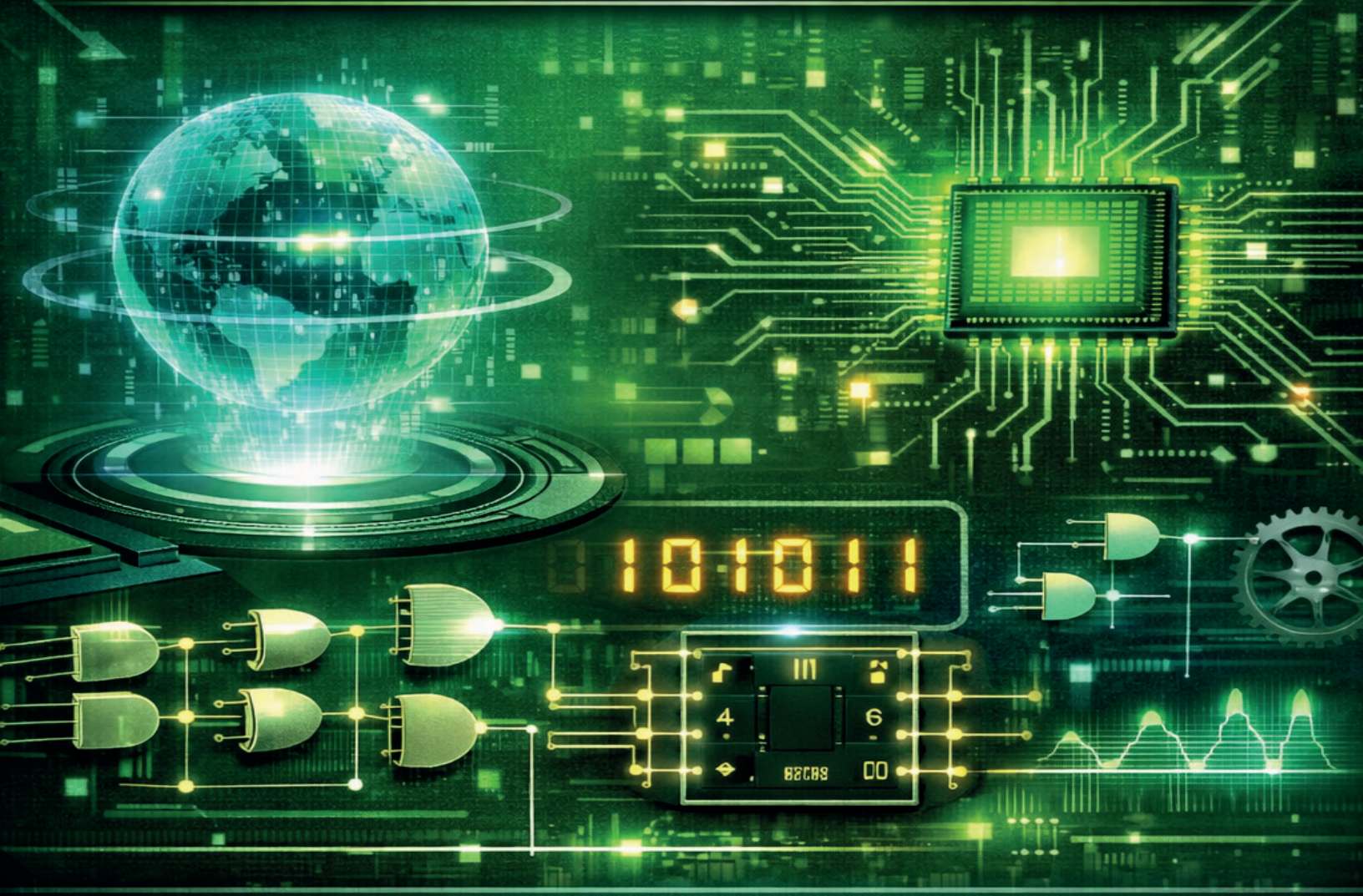
ELECTRÓNICA DIGITAL

Sistemas de Numeración, Lógica Combinacional y Secuencial



Irene Tustón, Christiam Núñez,
Mónica Carrión & Luis Sánchez

EDITORIAL
SAGA

Electrónica Digital

Sistemas de Numeración, Lógica Combinacional y Secuencial

Autores:

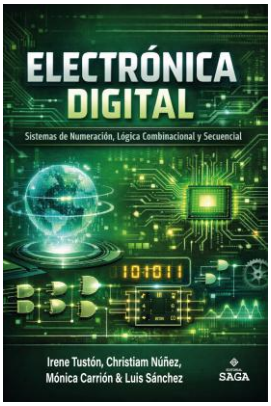
Irene Tustón Torres

Christiam Xavier Núñez Zavala

Mónica Alexandra Carrión Cevallos

Luis Miguel Sánchez Muyulema

EDITORIAL SAGA



Datos bibliográficos

ISBN:	**978-9907-803-03-7**
Título del libro:	Electrónica Digital Sistemas de Numeración, Lógica Combinacional y Secuencial
Autores:	Tustón Torres, Irene Núñez Zavala, Christiam Xavier Carrión Cevallos, Mónica Alexandra Sánchez Muyulema, Luis Miguel
Editorial:	SAGA
Materia:	621.38 - Electrónica. Ingeniería de las comunicaciones
Público objetivo:	Profesional / académico
Publicado:	2026-01-27
Número de edición:	1
Tamaño:	5Mb
Soporte:	Libro digital descargable
Formato:	Pdf (.pdf)
Idioma:	Español
DOI:	https://doi.org/10.63415/saga.2026.59

Hecho en Ecuador / Made in Ecuador

Sinopsis

Electrónica Digital: Sistemas de Numeración, Lógica Combinacional y Secuencial presenta una visión integral de los principios que sustentan el diseño y análisis de sistemas digitales contemporáneos, articulando fundamentos matemáticos, representación de la información y técnicas de implementación con rigor didáctico. El texto aborda los sistemas de numeración y sus conversiones con claridad operativa, integra la aritmética binaria y los complementos como herramientas formales, y desarrolla los códigos digitales empleados en la comunicación y el procesamiento de datos. La obra avanza hacia el álgebra de Boole, las funciones lógicas, las tablas de verdad y los métodos de simplificación, vinculándolos con compuertas y familias lógicas para una comprensión funcional del hardware. Posteriormente, el análisis combinacional se construye mediante sumadores, comparadores, codificadores, decodificadores, multiplexores y demultiplexores, fortaleciendo el razonamiento estructurado mediante ejemplos y actividades. El estudio de la lógica secuencial consolida señales de reloj, multivibradores, celdas binarias, flip-flops, diagramas de estado y máquinas secuenciales, con énfasis en sincronismo y comportamiento temporal. La propuesta didáctica combina teoría, práctica y autoevaluación, favoreciendo la aplicación en automatización, control y telecomunicaciones. Su enfoque progresivo y sistemático apoya la formación técnica y universitaria, promueve precisión conceptual y fomenta competencias para el diseño eficiente de soluciones digitales confiables.

Palabras clave: electrónica digital; sistemas de numeración; lógica combinacional; lógica secuencial; álgebra de boole; circuitos digitales

Synopsis

Digital Electronics: Number Systems, Combinational and Sequential Logic offers a comprehensive view of the principles that underpin modern digital system design and analysis, integrating mathematical foundations, data representation, and implementation techniques with strong pedagogical rigor. The book addresses number systems and conversions with operational clarity, develops binary arithmetic and complements as formal tools, and presents digital codes used for communication and data processing. It then advances to Boolean algebra, logical functions, truth tables, and simplification methods, connecting them with logic gates and logic families for a functional understanding of hardware. Combinational analysis is built through adders, comparators, encoders, decoders, multiplexers, and demultiplexers, strengthening structured reasoning through examples and guided activities. The treatment of sequential logic consolidates clock signals, multivibrators, binary cells, flip-flops, state diagrams, and sequential machines, emphasizing synchronism and temporal behavior. The didactic proposal blends theory, practice, and self-assessment, supporting applications in automation, control, and telecommunications. Its progressive and systematic approach supports technical and higher education, promotes conceptual precision, and fosters competencies for the efficient design of reliable digital solutions.

Keywords: digital electronics; number systems; combinational logic; sequential logic; boolean algebra; digital circuits

EDITORIAL
SAGA

Contenido

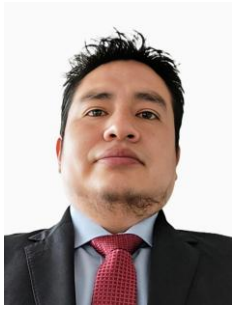
PRESENTACIÓN-ENMARCACIÓN

A. AUTORES

Datos Autor	
Nombres y apellidos	Irene Tustón Torres
Título de tercer nivel (información registrada en la SENESCYT)	Ingeniero en Electrónica, Telecomunicaciones y Redes
Título de cuarto nivel (información registrada en la SENESCYT)	Magister en Sistemas de Telecomunicaciones
ORCID:	https://orcid.org/0009-0009-3949-534X
Correo electrónico	itustn@byupathway.edu
Resumen hoja de vida	Ingeniera en Electrónica Telecomunicaciones y Redes y Magister en Sistemas de Telecomunicaciones. Actualmente estudiante del Bachelor in Family History en Brigham Young University. Experiencia profesional como técnico computacional en Self Reliant Institute. Experiencia docente en la Escuela Superior Politécnica de Chimborazo, Universidad Nacional de Chimborazo e Instituto Superior Tecnológico Carlos Cisneros. Participación en proyectos de planificación institucional, vinculación e investigación.

Datos Autor	
Nombres y apellidos	Christiam Xavier Núñez Zavala
Título de tercer nivel (información registrada en la SENESCYT)	Ingeniero en Electrónica, Control y Redes Industriales
Título de cuarto nivel (información registrada en la SENESCYT)	Magíster en Sistemas de Control y Automatización Industrial.
ORCID:	https://orcid.org/0000-0001-8162-5616
Correo electrónico	cnunez@unach.edu.ec
Resumen hoja de vida	Ingeniero en Electrónica, Control y Redes Industriales de la Escuela superior Politécnica de Chimborazo en el año de 2011, Magister en Sistemas de Control y Automatización Industrial en la Superior Politécnica de Chimborazo año de 2017. Jefe de Automatización en la empresa Ecuatoriana de Cerámica planta Azulejos de la Ciudad de Riobamba. Encargado de soporte de líneas de comunicación Grupo Cerámico. Docente de la Escuela Superior Politécnica de Chimborazo en el año 2012 en la Facultad de Informática y Electrónica en las carreras de: Ingeniería Electrónica en Control y Redes Industrial, Ingeniería Electrónica en Telecomunicaciones. Docente en la Universidad Nacional de Chimborazo en la Unidad de Admisión y Nivelación en el año 2017. Analista del departamento de Tecnologías de la Información y Comunicación de la Universidad nacional de Chimborazo. Docente de la Senescyt en el Instituto Tecnológico Carlos Cisneros de la Ciudad de Riobamba en el año 2018. Actualmente Docente de la Universidad Nacional de Chimborazo en la Facultad de Ciencias de la Educación Humanas y Tecnologías, Carrera de Pedagogía de la Informática. Miembro del grupo de Investigación Umayuk de la Universidad nacional de Chimborazo.

Datos Autor	
Nombres y apellidos	Mónica Alexandra Carrión Cevallos
Título de tercer nivel (información registrada en la SENESCYT)	Ingeniera de Mantenimiento
Título de cuarto nivel (información registrada en la SENESCYT)	Magister en seguridad industrial, mención salud ocupacional Magister en matemática aplicada, mención matemática computacional
ORCID:	https://orcid.org/0000-0003-0928-1307?lang=es
Correo electrónico	moncacevallos@yahoo.es
Resumen hoja de vida	Ingeniera en Mantenimiento, Magíster en Seguridad Industrial y Magíster en Matemática Aplicada, con más de 15 años de experiencia como docente universitaria en la Escuela Superior Politécnica de Chimborazo (ESPOCH) y en el Instituto Superior Universitario Carlos Cisneros, desempeñándome en docencia técnica y en las áreas de Matemática e Investigación; he participado activamente en proyectos de investigación, presentado ponencias internacionales en Argentina y Perú, y publicado cuatro artículos científicos en revistas indexadas, consolidando un perfil académico y profesional orientado a la excelencia, la innovación y el aporte al desarrollo científico y educativo.

Datos Autor	
Nombres y apellidos	Luis Miguel Sánchez Muyulema
Título de tercer nivel (información registrada en la SENESCYT)	Ingeniero en Electrónica Control y Redes Industriales
Título de cuarto nivel (información registrada en la SENESCYT)	Magister en Sistemas de Control y Automatización Industrial
ORCID:	https://orcid.org/0000-0001-5078-3734
Correo electrónico	luis.sanchez@ute.edu.ec
Resumen hoja de vida	Ingeniero en Control y Redes Industriales, con una Maestría en Sistemas de Control y Automatización Industrial. Docente de la carrera de Mecatrónica en la Universidad UTE, con una amplia experiencia en educación STEAM y robótica educativa. Ha liderado proyectos de investigación en automatización industrial y obtenido múltiples reconocimientos en competencias internacionales de robótica como RobotChallenge China 2024. Su labor está enfocada en la investigación, desarrollo tecnológico y la promoción de la robótica, contribuyendo al crecimiento de la carrera y motivando a las nuevas generaciones de ingenieros.

B. INTRODUCCIÓN

Descripción

La asignatura de Electrónica Digital es de naturaleza teórico–práctica y se complementa con el uso de software especializado para el análisis y simulación de circuitos. Su propósito es proporcionar a los estudiantes los fundamentos necesarios para diseñar y construir sistemas digitales mediante el empleo de compuertas lógicas, circuitos combinacionales y circuitos secuenciales. A través de actividades prácticas y el estudio de dispositivos digitales modernos, el estudiante desarrolla la capacidad de comprender, implementar y evaluar soluciones electrónicas aplicadas a distintos contextos tecnológicos.

Competencias

El aporte de la asignatura de Electrónica Digital al egresado de la carrera de Tecnología Superior en Electrónica se refleja en la capacidad para instalar, gestionar y brindar mantenimiento a sistemas electrónicos, de control y telecomunicaciones. Esto incluye el dominio de los principios y aplicaciones de los semiconductores, así como su integración en procesos de automatización industrial y en la transmisión y recepción de señales en sistemas de comunicación analógicos y digitales.

Además, la asignatura busca fortalecer las siguientes habilidades específicas:

- Identificar, interpretar y aplicar los principios fundamentales de la electrónica digital.
- Reconocer y analizar el comportamiento de los elementos activos y pasivos que intervienen en la construcción de circuitos.
- Utilizar adecuadamente herramientas de colaboración digital para apoyar procesos de diseño, documentación y simulación.

Objetivos de la asignatura

La asignatura está orientada a que el estudiante desarrolle las competencias necesarias para aprender, construir, analizar y resolver problemas mediante el uso de técnicas digitales,

fortaleciendo un pensamiento analítico, crítico y creativo. En este sentido, se plantean los siguientes objetivos:

- Capacitar al estudiante, de manera teórica y experimental, en técnicas de análisis y diseño de circuitos digitales combinacionales.
- Relacionar el avance tecnológico actual con las prácticas e innovaciones presentes en la industria.
- Fomentar el análisis lógico y la capacidad de razonamiento del estudiante para la solución eficiente de problemas digitales.
- Desarrollar habilidades de pensamiento lógico y estructurado mediante la aplicación de métodos propios de la electrónica digital.

UNIDAD 1: SISTEMAS DE NUMERACIÓN

1.1 Conversiones

La información que se procesa en cualquier sistema digital debe estar representada numéricamente. Para ello, es necesario utilizar un sistema de numeración que se ajuste a las características intrínsecas de este tipo de señales (Floyd, 2006).

Un sistema de numeración se define como un conjunto de símbolos capaces de representar cantidades numéricas. Asimismo, la base del sistema corresponde al número de símbolos distintos que se emplean para expresar dichas cantidades. Cada uno de estos símbolos recibe el nombre de dígito (Floyd, 2006). Los sistemas de numeración más utilizados se muestran en la Tabla 1.

Tabla 1

Sistemas de numeración más utilizados

Sistema de numeración	Dígitos
Sistema decimal o de base 10	Consta de diez dígitos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
Sistema binario o de base 2	Consta de dos dígitos (bits): {0, 1}
Sistema octal o de base 8	Consta de ocho dígitos: {0, 1, 2, 3, 4, 5, 6, 7}
Sistema hexadecimal o de base 16	Consta de dieciséis dígitos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Fuente: Floyd (2006).

1.1.1 Conversión de decimal a binario

Para convertir un número decimal a su equivalente en binario, se deben seguir los pasos descritos por Floyd (2006):

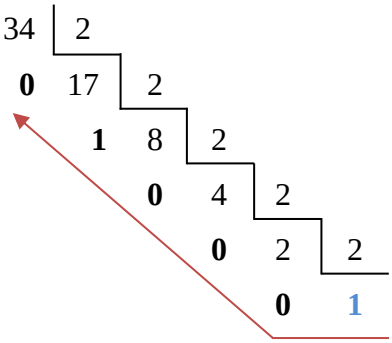
1. Realizar divisiones sucesivas entre el número decimal y la base del sistema binario (2), registrando en cada operación el cociente y el residuo. El proceso continúa hasta que el cociente final sea 0.
2. Formar el número binario tomando los residuos obtenidos en orden inverso, es decir, desde el último residuo calculado hasta el primero. Esta secuencia constituye la representación binaria del número original.

Ejemplo

Convertir el número decimal 34 en su equivalente número binario.

Solución

Para obtener el equivalente binario, dividimos sucesivamente entre 2 y registramos los residuos:



$34_{10} = 100010_2$

1.1.2 Conversión de binario a decimal

Para convertir un número binario a un número decimal, debemos seguir los siguientes pasos descritos por Floyd (2006):

1. Multiplicar cada bit por la potencia de 2 que le corresponde (2^n … 2^3, 2^2, 2^1, 2^0), iniciando desde el bit menos significativo (derecha) hacia la izquierda.
2. Sumar todos los productos parciales obtenidos en el paso anterior. El resultado final es el valor decimal del número binario.

Ejemplo

Convertir el número binario 1011 en su equivalente número decimal.

Solución

Asignamos la potencia de 2 correspondiente a cada bit, desde la derecha:

Potencias de 2 2^3 2^2 2^1 2^0

Número binario 1 0 1 1 = $(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

= 8 + 0 + 2 + 1 = **11**

$1011_2 = 11_{10}$

1.1.3 Conversión de decimal a hexadecimal

Para convertir un número decimal a su equivalente en hexadecimal, se deben seguir los pasos descritos por Tocci (2011):

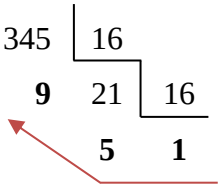
1. Realizar divisiones sucesivas entre el número decimal y la base del sistema hexadecimal (16), registrando el cociente y el residuo en cada operación. El proceso continúa hasta que el cociente final sea menor que 16.
2. Formar el número hexadecimal tomando los residuos y el último cociente en orden inverso, es decir, desde el último valor obtenido hasta el primero.

Ejemplo

Convertir el número decimal 345 en su equivalente número hexadecimal.

Solución

Para obtener el equivalente binario, dividimos sucesivamente entre 2 y registramos los residuos:



$$345_{10} = 159_{16}$$

1.1.4 Conversión de hexadecimal a decimal

Para convertir un número hexadecimal a un número decimal, debemos seguir los siguientes pasos descritos por Tocci (2011):

1. Multiplicar cada dígito por la potencia de 16 que le corresponde (16^n … 16^3, 16^2, 16^1, 16^0), iniciando desde el dígito menos significativo (derecha).
2. Sumar los productos parciales obtenidos en el paso anterior para obtener el número decimal final.

Ejemplo

Convertir el número hexadecimal 78 en su equivalente número decimal.

Solución

Asignamos la potencia de 16 correspondiente a cada dígito:

Potencias de 16 16^1 16^0

Número binario 7 $8 = (7 \times 16^1) + (8 \times 16^0)$

$= 112 + 8 =$ **120**

$$78_{16} = 120_{10}$$

1.1.5 Conversión de binario a hexadecimal

Para convertir un número binario a su equivalente hexadecimal, se deben seguir los pasos descritos por Floyd (2006):

1. Agrupar los dígitos binarios de cuatro en cuatro, comenzando desde la derecha. Si el último grupo no contiene cuatro bits, se agregan ceros a la izquierda para completarlo.
2. Sustituir cada grupo de cuatro bits por su equivalente hexadecimal, según la tabla estándar de conversiones binarias.
3. Formar el número hexadecimal leyendo los dígitos resultantes de izquierda a derecha.

Ejemplo

Convertir el número binario 01001111 en su equivalente número hexadecimal.

Solución

0 1 0 0 1 1 1 1

4 **F**

$$01001111_2 = 4F_{16}$$

1.1.6 Conversión de hexadecimal a binario

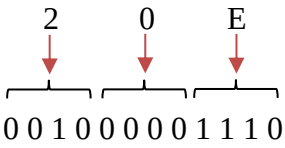
Para convertir un número hexadecimal a un número binario, debemos seguir los siguientes pasos descritos por Tocci (2011):

1. Sustituir cada dígito hexadecimal por su correspondiente grupo binario de cuatro bits, utilizando la tabla estándar de equivalencias (0–F).
2. Formar el número binario concatenando los grupos obtenidos, manteniendo el orden original de los dígitos hexadecimales (de izquierda hacia la derecha).

Ejemplo

Convertir el número hexadecimal 20E en su equivalente número binario.

Solución



$20E_{16} = 001000001110_2$

1.2 Aritmética Binaria

1.2.1 Suma Binaria

Las cuatro reglas fundamentales para sumar dígitos binarios, según Floyd (2006), son las siguientes:

- 0 + 0 = 0 Suma 0 con acarreo 0
- 0 + 1 = 1 Suma 1 con acarreo 0
- 1 + 0 = 1 Suma 1 con acarreo 0
- 1 + 1 = 10 Suma 0 con acarreo 1

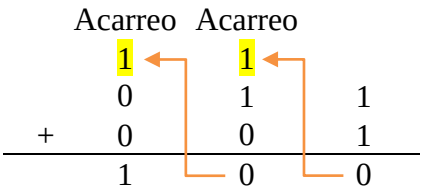
Estas reglas constituyen la base de todas las operaciones de suma en sistemas digitales.

Cuando se suman números binarios de varias cifras, la última regla (1 + 1 = 10) produce una suma parcial de 0 y un acarreo de 1 que se transfiere a la siguiente columna hacia la izquierda. Este procedimiento es equivalente al acarreo en el sistema decimal, (Tocci, 2011).

Ejemplo

Sumar los números binarios 011 + 001.

Solución



La suma de 011 + 001 es 100

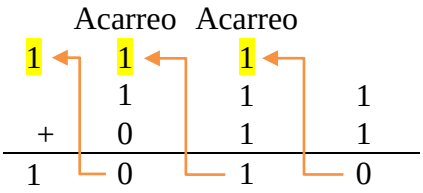
Cuando en una suma binaria se genera un acarreo igual a 1, puede presentarse una situación en la que deban sumarse tres bits simultáneamente: un bit del primer número, un bit del segundo y el bit de acarreo proveniente de la columna anterior. En este caso, el procedimiento se realiza siguiendo las reglas establecidas por Floyd (2006).

- 1 + 0 + 0 = 01 Suma 1 con acarreo 0
- 1 + 1 + 0 = 10 Suma 1 con acarreo 0
- 1 + 0 + 1 = 10 Suma 1 con acarreo 0
- 1 + 1 + 1 = 11 Suma 1 con acarreo 1

Ejemplo

Sumar los números binarios 111 + 011.

Solución



La suma de 111 + 011 es 1010

1.2.2 Resta Binaria

Las cuatro reglas fundamentales para realizar restas en sistema binario, según Floyd (2006), son:

- 0 – 0 = 0
- 1 – 1 = 0
- 1 – 0 = 1
- 10 – 1 = 1 con acarreo negativo de 1

Ejemplo

Restar los números binarios 11 - 10.

Solución

	1	1
–	1	0
	0	1

La resta de 11 – 10 es 01

Cuando se restan números, algunas veces se genera un acarreo negativo que pasa a la siguiente columna de la izquierda. En binario, sólo se produce un acarreo negativo cuando se intenta restar 1 de 0. En este caso, cuando se acarrea un 1 a la siguiente columna de la izquierda, en la columna que se está restando se genera un 10, y entonces debe aplicarse la última de las cuatro reglas enumeradas (Floyd, 2006).

Cuando se restan números binarios, en ocasiones se genera un acarreo negativo, también llamado préstamo, que debe trasladarse a la columna inmediata de la izquierda. En el sistema binario, este préstamo ocurre únicamente cuando se intenta realizar la operación:

$$0 - 1$$

Dado que 0 no puede restarse con 1 en base 2 sin apoyo de otra columna, se toma prestado un 1 de la columna situada a la izquierda. Ese préstamo equivale a agregar 2 en términos decimales, lo cual en binario se representa como:

$$0 \longrightarrow 10_2$$

Una vez convertido en 10_2, puede aplicarse la última regla fundamental de la resta binaria:

$$10 - 1 = 1$$

Por lo tanto, el préstamo genera dos efectos simultáneos:

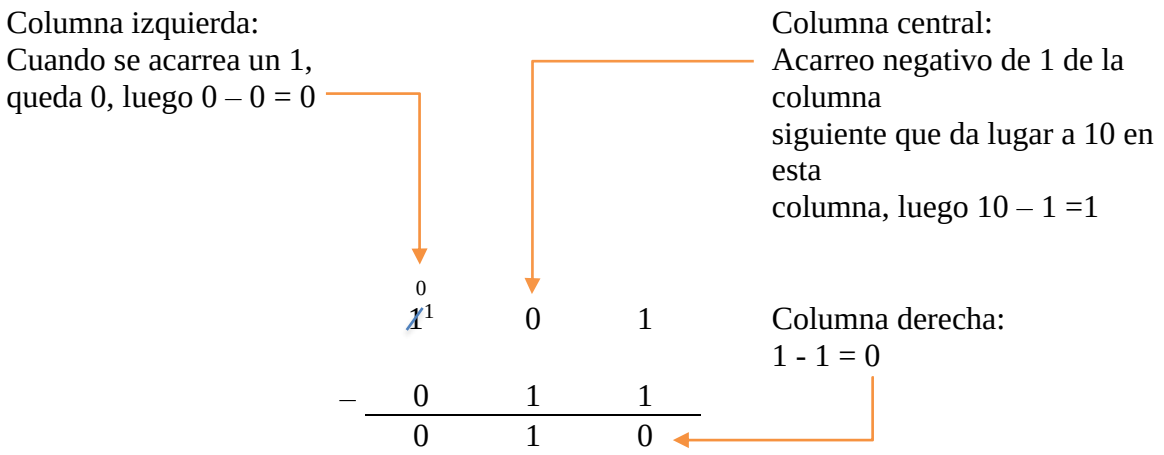
1. La columna donde ocurre la resta se transforma en 10_2.
2. Se envía un acarreo negativo (préstamo) a la siguiente columna izquierda, que deberá considerarse al continuar la operación.

Este procedimiento es análogo al préstamo en el sistema decimal, pero adaptado a la base 2 (Floyd, 2006).

Ejemplo

Restar los números binarios 101 – 011.

Solución



El resultado de restar 101 – 011 es 010.

1.2.3 Multiplicación Binaria

Las cuatro reglas básicas de la multiplicación de bits, según Floyd (2006) son:

- 0 x 0 = 0
- 0 x 1 = 0
- 1 x 0 = 0
- 1 x 1 = 1

La multiplicación de números binarios se realiza siguiendo el mismo procedimiento empleado en el sistema decimal. Para ello:

1. Se generan los productos parciales, multiplicando cada bit del multiplicador por cada bit del multiplicando.
2. Cada producto parcial se desplaza una posición hacia la izquierda respecto del anterior, tal como ocurre con los decimales cuando se avanza a una nueva cifra del multiplicador.
3. Finalmente, se suman todos los productos parciales, aplicando las reglas de suma binaria previamente estudiadas (Floyd, 2006).

