ELECTRÓNICA DIGITAL

EJEMPLOS Y EJERCICIOS

OSCAR IGNACIO BOTERO HENAO

Electrónica Digital

Ejemplos y Ejercicios

Oscar Ignacio Botero Henao

Fondo Editorial RED Descartes

Córdoba (España)
2025

Título de la obra:
Electrónica Digital
Ejemplos y Ejercicios

Autor:
Oscar Ignacio Botero Henao

Código JavaScript para el libro: Joel Espinosa Longi, IMATE, UNAM.
Recursos interactivos: DescartesJS, WebSim
Fuentes: Lato y UbuntuMono
Figura de portada: ilustración generada por Nightcafe AI

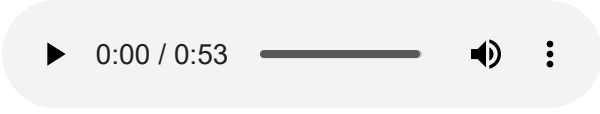
Red Educativa Digital Descartes
Córdoba (España)
descartes@proyectodescartes.org
https://proyectodescartes.org

Proyecto iCartesiLibri
https://proyectodescartes.org/iCartesiLibri/index.htm

ISBN: 978-84-10368-22-4

Tabla de contenido

Prefacio



La electrónica digital es una disciplina fundamental en el campo de la ingeniería y la tecnología, abarcando desde los principios básicos de los sistemas numéricos hasta el diseño y la implementación de circuitos complejos con mediana escala de integración. Esta **primera parte** del libro ***"ELECTRÓNICA DIGITAL, Ejemplos y Ejercicios"*** ha sido concebido con el objetivo de proporcionar a estudiantes y profesionales una herramienta práctica y accesible para el estudio y la comprensión de los conceptos esenciales de la electrónica digital.

A diferencia de otros textos que se centran en la teoría y la explicación detallada de los principios y conceptos, esta obra se dedica exclusivamente a la resolución de Ejemplos y la proposición de Ejercicios con respuestas. Creo firmemente que la mejor manera de aprender y dominar los conceptos de la electrónica digital es a través de la práctica constante y la aplicación directa de los conocimientos adquiridos.

Los temas abordados que incluye este libro son:

- **Capítulo 1: Conversiones Sistemas Numéricos:**
  - Este tema abarca la transformación de números entre distintos sistemas de numeración. Cada sistema tiene su propia base: el binario (base 2), el octal (base 8), el decimal (base 10) y el hexadecimal (base 16). Estas conversiones son fundamentales para trabajar con datos en distintos niveles de abstracción en las áreas de la informática y la electrónica.

Capítulo 2: **Álgebra de Boole:**

- El álgebra Booleana es una rama de las matemáticas que se utiliza para el análisis y diseño de circuitos digitales. Emplea variables binarias y operaciones lógicas como AND, OR y NOT, permitiendo simplificar y optimizar circuitos lógicos con el fin de reducir costos en su implementación y mejorar su eficiencia.

Capítulo 3: **Operaciones Lógicas Binarias:**

- Este capítulo muestra con ejemplos sobre cómo realizar operaciones aritméticas básicas como la suma, la resta, la multiplicación y la división utilizando números binarios. Es esencial para entender el funcionamiento interno de los microprocesadores y otros componentes digitales.
- Se exploran los circuitos lógicos para llevar a cabo operaciones con números binarios como: sumadores, restadores, multiplicadores y divisores, que son bloques fundamentales en la construcción de unidades aritmético-lógicas (ALU) en computadoras.

Capítulo 4: **Funciones Lógicas:**

- Este capítulo se enfoca en ejemplos que develan el procedimiento y solución de funciones lógicas utilizando los diferentes tipos de compuertas lógicas, sus expresiones Booleanas, tablas de verdad y la simbología propia de cada tipo.

Capítulo 5: **Minterm y Maxterm, Lógica AOI:**

- La lógica AOI (AND-OR-NOT) es una técnica utilizada para implementar funciones lógicas. Los minterms y maxterms son representaciones canónicas de funciones Booleanas que se emplean para estructurar el diseño de circuitos

lógicos con base en solo compuertas lógicas AND, NOT y OR y que simplifican el diseño e implementación.

 **Capítulo 6: Mapas de Karnaugh:**

- Los mapas de Karnaugh son una herramienta gráfica utilizada para simplificar funciones lógicas de manera visual, inventada por el físico y matemático estadounidense Maurice Karnaugh. Facilitan la minimización de expresiones Booleanas, ayudando a diseñar circuitos más sencillos y eficaces.

Cada capítulo contiene una serie de **Videos explicativos, Objetos Interactivos, archivos con Simulaciones y Hojas de Datos de los circuitos integrados**, diseñados para complementar y enriquecer el aprendizaje. Estos recursos adicionales permiten al lector visualizar y experimentar con los conceptos en un entorno dinámico y práctico que enriquecen la experiencia de aprendizaje al proporcionar una perspectiva práctica y aplicable; donde, cada uno de estos temas es crucial para desarrollar una comprensión sólida de la Electrónica Digital. En la parte final del libro encuentra unas actividades de **Repaso** que sirven de autoexamen para comprobar los conocimientos y la comprensión de los temas trabajados; igualmente, varios **Anexos** como información complementaria.

Invito a los lectores a sumergirse en este viaje de exploración y descubrimiento, utilizando los ejemplos resueltos y los ejercicios propuestos como una guía para alcanzar una comprensión profunda y aplicada del mundo de la Electrónica Digital. Espero que este libro se convierta en un aliado indispensable en su formación y desarrollo profesional.

Oscar Ignacio Botero Henao
Autor

Capítulo I

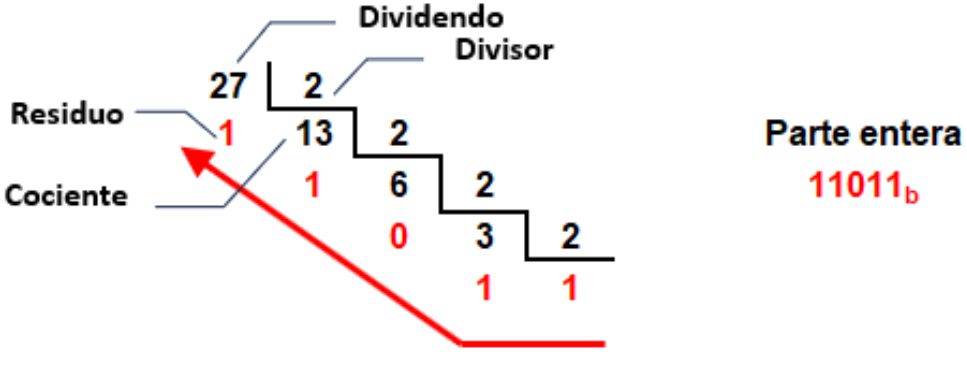
Conversiones: Sistemas Numéricos

1.1 Conversión de Decimal a Binario

Para la conversión de la parte entera, se va dividiendo la cantidad decimal por la base 2 (divisiones sucesivas), tantas veces como sea posible, hasta que llegar a un cociente de 1. Luego, se toma el último "Cociente" y todos los "Residuos" comenzando con el último (bit más significativo - MSB) hasta llegar al primero (bit menos significativo - LSB) como lo indica la flecha en la figura.[1][2][3][4][5][7]

Para la conversión de la parte fraccionaria se multiplica sucesivamente por la base 2, hasta la cantidad de dígitos fraccionarios que se requieran o hasta que el resultado del producto sea cero (0); para la respuesta se toman los valores enteros de cada producto y los valores que sobran se siguen multiplicando por 2.

★ Ejemplo 1: ¿Convertir $27{,}375_d$ a binario?

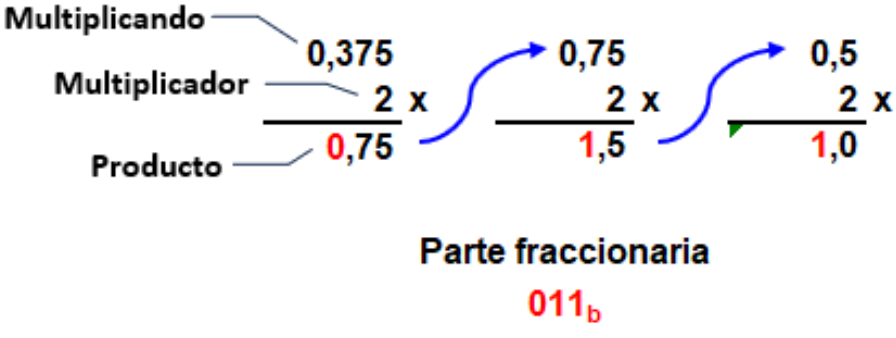


Otra forma de evidenciar la solución es:

27 ÷ 2 = 13 y residuo **1**
13 ÷ 2 = 6 y residuo **1**

6 ÷ 2 = 3 y residuo **0**
3 ÷ 2 = **1** y residuo **1**
Parte entera = 11011_b

Para la parte fraccionaria se utiliza la multiplicación:



Otra forma de evidenciar la solución es:

0,375 x 2 = **0** y residuo 75
0,75 x 2 = **1** y residuo 5
0,5 x 2 = **1** y residuo 0
Parte fraccionaria = 011_b
➤ **R/. $27{,}375_d = 11011{,}011_b$** ✓

★ **Ejemplo 2: ¿Convertir $103{,}85_{10}$ a binario?**

103 ÷ 2 = 51 y residuo **1**
51 ÷ 2 = 25 y residuo **1**
25 ÷ 2 = 12 y residuo **1**
12 ÷ 2 = 6 y residuo **0**
6 ÷ 2 = 3 y residuo **0**
3 ÷ 2 = **1** y residuo **1**
Parte entera = 1100111_b

0,85 x 2 = **1** y residuo 7
0,7 x 2 = **1** y residuo 4
0,4 x 2 = **0** y residuo 8
0,8 x 2 = **1** y residuo 6
0,6 x 2 = **1** y residuo 2
0,2 x 2 = **0** y residuo 4 y se repiten las operaciones a partir del residuo 4
Parte fraccionaria = 110110_b
➤ **R/. $103,85_d = 1100111,110110_b$** ✓

1.2 Conversión de Binario a Decimal

Para la conversión se utiliza el "método de la suma de pesos o ponderaciones", donde se toman los dígitos de la parte entera de derecha a izquierdo y se multiplica cada dígito por 2 elevado a la potencia correspondiente, comenzando con 2^0 e incrementando en uno (1) el valor de la potencia, a esto se le denomina **"ponderar cada digito"**; luego, se realiza la suma de todos los resultados parciales de la parte entera. Para la parte fraccionaria, se toman los dígitos de izquierda a derecha con exponentes negativos comenzando con 2^{-1} y se multiplica por cada dígito que está en esa posición; finalmente, se suman los resultados parciales de la parte fraccionaria. El resultado final es la unión de los dos resultados.

★ **Ejemplo 1: ¿Convertir $101,011_b$ a decimal?**

$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$
$1 \times 4 + 0 \times 2 + 1 \times 1 + 0 \times 1/2 + 1 \times 1/4 + 1 \times 1/8$
$4 + 1 + 1/4 + 1/8$
$4 + 1 + 0,25 + 0,125$
➤ **R/. $101,011_b = 5,375_d$** ✓

★ Ejemplo 2: ¿Convertir $1101{,}101_b$ a decimal?

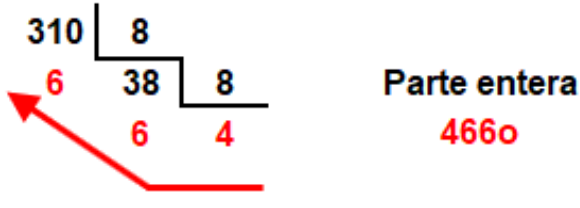
$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$
$1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 + 1 \times 1/2 + 0 \times 1/4 + 1 \times 1/8$
$8 + 4 + 1 + 1/2 + 1/8$
$8 + 4 + 1 + 0{,}5 + 0{,}125$
➤ R/. $1101{,}101_b = 13{,}625_d$ ✓

1.3 Conversión de Decimal a Octal

Para la conversión de la parte entera, se va dividiendo la cantidad decimal por la base 8 (divisiones sucesivas), tantas veces como sea posible, hasta que llegar a un cociente menor que 8.

Luego, se toma el último "Cociente" y todos los "Residuos" comenzando con el último (bit más significativo - MSB) hasta llegar al primero (bit menos significativo - LSB) como lo indica la flecha en la siguiente figura.

★ Ejemplo 1: ¿Convertir $310{,}625_d$ a octal?



Otra forma de evidenciar la solución es:

310 ÷ 8 = 38 y residuo **6**
38 ÷ 8 = **4** y residuo **6**
Parte entera = 466_o

Para la parte fraccionaria se utiliza la multiplicación:

0,625 x 8 = **5** y residuo 0
Parte fraccionaria = 5_o
➤ **R/. $310,625_d = 466,5_o$** ✓

★ **Ejemplo 2: ¿Convertir $415,75_d$ a octal?**

415 ÷ 8 = 51 y residuo **7**
51 ÷ 8 = **6** y residuo **3**
Parte entera = 637_o

0,75 x 8 = **6** y residuo 0
Parte fraccionaria = 6_o
➤ **R/. $415,75_d = 637,6_o$** ✓

1.4 Conversión de Octal a Decimal

Para la conversión se utiliza el "método de la suma de pesos o ponderaciones".

Para la conversión de Octal a Decimal, se toman los dígitos de la parte entera de derecha a izquierda y se multiplica cada dígito por 8 elevado a la potencia correspondiente, comenzando con 8^0 e incrementando en uno (1) el valor de la potencia, a esto se le denomina **"ponderar cada dígito"**; luego, se realiza la suma de todos los resultados parciales de la parte entera. Para la parte fraccionaria, se toman los dígitos de izquierda a derecha con exponentes negativos

comenzando con 8^{-1} y se multiplica por cada dígito que está en esa posición; finalmente, se suman los resultados parciales de la parte fraccionaria. El resultado final es la unión de los dos resultados.

★ Ejemplo 1: ¿Convertir $521{,}043_o$ a decimal?

$5 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 + 0 \times 8^{-1} + 4 \times 8^{-2} + 3 \times 8^{-3}$
$5 \times 64 + 2 \times 8 + 1 \times 1 + 0 \times 1/8 + 4 \times 1/64 + 3 \times 1/512$
$320 + 16 + 1 + 0{,}0625 + 0{,}005859375$
➤ R/. $521{,}043_o = 337{,}068359375_d$ ✓

★ Ejemplo 2: ¿Convertir $713{,}51_o$ a decimal?

$7 \times 8^2 + 1 \times 8^1 + 3 \times 8^0 + 5 \times 8^{-1} + 1 \times 8^{-2}$
$7 \times 64 + 1 \times 8 + 3 \times 1 + 5 \times 1/8 + 1 \times 1/64$
$448 + 8 + 3 + 0{,}625 + 0{,}015625$
➤ R/. $713{,}51_o = 459{,}640625_d$ ✓

1.5 Conversión de Decimal a Hexadecimal

Para la conversión de la parte entera, se va dividiendo la cantidad decimal por la base 16 (divisiones sucesivas), tantas veces como sea posible, hasta que llegar a un cociente menor que 16. Luego, se toma el último "Cociente" y todos los "Residuos" comenzando con el último (bit más significativo - MSB) hasta llegar al primero (bit menos significativo - LSB) como lo indica la flecha en la siguiente figura.

★ Ejemplo 1: ¿Convertir $158{,}25_d$ a hexadecimal?

Otra forma de evidenciar la solución es:

158 ÷ 16 = 9 y residuo 14 --> E
Parte entera = $9E_h$

0,25 x 16 = 4 y residuo 0
Parte fraccionaria = 4_h
➤ R/. $158,25_d = 9E,4_h$ ✓

★ Ejemplo 2: ¿Convertir $92,15_d$ a hexadecimal?

92 ÷ 16 = 5 y residuo 12 --> C
Parte entera = $5C_h$

0,15 x 16 = 2 y residuo 4
0,4 x 16 = 6 y residuo 4
0,4 x 16 = 6 y residuo 4
Parte fraccionaria = 266_h
➤ R/. $92,15_d = 5C,266_h$ ✓

1.6 Conversión de Hexadecimal a Decimal

Para la conversión se utiliza el "método de la suma de pesos o ponderaciones".

Para la conversión de Hexadecimal a Decimal, se toman los dígitos de la parte entera de derecha a izquierda y se multiplica cada dígito por 16 elevado a la potencia correspondiente, comenzando con 16^0 e incrementando en uno (1) el valor de la potencia, a esto se le denomina **"ponderar cada dígito"**; luego, se realiza la suma de todos los resultados parciales de la parte entera. Para la parte fraccionaria, se toman los dígitos de izquierda a derecha con exponentes negativos comenzando con 16^{-1} y se multiplica por cada dígito que está en esa posición; finalmente, se suman los resultados parciales de la parte fraccionaria. El resultado final es la unión de los dos resultados parciales.

★ Ejemplo 1: ¿Convertir $3AF,D_h$ a decimal?

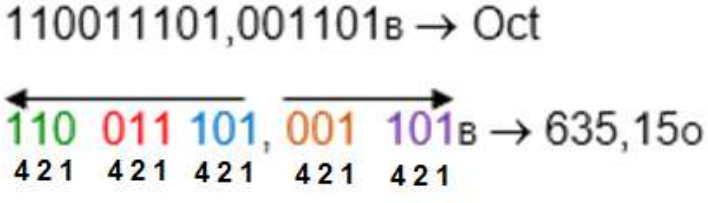
$3 \times 16^2 + A \times 16^1 + F \times 16^0 + D \times 16^{-1}$
$3 \times 256 + 10 \times 16 + 15 \times 1 + 13 \times 1/16$
$768 + 160 + 15 + 0{,}8125$
➤ R/. $3AF,D_h = 943{,}8125_d$ ✓

★ Ejemplo 2: ¿Convertir $ABC,37_h$ a decimal?

$A \times 16^2 + B \times 16^1 + C \times 16^0 + 3 \times 16^{-1} + 7 \times 16^{-2}$
$10 \times 256 + 11 \times 16 + 12 \times 1 + 3 \times 1/16 + 7 \times 1/256$
$2560 + 176 + 12 + 0{,}1875 + 0{,}02734375$
➤ R/. $ABC,37_h = 2748{,}21484375_d$ ✓

1.7 Conversión de Binario a Octal

Este método consiste en agrupar de a tres (3) bits. Para la parte entera se comienza de derecha a izquierda, o sea del bit menos significativo (LSB) hasta el más significativo (MSB), para la parte fraccionaria se agrupan en sentido contrario (de izquierda a derecha). Cuando las cifras no están completas, o sea de a tres (3) bits se completan con ceros (0) a la izquierda para la parte entera y con ceros a la derecha para la parte fraccionaria.



★ Ejemplo 1: ¿Convertir $1001011011,1010_b$ a octal?

Parte entera: 001 001 011 011$_b$, agrupados de a 3 bits = 1 1 3 3$_o$
Parte fraccionaria: 101 000$_b$, agrupados de a 3 bits = 5 0$_o$

➤ R/. $1001011011,1010_b = 1133,50_o$ ✓

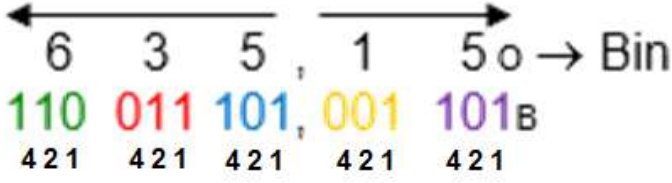
★ Ejemplo 2: ¿Convertir $111010011,101011_b$ a octal?

Parte entera: 111 010 011$_b$, agrupados de a 3 bits = 7 2 3$_o$
Parte fraccionaria: 101 011$_b$, agrupados de a 3 bits = 5 3$_o$

➤ R/. $111010011,101011_b = 723,53_o$ ✓

1.8 Conversión de Octal a Binario

Cada carácter octal se representa en 3 bits, tanto para la parte entera como para la parte fraccionaria.



★ Ejemplo 1: ¿Convertir $37,61_o$ a binario?

Parte entera: 37_o, representados en 3 bits cada carácter = 011 111_b
Parte fraccionaria: 61_o, representados en 3 bits cada carácter = 110 001_b
➤ **R/. $37,61_o = 011111,110001_b$** ✓

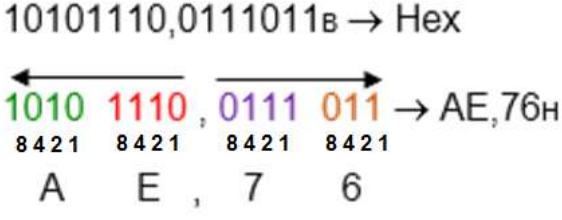
★ Ejemplo 2: ¿Convertir $152,731_o$ a binario?

Parte entera: 152_o, representados en 3 bits cada carácter = 001 101 010_b
Parte fraccionaria: 731_o, representados en 3 bits cada carácter = 111 011 001_b
➤ **R/. $152,731_o = 1101010,111011001_b$** ✓

1.9 Conversión de Binario a Hexadecimal

Este método es similar a la conversión de Binario a Octal, pero agrupando de a cuatro (4) bits.

Para la parte entera se comienza de derecha a izquierda, o sea del bit menos significativo (LSB) hasta el más significativo (MSB), para la parte fraccionaria se procede a agrupar en sentido contrario, o sea de izquierda a derecha. Cuando las cifras no están completas, o sea de a cuatro (4) bits se completan con ceros (0) a la izquierda para la parte entera y con ceros a la derecha para la parte fraccionaria.



★ Ejemplo 1: ¿Convertir $1101101111,101011_b$ a hexadecimal?

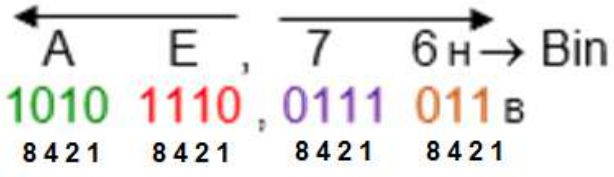
Parte entera: $11\ 0110\ 1111_b$, agrupados en 4 bits = $36F_h$
Parte fraccionaria: $1010\ 11_b$, agrupados en 4 bits = AC_h
➤ R/. $1101101111,101011_b = 36F,AC_h$ ✓

★ Ejemplo 2: ¿Convertir $111101011,1000101_b$ a hexadecimal?

Parte entera: $1\ 1110\ 1011_b$, agrupados en 4 bits = $1EB_h$
Parte fraccionaria: $1000\ 101_b$, agrupados en 4 bits = $8A_h$
➤ R/. $111101011,1000101_b = 1EB,8A_h$ ✓

1.10 Conversión de Hexadecimal a Binario

Cada carácter hexadecimal se representa en 4 bits, tanto para la parte entera como la fraccionaria; similar a la conversión de octal a binario.



★ **Ejemplo 1: ¿Convertir $A46,1F_h$ a binario?**

Parte entera: $A46_h$, representada en 4 bits = $1010\ 0100\ 0110_b$
Parte fraccionaria: $1F_h$, representada en 4 bits = $0001\ 1111_b$
➤ **R/. $A46,1F_b = 1010\ 0100\ 0110,0001\ 1111_b$** ✓

★ **Ejemplo 2: ¿Convertir $356,01_h$ a binario?**

Parte entera: 356_h, representada en 4 bits = $11\ 0101\ 0110_b$
Parte fraccionaria: 01_h, representada en 4 bits = $0000\ 0001_b$
➤ **R/. $356,01_h = 11\ 0101\ 0110,0000\ 0001_b$** ✓

1.11 Conversión de Octal a Hexadecimal

Para la conversión se realizan dos pasos, primero se convierte de Octal a Decimal y luego de Decimal a Hexadecimal.

★ **Ejemplo 1: ¿Convertir $147,25_o$ a hexadecimal?**

$1 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 + 2 \times 8^{-1} + 5 \times 8^{-2}$
$1 \times 64 + 4 \times 8 + 7 \times 1 + 2 \times 1/8 + 5 \times 1/64$

$64 + 32 + 7 + 0,25 + 0,078125$
R/. $147,25_o = 103,328125_d$

Ahora, se convierte de Decimal a Hexadecimal:

103 ÷ 16 = **6** y residuo **7**
Parte entera = 67_h

0,328125 x 16 = **5** y residuo 25
0,25 x 16 = **4** y residuo 0
Parte fraccionaria = $0,54_h$
R/. $103,328125_d = 60,54_h$
➤ **R/. $147,25_o = 60,54_h$** ✓

★ **Ejemplo 2: ¿Convertir $100,75_o$ a hexadecimal?**

$1 \times 8^2 + 0 \times 8^1 + 0 \times 8^0 + 7 \times 8^{-1} + 5 \times 8^{-2}$
$1 \times 64 + 0 \times 8 + 0 \times 1 + 7 \times 1/8 + 5 \times 1/64$

64 + 0,875 + 0,078125
R/. $100,75_o = 64,953125_d$

Ahora, se convierte de Decimal a Hexadecimal:

64 ÷ 16 = **4** y residuo **0**
Parte entera = 40_h

0,953125 x 16 = **15 --> F** y residuo 25
0,25 x 16 = **4** y residuo 0
Parte fraccionaria = $0,F4_h$
R/. $64,953125_d = 40,F4_h$
➤ **R/. $100,75_o = 40,F4_h$** ✓

1.12 Conversión de Hexadecimal a Octal

Para la conversión se realizan dos pasos, primero se convierte de Hexadecimal a Decimal y luego de Decimal a Octal.

★ Ejemplo 1: ¿Convertir $1A7,5_h$ a octal?

$1 \times 16^2 + A \times 16^1 + 7 \times 16^0 + 5 \times 16^{-1}$
$1 \times 256 + 10 \times 16 + 7 \times 1 + 5 \times 1/16$
$256 + 160 + 7 + 0,3125$
R/. $1A7,5_h = 423,3125_d$

Ahora, se convierte de Decimal a Octal:

$423 \div 8 = 52$ y residuo **7**
$52 \div 8 =$ **6** y residuo **4**
Parte entera = 647_o

$0,3125 \times 8 =$ **2** y residuo 5
$0,5 \times 8 =$ **4** y residuo 0
Parte fraccionaria = 24_o
R/. $423,3125_d = 647,24_o$
➤ R/. $1A7,5_d = 647,24_o$ ✓

★ Ejemplo 2: ¿Convertir $61D,3_h$ a octal?

$6 \times 16^2 + 1 \times 16^1 + D \times 16^0 + 3 \times 16^{-1}$
$6 \times 256 + 1 \times 16 + 13 \times 1 + 3 \times 1/16$
$1536 + 16 + 13 + 0,1875$
R/. $61D,3_h = 1565,1875_d$

Ahora, se convierte de Decimal a Octal:

1565 ÷ 8 = 195 y residuo **5**
195 ÷ 8 = **24** y residuo **3**
24 ÷ 8 = **3** y residuo **0**
Parte entera = 3035_o

0,1875 x 8 = **1** y residuo 5
0,5 x 8 = **4** y residuo 0
Parte fraccionaria = 14_o
R/. $1565{,}1875_d = 3035{,}14_o$
➤ **R/. $61D{,}3_d = 3035{,}14_o$** ✓

1.13 Ejercicios propuestos con respuesta

1.13.1 Decimal a Binario

1. *¿Convertir $91{,}12_d$ a binario?* ***R/. $1011011{,}000111_b$***
2. *¿Convertir $146{,}56_d$ a binario?* ***R/. $10010010{,}10001111_b$***
3. *¿Convertir $82{,}55_d$ a binario?* ***R/. $1010010{,}1000110_b$***
4. *¿Convertir $200{,}62_d$ a binario?* ***R/. $11001000{,}10011110101_b$***
5. *¿Convertir $169{,}48_d$ a binario?* ***R/. $10101001{,}011110101_b$***

1.13.2 Binario a Decimal

1. *¿Convertir $11010011{,}01101_b$ a decimal?* ***R/. $211{,}40625_d$***
2. *¿Convertir $1000110010{,}101_b$ a decimal?* ***R/. $562{,}625_d$***
3. *¿Convertir $1001101110{,}1101_b$ a decimal?* ***R/. $622{,}8125_d$***

4. ¿Convertir $101010101,1111_b$ a decimal? **R/. $341,9375_d$**
5. ¿Convertir $11001111,111_b$ a decimal? **R/. $207,875_d$**

1.13.3 Decimal a Octal

1. ¿Convertir $84,15_d$ a octal? **R/. $124,114_o$**
2. ¿Convertir $125,3_d$ a octal? **R/. $175,231463_o$**
3. ¿Convertir $142,75_d$ a octal? **R/. $216,6_o$**
4. ¿Convertir $452,06_d$ a octal? **R/. $704,036560507_o$**
5. ¿Convertir $700,33_d$ a octal? **R/. $1274,250753412_o$**

1.13.4 Octal a Decimal

1. ¿Convertir $157,31_o$ a decimal? **R/. $111,390625_d$**
2. ¿Convertir $740,123_o$ a decimal? **R/. $480,1621094_d$**
3. ¿Convertir $333,333_o$ a decimal? **R/. $219,4277344_d$**
4. ¿Convertir $1005,462_o$ a decimal? **R/. $517,5976563_d$**
5. ¿Convertir $2100,504_o$ a decimal? **R/. $1088,632813_d$**

1.13.5 Decimal a Hexadecimal

1. ¿Convertir $101,8_d$ a hexadecimal? **R/. $65,CCC_h$**
2. ¿Convertir $197,8125_d$ a hexadecimal? **R/. $C5,D_h$**
3. ¿Convertir $222,46_d$ a hexadecimal? **R/. $DE,75C28F5C28_h$**

4. ¿Convertir $415{,}68_d$ a hexadecimal? **R/. $19F{,}AE147AE147_h$**
5. ¿Convertir $375{,}208_d$ a hexadecimal? **R/. $177{,}353F7CED91_h$**

1.13.6 Hexadecimal a Decimal

1. ¿Convertir $9B0{,}D1_h$ a decimal? **R/. $2480{,}816406_d$**
2. ¿Convertir $FEA{,}7C_h$ a decimal? **R/. $4074{,}484375_d$**
3. ¿Convertir $824{,}B1_h$ a decimal? **R/. $2084{,}691406_d$**
4. ¿Convertir $CE5{,}0F_h$ a decimal? **R/. $3301{,}058594_d$**
5. ¿Convertir $DEA{,}31_h$ a decimal? **R/. $3562{,}191406_d$**

1.13.7 Binario a Octal

1. ¿Convertir $1\ 010\ 111{,}111\ 01_b$ a octal? **R/. $127{,}72_o$**
2. ¿Convertir $101\ 0101\ 110{,}001\ 111\ 01_b$ a octal? **R/. $1256{,}172_o$**
3. ¿Convertir $10\ 000\ 110\ 110{,}110\ 000\ 11_b$ octal? **R/. $2066{,}606_o$**
4. ¿Convertir $11\ 111\ 110\ 011{,}101\ 010\ 10_b$ a octal? **R/. $3763{,}524_o$**
5. ¿Convertir $1\ 100\ 000\ 001{,}100\ 01_b$ a octal? **R/. $1401{,}42_o$**

1.13.8 Octal a Binario

1. ¿Convertir $100{,}707_o$ a binario?
   R/. $1\ 000\ 000{,}111\ 000\ 111_b$

2. *¿Convertir $2530,5541_o$ a binario?*
   R/. $10\ 101\ 011\ 000,101\ 101\ 100\ 001_b$
3. *¿Convertir $1069,2263_o$ binario?*
   R/. $1\ 000\ 110\ 011,010\ 010\ 110\ 011_b$
4. *¿Convertir $1313,3131_o$ a binario?*
   R/. $1\ 011\ 001\ 011,011\ 001\ 011\ 001_b$
5. *¿Convertir $5714,265_o$ a binario?*
   R/. $101\ 111\ 001\ 100,010\ 110\ 101_b$

1.13.9 Binario a Hexadecimal

1. *¿Convertir $1001000111,1100101_b$ a hexadecimal?*
   R/. $247,CA_h$
2. *¿Convertir $111100000011001,0011100011_b$ a hexadecimal?*
   R/. $3C19,38C_h$
3. *¿Convertir $100000011111,0001011110_b$ a hexadecimal?*
   R/. $81F,178_h$
4. *¿Convertir $100111000001,010011001101_b$ a hexadecimal?*
   R/. $9C1,4CD_h$
5. *¿Convertir $111111111101,00000111011_b$ a hexadecimal?*
   R/. $1FFD,076_h$

1.13.10 Hexadecimal a Binario

1. *¿Convertir $664,E23_h$ a binario?*
   R/. $0110\ 0110\ 0100,1110\ 0010\ 0011_b$

2. *¿Convertir $126,D73_h$ a binario?*
   R/. $0001\ 0010\ 0110,\ 1101\ 0111\ 0011_b$
3. *¿Convertir $430,CD1_h$ a binario?*
   R/. $0100\ 0011\ 0000,1100\ 1101\ 0001_b$
4. *¿Convertir $987,654_h$ a binario?*
   R/. $1001\ 1000\ 0111,0110\ 0101\ 0100_b$
5. *¿Convertir $61F,308_h$ a binario?*
   R/. $0110\ 0001\ 1111,0011\ 0000\ 1000_b$

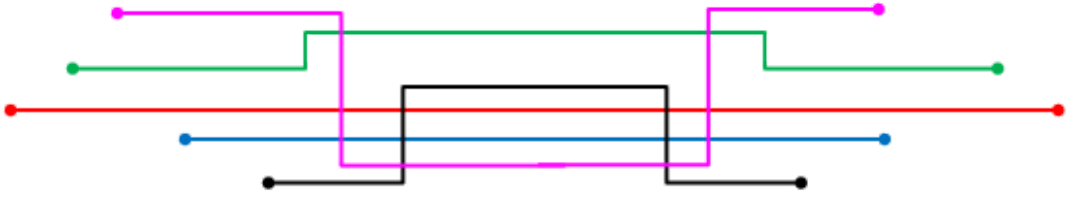
1.13.11 Octal a Hexadecimal

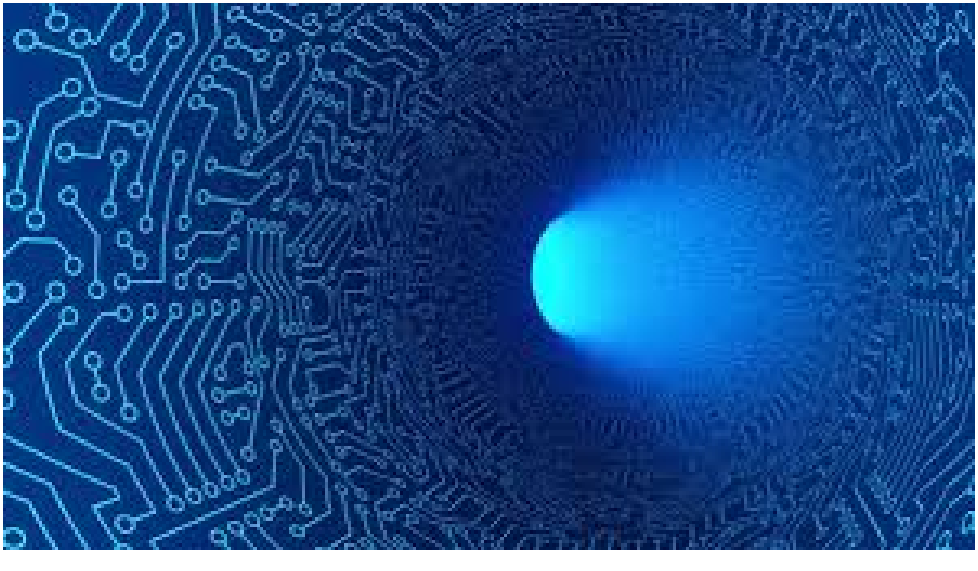
1. *¿Convertir $465,25_o$ a hexadecimal?*
   R/. $135,54_h$
2. *¿Convertir $71,32_o$ a hexadecimal?*
   R/. $39,68_h$
3. *¿Convertir $2407,53_o$ a hexadecimal?*
   R/. $507,AC_h$
4. *¿Convertir $3562,17_o$ a hexadecimal?*
   R/. $772,3C_h$
5. *¿Convertir $5371,06_o$ a hexadecimal?*
   R/. $AF9,18_h$

1.13.12 Hexadecimal a Octal

1. *¿Convertir $381,0A_h$ a octal?*
   R/. $1601,024_o$

2. *¿Convertir $6E9,B5_h$ a octal?*
   R/. $3351,552_o$
3. *¿Convertir $41B,2_h$ a octal?*
   R/. $2033,1_o$
4. *¿Convertir $FEA,03_h$ a octal?*
   R/. $7752,006_o$
5. *¿Convertir $58,9_h$ a octal?*
   R/. $130,44_o$





AYUDAS

COMPLEMENTOS

1.14 Videos

Vídeos sobre conversiones del sistema Decimal:[1]

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de DECIMAL a BINARIO con PARTE FRACCIONARIA - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de DECIMAL a OCTAL con PARTE FRACCIONARIA - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de DECIMAL a HEXADECIMAL con PARTE FRACCIONARIA - Ejercicio #1"***, licencia de atribución de Creative Commons.

[1] Videos realizados por Marisol Maldonado Olmos *(Pasos por Ingeniería)*, CONVERSIÓN entre sistemas numéricos...

Vídeos sobre conversiones del sistema Binario:[2]

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de BINARIO a OCTAL - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de BINARIO a DECIMAL con PARTE FRACCIONARIA - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos *(Pasos por Ingeniería)*, titulado ***"CONVERTIR de BINARIO a HEXADECIMAL - Ejercicio #1"***, licencia de atribución de Creative Commons.

[2] Videos realizados por Marisol Maldonado Olmos *(Pasos por Ingeniería)*, CONVERSIÓN entre sistemas numéricos...

Vídeos sobre conversiones del sistema Hexadecimal:[3]

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de HEXADECIMAL a BINARIO - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de HEXADECIMAL a OCTAL - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de HEXADECIMAL a DECIMAL con PARTE FRACCIONARIA - Ejercicio #1"***, licencia de atribución de Creative Commons.

[3] Videos realizados por Marisol Maldonado Olmos *(Pasos por Ingeniería)*, CONVERSIÓN entre sistemas numéricos...

Vídeos sobre conversiones del sistema Octal:[4]

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de OCTAL a BINARIO - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de OCTAL a DECIMAL con PARTE FRACCIONARIA - Ejercicio #1"***, licencia de atribución de Creative Commons.

Video realizado por Marisol Maldonado Olmos (Pasos por Ingeniería), titulado ***"CONVERTIR de OCTAL a HEXADECIMAL - Ejercicio #1"***, licencia de atribución de Creative Commons.

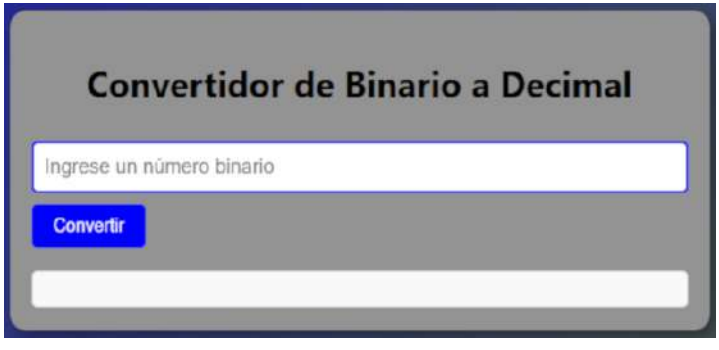
[4] Videos realizados por Marisol Maldonado Olmos *(Pasos por Ingeniería)*, CONVERSIÓN entre sistemas numéricos...

1.15 Objetos Interactivos

Haga clic sobre la imagen para abrir el objeto interactivo:

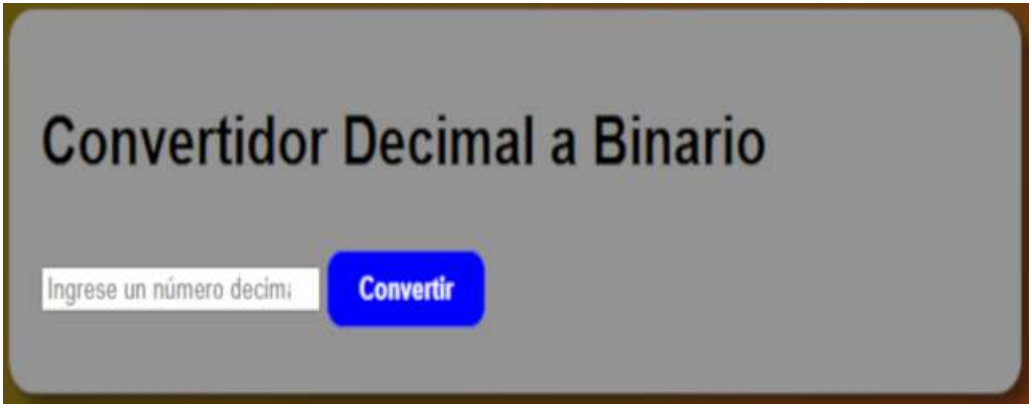
> Estos objetos interactivos fueron generados con la *asistencia* de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/

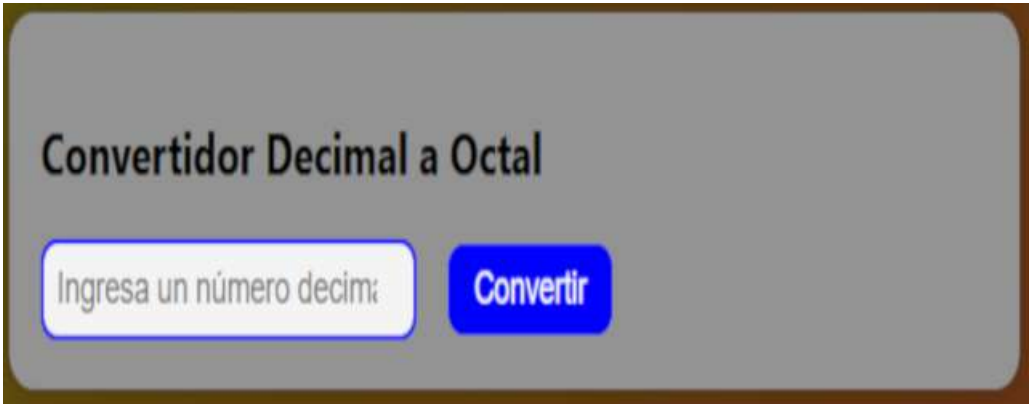
Convertidores partiendo del sistema Binario:

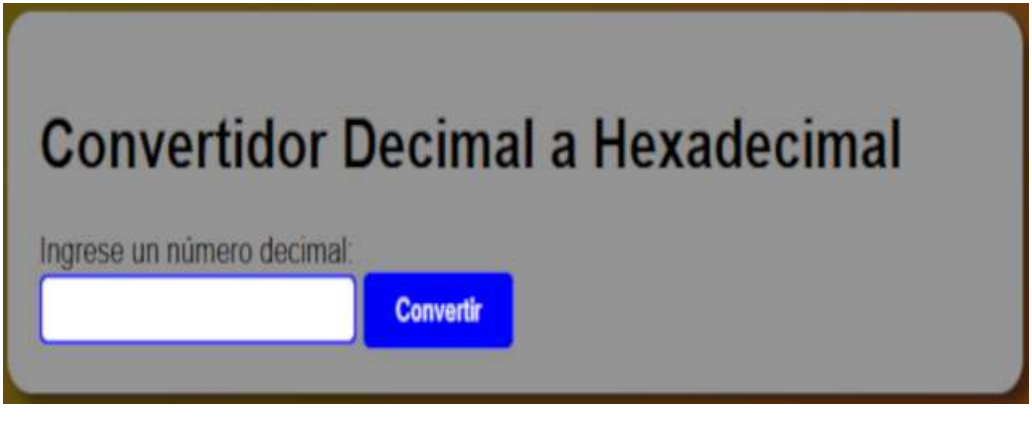


Haga clic sobre la imagen para abrir el objeto interactivo:

Convertidores partiendo del sistema Decimal:

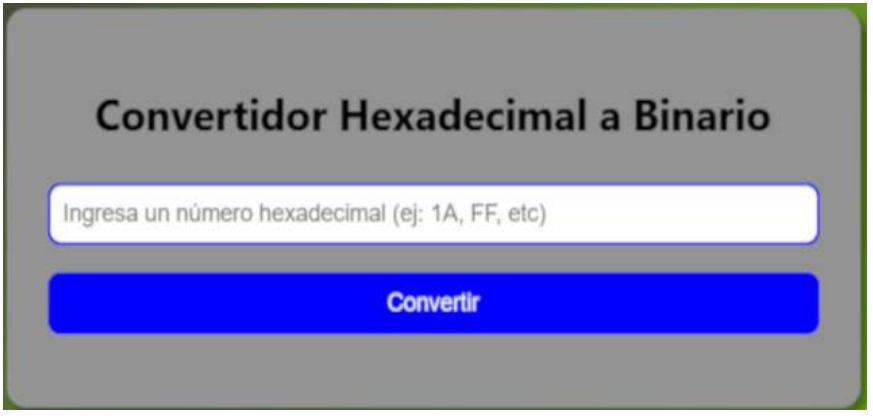


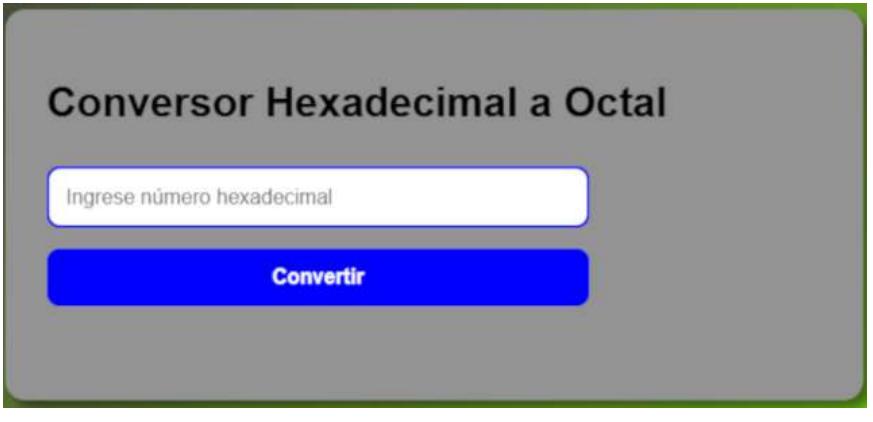


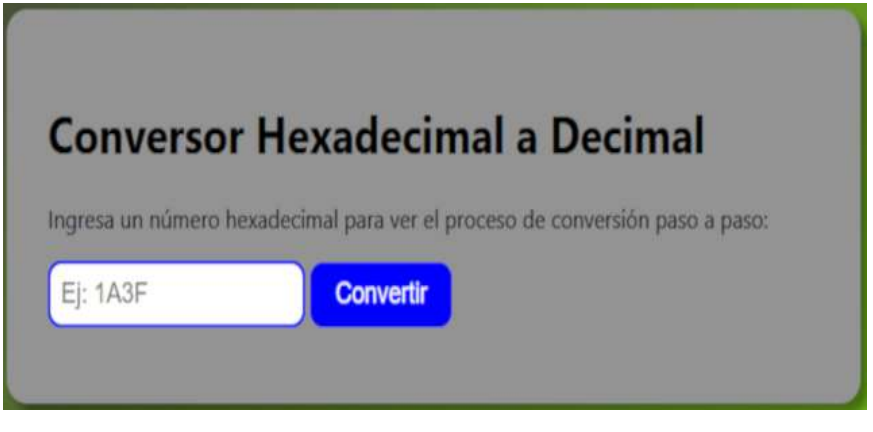


Haga clic sobre la imagen para abrir el objeto interactivo:

Convertidores partiendo del sistema Hexadecimal:

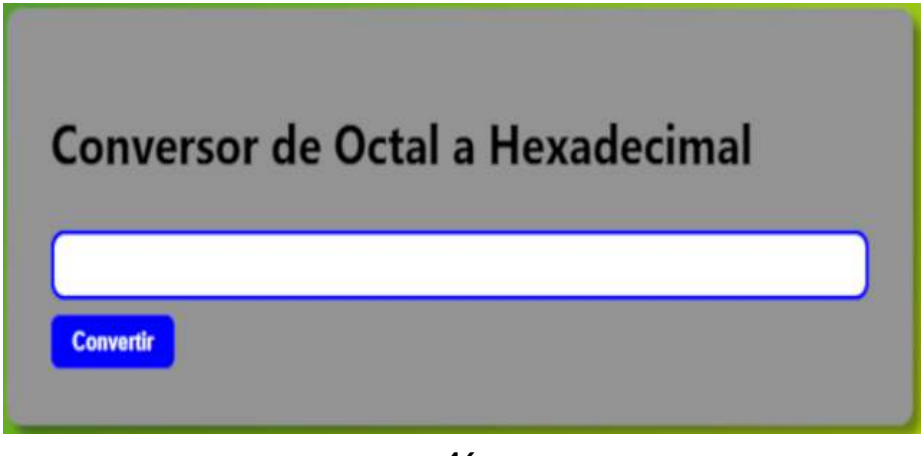






Haga clic sobre la imagen para abrir el objeto interactivo:

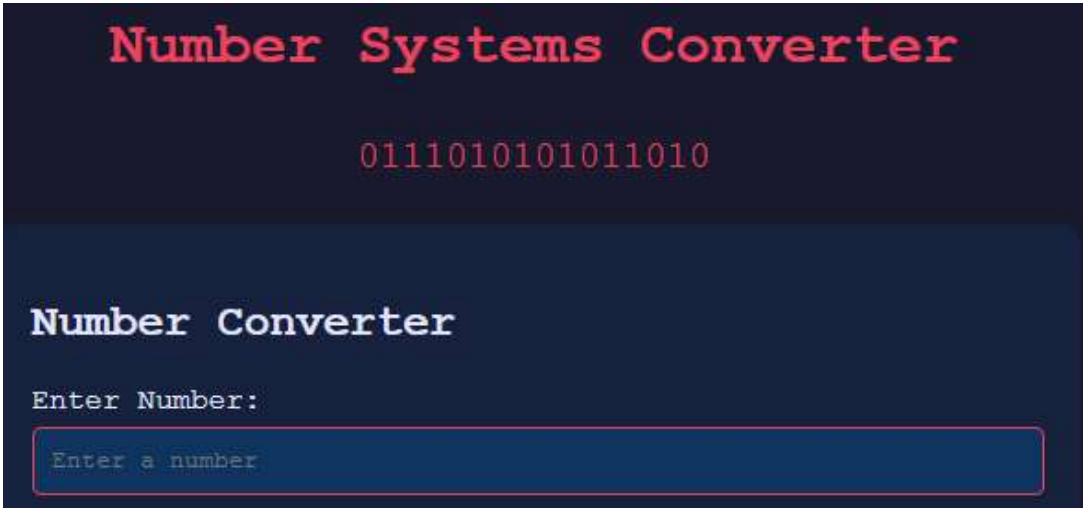
Convertidores partiendo del sistema Octal:



