forall a \in B \quad a+a=a \ , \quad a*a=a$

Delimitación:

$\forall a \in B \quad a+1=1 \ , \quad a*0=0$

Asociación:

$\forall a,b,c \in B \quad (a+b)+c=a+(b+c) \ , \quad (a*b)*c=a*(b*c)$

Absorción:

$\forall a,b \in B \quad a+(a*b)=a \ , \quad a*(a+b)=a$

TEOREMA 3. El complemento de un elemento es único.

TEOREMA 4. El complemento del complemento de un elemento es igual al elemento: $(a')'=a$

TEOREMA 5. El complemento de 0 es 1 y el de 1 es 0: $0'=1 \quad 1'=0$

TEOREMA 6. Ley de Morgan. Muestra que para cada par de elementos a y b en un álgebra booleana B se cumple $(a+b)'=a'*b' \quad (a*b)'=a'+b'$.

TEOREMA 7. Las siguientes operaciones son equivalentes:

$(1)a+b=b, \quad (2)a*b=a, \quad (3)a'+b=1, \quad (4)a*b'=0$

Axiomas del álgebra booleana

Se establecen así los siguientes **axiomas** (Tabla 1):

Tabla 1: *Axiomas del álgebra booleana*

Operación:	*(AND)	+ (OR)	' (INVERT)
Axiomas:	0 * 0 = 0 0 * 1 = 0 1 * 0 = 0 1 * 1 = 1	0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 1	1′ = 0 0′ = 1

Funciones booleanas

En su forma matemática primitiva, las expresiones booleanas se representan por medio de funciones. Se define una ***función booleana*** o ***polinomio booleano*** como una expresión booleana de n variables, para la cual, siendo x una variable booleana, una función booleana de n variables, queda denotada por $f(x_1, x_2, \ldots, x_n)$. Considerando que una función booleana f es el resultado de una expresión booleana de n variables $x_1, x_2, \ldots, x_n$; y que el resultado de una expresión booleana, puede asumir como valor un 0 o un 1 ; es posible

afirmar por medio de la deducción, que *una función booleana es otra variable cuyo valor puede asumir un 0 o un 1*.

Álgebra booleana aplicada al diseño de circuitos

Las expresiones booleanas representan operaciones booleanas, y a cada operación lógica le corresponde un elemento concreto a nivel de hardware, por lo que cuanto mayor sea la complejidad de una expresión, mayor será la complejidad de los circuitos, y por lo tanto, mayores serán los recursos necesarios para producirlos (mayor tamaño, mayor cantidad de materiales, mayor costo). Por este motivo, el foco de la aplicación del álgebra booleana en el diseño lógico de circuitos, se encuentra en la simplificación de las expresiones algebraicas.

Reducción de expresiones booleanas

A fin de reducir la complejidad de las expresiones algebraicas, se requiere un procedimiento de cinco pasos, mediante el cual se deben aplicar todas las leyes del álgebra booleana. Dicho procedimiento consiste en:

1. Multiplicar todas las variables a fin de remover paréntesis.
2. Eliminar términos duplicados. Ej.: $ABC \cdot ABC = ABC$.
3. Eliminar todo término compuesto de una variable y su negación. Ej.: $ABB'C+D = AC+D$.

4. Cuando haya términos que solo se diferencien por una variable, eliminar el más largo de ellos. Ej.: $A\overline{B}C+A\overline{B} = A\overline{B}$ [1].

5. Si dos términos son iguales y una (o más) variables se encuentra afirmada en un término y negada en otro, eliminar las variables complementadas de ambos términos. Ej.: $A\overline{B}C+ABC = AC$

Componentes de una expresión booleana

Una expresión booleana se compone de *términos* y operadores booleanos. Según puede concluirse a partir de lo estudiado en la diversa bibliografía, podrían obtenerse dos términos primitivos:

1. *Término producto:* aquel cuyas variables son el resultado de una operación AND. Por ejemplo, dadas dos variables A, B cualesquiera, un término producto podría ser AB .

2. *Término suma:* aquel cuyas variables son el resultado de una operación OR. Por ejemplo, dadas dos variables A, B cualesquiera, un término suma sería $A+B$.

En los términos, las variables pueden aparecer complementadas o no complementadas, indistintamente, pero solo deben aparecer una vez, ya que previamente, las expresiones debieron ser reducidas.

---

1 Se emplea la notación ABC y $\bar{A}$ en lugar de x_1, x_2, x_3 y x' para representar variables y sus complementos, puesto que en la bibliografía se prefiere la primera notación en el contexto de la teoría informática mientras que la segunda, en un contexto de teoría matemática.

Cuando un término contiene el total de las variables de una función, se los denomina *minitérmino* o *maxitérmino*, según si las variables responden a una operación AND u OR respectivamente. Cada uno de estos términos, son también conocidos matemáticamente como *polinomio mínimo* y *polinomio máximo*, y se definen como sigue:

1. *minitérmino:* aquel término producto compuesto por el total de las variables de una función. Por ejemplo, dadas tres variables A, B, C cualesquiera, un *minitérmino* podría ser ABC. En un *minitérmino*, cada variable no complementada asume 1 como valor, y 0 en caso contrario. Un *minitérmino* es denotado por m_i donde i es el valor decimal correspondiente a la combinación binaria del término obtenida.

2. *maxitérmino:* aquel término suma compuesto por el total de las variables de una función. Por ejemplo, dadas tres variables A, B, C cualesquiera, un *maxitérmino* podría ser $A+B+C$. En un *maxitérmino*, cada variable no complementada asume 0 como valor, y 1 en caso contrario. Un *maxitérmino* es denotado por M_i donde i, al igual que en el caso anterior, es el valor decimal correspondiente a la combinación binaria del término obtenida.

Así, a partir de una expresión $AB\bar{C}$ se obtiene la combinación binaria 110, cuyo valor decimal es 6. Por lo tanto, la expresión $AB\bar{C}$ podría ser denotada como m_6, y $A+B+\bar{C}$ como M_1.

Dado que se trata de un álgebra booleana (la cual asume 1 de dos valores posibles), es correcto enunciar el siguiente teorema:

TEOREMA: Para n variables $x_{1,}x_{2,}...x_n$, existen 2^n términos posibles.

De esta forma, para dos variables A,B cualesquiera, podrían obtenerse los cuatro polinomios mínimos AB , $A\overline{B}$, $\bar{A}B$ y $\bar{A}\bar{B}$, o máximos.

Formas de simplificación de una función booleana

La simplificación de funciones booleanas es una forma de representar las mismas de manera tal que sea posible determinar si la reducción realizada sobre la expresión, es o no la reducción máxima posible.

Existen diferentes formas de representar una función booleana, entre las cuales pueden encontrarse las siguientes:

1. Formas normales disyuntiva y conjuntiva.
2. Formas canónicas disyuntiva y conjuntiva.
3. Tablas de verdad.
4. Diagramas de Venn.
5. Mapas de Karnaugh.

Estas formas se explican a continuación.

