**Álgebra de circuitos digitales**

El álgebra de circuitos digitales tiene como base el uso del álgebra booleana como método matemático para poder modelar y diseñar el comportamiento de los sistemas lógicos. A través de esta herramienta, es posible expresar funciones lógicas mediante variables binarias y operadores lógicos básicos, describiendo cómo responden los circuitos digitales ante distintas combinaciones de entrada. Esta representación algebraica resulta esencial para el diseño de compuertas lógicas y estructuras combinacionales en el que la práctica del álgebra booleana ha convertido en el asidero del diseño lógico contemporáneo en la electrónica digital[1].

**Álgebra booleana**

El libro “Electronics for Physicists” expone el álgebra booleana como una de las herramientas matemáticas más importantes para analizar los circuitos digitales. El álgebra booleana se fundamenta en los valores binarios, es decir, verdadero (1) o falso (0), que representan los dos estados de la corriente eléctrica: encendido o apagado. A partir de este punto se empieza a describir cómo se comportan las señales digitales que circulan en los diferentes sistemas electrónicos. El libro menciona que el álgebra booleana permite representar y manipular funciones lógicas que determinan el funcionamiento de las compuertas lógicas digitales. El vínculo entre las matemáticas y la electrónica garantiza una comprensión y facilita el diseño de circuitos. Esta álgebra es un lenguaje fundamental a la hora de tratar con lógica digital. Por otra parte, el propio autor presenta los principios elementales de este tipo de álgebra antes de mostrar ejemplos concretos. Por este motivo, establece una fundamentación clara para comprender cómo los circuitos digitales procesan información [2].

**Simplificación de expresiones booleanas**

La simplificación de expresiones booleanas es uno de los procedimientos necesarios en el diseño eficiente de circuitos digitales, ya que permite reducir el número de compuertas lógicas utilizadas, y en consecuencia optimizar el coste o la velocidad de funcionamiento. Tradicionalmente se ha recurrido a los mapas de Karnaugh o al algoritmo de Quine-McCluskey como métodos de simplificación de funciones lógicas, aunque tienen fuertes limitaciones para manejar grandes cantidades de variables. El artículo de Feng, Zhao y Cui plantea un nuevo procedimiento de simplificación de expresiones booleanas que se basa en el producto semi-tensorial de matrices, para transformar funciones booleanas en formas algebraicas, y para aplicar criterios estructurales que permiten la simplificación. Con el procedimiento introducido la minimización puede automatizarse de forma más sencilla mediante algoritmos. Se superan las limitaciones impuestas por los métodos clásicos en cuanto al número de variables que imponen procedimientos clásicos, por lo que su propuesta puede ser vista como una herramienta altamente eficaz en la simplificación lógica en situaciones complejas. Por último, se demuestra que su método tiene una complejidad computacional aceptable (O(2ⁿ)) y se puede incluir dentro de los procesos de diseño de circuitos sin tener que recurrir a representaciones gráficas o tabulares. El algoritmo además tiene en cuenta el orden de las variables lo cual, influye en el nivel de simplificación alcanzado[3].

**Leyes del álgebra booleana**

El álgebra de Boole es un sistema matemático que se basa en variables binarias (0 y 1), fundamentales para el diseño y análisis de circuitos digitales. Dentro de este sistema, se establecen diversas leyes que permiten simplificar expresiones lógicas [4]:

|  |  |  |
| --- | --- | --- |
| **Ley** | **Expresión** | **Descripción** |
| **Identidad** | a + 0 = a  a \* 1 = a | Si se operar con 0 (en suma) o 1 (en producto) no se altera el valor de la variable. |
| **Idempotente** | a + a = a  a \* a = a | Si se repite una variable no cambia el resultado. |
| **Complemento** | a + a′ = 1  a \* a′ = 0 | Una variable y su negación siempre producen un resultado constante. |
| **Conmutativa** | a + b = b + a  a \* b = b \* a | El orden de los operandos no afecta el resultado. |
| **Asociativa** | a + (b + c) = (a + b) + c  a \* (b \* c) = (a \* b) \* c | El modo en que se agrupan los términos no cambia el valor. |
| **Distributiva** | a \* (b + c) = ab + ac  a + (b \* c) = (a + b) (a + c) | La multiplicación se distribuye sobre suma (y viceversa). |
| **Absorción** | a + ab = a  a (a + b) = a | Cuando “a” ya está presente, cualquier combinación con “b” no altera el resultado. |
| **De Morgan** | (a + b) ′ = a′ \* b′  (a \* b) ′ = a′ + b′ | Permiten reescribir negaciones de expresiones compuestas. |

Estas leyes se aplican bajo el principio de dualidad, el cual indica que cualquier expresión válida en álgebra de Boole sigue siendo válida si se intercambian los operadores + y ·, así como los valores 0 y 1 [4].

**Formas Canónicas del Álgebra Booleana (SOP y POS)**

Dentro del álgebra booleana, se utilizan ampliamente, como expresiones estándares, las conocidas como Sum-of-Product (SOP) y Product-of-Sum (POS) en la simplificación lógica de funciones. La forma SOP es una suma (OR) de productos llamados minterms, y la forma POS un producto (AND) de sumas (OR) llamados maxterms. Ambas formas requieren que contengan todas las variables del sistema en su forma directa o complementada. El artículo de Nugroho explica que, por ejemplo, una función en forma SOP podría expresarse como , mientras que en forma POS se representaría como . Estas representaciones no solo son esenciales para el análisis teórico, sino que también constituyen la base sobre la cual operan diversos métodos de simplificación automatizada de funciones booleanas [5].

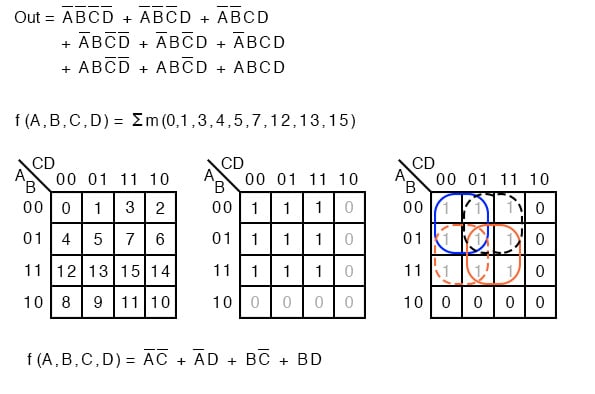


Figura 1. Ejemplo práctico de simplificación de función booleana

**Referencias**

[1] “Boolean Algebra and Logic Gates,” in *Foundations for Fintech*, vol. Volume 1, in Global Fintech Institute - World Scientific Series on Fintech, vol. Volume 1. , WORLD SCIENTIFIC, 2021, pp. 147–160. doi: doi:10.1142/9789811238819\_0007.

[2] B. H. Suits, “Digital I,” in *Electronics for Physicists: An Introduction*, B. H. Suits, Ed., Cham: Springer International Publishing, 2020, pp. 247–275. doi: 10.1007/978-3-030-39088-4\_12.

[3] 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.

[4] K. Erciyes, “Boolean Algebras and Combinational Circuits,” in *Discrete Mathematics and Graph Theory: A Concise Study Companion and Guide*, K. Erciyes, Ed., Cham: Springer International Publishing, 2021, pp. 173–195. doi: 10.1007/978-3-030-61115-6\_9.

[5] E. D. Nugroho, “Development of Applications for Simplification of Boolean Functions using Quine-McCluskey Method,” *Telematika*, vol. 18, no. 1, p. 27, Mar. 2021, doi: 10.31315/telematika.v18i1.3195.