Haga clic sobre la imagen para abrir el objeto interactivo:

Convertidor entre los sistemas numéricos Binario, Decimal y Hexadecimal

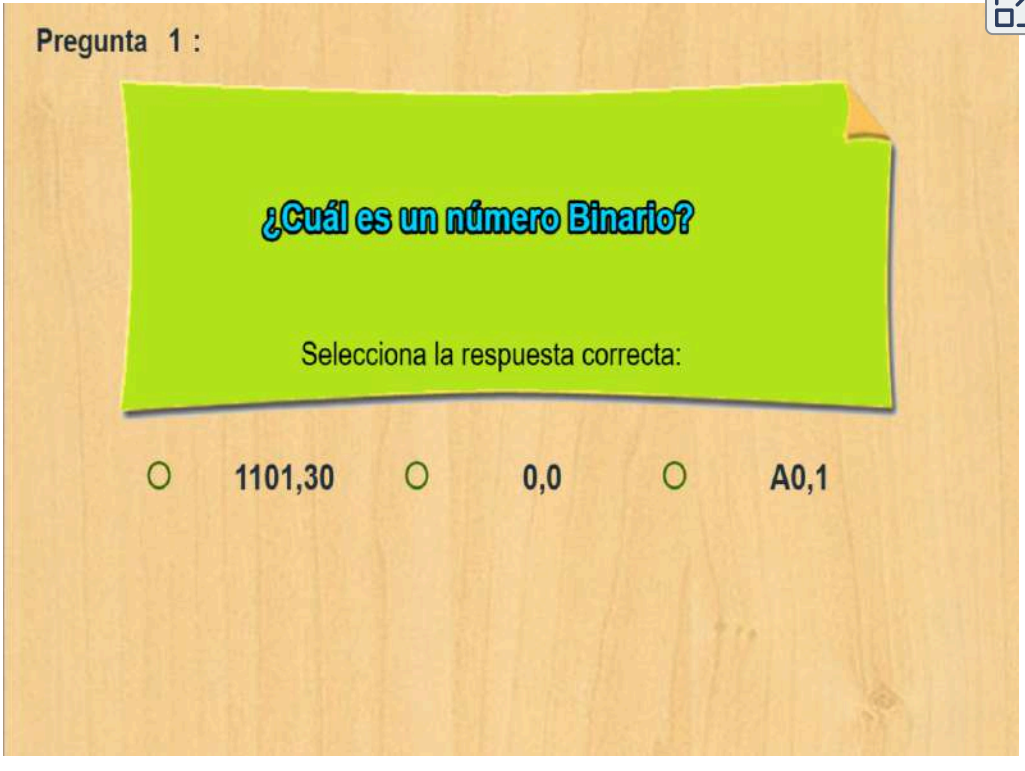
Objeto Interactivo para la conversión de valores enteros entre los sistemas numéricos Binario, Decimal y Hexadecimal.[5]



[5] Objeto Interactivo generado con la *asistencia* de la Inteligencia Artificial (IA) por "Websim.ai", Convertidor de Sistemas Numéricos

Seleccione la opción correcta

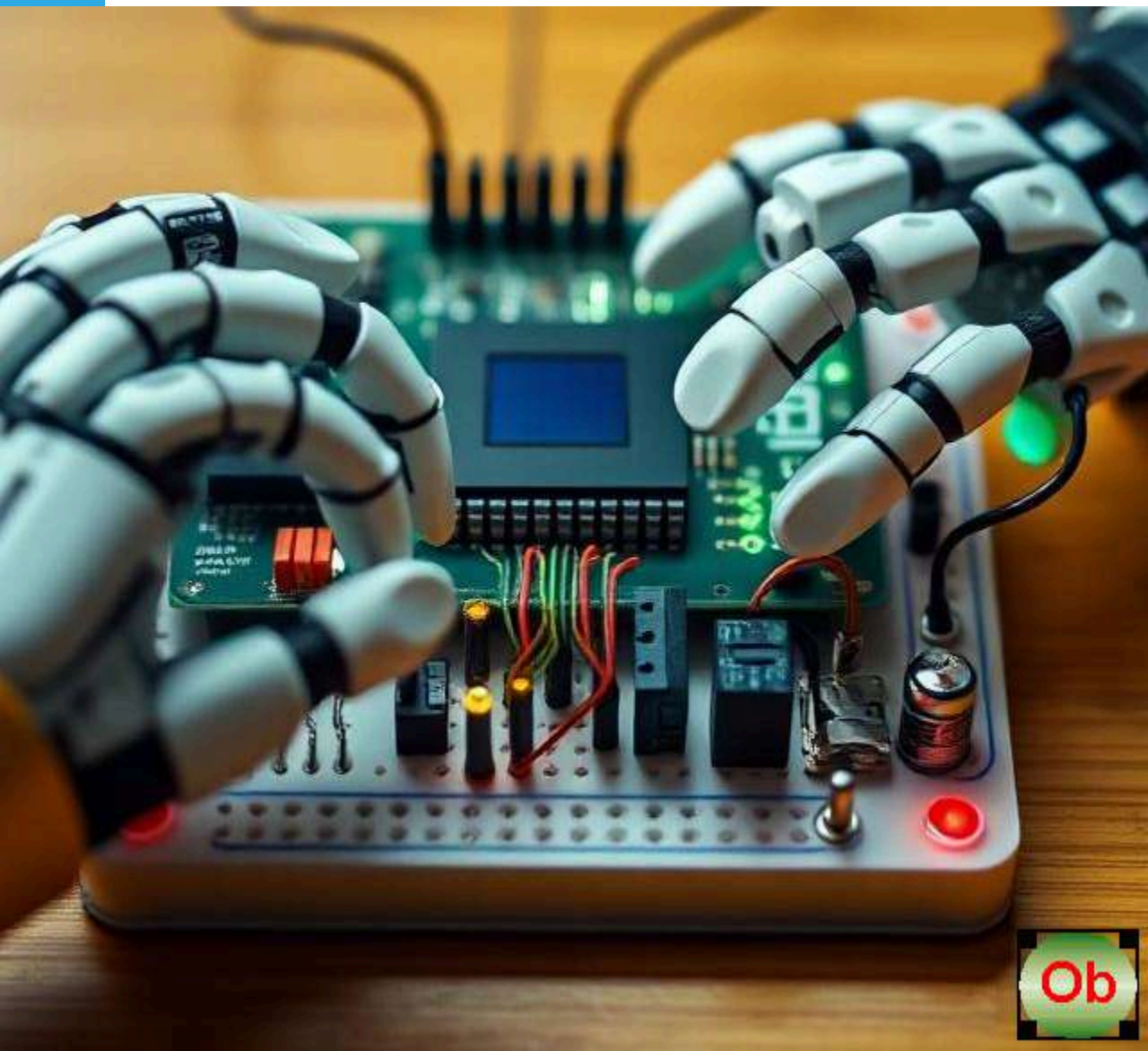
Objeto Interactivo de selección múltiple con una única respuesta: [6]

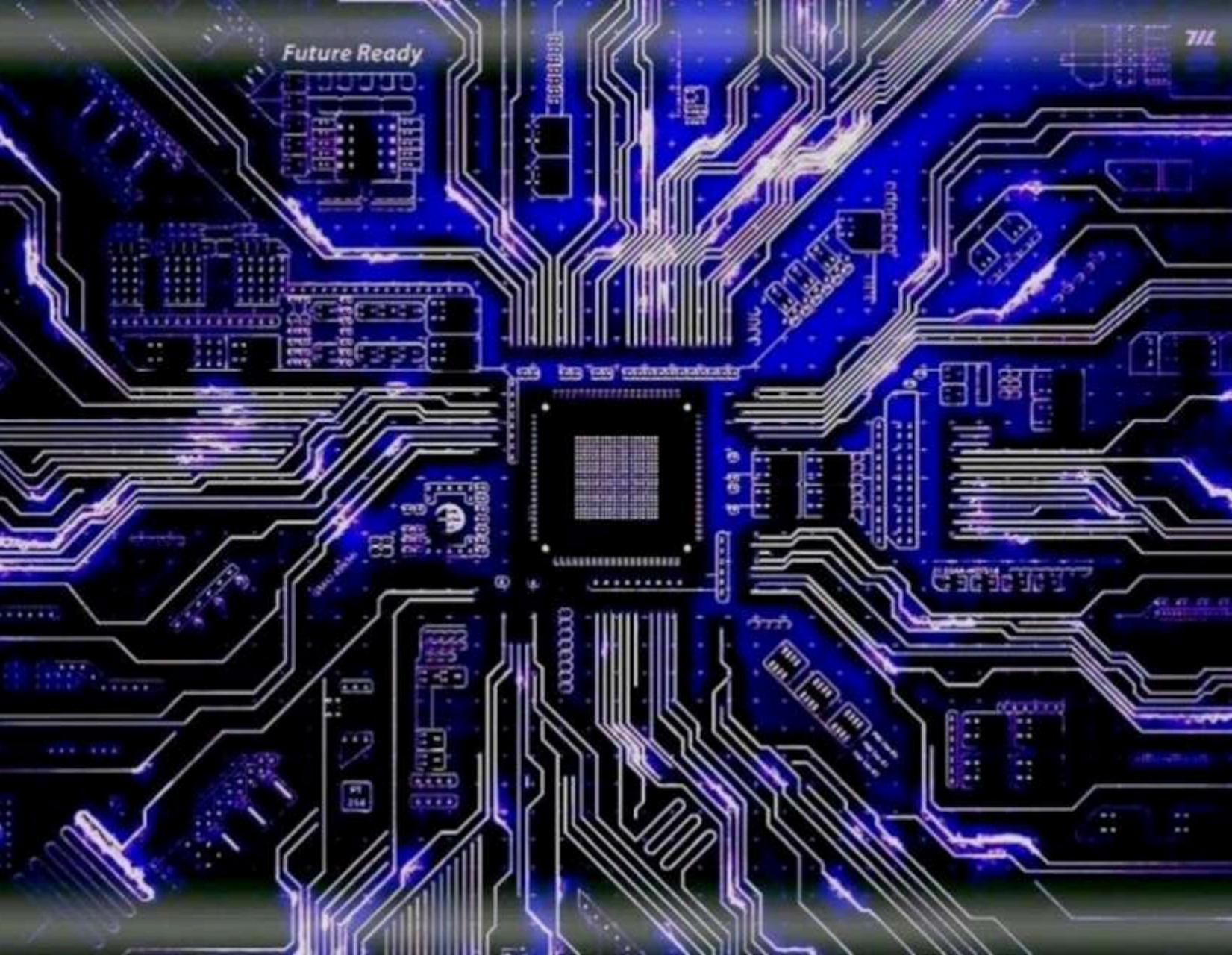


[6] Plantilla del Objeto Interactivo obtenida del sitio web del Proyecto Descartes, intervenida por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Capítulo II

Álgebra de Boole




Future Ready

El álgebra de Boole se utiliza para representar en forma de ecuaciones un circuito de lógica digital y de esta forma poder solucionarlos.[2][3][4][5]

Regla	Nombre	Fórmula
1	Leyes del producto	$A \times 0 = 0$
2		$A \times 1 = A$
3	Leyes de la suma	$A + 0 = A$
4		$A + 1 = 1$
5	Leyes de la tautología	$A \times A = A$
6		$A + A = A$
7	Leyes de los complementos	$A \times \overline{A} = 0$
8		$A + \overline{A} = 1$
9	Ley de la doble negación	$\overline{\overline{A}} = A$
10	Leyes conmutativas	$A \times B = B \times A$
11		$A + B = B + A$
12	Leyes distributivas	$A \times (B + C) = AB + AC$
13		$A + BC = (A + B) \times (A + C)$
14	Leyes asociativas	$ABC = (AB) \times C = A \times (BC) = (AC) \times B$
15		$A + B + C = (A + B) + C = A + (B + C) = (A + C) + B$

Tabla: Leyes básicas del Álgebra de Boole

Regla	Nombre	Fórmula
16	Leyes de absorción	$A \times (A + B) = A$
17		$A + (A \times B) = A$
18		$A \times (\overline{A} + B) = A \times B$
19		$\overline{A} \times (A + B) = \overline{A} \times B$
20		$\overline{A} + (A \times B) = \overline{A} + B$
21		$A + (\overline{A} \times B) = A + B$
22		$A + (\overline{A} \times \overline{B}) = A + \overline{B}$
23		$(A + B) \times (A + \overline{B}) = A$
24		$(A \times B) + (A \times \overline{B}) = A$
25	Leyes de Morgan	$\overline{A + B} = \overline{A} \times \overline{B}$
26		$\overline{A \times B} = \overline{A} + \overline{B}$

Tabla: CONTINUACIÓN: Leyes básicas del Álgebra de Boole

2.1 Demostración de las Reglas de Absorción

Regla 16: aplicar la regla 12 (ley distributiva)

$$F = A \times (A + B) \therefore F = (A \times A) + (A \times B)$$

$$F = (A) + (A \times B) \therefore F = A(1 + B)$$

$$F = A \times 1 \therefore \boldsymbol{F = A}$$

Regla 17: sacar factor común A

$$F = A + (A \times B) \therefore F = A(1 + B) \therefore \boldsymbol{F = A}$$

Regla 18: aplicar la regla 12 (ley distributiva)

$$F = A \times (\overline{A} + B) \therefore F = (A \times \overline{A}) + (A \times B)$$

$$F = 0 + (A \times B) \therefore \boldsymbol{F = A \times B}$$

Regla 19: aplicar la regla 12 (ley distributiva)

$$F = \overline{A} \times (A + B) \therefore F = (\overline{A} \times A) + (\overline{A} \times B)$$

$$F = 0 + (A \times B) \therefore \boldsymbol{F = A \times B}$$

Regla 20: aplicar la regla 13 (ley distributiva)

$$F = \overline{A} + (A \times B) \therefore F = (\overline{A} + A) \times (\overline{A} + B)$$

$$F = 1 \times (\overline{A} + B) \therefore \boldsymbol{F = \overline{A} + B}$$

Regla 21: aplicar la regla 13 (ley distributiva)

$$F = A + (\overline{A} \times B) \therefore F = (A + \overline{A}) \times (A + B)$$

$$F = 1 \times (A + B) \therefore \boldsymbol{F = A + B}$$

Regla 22: aplicar la regla 13 (ley distributiva)

$$F = A + (\overline{A} \times \overline{B}) \therefore F = (A + \overline{A}) \times (A + \overline{B})$$

$$F = 1 \times (A + \overline{B}) \therefore \boldsymbol{F = A + \overline{B}}$$

Regla 23: aplicar la regla 12 (ley distributiva)

$$F = (A + B) \times (A + \overline{B})$$

$$F = (A \times A) + (A \times \overline{B}) + (A \times B) + (B \times \overline{B})$$

$$F = A + (A \times \overline{B}) + (A \times B) + 0$$

$$F = A(1 + \overline{B} + B) \therefore \boldsymbol{F = A}$$

Regla 24: sacar factor común A

$$F = (A \times B) + (A \times \overline{B}) \therefore F = A \times (B + \overline{B})$$

$$F = A \times 1 \therefore \boldsymbol{F = A}$$

2.2 Ejemplos

Solución de los siguientes ejercicios utilizando las reglas del álgebra de Boole:

★ Ejemplo 1:

$$X = (A \times B \times C) + (B \times C) \qquad \boldsymbol{R/.} \quad \boldsymbol{X = (B \times C)}$$

Sacar factor común $(B \times C)$:

$$X = (B \times C) \times (A + 1) \therefore X = (B \times C) \times (1)$$

$$X = (B \times C)$$

★ Ejemplo 2:

$$J = X(\overline{X} + Y) \qquad \boldsymbol{R/.} \quad \boldsymbol{J = (X \times Y)}$$

Aplicar la regla 12:

$$J = (X \times \overline{X}) + (X \times Y)$$

Aplicar la regla 7:

$$\boxed{J = (X \times Y)}$$

★ Ejemplo 3:

$$T = (A \times B) + (A \times B \times C) + (A \times B \times \overline{C})$$

$$\boldsymbol{R/.\quad T = (A \times B)}$$

Sacar factor común $(A \times B)$ del segundo y tercer término:

$$T = (A \times B) + (A \times B)(C + \overline{C})$$

Aplicar la regla 8:

$$T = (A \times B) + (A \times B)$$

Aplicar la regla 6:

$$\boxed{T = (A \times B)}$$

★ Ejemplo 4:

$$M = (X + Y) \times (X + \overline{Y})$$

$$\boldsymbol{R/.\quad M = X}$$

- Multiplicar los términos:

$$M = XX + X\overline{Y} + XY + Y\overline{Y}$$

- Aplicar las reglas 5 y 7:

$$M = X + X\overline{Y} + XY$$

- Sacar factor común X del segundo y tercer término:

$$M = X + X(\overline{Y} + Y)$$

- Aplicar la regla 7:

$$M = X + X \therefore \boxed{M = X}$$

★ Ejemplo 5:

$$J = AB + A(B + C) + (B + C)$$

$$\boldsymbol{R/.\quad J = B + AC}$$

- Solucionar los productos del segundo y tercer término:

$$J = AB + AB + AC + BB + BC$$

- Aplicar la regla 6 al primer y segundo término y la regla 5 al cuarto término:

$$J = AB + AC + B + BC$$

- Sacar factor común B del primer y tercer término:

$$J = B(A + 1) + AC + BC$$

- Aplicar la regla 4 al primer término:

$$J = B + AC + BC$$

Sacar factor común B del primer y tercer término:

$$J = B(1 + C) + AC$$

Aplicar la regla 4 al primer término:

$$\boxed{J = B + AC}$$

★ Ejemplo 6:

$$S = A(\overline{B}C + C + \overline{C}D + \overline{C}) \quad \boldsymbol{R/.} \quad \boldsymbol{S = A}$$

Solucionar el producto:

$$S = A\overline{B}C + AC + A\overline{C}D + A\overline{C}$$

Sacar factor común AC del primer y segundo término; y $A\overline{C}$ del tercer y cuarto término:

$$S = AC(\overline{B} + 1) + A\overline{C}(D + 1)$$

Aplicar la regla 4:

$$S = AC + A\overline{C}$$

Sacar factor común A:

$$S = A(C + \overline{C}) \therefore \boxed{S = A}$$

★ Ejemplo 7:

$$Y = A\overline{B} + BC + B + A\overline{B}C \quad \boldsymbol{R/.} \quad \boldsymbol{Y = A + B}$$

Factor común $A\overline{B}$ del primero y cuarto término. Factor común B del segundo y tercer término:

$$Y = A\overline{B}(1 + C) + B(C + 1)$$

Aplicar la regla 4:

$$Y = A\overline{B} + B \therefore \boxed{Y = A + B}$$

★ Ejemplo 8:

$$G = \overline{(A + \overline{B} + C) + (B + \overline{C})} \quad \boldsymbol{R/.} \quad \boldsymbol{G = 0}$$

Aplicar la regla 25 (ley de Morgan):

$$G = \overline{(A + \overline{B} + C)} \times \overline{(B + \overline{C})}$$

Aplicar la regla 26 (ley de Morgan):

$$G = (\overline{A} \times B \times \overline{C}) \times (\overline{B} \times C) \therefore \boxed{G = 0}$$

★ Ejemplo 9:

$$N = \overline{(\overline{A}B + A\overline{B})} \quad \boldsymbol{R/.} \quad \boldsymbol{N = \overline{A \oplus B}}$$

Aplicar la regla 25 (ley de Morgan):

$$N = \overline{(\overline{A} \times B)} \times \overline{(A \times \overline{B})}$$

$$N = (A + \overline{B}) \times (\overline{A} + B)$$

Aplicar la regla 25 (ley de Morgan):

$$N = \overline{(\overline{A} \times B)} \times \overline{(A \times \overline{B})}$$

$$N = (A + \overline{B}) \times (\overline{A} + B)$$

Solucionar el producto:

$$N = A\overline{A} + AB + \overline{A}\,\overline{B} + \overline{B}B$$

Función lógica de la compuerta XNOR:

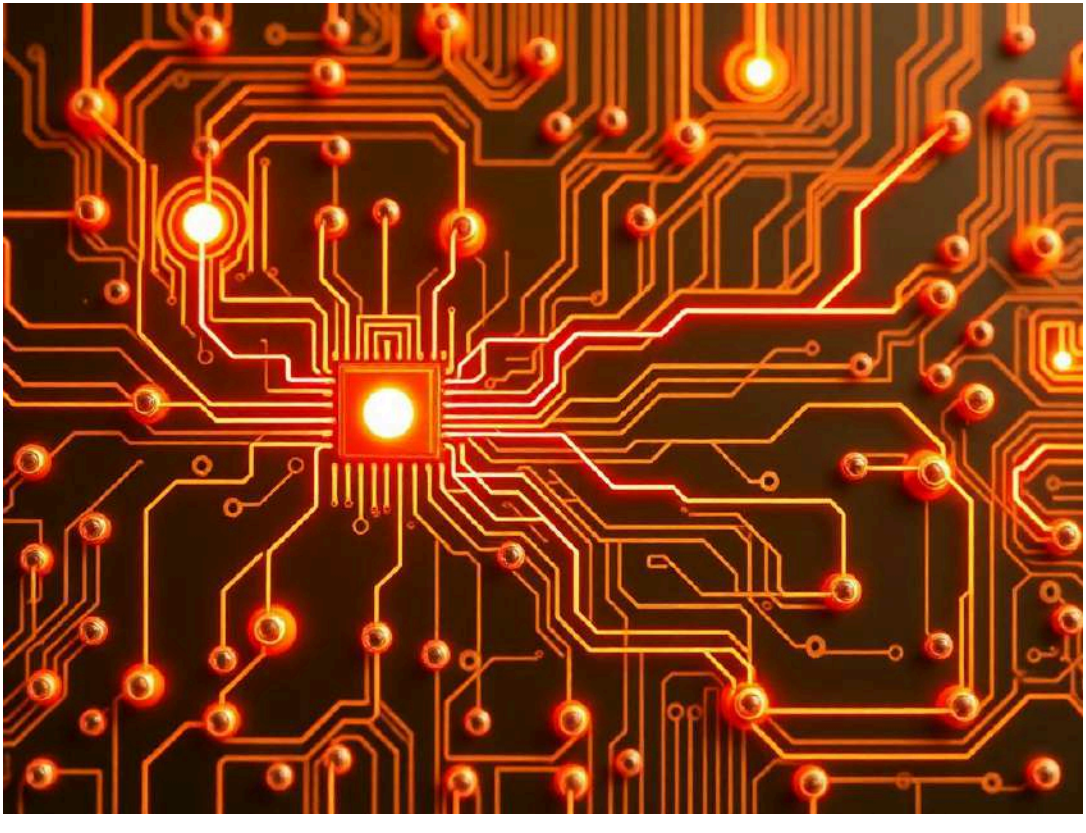
$$N = \overline{A \oplus B}$$

★ Ejemplo 10:

$$K = \overline{A + \overline{B} + \overline{CD}} \quad \boldsymbol{R/.} \quad \boldsymbol{K = \overline{A}BCD}$$

Aplicar la regla 25 (ley de Morgan):

$$K = \overline{A}BCD$$



2.3 Ejercicios propuestos con respuestas

Solucionar los siguientes ejercicios utilizando las reglas del álgebra de Boole (ver la tabla con las leyes básicas del álgebra de Boole):

1. $R = \overline{AB + AC} + \overline{A}\,\overline{B}\,C$
   R/. $\boldsymbol{R = \overline{A} + \overline{B}\,\overline{C}}$
2. $X = (\overline{A} + B) \times (A + B + D) \times \overline{D}$
   R/. $\boldsymbol{X = B\overline{D}}$
3. $F = \overline{(X + Y) \times \overline{(X\overline{Y} + Z)}}$
   R/. $\boldsymbol{F = \overline{Y} + Z}$
4. $F = \overline{\overline{(\overline{X}\,\overline{Y} + XZ)} \times \overline{(\overline{X} + \overline{Y}Z)}}$
   R/. $\boldsymbol{F = \overline{X} + Z}$
5. $F = M(X + Y) + MY + Y(X + Y) + MXY$
   R/. $\boldsymbol{F = MX + Y}$
6. $W = A\overline{B} + A\overline{(B + C)} + B\overline{(B + C)}$
   R/. $\boldsymbol{W = A\overline{B}}$
7. $H = (A + B + C) \times (A + B)$
   R/. $\boldsymbol{H = A + B}$
8. $H = (AB) + A(B + C) + B(B + C)$
   R/. $\boldsymbol{H = B + AC}$
9. $L = (X + Y) \times (\overline{X} + Z) \times (Y + Z)$
   R/. $\boldsymbol{L = XZ + Y(\overline{X} + Z)}$
10. $F = \overline{A}B + \overline{ABC} + C(\overline{B} + A)$
    R/. $\boldsymbol{F = 1}$

11. $T = \overline{\overline{A}(C+D) + \overline{B}(A+D) + \overline{A}\,\overline{B}\,\overline{C}}$

R/. $\boldsymbol{T = (A + \overline{C}\,\overline{D}) \times B}$

12. $Z = \left[A\overline{B}(C + BD)\right] + \overline{A}\,\overline{B}$

R/. $\boldsymbol{Z = \overline{B}C}$

13. $P = ABC + AC + C$

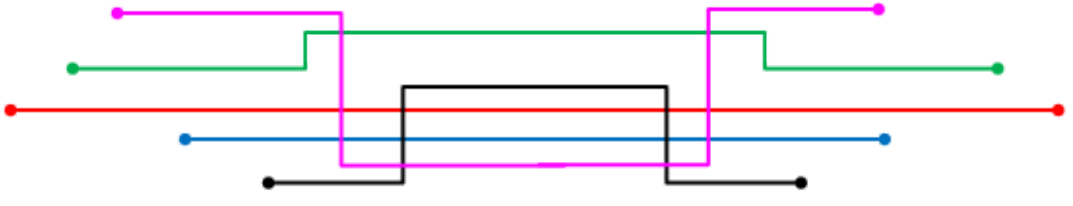
R/. $\boldsymbol{P = C}$

14. $K = \overline{A}\,\overline{B} + \overline{A}B + A\overline{B} + AB$

R/. $\boldsymbol{K = 1}$

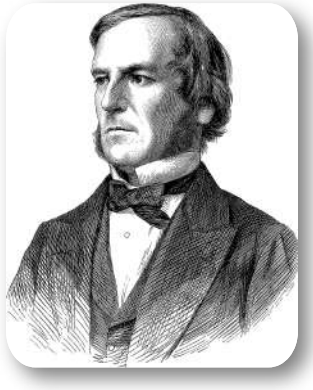
15. $J = \overline{\overline{XY}\,Z} + XZ$

R/. $\boldsymbol{J = \overline{Z}}$



"No importa qué tan correcto un teorema matemático puede parecer, uno nunca debe estar convencido de que no había algo imperfecto en ello hasta que también da la impresión de ser hermosa".

George Boole



AYUDAS

COMPLEMENTOS

2.4 Videos

Haga clic en la esquina superior derecha para expandir el video.

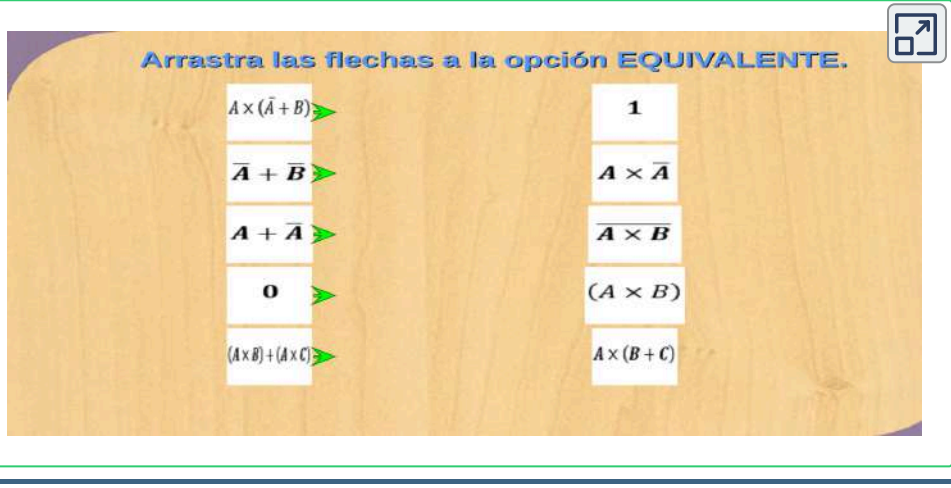
Videos demostrativos sobre la solución de ejercicios con el Álgebra de Boole [7]

[7] Videos realizados por "Les Ingenieurs", Simplificación por ÁLGEBRA BOOLEANA ejercicios [Aprende con 4 ejercicios] y CÓMO Simplificar FUNCIONES BOOLEANAS [Álgebra de Boole] Electrónica Digital

2.5 Objetos Interactivos

Haga clic sobre la imagen para abrir el objeto interactivo:

Objetos Interactivos utilizando el Álgebra Booleana.[8]



[8] El primer objeto interactivo fue generado con la *asistencia* de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/. La plantilla del segundo fue obtenida del sitio web del Proyecto Descartes, intervenida por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Capítulo III

Operaciones Lógicas Binarias

Las ***Operaciones Lógicas Binarias*** se basan en el ***Sistema numérico Binario*** y se implementan mediante circuitos lógicos combinacionales que realizan operaciones básicas como: suma, resta, multiplicación y división.[2][3][4][5][6][7][8][9][10]

3.1 Suma lógica Binaria

Se suman los dos términos sumandos y si el resultado excede se agrega un '1' a la izquierda del sumando que se está analizando (acarreo = carry).

Suma Binaria
0+0 = 0
0+1 = 1
1+0 = 1
1+1 = 0 y un acarreo de '1' (carry)

Tabla: Reglas de la Suma Binaria

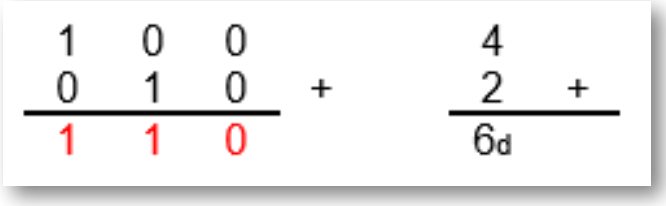
Solucionar las siguientes Sumas Binarias y verificar sus resultados con el sistema numérico decimal:

NOTA: **para la explicación de los ejemplos, se nombrarán las columnas de derecha a izquierda, así:**

←

C4 C3 C2 C1
(MSB) ... (LSB)

★ Ejemplo 1:



Paso 1: se suman los dos primeros sumandos de C1:

$$0 + 0 = 0$$

Paso 2: se suman los sumandos de C2:

$$0 + 1 = 1$$

Paso 3: se suman los sumandos de C3:

$$1 + 0 = 1$$

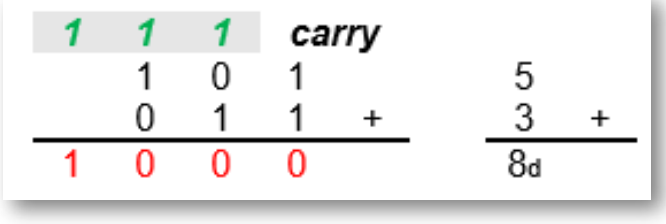
Paso 4: la respuesta binaria de la suma es:

$$100 + 010 = 110_b$$

Paso 5: verificación en el sistema numérico decimal:

$$4 + 2 = 6_d$$

★ Ejemplo 2:



Paso 1: se suman los sumandos de C1:

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 2: se suma el carry anterior con los dos sumandos de C2:

$$1 + 0 = 1$$

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 3: se suma el carry anterior con los dos sumandos de C3:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 0 = 0$$

Paso 4: se suma el carry anterior que está en C4:

$$1$$

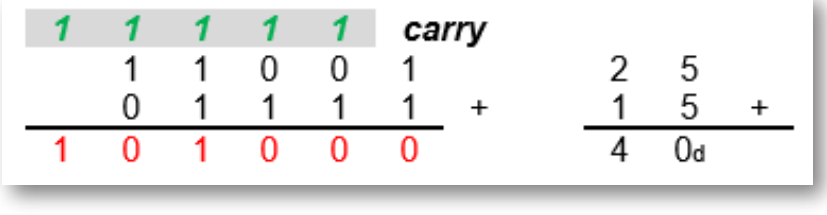
Paso 5: la respuesta binaria de la suma es:

$$101 + 011 = 1000_b$$

Paso 6: verificación en el sistema numérico decimal:

$$5 + 3 = 8_d$$

★ Ejemplo 3:



Paso 1: se suman los sumandos de C1:

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 2: se suma el carry anterior con los dos sumandos de C2:

$$1 + 0 = 1$$

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 3: se suma el carry anterior con los dos sumandos de C3:

$$1 + 0 = 1$$

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 4: se suma el carry anterior con los dos sumandos de C4:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 1 = 1$$

Paso 5: se suma el carry anterior con los dos sumandos de C5:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 0 = 0$$

Paso 6: se suma el carry anterior que está en C6:

$$1$$

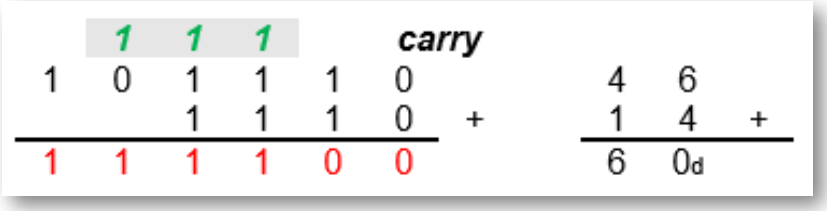
Paso 7: la respuesta de la suma es:

$$11001 + 01111 = 101000_b$$

Paso 8: verificación en el sistema numérico decimal:

$$25 + 15 = 40_d$$

★ Ejemplo 4:



Paso 1: se suman los sumandos de C1:

$$0 + 0 = 0$$

Paso 2: se suman los dos sumandos de C2:

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 3: se suma el carry anterior con los dos sumandos de C3:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 1 = 1$$

Paso 4: se suma el carry anterior con los dos sumandos de C4:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 1 = 1$$

Paso 5: se suma el carry anterior con el sumando de C5:

$$1 + 0 = 1$$

Paso 6: el sumando de C6:

$$1$$

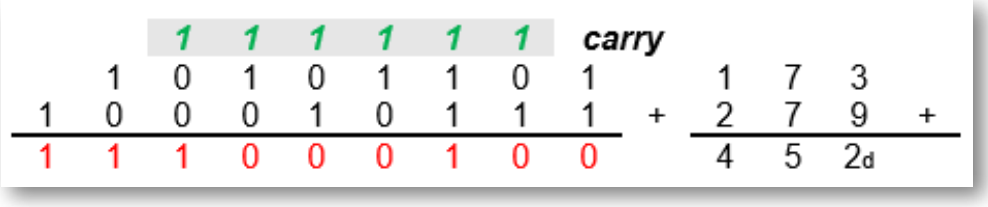
Paso 7: la respuesta de la suma es:

$$101110 + 1110 = 111100_b$$

Paso 8: verificación en el sistema numérico decimal:

$$46 + 14 = 60_d$$

★ Ejemplo 5:



Paso 1: se suman los sumandos de C1:

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 2: se suman los dos sumandos de C2:

$$1 + 0 = 1$$

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 3: se suma el carry anterior con los dos sumandos de C3:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 1 = 1$$

Paso 4: se suma el carry anterior con los dos sumandos de C4:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 0 = 0$$

Paso 5: se suma el carry anterior con los dos sumandos de C5:

$$1 + 0 = 1$$

$$1 + 1 = 0 \quad y \quad carry = 1$$

Paso 6: se suma el carry anterior con los dos sumandos de C6:

$$1 + 1 = 0 \quad y \quad carry = 1$$

$$0 + 0 = 0$$

Paso 7: se suma el carry anterior con los dos sumandos de C7:

$$1 + 0 = 1$$

$$1 + 0 = 1$$

Paso 8: se suman los sumandos de C8:

$$1 + 0 = 1$$

Paso 9: el sumando de C9:

$$1$$

Paso 10: la respuesta de la suma es:

$$10101101 + 100010111 = 111000100_b$$

Paso 11: verificación en el sistema numérico decimal:

$$173 + 279 = 452_d$$

3.2 Resta lógica Binaria

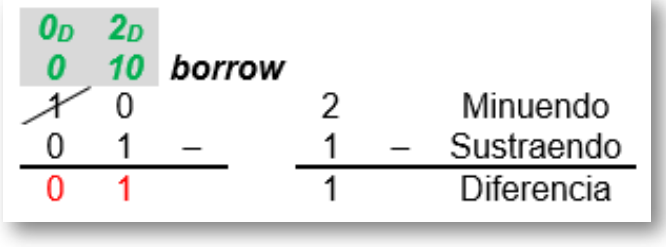
Se sustrae del minuendo el sustraendo, entregando como resultado la diferencia; si el sustraendo excede el minuendo se extrae el '1' del minuendo que está a la izquierda convirtiéndose el de la izquierda en '0', equivaliendo el nuevo minuendo que se está analizando al valor $10_B = 2_D$.

Resta Binaria
0-0 = 0
0-1 = 1 y un borrow de '1'
1-0 = 1
1-1 = 0

Tabla: Reglas de la Resta Binaria

Solucionar las siguientes Restas Binarias y verificar sus resultados con el sistema numérico decimal:

★ Ejemplo 1:



Paso 1: se resta del minuendo el sustraendo de C1; como no tiene pide prestado al minuendo de C2:

$$0 - 1 = 1 \quad y \quad presta = 1$$

El minuendo de C2 queda convertido en '0'

Paso 2: se restan los términos de C2:

$$0 - 0 = 0$$

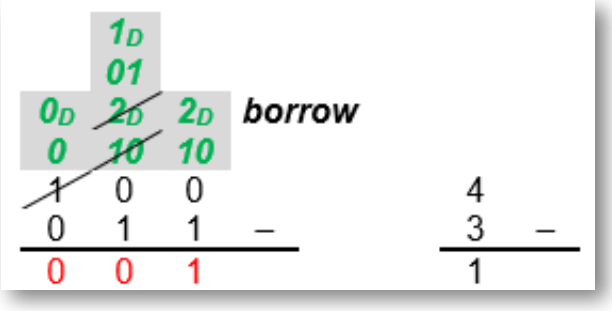
Paso 3: la respuesta de la resta es:

$$10 - 01 = 01_b$$

Paso 4: verificación en el sistema numérico decimal:

$$2 - 1 = 1_d$$

★ Ejemplo 2:



Paso 1: se resta del minuendo el sustraendo de C1; como no tiene pide prestado al minuendo de C2 y el minuendo de C2 no tiene, entonces le pide prestado al minuendo de C3:

$$0 - 1 = 1 \quad y \quad presta = 1$$

Inicialmente, el minuendo de C2 queda convertido en '10', pero al prestarle al minuendo de C1 que queda convertido en '01'

El minuendo de C3 queda convertido en '0'

Paso 2: se resta del minuendo el sustraendo de C2:

$$1 - 1 = 0$$

Paso 3: se resta del minuendo el sustraendo de C3:

$$0 - 0 = 0$$

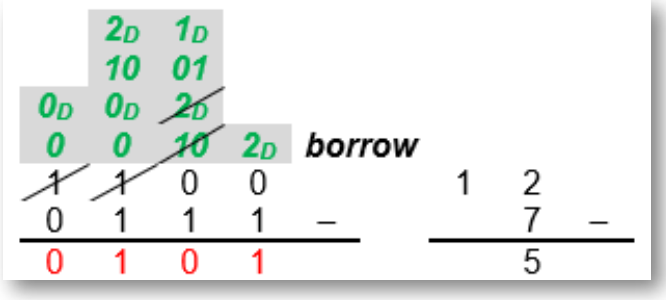
Paso 4: la respuesta de la resta es:

$$100 - 011 = 001_b$$

Paso 5: verificación en el sistema numérico decimal:

$$4 - 3 = 1_d$$

★ Ejemplo 3:



Paso 1: se resta del minuendo el sustraendo de C1; como no tiene pide prestado al minuendo de C2 y el minuendo de C2 no tiene, entonces le pide prestado al minuendo de C3:

$$0 - 1 = 1 \quad y \quad presta = 1$$

Inicialmente, el minuendo de C2 queda convertido en '10', pero al prestarle al minuendo de C1 que queda convertido en '01'

El minuendo de C3 queda convertido en '0'

Paso 2: se resta del minuendo el sustraendo de C2:

$$1 - 1 = 0$$

Paso 3: se resta del minuendo el sustraendo de C3; como no tiene pide prestado al minuendo de C4:

$$10 - 1 = 1$$

Paso 4: se resta del minuendo el sustraendo de C4:

$$0 - 0 = 0$$

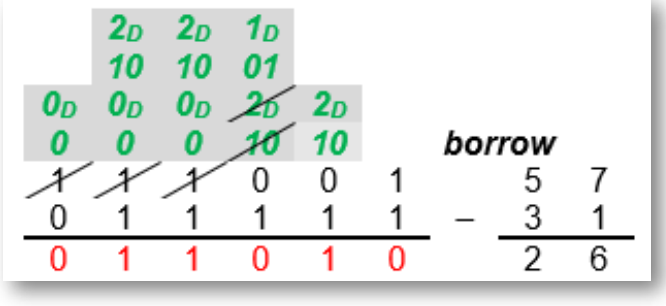
Paso 5: la respuesta de la resta es:

$$1100 - 0111 = 0101_b$$

Paso 6: verificación en el sistema numérico decimal:

$$12 - 7 = 5_d$$

★ Ejemplo 4:



Paso 1: se resta del minuendo el sustraendo de C1; como no tiene pide prestado al minuendo de C2 y el minuendo de C2 no tiene, entonces le pide prestado al minuendo de C3:

$$1 - 1 = 0$$

Paso 2: se resta del minuendo el sustraendo de C2; como no tiene pide prestado al minuendo de C3 y el minuendo de C3 no tiene, entonces le pide prestado al minuendo de C4:

$$10 - 1 = 1 \quad y \quad presta = 1$$

Inicialmente, el minuendo de C3 queda convertido en '10', pero al prestarle al minuendo de C2 que queda convertido en '01'

El minuendo de C4 queda convertido en '0'

Paso 3: se resta del minuendo el sustraendo de C3:

$$01 - 1 = 0$$

Paso 4: se resta del minuendo el sustraendo de C4; como no tiene pide prestado al minuendo de C5:

$$10 - 1 = 1$$

El minuendo de C5 queda convertido en '0'

Paso 5: se resta del minuendo el sustraendo de C5; como no tiene pide prestado al minuendo de C6:

$$10 - 1 = 1$$

El minuendo de C6 queda convertido en '0'

Paso 6: se resta del minuendo el sustraendo de C6:

$$0 - 0 = 0$$

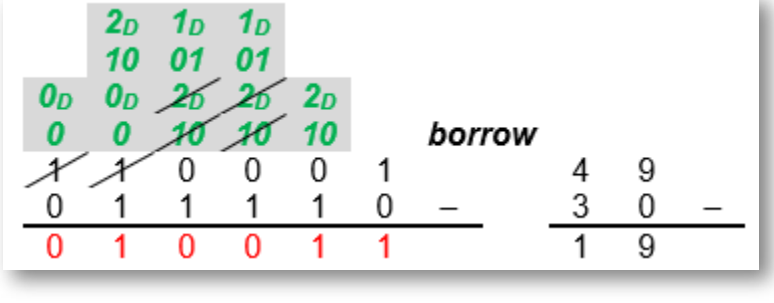
Paso 7: la respuesta de la resta es:

$$111001 - 011111 = 011010_b$$

Paso 8: verificación en el sistema numérico decimal:

$$57 - 31 = 26_d$$

★ Ejemplo 5:



Paso 1: se resta del minuendo el sustraendo de C1:

$$1 - 0 = 1$$

Paso 2: se resta del minuendo el sustraendo de C2; como no tiene pide prestado al minuendo de C3 y el minuendo de C3 no tiene, entonces le pide prestado al minuendo de C4; como no tiene pide prestado al minuendo de C5:

$$10 - 1 = 1 \quad y \quad presta = 1$$

Inicialmente, el minuendo de C3 queda convertido en '10', pero al prestarle al minuendo de C2 que queda convertido en '01'

Inicialmente, el minuendo de C4 queda convertido en '10', pero al prestarle al minuendo de C3 que queda convertido en '01'

El minuendo de C5 queda convertido en '0'

Paso 3: se resta del minuendo el sustraendo de C3:

$$01 - 1 = 0 \quad y \quad presta = 1$$

Paso 4: se resta del minuendo el sustraendo de C4:

$$01 - 1 = 0 \quad y \quad presta = 1$$

Paso 5: se resta del minuendo el sustraendo de C5; como no tiene pide prestado al minuendo de C6:

$$10 - 1 = 1 \quad y \quad presta = 1$$

El minuendo de C6 queda convertido en '0'

Paso 6: se resta del minuendo el sustraendo de C6:

$$0 - 0 = 0$$

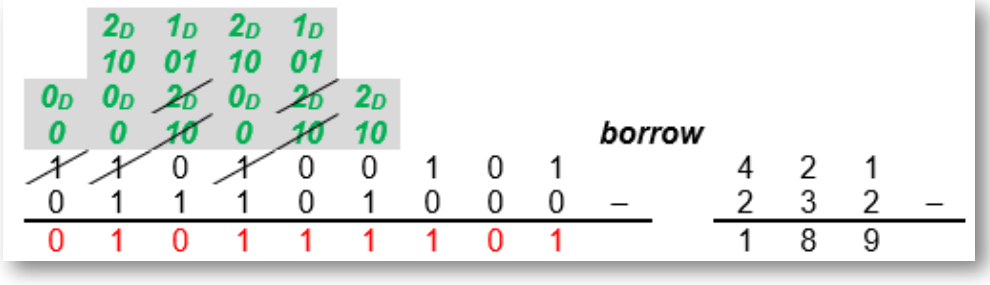
Paso 7: la respuesta de la resta es:

$$110001 - 011110 = 010011_b$$

Paso 8: verificación en el sistema numérico decimal:

$$49 - 30 = 19_d$$

★ Ejemplo 6:



Paso 1: se resta del minuendo el sustraendo de C1:

$$1 - 0 = 1$$

Paso 2: se resta del minuendo el sustraendo de C2:

$$0 - 0 = 0$$

Paso 3: se resta del minuendo el sustraendo de C3:

$$1 - 0 = 1$$

Paso 4: se resta del minuendo el sustraendo de C4; como no tiene pide prestado al minuendo de C5 y el minuendo de C5 no tiene, entonces le pide prestado al minuendo de C6:

$$10 - 1 = 1 \quad y \quad presta = 1$$

Inicialmente, el minuendo de C5 queda convertido en '10', pero al prestarle al minuendo de C4 que queda convertido en '01'

El minuendo de C6 queda convertido en '0'

Paso 5: se resta del minuendo el sustraendo de C5:

$$01 - 0 = 1$$

Paso 6: se resta del minuendo el sustraendo de C6; como no tiene pide prestado al minuendo de C7 y el minuendo de C7 no tiene, entonces le pide prestado al minuendo de C8:

$$10 - 1 = 1 \quad y \quad presta = 1$$

Inicialmente, el minuendo de C7 queda convertido en '10', pero al prestarle al minuendo de C6 que queda convertido en '01'

El minuendo de C8 queda convertido en '0'

Paso 7: se resta del minuendo el sustraendo de C7:

$$01 - 1 = 0$$

Paso 8: se resta del minuendo el sustraendo de C8; como no tiene pide prestado al minuendo de C9:

$$10 - 1 = 1$$

El minuendo de C9 queda convertido en '0'

Paso 9: se resta del minuendo el sustraendo de C9:

$$0 - 0 = 0$$

Paso 10: la respuesta de la resta es:

$$110100101 - 011101000 = 010111101_b$$

Paso 11: verificación en el sistema numérico decimal:

$$421 - 232 = 189_d$$

Paso 11: verificación en el sistema numérico decimal:

$$421 - 232 = 189_d$$

3.2.1 Otro método de solución

La sustracción o resta es una suma cambiando el signo del sustraendo:

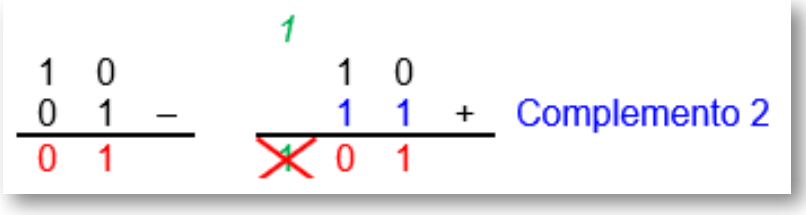
- El signo de un número binario positivo o negativo se cambia hallando su complemento 2.
- Para restar dos números con signo, se calcula el complemento 2 del sustraendo y luego se le suma al minuendo, aplicando las reglas de la suma.
- Los acarreos pueden generar un bit de ***"desborde"*** al final de la operación en el extremo izquierdo, este se descarta.

Una forma rápida y fácil de hallar el complemento 2 de un número es:

1. Se toma el número de derecha a izquierda (desde el LSB hasta el MSB) y se escriben igual hasta encontrar el primer '1' incluyéndolo.
2. A partir del primer '1', se cambian los '1' por '0' y viceversa, hasta llegar al final del número.
3. El resultado final es el valor obtenido.