FORMA NORMAL DISYUNTIVA. Suma de productos (SOP). Es una función cuyos términos son la suma de términos producto. Por ejemplo,

$f(A,B,C)=AB+\bar{C}B$ (en el ejemplo AB y $\bar{C}B$ son dos términos producto que se suman).

FORMA CANÓNICA DISYUNTIVA. Toda vez que los términos producto de una forma disyuntiva sean *minitérminos*, se considera una forma canónica y puede simplificarse reemplazando los términos producto por el *minitérmino* correspondiente. De esta forma, $f(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}$ podría ser expresado como $f(A,B,C)=m_1+m_2$ o por medio de un listado de los decimales correspondientes, con la siguiente expresión: $f(A,B,C)=\sum(1,2)$.

FORMA NORMAL CONJUNTIVA. Producto de sumas (POS). Es una función cuyos términos son la multiplicación de términos suma. Por ejemplo, $f(A,B,C)=(A+B)(\bar{C}+B)$ (en el ejemplo, $A+B$ y $\bar{C}+B$ son dos términos suma que se multiplican).

FORMA CANÓNICA CONJUNTIVA. Toda vez que los términos suma de una forma conjuntiva sean *maxitérminos*, se considera una forma canónica y puede simplificarse reemplazando los términos suma por el *maxitérmino* correspondiente. De esta forma, $f(A,B,C)=(A+B+C)(\bar{A}+B+\bar{C})$ puede ser expresado como $f(A,B,C)=M_0 \cdot M_5$ [2] o por medio de un listado de sus decimales con la siguiente expresión: $f(A,B,C)=\prod(0,5)$.

Se debe notar que en el proceso de simplificación de las formas canónicas, el procedimiento se inicia con la reorganización de los términos a fin de obtener los

2 Se utiliza la notación de punto medio como reemplazo del asterisco para diferenciar la notación matemática de la empleada en teoría de conmutación de circuitos.

decimales como ordinales ascendentes. De esta forma, si la expresión original fuese $\bar{A}B\bar{C}+\bar{A}\bar{B}C$ se reorganizaría como $\bar{A}\bar{B}C+\bar{A}B\bar{C}$ para evitar obtener m_2+m_1 .

TABLAS DE VERDAD. Una tabla de verdad recoge todas las combinaciones de valores posibles para las variables de una función.

Tabla 2: *Tabla de verdad para la función* $f(A,B,C)=AB+\bar{C}B$

A	B	C	$\bar{C}$	AB	$\bar{C}B$	$AB+\bar{C}B$
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	1	0	1
Variables de la función				*Expresiones individuales*		*Función*

DIAGRAMAS DE VENN. Para representar una función booleana en un diagrama de Venn, se considera cada variable como un conjunto, donde la operación AND es una intersección, OR una unión, y el complemento (INVERT) es todo lo que no está dentro del conjunto.

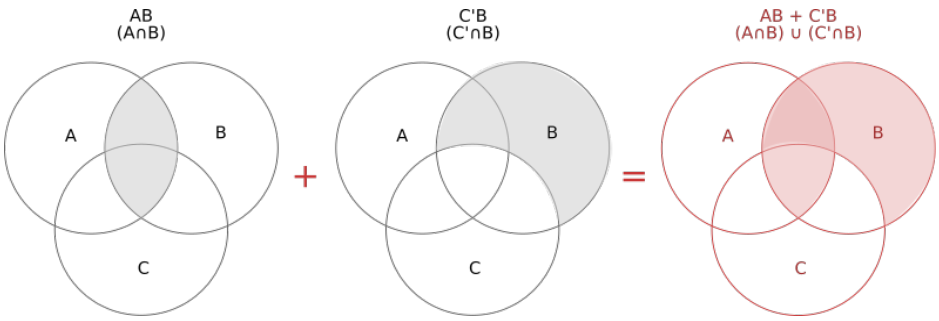


Gráfico 1: *Diagram de Venn para la función* $f(A,B,C)=AB+\bar{C}B$

MAPAS DE KARNAUGH. Los mapas de Karnaugh (o K-map) son una forma de comprimir las tablas de verdad, y tienen por objetivo llevar las mismas a la mínima cantidad de términos necesarios para representar una función. En estos diagramas, los *minitérminos* se representan en cuadrados, por lo que siempre habrá 2^n cuadrados, para n = cantidad de variables. De esta forma, los diagramas para 2, 3 y 4 variables tendrían $2^2=4$, $2^3=8$ y $2^4=16$ cuadrados respectivamente, como puede verse a continuación:

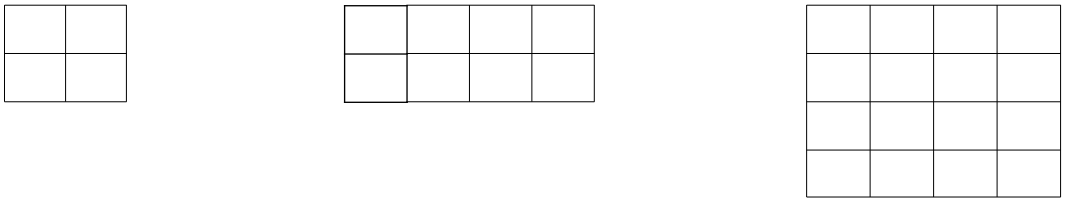


Gráfico 2: *Cuadrados del diagrama de Karnaugh para 2, 3 y 4 variables*

Los nombres de las variables se van ubicando en la esquina izquierda y agrupando con el siguiente patrón:

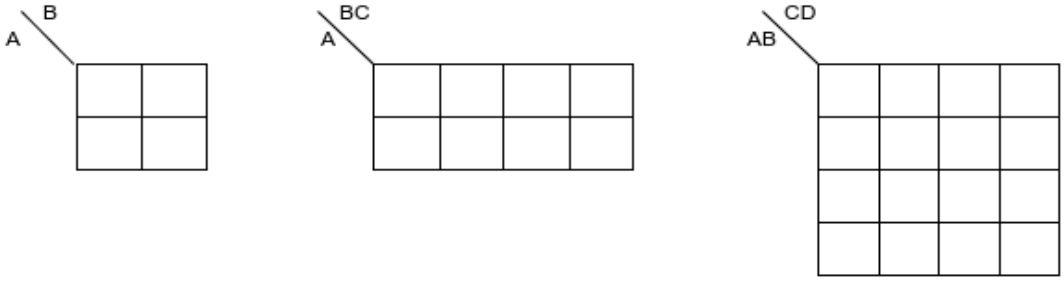


Gráfico 3: *Distribución de variables en diagramas de Karnaugh*

En los bordes del diagrama, sobre cada cuadrado se coloca el valor binario o combinación de valores binarios para las variables, de forma tal que los valores adyacentes solo difieran por 1. Por ejemplo, para dos variables CD, los dos

primeros valores serían 00 (C=0, y D=0), 01 (C se mantiene y solo D cambia), pero no podría ser 00, 11, ya que no solo C cambiaría sino también D. La diferencia siempre debe ser de un único valor:

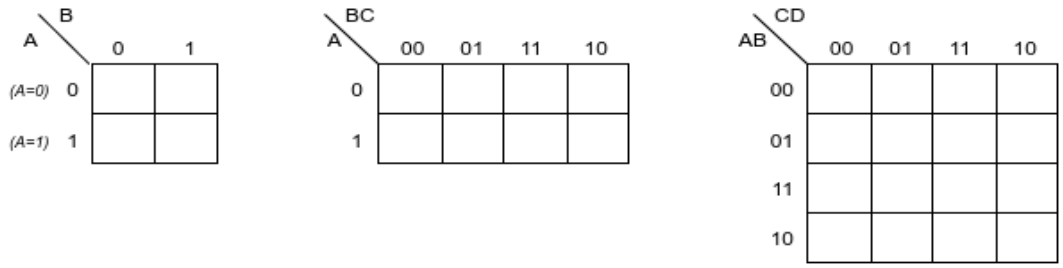


Gráfico 4: *Disposición de valores binarios en los diagramas de Karnaugh*

Finalmente, dentro de los cuadrados se colocan los números decimales correspondientes a la combinación binaria de las coordenadas vertical-horizontal para cada cuadrado. Así, el valor decimal para el cuadrado correspondiente a AB=10, CD=00 sería el decimal correspondiente al binario 1000=8:

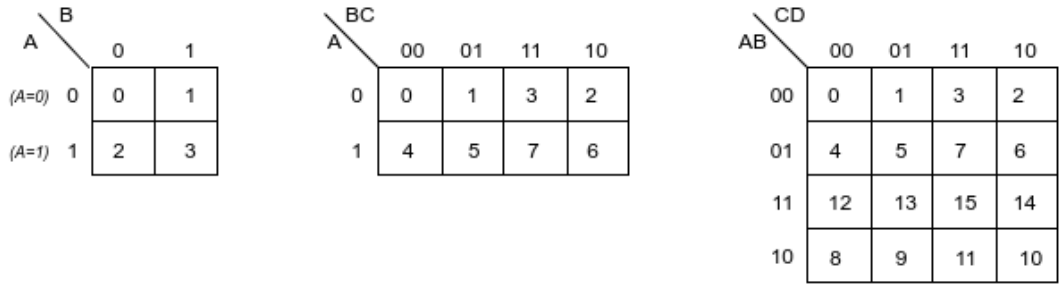


Gráfico 5: *Disposición de los números decimales en el diagrama de Karnaugh*

Una vez creados los diagramas, conviene identificar las filas y columnas en las que cada una de las variables siempre es 1, tal como se muestra en el diagrama de página siguiente (indicado con llaves).

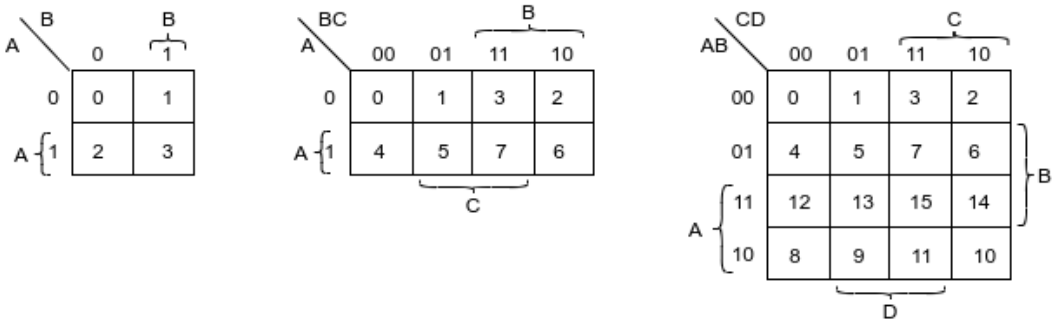


Gráfico 6: *Diagrama de Karnaugh: identificación de filas y columnas en las que cada variable siempre tiene 1 como valor.*

A continuación, se toma la expresión que se desea simplificar, se localizan los decimales de la expresión en el diagrama, y se los agrupa en cuadros adyacentes. Por ejemplo, la expresión $f(A,B,C)=\sum(3,4,6,7)$ se agruparía como se muestra en el Gráfico 7.

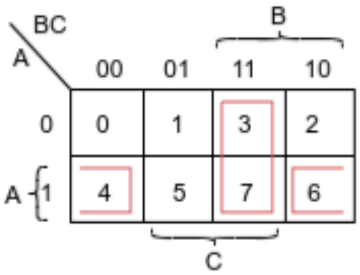


Gráfico 7: *Agrupación de cuadros adyacentes*

Teniendo en cuenta que las filas representan los AND y las columnas los OR (en caso de *minitérminos*), se debe elaborar una lista por cada expresión que genera 1 para cada *minitérmino* agrupado, siempre eligiendo la expresión más corta. Por ejemplo, en el caso anterior, la única expresión que da 1 tanto para el *minitérmino* 3 como para el 7, es $\boldsymbol{BC}$, pues $\bar{A}BC$ o $\bar{A}C$ solo son 1 para el 3 mientras que ABC solo para el 7. En el caso del minitérmino 4 (adyacente con el 6 por estar en los bordes, de allí que se dejase abierto el cuadro de selección) y el 6, $\boldsymbol{A\bar{C}}$ es la expresión que da 1 en ambos casos, pues $A\bar{B}$ solo es 1 en el 4 y AB en el 6. Por lo tanto, la simplificación de la expresión $f(A,B,C)=\sum(3,4,6,7)$ es $f(A,B,C)=BC+A\bar{C}$.

CAPÍTULO II

PUERTAS[3] LÓGICAS Y DISEÑO DE CIRCUITOS

La información binaria comentada en el capítulo previo, se representa físicamente en los equipos informáticos mediante señales eléctricas evaluadas de forma booleana por unos componentes denominados *puertas lógicas*. Estos componentes reciben unas cargas de voltaje, las evalúan y responden en consecuencia. Antes de describir estos componentes, se tratará de poner en términos concretos el proceso de evaluación, a fin de alcanzar una mejor comprensión. Pero para entender en profundidad qué son estas puertas lógicas y cómo funcionan, más allá de su uso y aplicación, es preciso responder a la pregunta, qué es realmente la información binaria y cómo es evaluada por un circuito lógico.

