**3.- Circuitos Lógicos y Sistemas Digitales**

Tal como mencionan Zhang et al. (2022), la lógica digital y los circuitos digitales son el fundamento de todas las aplicaciones electrónicas y permiten la representación y manipulación fiable de la información binaria; la evolución de los circuitos lógicos y de los sistemas digitales se ha dado gracias a la integración de modelos basados en algorítmica, de automatización y de materiales emergentes, como consecuencia de lo cual se han conseguido circuitos digitales más pequeños, más rápidos y con un mejor consumo energético [DOI: 10.1145/3489517.3530462].

La cada vez mayor demanda de dispositivos inteligentes y conectados que Wang et al. (2021), mencionan lleva a que cada vez se alcancen niveles de complejidad mayores que se traducen en la exigencia de mejoras constantes en el diseño y la funcionalidad de tales sistemas [DOI: 10.1109/DAC18074.2021.9586094].

**3.1.- Circuitos Lógicos Fundamentales**

Los circuitos lógicos se dividen en dos grandes categorías: combinacionales y secuenciales.

**3.1.1.- Circuitos Combinacionales**

Singh et al. (2020), sostienen que los circuitos combinacionales son aquellos en los que la salida solo dependía de las entradas actuales, que no poseían memoria interna, que usaban las operaciones lógicas inmediatamente y aquellas deterministas [DOI: 10.1007/s00034-020-01550-3].

Las compuertas lógicas básicas AND, OR, NOT, XOR, NAND y NOR son sistemas constitutivos de componentes que se pueden alcanzar, como son los propios sumadores, codificadores, decodificadores y multiplexores. Como mencionan Aljaedi et al. (2023), las compuertas NAND y NOR son universales y son los componentes esenciales de las tecnologías de lógica reversible para la computación cuántica y la nanoelectrónica [DOI: 10.1007/s10470-025-02317-z]. Las funciones de las compuertas AND, OR, NOT y XOR se resumen en la *Tabla 1*.

Tabla 1. Compuertas lógicas básicas y sus operaciones

|  |  |  |  |
| --- | --- | --- | --- |
| **Compuerta** | **Símbolo** | **Operación lógica** | **Tabla de verdad (A, B)** |
| AND | A ∧ B | A y B | 0 0 → 0 0 1 → 0 1 0 → 0 1 1 → 1 |
| OR | A ∨ B | A o B | 0 0 → 0 0 1 → 1 1 0 → 1 1 1 → 1 |
| NOT | ¬A | No A | 0 → 1 1 → 0 |
| XOR | A ⊕ B | A distinto B | 0 0 → 0 0 1 → 1 1 0 → 1 1 1 → 0 |

La *Figura 1* muestra los símbolos gráficos utilizados para representar estas compuertas.

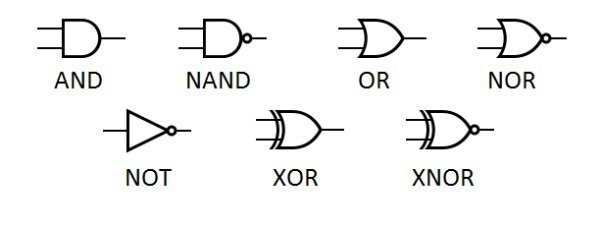


Figure 1. Símbolos de compuertas lógicas

En cuanto a Hamayun et al. (2023), muestran que las tecnologías de memristores pueden permitir la implementación de circuitos para la construcción de sumadores digitales que presentan elevada ratio de fiabilidad, bajo consumo energético y el bajo coste de la unidad aritmética y lógica [DOI:10.1016/j.sse.2023.108704].

**3.1.2.- Circuitos Secuenciales**

Zhang et al. (2023) discuten que los circuitos secuenciales tienen dentro de sí la memoria, es decir, su salida depende no sólo de las entradas que estén en estado presente, sino también de las condiciones que haya anterior y que por lo tanto son imprescindibles en control, el cronometrado, el almacenamiento y la sincronización [DOI: 10.1007/s11227-023-05134-1].

El conjunto de los principales bloques secuenciales es biestable (flip-flops), contadores, registros y máquinas de estado finito (FSM). Khan et al. (2023), se refieren a los flip-flops de tipo SR, D, T y JK como el fundamento de estos circuitos [DOI: 10.17762/ijritcc.v11i6.7302]. Los diferentes tipos de flip-flops se describen en la *Tabla 2*, junto con sus entradas, salidas y funciones.