Este método, aunque aplicado en base 2, conserva la misma estructura lógica que la multiplicación en base 10.

Ejemplo

Multiplicar los números binarios 111 x 101.

Solución

				1	1	1
			x	1	0	1
Productos parciales				1	1	1
			0	0	0	
	+	1	1	1		
	1	0	0	0	1	1

La multiplicación de 111 x 101 es 100011.

1.2.4 División Binaria

La división binaria sigue el mismo procedimiento que la división decimal (Floyd, 2006).

Ejemplo

Dividir los números binarios 1100 ÷ 100.

Solución

1	1	0	0	1	0	0
1	0	0	↓	1	1	
0	1	0	0			
-	1	0	0			
	0	0	0			

La división de 1100 ÷ 100 es 11.

1.3 Complementos

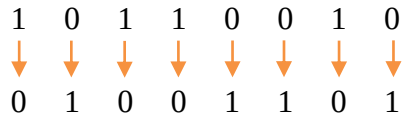
1.3.1 Complemento a 1

El complemento a 1 de un número binario se obtiene intercambiando cada bit, es decir, reemplazando los valores 1 por 0 y los valores 0 por 1 (Tocci, 2011).

Ejemplo

Determinar el complemento a 1 del número binario 10110010.

Solución



El complemento a 1 del número binario 10110010 es 01001101.

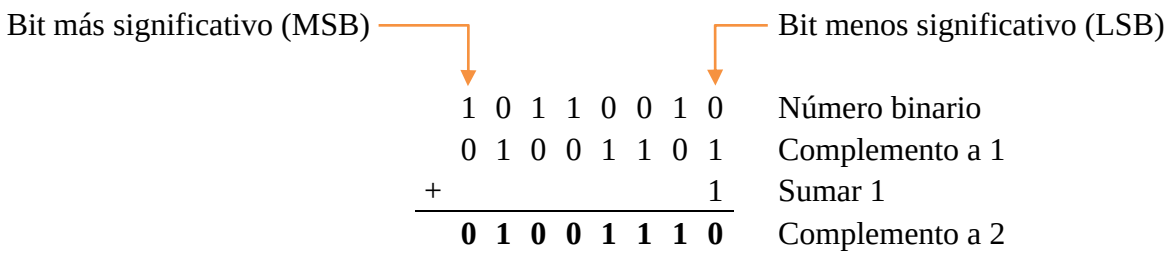
1.3.2 Complemento a 2

Método I: El complemento a 2 de un número binario se obtiene sumando 1 al bit menos significativo (LSB) del complemento a 1 (Tocci, 2011).

Ejemplo

Determinar el complemento a 2 del número binario 10110010.

Solución



El complemento a 2 del número binario 10110010 es 01001110.

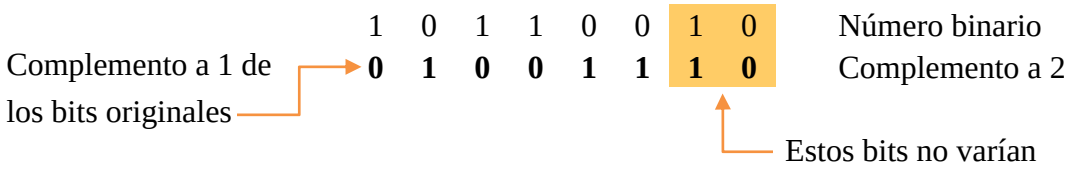
Método II: Un método alternativo para obtener el complemento a 2 de un número binario es el siguiente:

1. Se empieza por la derecha con el LSB y se escriben los bits como están hasta encontrar el primer 1, incluido éste.
2. Se calcula el complemento a 1 de los bits restantes (Tocci, 2011).

Ejemplo

Determinar el complementa a 2 del mismo número binario 10110010.

Solución



El complemento a 2 del número binario 10110010 es 01001110.

1.4 Códigos

1.4.1 Código Decimal Binario (BCD)

El código decimal binario (*BCD, Binary Coded Decimal*) es un sistema de representación en el cual cada dígito decimal se codifica de manera independiente mediante un grupo de bits binarios. Dado que el sistema BCD emplea únicamente diez combinaciones de código, la conversión entre números decimales y su representación BCD resulta sencilla y directa.

Debido a que los seres humanos leen y escriben naturalmente en el sistema decimal, el código BCD constituye una interfaz eficiente entre el mundo decimal y los sistemas digitales binarios. Ejemplos comunes de esta aplicación se encuentran en dispositivos de entrada, como los teclados, y en dispositivos de salida, como los visualizadores digitales (Tocci, 2011).

1.4.1.1 Código 8421

El código 8421 es un tipo de código decimal binario (BCD) en el cual cada dígito decimal, comprendido entre 0 y 9, se representa mediante un grupo de cuatro bits binarios. La denominación *8421* hace referencia a los pesos binarios asignados a cada uno de los cuatro bits, correspondientes a 2^3, 2^2, 2^1, 2^0, respectivamente.

La principal ventaja del código 8421 radica en la facilidad de conversión entre los números decimales y su representación binaria, lo que lo convierte en un método ampliamente utilizado en sistemas digitales que interactúan directamente con datos introducidos o visualizados por el

usuario (Tocci, 2011). Las diez combinaciones binarias que representan los dígitos decimales del 0 al 9 se presentan en la Tabla 2.

Tabla 2
Conversión decimal / BCD

Decimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Fuente: Tocci (2011).

El código 8421 es el tipo de código BCD más ampliamente utilizado. Por esta razón, cuando se hace referencia de manera general al código BCD, se entiende que se trata del código 8421, salvo que se indique explícitamente otro tipo de codificación (Tocci, 2011).

Nota: Es importante señalar que, al emplear cuatro bits, es posible representar dieciséis combinaciones binarias, comprendidas entre 0000 y 1111. Sin embargo, en el código BCD 8421 solo se utilizan diez de estas combinaciones, correspondientes a los dígitos decimales del 0 al 9. Las seis combinaciones restantes (1010, 1011, 1100, 1101, 1110 y 1111) no son válidas dentro de este sistema de codificación (Tocci, 2011).

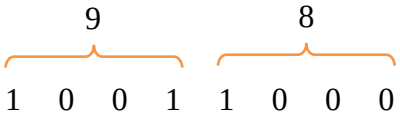
Conversión decimal a BCD

La conversión de un número decimal a su representación en código decimal binario (BCD) se realiza reemplazando cada dígito decimal por su correspondiente código binario de cuatro bits (Carpinelli, 2023).

Ejemplo

Convertir el número decimal 98 a un número en código BCD.

Solución



En código BCD el número es 10011000.

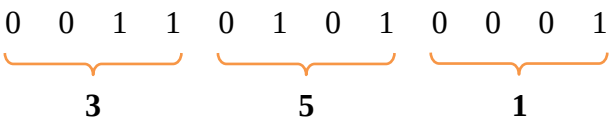
Conversión BCD a decimal

El procedimiento inicia desde la derecha, comenzando por el bit menos significativo (LSB), y consiste en dividir el código binario en grupos de cuatro bits. Luego, se determina el dígito decimal correspondiente a cada grupo, leyendo los resultados de derecha a izquierda (Carpinelli, 2023).

Ejemplo

Convertir el número en código BCD 001101010001 a un número decimal.

Solución



En decimal el número es 351.

1.4.2 Códigos Decimales

1.4.2.1 Código Gray

El código Gray es un código no ponderado y no aritmético, es decir, no posee pesos específicos asignados a las posiciones de sus bits. Su característica más relevante es que entre dos códigos consecutivos solo cambia un bit, lo que reduce significativamente la posibilidad de errores durante las transiciones entre valores adyacentes. Esta propiedad resulta especialmente útil en aplicaciones como los codificadores de posición angular, donde la probabilidad de error aumenta conforme se incrementa el número de cambios simultáneos de bits entre estados consecutivos (Tocci, 2011).

La Tabla 3 presenta el código Gray de cuatro bits correspondiente a los números decimales del 0 al 15. Como referencia, se incluyen también las representaciones binarias equivalentes. Al igual que el sistema binario, el código Gray puede definirse con cualquier número de bits; sin embargo, en este caso particular, se garantiza que entre dos valores sucesivos solo se modifica un único bit, manteniéndose los demás sin cambio. En el código Gray de cuatro bits mostrado, el bit que varía entre valores consecutivos corresponde al tercer bit contado desde la derecha, mientras que los demás permanecen constantes (Tocci, 2011).

Tabla 3
Código Gray de cuatro bits

Decimal	Binario	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Fuente: Tocci (2011).

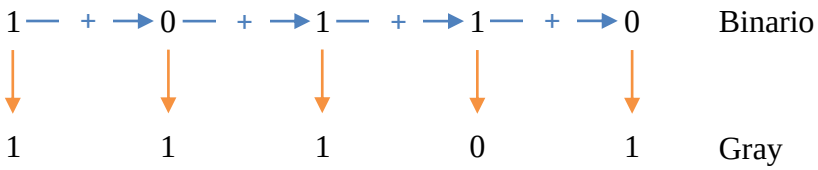
Conversión binario a Gray

El bit más significativo (Most Significant Bit, MSB) del código Gray es idéntico al MSB del número binario correspondiente. A partir de este bit y avanzando de izquierda a derecha, cada bit subsiguiente del código Gray se obtiene sumando módulo 2 (operación XOR) cada par de bits adyacentes del número binario. En este proceso, los acarreos se descartan, ya que la operación se realiza sin consideración de transporte (Tocci, 2011).

Ejemplo

Convertir el número binario 10110 a un número en código Gray.

Solución



En código Gray el número es 11101.

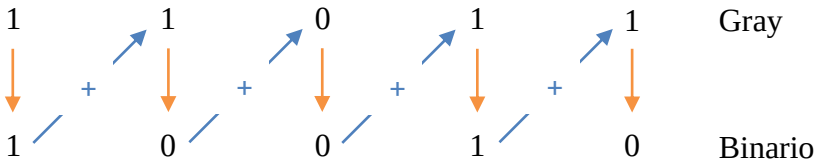
Conversión Gray a binario

El bit más significativo (bit más a la izquierda) en el código binario es el mismo que el correspondiente bit en código Gray. A cada bit del código binario generado se le suma el bit en código Gray de la siguiente posición adyacente. Los acarreos se descartan (Tocci, 2011).

Ejemplo

Convertir el número en código Gray 11011 a un número binario.

Solución



En binario el número es 10010.

1.4.2.2 Códigos alfanuméricos

Para la comunicación digital no solo se requieren números, sino también letras, símbolos y caracteres de control. Los códigos alfanuméricos permiten representar dígitos decimales y caracteres alfabéticos, así como otros símbolos e instrucciones necesarios para la transmisión y el procesamiento de la información (Carpinelli, 2023).

Un código alfanumérico básico debe representar al menos los diez dígitos decimales y las 26 letras del alfabeto, lo que corresponde a 36 símbolos distintos. Esta cantidad requiere un mínimo de seis bits, ya que cinco bits solo permiten 32 combinaciones posibles. Con seis bits se obtienen 64 combinaciones, lo que posibilita incluir, además de números y letras, símbolos adicionales y caracteres de control. El código ASCII es el sistema alfanumérico más ampliamente utilizado en los sistemas digitales (Floyd, 2006).

1.4.2.2.1 Código ASCII

El American Standard Code for Information Interchange (ASCII, código estándar americano para el intercambio de información) es un código alfanumérico universalmente aceptado, que se utiliza en la mayoría de las computadoras y otros equipos electrónicos. La mayor parte de los teclados de computadora se encuentran estandarizados de acuerdo con el código ASCII y, cuando se pulsa una letra, un número o un comando de control, es el código ASCII el que se introduce en la computadora (Tocci, 2011).

El código ASCII dispone de 128 caracteres, los cuales se representan mediante un código binario de 7 bits. No obstante, el código ASCII puede considerarse como un código de 8 bits, en el que el bit más significativo (MSB) siempre es 0. En notación hexadecimal, este código de 8 bits abarca los valores comprendidos entre 00 y 7F. Los primeros 32 caracteres ASCII corresponden a caracteres de control no gráficos, los cuales no se imprimen ni se presentan en pantalla y se utilizan exclusivamente para propósitos de control. Ejemplos de estos caracteres son nulo, avance de línea, inicio de texto y escape. Los demás caracteres son símbolos gráficos que pueden imprimirse o mostrarse en pantalla, e incluyen las letras del alfabeto en mayúsculas y minúsculas, los diez dígitos decimales, los signos de puntuación y otros símbolos comúnmente utilizados (Tocci, 2011).

La Figura 1 presenta un listado del código ASCII, junto con su representación decimal, hexadecimal y binaria para cada carácter y símbolo. En la primera columna se enumeran los nombres de los 32 caracteres de control, codificados en hexadecimal desde 00 hasta 1F, y en las columnas restantes se muestran los símbolos gráficos, representados en hexadecimal desde 20 hasta 7F (Floyd, 2006).

	Símbolos gráficos											
Hex	**Símbolo**	**Dec**	**Binario**	**Hex**	**Nombre**	**Dec**	**Binario**	**Hex**	**Símbolo**	**Dec**	**Binario**	**Hex**
00	space	32	0100000	20	@	64	1000000	40	`	96	1100000	60
01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
0D	-	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

Figura 1 *Código ASCII (American Standard Code for Information Interchange).*

Fuente: Floyd (2006).

Nombre	Decimal	Hex	Tecla	Descripción
NUL	0	00	CTRL @	Carácter nulo
SOH	1	01	CTRL A	Inicio de cabecera
STX	2	02	CTRL B	Inicio de texto
ETX	3	03	CTRL C	Fin de texto
EOT	4	04	CTRL D	Fin de transmisión
ENQ	5	05	CTRL E	Petición
ACK	6	06	CTRL F	Reconocimiento
BEL	7	07	CTRL G	Timbre
BS	8	08	CTRL H	Barra espaciadora
HT	9	09	CTRL I	Tabulador horizontal
LF	10	0A	CTRL J	Avance de línea
VT	11	0B	CTRL K	Tabulador vertical
FF	12	0C	CTRL L	Salto de página
CR	13	0D	CTRL M	Retorno de carro
SO	14	0E	CTRL N	Desplazamiento de salida
SI	15	0F	CTRL O	Desplazamiento de entrada
DLE	16	10	CTRL P	Escape de enlace de datos
DC1	17	11	CTRL Q	Dispositivo de control 1
DC2	18	12	CTRL R	Dispositivo de control 2
DC3	19	13	CTRL S	Dispositivo de control 3
DC4	20	14	CTRL T	Dispositivo de control 4
NAK	21	15	CTRL U	Confirmación negativa
SYN	22	16	CTRL V	Sincronismo
ETB	23	17	CTRL W	Fin del bloque de transmisión
CAN	24	18	CTRL X	Cancelación
EM	25	19	CTRL Y	Fin del dispositivo
SUB	26	1A	CTRL Z	Sustitución
ESC	27	1B	CTRL [	Escape
FS	28	1C	CTRL /	Separador de archivo
GS	29	1D	CTRL]	Separador de grupo
RS	30	1E	CTRL ^	Separador de registro
US	31	1F	CTRL _	Separador de unidad

Figura 2 *Caracteres de control ASCII.*

Fuente: Floyd (2006).

Además de los 128 caracteres ASCII estándar, existen 128 caracteres adicionales, conocidos como ASCII extendido, que fueron adoptados inicialmente por IBM para su uso en las computadoras personales. Debido a la amplia difusión de los PC, este conjunto de caracteres se incorporó en diversas aplicaciones, convirtiéndose en un estándar no oficial ampliamente utilizado (Tocci, 2011).

El código ASCII extendido emplea códigos de 8 bits, cuyos valores en notación hexadecimal se encuentran comprendidos entre 80 y FF. Este conjunto incluye caracteres alfabéticos no ingleses, símbolos monetarios, letras griegas, símbolos matemáticos y diversos caracteres gráficos, tales como gráficos de barras y símbolos sombreados (Tocci, 2011).

La Figura 3 presenta el conjunto de caracteres del código ASCII extendido, junto con sus representaciones decimal y hexadecimal (Floyd, 2006).

Nombre	Decimal	Hex	Nombre	Decimal	Hex	Nombre	Decimal	Hex	Nombre	Decimal	Hex
Ç	128	80	á	160	A0	└	192	C0	α	224	E0
ü	129	81	í	161	A1	┴	193	C1	ß	225	E1
é	130	82	ó	162	A2	┬	194	C2	Γ	226	E2
â	131	83	ú	163	A3	├	195	C3	π	227	E3
ä	132	84	ñ	164	A4	─	196	C4	Σ	228	E4
à	133	85	Ñ	165	A5	┼	197	C5	σ	229	E5
å	134	86	ª	166	A6	╞	198	C6	µ	230	E6
ç	135	87	º	167	A7	╟	199	C7	τ	231	E7
ê	136	88	¿	168	A8	╚	200	C8	Φ	232	E8
ë	137	89	⌐	169	A9	╔	201	C9	Θ	233	E9
è	138	8A	¬	170	AA	╩	202	CA	Ω	234	EA
ï	139	8B	½	171	AB	╦	203	CB	δ	235	EB
î	140	8C	¼	172	AC	╠	204	CC	∞	236	EC
ì	141	8D	¡	173	AD	═	205	CD	φ	237	ED
Ä	142	8E	«	174	AE	╬	206	CE	ε	238	EE
Å	143	8F	»	175	AF	╧	207	CF	∩	239	EF
É	144	90	░	176	B0	╨	208	D0	≡	240	F0
æ	145	91	▒	177	B1	╤	209	D1	±	241	F1
Æ	146	92	▓	178	B2	╥	210	D2	≥	242	F2
ô	147	93	│	179	B3	╙	211	D3	≤	243	F3
ö	148	94	┤	180	B4	╘	212	D4	⌠	244	F4
ò	149	95	╡	181	B5	╒	213	D5	⌡	245	F5
û	150	96	╢	182	B6	╓	214	D6	÷	246	F6
ù	151	97	╖	183	B7	╫	215	D7	≈	247	F7
ÿ	152	98	╕	184	B8	╪	216	D8	°	248	F8
Ö	153	99	╣	185	B9	┘	217	D9	∙	249	F9
Ü	154	9A	║	186	BA	┌	218	DA	·	250	FA
¢	155	9B	╗	187	BB	█	219	DB	√	251	FB
£	156	9C	╝	188	BC	▄	220	DC	^{n}	252	FC
¥	157	9D	╜	189	BD	▌	221	DD	^{2}	253	FD
Pts	158	9E	╛	190	BE	▐	222	DE	■	254	FE
f	159	9F	┐	191	BF	▀	223	DF		255	FF

Figura 3 *Caracteres ASCII extendidos.*

Fuente: Floyd (2006).

ACTIVIDADES

Actividad 1: Ejercicios sobre conversiones.

1. Convertir el número decimal 678 a su correspondiente número binario.
2. Convertir el número binario 1000111 a su correspondiente número decimal.
3. Convertir el número decimal 456 a su correspondiente número hexadecimal.
4. Convertir el número hexadecimal 23A a su correspondiente número decimal.
5. Convertir el número binario 1001111 a su correspondiente número hexadecimal.
6. Convertir el número hexadecimal 23C a su correspondiente número binario.

Actividad 2: Ejercicios sobre aritmética binaria.

1. Realizar las siguientes sumas binarias:
   a. 11 + 11
   b. 111 + 11

2. Realizar las siguientes restas binarias:
   a. 11 – 01
   b. 101 – 110

3. Realizar las siguientes multiplicaciones binarias:
   a) 101 x 11
   b) 1101 x 1010

4. Realizar las siguientes divisiones binarias:
   a. 110 ÷ 11
   b. 110 ÷ 10

Actividad 3: Ejercicios sobre complementos.

1. Determinar el complemento a 1 de los siguientes números binarios:
   a. 10101010

b. 11001100

2. Determinar el complemento a 2 de los siguientes números binarios:
   a. 11100011
   b. 10000111

Actividad 4: Ejercicios sobre códigos.

1. Convertir a BCD los siguientes números decimales:
   a. 35
   b. 98

2. Convertir a decimal los siguientes códigos BCD
   a. 1000
   b. 001101010001

3. ¿Cuál es la representación ASCII de los siguientes caracteres?
   a. K
   b. r
   c. +

Actividad 5: Autoevaluación.

1. Realizar las siguientes conversiones:
   a. Convertir el número decimal 12 a su correspondiente número binario.
   b. Convertir el número binario 1001 a su correspondiente número decimal.
   c. Convertir el número decimal 89 a su correspondiente número hexadecimal.
   d. Convertir el número hexadecimal 234D a su correspondiente número decimal.
   e. Convertir el número binario 11110000 a su correspondiente número hexadecimal.
   f. Convertir el número hexadecimal 456E a su correspondiente número binario.

2. Realizar las siguientes operaciones binarias:
   a. 110 + 100
   b. 111 – 100
   c. 111 x 101
   d. 100 ÷ 10

3. Realizar las siguientes operaciones con complementos:
   a. Determinar el complemento a 1 de 11101110.
   b. Determinar el complemento a 2 de 10000001.

4. Realizar las siguientes operaciones con códigos:
   a. Convertir a BCD el número decimal 170.
   b. Convertir a decimal el código BCD 10000110.
   c. Determinar la representación ASCII del carácter “$”.

UNIDAD 2: ÁLGEBRA DE BOOLE Y FUNCIONES

2.1 Álgebra de Boole, operaciones, teoremas, postulados, leyes y propiedades

2.1.1 Álgebra de Boole

El álgebra de Boole es una estructura algebraica que relaciona las operaciones lógicas fundamentales: suma, multiplicación y complementación o inversión.

A partir de estas operaciones lógicas básicas, es posible obtener operaciones más complejas que dan lugar a la definición y análisis de funciones lógicas. El álgebra de Boole opera con valores de tipo binario, generalmente representados por los estados 0 y 1, lo que la convierte en la base teórica del diseño y análisis de los sistemas digitales (Mano & Ciletti, 2014).

Los postulados del álgebra de Boole se muestran en la Figura 4.

Operación	Forma de representarla	Postulados básicos	
Suma	$F = a + b$	$0 + 0 = 0$ $0 + 1 = 1$ $1 + 1 = 1$ $a + a = a$	$a + 0 = a$ $a + 1 = 1$ $a + \overline{a} = 1$
Multiplicación	$F = a \cdot b$	$0 \cdot 0 = 0$ $0 \cdot 1 = 0$ $1 \cdot 1 = 1$	$a \cdot 0 = 0$ $a \cdot 1 = a$ $a \cdot a = a$ $a \cdot \overline{a} = 0$
Complementación o inversión	$F = \overline{a}$ $F = \overline{a \cdot b}$	$\overline{0} = 1$ $\overline{1} = 0$ $\overline{\overline{a}} = a$	

Figura 4 *Postulados del álgebra de Boole.*

Fuente: Floyd (2006).

Además de los postulados, se definen una serie de propiedades asociadas a las operaciones del álgebra de Boole. En la Tabla 4 se presentan las más importantes.

Tabla 4
Propiedades utilizadas en el álgebra de Boole

Propiedad conmutativa	$a + b = b + a$ $a.b = b.a$
Propiedad asociativa	$a.(b.c) = (a.b).c$ $a + (b + c) = (a + b) + c$
Propiedad distributiva	$a.(b + c) = a.b + a.c$ $a + (b.c) = (a + b).(a + c)$

Fuente: Floyd (2006).

Finalmente, en la Tabla 5 se presentan las leyes de De Morgan, las cuales resultan fundamentales para la simplificación de circuitos digitales (Floyd, 2006).

Tabla 5
Leyes De Morgan utilizadas en el Álgebra de Boole

Primera ley De Morgan	$\overline{a + b} = \bar{a} \, . \, \bar{b}$
Segunda ley De Morgan	$\overline{a.b} = \bar{a} + \bar{b}$

Fuente: Floyd (2006).

2.1.2 Función lógica

Se denomina función lógica a toda expresión algebraica formada por variables binarias que se relacionan mediante las operaciones básicas del álgebra de Boole.

En la Figura 5 se presenta un ejemplo de función lógica.

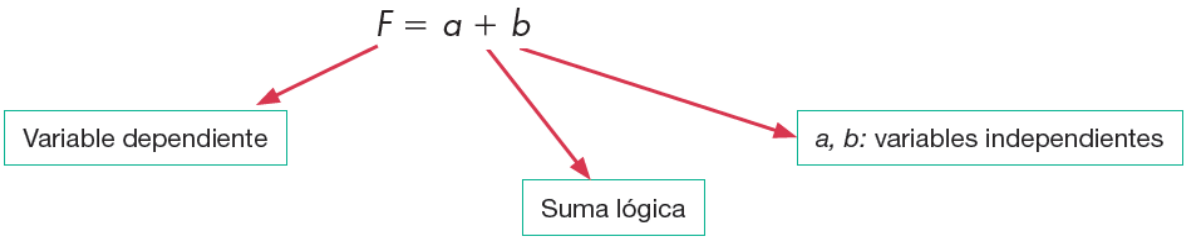


Figura 5 *Ejemplo de función lógica.*
Fuente: Floyd (2006).

2.1.3 Tabla de verdad

La tabla de verdad es una representación gráfica que muestra todos los valores que puede tomar una función lógica para cada una de las posibles combinaciones de las variables de entrada. Consiste en un cuadro formado por tantas columnas como variables tenga la función, más una columna adicional correspondiente a la salida, y por tantas filas como combinaciones binarias sea posible construir (Floyd, 2006).

El número de combinaciones posibles está dado por 2^n, donde n representa el número de variables de la función lógica (Floyd, 2006).

Ejemplo 1

Determinar el número de combinaciones posibles que se pueden generar con una variable y realizar la correspondiente tabla de verdad.

Solución

$Combinaciones = 2^n = 2^1 = 2$

x
0
1

Se pueden generar dos combinaciones binarias con una variable: 0, 1.

Ejemplo 2

Determinar el número de combinaciones posibles que se pueden generar con dos variables y realizar la correspondiente tabla de verdad.

Solución

$Combinaciones = 2^n = 2^2 = 4$

x	y
0	0
0	1
1	0
1	1

Se pueden generar cuatro combinaciones binarias con dos variables: 00, 01, 10, 11.

Ejemplo 3

Determinar el número de combinaciones posibles que se pueden generar con tres variables y realizar la correspondiente tabla de verdad.

Solución

$Combinaciones = 2^n = 2^3 = 8$

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Se pueden generar ocho combinaciones binarias con tres variables: 000, 001, 010, 011, 100, 101, 110, 111.

A partir de una función lógica es posible construir su correspondiente tabla de verdad (Floyd, 2006).

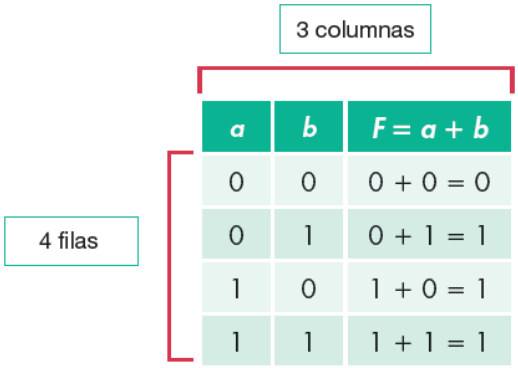
Ejemplo

Construir la tabla de verdad de la función: $F = a + b$.

Solución

La función lógica presenta dos variables de entrada, a y b; por lo tanto, se requieren dos columnas para representar dichas variables y una columna adicional para la función F.