En términos concretos, estos componentes (circuitos denominados *puertas lógicas*) reciben cero o más cargas de voltaje. Por ejemplo, un circuito que tuviese que evaluar una expresión AND de dos variables, en su forma más primitiva, podría tener dos conductores (cables, por ejemplo), uno para cada variable. Suponiendo que una variable "verdadera" (con valor booleano 1) se fuese a representar con una carga de 5 voltios y el valor 0 (falso), con ausencia de carga,

---

3 A lo largo de este trabajo se utiliza el término *puerta lógica* y no *compuerta* como bien puede encontrarse en mucha bibliografía en español, puesto que la definición del término en inglés según el diccionario de Cambridge, se corresponde con la segunda acepción del término según la vigésimo tercera edición del diccionario de la Real Academia Española, y debido a que la definición de «compuerta» según este mismo diccionario, no se corresponde con la función de los circuitos tratados en este trabajo.

cada conductor podría recibir una carga de 5 o de 0 voltios (es decir, no recibir ninguna carga). Esto implicaría que si las dos variables fuesen verdaderas, el receptor de la carga recibiría una carga total de 10 voltios. Si una sola de ellas fuese verdadera, recibiría solo 5 voltios. Y si ambas fuesen falsas, no recibiría ninguna carga (es decir, 0 voltios).

El circuito debería emitir entonces, 5 voltios en caso de una evaluación verdadera (porque 5 voltios sería la carga que representase al 1 booleano), y 0 voltios en caso contrario. La pregunta es entonces ¿cómo podría el circuito, lograr dicho resultado? ¿Cómo realizaría la evaluación? La respuesta es que lo lograría interceptando la salida, con una resistencia de 5 voltios. En un plano ideal con cargas invariables, cuando se recibiese una carga de 10 voltios, la señal de salida sería 5. Y si se recibiesen 5 (o no se recibiese ninguna carga), la señal de salida sería 0 voltios, es decir, sería la ausencia de señal eléctrica.

Esto significa que la evaluación realizada por dichos componentes es en realidad una ***reacción eléctrica*** que responde a leyes físicas y a propiedades de la materia como la conductancia de los materiales con los que se construyen los circuitos. Será luego, el microprocesador, quien basado en las respuestas eléctricas que reciba, se encargue de representar los resultados de forma más concreta.

Tener en cuenta todo este contexto, podría facilitar comprender qué son en realidad las puertas lógicas, cómo cumplen su función de *evaluar bits para manipular información*, y cómo son capaces de ejecutar operaciones lógicas y arrojar resultados comprensibles para un procesador.

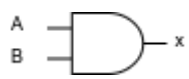
Funcionamiento de las puertas lógicas

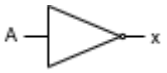
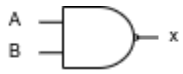
Las puertas lógicas son circuitos electrónicos digitales que manejan señales eléctricas en dos *estados o valores lógicos*, representados por diferentes cargas de voltajes. Dado que dicha representación es imprecisa (no se maneja con valores fijos como el ejemplo introductorio), suele estar determinada por rangos de voltaje y no por un voltaje concreto. Así, cualquier voltaje entre 0 V y 0.8 V representa el valor lógico referido como 0 y cualquier valor entre 2 V y 5 V, el valor lógico referido como 1. Por lo tanto, todo valor entre 0.8 V y 2 V se considera dentro de un rango indeterminado por lo que no puede predecirse el valor de respuesta.

Tipos de puertas lógicas

Para cada tipo de operación booleana existe un tipo de puerta lógica que le corresponde. Es posible encontrar un total de siete tipos de *puertas lógicas*, cada una con su operación booleana y su símbolo correspondiente (Tabla 3).

Tabla 3: *Resumen de tipos y características de puertas lógicas*

Puerta lógica	Datos de entrada	Datos de salida	Estado devuelto	Expresión algebraica
Puertas lógicas básicas				
AND 	2 o más	1	1, si todas las entradas tienen estado 1. 0, en caso contrario.	$x = AB$
OR 			0, si todas las entradas tienen estado 0. 1, en caso contrario.	$x = A + B$

Puerta lógica	Datos de entrada	Datos de salida	Estado devuelto	Expresión algebraica
NOT  *(o inverter)*	1	1	El estado opuesto al estado de entrada.	$x = \overline{A}$
Puertas lógicas universales				
NAND 	2 o más	1	0, cuando todos los valores de entrada tienen estado 1. 1, en caso contrario.	$x = \overline{AB}$
NOR 			1, cuando todos los valores de entrada tienen estado 0. 0, en caso contrario.	$x = \overline{A+B}$
Puertas lógicas exclusivas				
XOR 	2	1	1, si solo uno de los dos valores de entrada tiene estado 1. 0, en caso contrario.	$x = A \oplus B$ $= A\overline{B} + \overline{A}B$
XNOR 			1, cuando ambos valores de entrada tienen estado 0. 0 en caso contrario.	$x = A \odot B$ $= AB + \overline{A}\,\overline{B}$ $= \overline{A \oplus B}$ $= \overline{A\overline{B} + \overline{A}B}$

Cada una de estas puertas recibe uno o más valores de entrada (señales de entrada designadas por las variables A , B , C , ...) y un valor de salida (señal de salida designada por la variable x), cuyas combinaciones y sus correspondientes resultados se listan en una tabla de verdad como la que se mencionó en el informe previo[4]. Estas tablas de verdad, a su vez, responden a la expresión booleana que la puerta lógica lleva a cabo. Así, la tabla de verdad para

4 Informe I: Fundamentos de la Informática. Eugenia Bahit, Octubre de 2020.

la puerta lógica AND para las variables A, B va a ser la tabla de verdad para la expresión algebraica $x = AB$.

De todas las puertas lógicas, el total de las operaciones que un ordenador puede llevar a cabo, es posible ejecutarlas mediante las tres *puertas lógicas básicas*, AND, OR y NOT. La exactitud de las operaciones más complejas, depende de la forma en la que dichas puertas se conecten entre sí. Sin embargo, existen también dos *puertas lógicas universales*, NAND y NOR, contracciones de NOT-AND y NOT-OR, respectivamente, que pueden realizar las mismas operaciones que las tres puertas básicas. Otras dos puertas, XOR y XNOR, ejecutan operaciones OR y NOT-OR exclusivas, respectivamente.

Diseño lógico

El diseño lógico es el proceso que consiste en determinar la forma de conectar las diferentes puertas lógicas con otros bloques de circuitos, para llevar a cabo una operación determinada.

PROCESO DE DISEÑO. El proceso de diseño se inicia con el planteo del problema y finaliza con el dibujo del diagrama del circuito. El paso a paso puede resumirse en la siguiente lista:

1. Se plantea el problema.
2. Se estipulan las variables de entrada y salida que serán necesarias.
3. Se asigna una letra a cada una de las variables establecidas en el paso anterior.

4. Se realizan las tablas de verdad correspondientes.

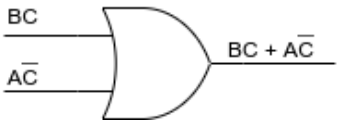
5. Se simplifican las expresiones/funciones booleanas.

6. Se dibuja el diagrama lógico.

Los cinco primeros pasos envuelven todos los conocimientos previos y el sexto paso, requiere la implementación de un mecanismo de conversión de expresiones booleanas a circuitos lógicos.