Solucionar las mismas Restas Binarias anteriores utilizando este método y verificar sus resultados:

★ Ejemplo 1:



Paso 1: hallar el complemento 2 del sustraendo de la operación:

- Se escribe igual hasta el primer '1', incluyéndolo = 1
- Se cambian los '1' por '0' y viceversa = 1
- El número completo es = 11

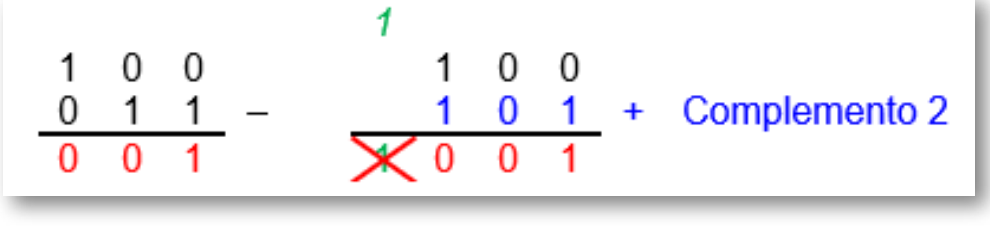
Paso 2: se suma el minuendo de la resta con el número obtenido en complemento 2:

$$10 + 11 = 101 \quad se\ descarta\ el\ 1$$

Paso 3: la respuesta de la operación es:

$$01_b = 1_d$$

★ Ejemplo 2:



Paso 1: hallar el complemento 2 del sustraendo de la operación:

- *Se escribe igual hasta el primer '1', incluyéndolo* = 1
- *Se cambian los '1' por '0' y viceversa* = 10
- *El número completo es* = 101

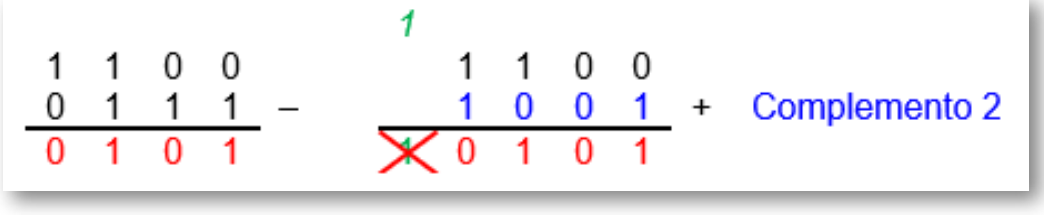
Paso 2: se suma el minuendo de la resta con el número obtenido en complemento 2:

$$100 + 101 = 1001 \quad se\ descarta\ el\ 1$$

Paso 3: la respuesta de la operación es:

$$001_b = 1_d$$

★ Ejemplo 3:



Paso 1: hallar el complemento 2 del sustraendo de la operación:

- Se escribe igual hasta el primer '1', incluyéndolo = 1
- Se cambian los '1' por '0' y viceversa = 100
- El número completo es = 1001

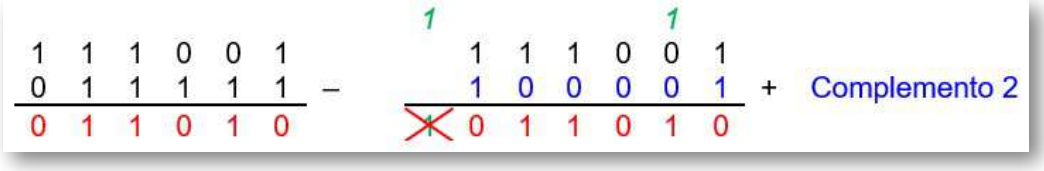
Paso 2: se suma el minuendo de la resta con el número obtenido en complemento 2:

$$1100 + 1001 = 10101 \quad se\ descarta\ el\ 1$$

Paso 3: la respuesta de la operación es:

$$0101_b = 5_d$$

★ Ejemplo 4:



Paso 1: hallar el complemento 2 del sustraendo de la operación:

- Se escribe igual hasta el primer '1', incluyéndolo = 1
- Se cambian los '1' por '0' y viceversa = 10000
- El número completo es = 100001

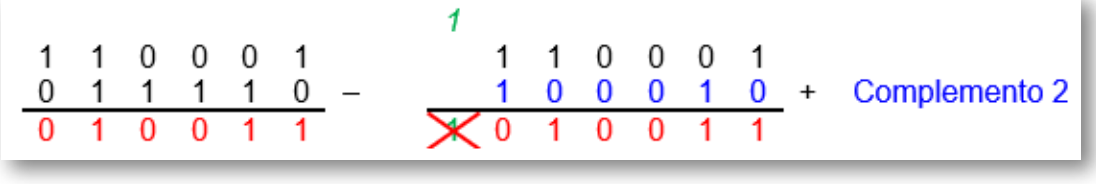
Paso 2: se suma el minuendo de la resta con el número obtenido en complemento 2:

$$111001 + 100001 = 1011010 \quad \textit{se descarta el } 1$$

Paso 3: la respuesta de la operación es:

$$011010_b = 26_d$$

★ Ejemplo 5:



Paso 1: hallar el complemento 2 del sustraendo de la operación:

- Se escribe igual hasta el primer '1', incluyéndolo = 10
- Se cambian los '1' por '0' y viceversa = 1000
- El número completo es = 100010

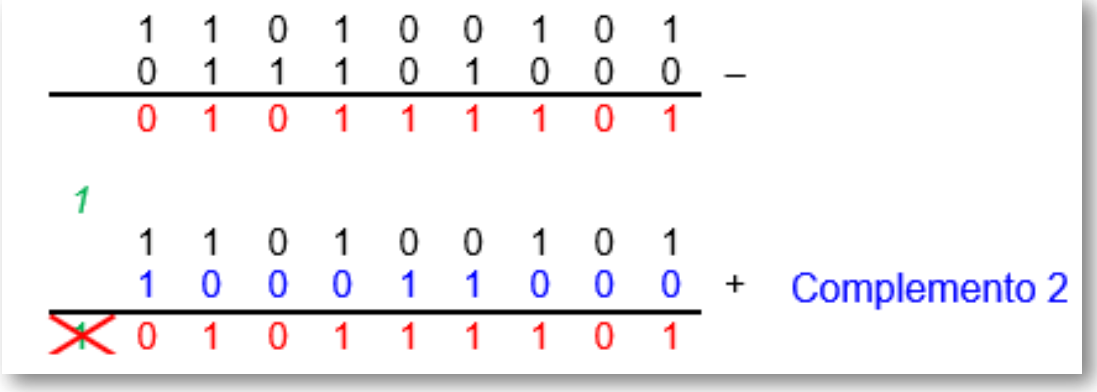
Paso 2: se suma el minuendo de la resta con el número obtenido en complemento 2:

$$110001 + 100010 = 10101 \quad se\ descarta\ el\ 1$$

Paso 3: la respuesta de la operación es:

$$010011_b = 19_d$$

★ Ejemplo 6:



Paso 1: hallar el complemento 2 del sustraendo de la operación:

- Se escribe igual hasta el primer '1', incluyéndolo = 1000
- Se cambian los '1' por '0' y viceversa = 10001
- El número completo es = 100011000

Paso 2: se suma el minuendo de la resta con el número obtenido en complemento 2:

$$110100101 + 100011000 = 10101 \quad se\ descarta\ el\ 1$$

Paso 3: la respuesta de la operación es:

$$010111101_b = 189_d$$

3.3 Multiplicación lógica Binaria

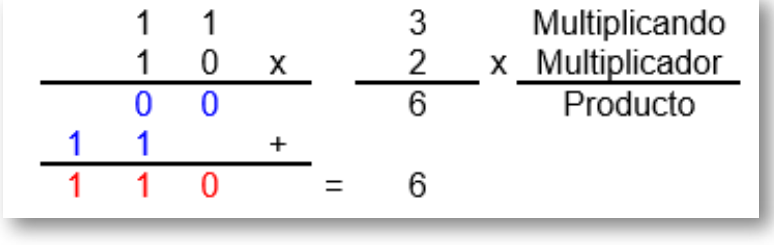
La multiplicación de números binarios se realiza de forma similar a la multiplicación de números decimales, salvo que la suma de los productos parciales se realiza en binario.

Multiplicación Binaria
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Tabla: Reglas de la Multiplicación Binaria

Solucionar las siguientes Multiplicaciones Binarias y verificar sus resultados con el sistema numérico decimal:

★ Ejemplo 1:



Paso 1: se realiza el producto de cada bit ***Multiplicador*** por cada bit del ***Multiplicando***:

$$0 \times 11 = 00$$

$$1 \times 11 = 11$$

Paso 2: se suman los anteriores resultados parciales respetando las posiciones que ocupan dentro de la solución del producto:

$$110$$

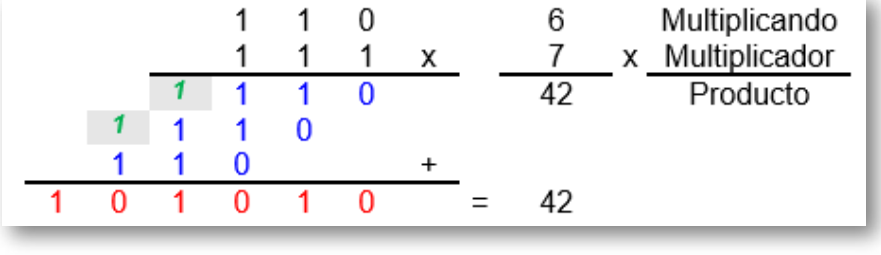
Paso 3: el resultado de la multiplicación es:

$$11 \times 10 = 110_b = 6_d$$

Paso 4: verificación en el sistema numérico decimal:

$$3 \times 2 = 6_d$$

★ Ejemplo 2:



Paso 1: se realiza el producto de cada bit ***Multiplicador*** por cada bit del ***Multiplicando***:

$$1 \times 110 = 110$$

$$1 \times 110 = 110$$

$$1 \times 110 = 110$$

Paso 2: se suman los anteriores resultados parciales respetando las posiciones que ocupan dentro de la solución del producto:

$$101010$$

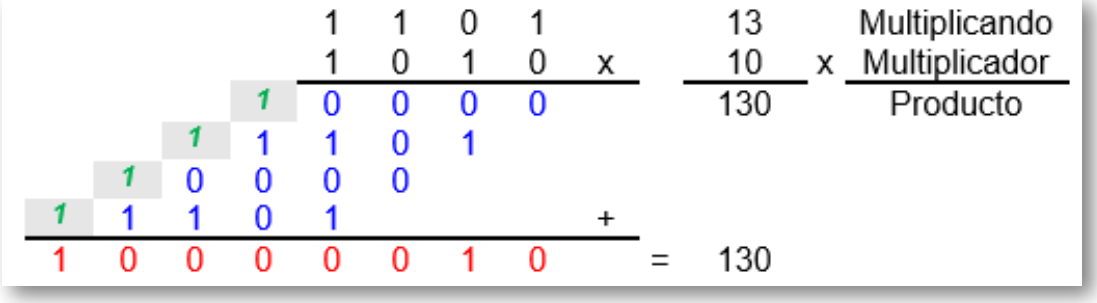
Paso 3: el resultado de la multiplicación es:

$$110 \times 111 = 101010_b = 42_d$$

Paso 4: verificación en el sistema numérico decimal:

$$6 \times 7 = 42_d$$

★ Ejemplo 3:



Paso 1: se realiza el producto de cada bit ***Multiplicador*** por cada bit del ***Multiplicando***:

$$0 \times 1101 = 0000$$

$$1 \times 1101 = 1101$$

$$0 \times 1101 = 0000$$

$$1 \times 1101 = 1101$$

Paso 2: se suman los anteriores resultados parciales respetando las posiciones que ocupan dentro de la solución del producto:

$$10000010$$

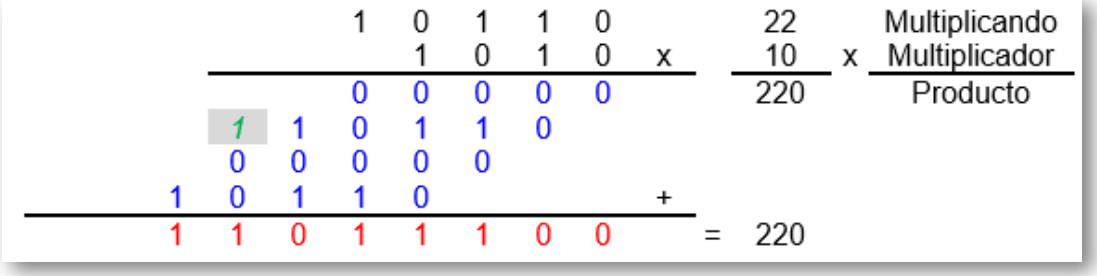
Paso 3: el resultado de la multiplicación es:

$$1101 \times 1010 = 10000010_b = 130_d$$

Paso 4: verificación en el sistema numérico decimal:

$$13 \times 10 = 130_d$$

★ Ejemplo 4:



Paso 1: se realiza el producto de cada bit ***Multiplicador*** por cada bit del ***Multiplicando***:

$$0 \times 10110 = 00000$$

$$1 \times 10110 = 10110$$

$$0 \times 10110 = 00000$$

$$1 \times 10110 = 10110$$

Paso 2: se suman los anteriores resultados parciales respetando las posiciones que ocupan dentro de la solución del producto:

$$11011100$$

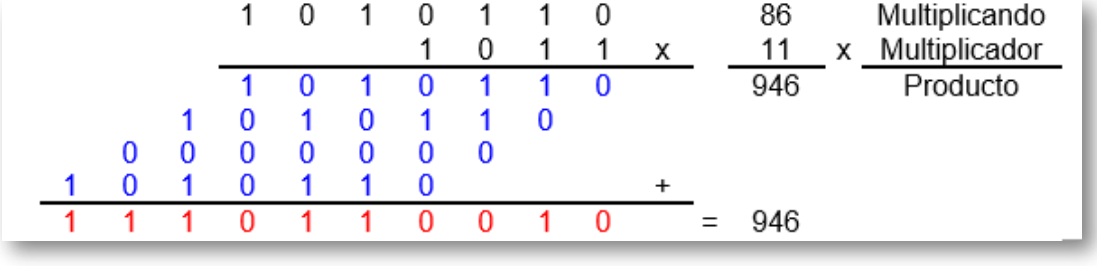
Paso 3: el resultado de la multiplicación es:

$$10110 \times 1010 = 11011100_b = 220_d$$

Paso 4: verificación en el sistema numérico decimal:

$$22 \times 10 = 220_d$$

★ Ejemplo 5:



Paso 1: se realiza el producto de cada bit ***Multiplicador*** por cada bit del **Multiplicando**:

$$1 \times 1010110 = 1010110$$

$$1 \times 1010110 = 1010110$$

$$0 \times 1010110 = 0000000$$

$$1 \times 1010110 = 1010110$$

Paso 2: se suman los anteriores resultados parciales respetando las posiciones que ocupan dentro de la solución del producto:

$$1110110010$$

Paso 3: el resultado de la multiplicación es:

$$1010110 \times 1011 = 1110110010_b = 946_d$$

Paso 4: verificación en el sistema numérico decimal:

$$86 \times 11 = 946_d$$

3.4 División lógica Binaria

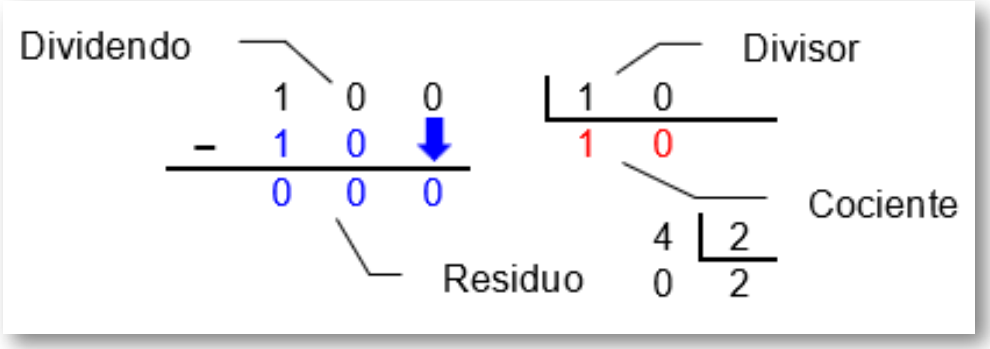
Se realiza de forma similar a la división de los números decimales, salvo que las multiplicaciones y restas internas al proceso de la división se hacen en sistema numérico binario.

División Binaria
0 ÷ 1 = 0
1 ÷ 1 = 1

Tabla: Reglas de la División Binaria

Solucionar las siguientes Divisiones Binarias y verificar sus resultados con el sistema numérico decimal:

★ Ejemplo 1:



Paso 1: como el Divisor tiene 2 bits (10), se toman igualmente 2 bits en el Dividendo (10):

- El $Dividendo > Divisor$ y lo contiene **1** vez
- Se realiza el producto de: $1 \times 10 = 10$
- Se halla la diferencia entre: $10 - 10 = 00$

Paso 2:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '000'
- Como el nuevo $Dividendo < Divisor$ se coloca un **0** en el Cociente
- Como no hay más bits para bajar, la operación termina.

Paso 3: los resultados del Cociente y el Residuo son:

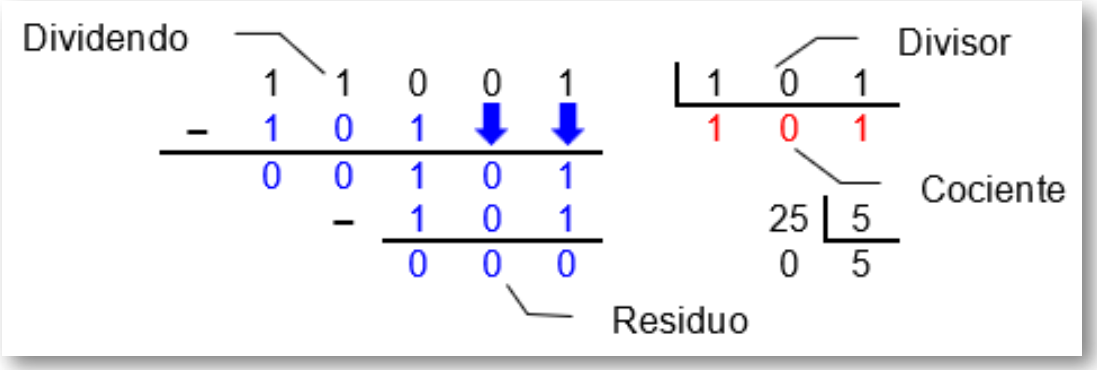
$$Cociente: \boxed{10_b = 2_d}$$

$$Residuo: \boxed{000_b = 0_d}$$

Paso 4: verificación en el sistema numérico decimal:

$$\boxed{4 \div 2 = 2_d}$$

★ Ejemplo 2:



Paso 1: como el Divisor tiene 3 bits (101), se toman igualmente 3 bits en el Dividendo (110):

- El $Dividendo > Divisor$ y lo contiene **1** vez
- Se realiza el producto de: $1 \times 101 = 101$
- Se halla la diferencia entre: $110 - 101 = 001$

Paso 2:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '0010'
- Como el $Dividendo < Divisor$ se coloca un **0** en el Cociente
- Se baja otro bit (1) y el nuevo Dividendo es '00101'
- Como el $Dividendo > Divisor$ se coloca un **1** en el Cociente
- Se realiza la operación: $1 \times 101 = 101$
- Se halla la diferencia entre: $101 - 101 = 000$
- Como no hay más bits para bajar, la operación termina

Paso 3: los resultados del Cociente y el Residuo son:

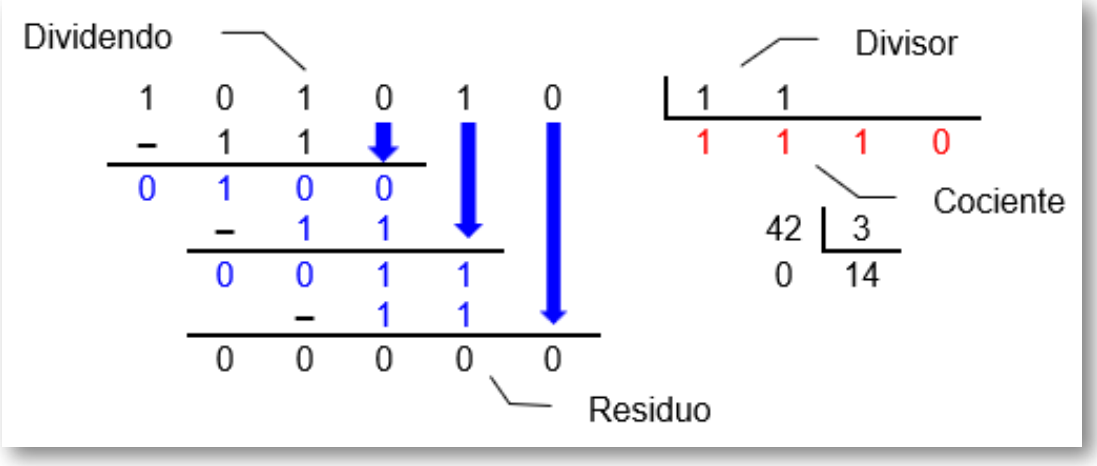
$$Cociente: \boxed{101_b = 5_d}$$

$$Residuo: \boxed{000_b = 0_d}$$

Paso 4: verificación en el sistema numérico decimal:

$$\boxed{25 \div 5 = 5_d}$$

★ Ejemplo 3:



Paso 1: como el Divisor tiene 2 bits (11), se toman igualmente 2 bits en el Dividendo (10):

- Como el $Dividendo < Divisor$, se toma otro bit del Dividendo (101)
- El $Dividendo > Divisor$ y lo contiene se coloca un **1** en el Cociente
- Se realiza el producto de: $1 \times 11 = 11$
- Se halla la diferencia entre: $101 - 11 = 010$

Paso 2:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '100'
- Como el nuevo $Dividendo > Divisor$ y lo contiene, se coloca un **1** en el Cociente
- Se realiza el producto de: $1 \times 11 = 11$
- Se halla la diferencia entre: $100 - 11 = 001$

Paso 3:

- Se baja el bit '1' del Dividendo y el nuevo Dividendo es '011'
- Como el nuevo $Dividendo > Divisor$ y lo contiene, se coloca un **1** en el Cociente
- Se realiza el producto de: $1 \times 11 = 11$
- Se halla la diferencia entre: $011 - 011 = 000$

Paso 4:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '000'
- Como el nuevo $Dividendo < Divisor$ y NO lo contiene, se coloca un **0** en el Cociente
- Como no hay más bits para bajar, la operación termina

Paso 5: los resultados del Cociente y el Residuo son:

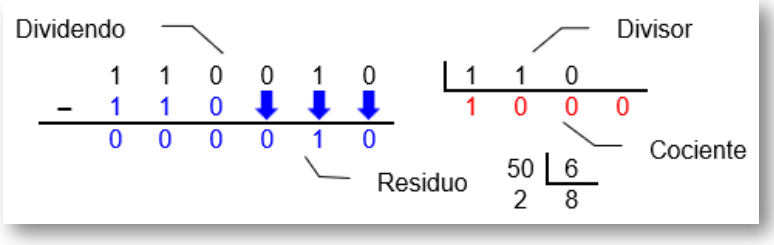
$$Cociente: \quad 1110_b = 14_d$$

$$Residuo: \quad 000_b = 0_d$$

Paso 6: verificación en el sistema numérico decimal:

$$42 \div 3 = 14_d$$

★ Ejemplo 4:



Paso 1: como el Divisor tiene 3 bits (110), se toman igualmente 3 bits en el Dividendo (110):

- El $Dividendo = Divisor$ y lo contiene se coloca un **1** en el Cociente
- Se realiza el producto de: $1 \times 110 = 110$
- Se halla la diferencia entre: $110 - 110 = 000$

Paso 2:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '000'
- Como el nuevo $Dividendo < Divisor$ y NO lo contiene, se coloca un **0** en el Cociente

Paso 3:

- Se baja el bit '1' del Dividendo y el nuevo Dividendo es '001'
- Como el nuevo $Dividendo < Divisor$ y NO lo contiene, se coloca un **0** en el Cociente

Paso 4:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '010'
- Como el nuevo $Dividendo < Divisor$ y NO lo contiene, se coloca un **0** en el Cociente
- Como no hay más bits para bajar, la operación termina

Paso 5: los resultados del Cociente y el Residuo son:

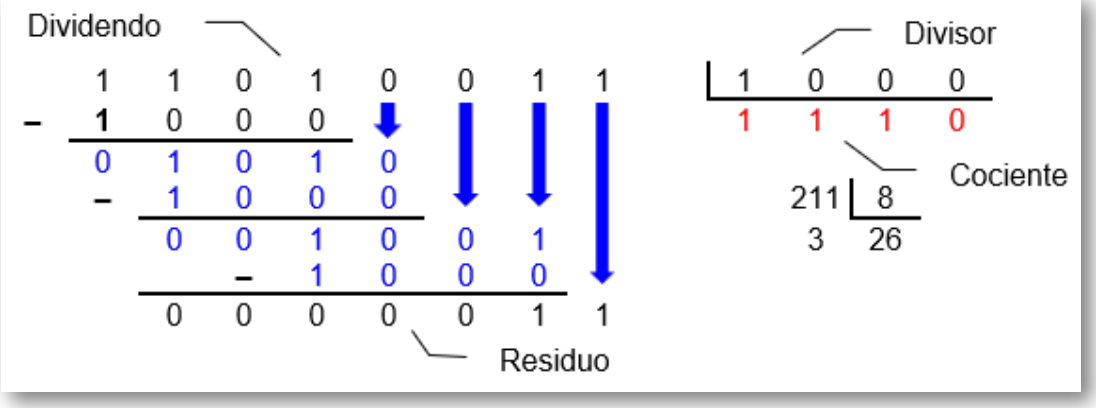
$$Cociente: 1000_b = 8_d$$

$$Residuo: 010_b = 2_d$$

Paso 6: verificación en el sistema numérico decimal:

$$50 \div 6 = 8_d \; y \; sobra \; 2_d$$

★ Ejemplo 5:



Paso 1: como el Divisor tiene 4 bits (1000), se toman igualmente 4 bits en el Dividendo (1101):

- El $Dividendo > Divisor$ y lo contiene se coloca un **1** en el Cociente
- Se realiza el producto de: $1 \times 1000 = 1000$
- Se halla la diferencia entre: $1101 - 1000 = 0101$

Paso 2:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '1010'
- Como el nuevo $Dividendo > Divisor$ y lo contiene, se coloca un **1** en el Cociente
- Se realiza el producto de: $1 \times 1000 = 1000$
- Se halla la diferencia entre: $1010 - 1000 = 0010$

Paso 3:

- Se baja el bit '0' del Dividendo y el nuevo Dividendo es '0100'
- Como el nuevo $Dividendo < Divisor$ y NO lo contiene, se coloca un **0** en el Cociente

Paso 4:

- Se baja el bit '1' del Dividendo y el nuevo Dividendo es '1001'
- Como el nuevo $Dividendo > Divisor$ y lo contiene, se coloca un **1** en el Cociente
- Se realiza el producto de: $1 \times 1000 = 1000$
- Se halla la diferencia entre: $1001 - 1000 = 0001$

Paso 5:

- Se baja el bit '1' del Dividendo y el nuevo Dividendo es '0011'
- Como el nuevo $Dividendo < Divisor$ y NO lo contiene, se coloca un $\mathbf{0}$ en el Cociente
- Como no hay más bits para bajar, la operación termina.

Paso 6: los resultados del Cociente y el Residuo son:

$$Cociente: \quad 11010_b = 26_d$$

$$Residuo: \quad 0011_b = 3_d$$

Paso 7: verificación en el sistema numérico decimal:

$$211 \div 8 = 26_d \; y \; sobra \; 3_d$$

3.5 Ejercicios propuestos con respuesta

Solucionar las siguientes Sumas Binarias y verificar sus resultados con el sistema numérico decimal:

1. $100111 + 11011_b$ ► ***R/. 1000010_b / 66_d***
2. $111011 + 10101_b$ ► ***R/. 1010000_b / 80_d***
3. $10111 + 1011 + 1101_b$ ► ***R/. 101111_b / 47_d***
4. $11100 + 1001 + 1100_b$ ► ***R/. 100_b / 49_d***
5. $11001 + 10100 + 11101_b$ ► ***R/. 1001010_b / 74_d***

Solucionar las siguientes Restas Binarias y verificar sus resultados con el sistema numérico decimal:

1. $10011 - 1011_b$ ► ***R/. 1000_b / 8_d***
2. $11010 - 1110_b$ ► ***R/. 1100_b / 12_d***
3. $10000 - 1111_b$ ► ***R/. 0001_b / 1_d***
4. $110101110 - 11101001_b$ ► ***R/. 11000101_b / 197_d***
5. $1001111 - 101010_b$ ► ***R/. 100101_b / 37_d***

Solucionar las siguientes Multiplicaciones Binarias y verificar sus resultados con el sistema numérico decimal:

1. $10010 \times 11_b$ ► ***R/. 110110_b / 54_d***
2. $1111 \times 101_b$ ► ***R/. 1001011_b / 75_d***
3. $11000 \times 110_b$ ► ***R/. 10010000_b / 144_d***
4. $11011 \times 100_b$ ► ***R/. 101101100100_b / 135_d***
5. $11110 \times 11_b$ ► ***R/. 1011010_b / 90_d***

Solucionar las siguientes Divisiones Binarias y verificar sus resultados con el sistema numérico decimal (► R/. Cociente y Residuo):

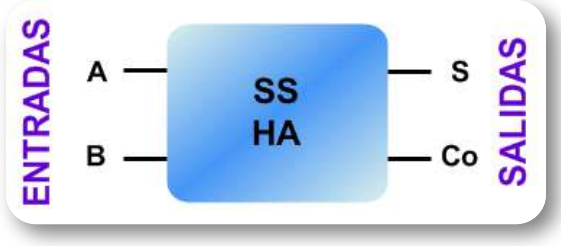
1. $111100 \div 100_b$ ► ***R/. 1111_b / 15_d***
2. $101100 \div 11_b$ ► ***R/. 1110 y 10_b / 14 y 2_d***
3. $1001101 \div 1000_b$ ► ***R/. 1001 y 101_b / 9 y 5_d***
4. $1101001 \div 1010_b$ ► ***R/. 11010 y 101_b / 10 y 5_d***
5. $100110010 \div 1100_b$ ► ***R/. 110001 y 110_b /25 y 6_d***

3.6 Circuitos Sumadores

Por medio de los circuitos lógicos combinacionales (compuertas lógicas) se pueden implementar circuitos que realicen sumas.

3.6.1 Semi-Sumador (SS)

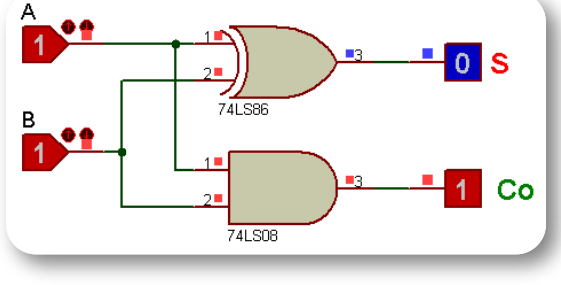
El circuito Semi Sumador ***(SS = Semi Sumador, HA = Half Adder)*** se utiliza para sumar dos bits de datos únicamente y su diagrama en bloques es:



En la tabla de verdad del ***Semi Sumador*** se observa que la salida (Σ) cumple con la tabla de verdad de la compuerta lógica ***XOR.***

Sumandos		Suma	Acarreo
A	B	Σ	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1
$A \oplus B$		Σ	Co

El plano esquemático del circuito semi sumador está compuesto por 2 compuertas lógicas: ***XOR*** y ***AND*** cada una con dos entradas para los sumandos ***A*** y ***B***; y la salida ***Σ*** para la Suma y ***Co*** para el acarreo (carry).



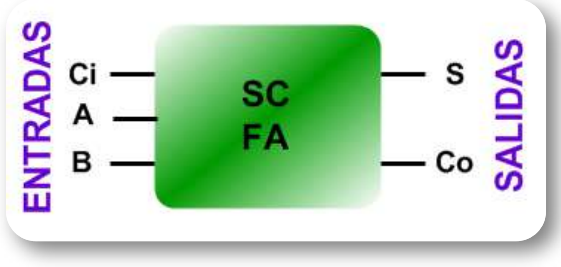
Las funciones lógicas del Semi Sumador son:

$$S = A \oplus B$$

$$C_o = A \times B$$

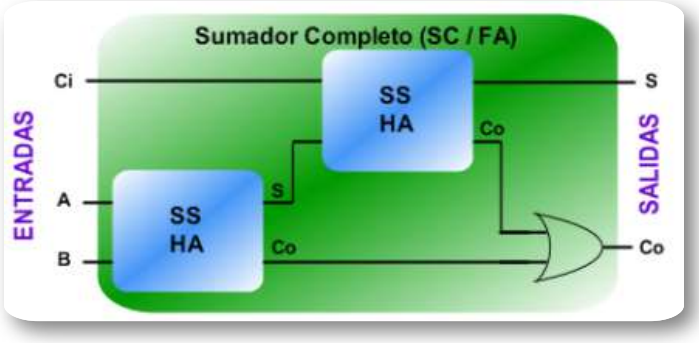
3.6.2 Sumador Completo (SC)

El circuito sumador completo ***(SC = Sumador Completo, FA = Full Adder)*** se utiliza para sumar 3 bits de datos y su diagrama en bloques es:



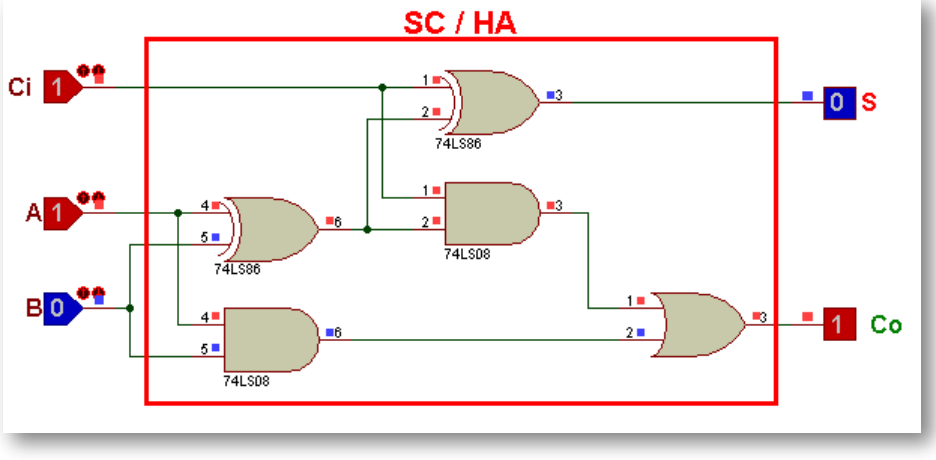
En la tabla de verdad se puede observar el comportamiento del circuito con un ***Acarreo de Entrada (Carry In = Ci)*** de '0' y de '1'.

Entradas			Salidas	
Sumandos		Acarreo Entrada	Suma	Acarreo Salida
A	B	Ci	Σ	Co
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
$A + B + C_i$			Suma	Acarreo



En el diagrama en bloques se observa que el ***Sumador Completo (SC)*** está conformado por 2 Semi Sumadores (SS) conectados en cascada.

El plano esquemático del Sumador Completo es:



Las funciones lógicas del Sumador Completo son:

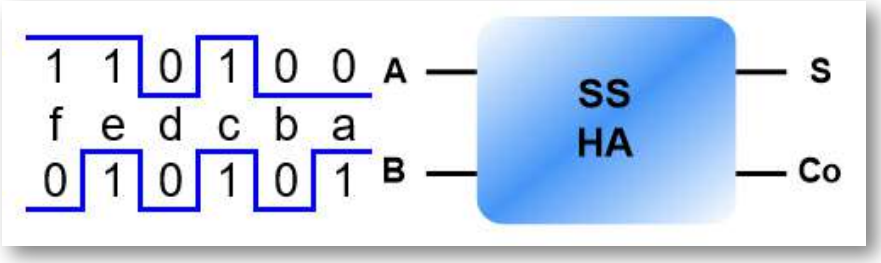
$$S = A \oplus B \oplus C_i$$

$$C_o = \{C_i \times (A \oplus B)\} + (A \times B)$$

3.6.3 Ejemplos

Hallar los estados lógicos en las salidas para cada señal en las entradas del Semi Sumador (SS), tabular toda la información y verificar la solución con un simulador electrónico:

★ Ejemplo 1:



Paso 1: hallar las salidas para cada pulso de entrada:

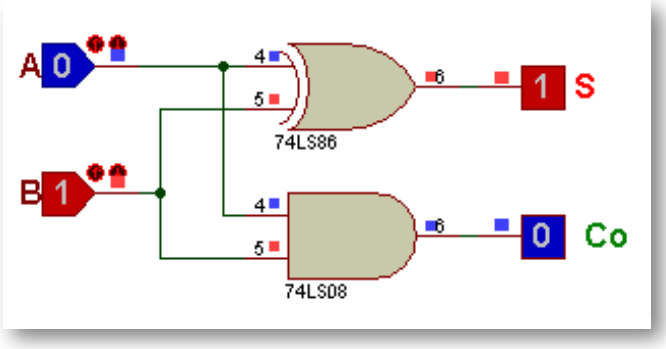
- Pulso a: entradas A = '0' y B = '1', las salidas suma = '1' y Co = '0'
- Pulso b: entradas A = '0' y B = '0', las salidas suma = '0' y Co = '0'
- Pulso c: entradas A = '1' y B = '1', las salidas suma = '0' y Co = '1'
- Pulso d: entradas A = '0' y B = '0', las salidas suma = '0' y Co = '0'
- Pulso e: entradas A = '1' y B = '1', las salidas suma = '0' y Co = '1'
- Pulso f: entradas A = '0' y B = '1', las salidas suma = '1' y Co = '0'

Paso 2: la tabulación de la información es:

- Tabla de verdad:

	Entradas		Salidas	
Pulso	B	A	Σ	Co
a	1	0	1	0
b	0	0	0	0
c	1	1	0	1
d	0	0	0	0
e	1	1	0	1
f	0	1	1	0

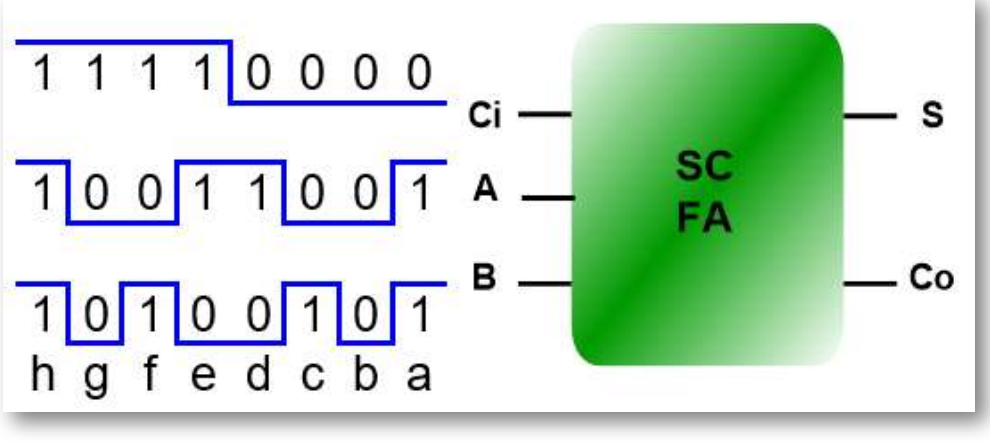
Una imagen de la simulación en Proteus:



Repositorio en [Github (Ejemplo-1-SS.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

Hallar los estados lógicos en las salidas para cada señal en las entradas del Sumador Completo (SC), tabular toda la información y verificar la solución con un simulador electrónico:



Paso 1: hallar las salidas para cada pulso de entrada:

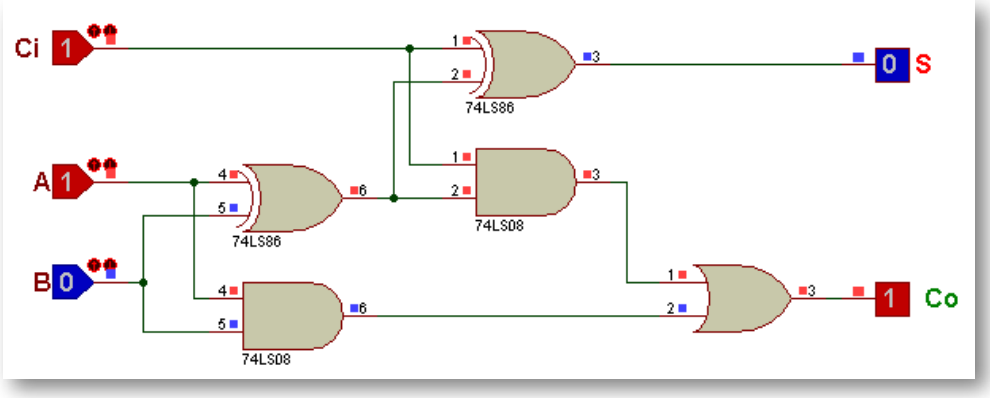
- Pulso a: entradas A = '1', B = '1' y Ci = '0', las salidas suma = '0' y Co = '1'
- Pulso b: entradas A = '0', B = '0' y Ci = '0', las salidas suma = '0' y Co = '0'
- Pulso c: entradas A = '0', B = '1' y Ci = '0', las salidas suma = '1' y Co = '0'
- Pulso d: entradas A = '1', B = '0' y Ci = '0', las salidas suma = '1' y Co = '0'
- Pulso e: entradas A = '1', B = '0' y Ci = '1', las salidas suma = '0' y Co = '1'
- Pulso f: entradas A = '0', B = '1' y Ci = '1', las salidas suma = '0' y Co = '1'
- Pulso g: entradas A = '0', B = '0' y Ci = '1', las salidas suma = '1' y Co = '0'
- Pulso h: entradas A = '1', B = '1' y Ci = '1', las salidas suma = '1' y Co = '1'

Paso 2: la tabulación de la información es:

- Tabla de verdad:

	Entradas			Salidas	
Pulso	Ci	B	A	Σ	Co
a	0	1	1	0	1
b	0	0	0	0	0
c	0	1	0	1	0
d	0	0	1	1	0
e	1	0	1	0	1
f	1	1	0	0	1
g	1	0	0	1	0
h	1	1	1	1	1

Una imagen de la simulación en Proteus:



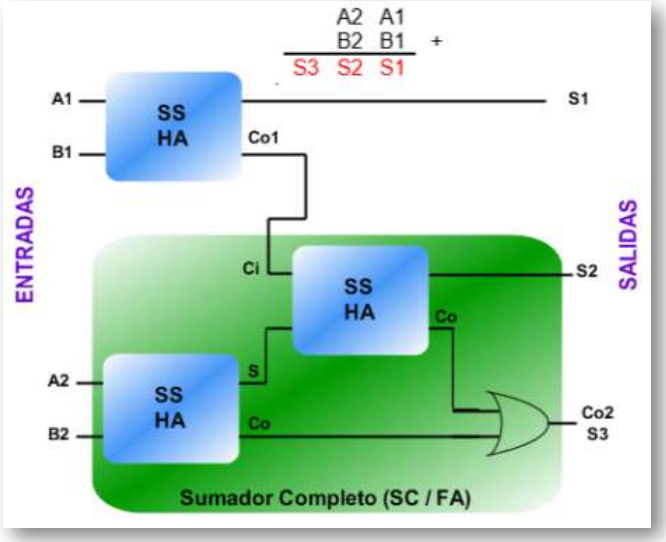
Repositorio en [Github (Ejemplo-2-SC.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

Diseñar el diagrama en bloques para un Sumador Completo de 2 sumandos cada uno de 2 bits (2x2); luego implementar el circuito en un software de simulación electrónica y verificar su funcionamiento.

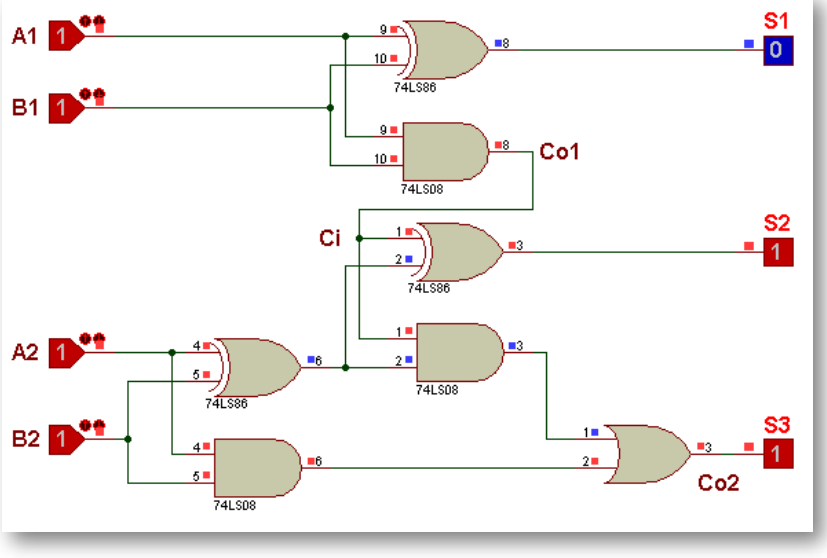
Paso 1: en el diagrama en bloques:

- El primer bloque es un Semi Sumador (SS) que realiza la operación entre A1 y B1 y genera la salida S1 y el acarreo Co1
- El segundo bloque es un Sumador Completo (SC), que recibe el dato del acarreo del bloque anterior (Co1) y los datos A2 y B2 de los sumandos, para generar las salidas S2 y el acarreo Co2 que hace las veces del tercer bit en la respuesta (S3)



Paso 2:

- Una imagen de la simulación en Proteus:
- La operación que se observa en la imagen es: $11 + 11 = 110_b$ que equivale a: $3 + 3 = 6_d$

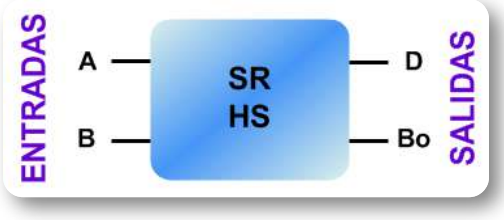


Repositorio en Github (Ejemplo-3-SC.dsn) con el archivo de la simulación con Proteus v7.8

3.7 Circuitos Restadores

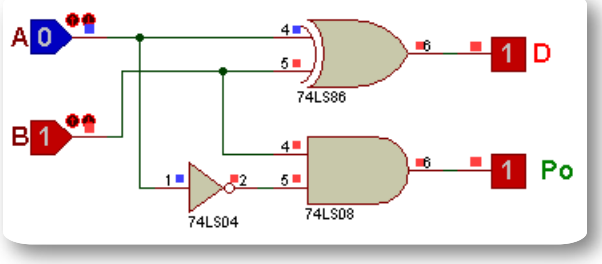
3.7.1 Semi-Restador (SR)

El circuito Semi Restador ***(SR = Semi Restador, HS = Half Substractor)*** se utiliza para restar dos bits de datos únicamente.



En la tabla de verdad del ***Semi Restador*** se observa que la salida (D) cumple con la tabla de verdad de la compuerta lógica ***XOR.***

Entradas		Salidas	
Minuendo	Sustraendo	Diferencia	Préstamo
A	B	D	Po
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0
$A - B$		D	Po



El plano esquemático del circuito semi restador está compuesto por 3 compuertas lógicas: ***XOR*** y ***AND*** cada una con dos entradas para los datos ***A*** y ***B***, y la ***NOT*** que se conecta al 'MINUENDO'; y las salidas ***D*** para la Diferencia y ***Bo*** para el préstamo (Borrow).

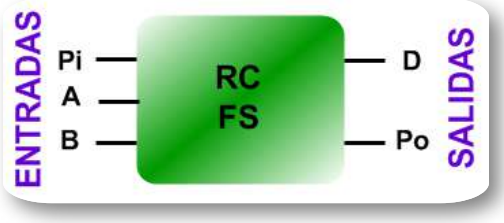
Las funciones lógicas del Semi Restador son:

$$D = A \oplus B$$

$$P_o = \overline{A} \times B$$

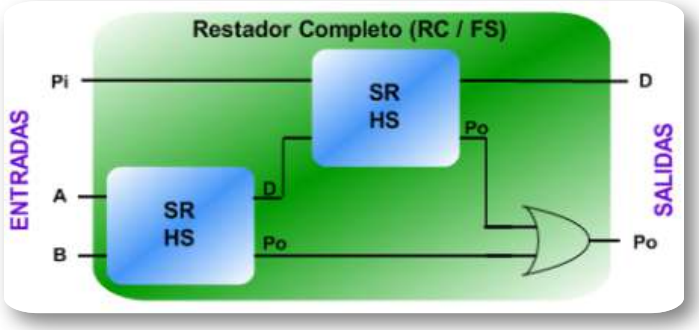
3.7.2 Restador Completo (RC)

El circuito restador completo ***(RC = Restador Completo, FS = Full Substractor)*** se utiliza para restar 3 bits de datos.



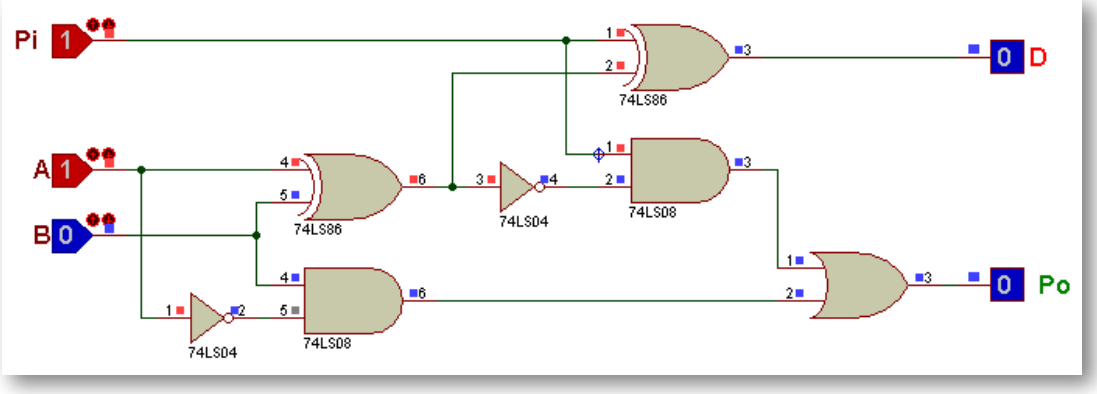
En la tabla de verdad del ***Restador Completo*** se observa que la salida (D) cumple con la tabla de verdad de la compuerta lógica ***XOR.***

Entradas			Salidas	
Minuendo	Sustraendo	Préstamo Entrada	Diferencia	Préstamo Salida
A	B	Pi	D	Po
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1
$A - B - C_i$			Diferencia	Préstamo



En el diagrama en bloques se observa que el ***Restador Completo (RC)*** está conformado por 2 Semi Restadores (SR) conectados en cascada.