Al tratarse de una función con dos variables binarias, es posible generar cuatro combinaciones distintas, ya que el número de combinaciones está dado por 2^n, siendo $n = 2$. En consecuencia, la tabla de verdad estará compuesta por cuatro filas, una para cada combinación posible de las variables de entrada.



a	b	$F = a + b$
0	0	0 + 0 = 0
0	1	0 + 1 = 1
1	0	1 + 0 = 1
1	1	1 + 1 = 1

Figura 6 *Tabla de verdad de la función* $F = a + b$.
Fuente: Floyd (2006).

2.2 Formas Canónicas de una función lógica Booleana

2.2.1 Primera forma canónica

Está formada por una suma de productos canónicos, es decir, productos que contienen todas las variables de la función, ya sea en su forma directa (normal) o en su forma complementada.

Existe una relación directa entre los productos canónicos y las combinaciones de las variables de entrada, de tal manera que cada producto canónico toma el valor 1 únicamente para una combinación específica y 0 para todas las demás. Para obtener el producto canónico de valor 1 asociado a una combinación determinada de las variables de entrada, se debe aplicar la siguiente regla: las variables que toman el valor 1 se representan en su forma natural, mientras que las variables que toman el valor 0 se representan en su forma complementada (Floyd, 2006).

Ejemplo

Representar el producto canónico para una función de tres variables de entrada $F(x, y, z)$, correspondiente a la combinación: $x = 0, y = 1, z = 0$.

Solución

Para obtener el producto canónico asociado a una combinación específica de las variables de entrada, se aplica la siguiente regla:

- Las variables que toman el valor 1 se representan en su forma natural.

- Las variables que toman el valor 0 se representan en su forma complementada.

Dada la combinación:

$$x = 0, y = 1, z = 0$$

Se tiene:

- $x = 0 \Rightarrow \bar{x}$
- $y = 1 \Rightarrow y$
- $z = 0 \Rightarrow \bar{z}$

Por lo tanto, el producto canónico correspondiente es:

$$F(x, y, z) = \bar{x}\, y\, \bar{z}$$

A los términos de la primera forma canónica se les denomina **minitérminos**, términos producto o productos canónicos.

Para obtener la primera forma canónica de una función lógica a partir de su tabla de verdad, basta con sumar los productos canónicos correspondientes a aquellas combinaciones de las variables de entrada para las cuales la salida de la función toma el valor 1.

2.2.2 Segunda forma canónica

Está formada por un producto de sumas canónicas, es decir, sumas que contienen todas las variables de entrada de la función, ya sea en su forma natural o en su forma complementada.

Existe una relación directa entre las sumas canónicas y las combinaciones de las variables de entrada, de tal manera que cada suma canónica toma el valor 0 únicamente para una combinación específica y 1 para todas las demás. Para obtener la suma canónica correspondiente a una combinación determinada de las variables de entrada, se aplica la siguiente regla: las variables que toman el valor 1 se representan en su forma complementada, mientras que las variables que toman el valor 0 se representan en su forma natural (Floyd, 2006).

Ejemplo

Representar la suma canónica para una función de tres variables de entrada $F(x, y, z)$, para la combinación: $x = 0$, $y = 1$, $z = 0$.

Solución

Para obtener la suma canónica asociada a una combinación específica de las variables de entrada, se aplica la siguiente regla:

- Las variables que toman el valor 1 se representan en su forma complementada.
- Las variables que toman el valor 0 se representan en su forma natural.

Dada la combinación:

$$x = 0, y = 1, z = 0$$

Se tiene:

- $x = 0 \Rightarrow x$
- $y = 1 \Rightarrow \bar{y}$
- $z = 0 \Rightarrow z$

Por lo tanto, la suma canónica correspondiente es:

$$F(x, y, z) = (x + \bar{y} + z)$$

A los términos de la segunda forma canónica se les denomina: **maxitérminos**, términos suma o sumas canónicas.

Para obtener la segunda forma canónica de una función lógica a partir de su tabla de verdad, basta con multiplicar las sumas canónicas correspondientes a aquellas combinaciones de las variables de entrada para las cuales la salida de la función toma el valor 0.

2.3 Simplificación de ecuaciones Booleanas: método algebraico, mapas de Karnaugh

En el diseño e implementación de circuitos digitales se emplean funciones booleanas para describir su comportamiento lógico. Antes de proceder al diseño y la implementación de dichos circuitos, es necesario simplificar al máximo la función lógica.

La simplificación de las funciones booleanas permite diseñar circuitos con el menor número posible de componentes electrónicos, lo que se traduce en una reducción de costos, complejidad y consumo. Esta simplificación puede realizarse mediante dos métodos principales:

I. Utilizando el método algebraico.
II. Utilizando el método de Karnaugh.

Por lo general, las formas canónicas no representan las expresiones más simplificadas de una función lógica, por lo que resulta necesario aplicar alguno de los métodos de simplificación mencionados.

2.3.1 Método algebraico para simplificación de funciones

Es el método que utiliza las propiedades y teoremas del álgebra de Boole para realizar la simplificación de funciones lógicas. No se trata de un método mecánico, sino de un procedimiento que requiere criterio, experiencia y conocimiento del álgebra de Boole (Floyd, 2006).

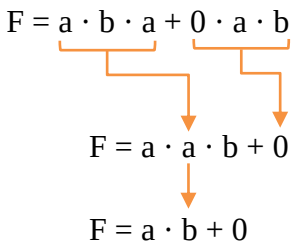
Ejemplo 1

Simplificar la función $F = (a\ .\ 1)\ .(b\ .\ b).(a\ .1) + (a\ .\ 0)\ .\ (a\ .\ a)\ .\ (b\ .1)$

Solución

Aplicamos las propiedades y postulados de Boole a cada paréntesis:

a. Propiedad conmutativa: $a.b.a\ =\ a.a.b$
b. Postulado: $a.0\ =\ 0$
c. Postulado: $a.a\ =\ a$
d. Postulado: $a + 0\ =\ a$



F = a · b

La función simplificada es: $F = a\ .\ b$

Ejemplo 2

Simplificar la función $F = \overline{a+b} \cdot (a+b)$

Solución

Aplicamos las propiedades y postulados de Boole a cada paréntesis:

a. Primera ley de De Morgan $\overline{a+b} = \bar{a} \cdot \bar{b}$

b. Propiedad distributiva: $\bar{a} \cdot \bar{b} \cdot (a+b) = \bar{a} \cdot \bar{b} \cdot a + \bar{a} \cdot \bar{b} \cdot b$

c. Postulado: $\bar{a} \cdot a = 0$

$F = \overline{a+b} \cdot (a+b)$

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

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

$F = 0 \cdot \bar{b} + \bar{a} \cdot 0$

$\boldsymbol{F = 0}$

La función simplificada es: $F = 0$

2.3.2 Método de Karnaugh para simplificación de funciones

El Método de Karnaugh es un método de simplificación de funciones mecánico; es decir, no hay que tener presente ninguna ley matemática. Permite simplificar funciones con n variables de una forma sencilla Floyd (2006).

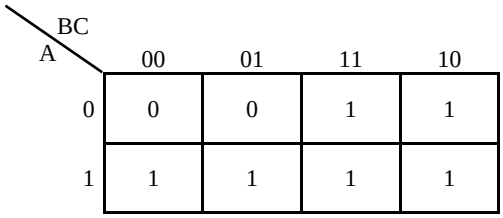
Ejemplo 1

Obtener la función simplificada de la siguiente tabla de verdad.

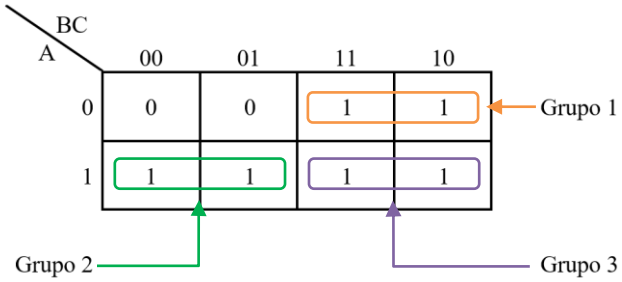
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Solución

a. Se dibuja el diagrama de Karnaugh y se colocan, en las casillas correspondientes, los valores "1" y "0" obtenidos a partir de la tabla de verdad.



b. Se agrupan los "1" adyacentes siempre en potencias de 2^n, de manera horizontal o vertical.



c. Se genera una expresión algebraica por cada grupo, utilizando la primera forma canónica.

Grupo 1: $\bar{A}BC + \bar{A}B\bar{C} = \bar{A}B(C + \bar{C}) = \bar{A}B$

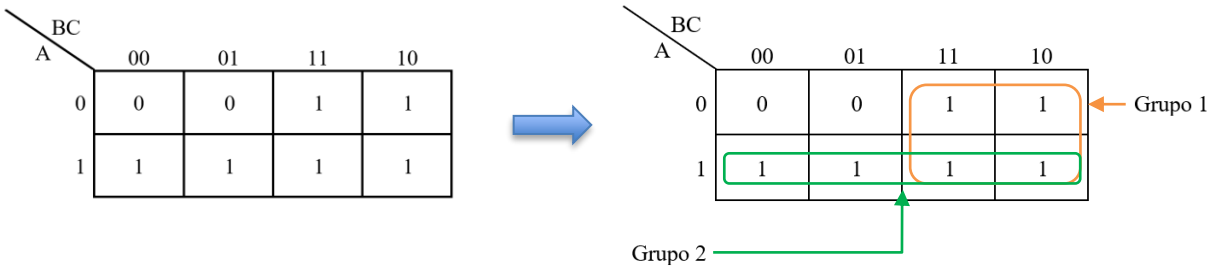
Grupo 2: $A\bar{B}\bar{C} + A\bar{B}C = A\bar{B}(\bar{C} + C) = A\bar{B}$

Grupo 2: $ABC + AB\bar{C} = AB(C + \bar{C}) = AB$

d. Se suman los productos de cada expresión para obtener la función lógica simplificada.

$F = \bar{A}B + A\bar{B} + AB$

Para obtener una función aún más simplificada, se deben tomar el mayor número de "1" en cada grupo Floyd (2006). En el ejemplo anterior la tabla original nos quedaría de la siguiente manera:



Las expresiones algebraicas correspondientes a cada grupo, utilizando la primera forma canónica, son las siguientes:

Grupo 1: B

Grupo 2: A

Por lo tanto, la nueva función lógica se obtiene como la suma de las expresiones algebraicas generadas, quedando expresada como:

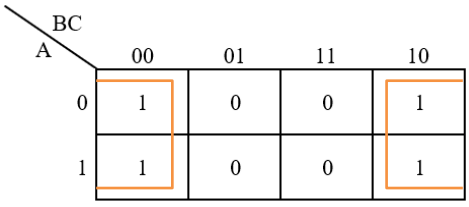
$F = A + B$

Ejemplo 2

Simplificar la siguiente función descrita por su tabla de verdad, utilizando el Método de Karnaugh:

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

a. Se dibuja el diagrama de Karnaugh, y colocamos en las casillas correspondientes los “1” y “0” de la tabla de verdad. Tomamos el mayor número “1” adyacentes y los agrupamos.



b. Se genera una expresión algebraica por cada grupo, utilizando la primera forma canónica.

Grupo 1: $\bar{C}$

e. Se suma los productos de cada expresión algebraica para obtener la función. En este caso, la función sería igual a la única expresión algebraica generada.

$F = \bar{C}$

2.4 Compuertas lógicas, diseño e implementación de funciones

Las puertas lógicas son circuitos digitales integrados cuyo funcionamiento se fundamenta en las operaciones y postulados del álgebra de Boole.

Las principales puertas lógicas se muestran en la Figura 7.

Nombre de la puerta	Equivalencia eléctrica y símbolo lógico: a) Equivalente eléctrico b) Símbolo ANSI c) Símbolo lógico tradicional	Tabla de verdad y función lógica
Puerta NOT	a) $\bar{A}$ b) 1 c)	NOT $s = \bar{a}$
Puerta OR (O)	a) $A + B$ b) ≥1 c)	$s = a + b$
Puerta AND (Y)	a) $A \cdot B$ b) & c)	$s = a \cdot b$
Puerta X-OR (OR exclusiva)	a) $A\oplus B = A\bar{B} + \bar{A}B$ b) =1 c)	$s = a \cdot \bar{b} + \bar{a} \cdot b$
Puerta NOR (No O)	a) $\overline{A+B} = \bar{A} \cdot \bar{B}$ b) ≥1 c)	$s = \overline{a + b}$
Puerta NAND (No Y)	a) $\overline{A \cdot B} = \bar{A} + \bar{B}$ b) & c)	$s = \overline{a \cdot b}$
Puerta X-NOR (NOR exclusiva)	a) $\overline{A\oplus B} = AB + \bar{A}\bar{B}$ b) =1 c)	$s = \bar{a} \cdot \bar{b} + a \cdot b$

a	s
A	X
0	1
1	0

a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

a	b	s
0	0	1
0	1	0
1	0	0
1	1	0

a	b	s
0	0	1
0	1	1
1	0	1
1	1	0

a	b	s
0	0	1
0	1	0
1	0	0
1	1	1

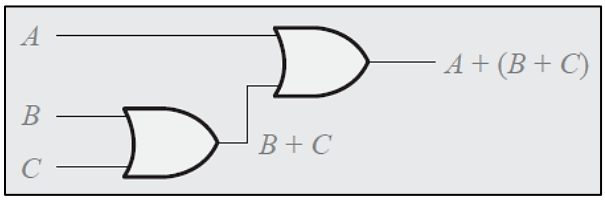
Figura 7 *Puertas lógicas.*

Fuente: Floyd (2006).

Ejemplos

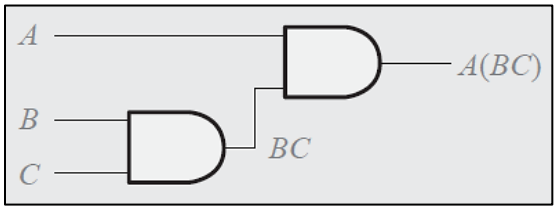
1. Diseñar un circuito con puertas lógicas, que responda a la función $F = A + (B + C)$.

Solución:



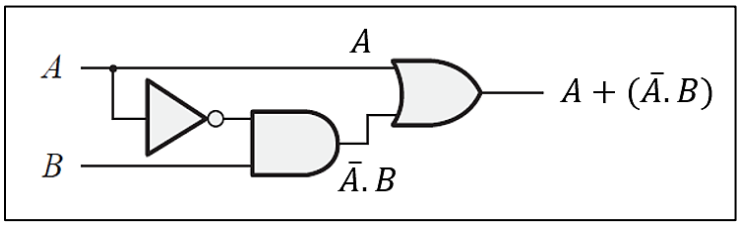
2. Diseñar un circuito con puertas lógicas, que responda a la función $F = A.(B.C)$.

Solución:



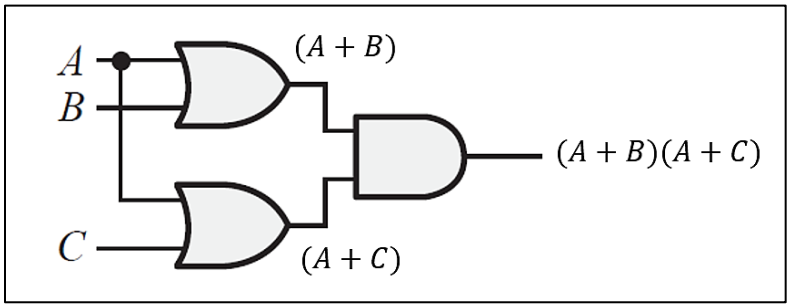
3. Diseñar un circuito con puertas lógicas, que responda a la función $F = A + (\bar{A}.B)$.

Solución:



4. Diseñar un circuito con puertas lógicas, que responda a la función $F = (A + B)(A + C)$.

Solución:



2.4.1 Circuitos integrados digitales comerciales

Una de las principales metas de los fabricantes de componentes electrónicos es incrementar el número de componentes básicos que pueden integrarse en una sola pastilla de silicio, ya que esto permite la reducción del tamaño de los circuitos, así como del volumen y el peso de los sistemas electrónicos (Mano & Ciletti, 2014).

Los componentes básicos de los circuitos integrados digitales son las puertas lógicas, las cuales se implementan dentro de un chip mediante distintas tecnologías de fabricación, entre las que destacan TTL (Transistor-Transistor Logic) y CMOS (Complementary Metal-Oxide-Semiconductor) (Mano & Ciletti, 2014).

Cada chip o circuito integrado, como se muestra en la Figura 8, dispone de una hoja de características o datasheet proporcionada por el fabricante, en la cual se especifican sus parámetros eléctricos, funcionales y mecánicos, información indispensable para su correcta selección e implementación en un circuito digital (Mano & Ciletti, 2014).

Figura 8 *Chip integrado.*
Fuente: Mano y Ciletti (2014).

Cada circuito integrado posee una designación o serie, como se ilustra en la Figura 9. Los dos primeros dígitos indican la tecnología utilizada en su fabricación, mientras que los dos últimos dígitos señalan el tipo de puerta lógica correspondiente.

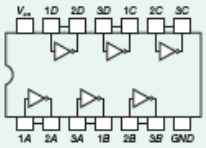
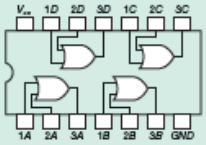
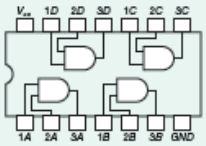
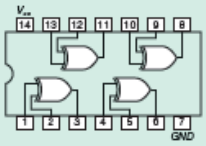
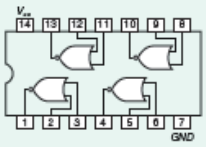
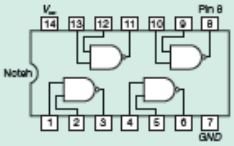
Tipo de puerta (y nombre del circuito integrado)	Chip integrado	N.º de puertas
La puerta lógica NOT (7404)		Tiene seis puertas NOT de una entrada cada una.
La puerta lógica OR (7432)		Tiene cuatro puertas OR de dos entradas cada una.
La puerta lógica AND (7408)		Tiene cuatro puertas AND con dos entradas cada una.
La puerta lógica X-OR (7486)		Tiene cuatro puertas X-OR con dos entradas cada una.
La puerta lógica NOR (7402)		Tiene cuatro puertas NOR con dos entradas cada una.
La puerta lógica NAND (7400)		Tiene cuatro puertas NAND con dos entradas cada una.

Figura 9 *Chips integrados y números de puertas lógicas.*

Fuente: Mano y Ciletti (2014).

2.5 Familias lógicas e interfaces

Como consecuencia de las diferentes técnicas de fabricación de los circuitos integrados, es posible encontrar diversas familias lógicas, las cuales se clasifican en función del tipo de transistor utilizado en su construcción (Mano & Ciletti, 2014).

De este modo, cuando se emplean transistores bipolares, se obtiene la familia lógica denominada TTL (Transistor-Transistor Logic), mientras que el uso de transistores unipolares da lugar a la familia CMOS (Complementary Metal-Oxide-Semiconductor). Cada una de estas familias

presenta ventajas y desventajas particulares; por esta razón, en el diseño de sistemas digitales se selecciona la familia lógica más adecuada según los requerimientos de la aplicación (Floyd, 2006).

Las características generales que se consideran en las familias lógicas integradas son las siguientes:

- Alta velocidad de propagación.
- Mínimo consumo.
- Bajo coste.
- Máxima inmunidad al ruido y a las variaciones de temperatura.

2.5.1 Familia Lógica TTL

Las siglas TTL corresponden a Lógica Transistor-Transistor (Transistor-Transistor Logic). En esta familia lógica, las puertas digitales están constituidas fundamentalmente por resistencias, diodos y transistores bipolares (Mano & Ciletti, 2014).

La familia TTL comprende varias series comerciales, entre las cuales se encuentra la serie 74, ampliamente utilizada en aplicaciones digitales. Las principales características de esta serie se describen en la Figura 10.

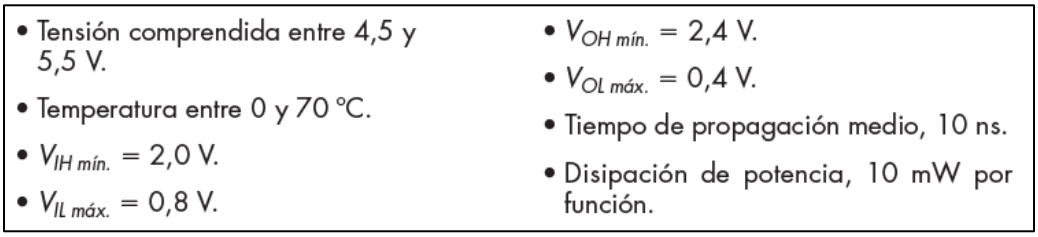


Figura 10 *Características de la serie 74.*

Fuente: Floyd (2006).

Otra serie perteneciente a la familia TTL es la serie 54, la cual presenta las mismas características funcionales que la serie 74, con la diferencia de que su rango de temperatura de operación se encuentra comprendido entre −55 °C y 125 °C. Debido a esta característica, la serie 54 se emplea principalmente en aplicaciones militares, industriales y espaciales, donde se requieren condiciones de operación más exigentes.

Las puertas lógicas más utilizadas corresponden a la serie 74, ya que se trata de la variante más comercial y de uso general. En particular, son ampliamente empleadas las puertas cuya referencia es 74Lxx, donde la letra L indica *Low-power* (bajo consumo). Las principales características de esta subfamilia se describen en la Figura 11.

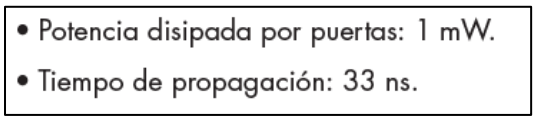


Figura 11 *Características de la serie 74Lxx.*
Fuente: Floyd (2006).

A su vez, se encuentra la subfamilia 74Sxx, en la cual la letra S significa Schottky. Sus principales características se muestran en la Figura 12.

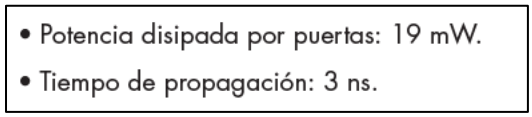


Figura 12 *Características de la serie 74Lxx.*
Fuente: Floyd (2006).

Finalmente, la subfamilia LS (74LSxx) corresponde a Low-power Schottky, y sus principales características se muestran en la Figura 13.

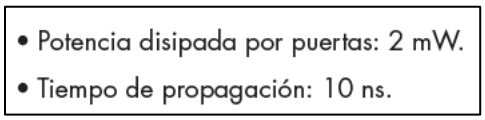


Figura 13 *Características de la serie 74LSxx.*
Fuente: Floyd (2006).

2.5.2 Familia Lógica CMOS

En esta familia lógica, el componente básico es el transistor MOS (Metal-Oxide-Semiconductor). Los circuitos integrados CMOS combinan tecnologías NMOS, constituidas por transistores de canal N, y PMOS, cuyo elemento fundamental es el transistor MOS de canal P.

La familia CMOS básica, que aparece en los catálogos de los fabricantes, corresponde a la serie 4000. Sus características más importantes se describen en la Figura 14.

- La tensión de alimentación varía entre 3 y 18 V.
- El rango de temperaturas oscila entre −40 y 85 °C.
- Los niveles de tensión son: $V_{IL\ mín.} = 3{,}5$ V; $V_{IL\ máx.} = 1{,}5$ V; $V_{OH\ mín.} = 4{,}95$ V; $V_{OL\ máx.} = 0{,}05$ V.
- Los tiempos de propagación varían inversamente con la tensión de alimentación, siendo de 60 ns para 5 V y de 30 ns para 10 V.
- La potencia disipada por puerta es de 10 nW.

Figura 14 *Características de la serie 74LSxx.*

Fuente: Floyd (2006).

Inicialmente, los circuitos integrados CMOS se fabricaron con la misma disposición de puertas lógicas que las familias TTL. De este modo, se desarrolló la familia 74C, compatible a nivel funcional con la familia TTL, cuyas características eléctricas son muy similares a las de la familia CMOS serie 4000.

Posteriormente, gracias a los avances en las técnicas de fabricación, se desarrollaron las series 74HC (*High-speed CMOS*) y 74HCT (*High-speed CMOS compatible with TTL*). Estas familias presentan características comparables a las subfamilias LS de la familia TTL, pero con consumos de potencia significativamente menores.

Las series CMOS más utilizadas corresponden a la familia 74HCxx, donde HC significa *High-speed CMOS*. El tiempo de propagación típico de estas series es del orden de 8 ns, y operan con tensiones de alimentación comprendidas entre 2 V y 6 V.

2.5.3 Compatibilidad entre las familias lógica TTL y CMOS

Cuando se desea interconectar distintas familias lógicas, es necesario considerar su compatibilidad, tanto en términos de corriente como de tensión.

- ***Compatibilidad de corriente***

  Para conectar la salida de un circuito con la entrada de otro, el circuito que actúa como salida debe ser capaz de suministrar la corriente necesaria para alimentar la entrada del

circuito receptor. Por lo tanto, debe cumplirse que la corriente de salida disponible sea mayor o igual que la corriente requerida por la entrada del circuito conectado.

La compatibilidad de corriente entre las familias TTL y CMOS se ilustra en la Figura 15.

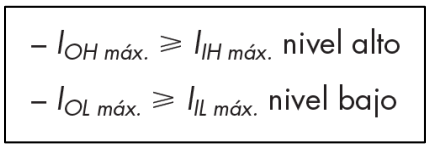


Figura 15 *Compatibilidad de corriente entre TTL y CMOS.*
Fuente: Floyd (2006).

- ***Compatibilidad de tensión***

  Para conectar la salida de un circuito con la entrada de otro, es necesario verificar que los niveles de tensión de salida del primer circuito se encuentren dentro de los rangos de tensión aceptables por la entrada del segundo circuito.

La compatibilidad de tensión entre las familias TTL y CMOS se muestra en la Figura 16.

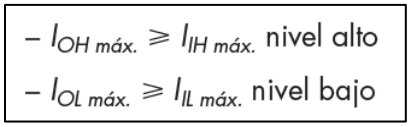


Figura 16 *Compatibilidad de tensión entre TTL y CMOS.*
Fuente: Floyd (2006).

- ***Compatibilidad de nivel***

  Dado que la compatibilidad de corriente suele cumplirse en la mayoría de los casos, resulta necesario verificar la compatibilidad de nivel lógico, en particular el nivel alto, ya que este es el más crítico al interconectar circuitos TTL y CMOS.

La compatibilidad de nivel entre las familias TTL y CMOS se presenta en la Figura 17.

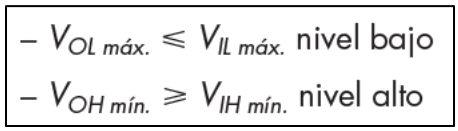


Figura 17 *Compatibilidad de nivel entre TTL y CMOS.*
Fuente: Floyd (2006).

ACTIVIDADES

Actividad 1: Ejercicios sobre tabla de verdad.

1. Determinar el número de combinaciones posibles que se pueden generar con cuatro variables y realizar la correspondiente tabla de verdad.

2. Determinar el número de combinaciones posibles que se pueden generar con cinco variables y realizar la correspondiente tabla de verdad.

Actividad 2: Ejercicios sobre formas canónicas.

1. Representar el producto canónico para una función de tres variables de entrada $F(x, y, z)$, para la combinación: $x = 1, y = 1, z = 0$.

2. Representar la suma canónica para una función de tres variables de entrada $F(x, y, z)$, para la combinación: $x = 0, y = 1, z = 1$.

Actividad 3: Ejercicios sobre simplificación de ecuaciones booleanas.