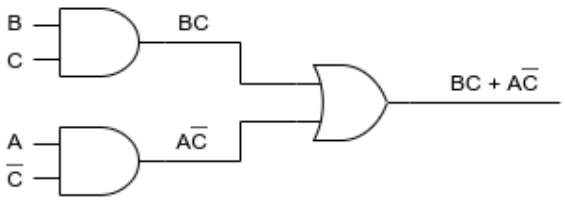
CONVERSIÓN DE EXPRESIONES BOOLEANAS.

El proceso de conversión podría decirse que consiste en dibujar el diagrama comenzando "de afuera hacia adentro", esto es, comenzar por la expresión que se desea convertir y continuar, uno a uno, por cada uno de sus términos. Así, una expresión de suma se iniciaría con una puerta OR y una de producto, por una puerta AND.



Tomando como ejemplo la expresión de la función vista al final del capítulo anterior, $f(A,B,C)=BC+A\bar{C}$, la conversión comenzaría por una puerta OR con dos entradas BC y $A\bar{C}$.

A continuación, se seguiría por cada una de las entradas, BC y $A\bar{C}$ mediante dos puertas AND.



Para finalmente, incorporar el INVERT (NOT) para el último componente:

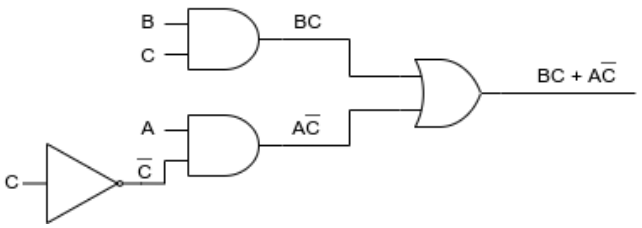


Gráfico 8: *Diseño lógico para la expresión algebraica BC + AC'*

CONVERSIÓN DE DIAGRAMAS.

Para el proceso inverso, se comienza por las entradas en vez de hacerlo por las salidas. Así, partiendo del diagrama anterior, se llegaría a la expresión $BC+A\bar{C}$:

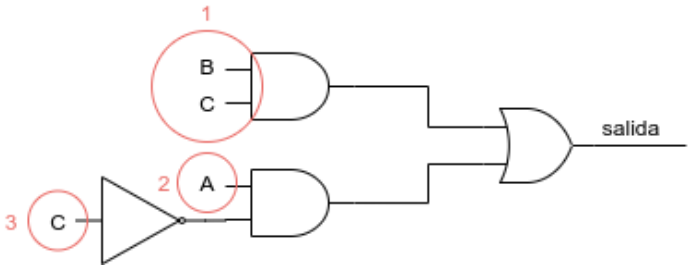


Gráfico 9: *Proceso de conversión desde un diagrama lógico a expresión algebraica*

CAPÍTULO III

CIRCUITOS LÓGICOS DIGITALES

A diferencia de los circuitos analógicos donde los niveles de voltaje varían permanentemente dentro de un rango, en los circuitos digitales, como bien se comentó en el capítulo anterior, se asumen un número finito de valores discretos de voltaje, lo que da lugar a que el diseño de los mismos, haga uso del álgebra booleana y a menudo se los denomine *circuitos de conmutación*.

Funciones de los circuitos lógicos digitales

1. **Operaciones aritméticas** de suma, resta, multiplicación y división son llevadas a cabo por circuitos denominados *adicionador*, *sustractor*, *multiplicador* y *divisor* respectivamente.

2. **Operaciones de codificación**, llevadas a cabo por un *codificador* que convierte los datos de entrada en sus respectivo código binario.

3. **Operaciones de decodificación**, que se llevan a cabo por un *decodificador* que realiza la operación inversa que el *codificador*.

4. **Operaciones de multiplexación**, llevadas a cabo por un *multiplexor* que permite conmutar información desde varias líneas de entrada a una sola dentro de una secuencia determinada.

5. **Operaciones de demultiplexación** (opuesta a la de multiplexación).

6. **Operaciones de comparación**, llevadas a cabo por un *comparador*, evalúa si dos variables son iguales y si no lo son, indica cual de ellas es la más grande.

7. **Operaciones de conversión**, llevas a cabo por un *conversor de código*, tienen por objetivo convertir una entrada desde un código a otro.

8. **Operaciones de almacenamiento**, llevadas a cabo por *registros*, que tienen por función almacenar información de forma temporal.

9. **Operaciones de cuenta**, llevadas a cabo por un *contador*, tienen por función contar la cantidad de pulsos (señales) que le son enviados. Los contadores también pueden ser usados para llevar a cabo **operaciones de división de frecuencia**.

10. **Operaciones de transmisión**, llevadas a cabo por *transmisores* (que transmiten las señales/información de un lugar a otro) y receptores (que reciben la información transmitida).

Circuitos integrados

Los circuitos integrados (CI) monolíticos, comúnmente denominados *chips*, son circuitos electrónicos construidos en una única pieza de, generalmente, silicona: un material semiconductor.

Existen distintos niveles de integración

SOBRE LOS SEMICONDUCTORES.

Cuando las cargas eléctricas se desplazan de una posición a otra dentro de un objeto, se produce un fenómeno físico

para los circuitos, dependiendo de la cantidad de puertas lógicas que empleen (Tabla 4).

Por otra parte, los circuitos pueden ser clasificados como analógicos o digitales, dependiendo de si requieren o no, componentes externos para llevar a cabo sus operaciones.

Los circuitos digitales son los abarcados en este trabajo, y son aquellos que no requieren componentes externos para llevar a cabo sus operaciones.

conocido como *conducción eléctrica*.

Algunos materiales facilitan la conducción y otros la repelen. A los primeros se los denomina *conductores* y a los segundos, *aislantes*.

Los ***semiconductores*** son un tipo de material intermedio entre los conductores y los aislantes, que permiten un mejor control de las cargas eléctricas y temperaturas del material, a través de una técnica de introducción de impurezas denominada *dopado de semiconductores*.

Tabla 4: *Niveles de integración en los circuitos integrados*

Tipo:	**SSI**	**MSI**	**LSI**	**VLSI**	**ULSI**
Cantidad de puertas lógicas:	< 12	≥ 12 ≤ 99	≥ 100 ≤ 9999	$\geq 10{,}000$ $\leq 99{,}999$	$\geq 100{,}000$

Las siglas de cada columna de la tabla precedente, corresponden a las denominaciones en inglés de los diferentes niveles de integración de los circuitos, siendo estas *Small*, *Medium*, *Large*, *Very Large* y *Ultra Large Scale Integration*, respectivamente.

Los circuitos integrados pueden clasificarse, a su vez, en dos clases de circuitos:

- Los *circuitos combinados*, donde el valor de salida depende exclusivamente de los valores de entrada dados en el presente;
- y los *circuitos secuenciales*, donde el valor de salida depende de los valores de entrada pasados y presentes. Por ello, los circuitos secuenciales tienen memoria y los combinados, no.

Ambos circuitos serán explicados a lo largo de este capítulo.