El plano esquemático del Restador Completo es:



Las funciones lógicas del Restador Completo son:

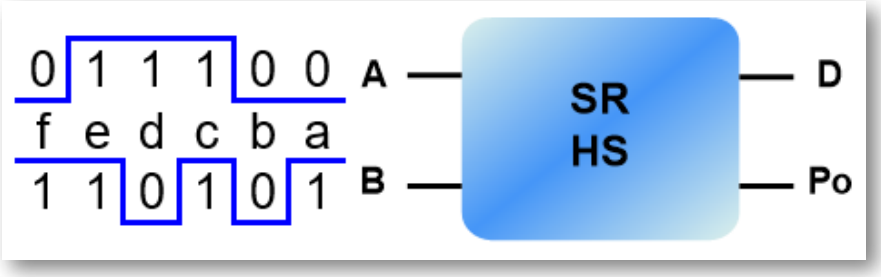
$$\boldsymbol{D = A \oplus B \oplus P_i}$$

$$\boldsymbol{P_o = \{P_i \times \overline{(A \oplus B)}\} + (\overline{A} \times B)}$$

3.7.3 Ejemplos

Hallar los estados lógicos en las salidas para cada señal en las entradas del Semi Restador (SR), tabular toda la información y verificar la solución con un simulador electrónico:

★ Ejemplo 1:



Paso 1: hallar las salidas para cada pulso de entrada:

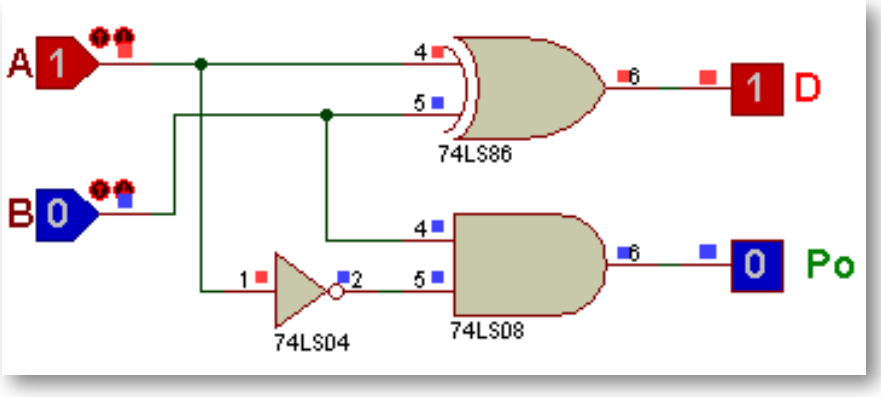
- Pulso a: entradas A = '0' y B = '1', las salidas diferencia = '1' y Po = '1'
- Pulso b: entradas A = '0' y B = '0', las salidas diferencia = '0' y Po = '0'
- Pulso c: entradas A = '1' y B = '1', las salidas diferencia = '0' y Po = '0'
- Pulso d: entradas A = '1' y B = '0', las salidas diferencia = '1' y Po = '0'
- Pulso e: entradas A = '1' y B = '1', las salidas diferencia = '0' y Po = '0'
- Pulso f: entradas A = '0' y B = '1', las salidas diferencia = '1' y Po = '1'

Paso 2: la tabulación de la información es:

- Tabla de verdad:

	Entradas		Salidas	
Pulso	Minuendo	Sustraendo	Diferencia	Préstamo
	A	B	D	Po
a	0	1	1	1
b	0	0	0	0
c	1	1	0	0
d	1	0	1	0
e	1	1	0	0
f	0	1	1	1

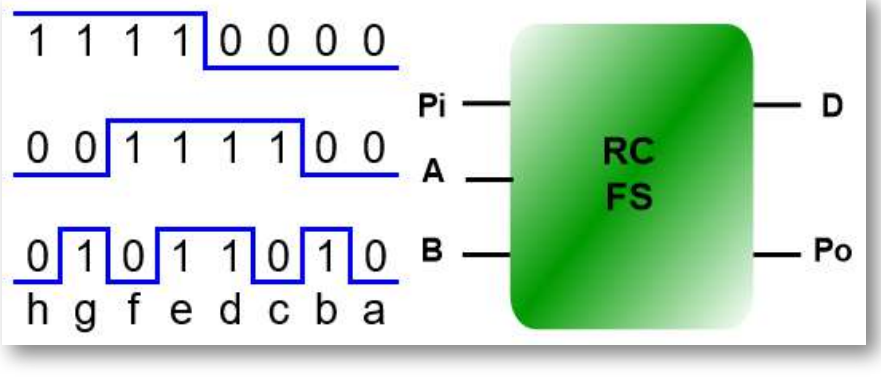
Una imagen de la simulación en Proteus:



Repositorio en [Github (Ejemplo-1-SR.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

Hallar los estados lógicos en las salidas para cada señal en las entradas del Restador Completo (RC), tabular toda la información y verificar la solución con un simulador electrónico:



Paso 1: hallar las salidas para cada pulso de entrada:

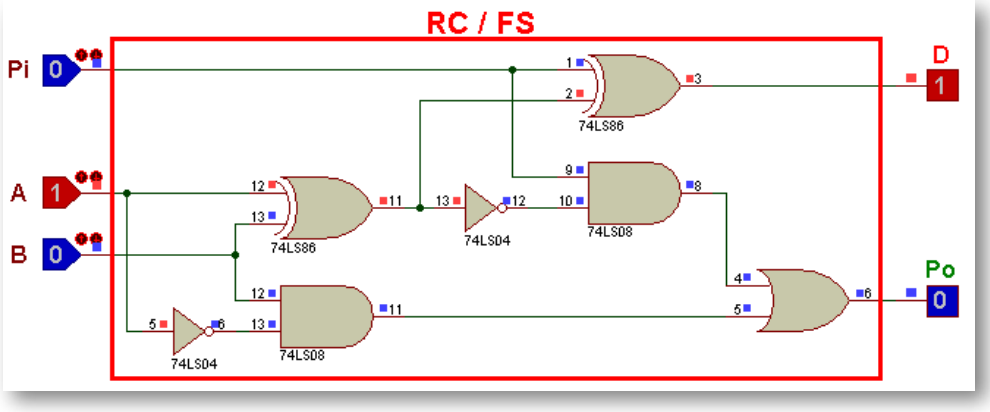
- Pulso a: entradas A = '0', B = '0' y Pi = '0', las salidas diferencia = '0' y Po = '0'
- Pulso b: entradas A = '0', B = '1' y Pi = '0', las salidas diferencia = '1' y Po = '1'
- Pulso c: entradas A = '1', B = '0' y Pi = '0', las salidas diferencia = '1' y Po = '0'
- Pulso d: entradas A = '1', B = '1' y Pi = '0', las salidas diferencia = '0' y Po = '0'
- Pulso e: entradas A = '1', B = '1' y Pi = '1', las salidas diferencia = '1' y Po = '1'
- Pulso f: entradas A = '1', B = '0' y Pi = '1', las salidas diferencia = '0' y Po = '0'
- Pulso g: entradas A = '0', B = '1' y Pi = '1', las salidas diferencia = '0' y Po = '1'
- Pulso h: entradas A = '0', B = '0' y Pi = '1', las salidas diferencia = '1' y Po = '1'

Paso 2: la tabulación de la información es:

- Tabla de verdad:

	Entradas			Salidas	
Pulso	Minuendo	Sustraendo	Préstamo Entrada	Diferencia	Préstamo Salida
	A	B	Pi	D	Po
a	0	0	0	0	0
b	0	1	0	1	1
c	1	0	0	1	0
d	1	1	0	0	0
e	1	1	1	1	1
f	1	0	1	0	0
g	0	1	1	0	1
h	0	0	1	1	1

Una imagen de la simulación en Proteus:



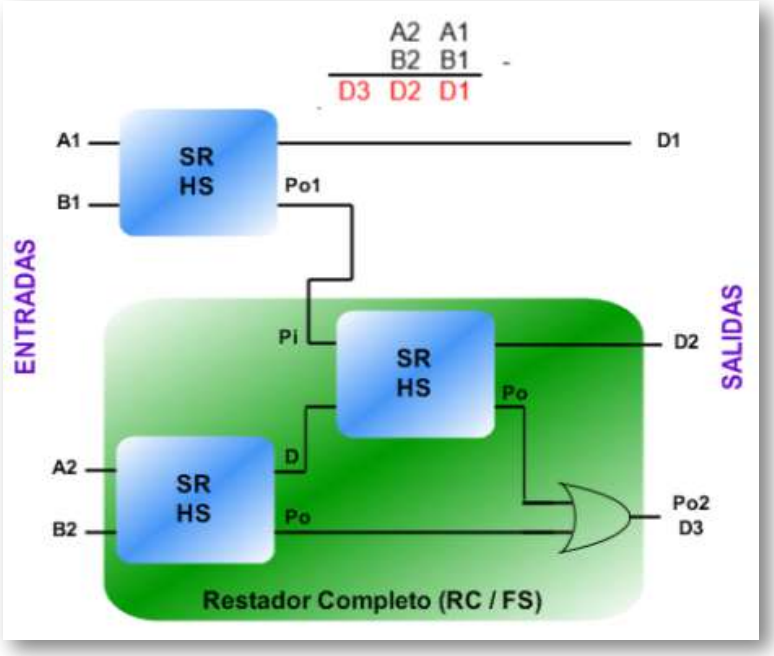
Repositorio en Github (Ejemplo-2-RC.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

Diseñar el diagrama en bloques para un Restador Completo de 2x2; luego implementar el circuito en un software de simulación electrónica y verificar su funcionamiento.

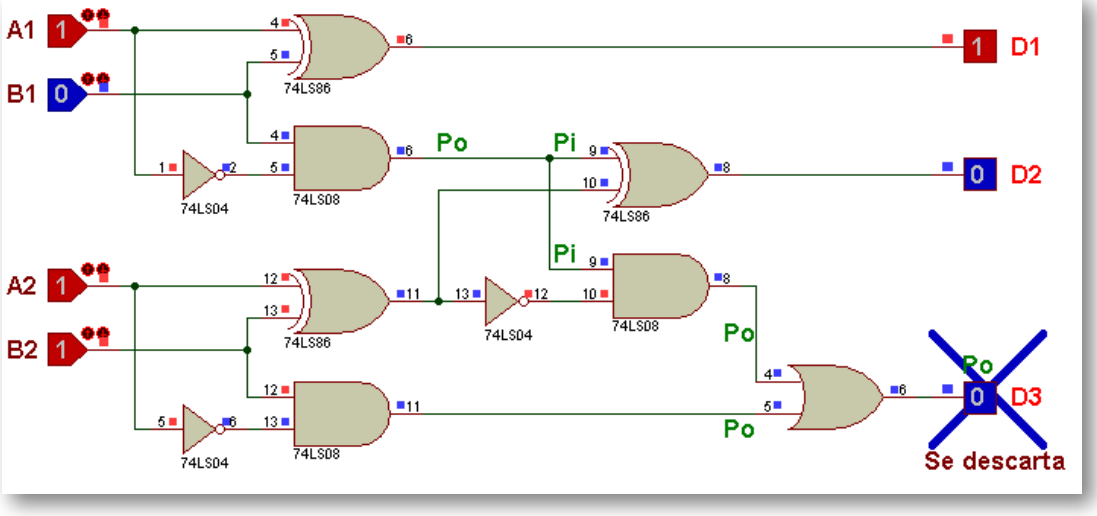
Paso 1: en el diagrama en bloques:

- El primer bloque es un Semi Restador (SR) que realiza la operación entre A1 y B1 y genera la salida D1 y el préstamo Po1.
- El segundo bloque es un Restador Completo (RC), que recibe el dato del préstamo del bloque anterior (Po1) y los datos A2 y B2, para generar las salidas D2 y el préstamo Po2 que hace las veces del tercer bit en la respuesta (D3), pero este dato se descarta.



Paso 2: una imagen de la simulación en Proteus:

- La operación que se observa en la imagen es: $11 - 10 = 01_b$ que equivale a: $3 - 2 = 1_d$



Repositorio en Github (Ejemplo-3a-RC.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 4:

Diseñar el diagrama en bloques para un Restador Completo de 2x2 basado en Sumadores Completos (SC); luego implementar el circuito en un software de simulación electrónica y verificar su funcionamiento.

Paso 1: obtener el Complemento 2:

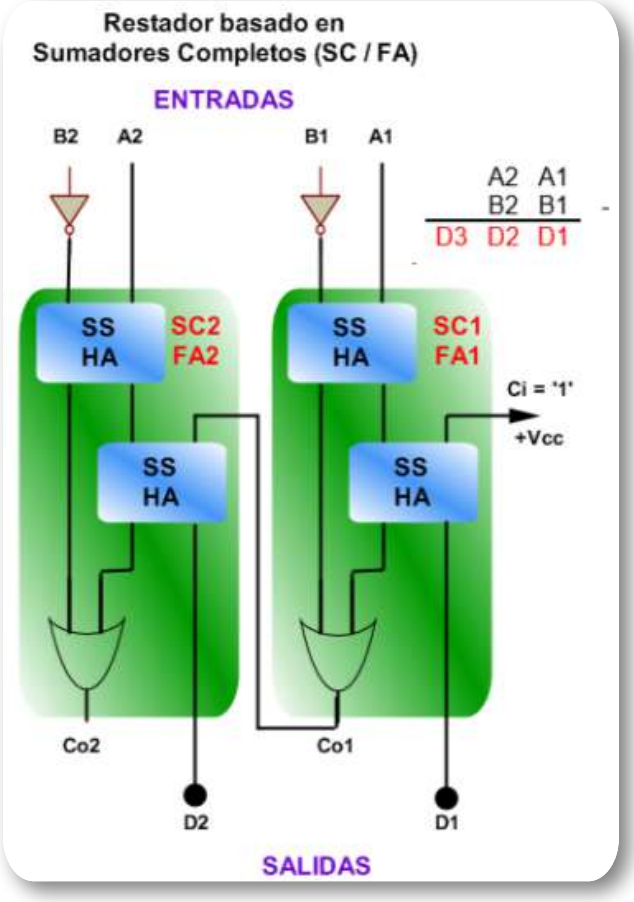
- Para realizar una **Resta** utilizando la **Suma**, es necesario hallar el **Complemento 2** al **Sustraendo** y una de las formas es obteniendo primero el **Complemento 1** y luego **sumarle '1'**.
- El **Complemento 1** se obtiene cambiando los '0' por '1' y viceversa. Ej: 10_b el Complemento 1 es: 01_b y luego, se le suma '1' para obtener el **Complemento 2** que será el nuevo Sumando para la operación.

Paso 2: implementación del circuito para el Complemento 2:

- Para implementar el Complemento 2 del Sustraendo se utiliza una compuerta **NOT** en la entrada del bit **"B"**
- Para sumarle '1' se requiere que el primer bloque sea un Sumador Completo (SC), que posee 3 entradas: 2 para los bits A1 y B1 y una tercera para **Ci** que sirve para adicionarle el '1' y de esa forma obtener el **Complemento 2** del Sustraendo.

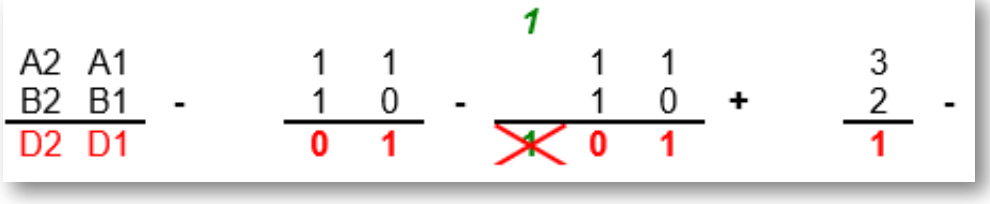
Paso 3: en el diagrama en bloques utilizando Sumadores Completos (SC):

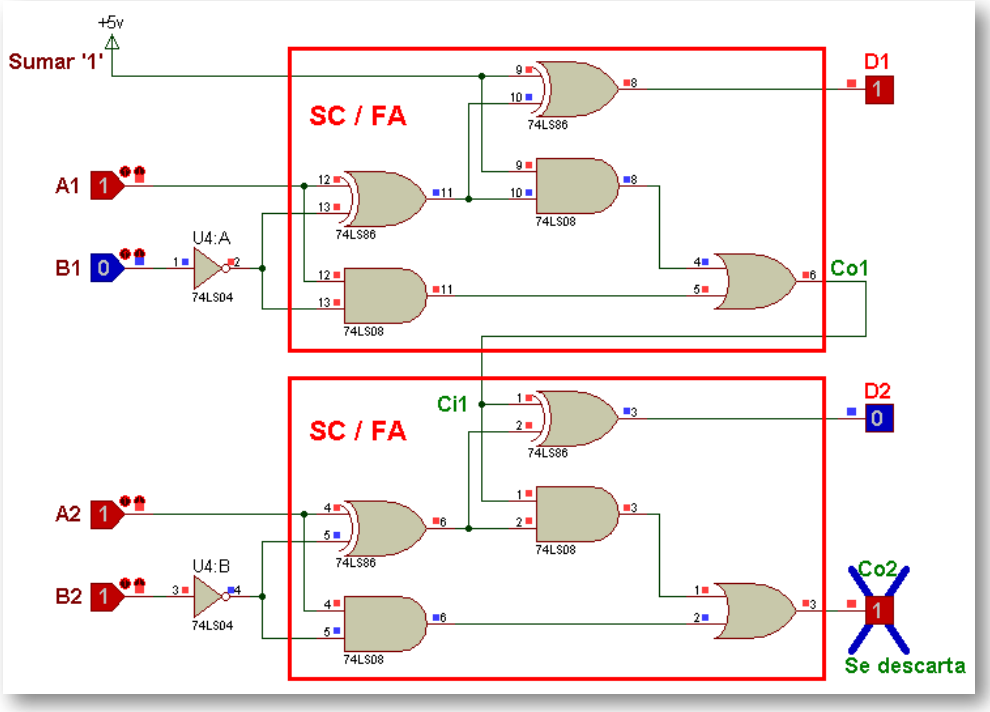
- Para operar 2 bits se requieren 2 Sumadores Completos (SC) conectados en cascada.
- El primer bloque realiza la operación: $A1 + B1 + 1$ y genera la salida D1 y el acarreo Co1 que se conectará al siguiente bloque Sumador.
- El segundo bloque recibe el dato del acarreo del bloque anterior (Co1) y adiciona los datos A2 y B2, para generar las salidas D2 y el acarreo Co2 que es descartado para el resultado.



Paso 4: imagen de la simulación en Proteus:

- La operación que se observa en la imagen es: $11 - 10 = 01_b$ que equivale a: $3 - 2 = 1_d$





Repositorio en [Github (Ejemplo-3b-RC.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 5:

Diseñar el diagrama en bloques para un Sumador y Restador Completo de 2x2; luego implementar el circuito en un software de simulación electrónica y verificar su funcionamiento.

Paso 1: obtener el Complemento 2:

- Para realizar una **Resta** utilizando la **Suma**, es necesario hallar el **Complemento 2** al **Sustraendo** y una de las formas es obteniendo primero el **Complemento 1** y luego **sumarle '1'**.

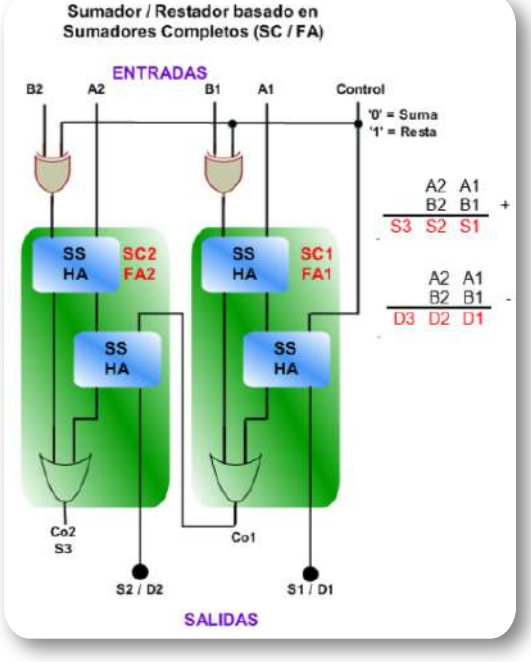
El **Complemento 1** se obtiene cambiando los '0' por '1' y viceversa. Ej: 10_b el Complemento 1 es: 01_b y luego, se le suma '1' para obtener el **Complemento 2** que será el nuevo Sumando para la operación.

Paso 2: implementación del circuito para el control de **"Suma"** y **"Resta"** (Complemento 2):

XOR				
Entradas		Salida		Operación
M	B	S		
0	0	0	NO cambia el dato de "B"	Suma
0	1	1		
1	0	1	INVIERTE el dato de "B"	Resta
1	1	0		

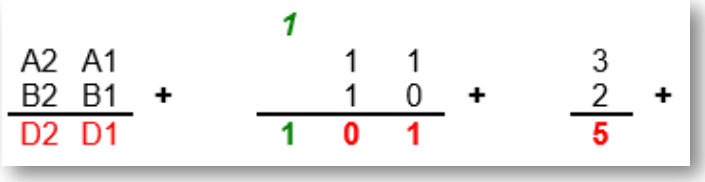
- Para realizar la **Suma** se requiere que el bit de **"B" no cambie** y para la **Resta que se invierta**
- Para implementar el Complemento 2 del Sustraendo se requiere **Negar** la entrada **"B"** y luego **Sumar '1'**
- Por medio de una compuerta **XOR** se controla la selección del tipo de operación:

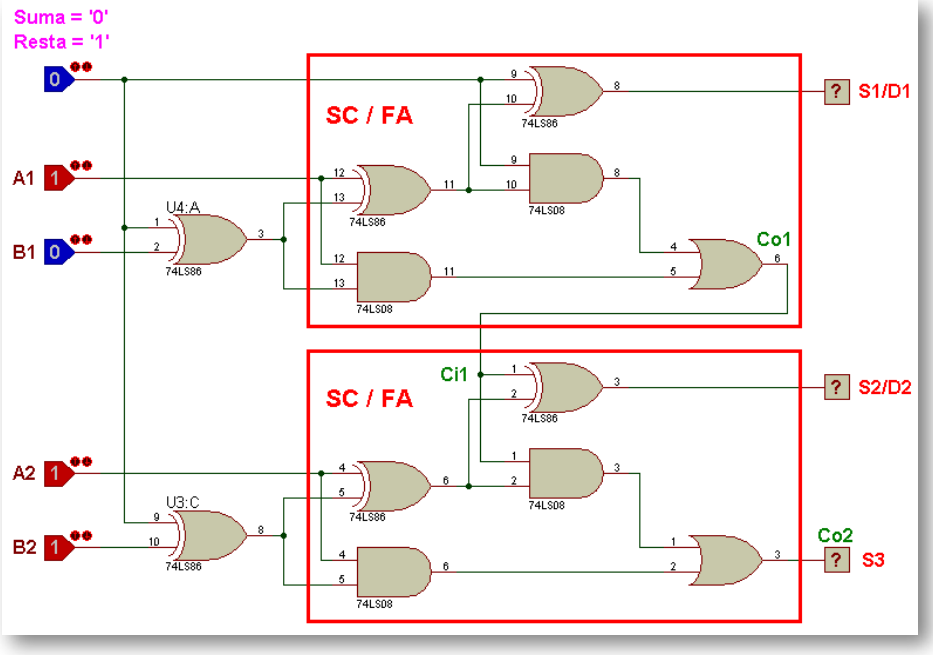
Paso 3: diagrama en bloques del sistema Suma / Resta utilizando Sumadores Completos (SC):



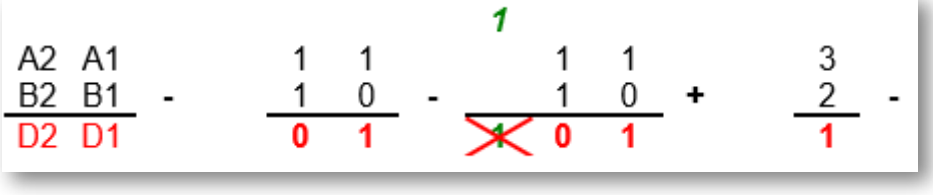
Paso 4: imagen de la simulación en Proteus:

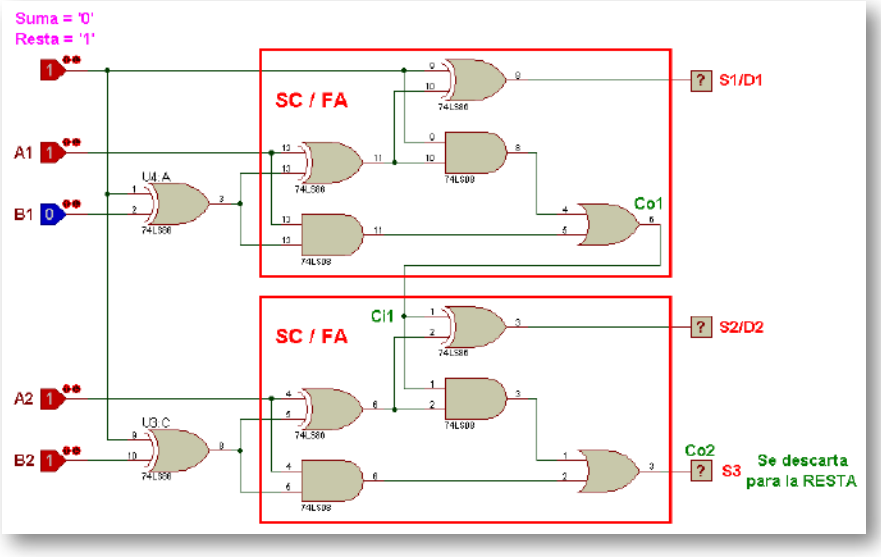
- La operación que se observa en la imagen es: $11 + 10 = 101_b$ que equivale a: $3 + 2 = 5_d$





- La operación que se observa en la imagen es: $11 - 10 = 01_b$ que equivale a: $3 - 2 = 1_d$



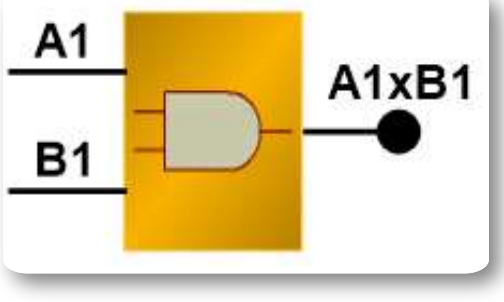


Repositorio en Github (Ejemplo-3c-RC.dsn) con el archivo de la simulación con Proteus v7.8

3.8 Multiplicador

La multiplicación de números binarios se realiza de forma similar a la de números decimales, multiplicando cada bit del ***MULTIPLICADOR*** por cada bit del ***MULTIPLICANDO*** y se utilizan las reglas de las operaciones binarias de la Multiplicación y la Suma.

El bloque **Multiplicador** de 2 bits está conformado por una compuerta **AND** de 2 entradas:

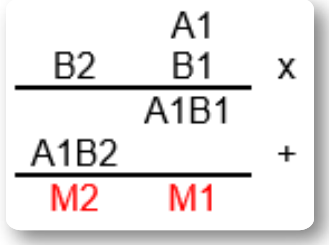


3.8.1 Ejemplos

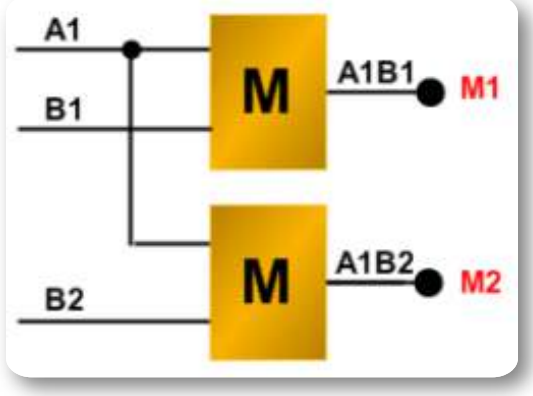
★ Ejemplo 1:

Diseñar el diagrama en bloques para un circuito Multiplicador de 1 bit para el multiplicando y 2 bits para el multiplicador, implementar el circuito combinacional y verificarlo con un software de simulación electrónica.

Paso 1: la estructura para la operación es:



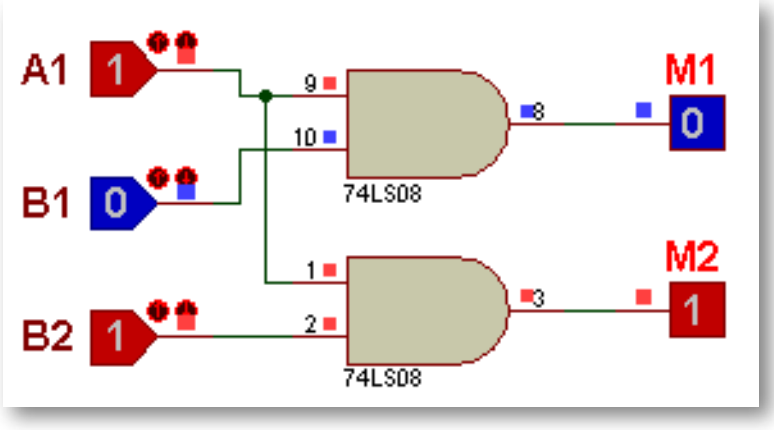
Paso 2: el diagrama en bloques:



- El primer bloque es una compuerta **AND** de 2 entradas que realiza las operaciones entre el Multiplicador B1 y el Multiplicando A1 y genera la respuesta **A1B1**.
- El segundo bloque es una compuerta **AND** de 2 entradas que realiza las operaciones entre el Multiplicador B2 y el Multiplicando A1 y genera la respuesta **A1B2**.
- El diagrama completo lo conforman 2 bloques Multiplicadores, donde sus respectivas respuestas son a la vez los resultados finales **M1** y **M2** de la operación.

Paso 3: la implementación en un software de simulación electrónica:

- Una imagen de la simulación en Proteus:



Repositorio en Github (Ejemplo-1-M.dsn) con el archivo de la simulación con Proteus v7.8

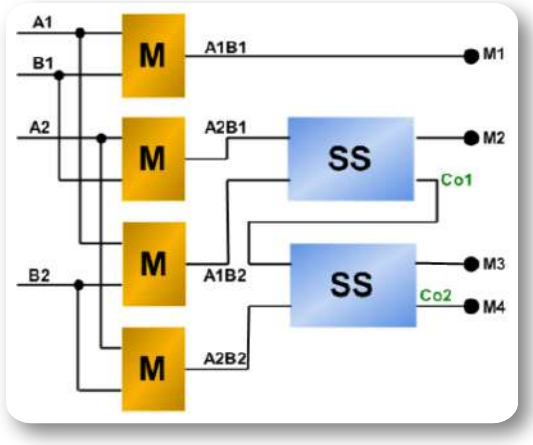
★ Ejemplo 2:

Diseñar el diagrama en bloques para un circuito Multiplicador de 2 bits para el multiplicando y 2 bits para el multiplicador, implementar el circuito combinacional y verificarlo con un software de simulación electrónica.

Paso 1: la estructura para la operación es:

		A2	A1	
		B2	B1	x
Co2	*Co1*	A2B1	A1B1	
	A2B2	A1B2		+
M4	M3	M2	M1	

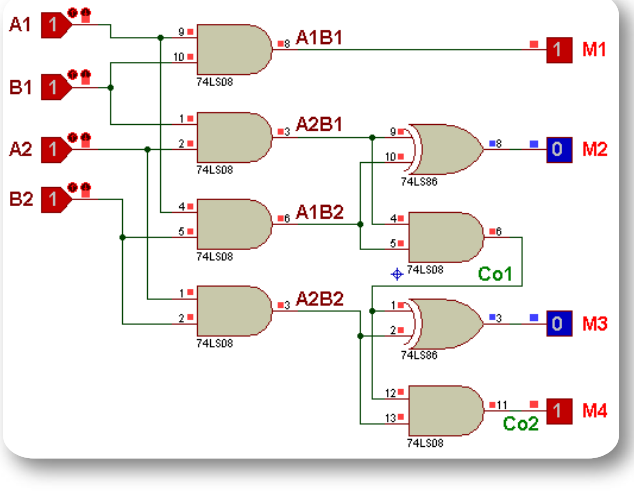
Paso 2: el diagrama en bloques:

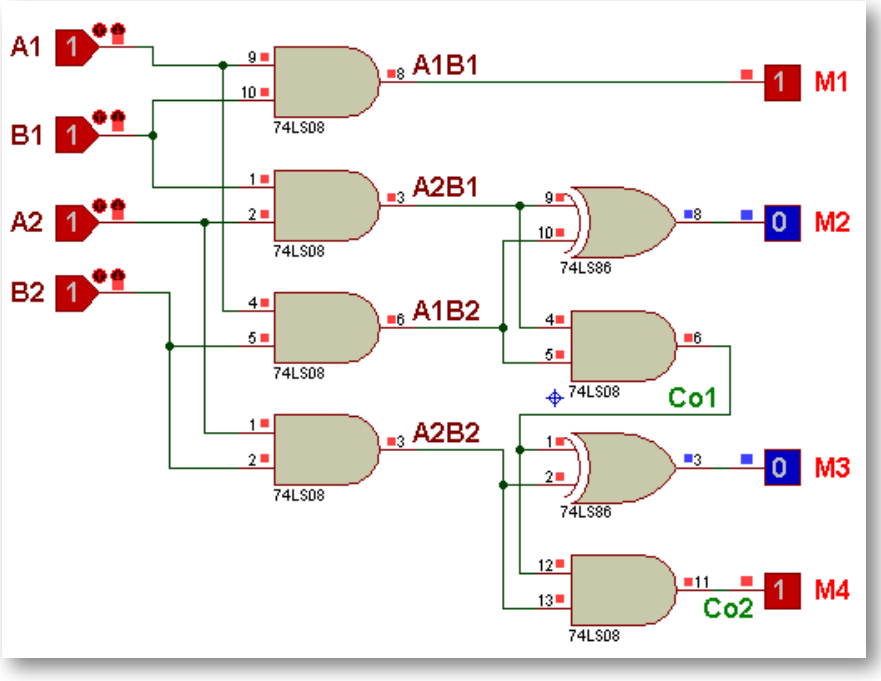


- Los 4 primeros bloques se componen cada uno de una compuerta AND de 2 entradas que realiza las operaciones entre los Multiplicadores y los Multiplicandos de A y B (A1B1, A2B1, A1B2 y A2B2).
- La salida del bloque multiplicador A1B1 equivale a la misma respuesta de salida **M1.**
- Un bloque Semi Sumador (SS) para los sumandos A2B1 y A1B2, que genera las respuestas **M2** y el acarreo **Co1.**
- Otro bloque Semi Sumador (SS) para los sumandos A2B2 y el **Co1**, que genera las respuestas **M3** y el acarreo **Co2** que equivale a **M3.**

Paso 3: la implementación en un software de simulación electrónica:

- Una imagen de la simulación en Proteus:





Repositorio en [Github (Ejemplo-2-M.dsn)] con el archivo de la simulación con Proteus v7.8

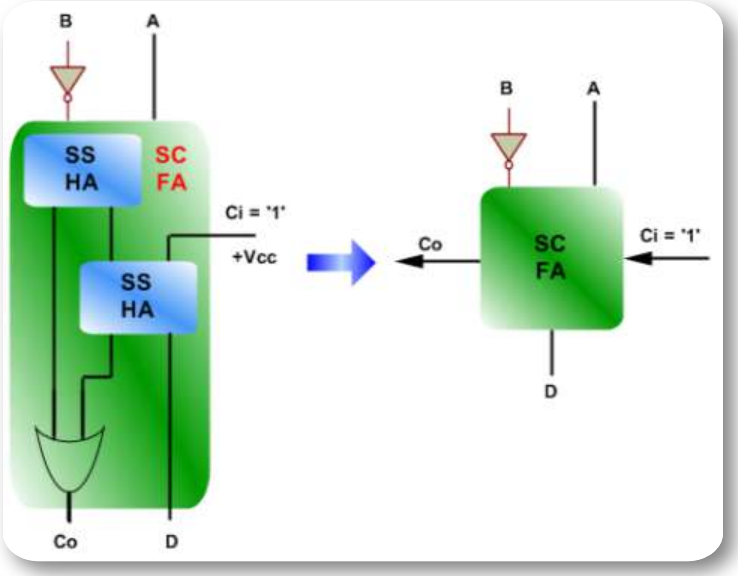
3.9 Divisor

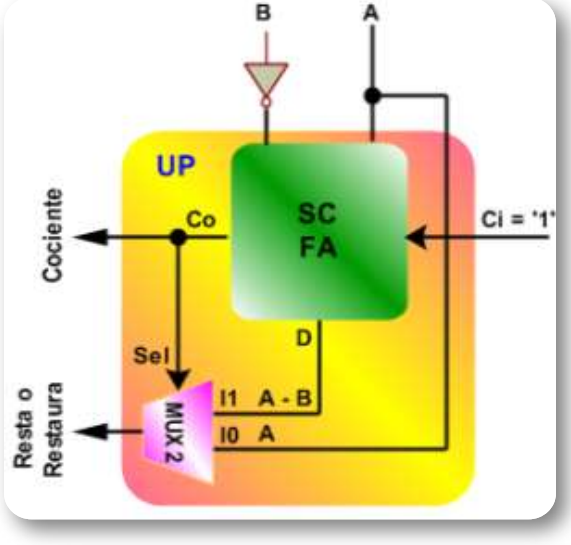
Para implementar el circuito Combinacional para ejecutar la División se requiere como base una **Unidad de Procesamiento (UP)** que realice el Complemento 2 y luego haga una comparación para saber si el resultado de la resta es positivo o negativo.

Unidad de Procesamiento (UP):

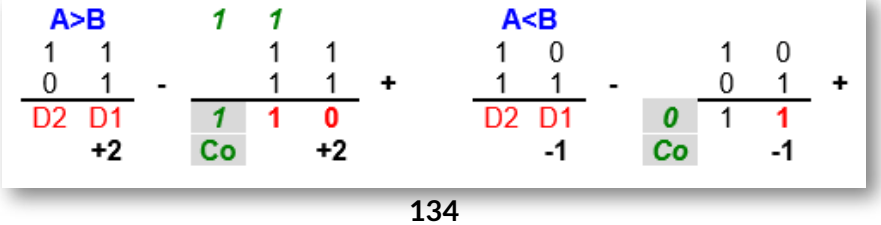
La operación de la Resta se realiza por el método del Complemento 2, donde primero se obtiene el Complemento 1 al Sustraendo y luego se le suma '1'.

Esta operación se implementa con un **Sumador Completo (SC)**, donde la entrada del bit **(B)** es **NEGADA** obteniendo así el Complemento 1; además, por la entrada **Ci** se ingresa el '**1**' que se le suma al Complemento 1 y de esta forma se tiene el Complemento 2.





La Unidad de Procesamiento se completa con un **Multiplexor de 2 canales (2x1)**, que selecciona entre el resultado de la **Resta (D)** o el bit de la entrada **A** por medio del Selector que es controlado por el **acarreo de salida (Co)**.



Si el **Acarreo (Co) es '1'**, indica que el resultado de la resta es positivo y la salida del **Multiplexor (MUX)** será la **diferencia (D)** entre **A y B (A-B)**.

Si el **Acarreo (Co) es '0'**, indica que el resultado de la resta es negativo y la salida del **Multiplexor (MUX)** será el **bit de A**.

Para desplazar a la derecha el valor del Divisor en las operaciones internas se lleva a cabo de acuerdo a la disposición que ocupa cada una de las UP dentro del sistema completo.

La cantidad de filas es igual a la cantidad de bits del **DIVIDENDO**.

Como el **DIVISOR** tiene 2 bits, las 2 primeras filas tienen 2 UP y el resto tendrán una UP más.

El **DIVIDENDO** tiene la misma cantidad de bits que el **COCIENTE** y el **DIVISOR** tiene la misma cantidad de bits que el **RESIDUO**.

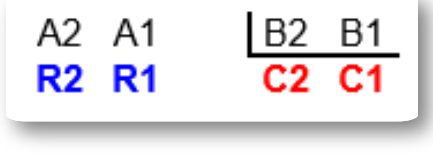
Si se divide por **1**, el valor del **Dividendo** es igual al del **Cociente** y el del **Divisor** es igual al del **Residuo**.

3.9.1 Ejemplos

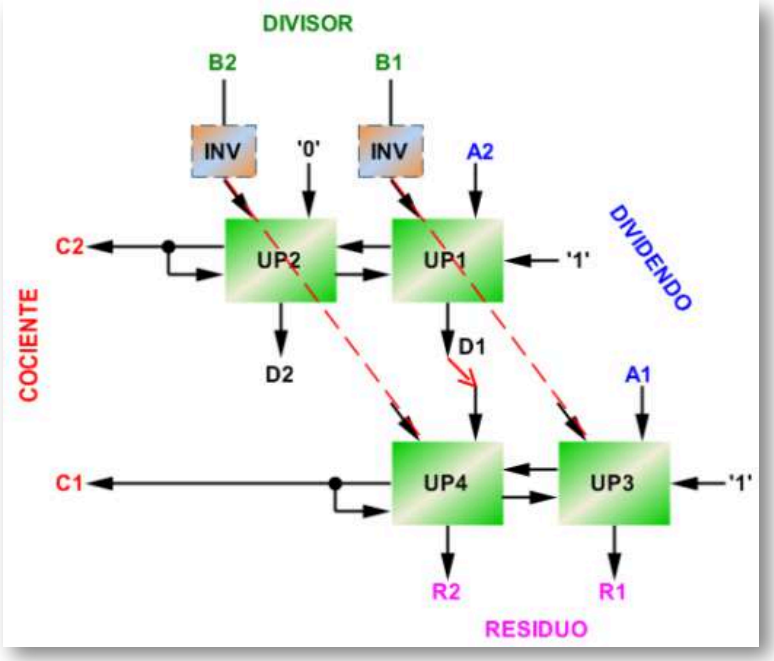
★ Ejemplo 1:

Diseñar el diagrama en bloques para un circuito Divisor de 2x2 bits, implementar el circuito combinacional y verificarlo con un software de simulación electrónica.

Paso 1: la estructura para la operación es:



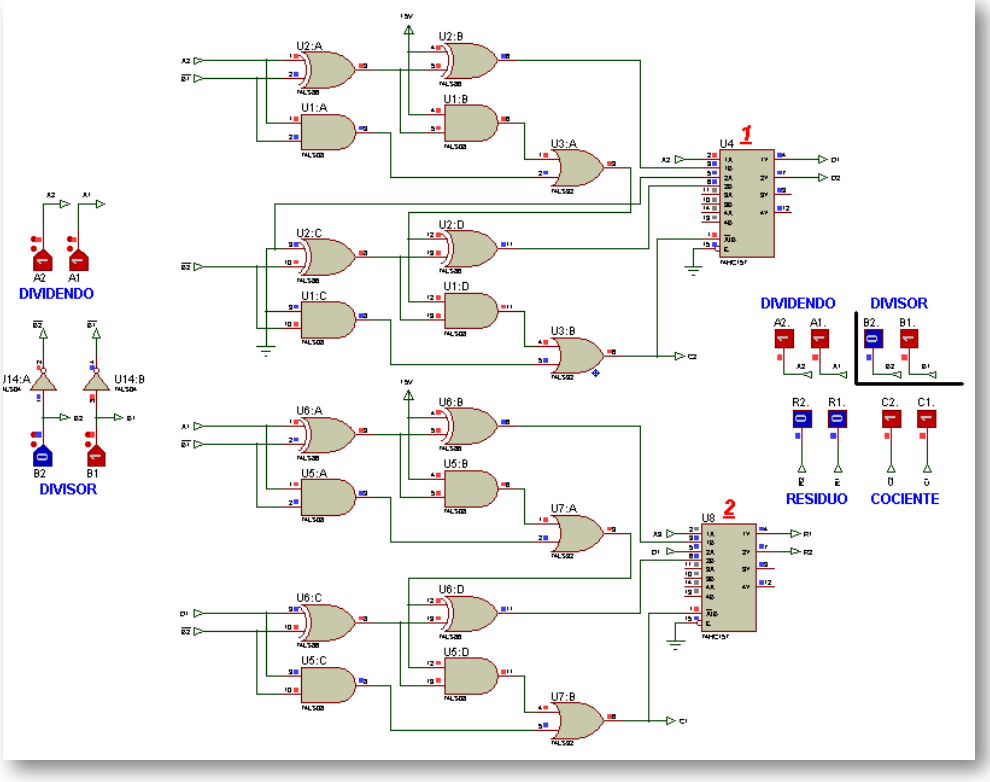
Paso 2: diagrama de bloques:



- El **DIVIDENDO** tiene 2 bits, entonces el sistema consta de 2 filas.
- El **DIVISOR** tiene 2 bit, o sea que las dos primeras filas contienen 2 UP y si tuviese más filas, las restantes tendrían una UP de más.
- Para la **UP2**, el bit para ingresar por la entrada **A** es '0' para completar los datos.
- El **DIVISOR** debe pasar a través de una compuerta **NOT** para obtener el Complemento 1.
- Para la primera UP de cada fila (la del extremo derecho, UP1, UP3), por la entrada **Ci** se ingresa el 1 para obtener su Complemento 2.
- El **DIVIDENDO** tiene la misma cantidad de bits que el **COCIENTE**.

- El **DIVISOR** tiene la misma cantidad de bits que el **RESIDUO**.
- Si se divide por 1, el valor del Dividendo es igual al del Cociente y el del Divisor es igual al del Residuo
- La línea discontinua de color rojo muestra la propagación y el desplazamiento del **DIVISOR** para cada comparación con el Dividendo.

Paso 3: la implementación en un software de simulación electrónica:

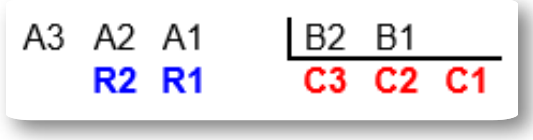


Repositorio en [Github (Divisor 2x2bits.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

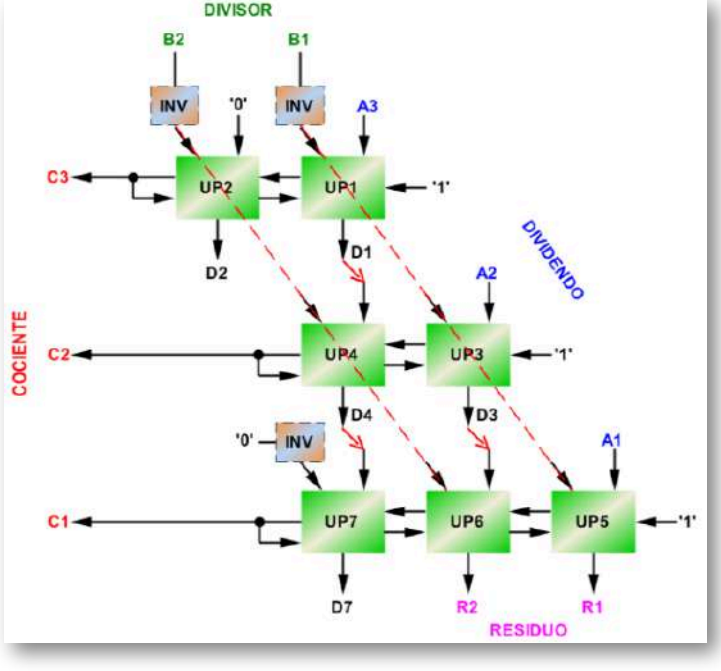
Diseñar el diagrama en bloques para un circuito Divisor de 3x2 bits, implementar el circuito combinacional y verificarlo con un software de simulación electrónica.

Paso 1: la estructura para la operación es:



Paso 2: diagrama de bloques:

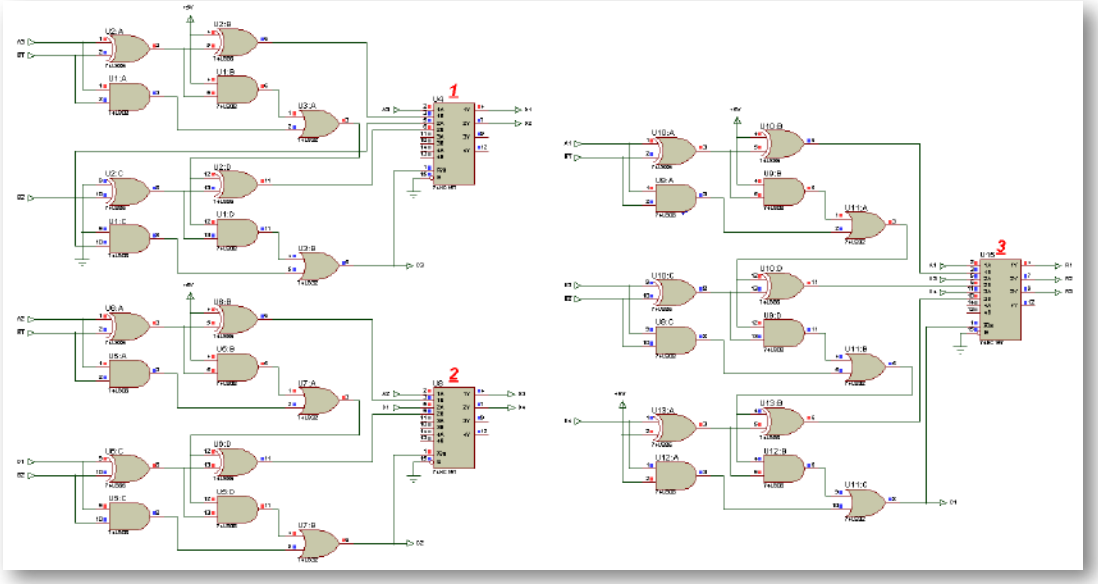
- El **DIVIDENDO** tiene 3 bits, por ello el sistema consta de 2 filas.
- El **DIVISOR** tiene 2 bit, o sea que las dos primeras filas contienen 2 UP y las restantes tendrían una UP de más (en este ejemplo la tercera fila).



- Para la **UP2**, el bit para ingresar por la entrada **A** es **'0'** para completar los datos y sumar '0' no afecta la operación.
- Para la **UP7**, el bit para ingresar por la entrada **B** es **'0'** y se hace por medio de una compuerta **NOT** para obtener el Complemento 1.
- El **DIVISOR** debe pasar a través de una compuerta **NOT** para obtener el Complemento 1.
- Para la primera UP de cada fila (la del extremo derecho, UP1, UP3, UP5), por la entrada **Ci** se ingresa el **1** para sumar y obtener su Complemento 2.
- El **DIVIDENDO** tiene la misma cantidad de bits que el **COCIENTE**.
- El **DIVISOR** tiene la misma cantidad de bits que el **RESIDUO**.
- Si se divide por 1, el valor del Dividendo es igual al del Cociente y el del Divisor es igual al del Residuo.
- La línea discontinua de color rojo muestra la propagación y el desplazamiento del **DIVISOR** para cada comparación interna en el proceso.