1. Utilizar el método algebraico para simplificar la siguiente función:
   a. $F = \overline{(a + b) \cdot (c + d)}$
   b. $F = \overline{(a + b)} + (c + d)$

2. Utilizar el método de Karnaugh para obtener la función simplificada de la siguiente tabla de verdad:

x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Actividad 4: Autoevaluación.

1. Simplificar la siguiente función.
   a. $F = c \cdot b \cdot a + \bar{c} \cdot \bar{b} \cdot a + \bar{c} \cdot b \cdot a$
   b. $F = \bar{c} \cdot b \cdot \bar{a} + \bar{c} \cdot b \cdot a + \bar{c} \cdot \bar{b} \cdot \bar{a}$

2. Desarrollar la tabla de verdad de la siguiente función:
   a. $F = \overline{(a+b)} \cdot (a+b)$
   b. $F = \overline{(a+b)} \cdot (a+\bar{c})$

3. Utilizar el método de Karnaugh para obtener la función simplificada de la siguiente tabla de verdad:

x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

4. Diseñar el circuito con puertas lógicas, de la función $F = A + (\bar{A}.B) + C$

UNIDAD 3: ANÁLISIS LÓGICO COMBINACIONAL

3.1 Sumadores, restadores, multiplicadores, comparadores

3.1.1 Sumadores

3.1.1.1 Sumadores Básicos

Semi sumador

Un semi sumador recibe dos dígitos binarios en sus entradas y genera dos dígitos binarios en sus salidas: un bit de suma y un bit de acarreo, de acuerdo con las reglas básicas de la suma binaria (Mano & Ciletti, 2014).

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

Figura 18 *Reglas de la suma binaria.*
Fuente: Mano & Ciletti (2014).

Los semi sumadores se representan mediante el símbolo lógico ilustrado en la Figura 19.

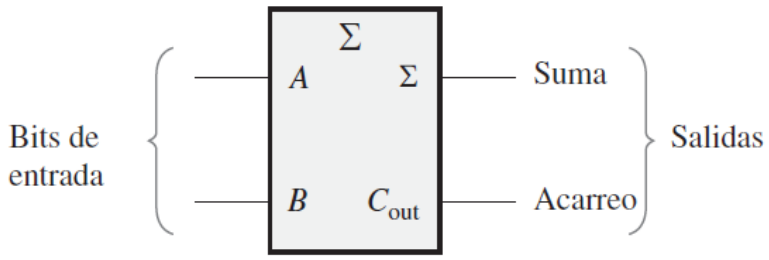


Figura 19 *Símbolo lógico de un semi sumador.*
Fuente: Mano & Ciletti (2014).

En la Figura 20 se muestra la tabla de verdad de un semi sumador (Mano & Ciletti, 2014).

A	B	C_{OUT}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = suma
C_{out} = acarreo de salida
A y B = variables de entrada (operandos)

Figura 20 *Tabla de verdad de un semi sumador.*
Fuente: Mano & Ciletti (2014).

A partir del funcionamiento lógico de un semi sumador, expuesto en la Figura 20, las expresiones correspondientes a la suma y al acarreo de salida pueden obtenerse como funciones de entrada.

- La salida de acarreo (C_{out}) toma el valor 1 únicamente cuando A y B son iguales a 1; por lo tanto, C_{out} puede expresarse mediante una operación lógica AND entre las variables de entrada:

$$C_{out} = AB$$

- La salida correspondiente a la suma (Σ) toma el valor 1 únicamente cuando las variables A y B son distintas; en consecuencia, la suma puede expresarse mediante una operación OR exclusiva (XOR) entre las variables de entrada:

$$\Sigma = A \oplus B$$

A partir de estas expresiones lógicas, es posible desarrollar la implementación lógica del funcionamiento de un semi sumador (Torres, 2020).

La salida de acarreo se obtiene mediante una puerta AND, con A y B como entradas, mientras que la salida de la suma se obtiene mediante una puerta OR exclusiva, tal como se muestra en la Figura 21.

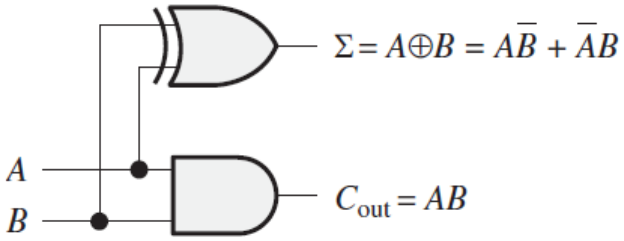


Figura 21 *Diagrama lógico de un semi sumador.*
Fuente: Mano & Ciletti (2014).

3.1.1.2 Sumadores Completos

Un sumador completo acepta dos bits de entrada y un acarreo de entrada, y genera una salida de suma y un acarreo de salida (Mano & Ciletti, 2014).

La diferencia principal entre un sumador completo y un semi sumador radica en que el sumador completo admite un acarreo de entrada. El símbolo lógico de un sumador completo se muestra en la Figura 22, y la tabla de verdad, presentada en la Figura 23, describe su funcionamiento.

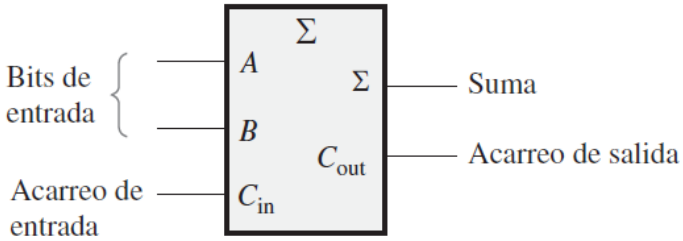


Figura 22 *Diagrama lógico de un sumador completo.*
Fuente: Mano & Ciletti (2014).

A	B	C_{in}	C_{OUT}	Σ
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

C_{in} = acarreo de entrada. Algunas veces se designa como *CI*.
C_{out} = acarreo de salida. Algunas veces se designa como *CO*.
Σ = suma
A y B = variables de entrada (operandos)

Figura 23 *Tabla de verdad de un sumador completo.*
Fuente: Mano & Ciletti (2014).

debe sumar dos bits de entrada y un acarreo de entrada. Del semi sumador se conoce que la suma de los bits de entrada A y B se obtiene mediante la operación OR exclusiva (XOR) de dichas variables, es decir, $A \oplus B$. Para incorporar el acarreo de entrada (C_{in}) a los bits de entrada, es necesario aplicar nuevamente la operación OR exclusiva, obteniéndose la siguiente ecuación para la salida de suma del sumador completo:

$$\Sigma = (A \oplus B) \oplus C_{in}$$

Esto implica que, para implementar la función de suma del sumador completo, se pueden utilizar dos puertas OR exclusivas de dos entradas (Mano & Ciletti, 2014). La primera puerta XOR genera el término A $\oplus$ B, mientras que la segunda puerta XOR recibe como entradas la salida de la primera y el acarreo de entrada, tal como se ilustra en la Figura 24.

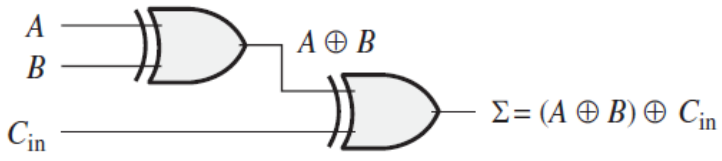


Figura 24 *Lógica para realizar la suma de tres bits.*
Fuente: Mano & Ciletti (2014).

El acarreo de salida toma el valor 1 cuando ambas entradas de la primera puerta XOR están a 1, o cuando ambas entradas de la segunda puerta XOR están a 1. En consecuencia, el acarreo de salida del sumador completo se obtiene a partir del producto lógico (AND) de las variables de entrada A y B, y del producto lógico entre A $\oplus$ B y el acarreo de entrada Cin (Mano & Ciletti, 2014). Posteriormente, se aplica la operación OR a estos dos términos, obteniéndose la siguiente expresión:

$$C_{out} = AB + (A \oplus B)C_{in}$$

Esta función lógica se implementa y se combina con la lógica de la suma para formar el circuito de un sumador completo, tal como se muestra en la Figura 25.

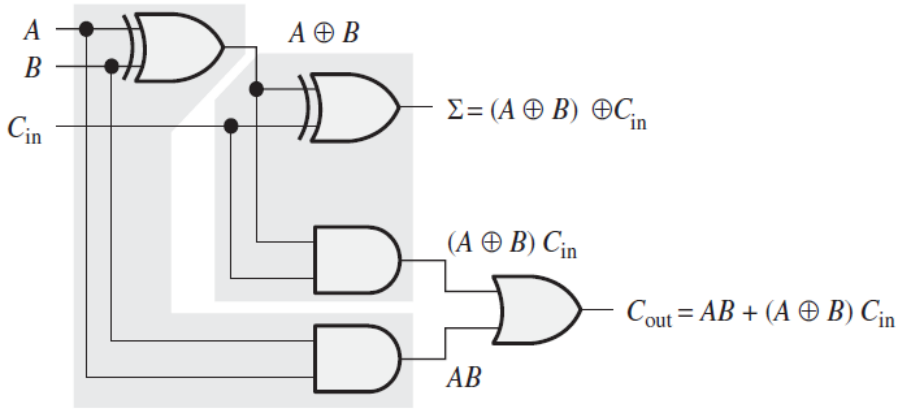


Figura 25 *Circuito lógico de un sumador completo.*
Fuente: Mano & Ciletti (2014).

Sumadores Binarios en Paralelo

Un único sumador completo es capaz de sumar dos números binarios de 1 bit y un acarreo de entrada. Para sumar números binarios de más de un bit, se tienen que utilizar sumadores completos adicionales. Cuando se suman dos números binarios, cada columna genera un bit de suma y un 1 ó 0, correspondiente al bit de acarreo, que se añade a la columna inmediata de la izquierda, como se muestra en la Figura 26 con dos números de 2 bits.

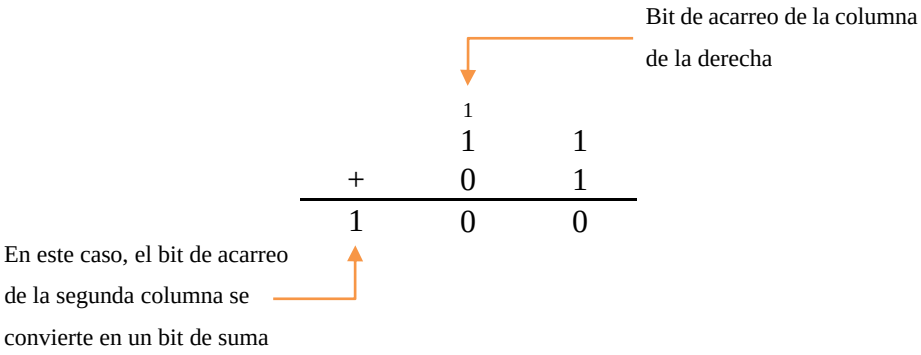


Figura 26 *Suma de dos números binarios con bit de acarreo.*

Fuente: Mano & Ciletti (2014).

Para sumar dos números binarios, se requiere un sumador completo por cada bit de los números que se desean sumar. De este modo, para números de dos bits se necesitan dos sumadores completos, para números de cuatro bits se requieren cuatro sumadores, y así sucesivamente.

La salida de acarreo de cada sumador se conecta a la entrada de acarreo del sumador de orden inmediatamente superior, como se muestra en la Figura 27 para el caso de un sumador de 2 bits. En la posición menos significativa, puede utilizarse un semi sumador o, alternativamente, puede fijarse a cero (masa) la entrada de acarreo de un sumador completo, dado que no existe acarreo de entrada en dicha posición (Torres, 2020).

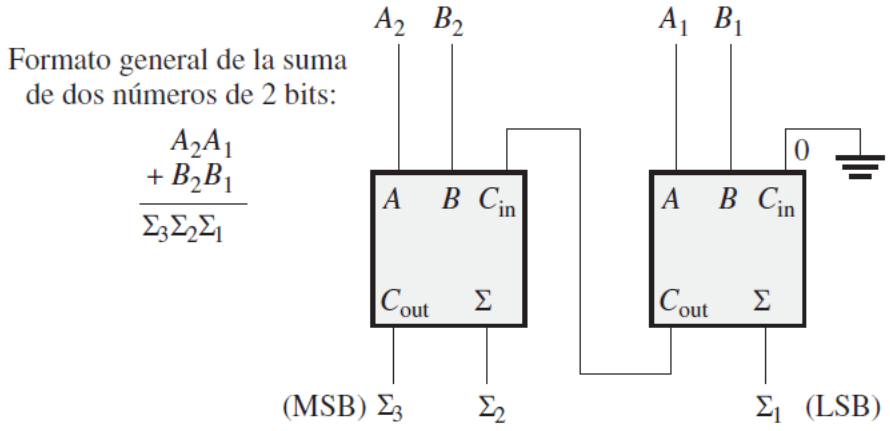


Figura 27 *Diagrama de bloques de un sumador en paralelo de 2 bits utilizando dos sumadores completos.*

Fuente: Mano & Ciletti (2014).

Los bits menos significativos (LSB) de los dos números se representan como A_1 y B_1. Los siguientes bits de orden superior se representan como A_2 y B_2. Los bits de suma obtenidos son Σ_1, Σ_2 y Σ_3. El acarreo de salida del sumador completo ubicado más a la izquierda se convierte en el bit más significativo (MSB) de la suma, correspondiente a Σ_3.

Esta configuración lógica se aplica a la suma de n bits, en la cual se requieren n sumadores completos conectados en cascada, de modo que el acarreo de salida de cada etapa se propaga a la siguiente (Mano & Ciletti, 2014).

En la Figura 28 se muestra el diagrama de bloques de un sumador básico de 4 bits, implementado mediante cuatro sumadores completos.

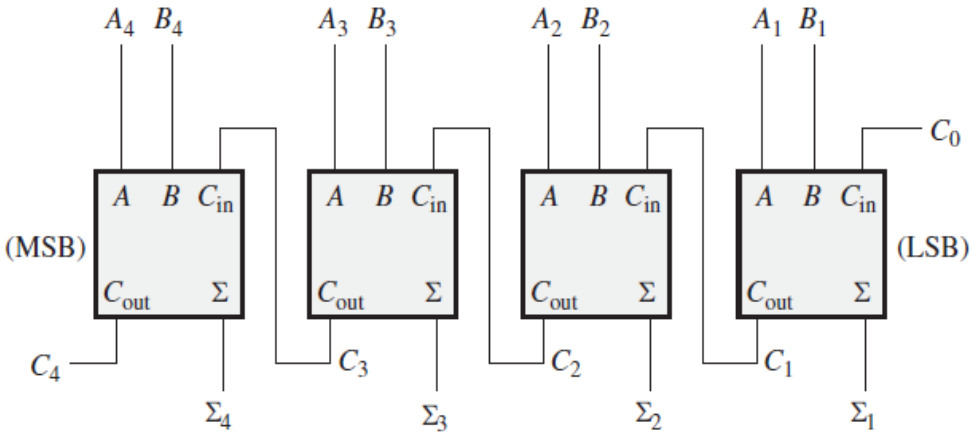


Figura 28 *Diagrama de bloques de un sumador en paralelo de 4 bits*.
Fuente: Mano & Ciletti (2014).

En la mayoría de las hojas de características suministradas por los fabricantes, se denomina C_0 al acarreo de entrada del sumador correspondiente al bit menos significativo (LSB). Para el caso de un sumador de cuatro bits, C_4 representa el acarreo de salida del sumador asociado al bit más significativo (MSB). Las salidas de suma se designan como Σ_1 (LSB) hasta Σ_4 (MSB) (Mano & Ciletti, 2014).

En la Figura 29 se muestra el símbolo lógico correspondiente.

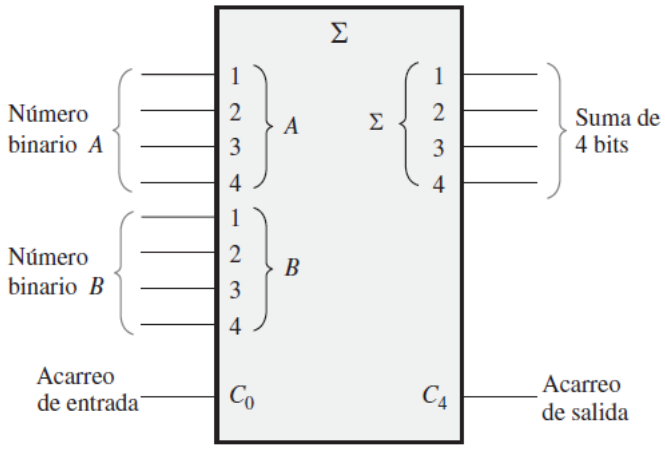


Figura 29 *Símbolo lógico de bloques de un sumador en paralelo de 4 bits.*

Fuente: Mano & Ciletti (2014).

La Figura 30 presenta la tabla de verdad de un sumador de 4 bits. El subíndice n representa la posición del bit dentro del sumador y puede tomar los valores 1, 2, 3 o 4 para un sumador de cuatro bits. El término C_{n-1}corresponde al acarreo generado por el sumador de la etapa anterior. Los acarreos C_1, C_2 y C_3 se generan de forma interna, mientras que C_0 corresponde al acarreo de entrada externo y C_4 representa el acarreo de salida del sumador.

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figura 30 *Tabla de verdad de un sumador en paralelo de 4 bits.*

Fuente: Mano & Ciletti (2014).

Sumador en Paralelo de 4 bits 74LS283

Un ejemplo de sumador paralelo de 4 bits disponible como circuito integrado es el 74LS283. En este dispositivo, la tensión de alimentación V_{CC} corresponde al pin 16, mientras que el pin 8 se conecta a tierra, configuración que resulta estándar en este tipo de integrados.

El diagrama de pines y el símbolo lógico de este dispositivo se muestran en la Figura 31 y la Figura 32, respectivamente. Este sumador se encuentra disponible tanto en la familia lógica TTL como en la familia CMOS (Mano & Ciletti, 2014).

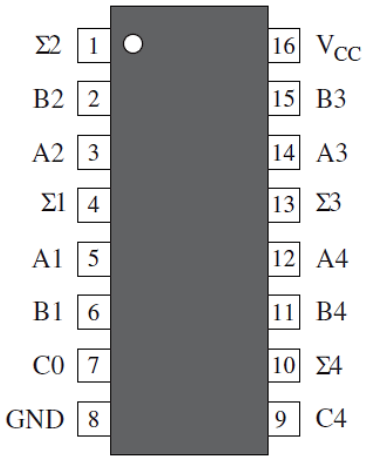


Figura 31 *Diagrama de pines del sumador en paralelo de 4 bits 74LS283.*
Fuente: Mano & Ciletti (2014).

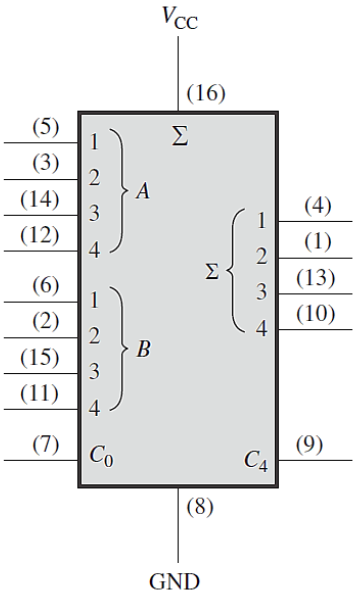


Figura 32 *Símbolo lógico del sumador en paralelo de 4 bits 74LS283.*
Fuente: Mano & Ciletti (2014).

Un sumador en paralelo de 4 bits se puede expandir para realizar sumas de dos números de 8 bits, utilizando dos sumadores de cuatro bits. La entrada de acarreo del sumador de menor orden (C_0) se conecta a tierra, ya que no existe acarreo en la posición del bit menos significativo, y la salida

de acarreo del sumador de menor orden se conecta a la entrada de acarreo del sumador de orden superior, como se muestra en la Figura 33. Este proceso se denomina conexión en cascada. En este caso, el acarreo de salida se designa como C_8, dado que se genera a partir del bit que se encuentra en la posición número ocho. El sumador de menor orden es el que realiza la suma de los cuatro bits menos significativos, mientras que el sumador de orden superior es el que suma los cuatro bits más significativos de los dos números binarios de 8 bits (Mano & Ciletti, 2014).

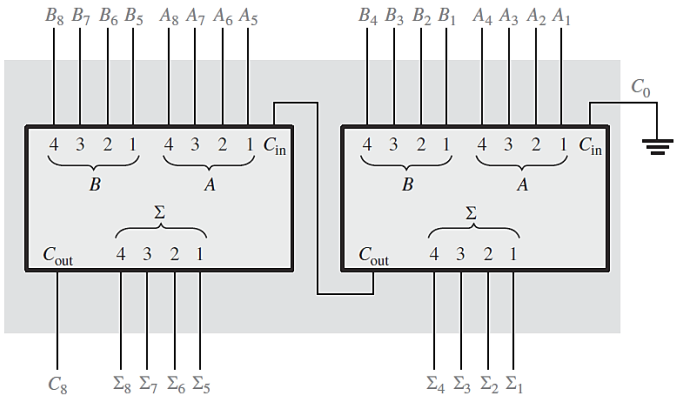


Figura 33 *Sumador de 4 bits conectados en cascada que forman un sumador de 8 bits.*
Fuente: Mano & Ciletti (2014).

Se utilizan dos sumadores en paralelo de 4 bits 74LS283 para implementar el sumador de 8 bits. La única conexión entre los dos 74LS283 es la que une la salida de acarreo (pin 9) del sumador de menor orden a la entrada de acarreo (pin 7) del sumador de orden superior, como se muestra en la Figura 34. El pin 7 del sumador de orden inferior se conecta a masa (no hay entrada de acarreo).

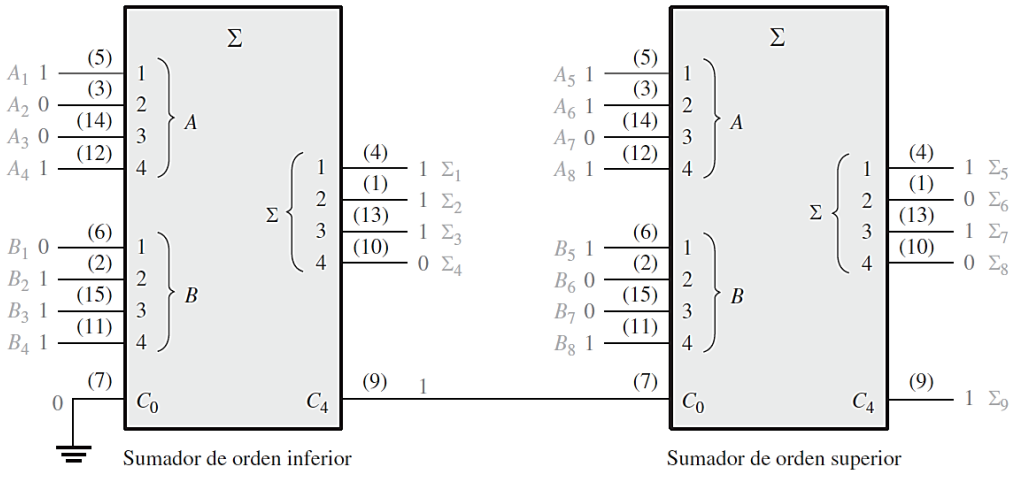


Figura 34 *Sumadores 74LS283 conectados como un sumador en paralelo de 8 bits.*
Fuente: Mano & Ciletti (2014).

De forma similar, se pueden emplear cuatro sumadores de 4 bits en cascada para sumar dos números de 16 bits, como se muestra en la Figura 35. El acarreo de salida se designa como C_{16}, ya que se genera a partir del bit que se encuentra en la posición dieciséis (Mano & Ciletti, 2014).

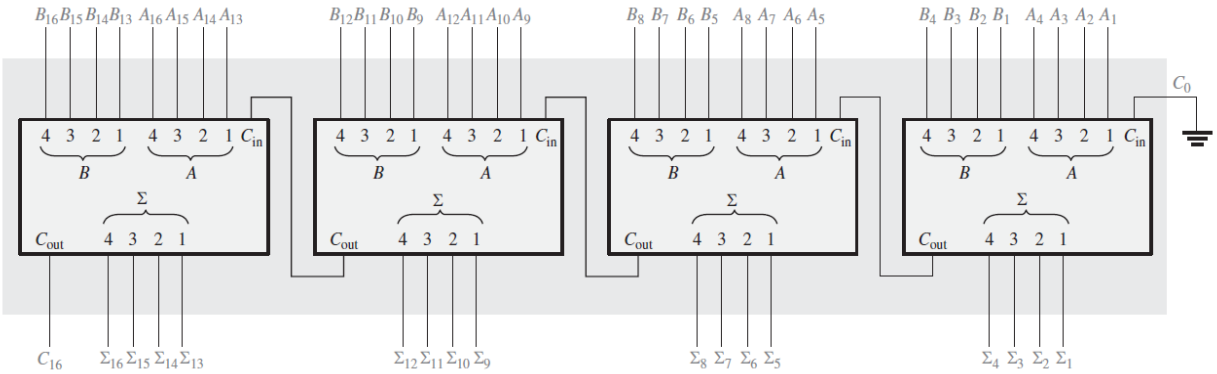


Figura 35 *Sumador de 4 bits conectados en cascada que forman un sumador de 16 bits.*
Fuente: Mano & Ciletti (2014).

La misma lógica de diseño e implementación se aplica para realizar las operaciones binarias de resta y multiplicación (Mano & Ciletti, 2014).

3.1.2 Comparadores

3.1.2.1 Igualdad

La puerta OR exclusiva (XOR) puede emplearse como un comparador básico, ya que su salida toma el valor 1 cuando sus dos bits de entrada son diferentes y 0 cuando ambos bits son iguales. La Figura 36 muestra una puerta OR exclusiva utilizada como comparador de dos bits.

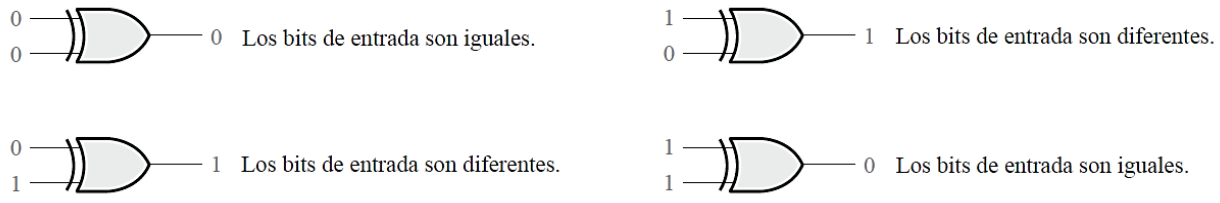


Figura 36 *Funcionamiento del comparador básico.*
Fuente: Rojas (2021).

Para comparar números binarios de dos bits, se requiere una puerta OR exclusiva adicional. Los bits menos significativos (LSB) de ambos números se comparan mediante la puerta G1, mientras que los bits más significativos (MSB) se comparan mediante la puerta G2, como se muestra en la Figura 37.

Si los dos números son iguales, sus bits correspondientes también lo serán y, en consecuencia, la salida de cada puerta OR exclusiva será 0. Por el contrario, si alguno de los pares de bits correspondientes no es idéntico, la salida de la puerta OR exclusiva correspondiente será 1 (Rojas, 2021).

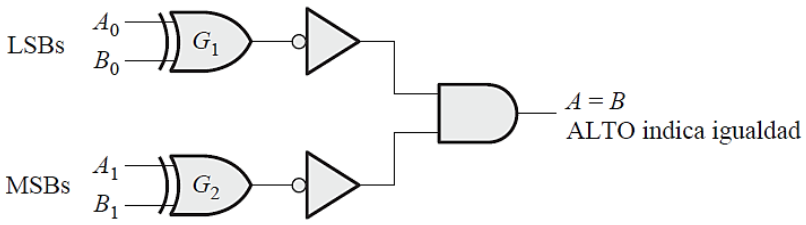


Figura 37 *Funcionamiento del comparador básico.*
Fuente: Rojas (2021).

