

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO



CIENCIAS DE LA INGENIERÍA

INGENIERÍA EN SOFTWARE

ECUADOR

ARITMÉTICA DE LA COMPUTADORA Y ALGEBRA BOOLE

AUTORES:

ALMAGRO INTRIAGO LAINER PATRICIO

FUERTES ARRAES EDSON DANIEL

NIEVES SANCHEZ JIMMY SAMUEL

SALTOS TELLO JOSEPH ORLANDO

DOCENTE:

ING. GUERRERO ULLOA GLEISTON CICERON

CURSO/PARALELO:

SEGUNDO SOFTWARE “B”

SPA 2025 – 2026

Aritmética Binaria

La aritmética binaria es todo un conjunto de operaciones matemáticas lógicas, que se realizan o se expresan en base 2. Este sistema es utilizado en casi todos los artefactos tecnológicos digitales modernos porque se adapta y se acopla perfectamente a la tecnología electrónica, además se utiliza únicamente dos dígitos que son el 0 y 1. Esta elección no es arbitraria en el sistema binario, si no es una consecuencia directa de la naturaleza en base de la naturaleza física de los circuitos, los cuales solo pueden estar en dos estados: apagado y encendido, así también como alto o bajo, 1 o 0 [1].

Representar cualquier número en el sistema de datos binario mediante varias combinaciones de bits. En el número decimal por ejemplo el 5 se representa como un 101 en binario. Siendo esto que la verdadera potencia del sistema binario se da a conocer al realizar operaciones aritméticas, tales como la suma, resta, multiplicación y división, ejercicios los cuales son la base de todo procesamiento digital ya sea en datos de información vital para el usuario [2].

Para la suma binaria, como regla básica se basa de manera simple con estas reglas $0+0=0$, $0+1=1$, $1+0=1$ y $1+1=10$, dando esto como resultado de 0 y se lleva 1 al siguiente bit. Este es fundamental para lo que permiten construir sumadores, que son circuitos fundamentales en unidades de control aritmético lógicas (ALU) de procesadores. La resta binaria sigue una lógica similar pero simplificada al diseño de circuitos restadores. Complementando así una estructura ordenada de pasos [3].

A diferencia del sistema decimal que utiliza dígitos del 0 al 9, en cambio en el sistema binario o aritmética binaria se usan apenas dos valores que son el 0 y 1, cada una de sus posiciones en una secuencia binaria se pueden representar en una potencia de 2, siendo un poco similar al sistema decimal solo que en este se representa en cada posición con una potencia de 10 [4].

En la multiplicación binaria se basa de varias sumas repetidas simultáneamente, al multiplicar algún número por 1 significa mantenerlo o conservarlo para luego usarlo con el siguiente, y multiplicarlo 0 lo anula, son muy eficientes para implementarlas en hardware [4].

Por otro lado, en la división binaria se pueden realizar utilizando técnicas similares a la división larga, pero en decimal, aunque tendríamos que adaptarlas al sistema binario.

Aritmética Decimal

La precisión mejora al diferenciar números exactos de aquellos con incertidumbre mediante marcas como $.L$ (baja fracción) y $.H$ (alta fracción). Por ejemplo, un número representado como 0.L indica que su valor está cerca de 0.2, mientras que 0.H estaría cerca de 0.7. Esto ayuda a detectar errores y evitar que operaciones posteriores traten como exactos números que en realidad tienen margen de error, mejorando la fiabilidad en cálculos financieros y científicos [5].

Para convertir un número decimal codificado en BCD a binario, el número se divide en grupos de 8 bits (equivalentes a 2 dígitos decimales). Por ejemplo, el número 32 se divide en dos dígitos: 3 y 2. El dígito 3 se multiplica por 10 para dar 30, y el dígito 2 se suma directamente, resultando en 32 en binario. Esta conversión directa y paralela reduce errores de redondeo y evita usar multiplicadores, mejorando velocidad y eficiencia en dispositivos FPGA [6].

La multiplicación decimal se acelera combinando software y hardware. Por ejemplo, para calcular 4×7 , se suma 4 repetidamente siete veces en software para generar los múltiplos, y el hardware acumula estos productos parciales de forma eficiente. Otros métodos usan hardware para combinar productos parciales de manera paralela, mejorando la velocidad a costa de más recursos. Estas técnicas reducen significativamente el tiempo y área requeridos en comparación con métodos puramente software o hardware [7].