Circuitos combinados y secuenciales

Un ***circuito combinado*** es un conjunto de m funciones booleanas para cada variable de salida, expresada en términos de n variables. Es decir que los mismos se componen de n variables de entrada (para las que se sabe que existen 2^n combinaciones posibles), m variables de salida (que son funciones booleanas de cada una de las combinaciones de entrada), y entre las mismas, una colección de puertas lógicas interconectadas.

Los circuitos combinados se componen de variables de entrada, puertas lógicas y variables de salida. Como bien se comentó al comienzo del *«Capítulo II. Puertas Lógicas»*[5], tanto entrada como salida, no son más que señales discretas de voltaje.

Las señales de entrada llegan a las puertas lógicas desde unos circuitos formados por una única celda binaria capaz de almacenar un único bit, denominados *flip-flops,* quienes además almacenan la señal de salida producida por estas puertas. Un *flip-flop*, formalmente denominado *multivibrador biestable*, al poder

5 Ver páginas 14 a 16.

almacenar un único bit, solo retiene un único estado: el valor de la variable (~5V o ~0V) y su complemento (~0V o ~5V, respectivamente).

La interconexión de los circuitos combinados con estos *flip-flops*, es la que se denomina ***circuito secuencial***, y la forma de interconexión se puede describir gráficamente mediante los diagramas de bloque correspondientes (Gráficos 10 y 11, respectivamente).

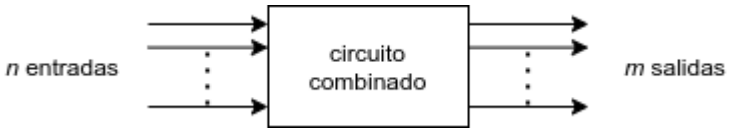


Gráfico 10: *Diagrama de bloques de un circuito combinado*

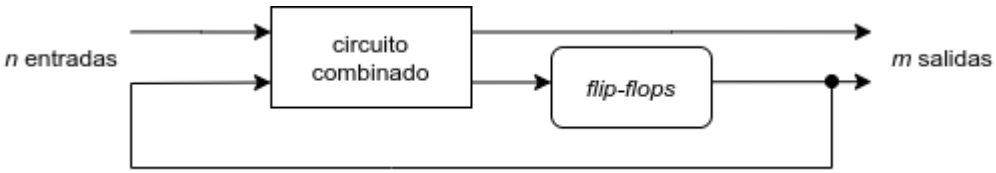


Gráfico 11: *Diagrama de bloques de un circuito secuencial*

Según se observa en los diagramas precedentes, los circuitos combinados son la parte central (el circuito propiamente dicho) independiente de las entradas o salidas almacenadas previamente. El circuito en sí solo se encarga de analizar las señales de entrada y arrojar las salidas correspondientes.

El diagrama de bloques del circuito secuencial, permite observar que dichas entradas pueden llegar al circuito combinado de forma externa, o también, desde *flip-flops* como salidas previas, y que las nuevas salidas pueden producirse de forma directa desde los circuitos combinados o desde los almacenes de memoria.

Esto significa que en el caso que las entradas al circuito combinado provengan de almacenes previos, dependerán por lo tanto de valores previos. Ambos diagramas explican lo que se comentó en página 24 al describir los tipos de circuitos.

Las principales diferencias entre los circuitos combinados y secuenciales son descritas en la Tabla 5.

Tabla 5: *Diferencias entre circuitos combinados y secuenciales.*

Aspecto	Circuitos combinados	Circuitos secuenciales
Dependencia de estados	No depende de las variables previas	Puede o no depender de variables previas
Unidad de almacenamiento	No requerida	Requerida para almacenar estados previos
Velocidad de salida	Más rápidos	Más lentos
Complejidad del diseño	Menos complejo	Más complejo

La Tabla 5, inspirada en las diferencias propuestas por Anand Kumar en *«Switching Theory and Logic Design»*, pp. 487, incluye una diferencia con respecto a la velocidad de salida, la cual se encuentra determinada por elementos de retraso de tiempo de los que se hablará en la siguiente sección y permitirán argumentar mejor dicha diferencia.

La complejidad del diseño se supone mayor en los circuitos secuenciales que en los combinados, dada la interconexión de los segundos con las unidades de almacenamiento, por lo que se requiere tener en cuenta una mayor cantidad de circuitos.

Circuitos secuenciales sincrónicos y asíncronos

Los *circuitos secuenciales sincrónicos* son aquellos circuitos cuyas operaciones se activan cuando un *pulso de reloj* es recibido por el circuito. Estos pulsos de reloj son señales emitidas de forma periódica por un dispositivo denominado *generador de pulsos de reloj*. Este concepto permitiría introducir un agreado al diagrama de bloques del Gráfico 11 (página 25) que refleje el sincronismo (Gráfico 12).

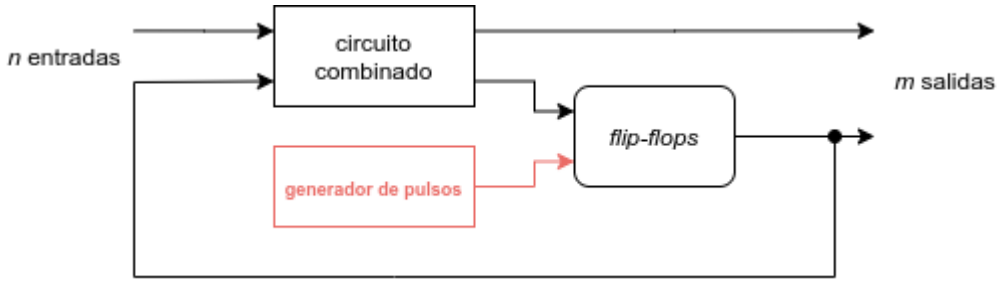


Gráfico 12: *Diagrama de bloques de un circuito secuencial sincrónico*

GENERADOR DE PULSOS PERIÓDICOS. Es un sistema que en sincronía con un reloj, genera una secuencia preestablecida de bits cada un cierto período de tiempo.

Los *circuitos secuenciales asíncronos* son aquellos que no son activados por una señal de reloj. Sin embargo, pueden tener elementos de retraso de tiempo (*delay*) que permiten controlar la retroalimentación del circuito y proveen al circuito de un almacenamiento temporal, tal y como se muestra en el siguiente diagrama de bloques (Gráfico 13).