Para obtener un único resultado de salida que indique la igualdad o desigualdad entre los dos números, se pueden usar dos inversores y una puerta AND, como muestra la Figura 37. La salida de cada puerta OR exclusiva se invierte y se aplica a la entrada de la puerta AND. Cuando los bits de entrada de cada OR exclusiva son iguales, lo que quiere decir que los bits de ambos números son iguales, las entradas de la puerta AND son 1, por lo que el resultado a su salida también será 1. Cuando los dos números no son iguales, al menos uno o ambos conjuntos de bits será distinto, lo que da lugar a, al menos, un 0 en una de las entradas de la puerta AND, y el resultado a su salida será 0. Por tanto, la salida de la puerta AND indica la igualdad (1) o desigualdad (0) entre dos números.

Un circuito comparador básico se puede ampliar para poder tratar cualquier número de bits. La puerta AND establece la condición de que todos los bits de los dos números que se comparan tienen que ser iguales si los números lo son (Rojas, 2021).

3.1.2.2 Desigualdad

Además de disponer de una salida que indica si los dos números son iguales, muchos circuitos integrados comparadores tienen salidas adicionales que indican cuál de los dos números que se comparan es el mayor. Esto significa que existe una salida que indica cuándo el número A es mayor que el número B (A > B) y otra salida que indica cuándo A es menor que B (A < B), como se muestra en el símbolo lógico del comparador de cuatro bits de la Figura 38.

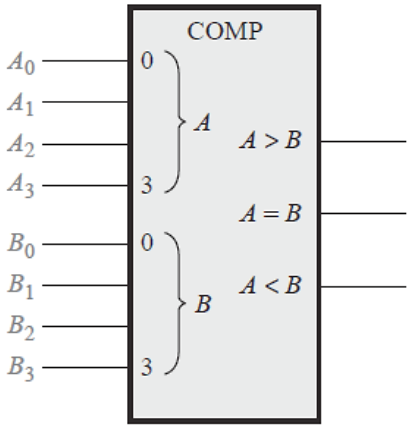


Figura 38 *Funcionamiento del comparador básico.*
Fuente: Rojas (2021).

Para determinar la desigualdad entre dos números binarios A y B, se examina, en primer lugar, el bit de mayor orden de cada número. Las posibles condiciones son las siguientes:

- Si $A_3 = 1$ y $B_3 = 0$, entonces A es mayor que B.
- Si $A_3 = 0$ y $B_3 = 1$, entonces A es menor que B.
- Si $A_3 = B_3$, entonces debemos examinar los siguientes bits de orden inmediatamente inferior.

Estas tres condiciones se aplican para cada posición de los bits dentro del número binario. El procedimiento general utilizado en un comparador consiste en verificar la desigualdad en cada posición de bit, comenzando siempre desde los bits más significativos (Rojas, 2021).

Cuando se encuentra una desigualdad en alguna de las posiciones de mayor orden, la relación entre los números queda establecida, y cualquier desigualdad en bits de menor orden debe ignorarse, ya que podría reflejar una relación completamente opuesta entre los números. Por lo tanto, la relación de más alto orden es la que tiene prioridad.

3.1.2.3 Comparador de magnitud de 4 bits 74HC85

El 74HC85 es un comparador de magnitud que también se encuentra disponible en otras familias de circuitos integrados. El diagrama de pines y el símbolo lógico de este dispositivo se muestran en la Figura 39 y la Figura 40, respectivamente.

Este dispositivo dispone de todas las entradas y salidas del comparador analizado anteriormente y, además, incorpora tres entradas de cascada: $A < B$, $A = B$ y $A > B$. Estas entradas permiten la interconexión en cascada de varios comparadores, lo que posibilita la comparación de números binarios de más de cuatro bits. Para ampliar el comparador, las salidas $A < B$, $A = B$ y $A > B$ del comparador de menor orden se conectan a las entradas de cascada del comparador de orden inmediatamente superior.

El comparador de menor orden debe presentar un nivel lógico ALTO en la entrada $A = B$ y un nivel lógico BAJO en las entradas $A < B$ y $A > B$. Este dispositivo está disponible en diversas familias lógicas CMOS y TTL (Rojas, 2021).

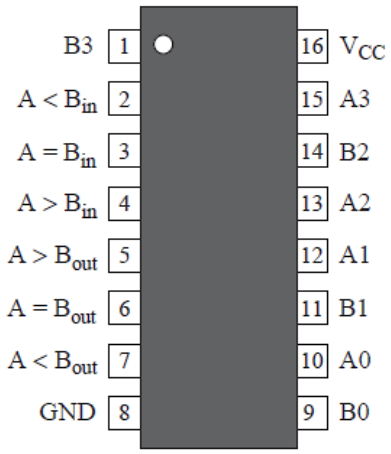


Figura 39 *Diagrama de pines del comparador de magnitud de 4 bits 74HC85.*

Fuente: Rojas (2021).

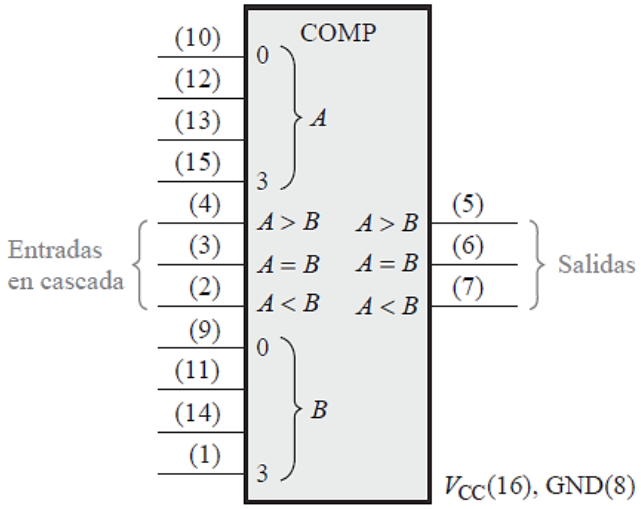


Figura 40 *Símbolo lógico del comparador de magnitud de 4 bits 74HC85.*

Fuente: Rojas (2021).

3.2 Decodificadores

La función básica de un decodificador es detectar la presencia de una determinada combinación de bits (código) en sus entradas y señalar la presencia de este código mediante un cierto nivel de salida. En su forma general, un decodificador posee n líneas de entrada para gestionar n bits y en una de las 2^n líneas de salida indica la presencia de una o más combinaciones de n bits. En esta sección, se presentan varios tipos de decodificadores. Los principios básicos se pueden extender a otros decodificadores (Rojas, 2021).

3.2.1 Decodificador binario básico

Supongamos que necesitamos determinar cuándo aparece el número binario 1001 en las entradas de un circuito digital. Se puede utilizar una puerta AND como elemento básico de decodificación, ya que produce una salida a nivel ALTO sólo cuando todas sus entradas están a nivel ALTO. Por tanto, debe asegurarse de que todas las entradas de la puerta AND estén a nivel ALTO cuando se introduce el número 1001, lo cual se puede conseguir invirtiendo los dos bits centrales (cuyos bits son 0), como se muestra en la Figura 41.

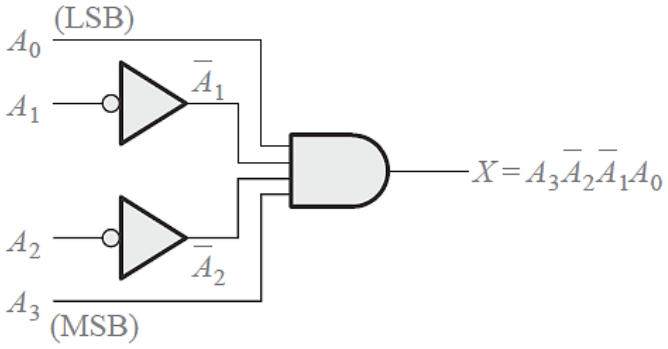


Figura 41 *Lógica de decodificación del código binario 1001 con una salida activa a nivel ALTO.*
Fuente: Rojas (2021).

Se debe comprobar que la salida sea siempre igual a 0, excepto cuando se aplican las entradas $A_0 = 1$, $A_1 = 0$, $A_2 = 0$ y $A_3 = 1$. En este caso, A_0 representa el bit menos significativo (LSB) y A_3 el bit más significativo (MSB).

Si se utiliza una puerta NAND en lugar de una puerta AND, una salida a nivel lógico BAJO indicará la presencia del código binario requerido, que en este caso corresponde a 1001.

3.2.2 Decodificador de 4 bits

Para decodificar todas las combinaciones posibles de cuatro bits, se requieren dieciséis salidas de decodificación, ya que $2^4 = 16$. Este tipo de dispositivo se denomina comúnmente decodificador de 4 líneas a 16 líneas, debido a que dispone de cuatro entradas y dieciséis salidas. También se conoce como decodificador 1 de 16, puesto que, para cualquier código aplicado en las entradas, solo una de las dieciséis salidas posibles se activa (Rojas, 2021).

En la Figura 42 se presenta la lista de los dieciséis códigos binarios y sus correspondientes funciones de decodificación.

Dígito decimal	Entradas binarias A_3	A_2	A_1	A_0	Función de decodificación	Salidas 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figura 42 *Funciones de decodificación y tabla de verdad para un decodificador de 4 líneas a 16 líneas con salidas activas a nivel bajo.*

Fuente: Rojas (2021).

Si se requiere una salida activa a nivel BAJO para cada número decodificado, el decodificador completo puede implementarse mediante puertas NAND e inversores. Para decodificar cada uno de los dieciséis códigos binarios se necesitan dieciséis puertas NAND; alternativamente, las puertas AND pueden emplearse cuando se desean salidas activas a nivel ALTO (Rojas, 2021).

El símbolo lógico de un decodificador de 4 líneas a 16 líneas (1 de 16) con salidas activas a nivel BAJO se muestra en la Figura 43. La etiqueta BIN/DEC indica que una entrada binaria produce su correspondiente salida decimal. Las etiquetas 8, 4, 2 y 1 asociadas a las entradas representan los pesos binarios de los bits de entrada $(2^3, 2^2, 2^1, 2^0)$.

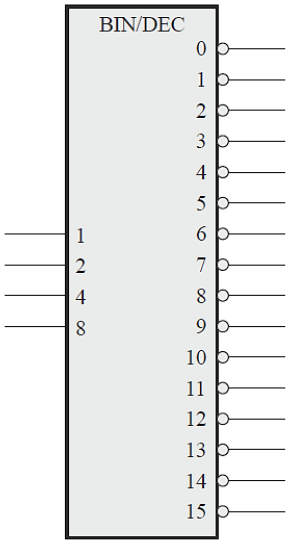
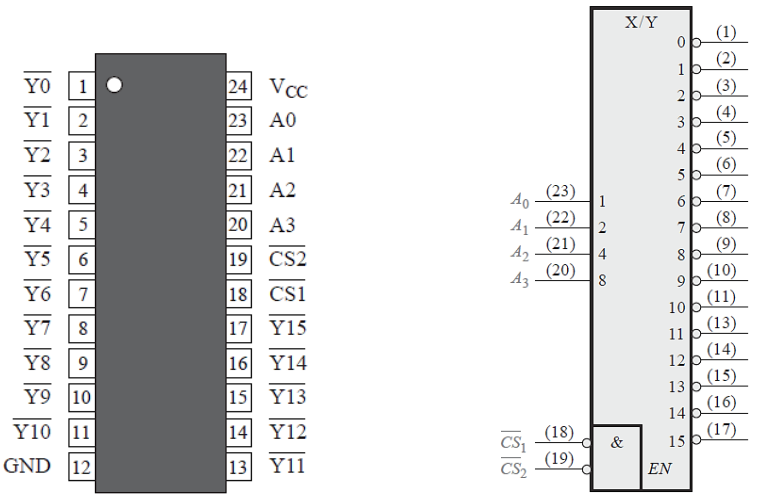


Figura 43 *Símbolo lógico de un decodificador de 4 líneas a 16 líneas.*
Fuente: Rojas (2021).

Decodificador 1 a 16 74HC154

El 74HC154 es un buen ejemplo de un decodificador en circuito integrado. Su diagrama de pines y símbolo lógico se muestran en la Figura 44. En este tipo de dispositivo existe una función de activación (enable, EN), que se implementa mediante una puerta NOR utilizada como negativa-AND. En las entradas de selección del chip, se requiere un nivel BAJO para obtener en la salida de la puerta de activación (EN, enable) un nivel ALTO (Roth & Kinney, 2014).

La salida de la puerta de activación se conecta a una entrada de cada puerta NAND del decodificador, por lo que debe estar a nivel ALTO para que las puertas NAND se activen. Si la puerta de activación no se activa mediante un nivel BAJO en ambas entradas, entonces las dieciséis salidas (Y) del decodificador estarán a nivel ALTO independientemente del estado de las cuatro variables de entrada A0, A1, A2 y A3. Este dispositivo puede estar disponible en otras familias CMOS o TTL.



(a) (b)

Figura 44 *Decodificador 1 de 16 74HC154 (a) diagrama de pines; (b) símbolo lógico.*
Fuente: Roth & Kinney (2014).

Dado que el 74HC154 puede procesar únicamente cuatro bits, es necesario utilizar dos decodificadores para manejar cinco bits (Roth & Kinney, 2014). El quinto bit, A_4, se conecta a las entradas de selección del chip de uno de los decodificadores y a las entradas de activación del otro decodificador, tal como se muestra en la Figura 45.

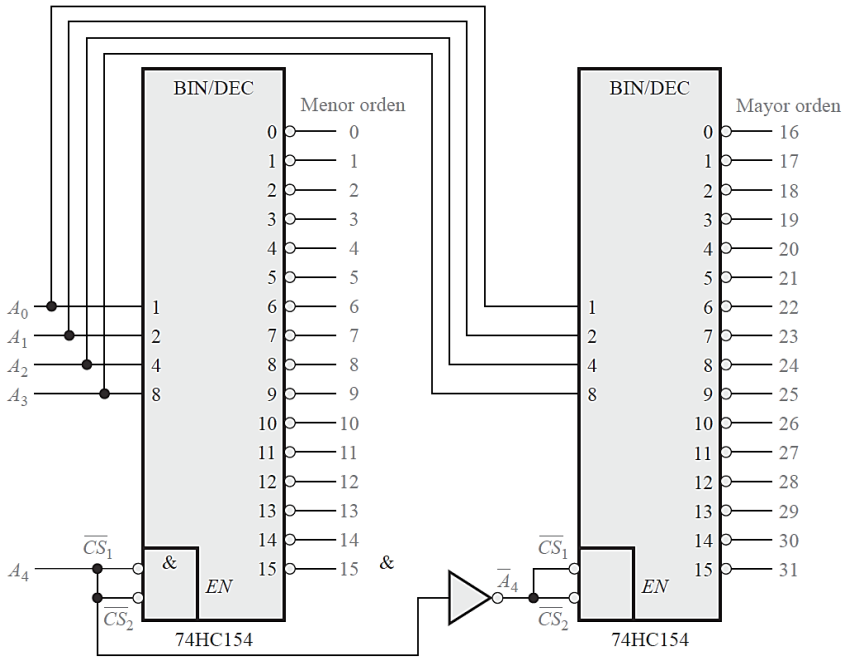


Figura 45 *Decodificador de 5 bits construido con dos 74HC154.*
Fuente: Roth & Kinney (2014).

3.2.3 Decodificador BCD a decimal

Un decodificador BCD a decimal convierte cada código BCD (código 8421) en uno de los diez posibles dígitos decimales. Con frecuencia, este dispositivo se denomina decodificador de 4 líneas a 10 líneas o decodificador 1 de 10.

El método de implementación es el mismo que el empleado para el decodificador de 4 líneas a 16 líneas, con la diferencia de que en este caso sólo se requieren diez puertas de decodificación, ya que el código BCD representa únicamente los dígitos decimales del 0 al 9.

En la Figura 46 se presenta la lista de los diez códigos BCD y sus correspondientes funciones de decodificación. Cada una de estas funciones se implementa mediante puertas NAND, con el fin de proporcionar salidas activas a nivel BAJO. En caso de requerirse salidas activas a nivel ALTO, se emplearían puertas AND para la decodificación. La lógica utilizada es idéntica a la de las diez primeras puertas del decodificador de 4 líneas a 16 líneas (Roth & Kinney, 2014).

Dígito decimal	Código BCD A_3	A_2	A_1	A_0	Función de decodificación
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

Figura 46 *Funciones de decodificación BCD.*

Fuente: Roth & Kinney (2014).

3.2.3.1 Decodificador BCD a decimal 74HC42

El 74HC42 es un circuito integrado decodificador BCD a decimal. Su símbolo lógico se muestra en la Figura 46, y cumple con las funciones de decodificación presentadas en la Figura 47.

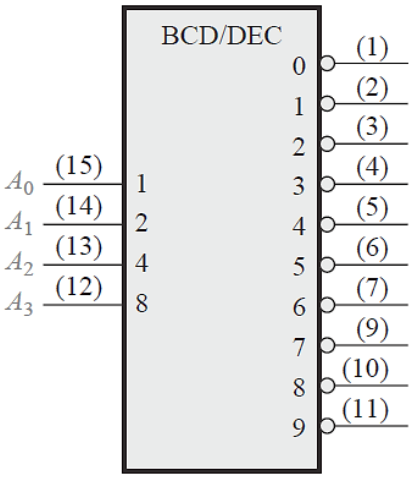


Figura 47 *Símbolo lógico del decodificador BCD a decimal 74HC42.*
Fuente: Roth & Kinney (2014).

3.2.4 Decodificador BCD a 7 segmentos

El decodificador BCD a 7 segmentos recibe el código BCD en sus entradas y proporciona salidas capaces de excitar un display de 7 segmentos, permitiendo la visualización de un dígito decimal. En la Figura 48 se presenta el diagrama lógico de un decodificador básico de 7 segmentos (Roth & Kinney, 2014).

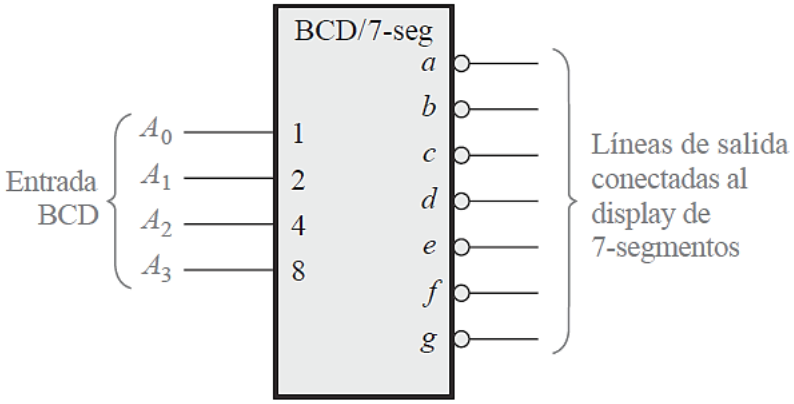


Figura 48 *Símbolo lógico del decodificador / controlador BCD a 7 segmentos.*
Fuente: Roth & Kinney (2014).

3.2.4.1 Decodificador / Controlador BCD a 7 segmentos 74LS47

El 74LS47 es un ejemplo de circuito integrado que decodifica una entrada BCD y controla un display de 7-segmentos. Además de estas características de decodificación y control, el 74LS47 posee características adicionales, como las indicadas en el símbolo lógico de la Figura 49.

Como indican los círculos del símbolo lógico, todas las salidas (de *a* a *g*) son activas a nivel BAJO, al igual que lo son $\overline{LT}$ (Lamp Test, entrada de comprobación), RBI (Ripple Blanking Input) y $\overline{BI}/\overline{RBO}$ (Blanking Input/Ripple Blanking Output).

Las salidas pueden controlar directamente un display de 7-segmentos en ánodo común. Además de decodificar una entrada BCD y generar las apropiadas salidas para 7-segmentos, el 74LS47 posee las funciones de entrada de comprobación y de supresión de cero. Este dispositivo puede estar disponible en otras familias CMOS o TTL (Roth & Kinney, 2014).

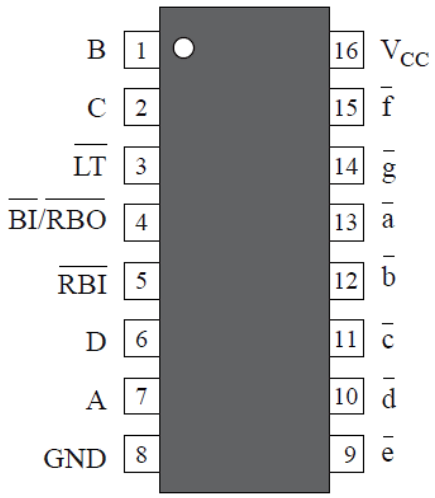


Figura 49 *Diagrama de pines para el codificador / controlador BCD a 7 segmentos.*
Fuente: Roth & Kinney (2014).

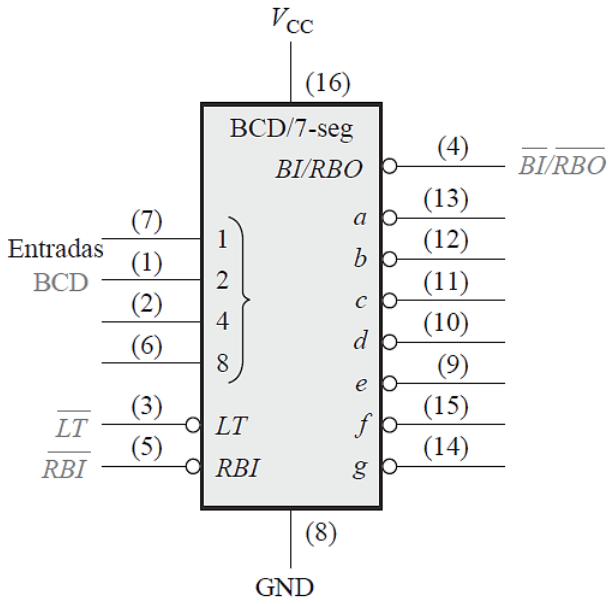


Figura 50 *Símbolo lógico para el codificador / controlador BCD a 7 segmentos*.
Fuente: Roth & Kinney (2014).

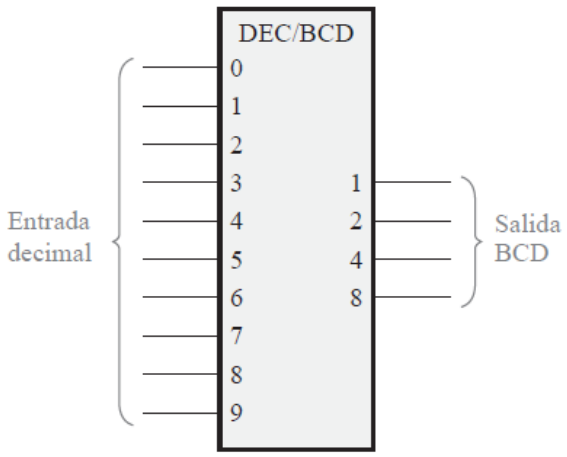
3.3 Codificadores

Un codificador es un circuito lógico combinacional que, esencialmente, realiza la función inversa de un decodificador. Un codificador permite aplicar un nivel activo en una de sus entradas, el cual representa un dígito – como un dígito decimal u octal –, y lo convierte en una salida codificada, por ejemplo, en BCD o binario.

Los codificadores también pueden diseñarse para codificar símbolos y caracteres alfabéticos. El proceso mediante el cual se convierten símbolos comunes o números en un formato codificado se denomina codificación (Roth & Kinney, 2014).

3.3.1 Codificador decimal a BCD

Este tipo de codificador tiene diez entradas, una para cada dígito decimal, y cuatro salidas que corresponden al código BCD, como se muestra en la Figura 51. Este es un codificador básico de 10 líneas a 4 líneas.



(a)

Dígito decimal	Código BCD			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

(b)

Figura 51 *Codificador decimal a BCD (a) símbolo lógico; (b) tabla de verdad.*

Fuente: Roth & Kinney (2014).

El código BCD (8421) se muestra en la Figura 50(b). A partir de esta tabla es posible determinar la relación entre cada bit del código BCD y los dígitos decimales, con el propósito de analizar la lógica del codificador.

Por ejemplo, el bit más significativo del código BCD, A_3, toma el valor 1 únicamente para los dígitos decimales 8 o 9. Por tanto, la expresión OR correspondiente al bit A_3, en función de los dígitos decimales, puede escribirse como:

$$A_3 = 8 + 9$$

El bit A_2 es igual a 1 para los dígitos decimales 4, 5, 6 y 7, y puede expresarse mediante la siguiente función OR:

$$A_2 = 4 + 5 + 6 + 7$$

El bit A_1 toma el valor 1 para los dígitos decimales 2, 3, 6 y 7, por lo que su expresión lógica es:

$$A_1 = 2 + 3 + 6 + 7$$

Finalmente, el bit menos significativo, A_0, es igual a 1 para los dígitos decimales 1, 3, 5, 7 y 9, y su expresión correspondiente es:

$$A_0 = 1 + 3 + 5 + 7 + 9$$

A partir de estas expresiones lógicas, se puede implementar el circuito lógico necesario para codificar en código BCD cada dígito decimal. El proceso consiste simplemente en aplicar la

operación OR a las entradas decimales apropiadas, con el fin de generar cada una de las salidas BCD. La lógica del codificador obtenida a partir de estas expresiones se muestra en la Figura 52.

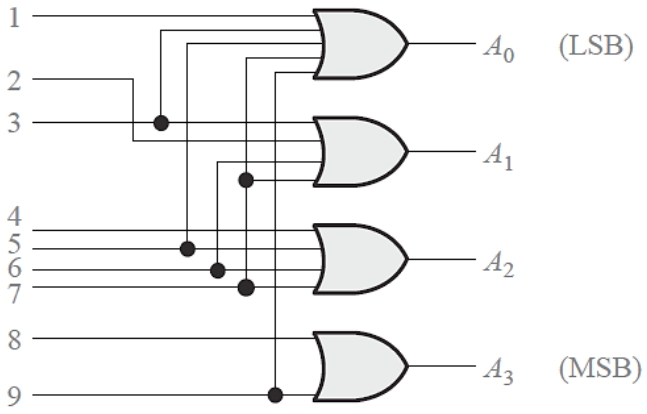
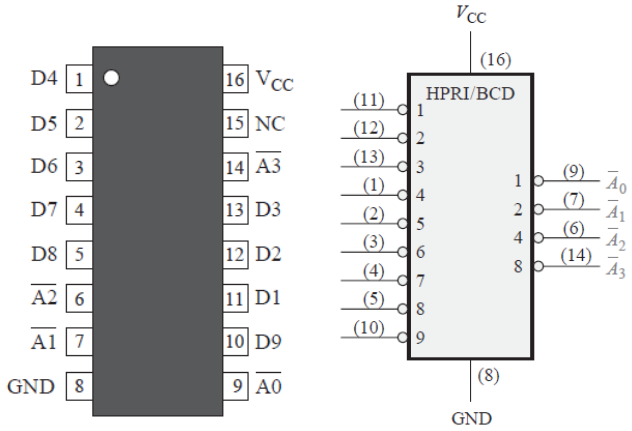


Figura 52 *Diagrama lógico básico de un codificador decimal a BCD.*
Fuente: Roth & Kinney (2014).

3.3.1.1 Codificador decimal a BCD 74HC147

El 74HC147 es un codificador con prioridad que dispone de entradas activas a nivel BAJO (0) correspondientes a los dígitos decimales del 1 al 9, y de salidas BCD activas a nivel BAJO, tal como se indica en el símbolo lógico mostrado en la Figura 53. Una salida BCD igual a cero se obtiene cuando ninguna de las entradas se encuentra activa. La numeración de los pines del dispositivo se presenta entre paréntesis en el símbolo lógico. Este circuito integrado se encuentra disponible en diversas familias lógicas CMOS y TTL (Roth & Kinney, 2014).