Tabla 2. Tipos de flip-flops y sus funciones

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Flip-Flop** | **Entradas** | **Salidas** | **Función principal** |
| SR | S, R | Q, Q' | Set/Reset |
| D | D, CLK | Q, Q' | Almacena dato en flanco |
| T | T, CLK | Q, Q' | Conmuta si T=1 |
| JK | J, K, CLK | Q, Q' | Versión robusta de SR |

La *Figura 2* muestra visualmente la estructura y funcionamiento básico de estos flip-flops.

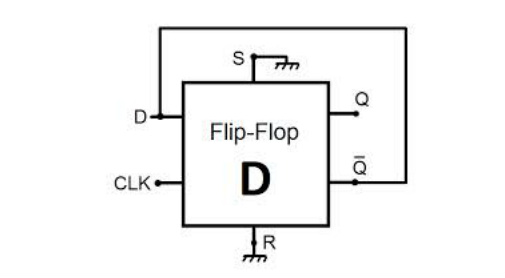


Figura 2. Representación de flip-flop tipo SR, D, T y JK

Singh et al. (2023) destacan que las tecnologías QCA permiten mediante el diseño de flip-flops y contadores un consumo energético y una densidad muy buenas, ideales para IoT y la computación embebida [DOI: 10.11648/j.jeee.20251301.14].

En la *Figura 3* se ilustra un ejemplo de diseño de circuitos secuenciales aplicando tecnología QCA.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

Figura 3. Diseño de circuitos secuenciales con tecnología QCA

**3.2.- Comparación y Diseño de Sistemas Combinacionales Básicos**

**3.2.1.- Sumadores**

Lo que los autores Singh et al. (2020) ponen de manifiesto es que los sumadores son necesarios para la ejecución de operaciones aritméticas binarias mediante un medio sumador para la suma simple o un sumador completo para la suma con acarreo [DOI: 10.1007/s00034-020-01550-3]. La *Tabla 3* resume las entradas, salidas y ecuaciones lógicas utilizadas en los sumadores más comunes.

Tabla 3. Tipos de sumadores y sus ecuaciones

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de sumador** | **Entradas** | **Salidas** | **Ecuaciones** |
| Half-Adder | A, B | S, Cout | S = A ⊕ B, Cout = A ∧ B |
| Full-Adder | A, B, Cin | S, Cout | S = A ⊕ B ⊕ Cin, Cout = (A ∧ B) ∨ (Cin ∧ (A ⊕ B)) |

Un circuito sumador es un circuito que permite la operación aritmética de la suma entre dos bits. Como se observa en la figura 4 existen dos tipos fundamentales de los mismos:

* **Half Adder (Sumador Medio):** Se encarga de sumar dos bits (A y B), produce una suma (S) y la salida de acarreo (C) [DOI: 10.1007/s00034-020-01550-3].
* **Full Adder (Sumador Completo):** Suma 3 bits (A, B y Cin) considerando el acarreo de entrada y es básico para operaciones en multibit [DOI: 10.1007/s00034-020-01550-3].

La *Figura 4* presenta los diagramas de los circuitos de un medio sumador y un sumador completo.

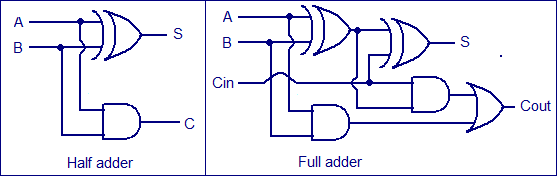


Figura 4. Circuitos de sumador medio y sumador completo

Hamayun et al. (2023) evidencian que la técnica Gate Diffusion Input (GDI) reduce la complejidad, retardo y consumo energético en sumadores [DOI: 10.1007/s00034-020-01550-3].

**3.2.2.- Restadores**

Kaur et al. (2024) afirman que los restadores se pueden implementar utilizando sumadores que ya están adaptados por complemento a dos, técnica la cual optimiza reutilización de hardware y recursos [DOI: 10.3390/jlpea14040057]. En la *Tabla 4* se comparan dos estrategias de diseño para circuitos restadores: clásico y por complemento a dos.