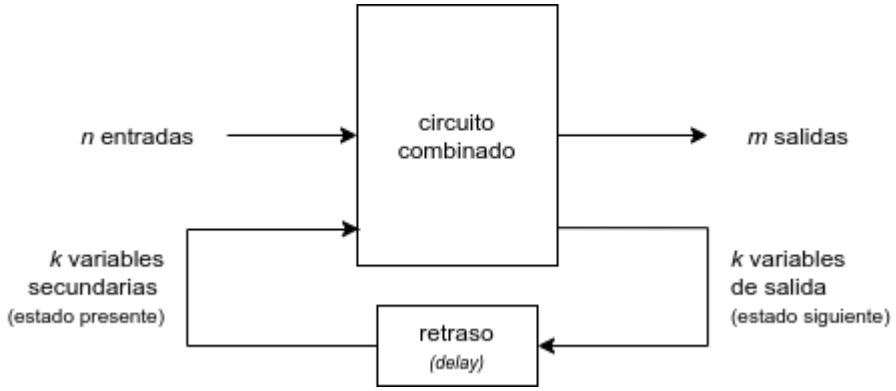


Gráfico 13: *Diagrama de bloques de un circuito secuencial asíncrono*

Como se observa en el Gráfico 13, en los circuitos secuenciales asíncronos, además de las n variables de entrada y m variables de salida, existen k *variables secundarias* de almacenamiento temporal destinadas a almacenar el estado presente (*variables secundarias*) y el estado siguiente (*variables secundarias de salida*).

Dado que las capacidades de almacenamiento son finitas, los estados de los circuitos también lo son. Por tal motivo, al hablar de circuitos secuenciales sincrónicos, se hace referencia a circuitos de estado finito o *máquinas de estado finito*, las cuáles se introducen brevemente en la siguiente sección, a fin de ser abarcadas en profundidad en trabajos futuros.

Máquinas de estado finito

En el contexto de la Teoría de Conmutación, una máquina de estado finito (MEF) —también llamada *autómata finito* (AF)— es un modelo abstracto que describe una máquina secuencial sincrónica. Su comportamiento se describe como una secuencia de eventos en un instante de tiempo discreto t .

Se trata de circuitos secuenciales cuyas características se limitan por:

- El *número de elementos de memoria* es finito, y por lo tanto, lo es su capacidad de almacenamiento.
- Los *estados internos* también son finitos, pues la capacidad de almacenamiento lo es, y por lo tanto solo podrán almacenarse un número finito de estados internos.
- Los *valores previos* afectan el comportamiento futuro del circuito en un número finito de formas, puesto que la cantidad de estados internos es finita.

Modelos de Mealy y Moore

Una MEF puede producir diferentes salidas según el modelo que implemente:

1. ***Modelo de Mealy:*** la salida es una función que depende del estado actual y de los valores previos.
2. ***Modelo de Moore:*** la salida es una función que solo depende del estado actual.

Matemáticamente, el **estado siguiente** $S(t+1)$ de una MEF queda determinado por $S(t+1) = f\{ S(t), x(t) \}$, siendo $S(t)$ el estado presente, $x(t)$ la entrada actual y $S(t+1)$ el estado siguiente.

Según cada modelo, el **valor de salida** $z(t)$ será determinado por $z(t) = g\{ S(t), x(t) \}$ para el modelo de Mealy (ya que depende de ambos

estados, previo y presente) y por $z(t) = g\{ S(t) \}$ para el de Moore (ya que solo depende del estado presente). A fin de ejemplificar gráficamente ambos modelos, se presentan a continuación los diagramas de bloques correspondientes, simplificados para una mejor comprensión (Gráficos 14 a 16).

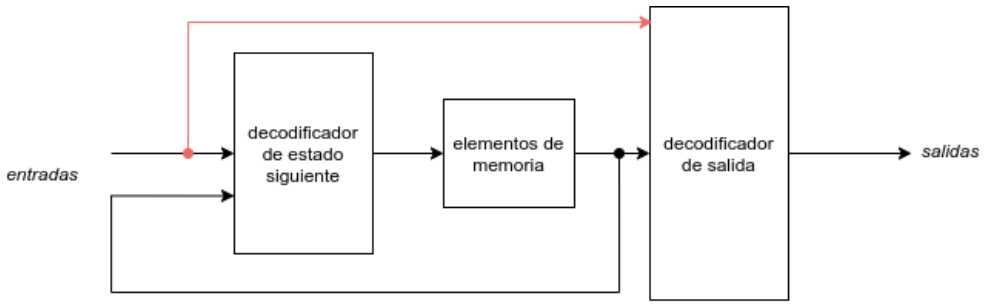


Gráfico 14: *Diagrama de bloques de una MEF según modelo de Mealy*

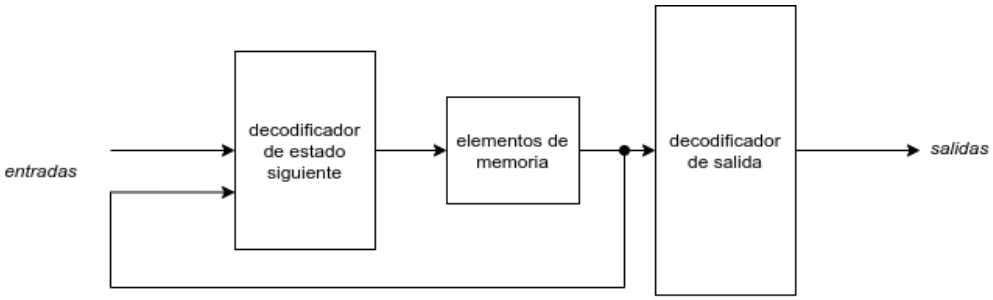


Gráfico 15: *Diagrama de bloques de una MEF según modelo de Moore con decodificador de salida*

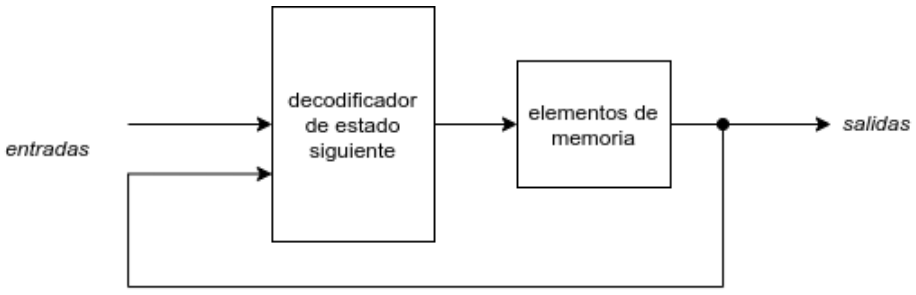


Gráfico 16: *Diagrama de bloques de una MEF según modelo de Moore (sin decodificador de salida)*

DIAGRAMAS DE ESTADO. Los diagramas de estado se utilizan para representar gráficamente las relaciones entre los tres estados posibles de un circuito secuencial —presente, pasado y futuro (o siguiente)— y la salida.

Para ello, el diagrama emplea los siguientes elementos:

- CÍRCULOS. También referidos como nodos o vértices (como en los grafos) se utilizan para representar los estados.
- NÚMERO BINARIO. Dentro del círculo, indica el estado representado por el círculo. En las líneas, indica: valor de entrada y salida (con la forma `<entrada>/<salida>`), en el caso del modelo de Mealy, y la entrada, para el modelo de Moore.
- LÍNEAS. Cuando conectan al círculo con sí mismo, indica que el siguiente estado es igual al estado presente. Cuando conectan dos círculos, indican transición de estado (de un estado a otro).