Los sumadores decimales de acarreo anticipado calculan todos los acarreos entre dígitos simultáneamente, en lugar de esperar que cada uno se propague en secuencia. Por ejemplo, al sumar $358 + 174$, en lugar de esperar que el acarreo entre las unidades ($8+4$), decenas ($5+7$) y centenas ($3+1$) se pase uno por uno, el circuito calcula desde el inicio si habrá acarreo en cada posición (C1, C2, C3) usando solo las entradas, lo que acelera la operación y mejora la eficiencia energética. Aunque este diseño requiere más transistores, la ganancia en velocidad es significativa [8].

Algebra de circuitos digitales

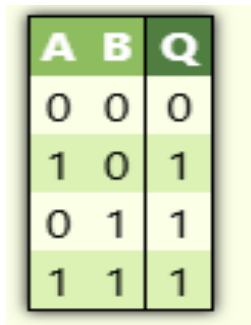
El algebra de los circuitos digitales se relaciona al manejo del algebra de Boole de los circuitos digitales, su funcionamiento principal se basa de 0 y 1 y en que cada puerta lógica sus resultados son diferentes [9].

Operaciones básicas del Álgebra de Boole

En ese sistema de operaciones básica se trabaja con 0 y 1 en su adverbio de operaciones básica como: Suma, Resta, Multiplicación, División [10].

Suma lógica (OR)

En la suma lógica de OR basada en la teoría de algebra de Boole en el formato de OR si en la suma se encuentra un 1 el resultado de esa salida digitales será 1, si en la suma se haya 2 ceros el resultado de la salida será 0 [11].

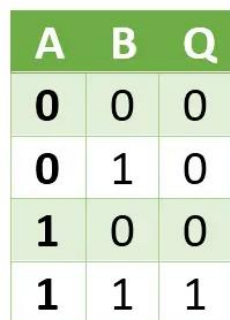


A	B	Q
0	0	0
1	0	1
0	1	1
1	1	1

Ilustración 1 Suma lógica (OR, +)

Multiplicación lógica (AND)

En la multiplicación lógica en AND en los termino de algebra de Boole la salida es 1 cuando al multiplica el elemento A y B sus valores son 1, mientras en unos de los elementos existan un cero la salida será 0 [12].



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Ilustración 2 Suma lógica (AND)

Negación (NOT)

En la negación es una base fundamental en el algebra de Boole donde que al ingresar el valor tanto 0 o 1 el valor se invierte [13].

Q	Q'
0	1
1	0

Ilustración 3 Negación (NOT)

Función Booleana

La función Booleana es una expresión matemática lógica, desarrollada por George Boole en el siglo XIX con la idea para poder representar relaciones de forma lógica usando variables binarias. En electrónica digital, las funciones Booleanas se volvieron fundamentales con esto ahora se usan para modelar el comportamiento de circuitos lógicos y creación de todo tipo de sistemas computacionales [14].

Siendo una expresión booleana que toma valores binarios tanto de entrada 0 y 1 como también poder producir un valor binario de salida. Estas son funciones implementadas y diseñadas dando vida a los circuitos digitales, con esto poder implementar decisiones lógicas, dentro de condicionales de un sistema al que se vaya a ser usado [15].

Una función Booleana puede representarse de varias formas como una tabla de verdad, una expresión algebraica o un circuito lógico. Las operaciones lógicas básicas de la función o lógica Booleana son AND conjunción, OR disyunción y NOT negación. Mediante de estas tres operaciones se pueden dar otras características más complejas o funciones más complejas como NAND, NOR, XOR y XNOR [16].

Dicho que los circuitos en la vida real es necesario que se minimicen el número de compuertas, esto para poderse optimizar el circuito, se utilizan técnicas de simplificación. El mapa de Karnaugh es una de las más comunes, una herramienta que permite agrupar combinaciones visualmente y con esto poder simplificar claro aplicando las leyes de la algebra Booleana .

Se pueden construir todo tipo de dispositivos digitales con la ayuda de las funciones booleanas, desde una alarma que se activa mediante ciertos parámetros condicionados por procesadores enteros y sistemas conectados entre sí [17].

Los decodificadores activan una única salida según la combinación de entradas, los codificadores hacen lo opuesto, multiplexores dan múltiples señales de entrada y salidas para poder interpretarlas y las memorias o registros las cuales se usan para determinar cuándo guarda o liberan datos. Así como en la programación en la lógica booleana también es fundamental las instrucciones condicionales if, while, etc. Estas se basan en operaciones Booleanas que evalúan si se cumple aquella condición o no [18].