Paso 3: la implementación en un software de simulación electrónica:

Repositorio en Github (Divisor 3x2bits.dsn) con el archivo de la simulación con Proteus v7.8



3.10 Ejercicios propuestos con respuestas

★ Ejercicios 1 y 2:

Diseñar el diagrama en bloques para los Sumadores Completos de 2x3 (filas x columnas) y de 3x3; luego implementar los circuitos en un software de simulación electrónica y verificar sus respectivos funcionamientos.

Estructura de una suma 2x3:

	A3	A2	A1	
	B3	B2	B1	+
S4	S3	S2	S1	

Diagrama en bloques de la solución del ejercicio 1 (Sumador 2x3):

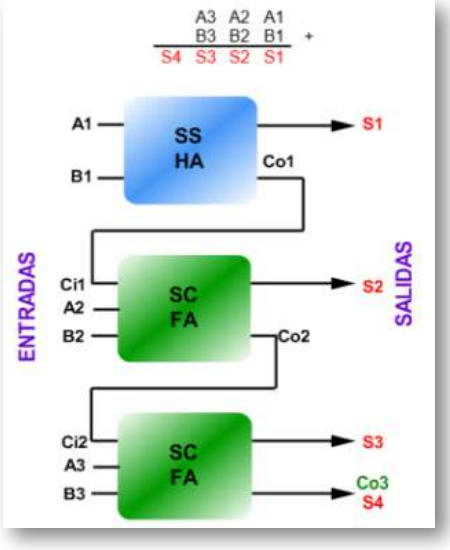
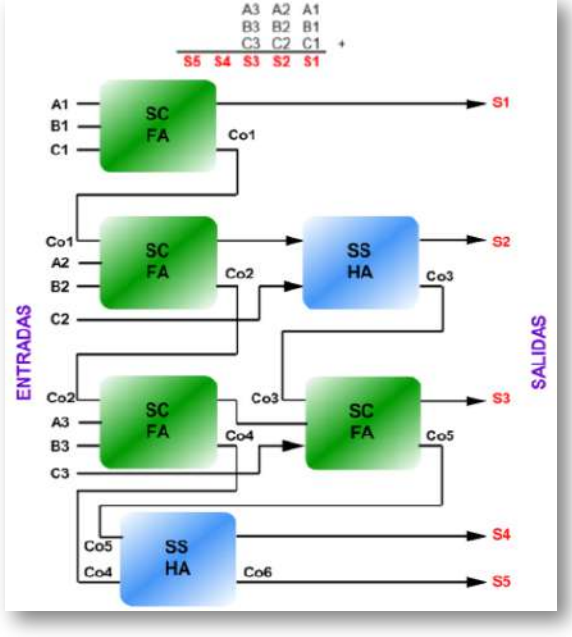


Diagrama en bloques de la solución del ejercicio 2 (Sumador 3x3):



★ Ejercicio 3:

Diseñar los diagramas en bloques para un Restador Completo de 2x3 (filas x columnas), utilizando Semi Restadores (SR) y Restadores Completos (RC) y el otro diagrama utilizando en método del Complemento 2. Implementar los circuitos combinacionales y verificarlos con un software de simulación electrónica.

Diagrama en bloques de la solución del ejercicio 3 (Restador 3x3) con Semi Restador (SR) y Restador Completo (RC):

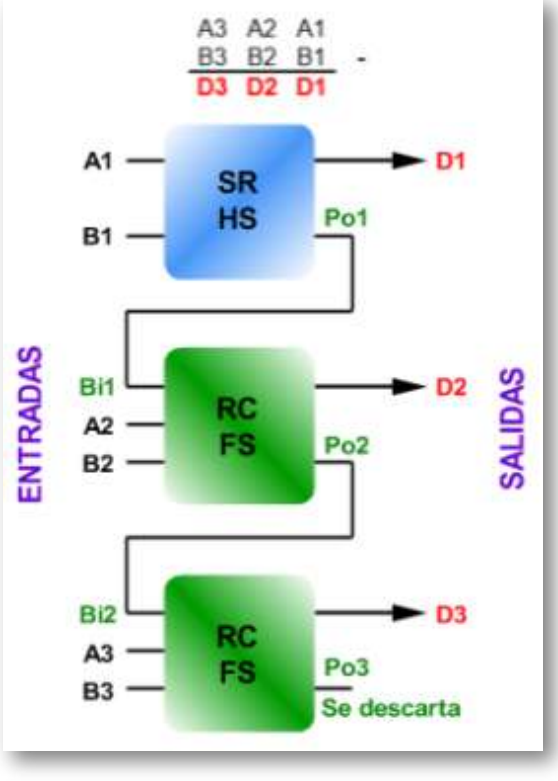
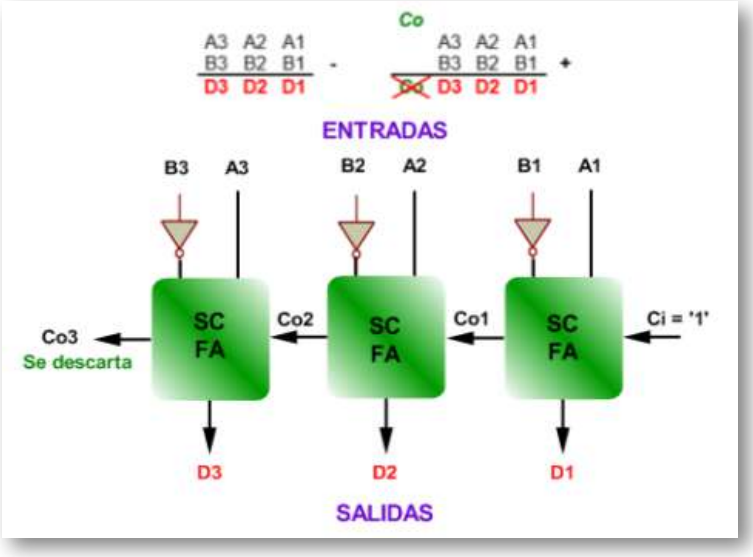


Diagrama en bloques de la solución del ejercicio 3 (Restador 3x3) con Sumadores Completos (SC), implementado por el método del Complemento 2:

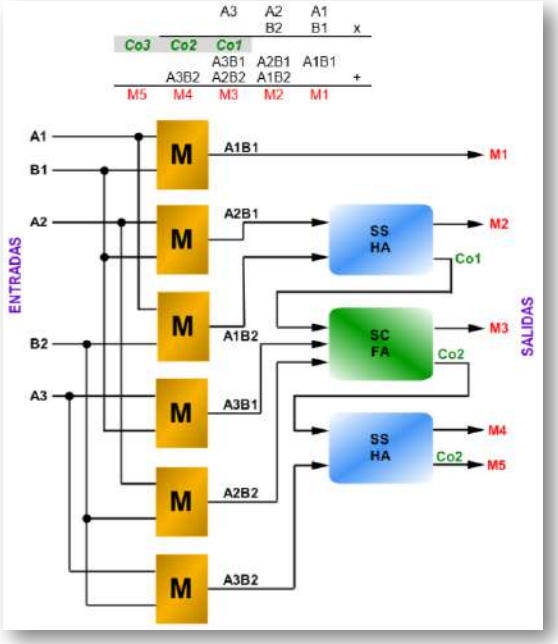


★ Ejercicios 4 y 5:

Diseñar los diagramas en bloques para los circuitos Multiplicadores de 3 bits para el Multiplicando x 2 bits para el Multiplicador, y de 3 bits para el Multiplicando x 3 bits para el Multiplicador. Implementar los circuitos combinacionales y verificarlos con un software de simulación electrónica.

Diagrama en bloques para la solución del ejercicio 4 de 3 bits para el Multiplicando y 2 bits para el Multiplicador:

Diagrama en bloques para la solución del ejercicio 5 de 3 bits para el Multiplicando y 3 bits para el Multiplicador:


A3 A2 A1
B2 B1 x
Co3 Co2 Co1
A3B1 A2B1 A1B1
A3B2 A2B2 A1B2 +
M5 M4 M3 M2 M1
A1
B1
A2
B2
A3
ENTRADAS
M
A1B1
A2B1
A1B2
A3B1
A2B2
A3B2
SS HA
SC FA
Co1
Co2
Co2
M1
M2
M3
M4
M5
SALIDAS

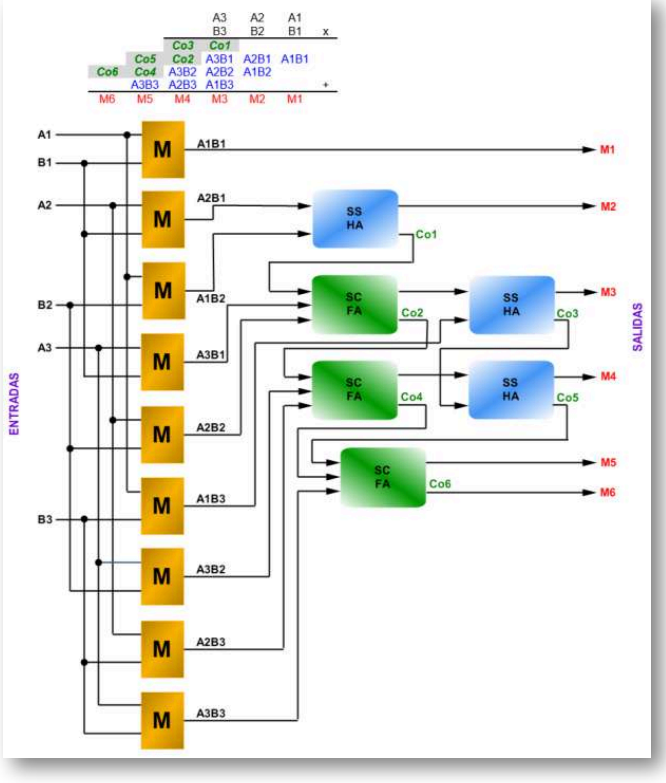
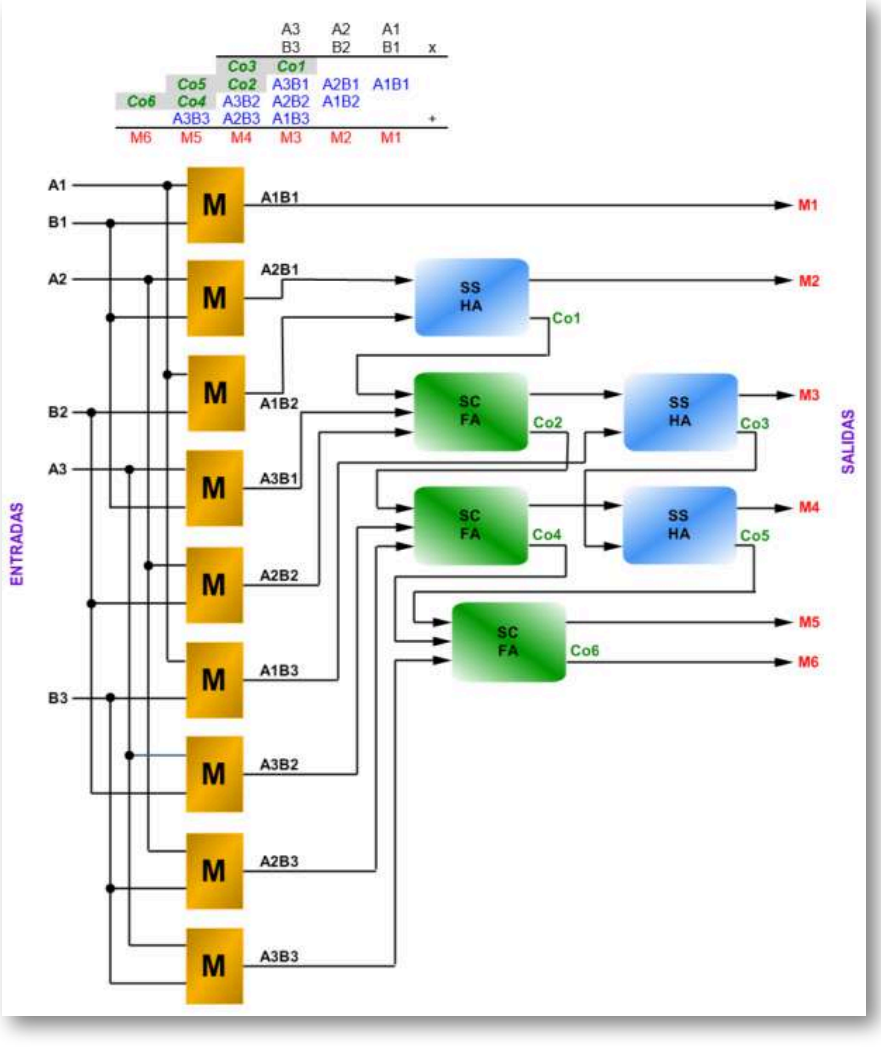

A3 A2 A1
B3 B2 B1 x
Co3 Co1
Co5 Co2 A3B1 A2B1 A1B1
Co6 Co4 A3B2 A2B2 A1B2
A3B3 A2B3 A1B3 +
M6 M5 M4 M3 M2 M1
A1
B1
A2
B2
A3
B3
ENTRADAS
M
A1B1
A2B1
A1B2
A3B1
A2B2
A1B3
A3B2
A2B3
A3B3
SS HA
SC FA
Co1
Co2
Co3
Co4
Co5
Co6
M1
M2
M3
M4
M5
M6
SALIDAS

Diagrama en bloques para la solución del ejercicio 5 de 3 bits para el Multiplicando y 3 bits para el Multiplicador:



★ Ejercicios 6:

Diseñar el diagrama en bloques para un circuito Divisor de 4x2 bits, implementar el circuito combinacional y verificarlo con un software de simulación electrónica.

La estructura de la operación es:

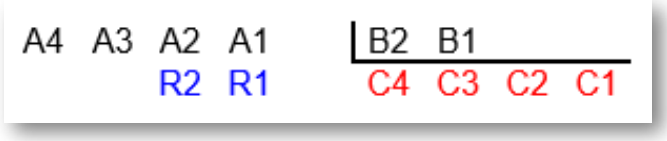
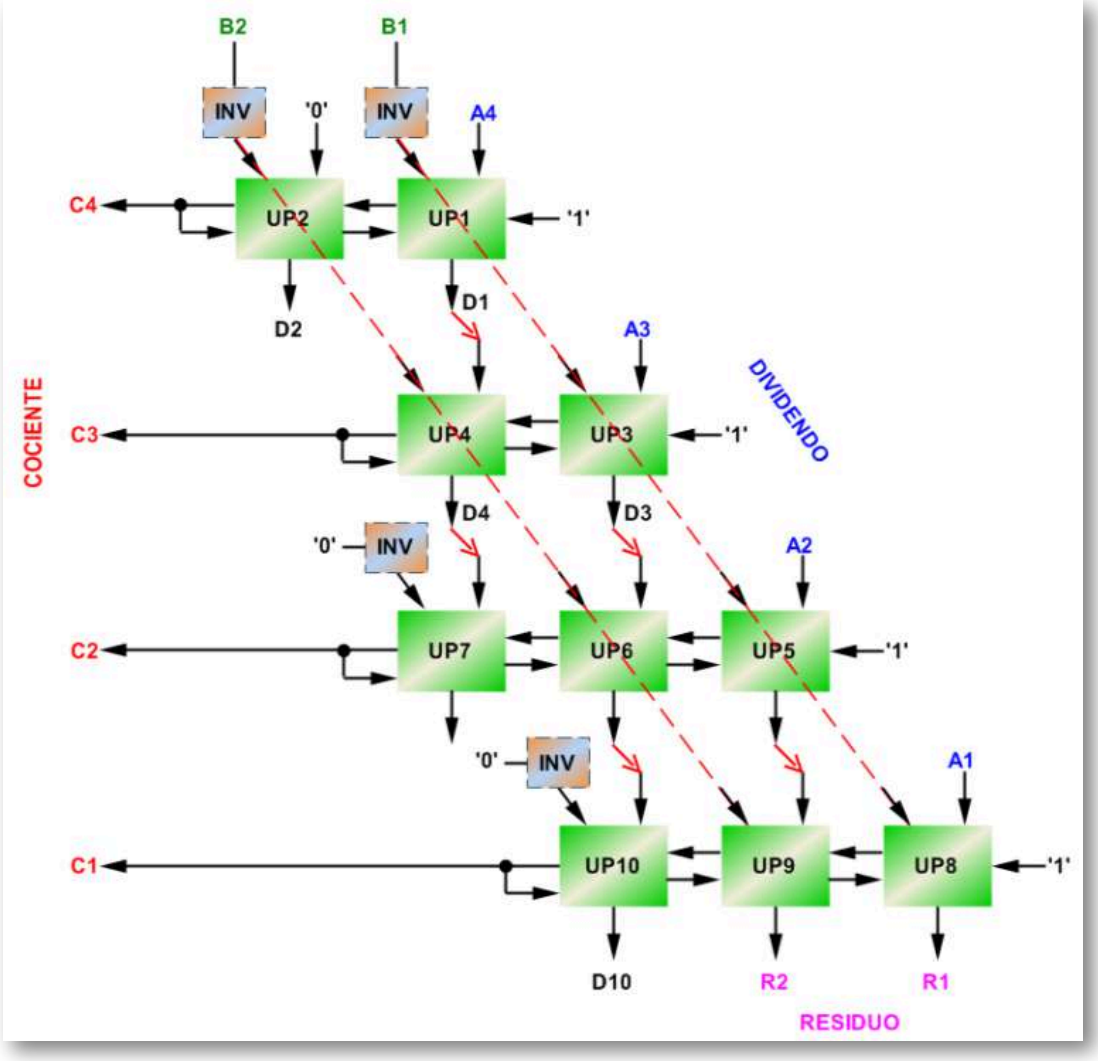


Diagrama en bloques para la solución del ejercicio 6 de 4x2 bits:



AYUDAS

COMPLEMENTOS

3.11 Videos

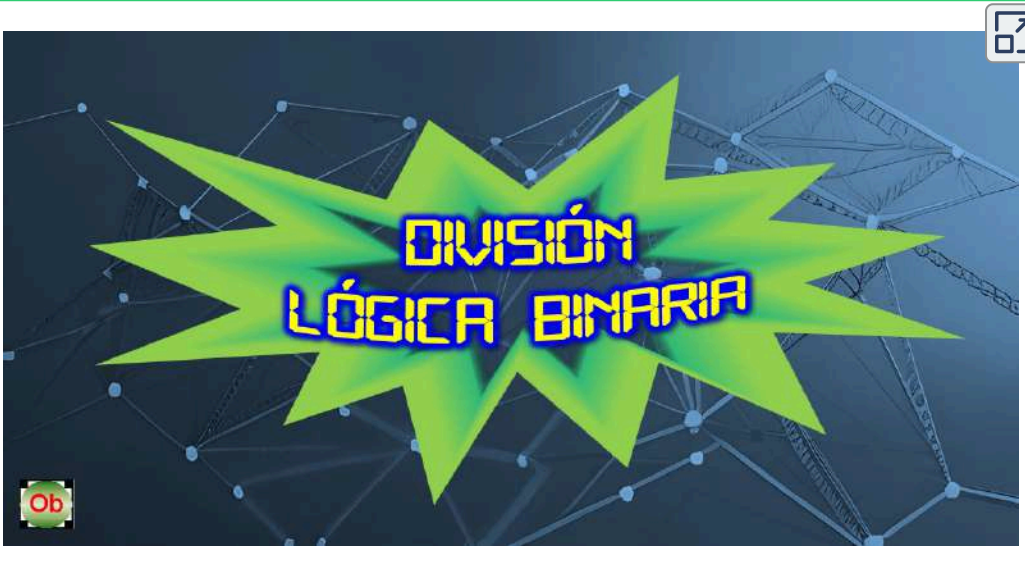
Haga clic en la esquina superior derecha para expandir el video.

Vídeos sobre solución de operaciones básicas en sistema Binario:[9]

[9] Videos realizados por Marisol Maldonado Olmos *(Pasos por Ingeniería)* SUMA BINARIA - Ejercicio #1 y RESTA BINARIA - Ejercicio #1

Haga clic en la esquina superior derecha para expandir el video.

Vídeos sobre solución de operaciones básicas en sistema Binario:[10]



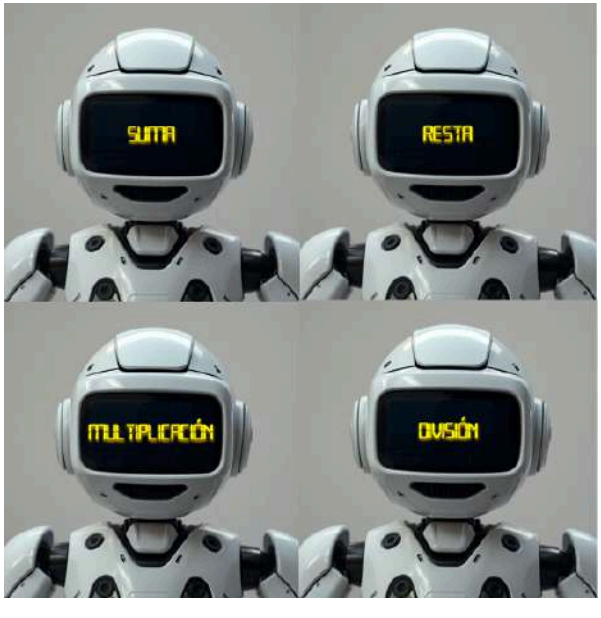
[10] Videos realizados por Marisol Maldonado Olmos *(Pasos por Ingeniería)* MULTIPLICACIÓN BINARIA - Ejercicio #1 y DIVISIÓN BINARIA - Ejercicio #1

Videos demostrativos creados por Oscar Ignacio Botero Henao sobre la solución del circuito combinacional del Sumador, Restador y Multiplicador Binario, la implementación y sus respectivas simulaciones.

Video explicativo: Sumador Binario, Circuitos Combinacionales - Solución y Simulación, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo: Restador Binario, Circuitos Combinacionales - Solución y Simulación, en el canal de youtube de Oscar Botero @oscarboterohenao

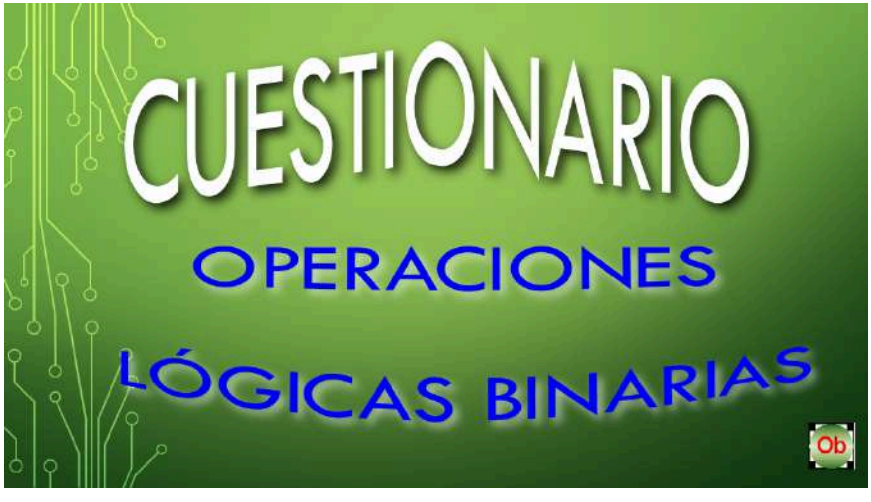
Video explicativo: Multiplicador Binario, Circuitos Combinacionales - Solución y Simulación, en el canal de youtube de Oscar Botero @oscarboterohenao



3.12 Objetos Interactivos

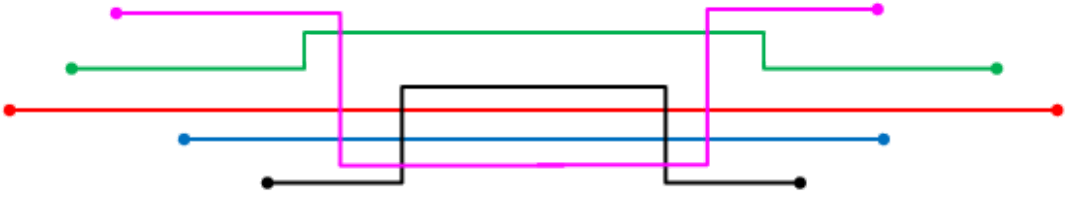
Haga clic en la esquina superior derecha para expandir el video.

Los objetos interactivos: el **Cuestionario sobre operaciones lógicas binarias** y la **Calculadora para las operaciones básicas binarias**, fueron generados con la *asistencia* de la Inteligencia Artificial (IA) en el sitio web: https://websim.ai/



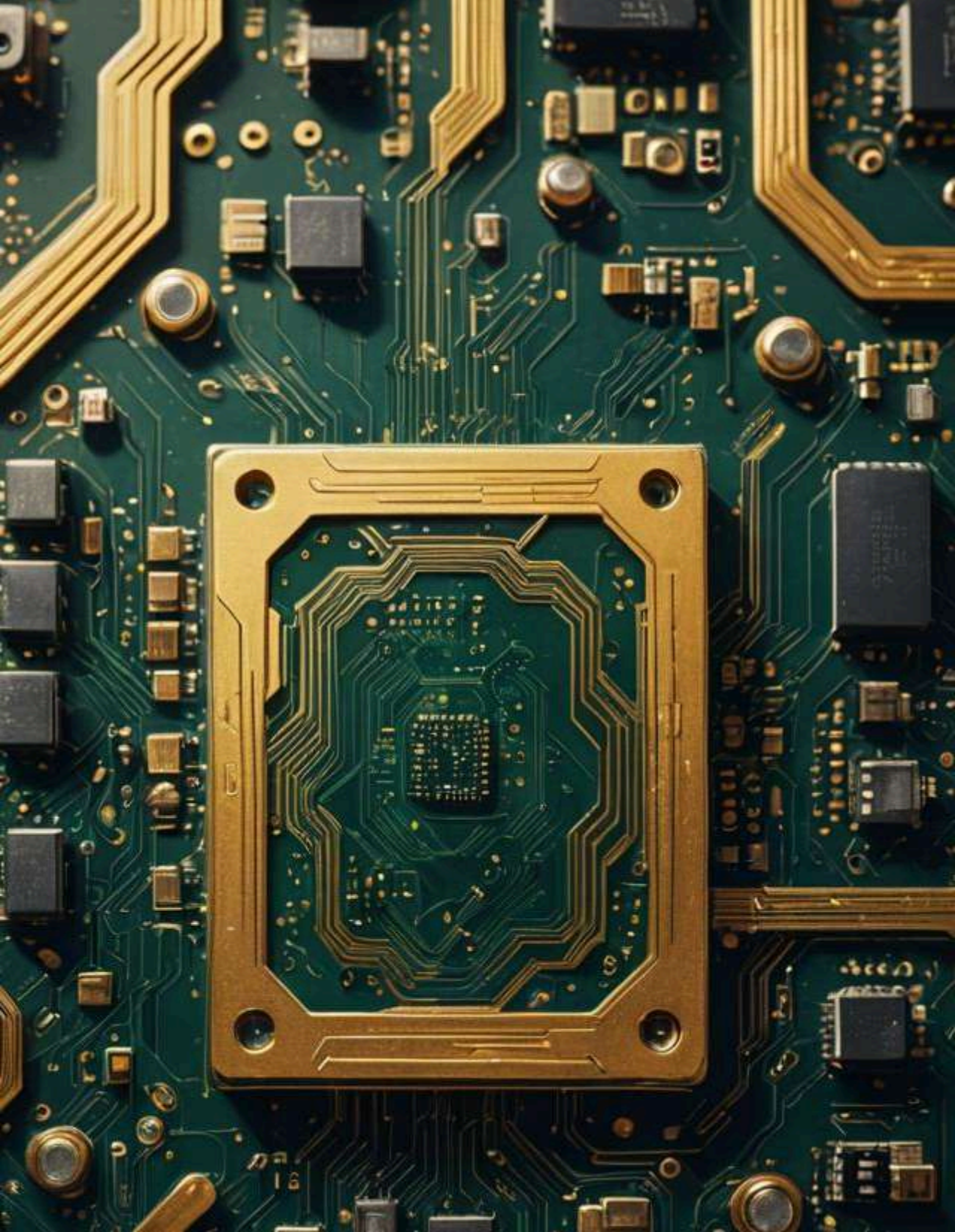
3.13 Simulaciones

Repositorio en Github con los archivos de las simulaciones de los ejemplos de este capítulo con Proteus v7.8



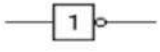
Capítulo IV

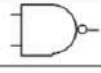
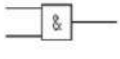
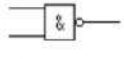
Funciones Lógicas

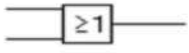
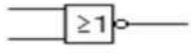


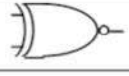
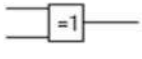
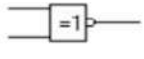
4.1 Compuertas Lógicas

Funciones Booleanas, símbolos, tablas de verdad y referencias comerciales y del software Proteus (clic en la imagen para aumentar) [1]

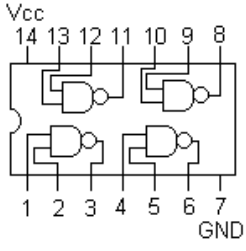
YES	NOT
$S = A$	$S = \bar{A}$
	
	
La salida es "1" cuando la entrada es "1"	La salida es "0" cuando la entrada es "1"
A | S 0 | 0 1 | 1	A | S 0 | 1 1 | 0
Proteus: 4050	Proteus: NOT 74LS04

AND	NAND
$S = A \times B$	$S = \overline{A \times B}$
	
	
La salida es "1" cuando TODAS las entradas son "1"	La salida es "0" cuando TODAS las entradas son "1"
B A | S 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	B A | S 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
Proteus: AND / AND_2 74LS08	Proteus: NAND / NAND_2 74LS00

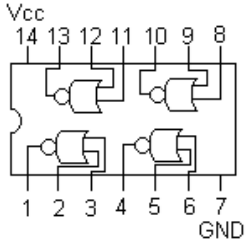
OR	NOR
$S = A + B$	$S = \overline{A + B}$
	
	
La salida es "1" cuando ALGUNA entrada es "1"	La salida es "0" cuando ALGUNA entrada es "1"
B A S 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **1**	B A S 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **0**
Proteus: OR 74LS32	Proteus: NOR / NOR_2 74LS02

XOR	XNOR
$S = A \oplus B$ $S = (A \times \bar{B}) + (\bar{A} \times B)$	$S = \overline{A \oplus B}$ $S = (A \times B) + (\bar{A} \times \bar{B})$
	
	
La salida es "1" cuando son CONTRARIAS las entradas	La salida es "0" cuando son CONTRARIAS las entradas
B A S 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **0**	B A S 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **1**
Proteus: XOR / XOR_2 74LS86	Proteus: 4077

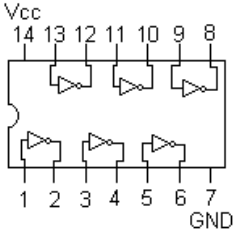
4.2 Diagramas internos de las compuertas familia TTL



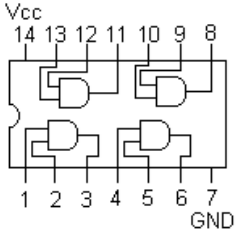
7400 NAND
QUAD 2 INPUT
$S = \overline{A \times B}$



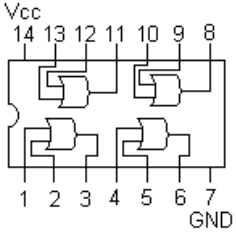
7402 NOR
QUAD 2 INPUT
$S = \overline{A + B}$



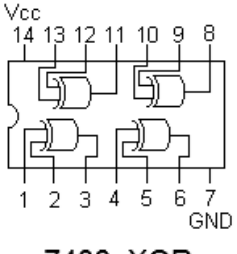
7404
HEX INVERTER
$S = \bar{A}$



7408 AND
QUAD 2 INPUT
$S = A \times B$



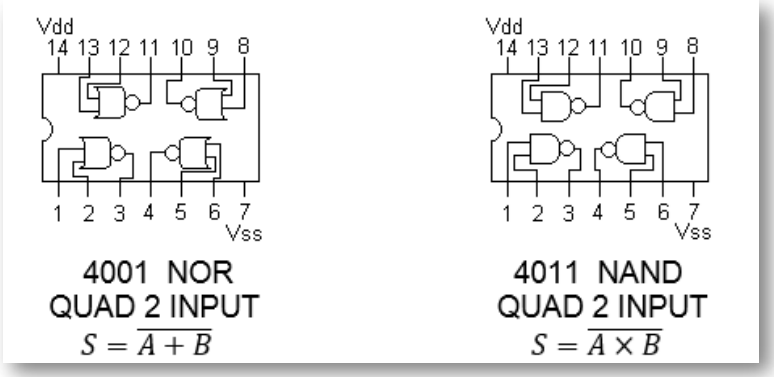
7432 OR
QUAD 2 INPUT
$S = A + B$

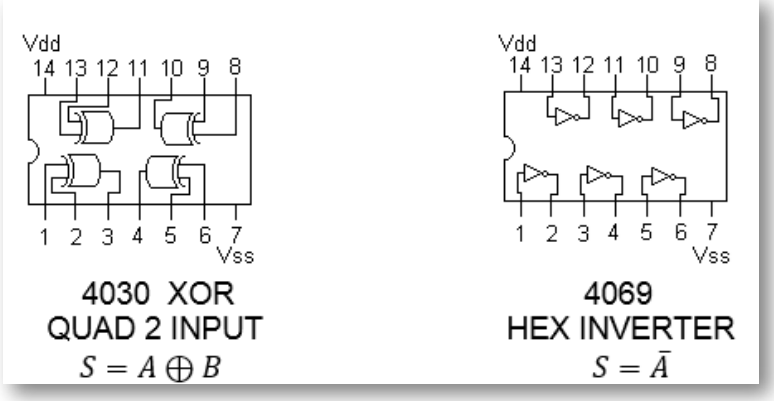


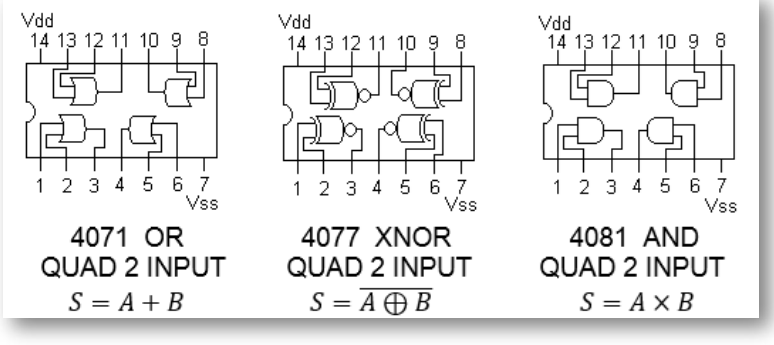
7486 XOR
QUAD 2 INPUT
$S = A \oplus B$

Hojas de Datos: 7400 / 7402 / 7404 / 7408 / 7432 / 7486

4.3 Diagramas internos de las compuertas familia CMOS







Hojas de Datos: 4001 / 4011 / 4030 / 4069 / 4071 / 4077 / 4081

4.4 Referencias comerciales por familias y sus descripciones

TIPO DE COMPUERTA	FAMILIA LÓGICA TTL	FAMILIA LÓGICA CMOS	DESCRIPCIÓN
AND	7408	4081	Quad 2 – input (4 compuertas de 2 entradas cada una)
	7411	4073	Triple 3 – input (3 compuertas de 3 entradas cada una)
	7421	4082	Dual 4 – input (2 compuertas de 4 entradas cada una)
NAND	7400	4011	Quad 2 – input (4 compuertas de 2 entradas cada una)
	7410	4023	Triple 3 – input (3 compuertas de 3 entradas cada una)
	7420 – 7440	4012	Dual 4 – input (2 compuertas de 4 entradas cada una)
	7430	4068	8 – input (1 compuerta de 8 entradas)
	74133	–	13 – input (1 compuerta de 13 entradas)
OR	7432	4071	Quad 2 – input (4 compuertas de 2 entradas cada una)
	–	4075	Triple 3 – input (3 compuertas de 3 entradas cada una)
	–	4072	Dual 4 – input (2 compuertas de 4 entradas cada una)
NOR	7402	4001	Quad 2 – input (4 compuertas de 2 entradas cada una)
	7427	4025	Triple 3 – input (3 compuertas de 3 entradas cada una)
	7425	4002	Dual 4 – input (2 compuertas de 4 entradas cada una)
	–	4078	8 – input (1 compuerta de 8 entradas)
NOT ó INV	7404	4069	Hex (6 compuertas de 1 entrada cada una)
YES ó BUFFER	7407	4050	Hex (6 compuertas de 1 entrada cada una)
XOR	7486 – 74136	4030 – 4070	Quad 2 – input (4 compuertas de 2 entradas cada una)
XNOR	–	4077	Quad 2 – input (4 compuertas de 2 entradas cada una)

4.5 Ejemplos basados en Funciones Lógicas

La fórmula para obtener la cantidad de combinaciones posibles de una función es:

$$2^n = \text{Número de combinaciones posibles}$$
$$n = \text{Cantidad de variables de la función}$$

NOTA: **para este libro el orden de la asignación de las variables dentro de las tablas es la siguiente:**

D C B A

(MSB) ... (LSB)

¿Hallar las tablas de verdad paso a paso de las funciones lógicas y luego implemente los circuitos en un software de simulación electrónica y verificar las soluciones?

★ Ejemplo 1:

$$F = \overline{(A \times B) + (A + B)}$$

F sobre la función completa; 1 sobre $(A \times B)$; 2 sobre $(A + B)$.

n = 2 variables tiene la función
2^n = 4 combinaciones posibles

No. Dec	2^1 B	2^0 A	1 A x B	2 A + B	F $\overline{1+2}$
0	0	0	0	0	1
1	0	1	0	1	0
2	1	0	0	1	0
3	1	1	1	1	0
			AND	OR	NOR

Paso 1: se comienza el diseño de la tabla, colocando las variables de Entrada ***"A y B"*** de la función, con todas las combinaciones de valores binarios posibles.

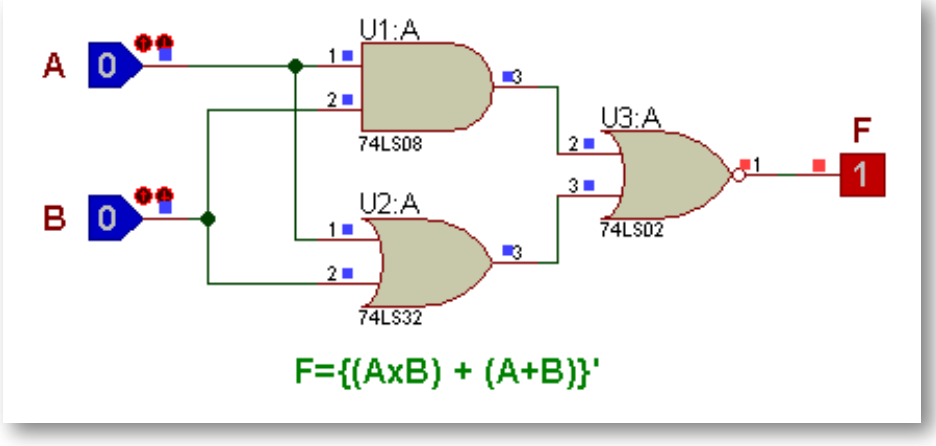
Paso 2: se desarrolla el primer término de la función que está etiquetado con el número **"1"**, dicho término pertenece a la función de una compuerta lógica **"AND de 2 entradas: A y B".**

Paso 3: se desarrolla el segundo término etiquetado con el número "2", éste pertenece a la función de una compuerta lógica **"OR de 2 entradas: A y B".**

Paso 4: se finaliza la solución de la tabla con la unión de los resultados parciales de los términos etiquetados como **"1"** y "2" por medio de una compuerta lógica **"NOR de 2 entradas".**

Paso 5: se implementa la función en un software de simulación electrónica, en este caso se utilizó el **"Proteus"** versión 7.8

Paso 6: se ejecuta la simulación y se valida la solución obtenida en la tabla.



Video explicativo de la solución: Función Lógica 1 - Solución, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo de la simulación: Función Lógica 1 - Simulación, en el canal de youtube de Oscar Botero @oscarboterohenao

Repositorio en Github ("P_función1.dsn") con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

$$F = \underbrace{(A \times B)}_{1} + \underbrace{\bar{C}}_{2} \qquad F = 1 + 2$$

n = 3 variables tiene la función
2^n = 8 combinaciones posibles

No. Dec	2^2 C	2^1 B	2^0 A	1 A x B	2 $\bar{C}$	F 1 + 2
0	0	0	0	0	1	1
1	0	0	1	0	1	1
2	0	1	0	0	1	1
3	0	1	1	1	1	1
4	1	0	0	0	0	0
5	1	0	1	0	0	0
6	1	1	0	0	0	0
7	1	1	1	1	0	1
				AND	NOT	OR

Paso 1: se colocan todas las variables de entrada **"A, B y C"**, con sus 8 combinaciones lógicas.

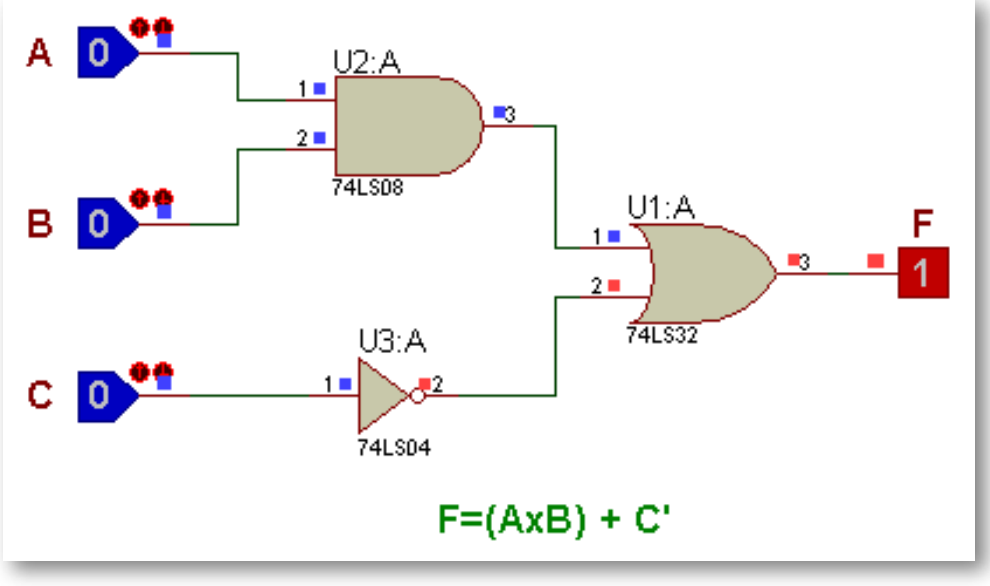
Paso 2: se soluciona el primer término de la función que está etiquetado con el número **"1"**, dicho término pertenece a la función de una compuerta lógica **"AND de 2 entradas: A y B".**

Paso 3: se soluciona el segundo término etiquetado con el número "2", éste pertenece a la función de una compuerta lógica **"NOT de 1 entrada: C".**

Paso 4: finaliza la solución de la tabla con la unión de los resultados parciales de los términos etiquetados como **"1"** y "2" por medio de una compuerta lógica **"OR de 2 entradas".**

Paso 5: se implementa la función en un software de simulación electrónica, en este caso se utilizó el **"Proteus"** versión 7.8

Paso 6: se ejecuta la simulación y se validan todas las respuestas obtenidas en la tabla.



Video explicativo de la solución: Función Lógica 2 - Solución, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo de la simulación: Función Lógica 2 - Simulación, en el canal de youtube de Oscar Botero @oscarboterohenao

Repositorio en Github ("P_función2.dsn") con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

$$F = \{\overbrace{\overline{(A+B)}}^{1} \times \overbrace{(B \times C)}^{2}\}^{3} \oplus \overbrace{\overline{(A \times C)}}^{4}$$

n = 3 variables tiene la función
$\mathbf{2^n}$ = 8 combinaciones posibles

No. Dec	2^2 C	2^1 B	2^0 A	1 $\overline{(A+B)}$	2 $(B \times C)$	3 1×2	4 $\overline{(A \times C)}$	F $3 \oplus 4$
0	0	0	0	1	0	0	1	1
1	0	0	1	0	0	0	1	1
2	0	1	0	0	0	0	1	1
3	0	1	1	0	0	0	1	1
4	1	0	0	1	0	0	1	1
5	1	0	1	0	0	0	0	0
6	1	1	0	0	1	0	1	1
7	1	1	1	0	1	0	0	0
				NOR	AND	AND	NAND	XOR

Paso 1: se colocan todas las variables de entrada $"A, B\ y\ C"$, con sus 8 combinaciones lógicas.

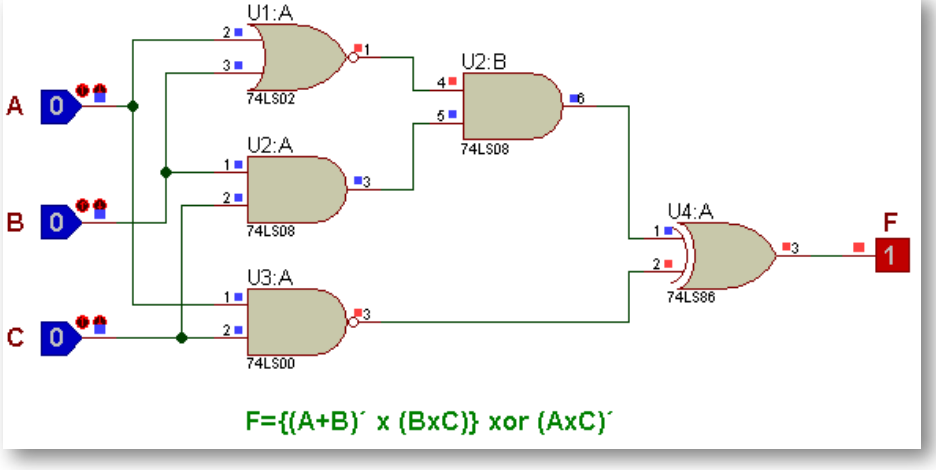
Paso 2: se soluciona el primer término de la función que está etiquetado con el número **"1"**, dicho término pertenece a la función de una compuerta lógica **"NOR de 2 entradas: A y B".**

Paso 3: se soluciona el segundo término etiquetado con el número **"2"**, éste pertenece a la función de una compuerta lógica **"AND de 2 entradas: B y C".**

Paso 4: se resuelve la llave que agrupa los términos etiquetados como **"1"** y **"2"** por medio de una compuerta lógica **"AND de 2 entradas".**

Paso 5: se soluciona el término etiquetado con el número **"4"**, éste pertenece a la función de una compuerta lógica **"NAND de 2 entradas: A y C".**

Paso 6: finaliza la solución de la tabla con la unión de los resultados parciales de los términos etiquetados como **3"** y **"4"** por medio de una compuerta lógica **"XOR de 2 entradas".**



Paso 7: se implementa la función en un software de simulación electrónica, en este caso se utilizó el **"Proteus"** versión 7.8

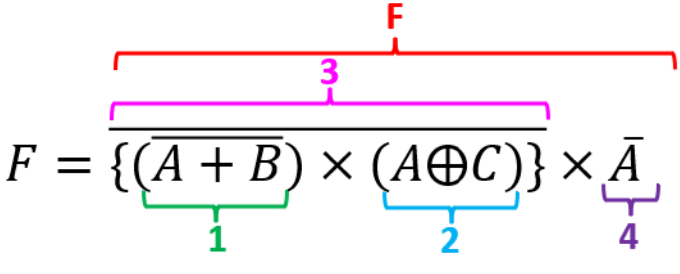
Paso 8: se ejecuta la simulación y se validan todas las respuestas obtenidas en la tabla.

Video explicativo de la solución: Función Lógica 3 - Solución, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo de la simulación: Función Lógica 3 - Simulación, en el canal de youtube de Oscar Botero @oscarboterohenao

Repositorio en Github ("P_función3.dsn") con el archivo de la simulación con Proteus v7.8

★ Ejemplo 4:



n = 3 variables tiene la función
2^n = 8 combinaciones posibles

No. Dec	2^2 C	2^1 B	2^0 A	1 $(\overline{A+B})$	2 $(A \oplus C)$	3 $\overline{1 \times 2}$	4 $\overline{A}$	F 3×4
0	0	0	0	1	0	1	1	*1*
1	0	0	1	0	1	1	0	*0*
2	0	1	0	0	0	1	1	*1*
3	0	1	1	0	1	1	0	*0*
4	1	0	0	1	1	0	1	*0*
5	1	0	1	0	0	1	0	*0*
6	1	1	0	0	1	1	1	*1*
7	1	1	1	0	0	1	0	*0*
				NOR	XOR	NAND	NOT	AND

Paso 1: se colocan todas las variables de entrada $''A, B \; y \; C''$, con sus 8 combinaciones lógicas.