Un ejemplo de estos diagramas puede verse en el siguiente gráfico:

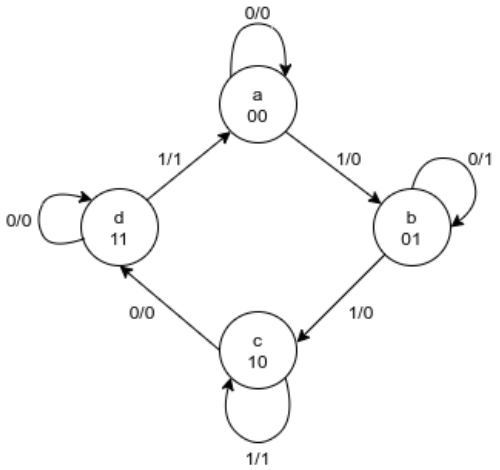


Gráfico 17: *Diagrama de estado en el modelo de Mealy*

Circuitos lógicos programables

Los *circuitos lógicos programables* (o dispositivos lógicos programables —DLP —) son circuitos integrados configurables, sobre los que se pueden implementar funciones lógicas.

Estos circuitos integrados son programables en la medida que permitan "romper" selectivamente, la interconexión entre las diferentes puertas lógicas, *flip-flops* y *registros* que lo componen (y dejar intactas otras), a fin de lograr las funciones deseadas.

REGISTROS. Un *registro* es un conjunto de *flip-flops* empleado para almacenar información binaria. Desde el momento en el que un *flip-flop* puede almacenar 1 bit, es considerado un *registro de bit único*.

Memorias programables de solo lectura (PROM)

Las memorias de solo lectura (ROM) programables (PROM, por sus siglas en inglés), son uno de los dispositivos que pueden encontrarse en la categoría de los DLP, y los hay de cuatro tipos:

1. Máscara de memoria programable de solo lectura (**MROM**[6]).
2. Memoria programable de solo lectura (**PROM**[7]).
3. Memoria programable borrable de solo lectura (**EPROM**[8]).

---

6 **MROM:** Mask read-only memory

7 **PROM:** Programmable read-only memory

8 **EPROM:** Erasable programmable read-only memory.

4. Memoria programable y eléctricamente borrable de solo lectura (**EEPROM**[9]/**E²PROM** o **EAROM**[10]).

Las características de estos cuatro tipos se describen en la Tabla 6.

Tabla 6: *Tipos de memorias de solo-lectura y sus características programables.*

Característica	**MROM**	**PROM**	**EPROM**	**EAROM**
Patrón de datos	Especificados por el(la) usuario(a). Programados por el(la) fabricante.	Especificados y programados eléctricamente por el(la) usuario(a).		
Reprogramable	NO		SI	
Método de borrado	No posee		Radiación ultravioleta por ~30’	Eléctricamente

Dispositivos lógicos programables combinados

También es posible encontrar DLP combinados. Estos DLP son circuitos integrados con puertas lógicas programables, dispuestas en dos series de puertas AND y OR, ordenadas de forma tal que puedan proveer una suma de productos AND-OR.

Los DLP combinados se encuentran en tres tipos:

1. **Dispositivos PROM:**

   Los cuales poseen una serie AND fija —como decodificador—, y una serie OR programable.

9 **EEPROM:** Electrically erasable and programmable read-only memory.

10 **EAROM:** Electrically alterable read-only memory.

2. **Los circuitos PAL**[11] (*lógica de series*[12] *programables*):

   Compuestos por una serie AND programable, y una serie OR fija.

3. **Los circuitos PLA**[13] (*series de lógica programables*):

   Los cuales constan de dos series AND y OR, ambas programables.

La diferencia entre estos tres tipos de circuitos, se resume en la Tabla 7.

Tabla 7: *Diferencias y similitudes entre los tres tipos de dispositivos lógicos programables combinados.*

Característica	**PROM**	**PAL**	**PLA**
Serie AND	Fija	Programable	
Serie OR	Programable	Fija	Programable
Minitérminos	Decodificados	Programados para obtener los deseados	
Funciones booleanas admitidas	Solo forma canónica de SOP	Forma normal o canónica de SOP.	

11 **PAL:** Programmable Array Logic.

12 Se utiliza el término «serie» como traducción al español del término inglés «*array*», por considerar que es el término que se corresponde en español, a la definición dada en el diccionario de Cambridge para el término *array*.

13 **PLA:** Programmable Logic Array.

CONCLUSIONES

Considerando que los bits utilizados para representar la información son señales eléctricas y que estos se transfieren y evalúan por medio de circuitos que facilitan tanto la conducción de dichas señales como el almacenamiento de las mismas, es posible llegar a la conclusión de que en términos concretos, las operaciones que se ejecutan en dichos circuitos son combinaciones de señales eléctricas que, haciendo uso de teorías físicas como la Teoría de la Conductividad, son evaluadas matemáticamente para producir resultados lógicos, también representados mediante señales eléctricas. A partir de ello y de la exposición realizada en los capítulos previos, además es posible concluir qué:

A) Las leyes y teoremas del álgebra booleana, así como sus mecanismos de análisis y simplificación de expresiones algebraicas, son la base teórica para entender y evaluar las señales eléctricas que representan la información.

B) Las puertas lógicas son la materialización del álgebra booleana, y que a través del diseño de circuitos, proveen los mecanismos necesarios para la creación de sistemas lógicos digitales de gran complejidad.

C) Al hablar de circuitos lógicos digitales, el conocimiento no solo se limita a la integración del álgebra booleana con los mecanismos de diseño de los mismos, sino que además, permite entender a la Teoría de Conmutación de Circuitos y el Diseño Lógico como a la formalización matemática que sirve de sustento para crear modelos abstractos que luego se empleen como base para la producción electrónica de los mismos.

BIBLIOGRAFÍA

[0] Gupta, M. (2020). *Discrete Mathematics* (19th ed., pp. 223-314). Meerut: Satyendra Rastogi Mitra.

[1] Hernández González, F. (2020). *Fundamentos físicos de las ciencias informáticas* (1st ed. pp. 140-144). Londres: Hackers & Developers Press.

[2] Kulkarni, V. (2013). *Theory of Computation* (1st ed. pp. 20-21). New Dehli: Oxford University Press.

[3] Kumar, A. (2014). *Switching Theory and Logic Design* (2nd ed. pp. 1-10, 111-666). Dehli: PHP Learning.

[4] Mano, M. (1993). *Computer System Architecture* (3rd ed., pp. 1-50). New Jersey: Pearson Education.

[5] McCluskey, E. (2003). *Encyclopedia of Computer Science* (pp. 1727–1731). United Kingdom: John Wiley and Sons Ltd.

[6] Rajaraman, V. & Adabala, N. (2015). *Fundamentals Of Computers* (6th ed. pp. 122-145). Deli: PHI Learning.

[7] Serway, A. & Jewett, J. (2003). *Física, Volumen 2* (3rd ed. pp, 667-671). Madrid: Thomson Learning.