Tabla 4. Comparación entre restador clásico y por complemento a dos

|  |  |  |
| --- | --- | --- |
| **Característica** | **Restador Clásico** | **Restador por Complemento a Dos** |
| **Estrategia de diseño** | Implementación directa con compuertas | Uso de sumador + inversión de bits + acarreo inicial |
| **Componentes principales** | XOR, AND, NOT, compuertas para resta directa | Sumador binario, inversor, generador de acarreo inicial |
| **Reutilización de hardware** | Limitada | Alta (reutiliza el sumador existente) |
| **Complejidad de implementación** | Media | Baja a media (dependiendo del sumador base) |
| **Eficiencia en hardware** | Menor eficiencia | Mayor eficiencia, menos elementos redundantes |
| **Velocidad** | Variable según diseño | Alta en arquitecturas optimizadas |
| **Uso en ALU modernas** | Menos común | Muy común (especialmente en microprocesadores y DSP) |

Los restadores efectúan la operación inversa a la de la suma. Al igual que los sumadores se consideran las versiones más simples:

* **Half Subtractor (Restador medio):** Resta dos bits y devuelve la diferencia y el bit de préstamo [DOI: 10.1016/j.sse.2023.108704].
* **Full Subtractor (Restador Completo):** Realiza la operación de una resta considerando un bit de préstamo anterior, análogo al acarreo en el caso de los sumadores [DOI: 10.1016/j.sse.2023.108704].

La *Figura 5* ilustra las configuraciones básicas de medio restador y restador completo.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

Figure 5. Circuitos de medio restador y restador completo

**3.2.3.- Multiplexores**

Lee et al. (2021) indican que los multiplexores permiten que una entrada sea seleccionada entre varias entradas en función de las señales de control, siendo importantes para direccionar la información. Los multiplexores estocásticos permiten que la precisión y eficiencia energética de los circuitos lógicos en FPGA sean mejorados [DOI: 2108.12326]. La *Tabla 5* muestra las entradas, señales de control y salidas del multiplexor 2:1.

Tabla 5. Configuración de un multiplexor 2:1

|  |  |  |  |
| --- | --- | --- | --- |
| MUX | Entradas | Señales de control | Salida |
| 2:1 | A, B | S | Y |

En la *Figura 6* se representa gráficamente el funcionamiento de un MUX 2:1.

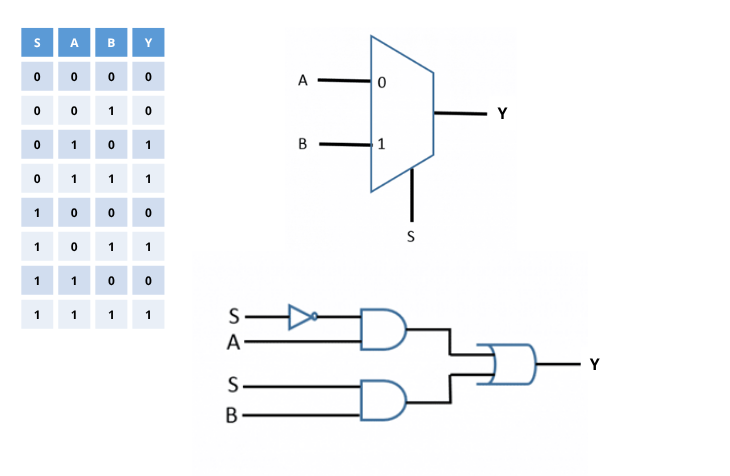


Figura 6. Multiplexor 2:1

**3.3.- Automatización del Diseño Lógico**

Lo evidencian Wang et al. (2021) quien como se aduce "el diseño digital también puede ser automatizado aprovechando algoritmos de autoaprendizaje para optimizar parámetros como tiempo, área, consumo, latencia" [DOI: 10.1109/DAC18074.2021.9586094].

Tal como describen Zhang et al. (2022) en cuanto a NovelRewrite "en vez de rediseñar completamente una red lógica existente, el proceso ahora puede ser optimizado facilitando iteraciones entre mejora total de diseño y mejora de diseño directo logrando mejorar los parámetros de diseño" [DOI: 10.1145/3489517.3530462].

Miller et al. (2023) exponen "Recurrent CircuitSAT Sampling (RCSS) extensible a una arquitectura de computación paralela basada en GPUs, el método asegura que la verificación funcional de circuitos complejos sea un proceso más rápido en cuanto a conjuntos" [DOI: 2502.21226].

Li et al. (2023) junto con Chen et al. (2024) proponen que RL-MUL 2.0 y PrefixLLM emplean modelos de lenguaje, así como aprendizaje automático, para poder sintetizar estructuras aritméticas, logrando alta eficiencia y menor tiempo de desarrollo [DOI: 10.1145/3711850], [DOI: 2412.02594].

Dubey et al. (2024) proponen ArithTreeRL para lograr construir árboles aritméticos adaptativos que optimizan la profundidad y la capacidad paralela, que le aportan de este modo beneficios al procesamiento digital de señales [DOI: 2405.06758].