Paso 2: se soluciona el primer término de la función que está etiquetado con el número **"1"**, dicho término pertenece a la función de una compuerta lógica **"NOR de 2 entradas: A y B".**

Paso 3: se soluciona el segundo término etiquetado con el número **"2"**, éste pertenece a la función de una compuerta lógica **"XOR de 2 entradas: A y C".**

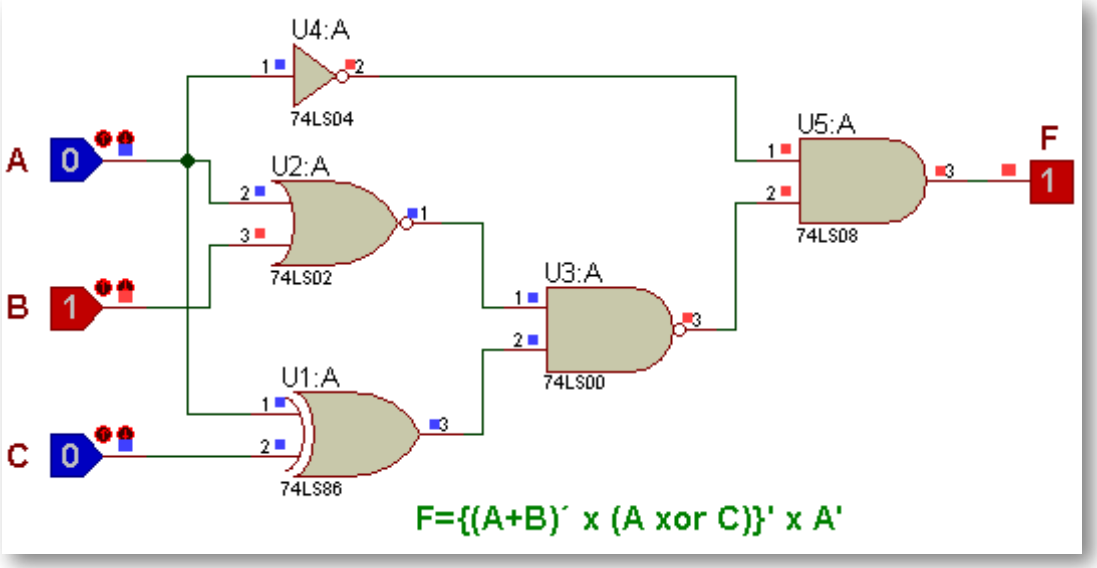
Paso 4: se resuelve la llave que agrupa los términos etiquetados como **"1"** y **"2"** por medio de una compuerta lógica **"NAND de 2 entradas".**

Paso 5: se soluciona el término etiquetado con el número **"4"**, éste pertenece a la función de una compuerta lógica **"NOT de 1 entrada: A".**

Paso 6: finaliza la solución de la tabla con la unión de los resultados parciales de los términos etiquetados como **3"** y **"4"** por medio de una compuerta lógica **"AND de 2 entradas".**

Paso 7: se implementa la función en un software de simulación electrónica, en este caso se utilizó el **"Proteus"** versión 7.8

Paso 8: se ejecuta la simulación y se validan todas las respuestas obtenidas en la tabla.



Video explicativo de la solución: Función Lógica 4 - Solución, en el canal de youtube de Oscar Botero @oscarboterohenao

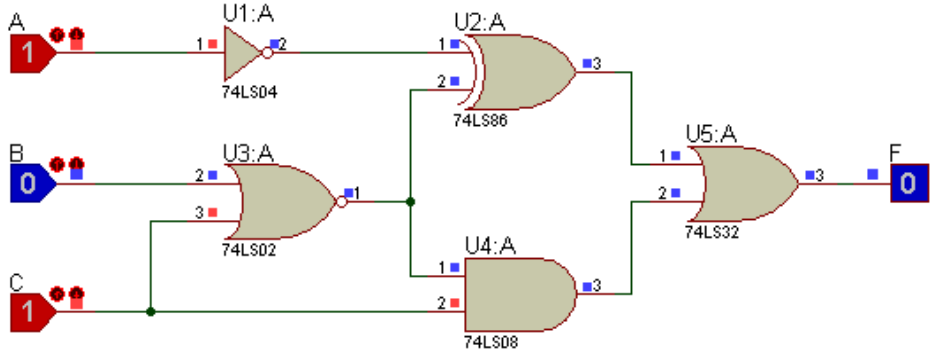
Video explicativo de la simulación: Función Lógica 4 - Simulación, en el canal de youtube de Oscar Botero @oscarboterohenao

Repositorio en Github ("P_función4.dsn") con el archivo de la simulación con Proteus v7.8

4.6 Ejemplos basados en el Circuito

¿Hallar las funciones lógicas de los circuitos, las tablas de verdad paso a paso e implementar los circuitos en un software para verificar las soluciones?

★ Ejemplo 1:



NOTA: para representar la negación se puede utilizar una comilla simple al final del término. Ej: A' = Ā

Paso 1:

- **Compuerta 1:** la señal "A" entra a una compuerta NOT o inversora que la niega:

$$\mathbf{1} = \overline{\boldsymbol{A}}$$

- **Compuerta 2:** las señales "B" y "C" entran a una compuerta NOR que las suma y niega:

$$\mathbf{2} = \overline{(\boldsymbol{B} + \boldsymbol{C})}$$

- **Compuerta 3:** las señales resultantes de las compuertas "1" y "2" entran a una compuerta XOR (OR exclusiva):

$$\mathbf{3} = \overline{\boldsymbol{A}} \oplus \overline{(\boldsymbol{B} + \boldsymbol{C})}$$

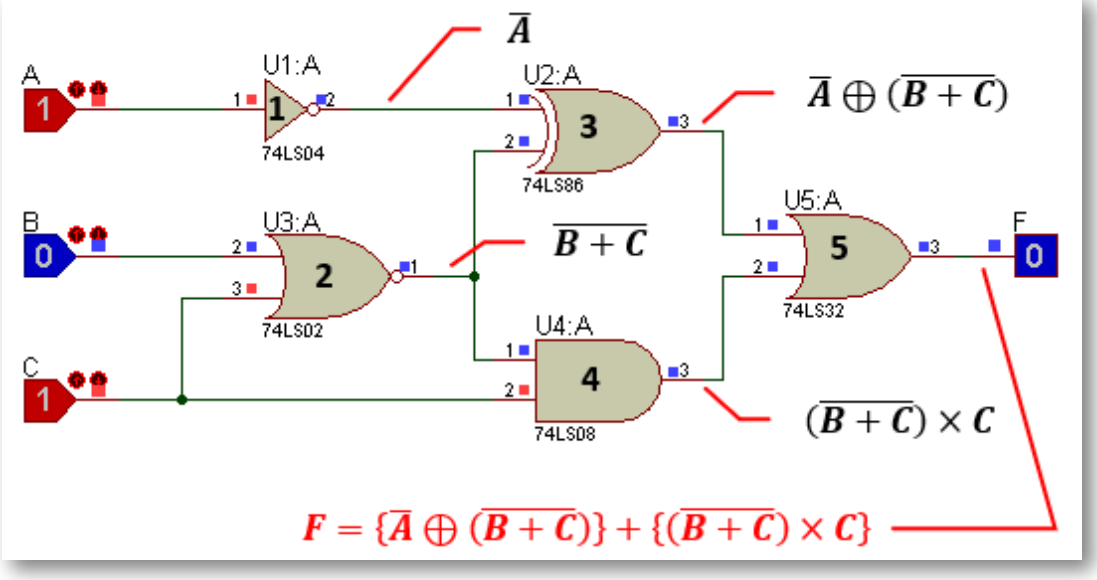
- **Compuerta 4:** la señal resultante de la compuerta "2" y la señal "C" entran a una compuerta AND que las multiplica:

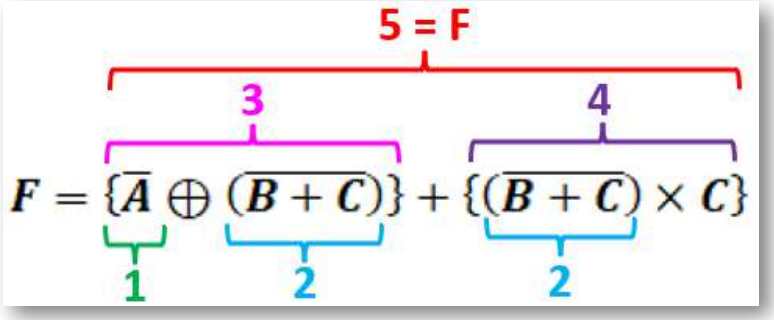
$$\mathbf{4} = \overline{(\boldsymbol{B} + \boldsymbol{C})} \times \boldsymbol{C}$$

- **Compuerta 5:** las señales resultantes de las compuertas "3" y "4" entran a una compuerta OR:

$$5 = F = \{\overline{A} \oplus \overline{(B + C)}\} + \{\overline{(B + C)} \times C\}$$

Paso 2: plano esquemático (SCH) con las etiquetas en cada salida de cada compuerta lógica





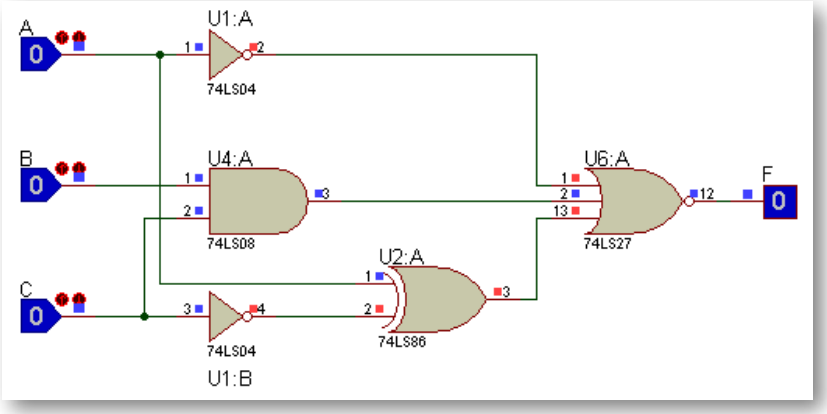
Paso 3: hallar la tabla de verdad paso a paso

No. Dec	2^2 C	2^1 B	2^0 A	1 $\overline{A}$	2 $(\overline{B+C})$	3 $1 \oplus 2$	4 $(\overline{B+C}) \times C$	F $3+4$
0	0	0	0	1	1	0	0	*0*
1	0	0	1	0	1	1	0	*1*
2	0	1	0	1	0	1	0	*1*
3	0	1	1	0	0	0	0	*0*
4	1	0	0	1	0	1	0	*1*
5	1	0	1	0	0	0	0	*0*
6	1	1	0	1	0	1	0	*1*
7	1	1	1	0	0	0	0	*0*
				NOT	NOR	XOR	AND	OR

Paso 4: ejecutar la simulación y validar todas las respuestas obtenidas.

Repositorio en Github ("P_circuito1.dsn") con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:



Paso 1: para hallar la función de salida, se halla y etiqueta cada salida de cada compuerta con su función resultante, comenzando por las entradas hasta llegar a la salida etiquetada como ***"F"***

- **Compuerta 1:** la señal "A" entra a una compuerta NOT o inversora que la niega:

$$\mathbf{1} = \overline{\boldsymbol{A}}$$

- **Compuerta 2:** las señales "B" y "C" entran a una compuerta AND que las multiplica:

$$\mathbf{2} = \boldsymbol{B} \times \boldsymbol{C}$$

- **Compuerta 3:** la señal "C" entra a una compuerta NOT o inversora que la niega:

$$\mathbf{3} = \overline{\boldsymbol{C}}$$

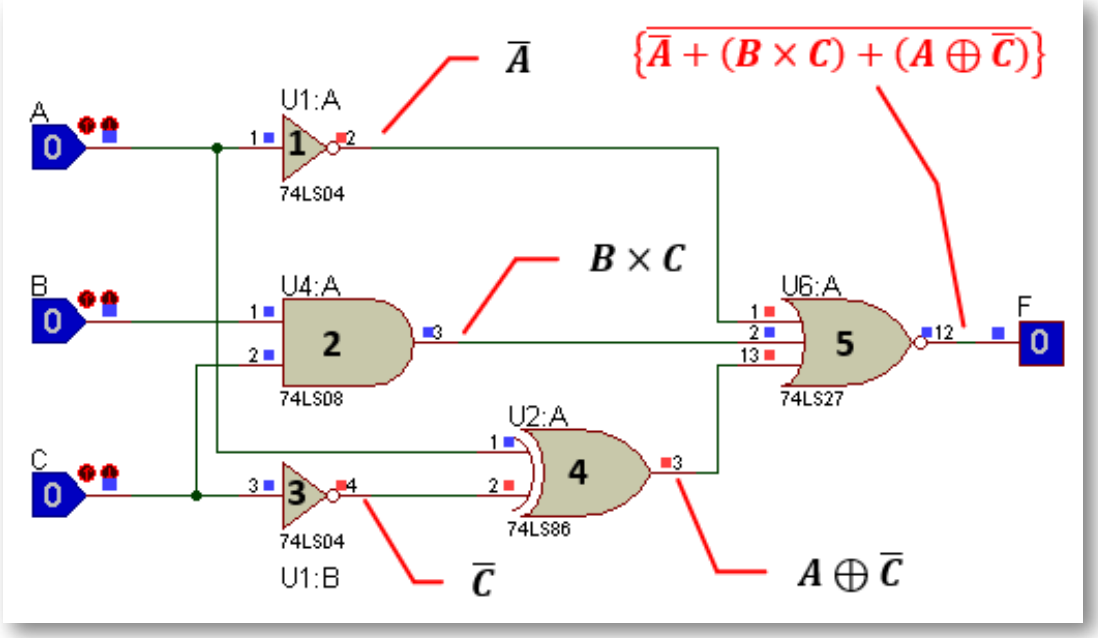
- **Compuerta 4:** la señal y la señal resultante de la compuerta "3" entran a una compuerta XOR (OR exclusiva):

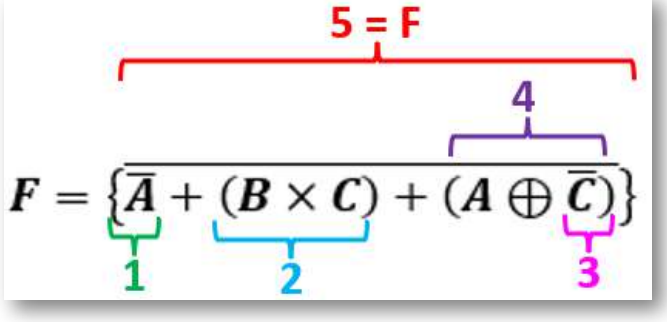
$$\mathbf{4} = \boldsymbol{A} \oplus \overline{(\boldsymbol{C})}$$

- **Compuerta 5:** las señales resultantes de las compuertas "1", "2" y "4" entran a una compuerta NOR de 3 entradas:

$$5 = F = \overline{\{\overline{A} + (B \times C) + (A \oplus \overline{C})\}}$$

Paso 2: dibujo del plano esquemático (SCH) con las etiquetas en cada salida de cada compuerta lógica





Paso 3: hallar la tabla de verdad paso a paso:

No. Dec	2^2 C	2^1 B	2^0 A	1 $\overline{A}$	2 $(B \times C)$	3 $\overline{C}$	4 $A \oplus \overline{C}$	F $\overline{1+2+4}$
0	0	0	0	1	0	1	1	0
1	0	0	1	0	0	1	0	1
2	0	1	0	1	0	1	1	0
3	0	1	1	0	0	1	0	1
4	1	0	0	1	0	0	0	0
5	1	0	1	0	0	0	1	0
6	1	1	0	1	1	0	0	0
7	1	1	1	0	1	0	1	0
				NOT	AND	NOT	XOR	NOR

Paso 4: ejecutar la simulación y validar todas las respuestas obtenidas.

Repositorio en Github ("P_circuito2.dsn") con el archivo de la simulación con Proteus v7.8

4.7 Ejercicios propuestos con respuestas

4.7.1 Basados en funciones lógicas

¿Hallar la tabla de verdad paso a paso de las siguientes funciones lógicas, implementar los circuitos en un software y verificar las soluciones?

NOTA: Las respuestas están en el siguiente orden ascendente como se presenta en la tabla: la primera respuesta pertenece al 0_b; 1_b; 2_b; ...; 13_b; 14_b; hasta llegar al 15_b

★ **Ejercicio 1:**

$$F = \left\{\overline{(A*B)+\overline{\overline{B}}}\right\}+C+(A\oplus C)$$

R/. 0,1,1,1,1,1,1,1

★ **Ejercicio 2:**

$$S = \left\{\overline{(\overline{A}+B)}*C\right\}+\left\{\overline{(B\oplus D)}*\overline{D}\right\}$$

R/. 1,1,0,0,1,1,0,0,0,0,0,0,0,1,0,0

★ **Ejercicio 3:**

$$F = \overline{\left\{\overline{\overline{(\overline{A}*C)}+\overline{\overline{A}}}\right\}*(B+C)}$$

R/. 1,1,1,0,1,0,1,0

★ **Ejercicio 4:**

$$S = \left[\left\{(A*C)+\overline{\overline{B}}\right\}*\overline{\overline{D}}\right]+\left(\overline{B+\overline{\overline{D}}}\right)$$

R/. 0,0,1,1,0,0,1,0,1,1,1,1,1,1,1,1

★ **Ejercicio 5:**

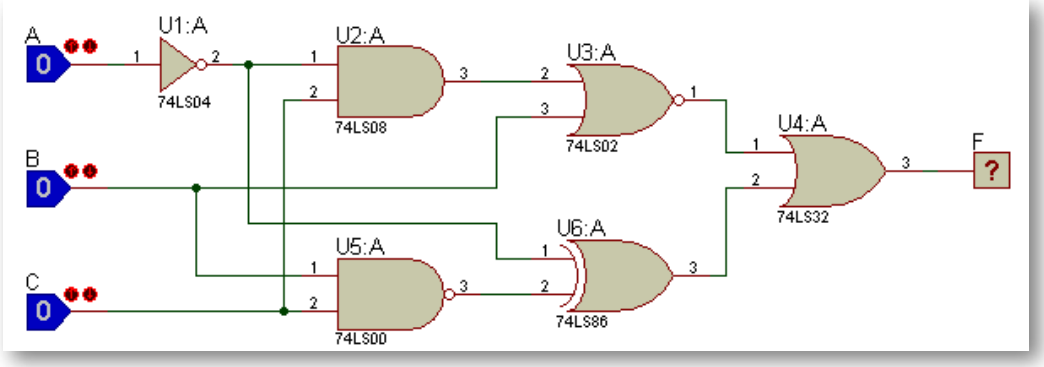
$$S = \left[(A+B)+(A\oplus D)\right]\oplus\left[\left(\overline{A\times B}\right)+(A+C)\right]$$

R/. 1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0

4.7.2 Basados en el circuito

¿Hallar las funciones lógicas, las tablas de verdad paso a paso e implemente los circuitos en un software y verifique los resultados obtenidos?

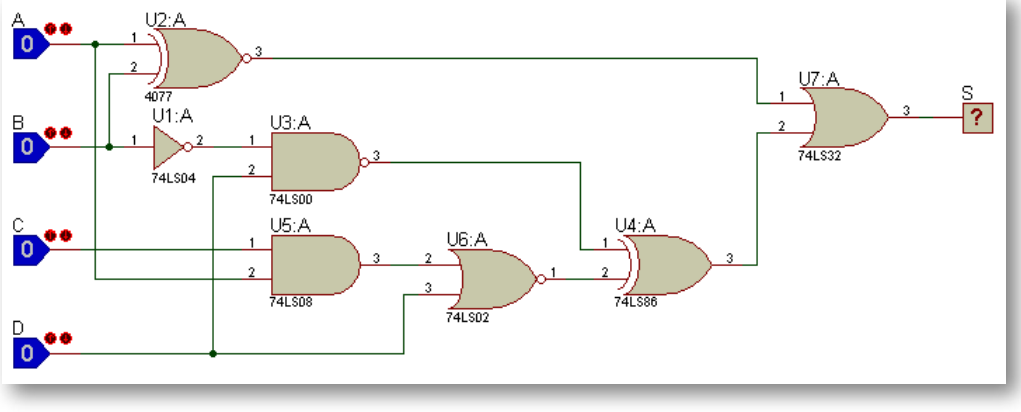
★ Ejercicio 1:



$$F = \left[\left(\overline{A} * C\right) + B\right] + \left[\left(\overline{B * C}\right) \oplus \overline{A}\right]$$

R/. 1,1,0,1,0,1,1,0

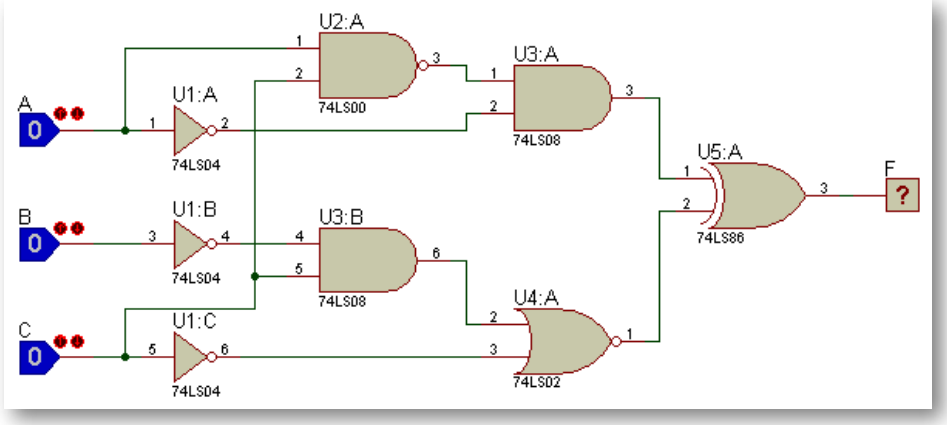
★ Ejercicio 2:



$$S = \left(\overline{A \oplus B}\right) + \left\{\left(\overline{\overline{B} * D}\right) \oplus \left[\overline{(A * C) + D}\right]\right\}$$

R/. 1,0,0,1,1,1,0,1,1,0,1,1,1,0,1,1

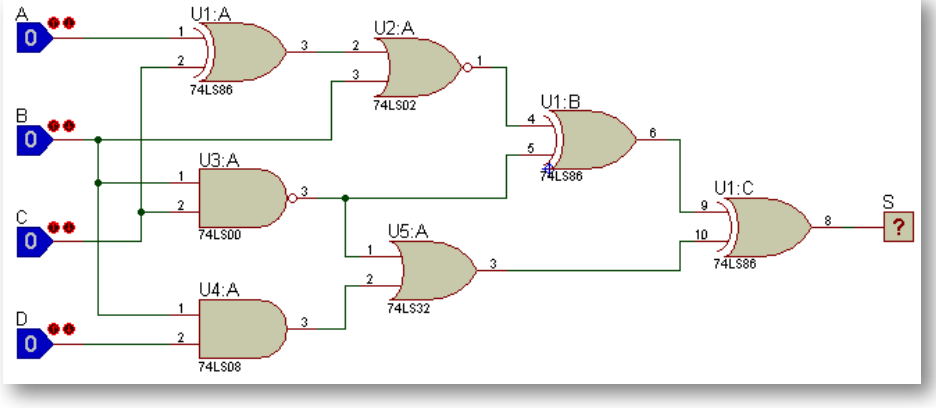
★ Ejercicio 3:



$$F = \left[\left(\overline{A * C}\right) * \overline{A}\right] \oplus \left[\left(\overline{\overline{B} * C}\right) + \overline{\overline{C}}\right]$$

R/. 1,0,1,0,1,0,0,1

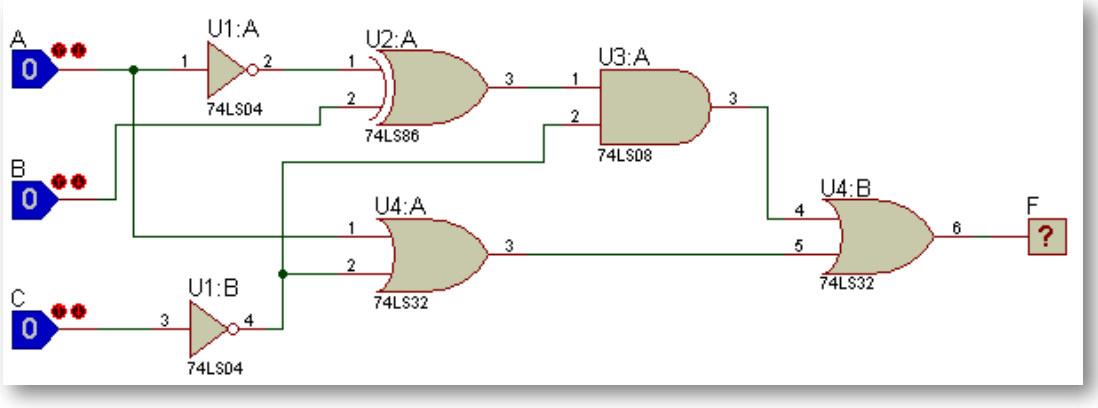
★ Ejercicio 4:



$$S = \left\{\left[\left(\overline{A \oplus C}\right) + B\right] \oplus \left(\overline{B * C}\right)\right\} \oplus \left[\left(\overline{B * C}\right) + (B * D)\right]$$

R/. 1,0,0,0,0,1,0,0,1,0,0,0,0,1,1,1

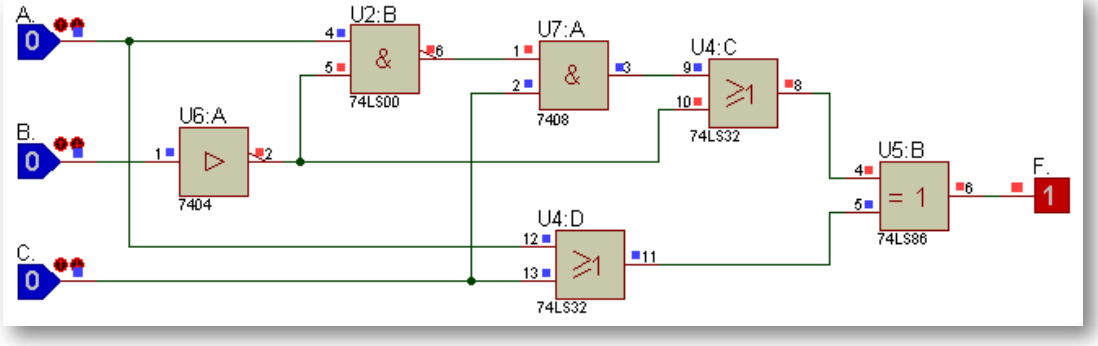
★ Ejercicio 5:



$$F = \left[\left(\overline{A} \oplus B\right) \times \overline{C}\right] + \left(A + \overline{C}\right)$$

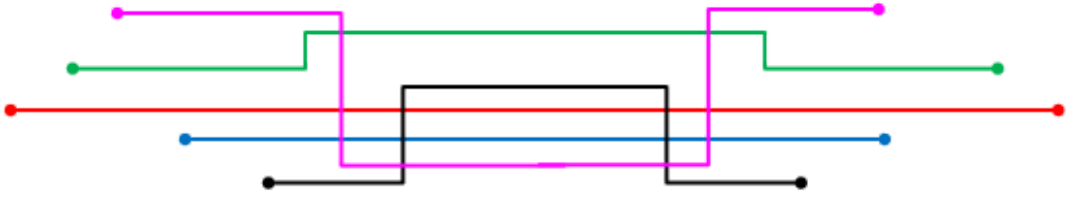
R/. 1,1,1,1,0,1,0,1

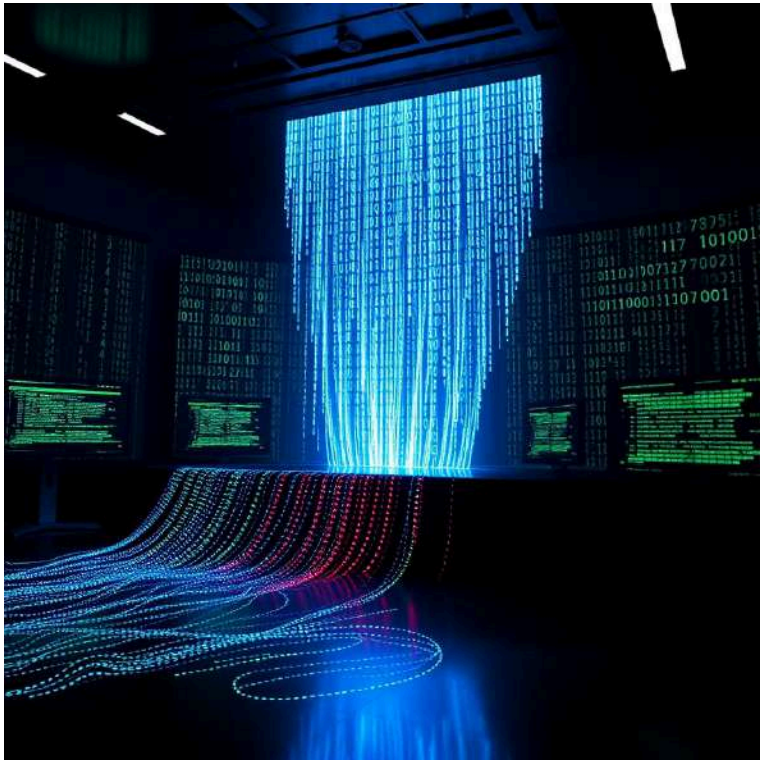
★ Ejercicio 6:



$$F = \left\{\left[\left(\overline{A * \overline{B}}\right) * C\right] + \overline{B}\right\} \oplus (A + C)$$

R/. 1,0,0,1,0,0,0,0






AYUDAS
COMPLEMENTOS

4.8 Videos

Video explicativo: Crear Tablas de Verdad para solución de Circuitos Digitales:, en el canal de youtube de Oscar Botero @oscarboterohenao

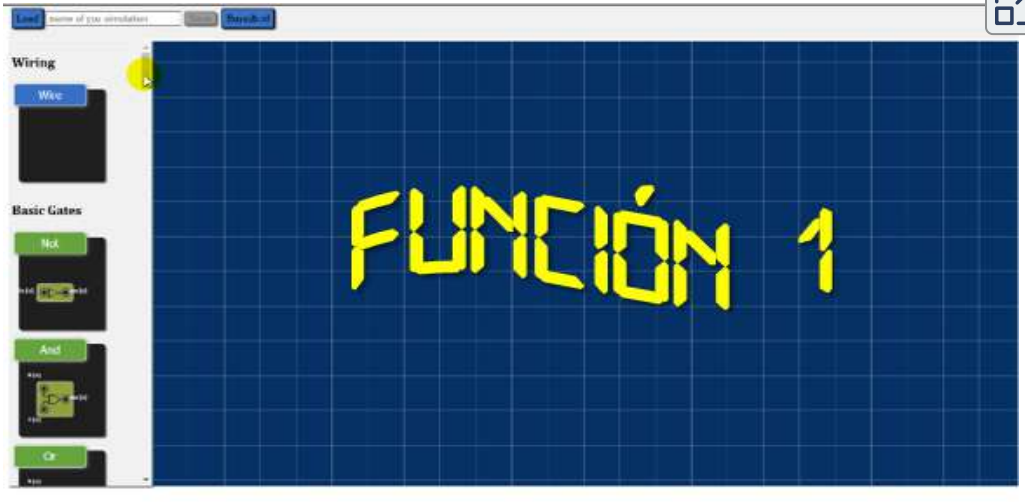
Video explicativo: Compuertas Lógicas Individuales - Simulación Proteus, en el canal de youtube de Oscar Botero @oscarboterohenao

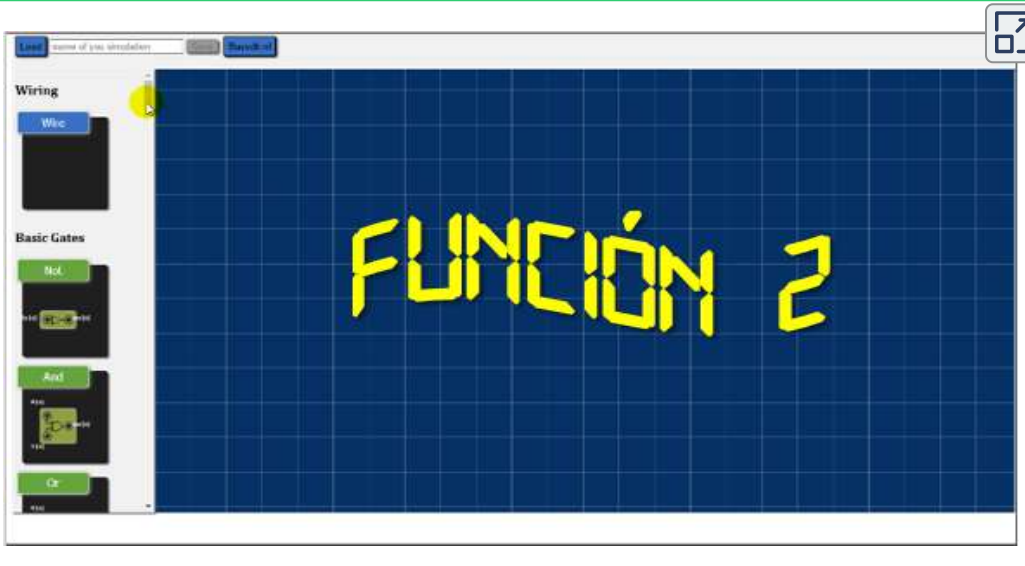
Video explicativo: Compuertas Lógicas con conexión Alámbrica - Simulación Proteus, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo de la solución: Compuertas Lógicas con conexión Inalámbrica - Simulación Proteus, en el canal de youtube de Oscar Botero @oscarboterohenao

Haga clic en la esquina superior derecha para expandir el video.

Videos demostrativos creados por Oscar Ignacio Botero Henao utilizando la aplicación "LogicsSandbox", simulando los ejemplos de las funciones 1 y 2 vista en las secciones 5.1 y 5.2 de este capítulo.





Esta aplicación se llama "LogicsSandbox" y simula circuitos digitales, fue creada por: Buys de Barbanson - (repositorio en Github). El autor expone que, esta versión no permite guardar las simulaciones y tiene errores.

Haga clic en la esquina superior derecha para expandir el video.

Video demostrativo creado por Oscar Ignacio Botero Henao utilizando la aplicación "LogicsSandbox", simulando el ejemplo de la "Función 2" vista en la sección 5.2

El profesor Arturo Javier Miguel del Prieto Paz (Perú), creó un simulador digital con Board Virtual llamado: "Constructor Virtual sobre Protoboards y Simulador de Circuitos Digitales".

4.9 Objetos Interactivos

Haga clic en la esquina superior derecha para expandir el video.

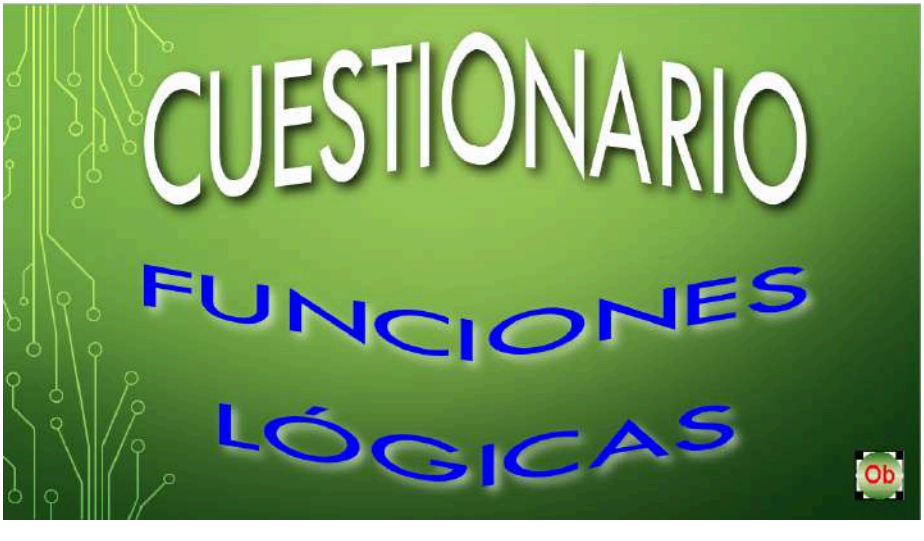
El primer Objeto Interactivo es sobre la simulación del comportamiento de las 8 compuertas lógicas. El segundo objeto interactivo circuitos combinacionales donde puede interactuar y observar el comportamiento, la tabla de verdad y las funciones lógicas y estados lógicos en todas las partes del circuito. [11]

[11] Los dos Objetos Interactivos fueron diseñados por Oscar Ignacio Botero Henao.

El Objeto Interactivo con 5 circuitos combinacionales fue diseñado por Oscar Ignacio Botero Henao.

Haga clic sobre la imagen para abrir el objeto interactivo:

Este objeto interactivo fue generado con la *asistencia* de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/



4.10 Simuladores Circuitos Digitales

➤ La empresa LabCenter Electronics ofrece un Demo de "Proteus" descargable (requiere registro), los archivos de simulación con Proteus utilizados en este libro se pueden ejecutar y modificar con el "Demo":

➤ Para ejecutar la aplicación "LogicsSandbox" dar clic en el siguiente ícono:

➤ Aplicación para simular circuitos digitales, "Constructor Virtual sobre Protoboards y Simulador de Circuitos Digitales" del Ing. Arturo Javier Miguel del Prieto Paz; haz clic en el ícono para descargarlo:

➤ La empresa National Instruments ofrece una versión estudiantil descargable del programa "Multisim" (requiere registro):

➤ La empresa Analog Devices ofrece una versión gratuita descargable del programa "LTspice" (requiere registro):

➤ El programa Logisim es una versión gratuita y descargable. Simula circuitos digitales esquemáticos:

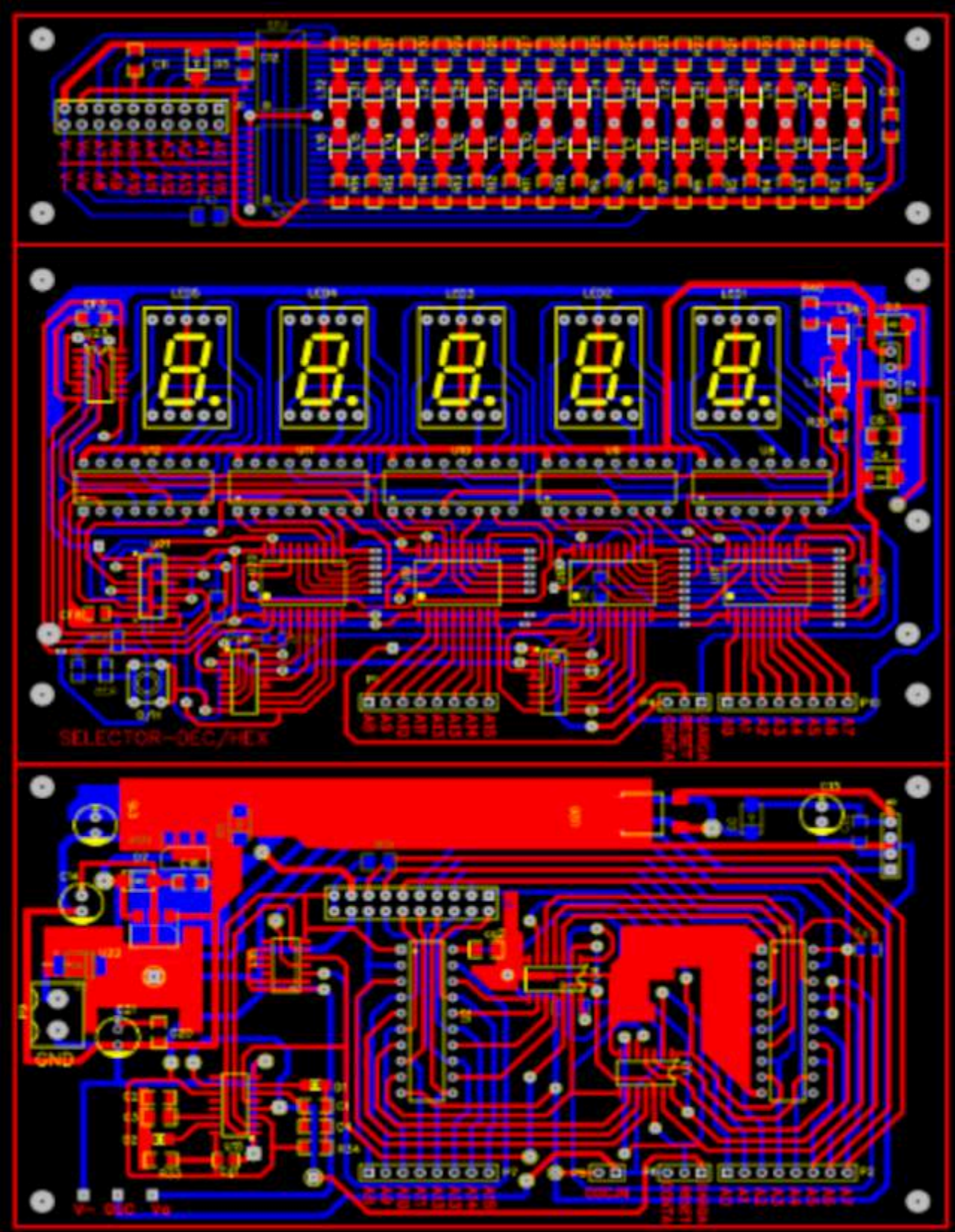
➤ La empresa DesignSoft diseñó el programa Tina-TI para Texas Instruments (TI), es una versión gratuita y descargable (requiere registro). Simula circuitos digitales esquemáticos:

➤ El programa Tinkercad es gratuito (requiere registro). Simula circuitos digitales en línea:

➤ El programa CircuitLab es gratuito (requiere registro). Simula circuitos digitales esquemáticos en línea:

Capítulo V

Minterm y Maxterm
Lógica AOI


SELECTOR-DEC/HEX
GND

Este método **"convierte o transforma"** cualquier función lógica en otra función con solo compuertas lógicas **AND, OR y NOT**; de allí su nombre de **"LÓGICA AOI"**.[1][2][3][4][10]

- **Minterm (Σ):** son todas las combinaciones para las que la función toma el valor de salida de '1'
- **Maxterm (∏):** son todas las combinaciones para las que la función toma el valor de salida de '0'
- **SOP (Suma de Productos):** es una expresión booleana compuesta por un único término producto o por una suma de términos producto
- **POS (Producto de Sumas):** es una expresión booleana compuesta por un único término suma o por un producto de sumas

NOTA: ***Las notaciones que se suelen utilizar para expresar la negación son:***

- **La variable A está negada:** $\bar{A}$; A'; $|A$; A^*
- **Todo el término está negado:** $\overline{(A+B)}$; $(A+B)'$; $|(A+B)$; $(A+B)^*$

5.1 Ejemplos basados en la función lógica

¿Halle la tabla de verdad e implemente el circuito combinacional para las siguientes funciones lógicas?

★ Ejemplo 1: Minterm = Minitérmino (Σ):

$$F(\Sigma) = (\overline{A} \times B \times C) + (A \times \overline{C})$$

Esta función es un minterm ya que es una **"Suma de Productos" (SOP)** y cada término de la función genera un '1' lógico en su salida. Las variables que están negadas en la función toman el valor de '0' y las que **"NO"** lo están toman el valor de '1'

En el primer término $(\overline{A} \times B \times C)$ están declaradas todas las variables de entrada. En el segundo término $(A \times \overline{C})$ la variable **'B'** no está declarada y puede tomar un valor de '1' o de '0', o sea que es "indiferente" el valor que tome dentro de este término.

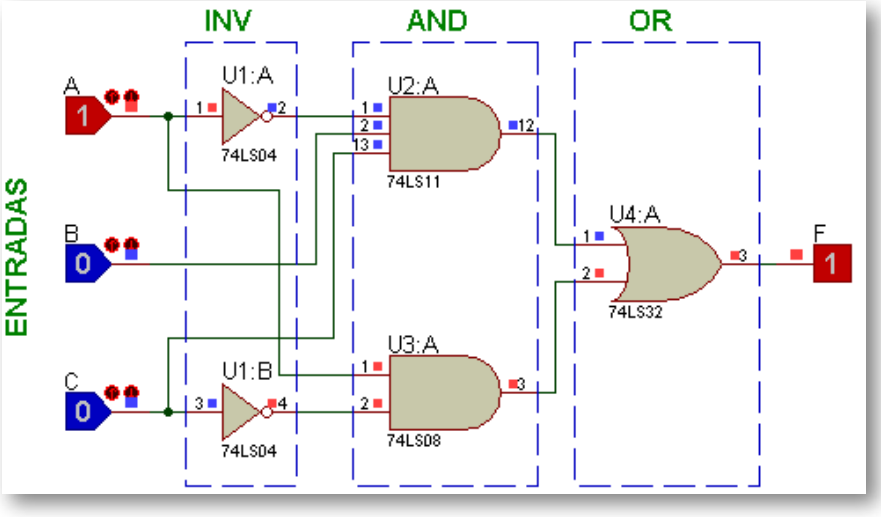
$$F(\Sigma) = (\overline{A} \times B \times C) + (A \times \overline{C})$$

$$F(\Sigma) = (0 \times 1 \times 1) + (1 \times 0)$$

Se construye la tabla de verdad completa:

C	B	A	F	Término
0	0	0	0	
0	0	1	1	$(A \times \overline{C})$
0	1	0	0	
0	1	1	1	$(A \times \overline{C})$
1	0	0	0	
1	0	1	0	
1	1	0	1	$(\overline{A} \times B \times C)$
1	1	1	0	

Se implementa el circuito en un software de simulación electrónica y se verifican los resultados obtenidos en la tabla de verdad; para este libro se utilizó el Proteus v7.8:



Repositorio en Github (Ejemplo_1.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2: Maxterm = Maxitérmino (∏):

$$\boldsymbol{F(\Pi} = (\boldsymbol{A} + \boldsymbol{B} + \overline{\boldsymbol{C}}) \times (\boldsymbol{A} + \overline{\boldsymbol{B}})$$

Esta función es un maxterm ya que es un **"Producto de Sumas" (POS)** y cada término de la función genera un '0' lógico en su salida.

Las variables que en la función están negadas toman el valor de '1' y las que **"NO"** lo están toman el valor de '0'

En el primer término $(A + B + \overline{C})$ están declaradas todas las variables de entrada. En el segundo término $(A + \overline{B})$ la variable **'C'** no está declarada y puede tomar un valor de '1' o de '0', o sea que es "indiferente" el valor que tome dentro de este término.

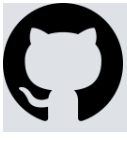
$$F(\Pi) = (A + B + \overline{C}) \times (A + \overline{B})$$

$$F(\Pi) = (0 + 0 + 1) \times (0 + 1)$$

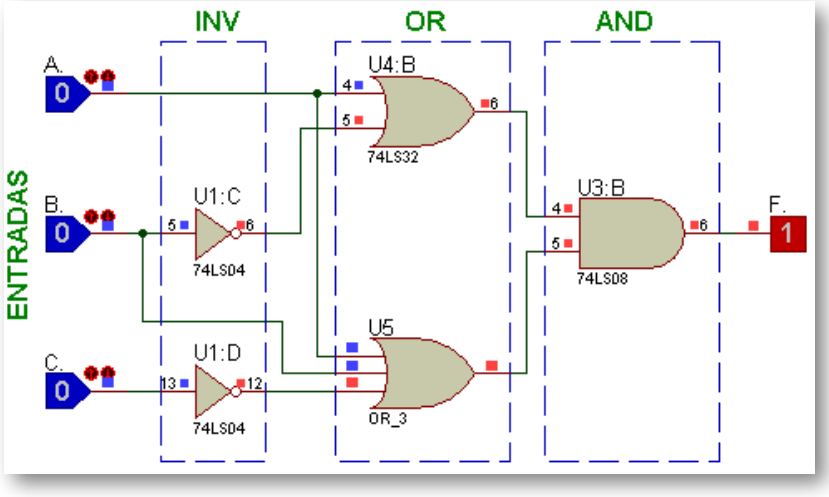
Se construye la tabla de verdad completa:

C	B	A	F	Término
0	0	0	1	
0	0	1	1	
0	1	0	0	$(A + \overline{B})$
0	1	1	1	
1	0	0	0	$(A + B + \overline{C})$
1	0	1	1	
1	1	0	0	$(A + \overline{B})$
1	1	1	1	

Se implementa el circuito en un software de simulación electrónica y se verifican los resultados obtenidos en la tabla de verdad; para este libro se utilizó el Proteus v7.8



Repositorio en Github (Ejemplo_1.dsn) con el archivo de la simulación con Proteus v7.8



5.2 Ejemplos basados en la tabla de verdad

Con base en la tabla de verdad, ¿Halle la función lógica de la forma más simple posible, cuál método aplica "Minterm o Maxterm" e implemente el circuito combinacional en un software de simulación electrónica y verifique la solución?

★ Ejemplo 1:

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

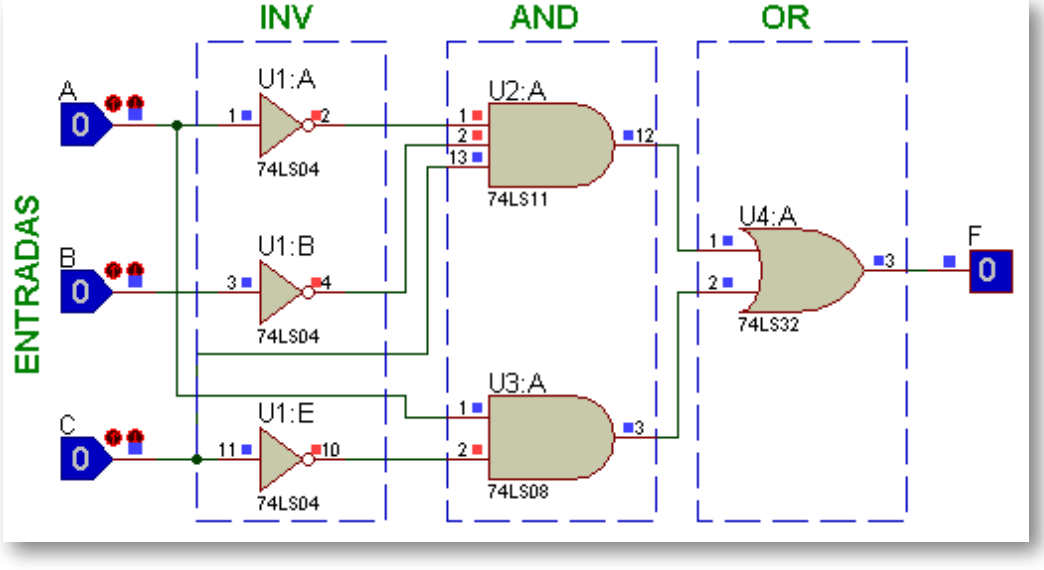
Se aplica el método **"Minterm (Σ)"** para obtener la función lógica, debido a la menor cantidad de '1' lógicos que se presentan en la salida de la tabla y por ende la función lógica dará más simplificada.

