Andy Mendoza

ALGEBRA DE CIRCUITOS DIGITALES

El álgebra de circuitos digitales o álgebra de Boole es un sistema matemático que opera con valores binarios 1 y 0, también denominados verdadero y falso, o de señal alta o señal baja. Su principal objetivo es simplificar funciones lógicas, para esto utiliza un conjunto de leyes, teoremas y axiomas que facilitan la tarea de simplificar expresiones lógicas [1].

Operadores

En el álgebra de circuitos existen tres operadores básicos esenciales:

- **Conjunción:** También denominado producto booleano se lo representa con "∧" (AND) o como una multiplicación, y devuelve 1 o un valor verdadero si ambas entradas son 1 [2].
- **Disyunción:** Conocido también como suma booleana se lo representa con "∨" (OR) o con el signo +, este devuelve 1 cuando al menos una de las entradas tiene una señal alta [2].
- **Negación:** Se lo representa con "¬" (NOT) o con una línea encima de la variable de entrada, por ejemplo "Ā", este cambia el valor de verdad de la entrada, si es 1 se convierte en 0 y viceversa [2].

Leyes y propiedades del álgebra de circuitos digitales

Para manejar y simplificar expresiones booleanas, se recurre a un conjunto de leyes que permiten transformar expresiones complejas en otras más simples, lo cual es clave para optimizar el diseño de circuitos lógicos [3].

- **Ley de identidad:** Sumar 0 o multiplicar por 1 no cambia el valor de la variable [3]. Por ejemplo:

$$B + 0 = B \text{ y } B \cdot 1 = B.$$

- **Ley idempotente:** Si sumamos o multiplicamos una variable consigo misma no cambia su valor de verdad [3]. Ejemplo:

$$P + P = P \text{ o } P \vee P = P$$

$$P \cdot P = 0 \text{ o } P \wedge P = P$$

- **Ley del complemento:** Una variable OR con su complemento o negación siempre es 1 o verdadero, y AND con su negación siempre es 0 o falso [4]. Por ejemplo:

$$A + \bar{A} = 1 \text{ o } A \vee \neg A = 1$$

$$A \cdot \bar{A} = 0 \text{ o } A \wedge \neg A = 0\}$$

- **Ley conmutativa:** El orden de los operandos no afecta el resultado, tanto para OR como para AND [4]. Ejemplo:

$$A + B = B + A \text{ o } A \vee B = B \vee A$$

$$A \cdot B = B \cdot A \text{ o } A \wedge B = B \wedge A$$

- **Ley asociativa:** Esta ley permite agrupar variables sin que el resultado se altere, es útil cuando hay más de dos variables [4]. Ejemplo:

$$A + (B + C) = (A + B) + C \text{ o } A \vee (A \vee B) \vee C$$

$$A \cdot (B \cdot C) = (A \cdot B) + C \text{ o } A \wedge (A \wedge B) \wedge C$$

- **Ley distributiva:** Como su nombre lo indica, permite distribuir una variable sobre una operación entre otras dos, lo que la hace útil para simplificar [4]. Ejemplo:

$$A \cdot (B + C) = (A \cdot B) + (A \cdot C) \text{ o } A \wedge (B \vee C) = (A \vee B) \wedge (A \vee C)$$

$$A + (B \cdot C) = (A + B) \cdot (A + C) \text{ o } A \vee (B \wedge C) = (A \wedge B) \vee (A \wedge C)$$

- **Leyes de Morgan:** Permiten transformar la negación de una conjunción en una disyunción de negaciones, y viceversa, lo que las hacen muy importantes para la simplificación de expresiones lógicas [5], [6].

$$\neg (A \wedge B) = \neg A \vee \neg B \text{ o } \overline{A \cdot B} = \overline{A} + \overline{B}$$

$$\neg (A \vee B) = \neg A \wedge \neg B \text{ o } \overline{A + B} = \overline{A} \cdot \overline{B}$$

FUNCIÓN BOOLE

Una función Boole es una expresión matemática que asigna un conjunto de valores binarios de entrada a un único valor binario de salida. A esta se la define formalmente como: $f: \{0,1\}n \rightarrow \{0,1\}$ donde n es el número de variables de entrada [7].

Para representar la función de n variables se realiza de la siguiente manera, por ejemplo: $f(x, y, z) = (x \cdot \bar{y}) + (y \cdot z)$ ubicando dentro de los paréntesis de la función las variables de entrada [8].