(a) (b)

Figura 53 *Codificador con prioridad decimal a BCD (a) símbolo lógico; (b) tabla de verdad.*
Fuente: Roth & Kinney (2014).

Cuando existen niveles BAJOS en varias entradas, el 74HC147 otorga prioridad al dígito decimal de mayor orden y lo representa en la salida. Por ejemplo, si se presentan niveles BAJOS en los pines de entrada 1, 4 y 13, el 74HC147 da prioridad al pin 4, ya que corresponde al dígito decimal de mayor orden entre las entradas activas, y por lo tanto representa el número decimal 7.
En consecuencia, los niveles de salida corresponden al código BCD del decimal 7, donde A_0 es el bit menos significativo (LSB) y A_3 es el bit más significativo (MSB). Para este caso, los niveles de salida son: A_0 a nivel BAJO, A_1 a nivel BAJO, A_2 a nivel BAJO y A_3 a nivel ALTO (Roth & Kinney, 2014).

3.4 Multiplexores y demultiplexores

3.4.1 Multiplexores

Un multiplexor (MUX) es un dispositivo que permite dirigir la información digital procedente de diversas fuentes a una única línea para ser transmitida a través de dicha línea a un destino común. El multiplexor básico posee varias líneas de entrada de datos y una única línea de salida. También posee entradas de selección de datos, que permiten conmutar los datos digitales provenientes de cualquier entrada hacia la línea de salida. A los multiplexores también se les conoce como selectores de datos (Hernández & Labado, 2022).

Un multiplexor (MUX) de cuatro entradas, dispone de dos líneas de selección de datos, dado que con dos bits se puede seleccionar cualquiera de las cuatro líneas de entrada de datos. El símbolo lógico se muestra en la Figura 54.

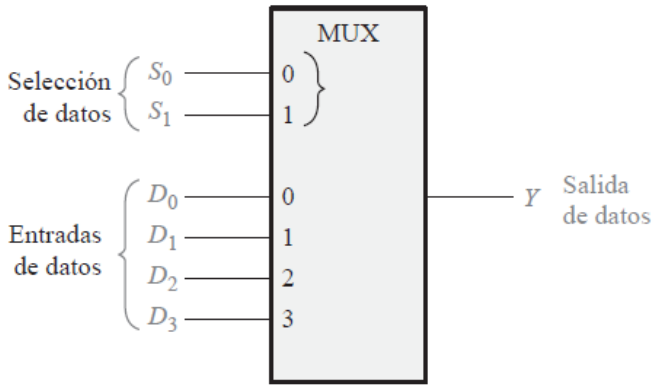


Figura 54 *Diagrama lógico básico de un multiplexor de 1 salida y 4 entradas.*
Fuente: Hernández & Labado (2022).

Un código binario de dos bits en las entradas de selección de datos (S) va a permitir que los datos de la entrada seleccionada pasen a la salida de datos. Si aplicamos un 0 binario (S1 = 0 y S0 = 0) a las líneas de selección de datos, los datos de la entrada D0 aparecerán en la línea de datos de salida. Si aplicamos un 1 binario (S1 = 0 y S0 = 1), los datos de la entrada D1 aparecerán en la salida de datos. Si se aplica un 2 binario (S1 = 1 y S0 = 0), obtendremos en la salida los datos de D2. Si aplicamos un 3 binario (S1 = 1 y S0 = 1), los datos de D3 serán conmutados a la línea de salida. El resumen del funcionamiento se puede ver en la Figura 55 (Hernández & Labado, 2022).

Entradas de selección de datos		**Entrada seleccionada**
S_1	**S_0**	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Figura 55 *Selector de datos de un multiplexor de 1 salida y 4 entradas.*
Fuente: Hernández & Labado (2022).

La salida de datos es igual al estado de la entrada de datos seleccionada. Por lo tanto, es posible deducir una expresión lógica para la salida en función de las entradas de datos y de las entradas de selección:

- La salida de datos es igual a D_0 sólo si $S_1 = 0$ y $S_0 = 0$; $Y = D_0\overline{S_1}\,\overline{S_0}$.
- La salida de datos es igual a D_1 sólo si $S_1 = 0$ y $S_0 = 1$; $Y = D_1\overline{S_1}S_0$.
- La salida de datos es igual a D_2 sólo si $S_1 = 1$ y $S_0 = 0$; $Y = D_2S_1\overline{S_0}$.
- La salida de datos es igual a D_3 sólo si $S_1 = 1$ y $S_0 = 1$; $Y = D_3S_1S_0$.

Al aplicar la operación OR a estos términos, se obtiene la expresión lógica completa para la salida de datos, la cual puede escribirse como:

$$Y = D_0\overline{S_1}\,\overline{S_0} + D_1\overline{S_1}S_0 + D_2S_1\overline{S_0} + D_3S_1S_0$$

La implementación de esta ecuación requiere cuatro puertas AND de tres entradas, una puerta OR de cuatro entradas y dos inversores para generar los complementos de S_1 y S_0, tal como se muestra

en la Figura 56. Dado que los datos pueden seleccionarse desde cualquiera de las líneas de entrada, este circuito también se conoce como selector de datos.

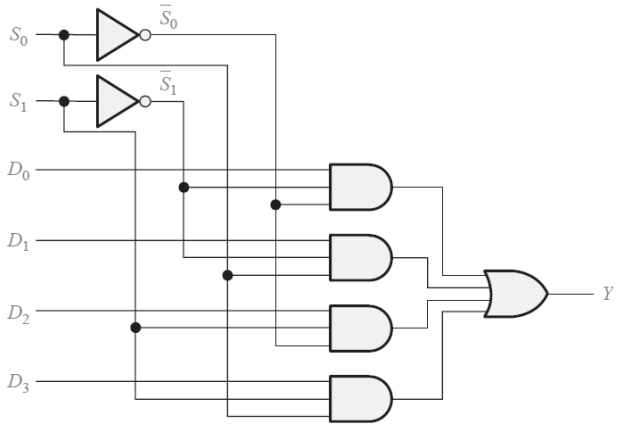


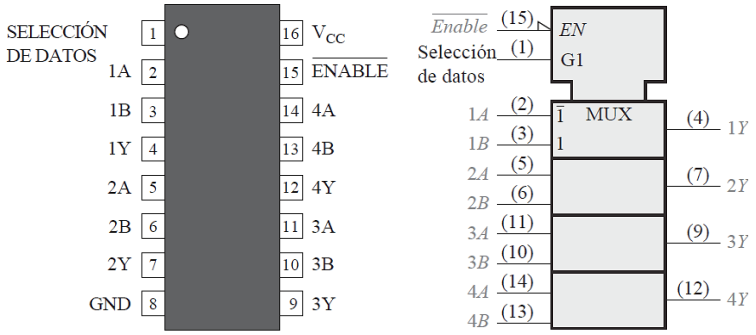
Figura 56 *Diagrama lógico de un multiplexor de 4 entradas.*
Fuente: Hernández & Labado (2022).

3.4.1.1 Cuádruple Multiplexor / Selector de datos de 2 entradas 74HC157

El 74HC157, al igual que su versión LS, está compuesto por cuatro multiplexores de dos entradas. Cada uno de estos multiplexores comparte una misma línea de selección de datos y una línea de habilitación (enable). Dado que en cada multiplexor sólo existen dos entradas de datos seleccionables, es suficiente disponer de una única entrada de selección. Un nivel lógico BAJO en la entrada de habilitación permite que el dato de entrada seleccionado se transfiera a la salida. Por el contrario, un nivel lógico ALTO en esta entrada impide el paso de los datos a la salida, es decir, inhabilita los multiplexores. Este circuito integrado se encuentra disponible en diversas familias lógicas CMOS y TTL (Hernández & Labado, 2022).

En la Figura 57 se muestra el diagrama de pines del 74HC157 junto con su símbolo lógico. Los cuatro multiplexores se representan mediante divisiones internas del bloque, mientras que las entradas comunes a todos ellos se indican en el bloque superior, denominado bloque común de control. Todas las etiquetas mostradas dentro de este bloque superior se aplican igualmente a los bloques inferiores correspondientes a cada multiplexor (Hernández & Labado, 2022).
La etiqueta G1 indica una relación lógica AND entre la entrada de selección de datos y las entradas de datos identificadas por 1 ó $\overline{1}$. El $\overline{1}$ indica que la operación AND se aplica al complemento de la

entrada G1. En otras palabras, cuando la entrada de selección se encuentra a nivel ALTO, se seleccionan las entradas B de los multiplexores, mientras que cuando la entrada de selección está a nivel BAJO, se seleccionan las entradas A. Para indicar una dependencia lógica AND, se utiliza siempre la letra G.



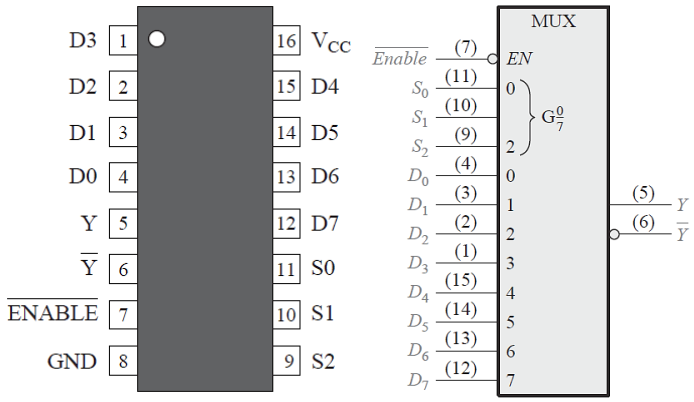
(a) (b)

Figura 57 *Cuádruple multiplexor/selector 74HC157 (a) diagrama de pines; (b) símbolo lógico.*

Fuente: Hernández & Labado (2022).

3.4.1.2 Multiplexor / Selector de datos de 8 entradas 74LS151

El 74LS151 dispone de ocho entradas de datos $(D_0–D_7)$ y, por lo tanto, requiere tres líneas de dirección o de selección de datos $(S_0–S_2)$. Se necesitan tres bits para seleccionar cualquiera de las ocho entradas de datos, ya que $2^3 = 8$. Un nivel lógico BAJO en la entrada de habilitación permite que los datos de entrada seleccionados se transfieran a la salida. En la Figura 58 se presentan el diagrama de pines y el símbolo lógico de este dispositivo (Hernández & Labado, 2022).



(a) (b)

Figura 58 *Multiplexor/selector de datos de 8 entradas 74LS151 (a) diagrama de pines; (b) símbolo lógico.*

Fuente: Hernández & Labado (2022).

No es necesario disponer de un bloque de control común en el símbolo lógico, ya que en este caso sólo se controla un único multiplexor, a diferencia del 74HC157, que integra cuatro multiplexores. La etiqueta $G\frac{0}{7}$ incluida en el símbolo lógico indica la relación lógica AND entre las entradas de selección de datos y cada una de las entradas de datos, desde D0 hasta D7. Este dispositivo se encuentra disponible en diversas familias lógicas CMOS y TTL (Hernández & Labado, 2022).

Para multiplexar 16 líneas de datos en una única línea de salida, se requieren dos multiplexores 74LS151, conectados tal como se muestra en la Figura 59.

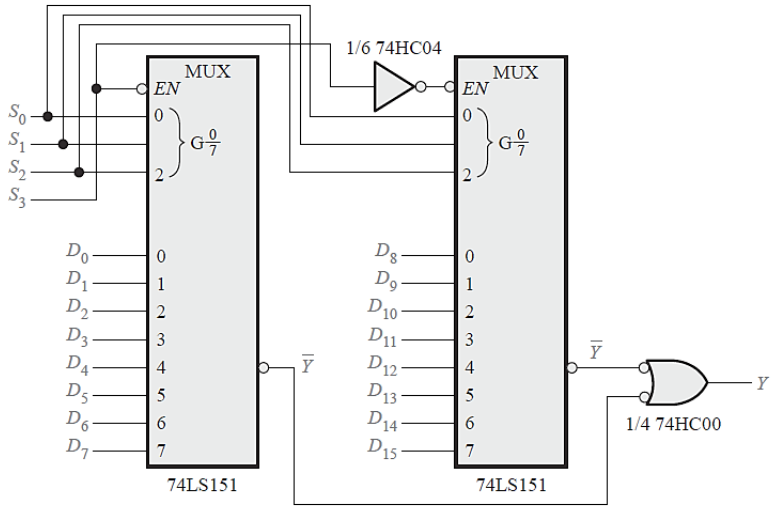


Figura 59 *Multiplexor de 16 entradas utilizando dos multiplexores 74HC157.*

Fuente: Hernández & Labado (2022).

3.4.2 Demultiplexores

Un demultiplexor (DEMUX) realiza, de forma básica, la función contraria a la de un multiplexor. Toma los datos de una única línea de entrada y los distribuye hacia un determinado número de líneas de salida. Por esta razón, el demultiplexor también se conoce como distribuidor de datos. Como se verá más adelante, los decodificadores pueden emplearse igualmente como demultiplexores (Hernández & Labado, 2022).

La Figura 60 muestra un circuito demultiplexor de 1 línea a 4 líneas. La línea de entrada de datos se conecta a todas las puertas AND, mientras que las dos líneas de selección activan una única puerta a la vez. De este modo, los datos presentes en la entrada se transmiten a través de la puerta seleccionada hasta la línea de salida correspondiente.

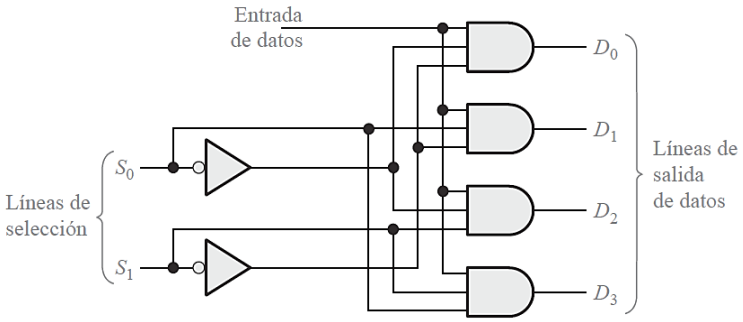


Figura 60 *Símbolo lógico para el cuádruple multiplexor/selector de datos de 2 entradas 74HC157.*
Fuente: Hernández & Labado (2022).

3.4.2.1 Demultiplexor 74HC154

Este dispositivo, así como otros decodificadores, se utiliza también en diversas aplicaciones como demultiplexor. El símbolo lógico de este circuito, cuando se emplea como demultiplexor, se muestra en la Figura 61. Cuando se utiliza con este fin, se usan las líneas de entrada como líneas de selección de datos, una de las entradas de activación del chip se usa como línea de entrada de datos y la otra se mantiene a nivel BAJO, para activar la puerta interna NAND que se encuentra en la parte inferior del diagrama. Este dispositivo puede estar disponible en otras familias CMOS o TTL (Hernández & Labado, 2022).

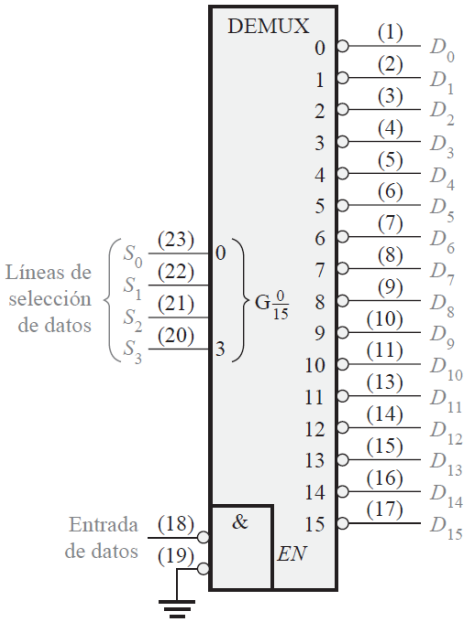


Figura 61 *Símbolo lógico para el decodificador 74HC154 utilizado como demultiplexador.*
Fuente: Hernández & Labado (2022).

ACTIVIDADES

Actividad 1: Cuestionario sobre sumadores.

1. Escribir la definición de sumador y explicar la diferencia entre semi sumador y sumador completo.
2. Graficar el símbolo lógico y el diagrama de pines del sumador en paralelo de 4 bits 74LS283.
3. Construir la tabla de verdad de un sumador completo.

Actividad 2: Cuestionario sobre comparadores.

1. Explicar el principio de funcionamiento de un comparador de magnitud.
2. Dibujar el símbolo lógico y el diagrama de pines del comparador 74HC85.
3. Explicar cómo se pueden conectar varios comparadores en cascada para comparar números binarios de más de 4 bits.

Actividad 3: Cuestionario sobre decodificadores.

1. Definir qué es un decodificador y mencionar dos aplicaciones prácticas.
2. Construir la tabla de verdad de un decodificador BCD a decimal.
3. Dibujar el símbolo lógico del decodificador 74HC42 e indicar el tipo de salidas que posee (activas a nivel ALTO o BAJO).

Actividad 4: Cuestionario sobre codificadores.

1. Definir qué es un codificador y explicar su diferencia con un decodificador.
2. Explicar el funcionamiento de un codificador con prioridad.
3. Dibujar el símbolo lógico del codificador 74HC147 y describir qué sucede cuando varias entradas se encuentran activas simultáneamente.mbolo lógico y diagrama de pines del codificador decimal a BCD 74HC147**.**

Actividad 5: Cuestionario sobre multiplexores y demultiplexores.

1. Escribir la definición de multiplexor.

2. Graficar el símbolo lógico y diagrama de pines del multiplexor de 2 entradas 74HC157.
3. Escribir la definición de demultiplexor.
4. Graficar el símbolo lógico y diagrama de pines del decodificador 74HC154 utilizado como demultiplexador.

Actividad 6: Autoevaluación.

1. Realizar el diseño de un sumador en paralelo de 12 bits, empleando sumadores 74LS283.

2. Determinar las salidas A = B, A > B y A < B para los números de entrada mostrados en el comparador de la siguiente figura:

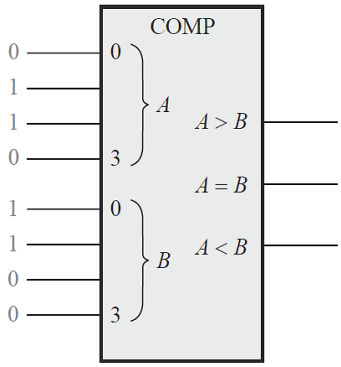


Figura 62 *Comparador de dos números de 4 bits.*
Fuente: Hernández & Labado (2022).

3. Realizar el diseño de un decodificador de 6 bits, empleando decodificadores 74HC154.

4. Indicar el estado de las salidas de un codificador 74HC147, si se tiene en las entradas niveles BAJOS en los pines 11, 13 y 5.

5. Realizar el diseño de un multiplexor de 24 entradas utilizando multiplexores 74HC157.

6. Un demultiplexor 74HC154 tiene en las líneas de selección de datos el código binario 1010 y la línea de entrada de datos está a nivel BAJO. Determinar los estados de las líneas de salida.

UNIDAD 4: ANÁLISIS LÓGICO SECUENCIAL

4.1 Señales de reloj y multivibradores

En electrónica y, en particular, en los circuitos digitales síncronos, una señal de reloj es una señal periódica utilizada para coordinar y sincronizar las acciones de dos o más circuitos digitales.
Los circuitos que emplean una señal de reloj para su sincronización pueden activarse en el flanco de subida, en el flanco de bajada o en ambos flancos. Un ejemplo de este último caso son las memorias DDR SDRAM, las cuales realizan operaciones tanto en el flanco ascendente como en el flanco descendente de la señal de reloj.

Una señal de reloj alterna periódicamente entre un nivel lógico ALTO y un nivel lógico BAJO y, desde el punto de vista gráfico, se representa comúnmente como una onda cuadrada, caracterizada por parámetros como el período, la frecuencia, el ancho de pulso y los flancos de transición.

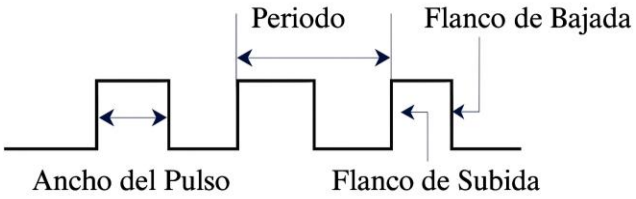


Figura 63 *Señal de reloj.*
Fuente: Hill & Peterson (2014).

La mayoría de los circuitos integrados complejos utilizan una señal de reloj para sincronizar sus diferentes bloques internos y para controlar los tiempos de propagación de las señales. A medida que aumenta la complejidad de los circuitos, la sincronización mediante la señal de reloj se vuelve más crítica y difícil de implementar. Un ejemplo representativo de circuito integrado complejo es el microprocesador, en el cual la correcta sincronización es fundamental para su funcionamiento adecuado (Hill & Peterson, 2014).

Un multivibrador es un circuito electrónico capaz de generar trenes de ondas cuadradas, utilizados comúnmente como señales de temporización o control. Existen tres tipos fundamentales de multivibradores: los monoestables, los astables o inestables, y los biestables.

4.1.1 Multivibrador monoestable

El monoestable es un circuito multivibrador que realiza una función secuencial, consistente en que, al recibir una excitación externa, cambia de estado y permanece en dicho estado durante un intervalo de tiempo determinado por una constante de tiempo del circuito. Transcurrido este período, la salida del monoestable retorna automáticamente a su estado original (Hill & Peterson, 2014).

En la Figura 64 se muestra el esquema de un circuito multivibrador monoestable, implementado mediante componentes discretos.

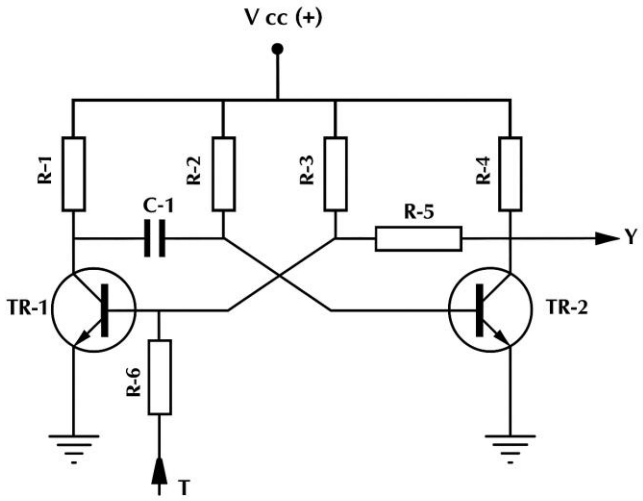


Figura 64 *Esquema de un circuito multivibrador monoestable.*
Fuente: Hill & Peterson (2014).

El funcionamiento del circuito multivibrador monoestable es el siguiente:

Al aplicar la tensión de alimentación (V_{CC}), ambos transistores comienzan a conducir, ya que sus bases reciben un potencial positivo a través de las resistencias R-2 y R-3. Sin embargo, debido a que los transistores no son exactamente idénticos – como consecuencia del proceso de fabricación y del grado de impurezas del material semiconductor –, uno de ellos conducirá antes o con mayor rapidez que el otro.

Supongamos que el transistor TR-2 es el que conduce primero. En este caso, la tensión en su colector será cercana a 0 V, por lo que la salida Y se encontrará a nivel BAJO. En estas condiciones,

la tensión aplicada a la base de TR-1, a través del divisor formado por R-3 y R-5, resulta insuficiente para que TR-1 entre en conducción, por lo que TR-1 permanece bloqueado de forma indefinida.

Si posteriormente se aplica un impulso de disparo de nivel ALTO en la entrada T, el transistor TR-1 comenzará a conducir y su tensión de colector descenderá hasta un valor próximo a 0 V. Como consecuencia, el condensador C-1, que se encontraba cargado a través de R-1 y de la unión base-emisor de TR-2, se descargará a través de TR-1 y R-2, aplicando un potencial negativo a la base de TR-2, lo que provocará su corte. En esta situación, la salida Y pasa a un nivel ALTO.

En este estado, la tensión aplicada a la base de TR-1 es suficiente para mantenerlo en conducción, aun cuando el impulso de disparo en la entrada T haya desaparecido.

A continuación, se inicia la carga del condensador C-1 a través de la resistencia R-2 y del transistor TR-1, hasta que la tensión en el punto común entre C-1 y R-2 – correspondiente a la base de TR-2 – alcanza un valor suficiente para que TR-2 vuelva a conducir, provocando el bloqueo de TR-1. De este modo, el circuito retorna a su estado estable inicial.

La duración del período cuasiestable del multivibrador monoestable viene determinada por los valores de C-1 y R-2, que definen la constante de tiempo del circuito (Hill & Peterson, 2014).

4.1.2 Multivibrador astable

En electrónica, un astable es un circuito multivibrador que no posee ningún estado estable, lo que significa que presenta dos estados inestables entre los cuales conmuta de forma continua, permaneciendo en cada uno de ellos durante un intervalo de tiempo determinado.

La frecuencia de conmutación depende, en general, de los procesos de carga y descarga de los condensadores del circuito. Entre sus principales aplicaciones se encuentran la generación de ondas periódicas, como señales de reloj, y la producción de trenes de pulsos.

En la Figura 65 se muestra el esquema de un circuito multivibrador astable, implementado mediante componentes discretos.

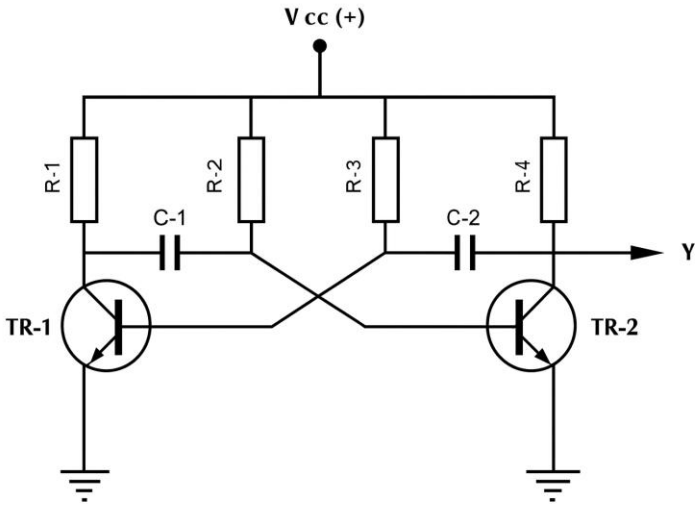


Figura 65 *Esquema de un circuito multivibrador astable.*
Fuente: Hill & Peterson (2014).

El funcionamiento del circuito multivibrador astable es el siguiente:

Al aplicar la tensión de alimentación (V_{CC}), ambos transistores comienzan a conducir, ya que sus bases reciben un potencial positivo a través de las resistencias R-2 y R-3. No obstante, debido a que los transistores no son exactamente idénticos – como consecuencia del proceso de fabricación y del grado de impurezas del material semiconductor –, uno de ellos conducirá antes o con mayor rapidez que el otro (Hill & Peterson, 2014).

Supongamos que el transistor TR-1 es el que conduce primero. En estas condiciones, la tensión en su colector será cercana a 0 V, por lo que el condensador C-1 comenzará a cargarse a través de la resistencia R-2. Inicialmente, esta carga genera una pequeña diferencia de potencial entre las placas del condensador, trasladando una tensión próxima a 0 V a la base de TR-2, lo que provoca que dicho transistor entre en estado de corte. Cuando la tensión en C-1 alcanza aproximadamente 0,6 V, el transistor TR-2 comienza a conducir, haciendo que la salida pase a nivel BAJO (tensión próxima a 0 V). En ese instante, el condensador C-2, que se había cargado previamente a través de R-4 y de la unión base-emisor de TR-1, se descarga, provocando el bloqueo de TR-1.
A continuación, el condensador C-2 inicia su proceso de carga a través de la resistencia R-3 y, al alcanzar una tensión cercana a 0,6 V, provoca nuevamente la conducción de TR-1, la descarga de C-1, el bloqueo de TR-2 y el paso de la salida Y a nivel ALTO (tensión próxima a V_{CC}).