La función lógica es:

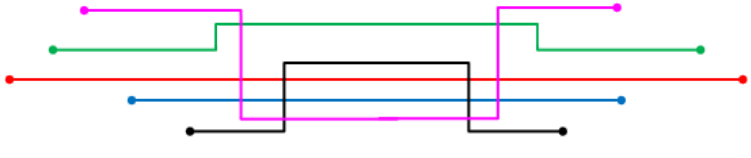
$$F(\Sigma) = (\overline{A} \times \overline{B} \times C) + (A \times \overline{C})$$

$$F(\Sigma) = (0 \times 0 \times 1) + (1 \times 0)$$

La implementación del circuito es:



Repositorio en [Github (Ejemplo_2.dsn)] con el archivo de la simulación con Proteus v7.8



★ Ejemplo 2:

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

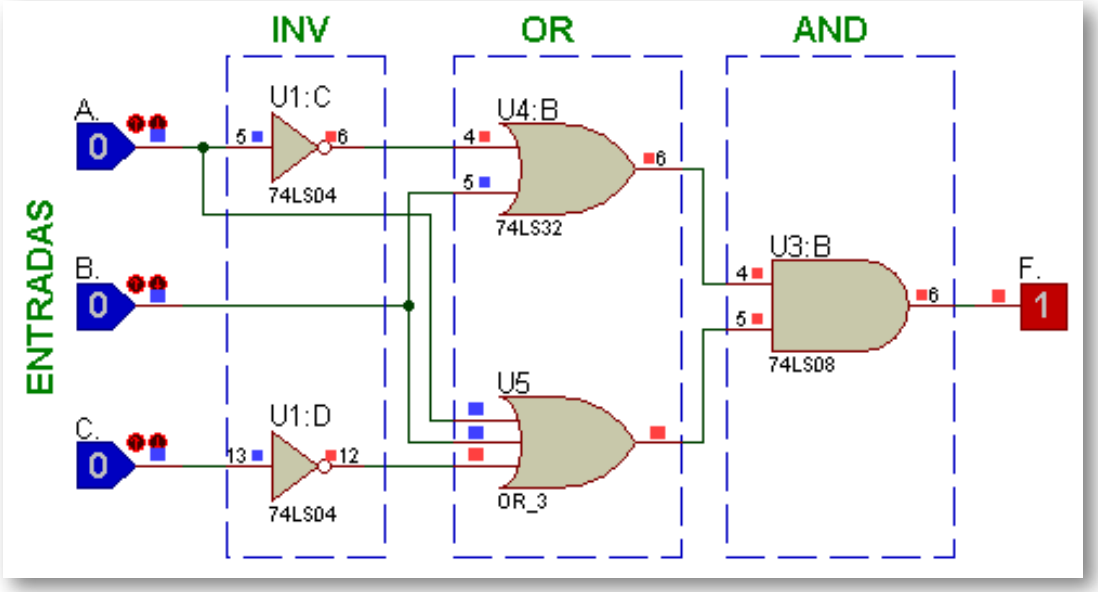
Se aplica el método **"Maxterm (∏)"** para obtener la función lógica, debido a la menor cantidad de '0' lógicos que se presentan en la salida de la tabla y por ende la función lógica dará más simplificada.

La función lógica es:

$$F(\Pi) = (\overline{A} + B) \times (A + B + \overline{C})$$

$$F(\Pi) = (1 + 0) \times (0 + 0 + 1)$$

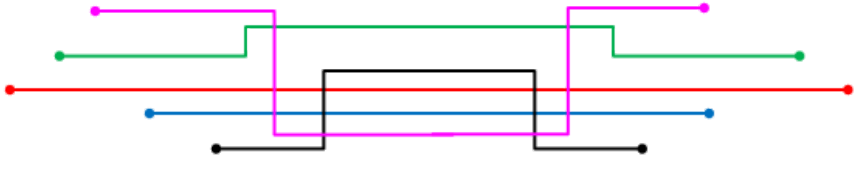
La implementación del circuito es:



Repositorio en [Github (Ejemplo_2.dsn)] con el archivo de la simulación con Proteus v7.8

5.3 Ejemplos basados en enunciados con condiciones

¿Halle la tabla de verdad completa, la función lógica más simple e implemente el circuito combinacional en un software de simulación electrónica y verifique la solución?



★ Ejemplo 1:

*Una alarma se activa cuando se cumplen las condiciones **a** o **b**:*

a. La variable C no esté en '0', la variable B esté en '0'

b. Las tres variables no estén en '1'

2^n = Número de combinaciones posibles
n = Cantidad de variables de la función

Las condiciones mencionan 3 variables, lo que lleva a 8 posibles combinaciones de las variables de entrada y cuando dichas condiciones se cumplan "Activa" una alarma, o sea que la salida vale '1'

Se buscan las combinaciones de las entradas que cumplan la condición ***a*** o ***b***:

La variable C está en '1' y la variable B está en '0', no se menciona la variable A, quiere decir que es "Indiferente" el valor que esta tome. Cumplen:

C	B	A	F	Con	SOP
1	0	0	1	a)	$(\overline{A} \times \overline{B} \times C)$
1	0	1	1	a)	$(A \times \overline{B} \times C)$

Las tres variables están en '0'. Cumple:

C	B	A	F	Con	SOP
0	0	0	1	b)	$(\overline{A} \times \overline{B} \times \overline{C})$

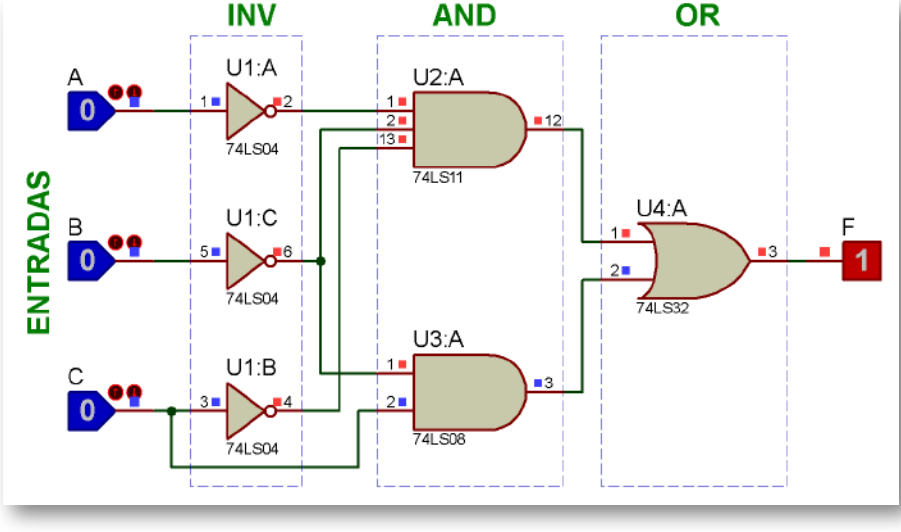
Se ubican las combinaciones anteriores en la tabla y se completa:

C	B	A	F	Con	SOP
0	0	0	1	b)	$(\overline{A} \times \overline{B} \times \overline{C})$
0	0	1	0		
0	1	0	0		
0	1	1	0		
1	0	0	1	a)	$(\overline{A} \times \overline{B} \times C)$
1	0	1	1	a)	$(A \times \overline{B} \times C)$
1	1	0	0		
1	1	1	0		

La función por Minterm (Σ) es:

$$F(\Sigma) = (\overline{A} \times \overline{B} \times \overline{C}) + (\overline{A} \times B \times C) + (A \times \overline{B} \times C)$$

El circuito implementado es:



Repositorio en Github (Ejemplo_3.dsn pág1) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

*Una lámpara se apaga cuando se cumplen las 2 condiciones **a** o **b**:*

a. Las variables B y C estén en '0'

b. Las tres variables no estén en '0'

Las condiciones mencionan 3 variables, lo que lleva a 8 posibles combinaciones de las variables de entrada y cuando dichas condiciones se cumplan "Apaga" una lámpara, o sea que la salida vale '0'

Se buscan las combinaciones de las entradas que cumplan la condición ***a*** o ***b***:

Las variables B y C están en '0', no se menciona la variable A, quiere decir que es "Indiferente" el valor que esta tome. Cumplen:

C	B	A	F	Con	POS
0	0	0	0	a)	$(A+B+C)$
0	0	1	0	a)	$(\overline{A}+B+C)$

Las tres variables están en '1'. Cumple:

C	B	A	F	Con	POS
1	1	1	0	b)	$(\overline{A}+\overline{B}+\overline{C})$

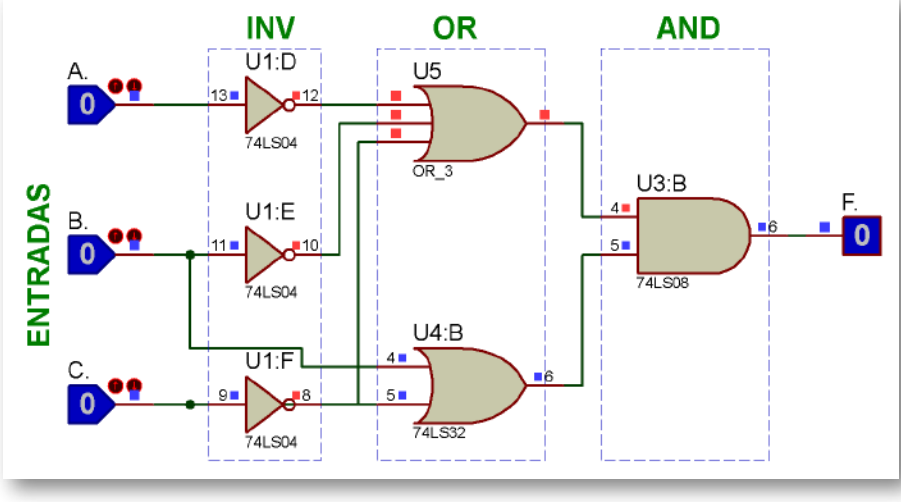
Se ubican las combinaciones anteriores en la tabla y se completa:

C	B	A	F	Con	POS
0	0	0	0	a)	$(A+B+C)$
0	0	1	0	a)	$(\overline{A}+B+C)$
0	1	0	1		
0	1	1	1		
1	0	0	1		
1	0	1	1		
1	1	0	1		
1	1	1	0	b)	$(\overline{A}+\overline{B}+\overline{C})$

La función desarrollada por Maxterm (∏) es:

$$F(\Pi) = (\overline{A}+\overline{B}+\overline{C}) \times (A+B+C) \times (\overline{A}+B+C)$$

El circuito implementado es:



Repositorio en Github (Ejemplo_3.dsn pág1) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

Se tiene un sistema que estampa la etiqueta en una pieza. En el sistema la variable A es un sensor que detecta la presencia el operario en la máquina ('1' = presente), la variable B registra la activación de la electroválvula de la prensa y la variable C censa que la pieza esté en la posición correcta. Diseñe los circuitos combinacionales que cumplan con las siguientes condiciones:

- *Una alarma sonora que se active cuando el sensor de posición de la pieza y la electroválvula están activadas y el operario no se encuentre en la máquina; o cuando la electroválvula se active, el operario esté presente en la máquina y la pieza no esté en la posición correcta.*
- *Un indicador lumínico que permanece activado mientras no se cumpla la activación de los tres sensores.*

El sistema consta de tres variables (A, B y C) y 8 posibles combinaciones, dos salidas: la alarma sonora y la lumínica.

La alarma sonora se activa cuando se cumplan las siguientes combinaciones:

C	B	A	S	SOP
0	1	1	1	$(A \times B \times \overline{C})$
1	1	0	1	$(\overline{A} \times B \times C)$

El indicador lumínico se desactiva cuando se cumpla la siguiente combinación:

C	B	A	L	POS
1	1	1	0	$(\overline{A}+\overline{B}+\overline{C})$

La tabla de verdad completa es:

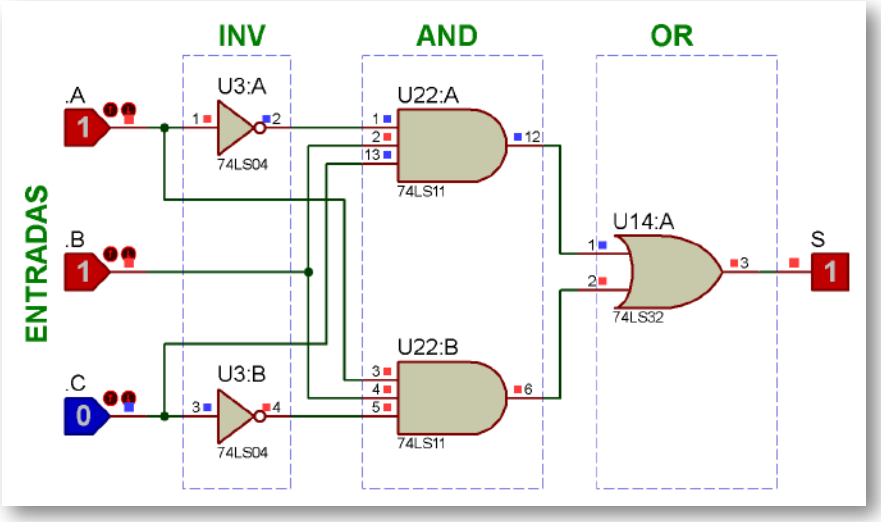
C	B	A	S	L	SOP	POS
0	0	0	0	1		
0	0	1	0	1		
0	1	0	0	1		
0	1	1	1	1	$(A\times B\times \overline{C})$	
1	0	0	0	1		
1	0	1	0	1		
1	1	0	1	1	$(\overline{A}\times B\times C)$	
1	1	1	0	0		$(\overline{A}+\overline{B}+\overline{C})$

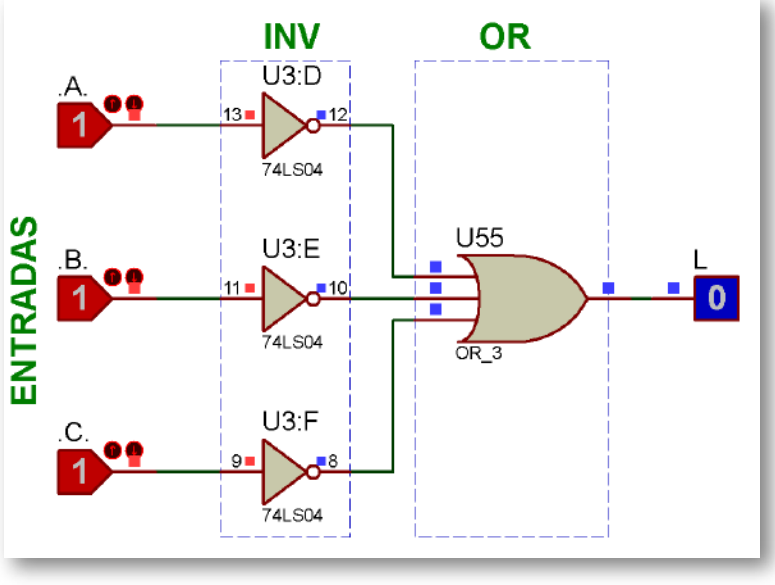
Las funciones lógicas para los dos circuitos combinacionales son:

$$S(\Sigma) = (A \times B \times \overline{C}) + (\overline{A} \times B \times C)$$

$$L(\Pi) = (\overline{A} + \overline{B} + \overline{C})$$

Los circuitos combinacionales para la alarma sonora y para el indicador lumínico son:





Repositorio en [Github (Ejemplo_3.dsn pág2)] con el archivo de la simulación con Proteus v7.8

5.4 Ejercicios propuestos con respuestas

5.4.1 Basados en funciones lógicas

¿Hallar las tablas de verdad completas, implemente los circuitos en un software de simulación electrónica y verifique los resultados obtenidos?

★ Ejercicio 1:

$$F = (A + \overline{B} + \overline{C} + D) * (\overline{A} + B + C + D) * (\overline{A} + B + C + \overline{D}) * (\overline{A} + B + \overline{C} + \overline{D})$$

R/. 1,0,1,1,1,1,0,1,1,0,1,1,1,0,1,1

★ Ejercicio 2:

$$F = (A + \overline{B} + \overline{C} + D) * (\overline{A} + B + C + D) * (\overline{A} + B + C + \overline{D}) * (\overline{A} + \overline{B} + \overline{C} + \overline{D})$$

R/. 1,0,1,1,1,1,0,1,1,0,1,1,1,1,1,0

★ Ejercicio 3:

$$F = (\overline{A} * B * C * \overline{D}) + (\overline{A} * B * \overline{C} * \overline{D}) + (A * \overline{B} * \overline{C} * D) + (A * B * \overline{C} * D) + (A * \overline{B} * \overline{C} * \overline{D})$$

R/. 0,1,1,0,0,0,1,0,0,1,0,1,0,0,0,0

★ Ejercicio 4:

$$F = (\overline{A} * \overline{B} * C * \overline{D}) + (\overline{A} * B * \overline{C} * \overline{D}) + (A * \overline{B} * \overline{C} * D) + (A * B * \overline{C} * D) + (A * \overline{B} * \overline{C} * \overline{D})$$

R/. 0,1,1,0,1,0,0,0,0,1,0,1,0,0,0,0

5.4.2 Basados en las tablas de verdad

¿Hallar la función lógica de la forma más simple de las siguientes tablas de verdad e implemente el circuito combinacional en un software de simulación electrónica y verifique la solución?

★ Ejercicios 1 a 4:

Los ejercicios pueden ser resueltos por ambos métodos (Σ o ∏), pero el criterio de selección para dar cumplimiento al enunciado de hallar la función más simple es observando la salida en la tabla de verdad y eligiendo la menor cantidad de '0' o '1' que hallan.

- Si la menor cantidad son los '1' se resuelve por Minterm (Σ)
- Si la menor cantidad son los '0' se resuelve por Maxterm (∏)

Ejercicio 1

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

Ejercicio 2

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

Ejercicio 3

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

Ejercicio 4

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

- **R/. Función lógica del ejercicio 1:**

  Se resolvió por Minterm (Σ)

$$F = (\overline{A} \times B \times \overline{C} \times \overline{D}) + (A \times B \times \overline{C} \times \overline{D}) + (\overline{A} \times B \times \overline{C} \times D) + (A \times B \times \overline{C} \times D)$$

- **R/. Función lógica del ejercicio 2:**

  Se resolvió por Minterm (Σ)

$$F = (\overline{A} \times B \times \overline{C} \times \overline{D}) + (A \times B \times \overline{C} \times \overline{D}) + (\overline{A} \times \overline{B} \times C \times D) + (\overline{A} \times B \times C \times D)$$

- **R/. Función lógica del ejercicio 3:**

  Se resolvió por Maxterm (∏)

$$F = (A + B + C + \overline{D}) \times (\overline{A} + B + C + \overline{D}) \times (A + \overline{B} + C + \overline{D}) \times (\overline{A} + \overline{B} + C + \overline{D})$$

- **R/. Función lógica del ejercicio 4:**

  Se resolvió por Maxterm (∏)

$$F = (A + \overline{B} + C + D) \times (\overline{A} + \overline{B} + C + D) \times (A + \overline{B} + \overline{C} + D) \times (\overline{A} + \overline{B} + \overline{C} + D)$$

5.4.3 Basados en enunciados con condiciones

★ Ejercicio 1:

Cada combinación de las entradas en una tabla de verdad equivale a un número en código binario y tiene su equivalente en código decimal. ¿Diseñe un circuito combinacional que active un selector electromecánico cuando el número de entrada sea un número impar?. El sistema tiene 3 variables de entrada.

$$F(\Sigma) = (A \times \overline{B} \times \overline{C}) + (A \times B \times \overline{C}) + (A \times \overline{B} \times C) + (A \times B \times C)$$

$$F(\pi) = (\overline{A} + B + C) + (\overline{A} + \overline{B} + C) + (\overline{A} + B + \overline{C}) + (\overline{A} + \overline{B} + \overline{C})$$

R/. 0,1,0,1,0,1,0,1,0

★ Ejercicio 2:

*¿Diseñe el circuito para una alarma que se activa cuando se cumplen las condiciones **a** o **b**:*

a. La variable D esté en '0' y las variables B y C no estén en '1'

b. Las variables C y D estén en '1' y la variable A esté en '0'

$$F(\Sigma) = (\overline{A} \times \overline{B} \times \overline{C} \times \overline{D}) + (A \times \overline{B} \times \overline{C} \times \overline{D}) + (\overline{A} \times \overline{B} \times C \times D) + (\overline{A} \times B \times C \times D)$$

R/. 1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,0

★ Ejercicio 3:

¿Diseñe el circuito para que se desactive una cantonera eléctrica cuando las variables A y B no estén en nivel bajo, el sistema posee cuatro variables de entrada?

$$F(\pi) = (\overline{A} + \overline{B} + C + D) \times (\overline{A} + \overline{B} + \overline{C} + D) \times (\overline{A} + \overline{B} + C + \overline{D}) \times (\overline{A} + \overline{B} + \overline{C} + \overline{D})$$

R/. 1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,0

★ Ejercicio 4:

Se desea controlar dos motores M1 y M2 por medio de tres interruptores identificados como A, B y C, de forma que se cumplan las siguientes condiciones:

a. Si el interruptor A está pulsado y los otros dos no, se activa M1

b. Si el interruptor C está pulsado y los otros dos no, se activa M2

c. Si los tres interruptores están cerrados se activan M1 y M2

d. En las demás condiciones los dos motores estarán desactivados

$$M1(\Sigma) = (A \times \overline{B} \times \overline{C}) + (A \times B \times C)$$
$$M2(\Sigma) = (\overline{A} \times \overline{B} \times C) + (A \times B \times C)$$

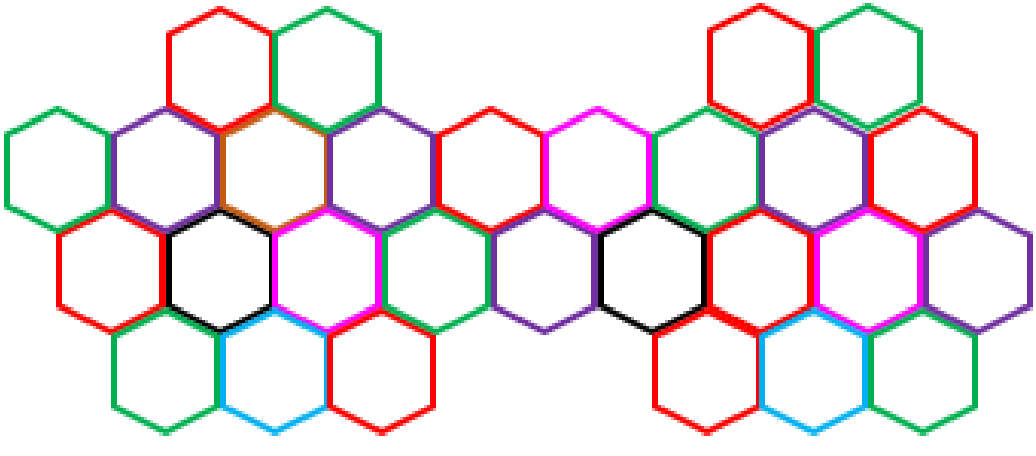
R/. M1 = 0,1,0,0,0,0,0,1 y M2 = 0,0,0,0,1,0,0,1

★ Ejercicio 5:

Cada combinación de las entradas en una tabla de verdad equivale a un número en código binario y tiene su equivalente en código decimal. ¿Diseñe el circuito que encienda un led (diodo emisor de luz) cuando cada combinación de entrada multiplicada por 9 de como resultado un número par; el diseño contiene tres variables? (el cero se toma como un número par)

$$M9(\Sigma) = (\overline{A} \times \overline{B} \times \overline{C}) + (\overline{A} \times B \times \overline{C}) + (\overline{A} \times \overline{B} \times C) + (\overline{A} \times B \times C)$$

R/. M9 = 1,0,1,0,1,0,1,0



AYUDAS

COMPLEMENTOS

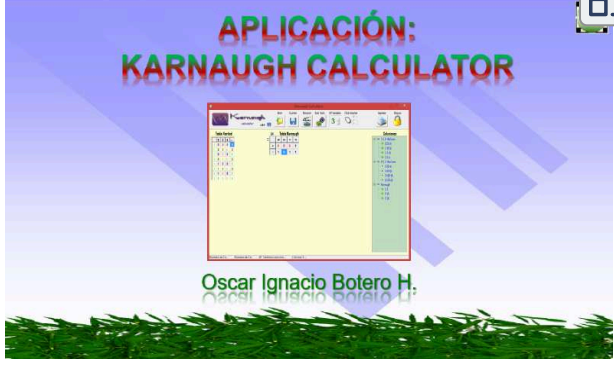
5.5 Videos

Video explicativo: Lógica AOI: Ejercicios 1, Circuitos Combinacionales - Soluciones y Simulaciones Proteus:, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo: Lógica AOI: Ejercicios 2, Circuitos Combinacionales - Soluciones y Simulaciones Proteus:, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo: Lógica AOI: Ejercicios 3, Circuitos Combinacionales - Soluciones y Simulaciones Proteus:, en el canal de youtube de Oscar Ignacio Botero Henao @oscarboterohenao

Video explicativo sobre el manejo de la aplicación ***"Karnaugh Calculator"***. Realizado por Oscar Ignacio Botero Henao.[12]



[12] La aplicación es portable y aquí la puede descargar comprimida.

5.6 Objetos Interactivos

Objetos Interactivos con algunos de los logotipos de software para la simulación de circuitos digitales y de términos utilizados en lógica AOI.[13]

[13] Plantillas de los Objetos Interactivos obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Haga clic sobre la imagen para abrir el objeto interactivo:

Este objeto interactivo fue generado con la *asistencia* de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/

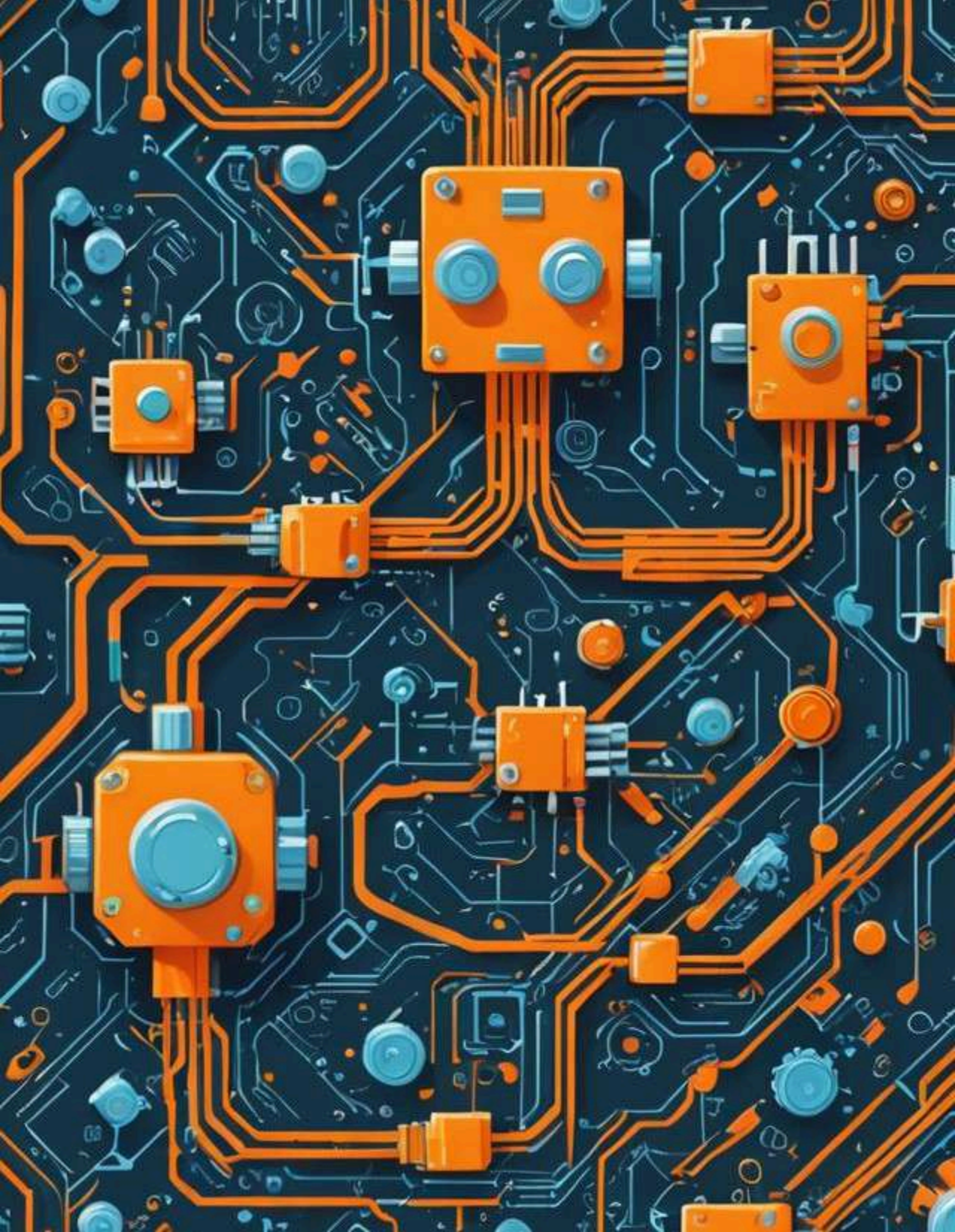
Cuestionario:

5.7 Simulaciones

Clic en la figura para acceder al repositorio de GitHub donde se encuentran los archivos de simulación de los ejemplos tratados en este capítulo:

Capítulo VI

Mapas de Karnaugh



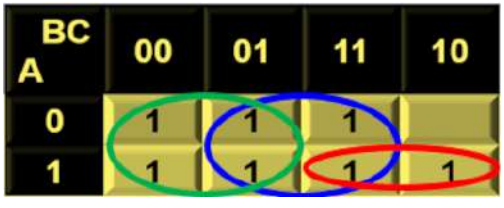
Los pasos que se siguen en la elaboración de un mapa de Karnaugh son: [1][3][4][7][8][9]

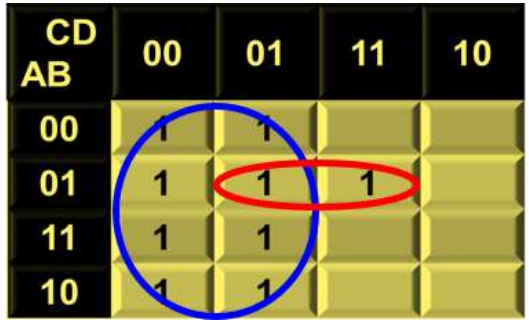
1. Obtener una expresión booleana en forma de minterm (Σ) o maxterm (∏)
2. Colocar '1' o '0' en el mapa de Karnaugh de acuerdo a la expresión (mapa de Karnaugh por minterm o por maxterm)
    a. No se pueden generar grupos en diagonal, solo de forma vertical y horizontal
    b. Los unos o los ceros pueden compartir grupos
    c. Unir los dos extremos, ya sea horizontal o verticalmente, para considerar las celdas de las esquinas como celdas adyacentes (doblar el mapa)
    d. Cuanto más grande sea un grupo, más simplificada va ser la función
3. Agrupar los conjuntos de valores adyacentes de unos ('1') o ceros ('0'), en potencias de 2: grupos de 2, 4, 8, 16, 32...
    a. Se encierran los '1' o '0' que **NO** sean adyacentes con otros del mismo valor, se llaman islas
    b. Se encierran los '1' o '0' que formen grupos de dos pero que NO formen grupos de cuatro '1' o '0'
    c. Se encierran los '1' o '0' que formen grupos de cuatro pero que NO formen grupos de ocho '1' o '0'
    d. Así sucesivamente hasta que todos los '1' o '0' del mapa sean agrupados
4. Agrupar los valores adyacentes:
    a. Se tienen en cuenta las variables adyacentes que **NO** cambian su estado lógico dentro de la agrupación

b. Se eliminan las variables que **SI** cambian su estado lógico dentro de la agrupación (eliminar los complementos)

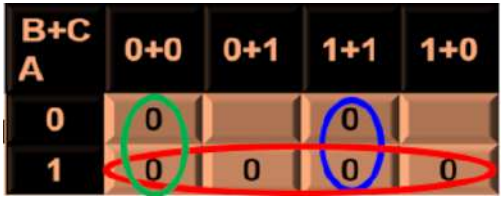
5. Enlazar con operadores **OR** los grupos obtenidos para formar la expresión simplificada en forma de minterm (Σ) y con operadores **AND** en forma de maxterm (∏)

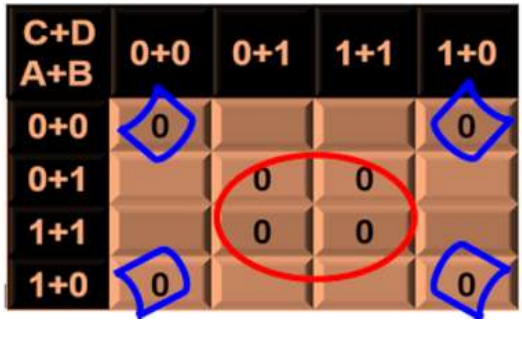
Mapas por Minterm (Σ):





Mapas por Maxterm (∏):





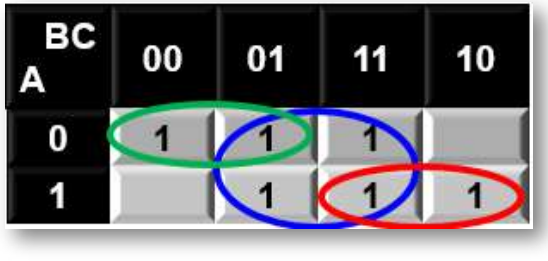
6.1 Ejemplos basados en la tabla de verdad

¿Con base en las siguientes tablas de verdad, halle las funciones lógicas simplificadas utilizando el método de Mapas de Karnaugh (mapas K) e implemente los circuitos combinacionales en un software de simulación electrónica y verifique las soluciones?

★ Ejemplo 1:

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

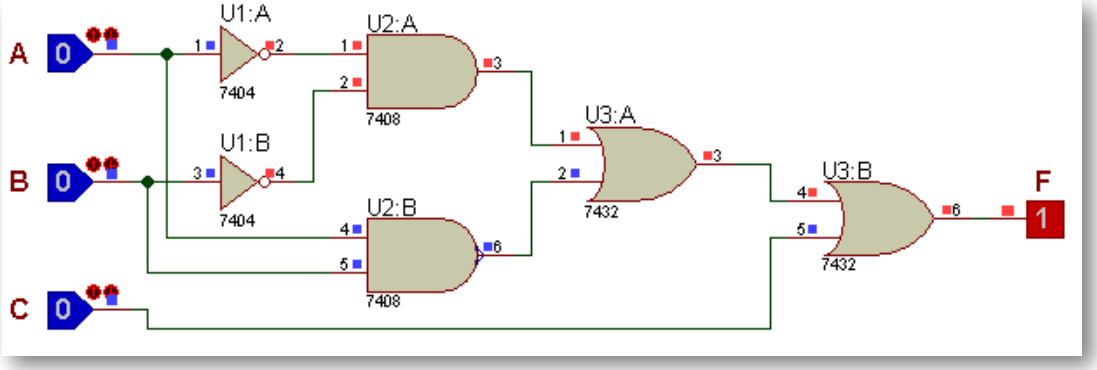
Se bajan los '1' de la tabla de verdad a las celdas correspondientes del mapa de Karnaugh de 3 variables por ***"MINTERM"*** y se agrupan:



La función simplificada es:

$$F(\Sigma) = (\overline{A} \times \overline{B}) + (A \times B) + C$$

El circuito implementado con la función simplificada es:



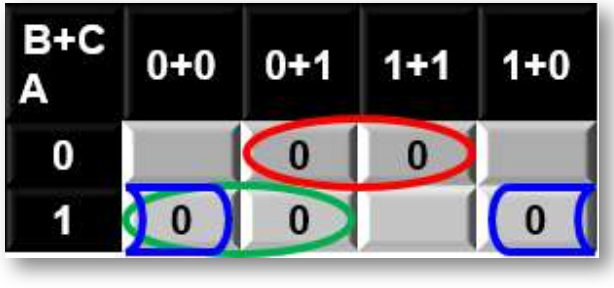
En la tabla de verdad hay seis '1' que pertenecen a 6 términos minterm y al simplificarla por el método de mapas K queda reducida la función a 3 términos y uno de ellos contiene una sola variable; de esta forma se evidencia la simplificación.

Repositorio en Github (Ejemplo 1-1.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

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

Se bajan los '0' de la tabla de verdad a las celdas correspondientes del mapa de Karnaugh de 3 variables por ***"MAXTERM"*** y se agrupan:



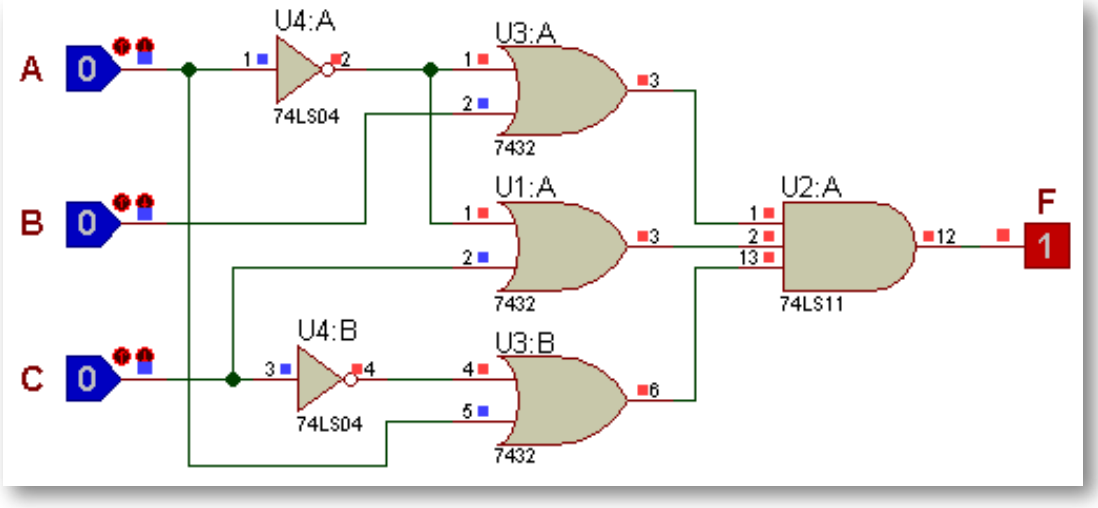
La función simplificada es:

$$F(\Pi) = (A + \overline{C}) \times (\overline{A} + B) \times (\overline{A} + C)$$

El primer término (color verde) y el último (color azul) equivalen a la función lógica de la compuerta XOR:

$$F(\Pi) = (A \oplus C) \times (\overline{A} + B)$$

El circuito implementado con la función simplificada es:



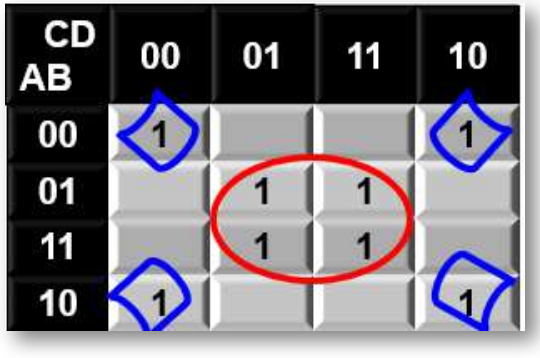
En la tabla de verdad hay cinco '0' que pertenecen a 5 términos maxterm y al simplificarla por el método de mapas K queda reducida la función a 3 términos; de esta forma se evidencia la simplificación.

Repositorio en Github (Ejemplo 1-2.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

D	C	B	A	F
0	0	0	0	*1*
0	0	0	1	*1*
0	0	1	0	*0*
0	0	1	1	*0*
0	1	0	0	*1*
0	1	0	1	*1*
0	1	1	0	*0*
0	1	1	1	*0*
1	0	0	0	*0*
1	0	0	1	*0*
1	0	1	0	*1*
1	0	1	1	*1*
1	1	0	0	*0*
1	1	0	1	*0*
1	1	1	0	*1*
1	1	1	1	*1*

Se bajan los '1' de la tabla de verdad a las celdas correspondientes del mapa de Karnaugh de 4 variables por ***"MINTERM"*** y se agrupan:



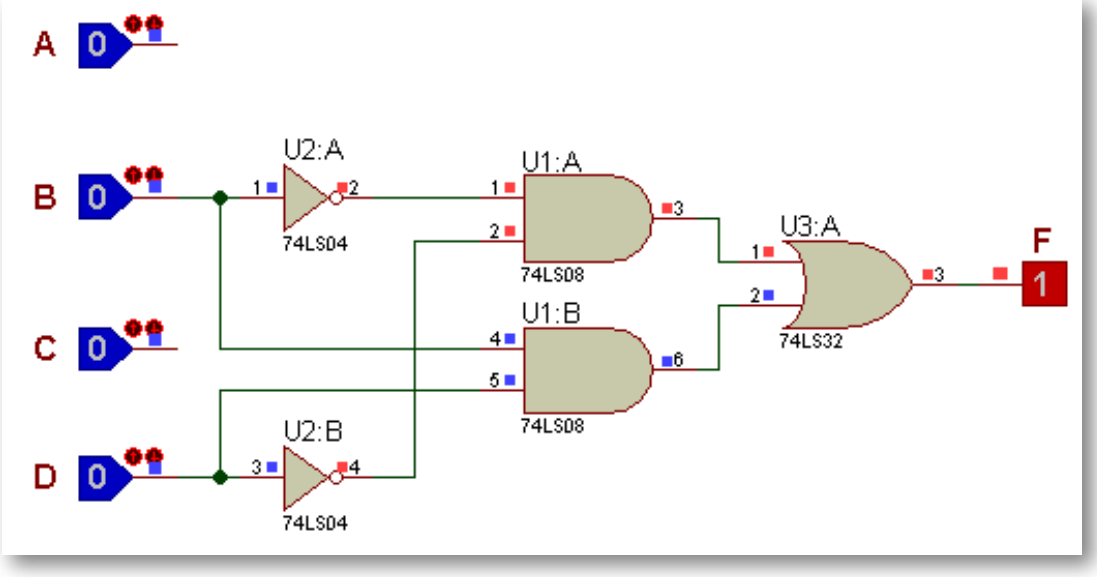
La función simplificada es:

$$F(\Sigma) = (\overline{B} \times \overline{D}) + (B \times D)$$

La función simplificada es equivalente a la función lógica de la compuerta XNOR:

$$F(\Sigma) = \overline{B \oplus D}$$

El circuito implementado con la función simplificada es:



En la tabla de verdad hay ocho '1' que pertenecen a 8 términos minterm y al simplificarla por el método de mapas K queda reducida la función a 2 términos; de esta forma se evidencia la simplificación.

Repositorio en Github (Ejemplo 1-3.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 4:

D	C	B	A	F
0	0	0	0	*0*
0	0	0	1	*0*
0	0	1	0	*1*
0	0	1	1	*1*
0	1	0	0	*0*
0	1	0	1	*0*
0	1	1	0	*1*
0	1	1	1	*1*
1	0	0	0	*0*
1	0	0	1	*1*
1	0	1	0	*1*
1	0	1	1	*1*
1	1	0	0	*1*
1	1	0	1	*1*
1	1	1	0	*1*
1	1	1	1	*1*

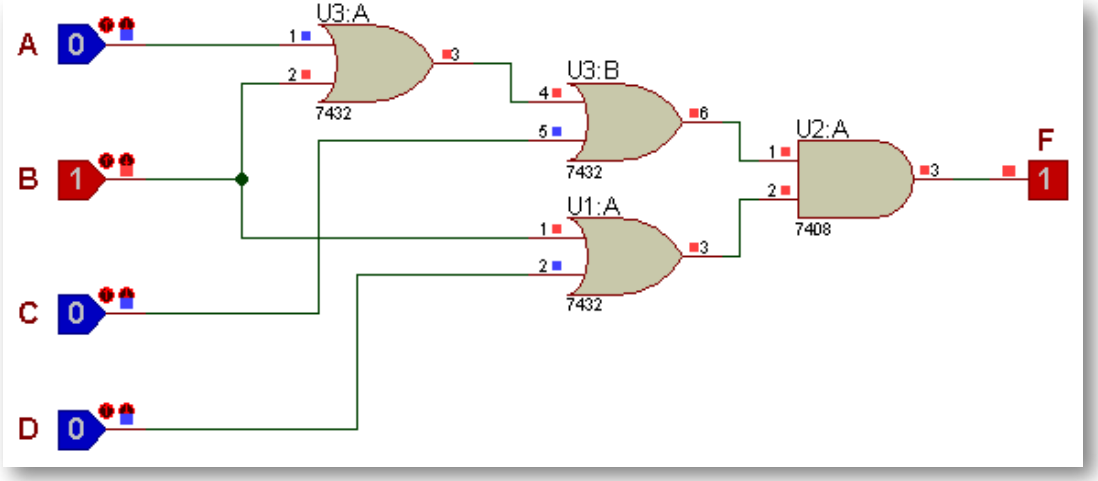
Se bajan los '0' de la tabla de verdad a las celdas correspondientes del mapa de Karnaugh de 4 variables por ***"MAXTERM"*** y se agrupan:

C+D A+B	0+0	0+1	1+1	1+0
0+0	0	0		*0*
0+1				
1+1				
1+0	0			*0*

La función simplificada es:

$$F(\Pi) = (A + B + C) \times (B + D)$$

El circuito implementado con la función simplificada es:



En la tabla de verdad hay cinco '0' que pertenecen a 5 términos maxterm y al simplificarla por el método de mapas K queda reducida la función a 2 términos; de esta forma se evidencia la simplificación.

Repositorio en Github (Ejemplo 1-4.dsn) con el archivo de la simulación con Proteus v7.8

6.2 Ejemplo con secuencia de números y display

★ Secuencia numérica 3-0-2-1 y display 7 segmentos:

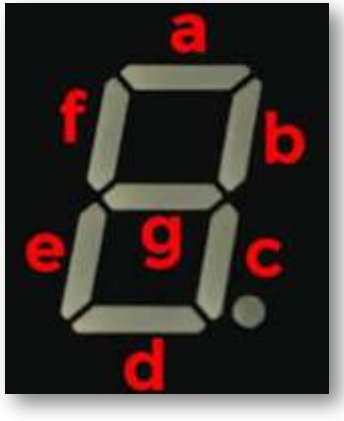
¿Halle la tabla de verdad completa, las funciones lógicas simplificadas por mapas de Karnaugh para cada segmento del display e implemente todo el circuito en un software de simulación electrónica; de forma que, al recorrer la tabla de verdad en orden ascendente muestre en un display de 7 segmentos de cátodo común la secuencia de números 3-0-2-1 y cuando se complete se encienda un led para indicar que finalizó la secuencia numérica?