Representación mediante circuitos lógicos

Para representar una función Boole en un circuito lógico, tenemos que utilizar las compuertas lógicas AND, OR y NOT, ya que estás representan los operadores del álgebra de Boole, para ello ubicamos las variables de entrada con la que otra variable que corresponda para utilizar las compuertas lógicas ya sea AND, OR o NOT [9]. A continuación, en la Figura 1 se puede visualizar la representación de un circuito y en la tabla 1 su tabla de verdad:

Representación de la función F(X, Y, Z) = X + YZ en un circuito lógico

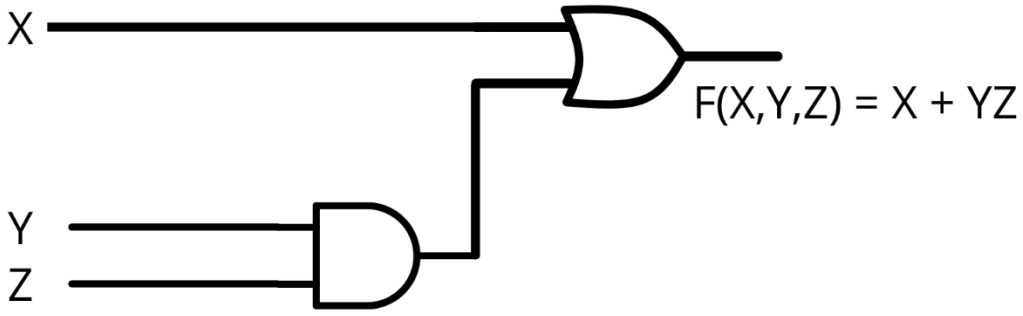


Figura 1. Representación de una función booleana en un circuito lógico

Tabla 1. Tabla de verdad de la función F(X, Y, Z) = X+ YZ

X	Y	Z	YZ	X + YZ
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Representación de mapas de Karnaugh

Un mapa de Karnaugh o también conocido como K-map, es una herramienta gráfica que permite visualizar los resultados de una función lógica para cada posible combinación de sus entradas. Cada celda representa una combinación particular de las entradas de la función o minitérmino, y el valor que contiene cada celda indica la salida de la función para esa combinación específica [10].

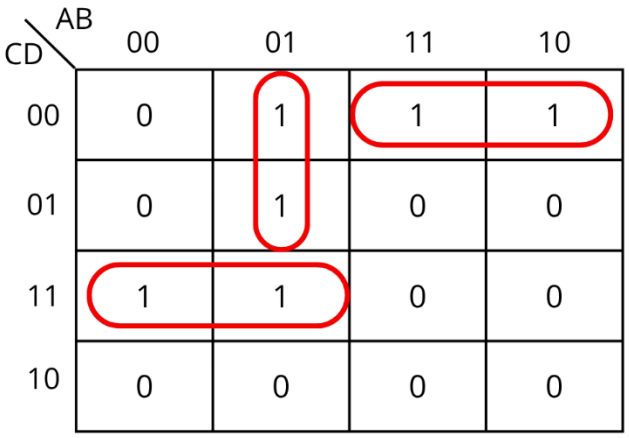
Para representar una función lógica en su forma canónica de minitérminos, simplemente ubicamos un 1 en las celdas que coinciden con sus minitérminos, colocando un 0 en las demás celdas. Después, tenemos que juntar en grupos de 2, 4 u 8 celdas adyacentes que contengan 1, con el propósito de simplificar la expresión lógica de una función [11].

A continuación, en la Figura 2 se presenta un ejemplo de un mapa de Karnaugh con cuatro variables y en la Tabla 2 su tabla de verdad:

Tabla 2. Tabla de verdad función F(A, B, C, D)

A	B	C	D	F(A,B,C,D)	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	1	$\bar{A}\bar{B}CD$
0	1	0	0	1	$\bar{A}B\bar{C}\bar{D}$
0	1	0	1	1	$\bar{A}B\bar{C}D$
0	1	1	0	0	
0	1	1	1	1	$\bar{A}BCD$
1	0	0	0	1	$A\bar{B}\bar{C}\bar{D}$
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	1	$AB\bar{C}\bar{D}$
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	0	

$$\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D}$$



$$\bar{A}B\bar{C} + A\bar{C}\bar{D} + \bar{A}CD$$

Figura 2. Mapa de Karnaugh de 4 variables

Formas canónicas

Las funciones booleanas se pueden representar de manera organizada utilizando formas canónicas. Esto es clave para poder simplificarlas y transformarlas en circuitos digitales. Las dos formas más utilizadas son SOP (suma de productos), y POS (producto de sumas) [12].