A partir de este punto, la secuencia se repite de forma indefinida, alternándose los estados de conducción y corte de los transistores. Los tiempos de conducción y bloqueo de cada transistor dependen de las constantes de tiempo definidas por las relaciones R-2/C-1 y R-3/C-2. Dado que estos tiempos no tienen por qué ser iguales, es posible obtener diferentes ciclos de trabajo modificando los valores de dichos componentes (Hill & Peterson, 2014).

4.1.3 Multivibrador biestable

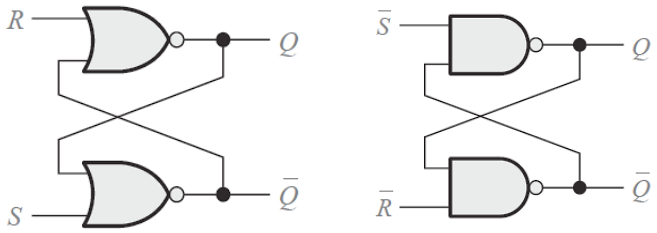
Un biestable (en inglés Flip-Flop, si es síncrono, y Latch, si es asíncrono), es un multivibrador capaz de permanecer en uno de dos estados posibles durante un tiempo indefinido en ausencia de perturbaciones. Esta característica es ampliamente utilizada en electrónica digital para memorizar información. El paso de un estado a otro se realiza variando sus entradas. Dependiendo del tipo de dichas entradas los biestables se dividen en:

- **Asíncronos. -** Solamente tienen entradas de control. El más empleado es el biestable RS.
- **Síncronos. -** Además de las entradas de control posee una entrada de sincronismo o de reloj.

El Latch (cerrojo) es un tipo de dispositivo de almacenamiento temporal de dos estados (biestable), que se suele agrupar en una categoría diferente a la de los Flip-Flops. Básicamente, los Latches son similares a los Flip-Flops, ya que son también dispositivos de dos estados que pueden permanecer en cualquiera de sus dos estados gracias a su capacidad de realimentación, lo que consiste en conectar (realimentar) cada una de las salidas a la entrada opuesta. La diferencia principal entre ambos tipos de dispositivos está en el método empleado para cambiar de estado.

Latch S-R (SET-RESET)

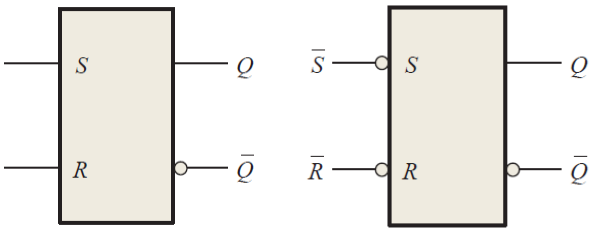
Un Latch es un tipo de dispositivo lógico biestable o multivibrador. Un Latch S-R (Set-Reset) con entrada activa a nivel ALTO se compone de dos puertas NOR acopladas, tal como se muestra en la Figura 66 (a); un Latch con entrada activa a nivel BAJO está formado por dos puertas NAND conectadas tal como se muestra en la Figura 66 (b). La salida de cada puerta se conecta a la entrada de la puerta opuesta. Esto origina la realimentación (feedback) regenerativa característica de todos los Latches y Flip-Flops (Mano & Ciletti, 2014).



(a) (b)

Figura 66 *(a) Latch S-R con entrada activa a nivel alto; (b) Latch $\bar{S}$-$\bar{R}$ con entrada activa a nivel bajo.*
Fuente: Mano & Ciletti (2014).

Los símbolos lógicos para los Latches S-R y $\bar{S}$-$\bar{R}$ se muestran en la Figura 67. La tabla de verdad para un Latch $\bar{S}$-$\bar{R}$ con entrada activa a nivel BAJO se observa en la Figura 68.



(a) (b)

Figura 67 *Símbolo lógico Latch S-R (a) con entrada activa a nivel alto; (b) con entrada activa a nivel bajo.*
Fuente: Mano & Ciletti (2014).

Entradas		Salidas		
$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	*Comentarios*
1	1	NC	NC	No cambio. El latch permanece en el estado que estaba.
0	1	1	0	Latch en estado SET.
1	0	0	1	Latch en estado RESET.
0	0	1	1	Condición no válida

Figura 68 *Tabla de verdad para un Latch $\bar{S}$-$\bar{R}$ con entrada activa a nivel bajo.*
Fuente: Mano & Ciletti (2014).

Un Latch puede permanecer en uno de dos estados: SET o RESET. El estado SET indica que la salida Qse encuentra a nivel ALTO, mientras que el estado RESET indica que la salida Qestá a nivel BAJO. Al aplicar determinadas formas de onda a las entradas $\bar{S}$y $\bar{R}$ del Latch, es posible determinar la correspondiente forma de onda en la salida Q(Mano & Ciletti, 2014).

Ejemplo

Determinar la salida Q de un Latch $\overline{S}$-$\overline{R}$, si se aplican las formas de onda de la Figura 69 a las entradas del Latch. Suponer que Q se encuentra inicialmente a nivel BAJO.

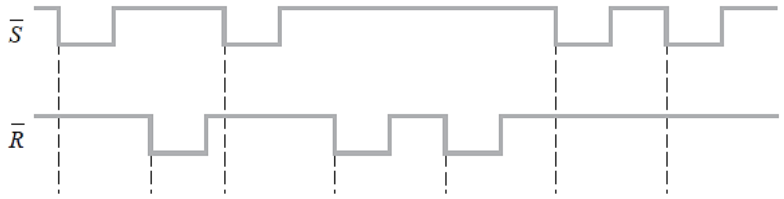


Figura 69 *Entradas aplicadas al Latch $\bar{S}$-$\bar{R}$.*
Fuente: Mano & Ciletti (2014).

Solución

La salida Q se determina en función de la tabla de verdad de un Latch $\bar{S}$–$\bar{R}$ con entradas activas a nivel BAJO, al cual se le aplican las formas de onda mostradas en la Figura 69. El comportamiento resultante de la salida Qse presenta en la Figura 70.

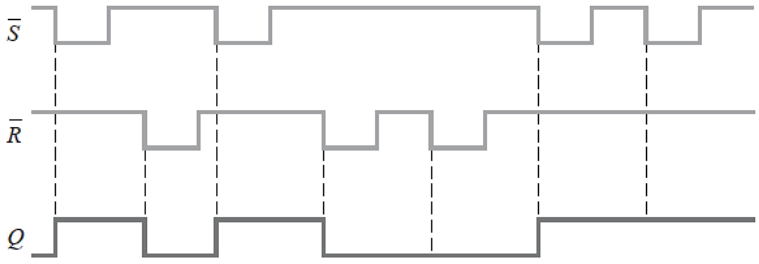


Figura 70 *Salida Q de un Latch $\bar{S}$-$\bar{R}$.*
Fuente: Mano & Ciletti (2014).

Latch SET-RESET 74LS279

El 74LS279 es un circuito integrado cuádruple Latch $\bar{S}$–$\bar{R}$. Su símbolo lógico se presenta en la Figura 71, mientras que el diagrama de pines se muestra en la Figura 72. Cabe destacar que dos de los Latches incorporan dos entradas $\bar{S}$, característica que amplía sus posibilidades de aplicación en sistemas de control y memoria básica (Mano & Ciletti, 2014).

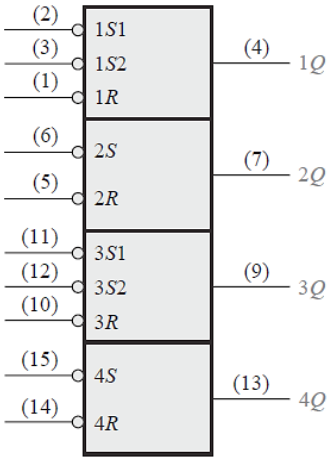


Figura 71 *Símbolo lógico del cuádruple Latch $\bar{S}$-$\bar{R}$ 74LS279.*
Fuente: Mano & Ciletti (2014).

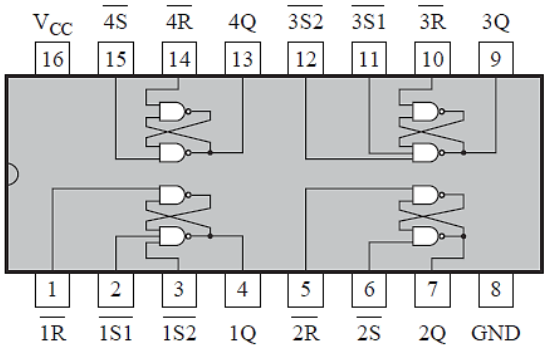


Figura 72 *Diagrama de pines del cuádruple Latch $\bar{S}$-$\bar{R}$ 74LS279.*
Fuente: Mano & Ciletti (2014).

Latch S-R con entrada de habilitación

El diagrama y el símbolo lógico de un Latch con entrada de habilitación se muestran en la Figura 73 y Figura 74. Las entradas *S* y *R* controlan el estado al que va a cambiar el Latch cuando se aplica un nivel ALTO a la entrada de habilitación (*EN, enable*) (Mano & Ciletti, 2014).

El Latch no cambia de estado hasta que la entrada *EN* está a nivel ALTO, pero, mientras que permanezca en este estado, la salida va a ser controlada por el estado de las entradas *S* y *R*. En este circuito, el estado no válido del Latch se produce cuando las dos entradas *S* y *R* están simultáneamente a nivel ALTO (Mano & Ciletti, 2014).

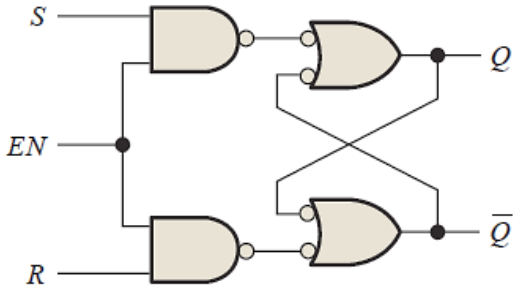


Figura 73 *Diagrama lógico de un Latch S-R con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

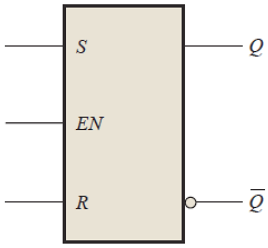


Figura 74 *Símbolo lógico de un Latch S-R con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

Ejemplo

Determinar la forma de onda de salida Q, si se aplican las señales de entrada mostradas en la Figura 75 a un Latch S-R con entrada de habilitación, que se encuentra inicialmente en estado de RESET.

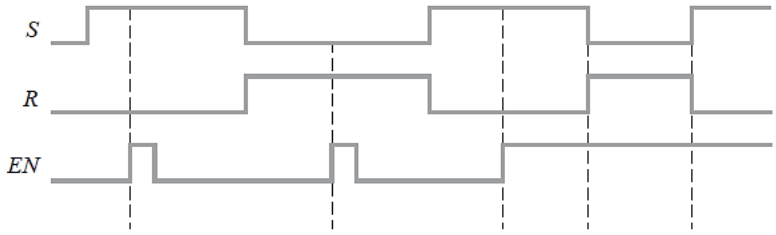


Figura 75 *Entradas aplicadas al Latch S-R con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

Solución

Siempre que la entrada S se encuentra a nivel ALTO y la entrada R a nivel BAJO, un nivel ALTO en la entrada de habilitación (EN) provoca que el Latch adopte el estado SET. De manera análoga,

cuando S está a nivel BAJO y R a nivel ALTO, un nivel ALTO en la entrada EN hace que el Latch pase al estado RESET. La forma de onda correspondiente a la salida Qse muestra en la Figura 76.

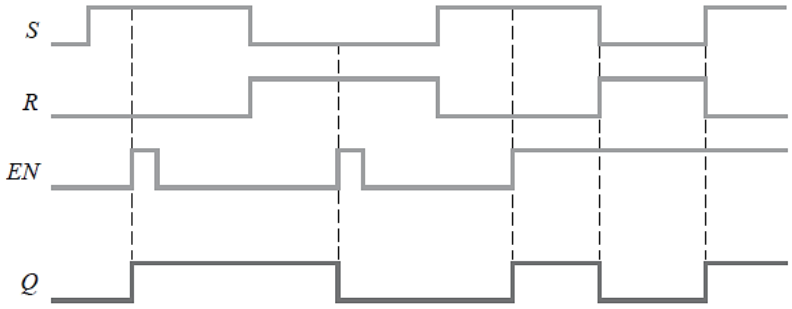


Figura 76 *Salida Q para el Latch S-R con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

Latch D con entrada de habilitación

Existe otro tipo de Latch con entrada de habilitación que se denomina Latch D. Se diferencia del Latch S-R en que sólo tiene una entrada, además de la de habilitación, *EN*. Esta entrada recibe el nombre de entrada de datos (*D*) (Mano & Ciletti, 2014).

La Figura 77 muestra el diagrama y la Figura 78 el símbolo lógico de este tipo de Latch. Cuando la entrada *D* está a nivel ALTO y la entrada *EN* también, el Latch se pone en estado SET. Cuando la entrada *D* está a nivel BAJO y la entrada *EN* está a nivel ALTO, el Latch se pone en estado RESET. Dicho de otra manera, la salida *Q* es igual a la entrada *D* cuando la entrada de habilitación *EN* está a nivel ALTO (Mano & Ciletti, 2014).

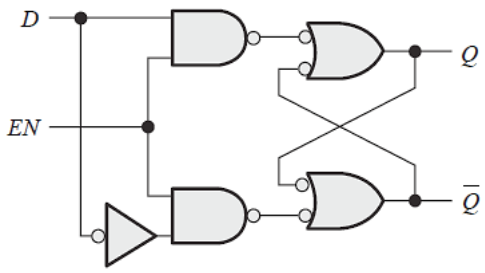


Figura 77 *Diagrama lógico para el Latch D con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

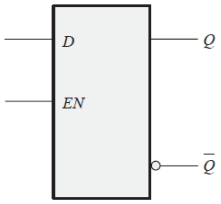


Figura 78 *Símbolo lógico para el Latch D con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

Ejemplo

Determinar la forma de onda de la salida Q cuando se aplican las señales de entrada mostradas en la Figura 79 a un Latch D con entrada de habilitación, considerando que el circuito se encuentra inicialmente en estado RESET.

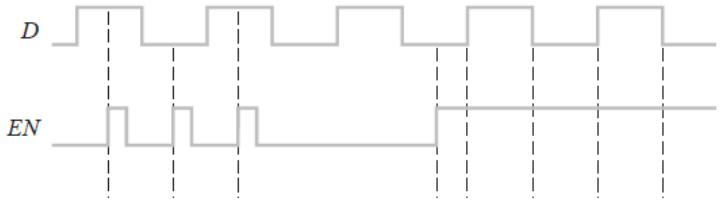


Figura 79 *Entradas aplicadas al Latch D con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

Solución

Siempre que las entradas D y EN se encuentren a nivel ALTO, la salida Qadoptará un nivel ALTO. Cuando la entrada D esté a nivel BAJO y EN permanezca a nivel ALTO, la salida Qpasará a nivel BAJO. En cambio, cuando EN se encuentra a nivel BAJO, el estado del latch no se ve afectado por la entrada D, manteniéndose el último valor almacenado. La forma de onda correspondiente a la salida Qse muestra en la Figura 80.

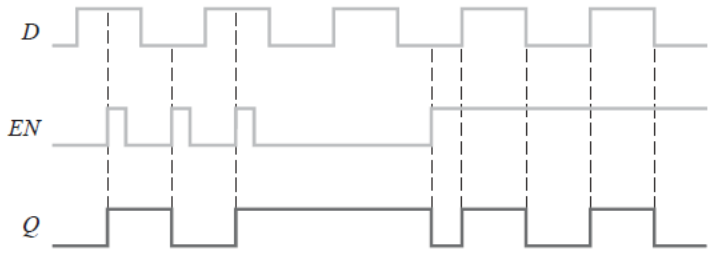


Figura 80 *Salida Q del Latch D con entrada de habilitación.*
Fuente: Mano & Ciletti (2014).

Latch D 74LS75

Un ejemplo de latch D con entrada de habilitación es el 74LS75, cuyo símbolo lógico se presenta en la Figura 81. Este dispositivo está compuesto por cuatro latches D. Cada entrada de habilitación EN, activa a nivel ALTO, es compartida por dos latches y se designa como entrada de control (C).

La tabla de verdad correspondiente a cada latch se muestra en la Figura 82. En dicha tabla, el símbolo X representa una condición indiferente (*don't care*). En este caso, cuando la entrada EN se encuentra a nivel BAJO, el valor de la entrada D es irrelevante, ya que las salidas no se ven afectadas y permanecen en el estado previamente almacenado (Floyd, 2006).

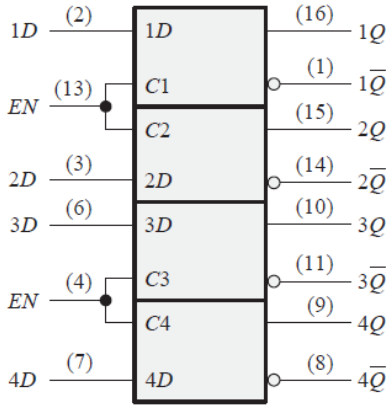


Figura 81 *Símbolo lógico para el cuádruple Latch D con entrada de habilitación 74LS75.*
Fuente: Mano & Ciletti (2014).

Entradas		Salidas		
D	EN	Q	$\overline{Q}$	Comentarios
0	1	0	1	RESET
1	1	1	0	SET
X	0	Q_0	$\overline{Q}_0$	No cambio

Nota: Q_0 es el nivel de salida previo antes de que se establecieran las condiciones de entrada indicadas

Figura 82 *Tabla de verdad para el cuádruple Latch D con entrada de habilitación 74LS75.*
Fuente: Mano & Ciletti (2014).

4.2 Lógica secuencial sincronismo

Si las entradas de control dependen de la de sincronismo se denominan síncronas y en caso contrario asíncronas. Por lo general, las entradas de control asíncronas prevalecen sobre las síncronas (Torres, 2020).

La entrada de sincronismo puede ser activada por nivel (alto o bajo) o por flanco (de subida o de bajada). Dentro de los biestables síncronos activados por nivel están los tipos RS y D, y dentro de los activos por flancos los tipos JK, T y D.

Los biestables síncronos activos por flanco (Flip-Flop) se crearon para eliminar las deficiencias de los Latches (biestables asíncronos o sincronizados por nivel).

4.3 Celdas binarias, Flip/Flops

Los Flip-Flops son dispositivos síncronos de dos estados, también conocidos como multivibradores biestables. En este caso, el término síncrono significa que la salida cambia de estado únicamente en un instante específico de una entrada de disparo denominada reloj (CLK), la cual recibe el nombre de entrada de control, C. Esto significa que los cambios en la salida se producen sincronizadamente con el reloj.

Un Flip-Flop disparado por flanco cambia de estado con el flanco positivo (flanco de subida) o con el flanco negativo (flanco de bajada) del impulso de reloj y es sensible a sus entradas sólo en esta transición del reloj. Los símbolos lógicos los Flip-Flops disparados por flanco: S-R, D y J-K se muestran en la Figura 83. Estos dispositivos pueden ser disparados por flanco positivo (no hay círculo en la entrada C) o por flanco negativo (hay un círculo en la entrada C). La clave para identificar un Flip-Flop disparado por flanco mediante su símbolo lógico la da el triángulo que se encuentra dentro del bloque en la entrada del reloj (C). El triángulo se denomina indicador de entrada dinámica (Mano & Ciletti, 2014).

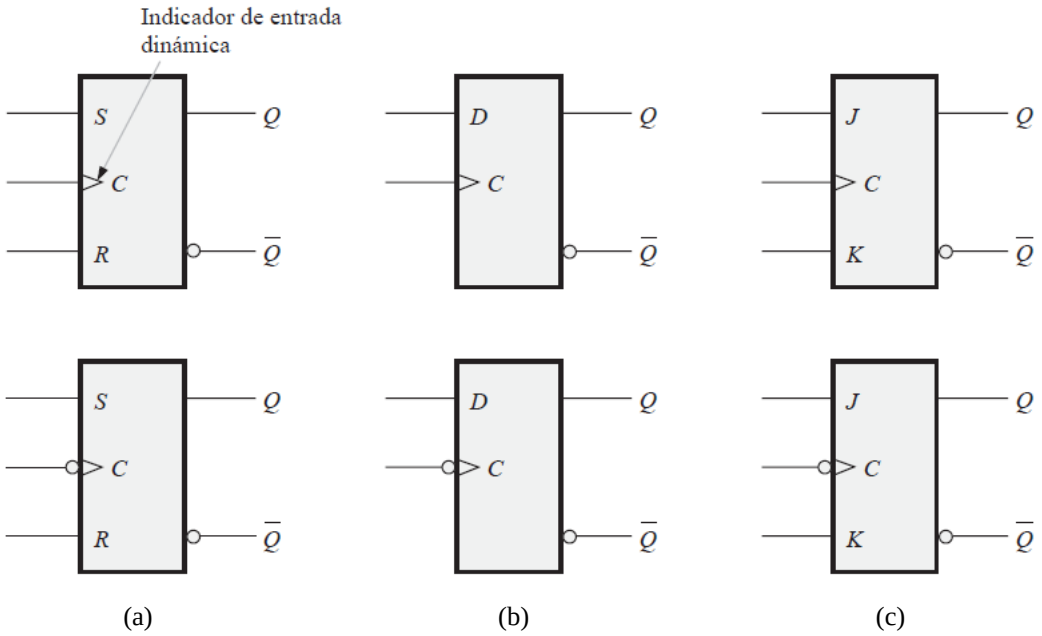


Figura 83 *Símbolos lógicos de los Flip-Flop disparados por flanco (parte superior: disparado por flanco positivo; parte inferior: disparado por flanco negativo): (a) S-R; (b) D; (c) J-K.*

Fuente: Mano & Ciletti (2014).

Flip-flor S-R disparado por flanco

Las entradas S y R de un Flip-Flop S-R se denominan entradas síncronas, dado que los datos en estas entradas se transfieren a las salidas del Flip-Flop sólo con el flanco de disparo del impulso del reloj. Cuando S está a nivel ALTO y R está a nivel BAJO, la salida Q se pone a nivel ALTO con el flanco de disparo del impulso de reloj, pasando el Flip-Flop al estado SET. Cuando S está a nivel BAJO y R está a nivel ALTO, la salida Q se pone a nivel BAJO con el flanco de disparo del impulso de reloj, pasando el Flip-Flop al estado RESET. Cuando tanto S como R están a nivel BAJO, la salida no cambia de estado. Cuando S y R están a nivel ALTO, se produce una condición no válida (Mano & Ciletti, 2014).

El funcionamiento básico de un Flip-Flop disparado por flanco positivo se muestra en la Figura 84, mientras que la tabla de verdad se puede ver en la Figura 85. Recordemos que un Flip-Flop no puede cambiar de estado excepto en el flanco de disparo de un impulso de reloj. Las entradas S y R se pueden cambiar en cualquier instante en que la entrada de reloj esté a nivel ALTO o nivel

BAJO (excepto durante un breve instante de tiempo en las proximidades de las transiciones de disparo del reloj) sin que varíe la salida válida (Mano & Ciletti, 2014).

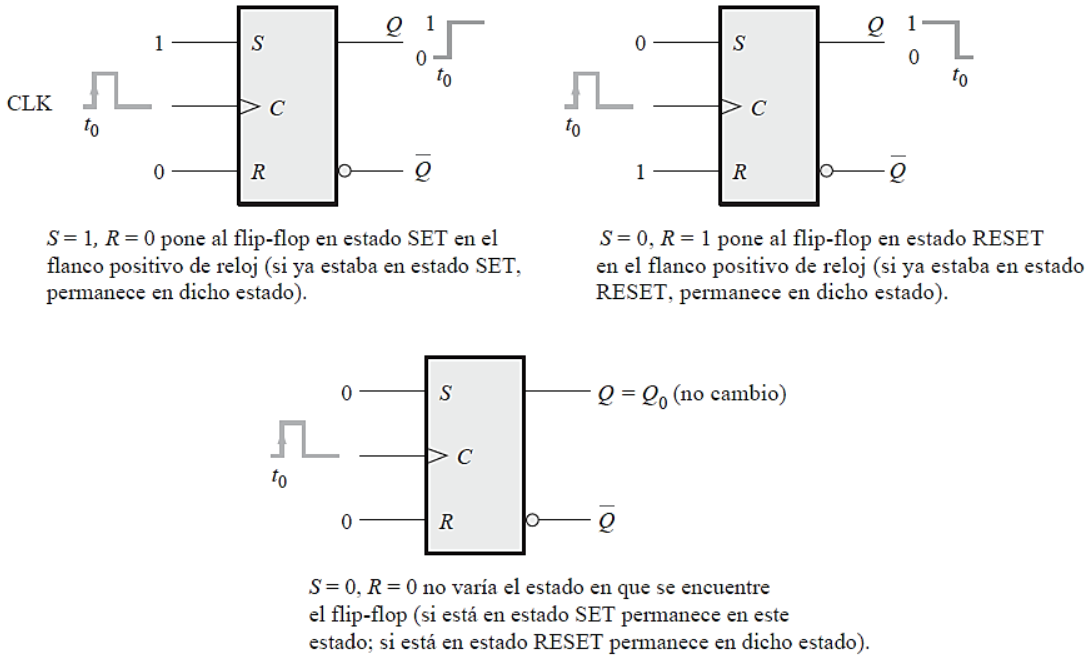


Figura 84 *Funcionamiento de un Flip-Flop S-R disparado por flanco positivo.*

Fuente: Mano & Ciletti (2014).

Entradas			Salidas		
S	***R***	***CLK***	***Q***	$\bar{Q}$	***Comentarios***
0	0	X	Q_0	$\bar{Q}_0$	No cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	No válida

↑ = transición del reloj de nivel BAJO a nivel ALTO
X = irrelevante ("condición indiferente")
Q_0 = nivel de salida previo a la transición del reloj

Figura 85 *Tabla de verdad de un Flip-Flop S-R disparado por flanco positivo.*

Fuente: Mano & Ciletti (2014).

El funcionamiento y tabla de verdad de un Flip-Flop S-R disparado por flanco negativo son las mismas que las de un dispositivo disparado por flanco positivo, excepto en que el flanco de bajada del impulso del reloj es, en este caso, el flanco de disparo (Mano & Ciletti, 2014).

Ejemplo

Determinar las formas de onda de salida Q y del Flip-Flop de la Figura 86, para las entradas S, R y CLK de la Figura 87. Suponer que el Flip-Flop disparado por flanco positivo se encuentra, inicialmente, en estado RESET.