- Como el display es de cátodo común, a los segmentos que encienden se les colocan un '1' lógico.
- Cada mapa es independiente y por ello cada uno se puede simplificar por ***"MINTERM"*** o por ***"MAXTERM".***
- La secuencia numérica está forma por 4 números y para recorrerla se requieren 2 variables ya que $2^2 = 4$.
- Se bajan los '1' o '0' de la tabla de verdad de cada segmento a las celdas correspondientes dentro de los mapas de Karnaugh de 2 variables (A y B).
- Verificar que pueden resultar mapas que se repiten. En este caso los mapas para los segmentos ***a*** y ***d*** son los mismos; entonces, se realiza un solo mapa y por ende se obtiene una sola función lógica resultante que controlará los dos segmentos del display.
- Para activar el LED se requiere un '1' en el número $3_d = 11_b$ y en el mapa de Karnaugh se refleja como una isla.
- La siguiente figura muestra los números decimales y los segmentos que encienden:

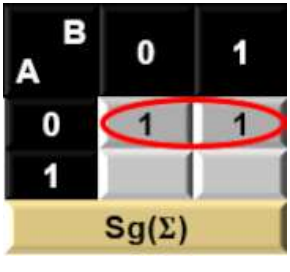
La tabla de verdad completa es:

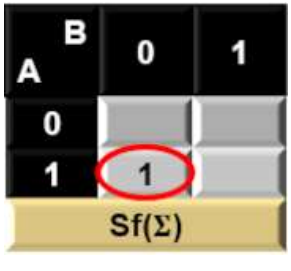
No.	Entradas		Segmentos del Display CC							No.	Sal
Dec	B	A	Sg	Sf	Se	Sd	Sc	Sb	Sa	Dis	Led
0	0	0	1	0	0	1	1	1	1	3	0
1	0	1	0	1	1	1	1	1	1	8	0
2	1	0	1	0	1	1	0	1	1	2	0
3	1	1	0	0	0	0	1	1	0	1	1

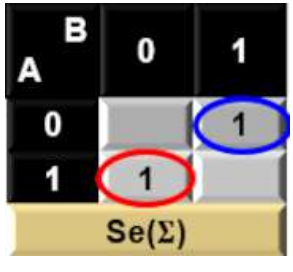
La asignación de los segmentos del display es:

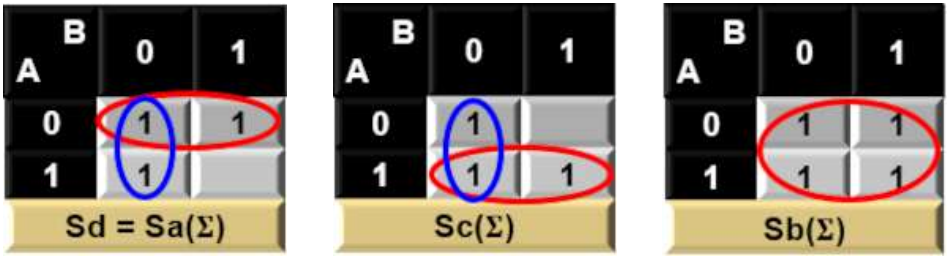


Los mapas de Karnaugh:









Funciones simplificadas por medio de los mapas de Karnaugh:

$$Sg(\Sigma) = (\overline{A})$$

$$Sf(\Sigma) = (A \times \overline{B})$$

$$Se(\Sigma) = (A \times \overline{B}) + (\overline{A} \times B) = (A \oplus B)$$

$$Sd = Sa(\Sigma) = \overline{A} + \overline{B}$$

$$Sc(\Sigma) = A + \overline{B}$$

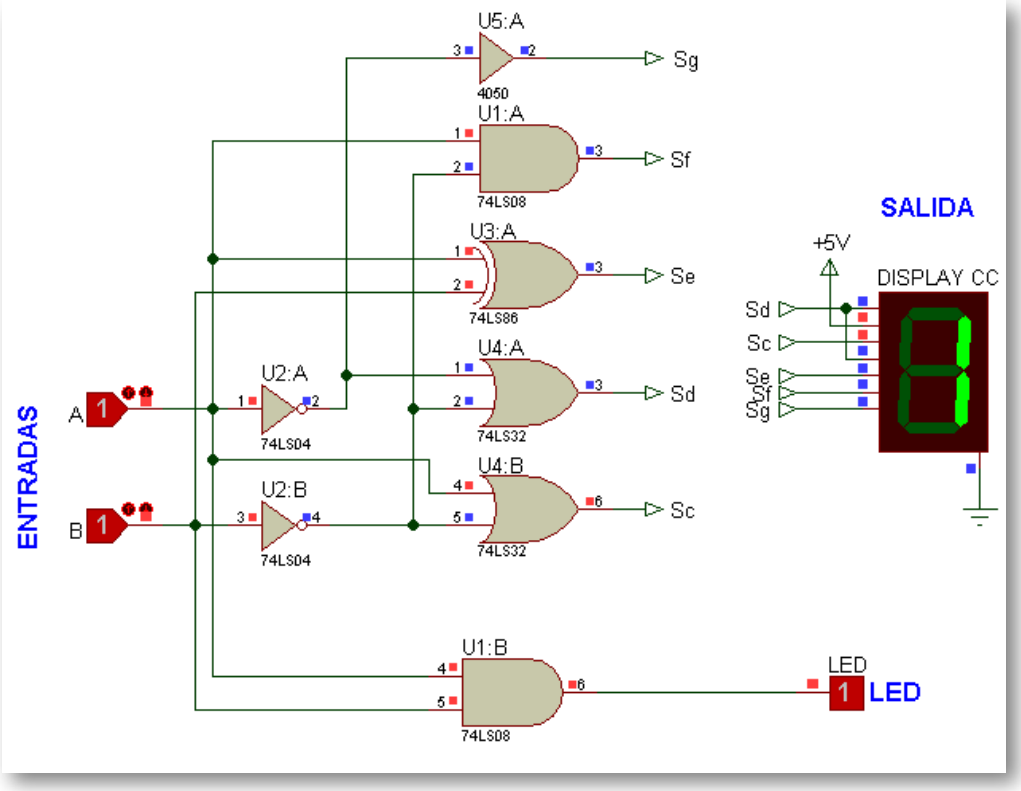
$$Sb(\Sigma) = +Vcc = +5v$$

El mapa de Karnaugh para controlar del LED (Diodo Emisor de Luz) estaría formado por una isla y es equivalente a leer este dato como un término Minterm:

$$Led(\Sigma) = (A \times B)$$

Repositorio en Github (Ejemplo 2.dsn) con el archivo de la simulación con Proteus v7.8

Plano esquemático (SCH) del circuito completo:

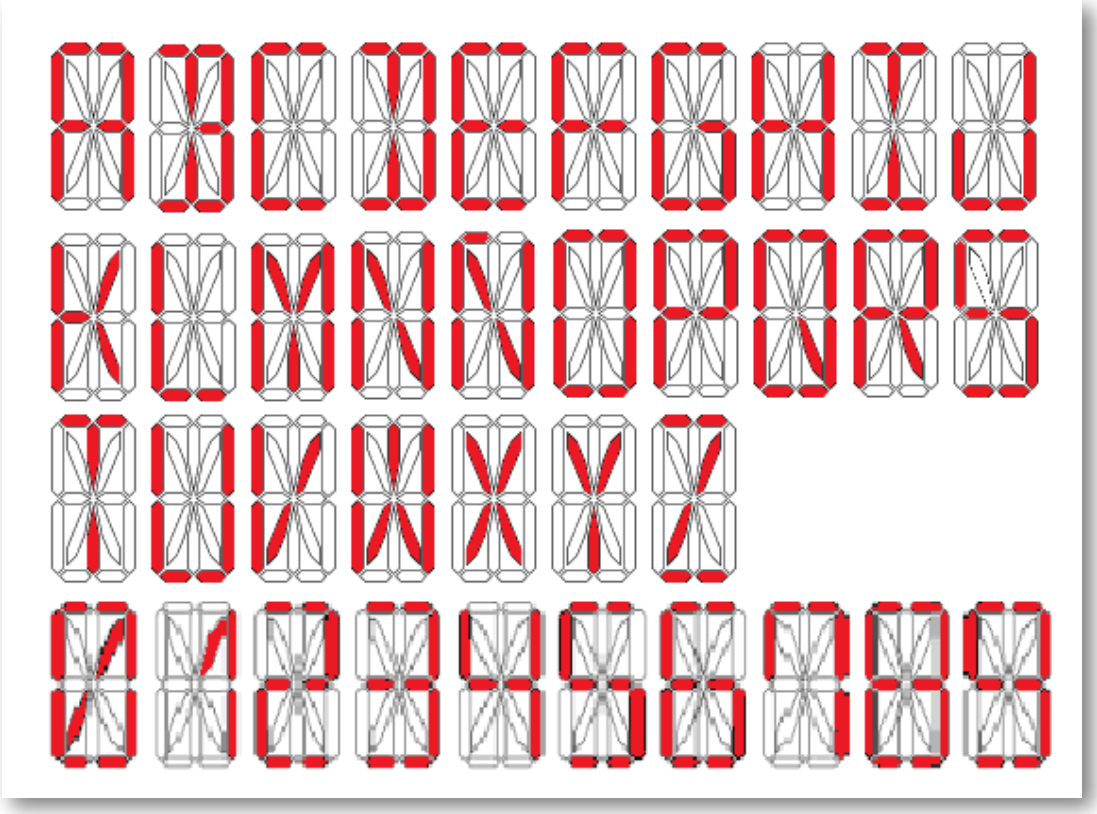


6.3 Ejemplo con un texto y display alfanumérico

★ Palabra "SUMA" y display Alfanumérico:

¿Halle la tabla de verdad completa, las funciones lógicas simplificadas por mapas de Karnaugh para cada segmento del display Alfanumérico e implemente todo el circuito en un software de simulación electrónica; de forma que, al recorrer la tabla de verdad en orden ascendente muestre en un display alfanumérico de 16 segmentos de cátodo común la palabra "SUMA" y con el último carácter debe encender el punto decimal (DP)?

- En la siguiente figura se visualizan todos los caracteres para el display alfanumérico de 16 segmentos:



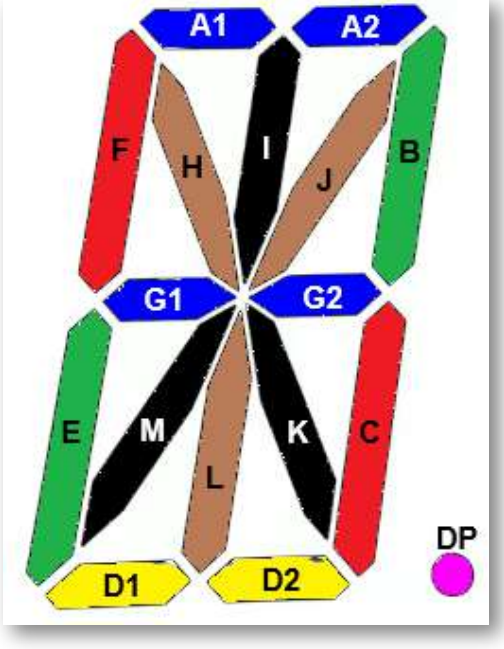
- El display alfanumérico es de cátodo común y los segmentos que se requieran encender se les envía un '1' lógico.
- Como cada mapa es independiente se puede simplificar por ***"MINTERM"*** o por ***"MAXTERM"*** cada uno.
- La palabra "SUMA" está forma por 4 caracteres o letras y para recorrerla se requieren 2 variables ya que $2^2 = 4$ combinaciones.
- Se bajan los '1' o '0' de la tabla de verdad de cada segmento a las celdas correspondientes dentro de los mapas de Karnaugh de 2 variables (A y B).

Se observa que hay segmentos donde se repiten todos los datos. Para este ejemplo se repiten: ***Sa1=Sa2=Sg1=Sg2***, ***Sb=Se***, ***Sc=Sf***, ***Sd1=Sd2***, ***Sh=Sj=Sl*** y ***Si=Sk=Sm***; entonces, se realiza un solo mapa para los segmentos repetidos y se obtiene una sola función lógica resultante que controlará esos segmentos del display.

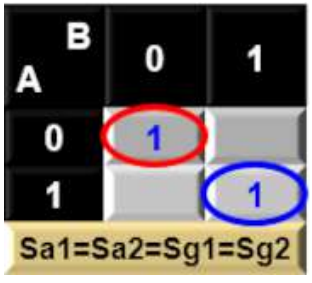
Para activar el punto decimal (DP) se requiere un '1' en el número $3_d = 11_b$ y en el mapa de Karnaugh se refleja como una isla.

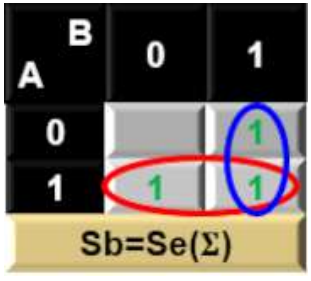
	Entradas		Segmentos del display alfanumérico																	
No.	B	A	Sa1	SA2	Sb	Sc	Sd1	Sd2	Se	Sf	Sg1	Sg2	Sh	Si	Sj	Sk	Sl	Sm	DP	
0	0	0	1	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	S
1	0	1	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	U
2	1	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	0	M
3	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	1	A.

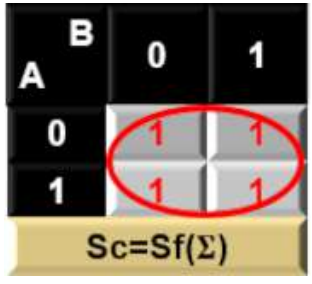
La distribución de los segmentos del display alfanumérico de 16 segmentos es:

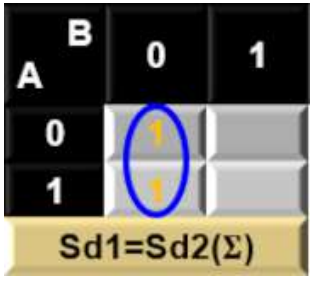


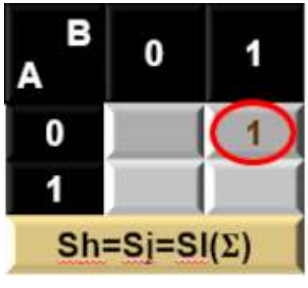
Mapas de Karnaugh y sus agrupaciones:

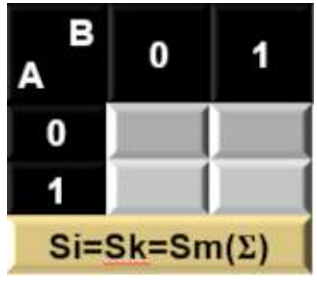












Funciones simplificadas por medio de los mapas de Karnaugh:

$$Sa1 = Sa2 = Sg1 = Sg2(\Sigma) = (\overline{A} \times \overline{B}) + (A \times B)$$

$$= (\overline{A \oplus B})$$

$$Sb = Se(\Sigma) = (A) + (B)$$

$$Sc = Sf(\Sigma) = +Vcc = +5v$$

$$Sd1 = Sd2(\Sigma) = (\overline{B})$$

$$Sh = Sj = Sl(\Sigma) = (\overline{A} \times B)$$

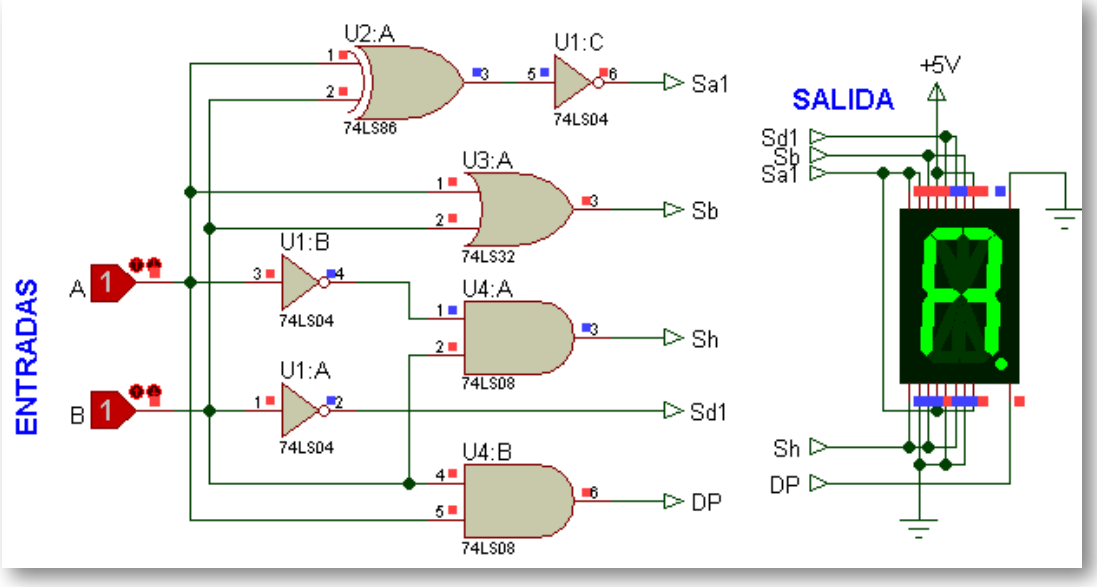
$$Si = Sk = Sm(\Sigma) = Gnd = 0v$$

El mapa de Karnaugh para controlar el punto decimal (DP) estaría formado por una isla y es equivalente a leer este dato como un término Minterm:

El mapa de Karnaugh para controlar el punto decimal (DP) estaría formado por una isla y es equivalente a leer este dato como un término Minterm:

$$DP(\Sigma) = (A \times B)$$

Plano esquemático (SCH) del circuito completo:



Repositorio en Github (Ejemplo 3.dsn) con el archivo de la simulación con Proteus v7.8

6.4 Ejercicios propuestos con respuestas

★ Ejercicios 1 y 2:

¿Con base en las 2 tablas de verdad y utilizando el método de mapas de Karnaugh; hallar las funciones lógicas simplificadas al máximo, la primera tabla por Minterm (Σ) y la segunda por maxterm (∏). Finalmente, implemente los dos circuitos en un software de simulación electrónica para sus respectivas verificaciones?

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

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

R/. Mapas de Karnaugh y sus funciones lógicas simplificadas, la primera por minterm y la segunda por maxterm:

BC / A	00	01	11	10
0	1			1
1	1		1	1

$$F(\Sigma) = (A \times B) + \overline{C}$$

B+C / A	0+0	0+1	1+1	1+0
0	0	0	0	0
1			0	0

$$F(\Pi) = A \times \overline{B}$$

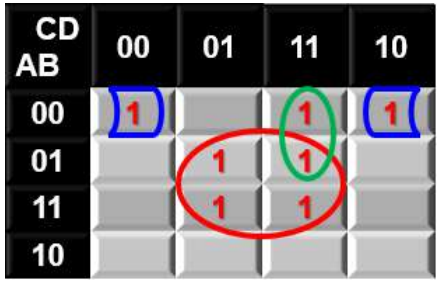
★ Ejercicio 3:

¿Con base en las 2 tablas de verdad y utilizando el método de mapas de Karnaugh; hallar las funciones lógicas simplificadas al máximo, la primera por Minterm (Σ) y la segunda por maxterm (∏). Finalmente, implemente los dos circuitos con un software de simulación electrónica para sus respectivas verificaciones?

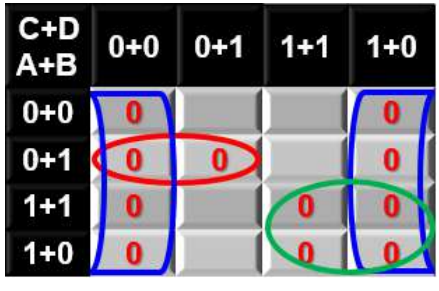
D	C	B	A	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

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

R/. Mapas de Karnaugh y sus funciones lógicas simplificadas, la primera por minterm y la segunda por maxterm:



$$F(\Sigma) = (B \times D) + (\overline{A} \times C \times D) + (\overline{A} \times \overline{B} \times \overline{D})$$



$$F(\Pi) = (A + \overline{B} + C) \times (\overline{A} + \overline{C}) \times D$$

★ Ejercicio 4:

¿Halle la tabla de verdad completa, las funciones lógicas simplificadas por mapas de Karnaugh para cada segmento del display de 7 segmentos de cátodo común e implemente todo el circuito en un software de simulación electrónica para su verificación; de forma que, al recorrer la tabla de verdad en orden ascendente se visualice la secuencia de los números del ***"Documento o Código de Identificación"?.***

R/. Las respuestas son personalizadas debido a que el número de la identificación es única

★ Ejercicio 5:

¿Halle la tabla de verdad completa, las funciones lógicas simplificadas por mapas de Karnaugh para cada segmento de un display Alfanumérico de cátodo común de 16 segmentos e implemente todo el circuito en un software de simulación electrónica; de forma que, al recorrer la tabla de verdad en orden ascendente muestre su ***"NOMBRE PROPIO"*** *y con el último carácter debe encender el punto decimal (DP)?*

R/. Las respuestas son personalizadas debido a su nombre propio.

★ Ejercicio 6:

¿Diseñe un contador decimal de 0 a 9 a base de compuertas lógicas (circuito combinacional) que se visualice en un display de 7 segmentos de ***"Ánodo Común"****; para las combinaciones de entrada que no se utilicen sus respectivas salidas deben mostrar el display apagado. Halle la tabla de verdad completa, las funciones simplificadas por mapas de Karnaugh y verifique su funcionamiento mediante un software de simulación electrónica?*

- *La siguiente figura muestra los números decimales y los segmentos que encienden:*

R/. Tabla de verdad:

No.	Entradas				Segmentos del Display AC							No.	Sal
Dec	D	C	B	A	Sg	Sf	Se	Sd	Sc	Sb	Sa	Dis	Led
0	0	0	0	0	1	0	0	0	0	0	0	8	0
1	0	0	0	1	1	1	1	1	0	0	1	1	0
2	0	0	1	0	0	1	0	0	1	0	0	2	0
3	0	0	1	1	0	1	1	0	0	0	0	3	0
4	0	1	0	0	0	0	1	1	0	0	1	4	0
5	0	1	0	1	0	0	1	0	0	1	0	5	0
6	0	1	1	0	0	0	0	0	0	1	1	6	0
7	0	1	1	1	1	1	1	1	0	0	0	7	0
8	1	0	0	0	0	0	0	0	0	0	0	8	0
9	1	0	1	1	0	0	1	1	0	0	0	9	1
10-15	-	-	-	-	1	1	1	1	1	1	1		0

R/. Cada segmento y su mapa son independientes; es por ello que unas funciones pueden ser simplificadas por Minterm y otras por Maxterm. A continuación, se muestran algunas posibles respuestas:

$$Sg(\Sigma) = (C \times D) + (B \times D) + (\overline{B} \times \overline{C} \times \overline{D}) + (A \times B \times C)$$

$$Sf(\Sigma) = (D \times C) + (B \times \overline{C}) + (A \times B) + (A \times \overline{C} \times \overline{D})$$

$$Se(\Sigma) = A + (B \times D) + (\overline{B} \times C)$$

$$Sd(\Sigma) = (B \times D) + (C \times D) + (\overline{A} \times \overline{B} \times C) + (A \times \overline{B} \times \overline{C}) + (A \times B \times C)$$

$$Sc(\Sigma) = (C \times D) + (B \times D) + (\overline{A} \times B \times \overline{C})$$

$$Sb(\Sigma) = (B \times D) + (C \times D) + (\overline{A} \times B \times C) + (A \times \overline{B} \times C)$$

$$Sa(\Sigma) = (\overline{A} \times C) + (B \times D) + (C \times D) + (A \times \overline{B} \times \overline{C} + \overline{D})$$

★ Ejercicio 7:

¿Diseñe un circuito combinacional con 3 variables de entrada. Cada combinación tiene su equivalente en código decimal y debe generar en código ***"BINARIO"*** *el cuadrado del valor de la entrada?*

R/. Tabla de verdad:

Entradas				X^2	2^5	2^4	2^3	2^2	2^1	2^0
No.	Binarias			R/.	32	16	8	4	2	1
dec	C	B	A	dec	F_5	F_4	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

R/. Funciones lógicas simplificadas por mapas de Karnaugh:

$$F_5(\Sigma) = (B \times C)$$

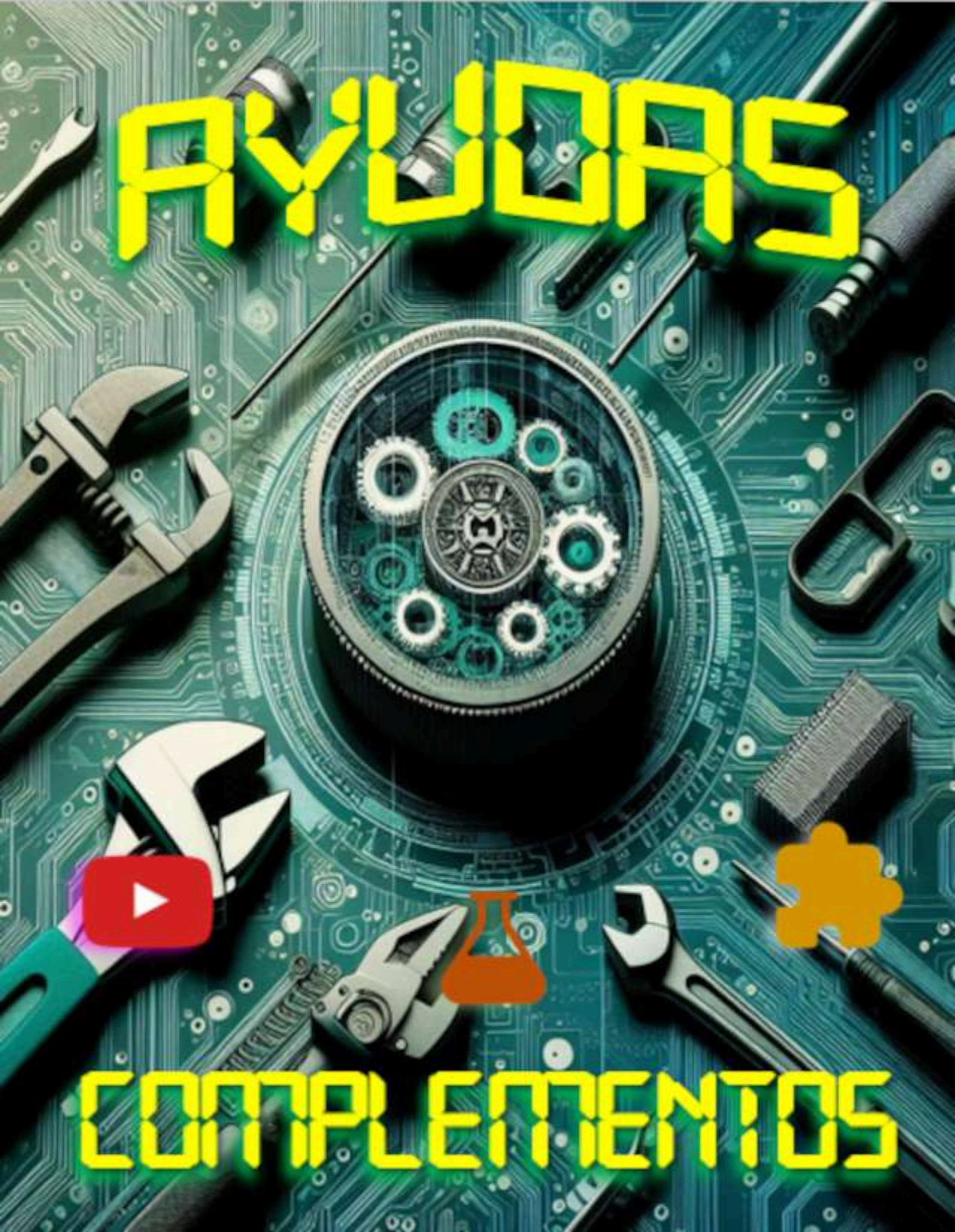
$$F_4(\Sigma) = (\overline{B} \times C) + (A \times C)$$

$$F_3(\Sigma) = (A \times \overline{B} \times C) + (A \times B \times \overline{C})$$

$$F_2(\Sigma) = (\overline{A} \times B)$$

$$F_1(\Sigma) = GND = 0V$$

$$F_0(\Sigma) = A$$


AYUDAS
COMPLEMENTOS

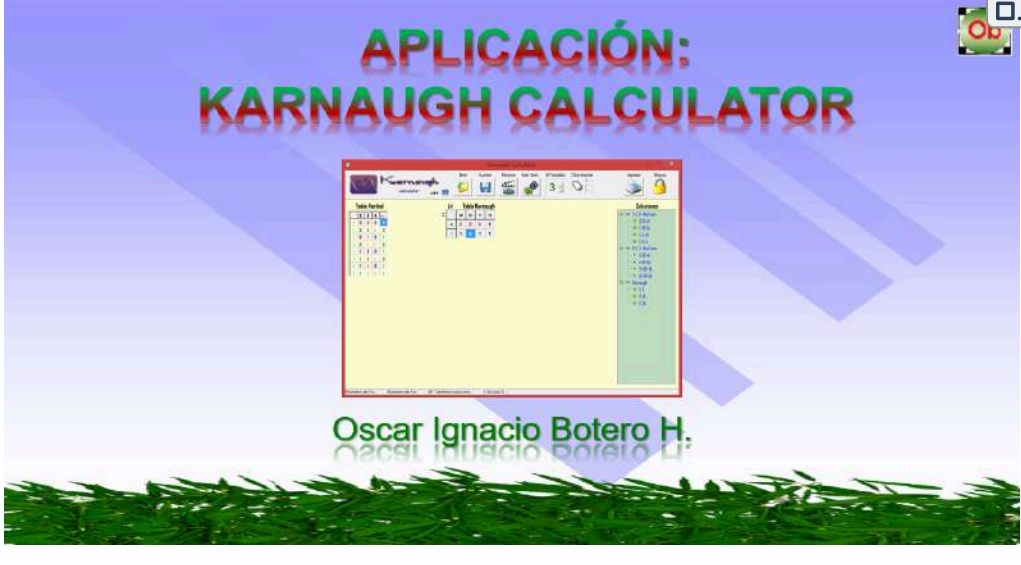
6.5 Videos

Video explicativo: Secuencia Numérica, Circuito Combinacional - Solución y Simulación Proteus:, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo: Conexión Display Alfanumérico - Simulación Proteus:, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo: Display Alfanumérico, Circuito Combinacional - Solución y Simulación Proteus:, en el canal de youtube de Oscar Botero @oscarboterohenao

Video explicativo realizado por Oscar Botero sobre el manejo de la aplicación ***"Karnaugh Calculator"***.



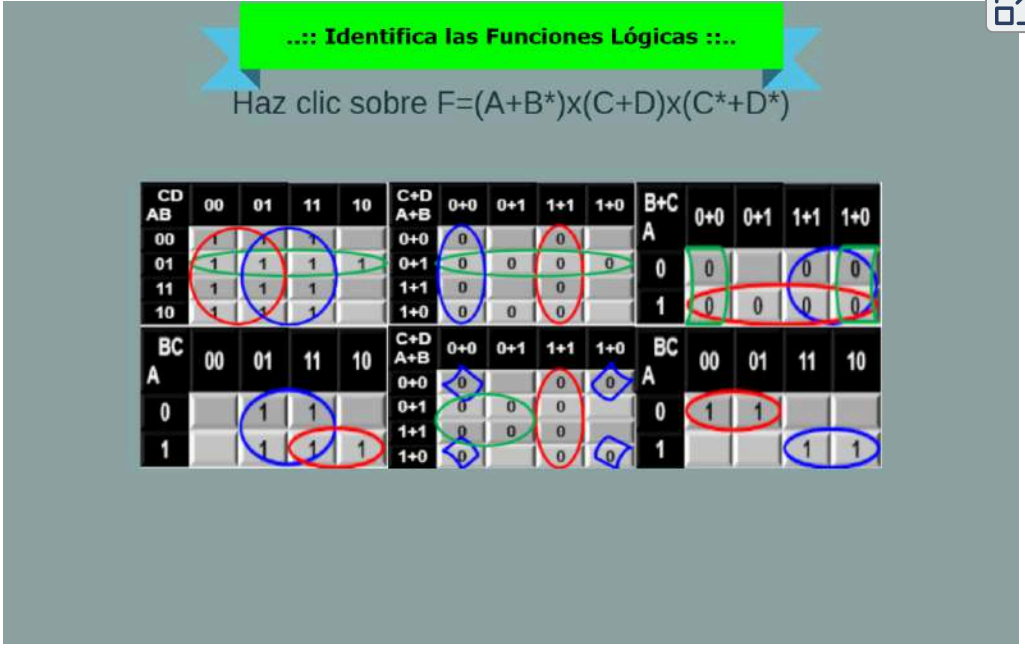
6.6 Objetos Interactivos

Objetos Interactivos con mapas de Karnaugh de 4 variables. ***El asterisco (*) significa que la variable que le antecede está negada.***[14]

[14] Plantilla de los Objetos Interactivos obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Objeto Interactivo con 6 funciones lógicas generadas por los mapas de Karnaugh. ¿Haz clic sobre el mapa de Karnaugh al que corresponde la función lógica mostrada? [15]

El asterisco (*) en las funciones significa que la variable que le antecede está negada.



[15] Plantilla del Objeto Interactivo obtenida del sitio web del Proyecto Descartes, intervenido por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Haga clic sobre la imagen para abrir el objeto interactivo:

Este objeto interactivo fue generado con la *asistencia* de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/

Cuestionario:

6.7 Simuladores y Simulaciones

➤ El programa Karnaugh Calculator- Minimizador es una versión Beta que permite ingresar los datos en una tabla de verdad (3 a 8 variables), para generar las funciones por Minterm, por Maxterm y simplificar por mapas de Karnaugh:

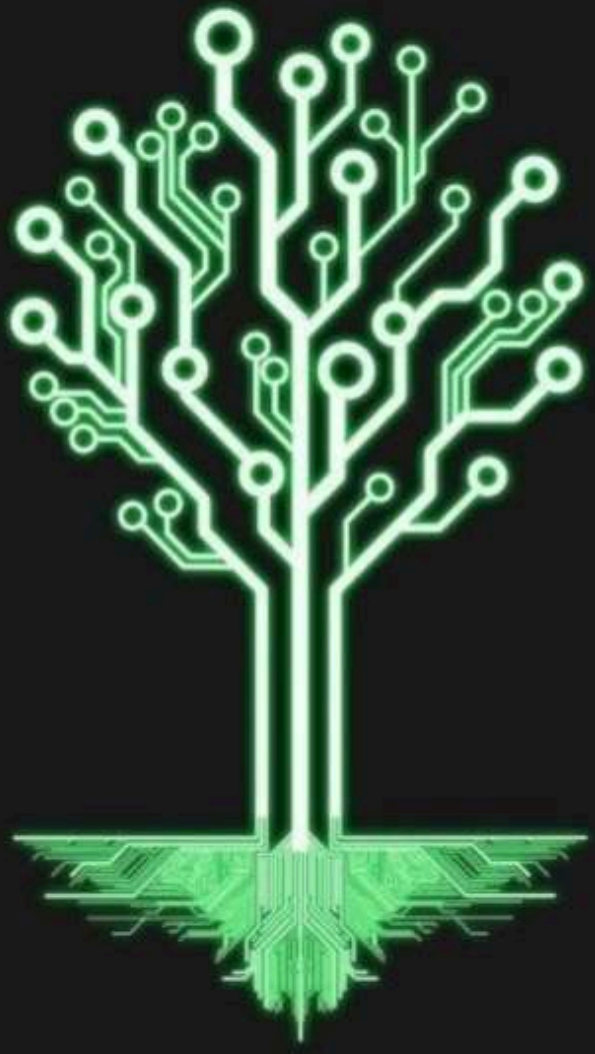
➤ Sublime.tool es una aplicación en línea que soluciona mapas de Karnaugh por minterm o por maxterm:

➤ Solucionador de mapas de Karnaugh en línea, genera las funciones por Minterm, por Maxterm, el código en lenguajes: VHDL y Verilog:

➤ APP "Solución para mapas de Karnaugh" de Adriano Moutinho para 3 o 4 variables, muestra varias versiones de circuitos lógicos optimizados de salida: el tradicional, solo con: inversores, compuertas NAND o compuertas NOR:

Repositorio en Github con los archivos de las simulaciones de este capítulo con Proteus v7.8

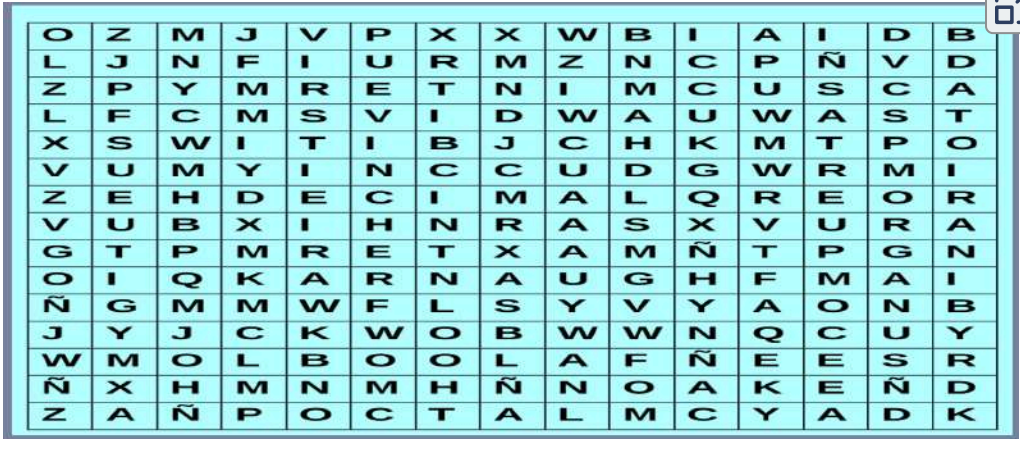

Ob
REPASO
REPASO



Actividades de Repaso

Sopa de letras con 10 términos pertenecientes al área de la Electrónica Digital y que fueron usados en este libro ***(volumen 1).***

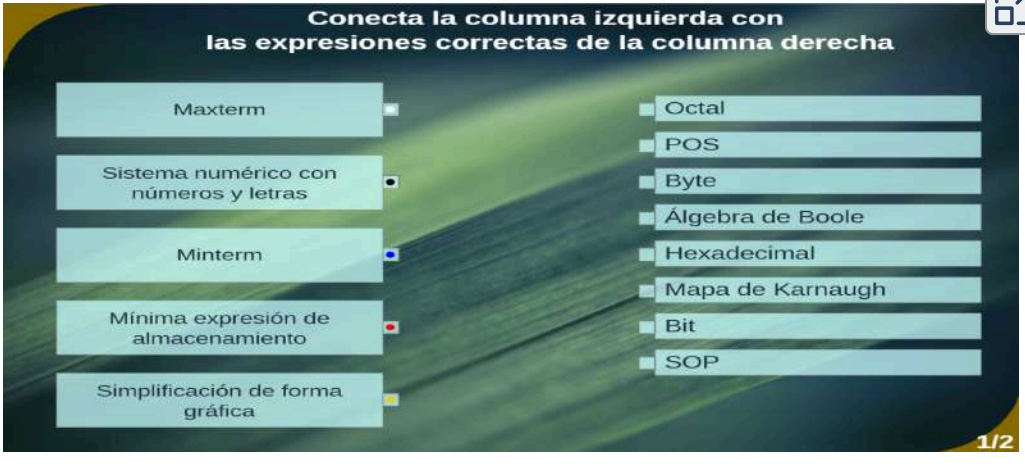
Preguntas de selección múltiple con única respuesta. [16]



O	Z	M	J	V	P	X	X	W	B	I	A	I	D	B
L	J	N	F	I	U	R	M	Z	N	C	P	Ñ	V	D
Z	P	Y	M	R	E	T	N	I	M	C	U	S	C	A
L	F	C	M	S	V	I	D	W	A	U	W	A	S	T
X	S	W	I	T	I	B	J	C	H	K	M	T	P	O
V	U	M	Y	I	N	C	C	U	D	G	W	R	M	I
Z	E	H	D	E	C	I	M	A	L	Q	R	E	O	R
V	U	B	X	I	H	N	R	A	S	X	V	U	R	A
G	T	P	M	R	E	T	X	A	M	Ñ	T	P	G	N
O	I	Q	K	A	R	N	A	U	G	H	F	M	A	I
Ñ	G	M	M	W	F	L	S	Y	V	Y	A	O	N	B
J	Y	J	C	K	W	O	B	W	W	N	Q	C	U	Y
W	M	O	L	B	O	O	L	A	F	Ñ	E	E	S	R
Ñ	X	H	M	N	M	H	Ñ	N	O	A	K	E	Ñ	D
Z	A	Ñ	P	O	C	T	A	L	M	C	Y	A	D	K

[16] Las plantillas para los Objetos Interactivos fueron obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

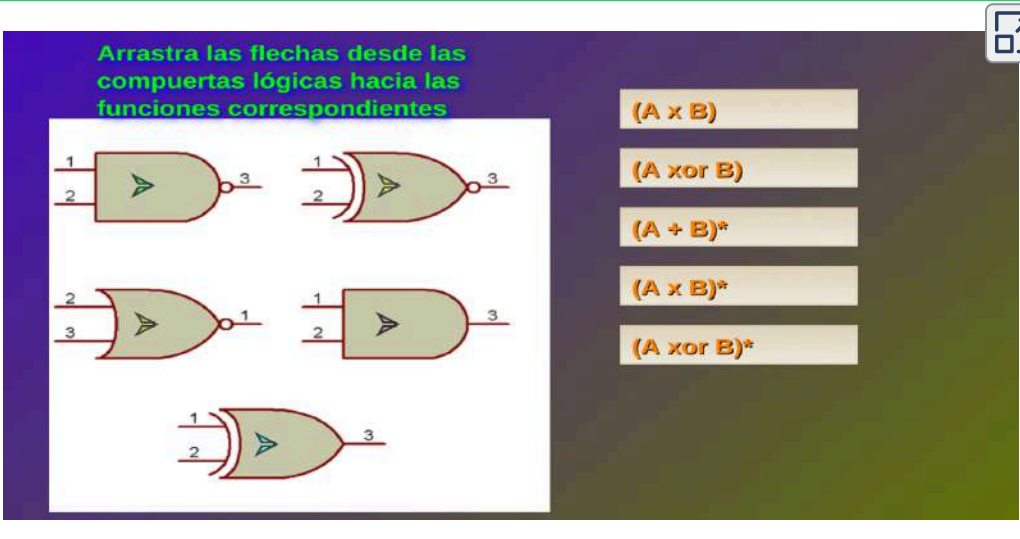
Emparejamiento con términos utilizados en este libro en el ***volumen 1.***[17]



Este objeto interactivo fue generado con la *asistencia* de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/

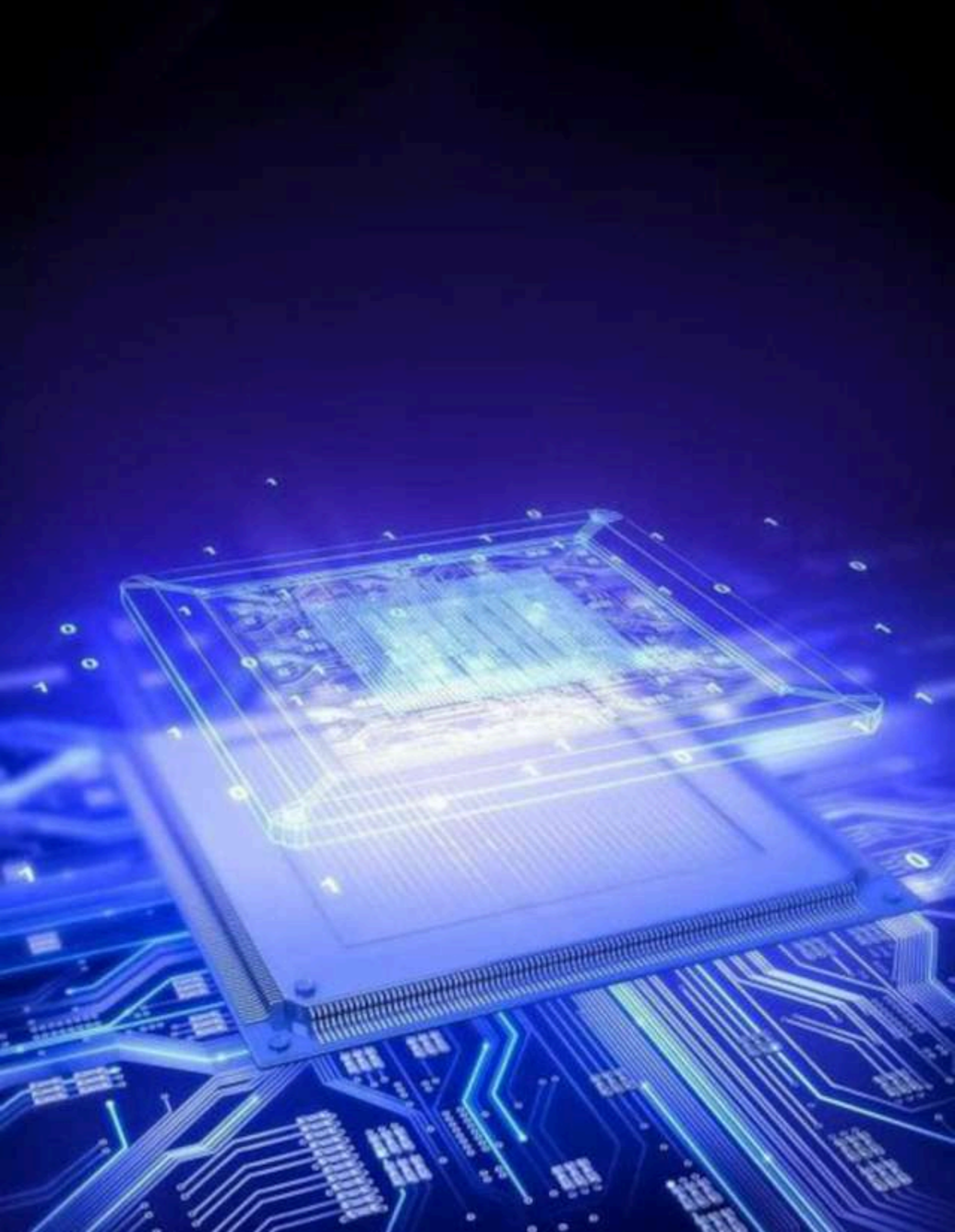
[17] Las plantillas para los Objetos Interactivos fueron obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

El primer Objeto Interactivo es un compendio sobre los tipos de compuertas lógicas, su función y tabla de verdad. El segundo Objeto Interactivo es un emparejamiento entre símbolo de compuertas lógicas y su función lógica.[18]



[18] Las plantillas para los Objetos Interactivos fueron obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS


Ob
ANEXOS



Anexo A
Hojas de Datos

Algunas Hojas de Datos [15]

Tipo	Familia Lógica TTL	Familia Lógica CMOS	Descripción
AND	7408	4081	4 compuertas de 2 entradas
	7411	4073	3 compuertas de 3 entradas
	7421	4082	2 compuertas de 4 entradas
NAND	7400	4011	4 compuertas de 2 entradas
	7410	4023	3 compuertas de 3 entradas
	7420	4012	2 compuertas de 4 entradas
	7430	4068	1 compuerta de 8 entradas
	74133	----	1 compuerta de 13 entradas
OR	7432	4071	4 compuertas de 2 entradas
	----	4075	3 compuertas de 3 entradas
	----	4072	2 compuertas de 4 entradas
NOR	7402	4001	4 compuertas de 2 entradas
	7427	4025	3 compuertas de 3 entradas
	7425	4002	2 compuertas de 4 entradas
	----	4078	1 compuerta de 8 entradas
NOT	7404	4069	6 compuertas de 1 entrada
YES	7407	4050	6 compuertas de 1 entrada
XOR	7486	4030	4 compuertas de 2 entradas
XNOR	----	4077	4 compuertas de 2 entradas

Familia Lógica TTL	Familia Lógica CMOS	Descripción
7483	4008	Sumador completo de 4 bits

Descripción
Display DUAL de 7 segmentos
Display DUAL de 14 segmentos
Display SENCILLO de 16 segmentos

Anexo B
Subfamilias de los CI TTL

Subfamilias TTL [11]

Subfamilia	Prefijo	Capacidad de salida FanOut	Disipación de potencia (mW)	Retardo de propagación (ns)
Estándar	74	10	10	9
Baja potencia	74L	20	1	33
Alta velocidad	74H	10	22	6
Schottky	74S	25	20	3
Schottky de baja potencia	74LS	10	2	9,5
Schottky avanzado	74AS	50	8	1,7
Schottky avanzado de baja potencia	74ALS	10	1,2	4
Alta velocidad	74F	20	4	3

Anexo C
Subfamilias de los CI CMOS

Subfamilias CMOS [12]

Subfamilia	Prefijo
Estándar	CD, HEF, HC
Bajo voltaje	LV
CMOS bajo voltaje	LVC
CMOS Avanzado	AC
CMOS Avanzado compatible con TTL	ACT

Material de la Compuerta CMOS [3]

Material	Capacidad de salida FanOut	Disipación de potencia (µW)	Retardo de propagación (ns)
Compuerta de Silicio serie 74HC	>100	2,5	8
Compuerta Metálica serie 4000	>50	1	50

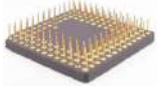
Anexo D
Sufijos de Fabricantes Actuales

Sufijos Fabricantes [12]

Fabricante	Logo	Código
Texas Instruments		TTL = SN
		CMOS = CD
Hitachi		TTL = HD
		CMOS = HEF y HCF
Motorola		MC, ON Semiconductor
Anteriormente: Philips Semiconductors Actualmente: NXP Semiconductors		NXP

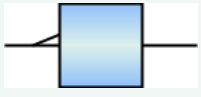
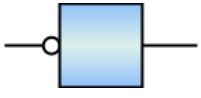
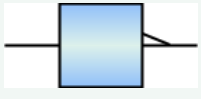
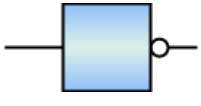
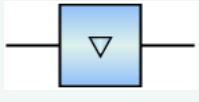
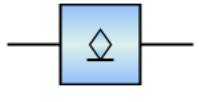
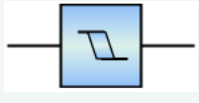
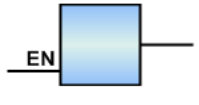
Anexo E
Principales Encapsulados CI

Principales Encapsulados [13][14]

Figura	Descripción
	**Dual InLine Packaging (DIP)**: tiene dos filas de pines paralelos a lo largo del cuerpo del encapsulado
	**Small Outline Package (SOP)**: son más pequeños y compactos; con terminales en "ala de gaviota o en J" que se ensambra sobre el PCB
	**Quad Flat Package (QFP)**: encapsulado cuadrado o rectangular con muchos pines; buena disipación de calor y alta escala de integración
	**Pin Grid Array (PGA)**: posee una gran cantidad de terminales situados en la parte inferior del encapsulado y se ensambla en zócalo
	**Ball Grid Array (BGA)**: en vez de pines utilizan unas bolas pequeñas de soldadura en la superficie inferior que proporciona un mejor rendimiento eléctrico y disipación térmica
	**Land Grid Array (LGA)**: los contactos se acoplan a las almohadillas correspondientes en la PCB

Anexo F
Símbolos estándar para CI

Símbolos estándar

Símbolo	Descripción
	Entrada activa en bajo, norma Comisión Internacional de Electromecánica (IEC)
	Entrada activa en bajo, norma Americana (ANSI)
	Salida activa en bajo, norma Comisión Internacional de Electromecánica (IEC)
	Salida activa en bajo, norma Americana (ANSI)
	Salida de Tres Estados (Tri-State)
	Salida de Colector Abierto (Open Collector)
	Salida Schmitt Trigger
	Entrada de Habilitación (EN = E)

Bibliografía

[1] O. I. Botero, «Electrónica Digital,» 2022. [En línea]. Disponible: https://proyectodescartes.org/iCartesiLibri/materiales_didacticos/Electronica_Digital/index.html. [Último acceso: 10 2024].

[2] R. L. Tokheim, Principios Digitales, España: Mc Graw Hill, 1995.

[3] T. L. Floyd, Fundamentos de Electrónica Digital, México: Limusa, 2003.

[4] R. J. Tocci, N. S. Widmer y G. L. Moss, Sistemas Digitales, Principios y Aplicaciones, México: Pearson, 2007.

[5] M. Mano, Diseño Digital, México: Pearson, 2003.

[6] J. P. Uyemura, Diseño de Sistemas Digitales - Un enfoque integrado, México: Thomson, 2000.

[7] J. F. Wakerly, Diseño Digital, Principios y Prácticas, México: Prentice Hall, 2006.

[8] E. Mandado, Sistemas Electrónicos Digitales, España: Marcombo, 2015.

[9] H. Taub y D. Schilling, Electrónica Digital Integrada, España: Marcombo, 1984.

[10] J. W. Bignell y R. L. Donovan, Electrónica Digital, México: CECSA, 1997.

[11] L. García, «Facultad de Ingeniería Eléctrica - Laboratorio de Electrónica Digital I,» [En línea]. Disponible: https://www.fie.umich.mx/lab-electronica/wp-content/uploads/sites/7/2021/10/Practica6-LEDI.pdf. [Último acceso: 12 2024].

[12] M. Ciencia, «Familias lógicas de circuitos integrados:historia,códigos BIC,RTL,DTL,ECL,TTL,MOS,CMOS,HCMOS,BiCMOS.,» 11 07 2024. [En línea]. Disponible: https://youtu.be/XCa7zquVzP8?si=kPjpU2bwwk6C59Bu. [Último acceso: 12 2024].

[13] Cadence Design Systems, Inc. , «IC Packaging Types,» 2025. [En línea]. Disponible: https://resources.pcb.cadence.com/blog/2023-ic-packaging-types. [Último acceso: 01 2025].

[14] Microchipotle, «Tipos de encapsulados en componentes electrónicos,» 2025. [En línea]. Disponible: https://microchipotle.com/tipos-de-encapsulados-en-componentes-electronicos/. [Último acceso: 01 2025].

[15] Alldatasheet, «Electronic Components Datasheet Search,» [En línea]. Available: https://www.alldatasheet.com/.


DESCARTES
matemáticas interactivas