SOP (Suma de productos)

En esta forma canónica, la función boolena se representa como una disyunción de conjunciones, donde se agrupa los minitérminos o casos donde la salida de la función es 1 y realiza productos que luego se suman [13]. Ejemplo:

Si la función vale 1 en los casos (A = 0, B = 1, C = 0) y (A = 1, B = 0, C = 1), su forma SOP sería: $f(A,B,C) = \bar{A}B\bar{C} + AB\bar{C}$

POS (Producto de sumas)

En la forma POS, se utiliza la conjunción de disyunciones, donde se agrupa los maxitérminos o los casos donde la salida de la función es 0, por medio del producto de sumas asociadas [13]. Ejemplo:

Si la función vale 0 en las combinaciones (A = 0, B = 0, C = 0) y (A = 1, B = 1, C = 1), su forma POS sería: $f(A,B,C) = (A + B + C) \cdot (\bar{A} + \bar{B} + \bar{C})$

REFERENCIAS BIBLIOGRÁFICAS

[1] E. M. Jiménez-Hernández, H. Oktaba, F. Díaz-Barriga, and M. Piattini, "Using web-based gamified software to learn Boolean algebra simplification in a blended learning setting," *Computer Applications in Engineering Education*, vol. 28, no. 6, pp. 1591–1611, Nov. 2020, doi: 10.1002/cae.22335.

[2] J. Feng, R. Zhao, and Y. Cui, "Simplification of logical functions with application to circuits," *Electronic Research Archive*, vol. 30, no. 9, pp. 3320–3336, 2022, doi: 10.3934/era.2022168.

[3] S. K. Yadav, "Boolean Algebras and Applications," in *Discrete Mathematics with Graph Theory*, Cham: Springer International Publishing, 2023, pp. 305–353. doi: 10.1007/978-3-031-21321-2_7.

[4] K. Erciyes, "Boolean Algebras and Combinational Circuits," in *Discrete Mathematics and Graph Theory*, Springer, 2021, ch. 9, pp. 173–195. doi: 10.1007/978-3-030-61115-6_9.

[5] M. Frické, "Boolean Logic," *KNOWLEDGE ORGANIZATION*, vol. 48, no. 2, pp. 177–191, 2021, doi: 10.5771/0943-7444-2021-2-177.

[6] D. Natarajan, "Overview of Digital Signal Processing," in *Fundamentals of Digital Electronics*, 1st ed., Springer, 2020, ch. 1, pp. 1–12. doi: 10.1007/978-3-030-36196-9_1.

[7] R. O'Donnell, *ANALYSIS OF BOOLEAN FUNCTIONS*. arXiv Edition, 2021. doi: 10.48550/arXiv.2105.10386.

[8] V. Bakoev, "A Method for Fast Computing the Algebraic Degree of Boolean Functions," in *Proceedings of the 21st International Conference on Computer Systems and Technologies*, New York, NY, USA: ACM, Jun. 2020, pp. 141–147. doi: 10.1145/3407982.3408005.

[9] A. Elahi, *Computer Systems: Digital Design, Fundamentals of Computer Architecture and ARM Assembly Language*, 2nd ed. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-93449-1.

[10] C. Wang and Y. Tao, "Karnaugh Maps of Logical Systems and Applications in Digital Circuit Design," *Circuits Syst Signal Process*, vol. 39, no. 5, pp. 2245–2271, May 2020, doi: 10.1007/s00034-019-01214-x.

[11] S. Kurgalin and S. Borzunov, *The Discrete Math Workbook*. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-42221-9.

[12] A. Al-Bayaty and M. Perkowski, "A concept of controlling Grover diffusion operator: a new approach to solve arbitrary Boolean-based problems," *Sci Rep*, vol. 14, no. 1, p. 23570, Oct. 2024, doi: 10.1038/s41598-024-74587-y.

[13] J. H. B. Rozo, “Using Boolean algebra to model the economic decision-making / Usando a álgebra booleana para modelar a tomada de decisão econômica,” *Brazilian Journal of Business*, vol. 3, no. 2, pp. 1413–1426, Apr. 2021, doi: 10.34140/bjbv3n2-009.