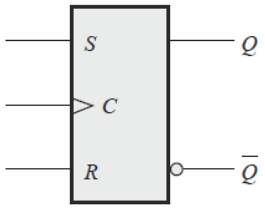


Figura 86 *Flip-Flop S-R disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

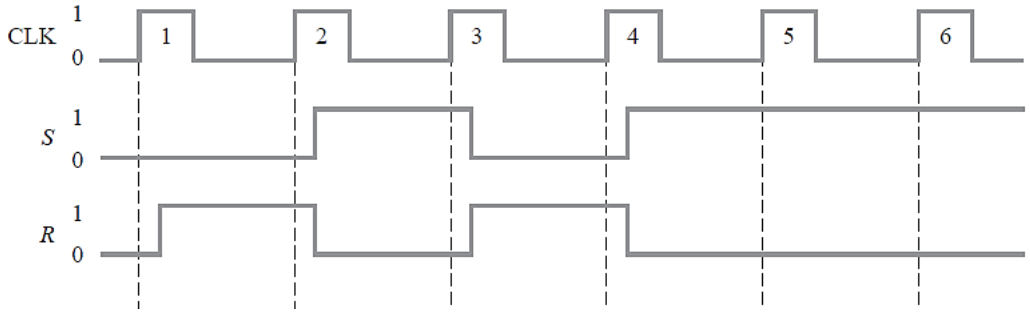


Figura 87 *Formas de onda de entrada aplicadas al Flip-Flop S-R disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

Solución

- Durante el impulso 1 de reloj, S está a nivel BAJO y R está a nivel BAJO; por lo tanto, Q no cambia (condición de retención).
- Durante el impulso 2 de reloj, S está a nivel BAJO y R está a nivel ALTO; por lo tanto, Q se mantiene a nivel BAJO (RESET).
- Durante el impulso 3 de reloj, S está a nivel ALTO y R está a nivel BAJO; por lo tanto, Q pasa a nivel ALTO (SET).
- Durante el impulso 4 de reloj, S está a nivel BAJO y R está a nivel ALTO; por lo tanto, Q pasa a nivel BAJO (RESET).

- Durante el impulso 5 de reloj, S está a nivel ALTO y R está a nivel BAJO; por lo tanto, Q pasa a nivel ALTO (SET).

- Durante el impulso 6 de reloj, S está a nivel ALTO y R está a nivel BAJO; por lo tanto, Q permanece a nivel ALTO (se mantiene en SET).

Una vez determinada la señal Q, la salida complementaria $\bar{Q}$se obtiene de forma directa como su complemento lógico. Las formas de onda resultantes para Qy $\bar{Q}$se muestran en la Figura 88, en función de las señales de entrada aplicadas (Mano & Ciletti, 2014).

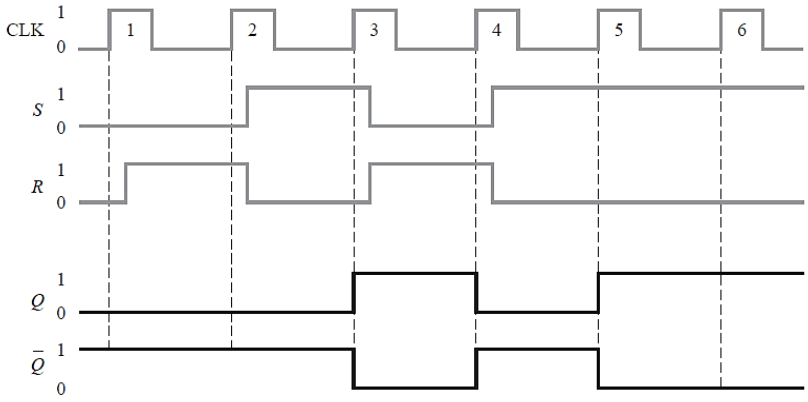


Figura 88 *Salidas Q y* $\bar{Q}$ *par el Flip-Flop S-R disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

Flip-Flop D disparado por flanco

El Flip-Flop D resulta muy útil cuando se necesita almacenar un único bit de datos (1 o 0). Si se añade un inversor a un Flip-Flop S-R obtenemos un flipflop D básico, como se muestra en la Figura 89, en la que se muestra uno disparado por flanco positivo (Mano & Ciletti, 2014).

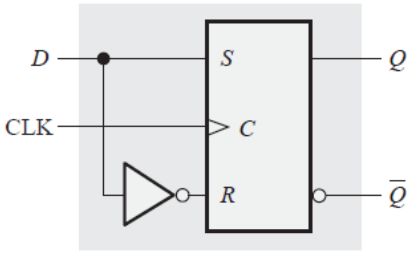


Figura 89 *Flip-Flop D disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

El flip-flop D mostrado en la Figura 90 dispone de una única entrada de datos (D), además de la entrada de reloj. Cuando, en el instante en que se produce un impulso de reloj, la entrada D se encuentra a nivel ALTO, el flip-flop pasa al estado SET y almacena un nivel lógico 1, capturando el valor presente en D durante el flanco activo del reloj. Por el contrario, si la entrada D está a nivel BAJO en el momento del disparo del reloj, el flip-flop entra en estado RESET y almacena un nivel lógico 0. En un flip-flop D disparado por flanco positivo, el almacenamiento del dato se realiza en el flanco de subida del impulso de reloj. En cambio, el funcionamiento de un flip-flop disparado por flanco negativo es análogo, con la diferencia de que el disparo ocurre en el flanco de bajada del impulso de reloj. En ambos casos, la salida Q adopta el valor presente en la entrada Dúnicamente en el instante del flanco activo del reloj, manteniéndose constante hasta el siguiente evento de disparo. El funcionamiento del flip-flop D disparado por flanco positivo se resume en la Figura 88, donde se observa que la salida Qsigue a la entrada Dúnicamente en cada flanco activo del reloj (Mano & Ciletti, 2014).

Entradas		Salidas		
D	***CLK***	***Q***	$\overline{Q}$	***Comentarios***
1	↑	1	0	SET (almacena un 1)
0	↑	0	1	RESET (almacena un 0)
↑ = transición del reloj de nivel BAJO a nivel ALTO				

Figura 90 *Tabla de verdad de un Flip-Flop D disparado por flanco positivo.*

Fuente: Mano & Ciletti (2014).

Ejemplo

Dadas las formas de onda de la Figura 91 para la entrada D y el reloj, determinar la onda de salida Q si el Flip-Flop parte del estado RESET.

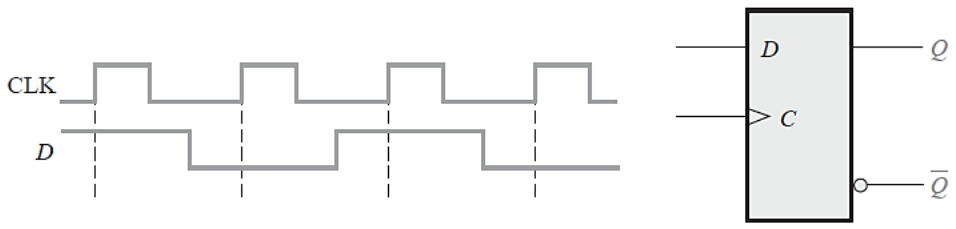


Figura 91 *Entrada D y reloj para el Flip-Flop D disparado por flanco positivo.*

Fuente: Mano & Ciletti (2014).

Solución

La salida Q adopta el estado presente en la entrada D cada vez que se produce un flanco positivo del reloj. La forma de onda resultante de la salida Q se muestra en la Figura 92.

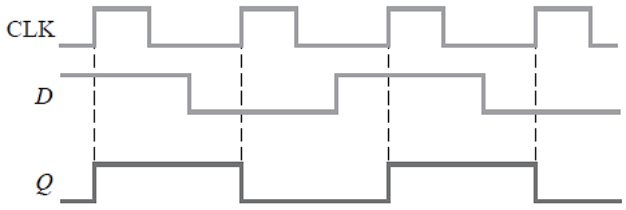
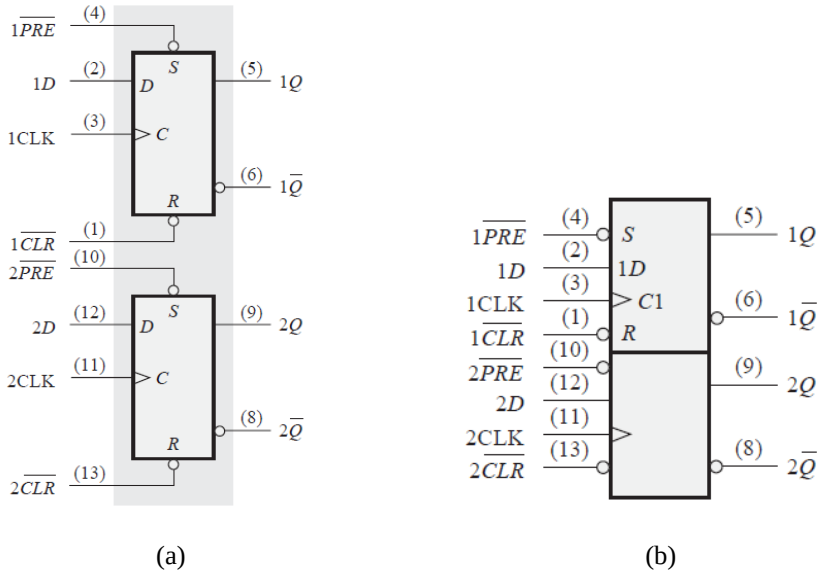


Figura 92 *Salida Q para el Flip-Flop D disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

Flip-Flop D 74AHC74

Este dispositivo CMOS contiene dos Flip-Flops D idénticos que son independientes entre sí, excepto en que comparten VCC y tierra. Son Flip-Flops disparados por flanco positivo y disponen de las entradas asíncronas de inicialización y borrado activas a nivel BAJO.

En la Figura 93 (a) se muestran los símbolos lógicos de cada Flip-Flop individual dentro del encapsulado, mientras que en la Figura 93 (b) podemos ver el bloque lógico, que representa el dispositivo completo. La numeración de los pines se indica entre paréntesis (Mano & Ciletti, 2014).



(a) (b)

Figura 93 *Doble Flip-Flop D por flanco positivo 74AHC74 (a) Símbolos lógicos individuales; (b) Bloque lógico.*
Fuente: Mano & Ciletti (2014).

Flip-Flop J-K disparado por flanco

El flip-flop J-K es un dispositivo versátil y uno de los tipos de flip-flop más ampliamente utilizados en los sistemas digitales. Su funcionamiento es idéntico al del flip-flop S-R en las condiciones de operación SET, RESET y retención de estado (sin cambio). La diferencia fundamental radica en que el flip-flop J-K no presenta condiciones no válidas, a diferencia del flip-flop S-R.

La Figura 94 muestra la lógica interna de un flip-flop J-K disparado por flanco positivo. Este dispositivo se diferencia del flip-flop S-R disparado por flanco en que la salida Qse realimenta a la entrada de la puerta G2, mientras que la salida complementaria $\bar{Q}$se realimenta a la entrada de la puerta G1, lo que permite eliminar las condiciones indeterminadas. Un flip-flop J-K puede implementarse también como un dispositivo disparado por flanco negativo; en este caso, la entrada de reloj se invierte, manteniéndose inalterado el principio de funcionamiento del circuito.

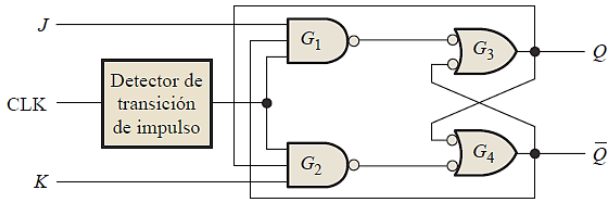


Figura 94 *Diagrama lógico simplificado de un Flip-Flop J-K disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

En la Figura 95 muestra la tabla de verdad del Flip-Flop J-K disparado por flanco, la cual resume su funcionamiento. No hay ningún estado no válido, como ocurría con el Flip-Flop S-R. La tabla de verdad de un dispositivo disparado por flanco negativo es idéntica, excepto en que se dispara durante el flanco de bajada del impulso de reloj (Mano & Ciletti, 2014).

Entradas			Salidas		
J	***K***	***CLK***	***Q***	$\bar{Q}$	*Comentarios*
0	0	↑	Q_0	$\bar{Q}_0$	No cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	Q_0	$\bar{Q}_0$	Basculación

↑ = transición del reloj de nivel BAJO a nivel ALTO
Q_0 = nivel de salida previo a la transición del reloj

Figura 95 *Tabla de verdad de un Flip-Flop J-K disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

Ejemplo

Las formas de onda de entrada mostradas en la Figura 96 se aplican a las entradas J, K y a la entrada de reloj, tal como se indica. Determinar la forma de onda de la salida Q, suponiendo que el Flip-Flop se encuentra inicialmente en estado RESET.

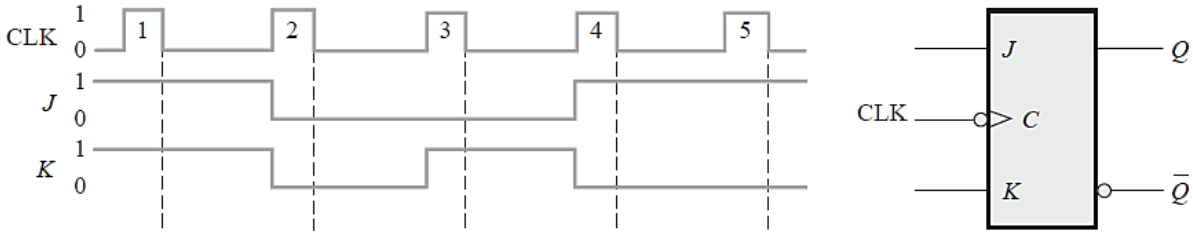


Figura 96 *Entrada J, K y reloj para el Flip-Flop J-K disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

Solución

- En primer lugar, dado que se trata de un Flip-Flop disparado por flanco negativo, como se indica mediante el círculo en la entrada de reloj, la salida Q sólo puede cambiar en el instante en que ocurre el flanco de bajada del impulso de reloj.

- En el primer impulso de reloj, las entradas J y K se encuentran a nivel ALTO. Debido a la condición de basculación (toggle), la salida Q cambia de estado y pasa a nivel ALTO.

- En el segundo impulso de reloj, se presenta la condición de no cambio en las entradas, por lo que la salida Q se mantiene a nivel ALTO.
- En el tercer impulso de reloj, la entrada J está a nivel BAJO y K a nivel ALTO, lo que produce una condición de RESET; en consecuencia, la salida Q pasa a nivel BAJO.

- En el cuarto impulso de reloj, J se encuentra a nivel ALTO y K a nivel BAJO, generando una condición de SET, por lo que la salida Q pasa nuevamente a nivel ALTO.
- Durante el quinto impulso de reloj, se mantiene la condición SET en las entradas J y K, de modo que la salida Q permanece a nivel ALTO.

La forma de onda resultante de la salida Qse muestra en la Figura 97.

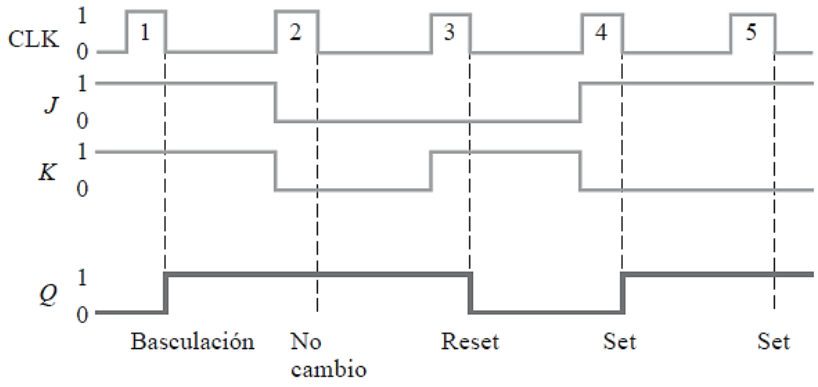


Figura 97 *Salida Q para el Flip-Flop J-K disparado por flanco positivo.*
Fuente: Mano & Ciletti (2014).

Flip-Flop J-K 74HC112

Este dispositivo CMOS contiene dos flip-flops idénticos, disparados por flanco negativo, y dispone además de entradas asíncronas de inicialización (SET) y borrado (RESET), ambas activas a nivel BAJO. Los símbolos lógicos correspondientes se presentan en la Figura 98 (Mano & Ciletti, 2014).

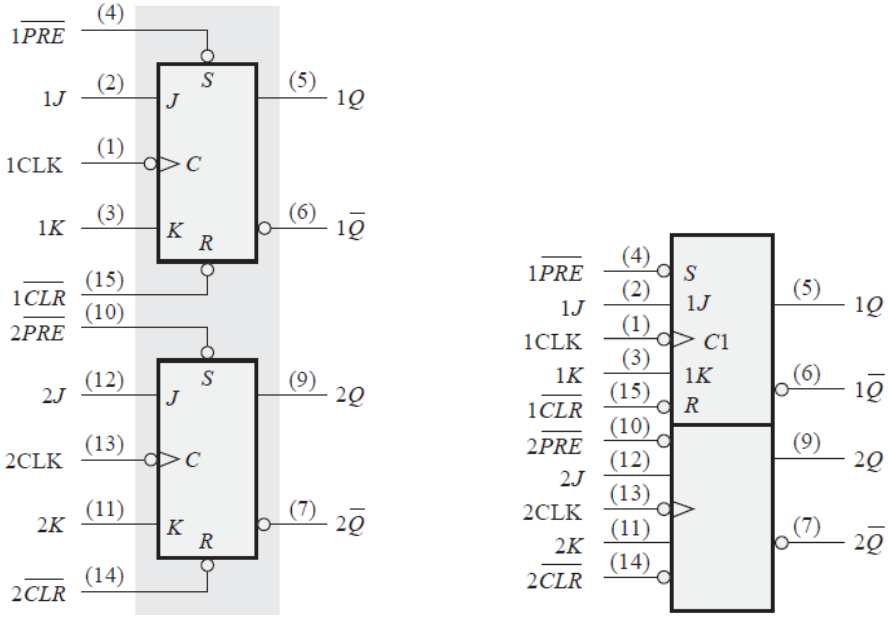


Figura 98 *Doble Flip-Flop J-K por flanco positivo 74HC112 (a) Símbolos lógicos individuales; (b) Bloque lógico.*
Fuente: Mano & Ciletti (2014).

4.4 Diagrama de estados

Un diagrama de estados representa la secuencia de estados por los que progresa un sistema o circuito cuando se aplica una señal de reloj. Cada estado describe una condición particular del sistema, y las transiciones entre estados están asociadas a los eventos, acciones o actividades que ocurren como resultado de la aplicación del reloj u otras señales de control (Torres, 2020).

En la Figura 99 se presenta un diagrama de estados correspondiente a un circuito específico que no posee entradas adicionales, aparte de la señal de reloj, y cuyas salidas están determinadas únicamente por los estados de los Flip-Flops que lo componen.

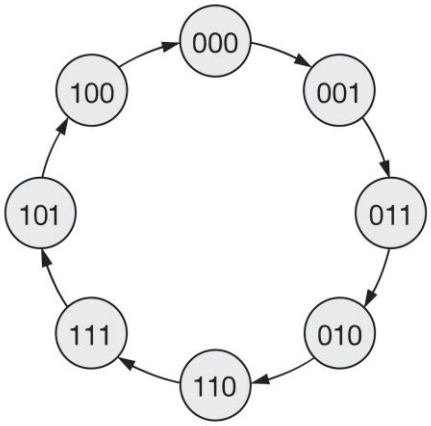


Figura 99 *Ejemplo de diagrama de estados de un circuito secuencial.*
Fuente: Hill & Peterson (2014).

Los diagramas de estados constituyen el primer paso en el procedimiento general de diseño de circuitos secuenciales (Hill & Peterson, 2014).

Una vez que el circuito secuencial ha sido definido mediante un diagrama de estados, el segundo paso consiste en obtener la tabla del estado siguiente. El estado siguiente corresponde al estado al que el sistema transita desde su estado actual cuando se aplica un impulso de reloj.

La tabla del estado siguiente se deriva directamente del diagrama de estados, tal como se ilustra en la Figura 100.

Estado actual			Estado siguiente		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

Figura 100 *Ejemplo de tabla del estado siguiente de un circuito secuencial.*

Fuente: Hill & Peterson (2014).

4.5 Máquinas secuenciales

El siguiente paso en el diseño de circuitos secuenciales consiste en elaborar una tabla de transiciones, basada en el funcionamiento del flip-flop utilizado. En esta tabla se enumeran todas las posibles transiciones de salida, mostrando cómo evoluciona la salida Qdel flip-flop al pasar del estado presente al estado siguiente cuando se aplica un impulso de reloj (Hill & Peterson, 2014).

En la Figura 101 se presenta la tabla de transiciones de un flip-flop J-K, obtenida a partir de la tabla del estado siguiente mostrada en la Figura 97. En dicha tabla, $Q_N$representa el estado presente del flip-flop (antes de la aplicación del impulso de reloj) y Q_{N+1}corresponde al estado siguiente (después del impulso de reloj). Para cada transición de salida se indican los valores de las entradas Jy Kque producen dicha transición. El símbolo X señala condiciones indiferentes, es decir, casos en los que la entrada puede tomar el valor lógico 0 o 1 sin afectar el resultado.

Transiciones de salida		Entradas del flip-flop	
Q_N	Q_{N+1}	J	K
0 →	0	0	X
0 →	1	1	X
1 →	0	X	1
1 →	1	X	0

Q_N: estado actual
Q_{N+1}: siguiente estado
X: condición "indiferente"

Figura 101 *Tabla de transiciones para un Flip-Flop J-K.*

Fuente: Hill & Peterson (2014).

Los mapas de Karnaugh se utilizan para determinar la lógica necesaria en las entradas *J* y *K* de cada Flip-Flop del circuito secuencial. A partir de dichos mapas se obtienen las expresiones lógicas simplificadas correspondientes a las entradas *J* y *K* de cada Flip-Flop. El paso final del diseño consiste en implementar la lógica combinacional derivada de estas expresiones y conectar los Flip-Flops de forma adecuada para obtener el circuito secuencial definitivo (Hill & Peterson, 2014).

A continuación, se resumen los pasos generales para el diseño de cualquier circuito secuencial:

1. Especificar la secuencia de funcionamiento y dibujar el diagrama de estados.
2. Obtener la tabla del estado siguiente a partir del diagrama de estados.
3. Desarrollar la tabla de transiciones, que muestra las entradas del Flip-Flop requeridas para cada transición de estado. Esta tabla es característica del tipo de Flip-Flop utilizado.
4. Transferir los valores de *J* y *K* desde la tabla de transiciones a los mapas de Karnaugh, utilizando un mapa independiente para cada entrada de cada Flip-Flop.
5. Obtener las expresiones lógicas a partir de los mapas de Karnaugh, formando los términos producto correspondientes para cada entrada del Flip-Flop.
6. Implementar las expresiones obtenidas mediante lógica combinacional y conectarlas a los Flip-Flops para construir el circuito secuencial final.

ACTIVIDADES

Actividad 1: Cuestionario sobre Señales de reloj y multivibradores.

1. ¿Qué es una señal de reloj?
2. ¿Qué es un multivibrador? ¿Cuáles son los multivibradores que existen?

Actividad 2: Cuestionario sobre Lógica secuencial sincronismo.

1. ¿Cuándo se denominan entradas de control síncronas?
2. ¿Cuándo se denominan entradas de control asíncronas?
3. ¿Cómo puede ser activada la entrada de sincronismo?

Actividad 3: Cuestionario sobre Celdas binarias Flip/Flops.

1. Realizar el símbolo lógico y tabla de verdad del Flip-Flop R-S disparado por flanco positivo.
2. Realizar el símbolo lógico y tabla de verdad del Flip-Flop D disparado por flanco positivo.
3. Realizar el símbolo lógico y tabla de verdad del Flip-Flop J-K disparado por flanco positivo.

Actividad 4: Cuestionario sobre Diagramas de estado y Máquinas secuenciales.

1. ¿Qué es un diagrama de estado?
2. ¿A qué denominamos máquina secuencial?

Actividad 5: Autoevaluación.

1. Determinar las formas de onda de salida Q de un Flip-Flop S-R, para las entradas S, R y CLK de la Figura 102. Suponer que el Flip-Flop disparado por flanco positivo se encuentra, inicialmente, en estado RESET.

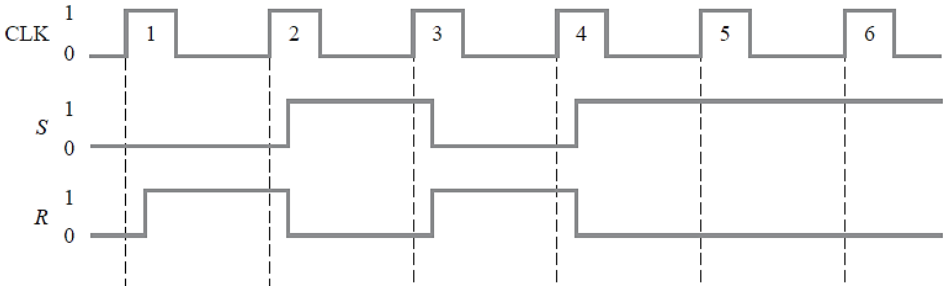


Figura 102 *Formas de onda de entrada aplicadas al Flip-Flop S-R disparado por flanco positivo.*
Fuente: Hill & Peterson (2014).

2. Dadas las formas de onda de la Figura 103 para la entrada D y el reloj, determinar la onda de salida Q si el Flip-Flop parte del estado RESET.

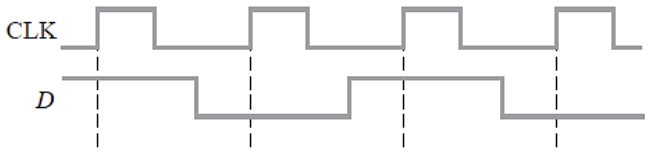


Figura 103 *Entrada D y reloj para el Flip-Flop D disparado por flanco positivo.*
Fuente: Hill & Peterson (2014).

3. Las formas de onda de entrada de la Figura 104 se aplican a las entradas J, K y de reloj, tal y como se muestra. Determinar la salida Q suponiendo que el Flip-Flop se encuentra inicialmente en estado RESET.

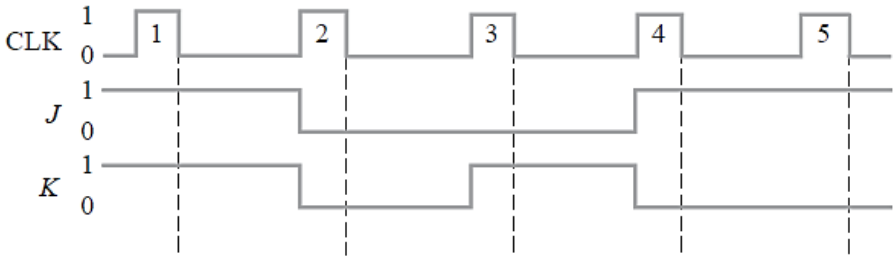


Figura 104 *Entrada J, K y reloj para el Flip-Flop J-K disparado por flanco positivo.*
Fuente: Hill & Peterson (2014).

BIBLIOGRAFÍA

Bibliografía básica

Floyd, T. L. (2006). *Fundamentos de sistemas digitales*. Prentice Hall.

Bibliografía complementaria

Carpinelli, J. D. (2023). *Introducción animada al diseño lógico digital*.

Hernández, F. & Labado, P. (2022). *Electrónica digital aplicada*.

Hill, F. & Peterson, G. (2014). *Sistemas digitales: diseño lógico*. Wiley.

Mano, M. & Ciletti, M. (2014). *Diseño digital*. Pearson.

Rojas, M. (2021). *Diseño de circuitos lógicos para ingeniería*.

Roth, C. H. & Kinney, L. (2014). *Fundamentos de diseño lógico*. Cengage Learning.

Tocci, R. J., Widmer, N. S., & Moss, G. (2011). *Sistemas digitales: Principios y aplicaciones*. Pearson.

Torres, J. (2020). *Electrónica digital: Diseño y análisis moderno*.


Red de Investigación
Científica y Desarrollo
Tecnológico Del Pacífico
101011
EDITORIAL
SAGA
ISBN: 978-9907-803-03-7
9 789907 803037