Circuitos lógicos

Los memristores son una opción nueva para mejorar los circuitos porque superan algunas limitaciones de los transistores tradicionales, como el material que usan, el consumo de energía y el costo. Desde que se descubrieron en 1971 y se empezaron a fabricar a nanoescala en 2008, han permitido avances importantes. Además, se pueden integrar fácilmente con la tecnología CMOS actual, ya que pueden colocarse en la misma capa de metal, lo que ahorra espacio y hace que los circuitos funcionen más rápido [19].

Los circuitos combinacionales mejorados usando puertas MRL que solo necesitan un memristor y un transistor NMOS por puerta. También se diseñó un módulo lógico que combina un transistor y cinco memristores para hacer operaciones como AND, OR y XOR. Esto ayuda a que los circuitos tengan menos retrasos, consuman menos energía y usen menos componentes que otros métodos. Además, las señales que generan son muy claras sin tener que usar partes extras, y su velocidad puede ser hasta diez mil veces mejor que la tecnología CMOS tradicional [19].

Los circuitos que usan transistores FeFET (Ferroelectric Field-Effect Transistor) muestran mejoras importantes frente a los tradicionales CMOS y otras tecnologías nuevas. Estos dispositivos pueden guardar información sin perderla, lo que permite hacer memorias más pequeñas y rápidas, y unidades lógicas eficientes. Además, estos diseños usan menos espacio, consumen menos energía y son más rápidos, especialmente cuando usan un tipo de lógica que reduce a la mitad los transistores necesarios, lo que también evita el gasto innecesario de energía [20].

La computación con ADN es una forma diferente de hacer circuitos usando moléculas biológicas, aprovechando que pueden hacer muchas cosas al mismo tiempo para resolver problemas complejos, como en medicina. Usan modelos y métodos especiales que permiten construir compuertas lógicas con ADN, aunque con algunas limitaciones. Para facilitar su diseño y funcionamiento, se creó un sistema llamado RTL2DNA, que

convierte los circuitos lógicos en experimentos en chips microfluídicos, manteniendo buen rendimiento y precisión incluso en circuitos grandes [21].

Por otra parte, se han diseñado circuitos moleculares que pueden manejar información dependiendo del tiempo en que llegan las señales, usando ADN y memoria temporal. Esto significa que pueden diferenciar el orden en que llegan las señales para tomar decisiones más complejas. Esta forma de trabajar es escalable y tiene potencial para usarse en redes neuronales hechas con ADN o en máquinas moleculares que aprendan y se comporten de forma inteligente [22].

Circuitos Lógicos de Sistemas Digitales

Los circuitos lógicos son combinaciones de compuertas lógicas que permiten procesar información digital (valores 0 y 1) dentro de un sistema digital. Forman la base de computadoras, calculadoras, sistemas de control, semáforos y dispositivos electrónicos [23].

Compuerta AND

Las compuertas AND es unas compuertas lógicas que tienen que tener mínimo 2 entradas y su salida es verdadero cuando los valores de entrada son verdaderos, si en los valores de entrada se encuentra un valor falso el resultado será falso [24].

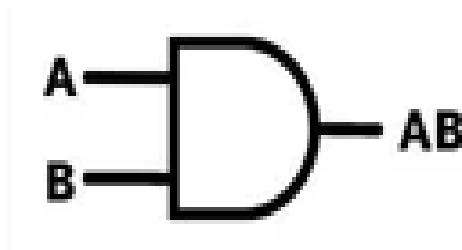


Ilustración 4 Compuerta AND

Compuerta OR

Las compuertas OR también es considerada una compuerta lógica que su composición se conforma como una suma booleana, Si en una de su entrada tiene el valor verdadero la salida será verdadera, pero si en la entrada de los valores son falsos el resultado será falso [25].

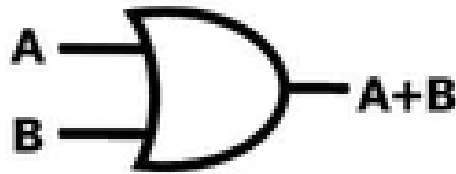


Ilustración 5 Compuerta OR

Compuerta NOT

La compuerta NOT se caracteriza por negar los valores que se ingresan en la compuerta NOT solamente se le ha permitido ingresar un valor [26].

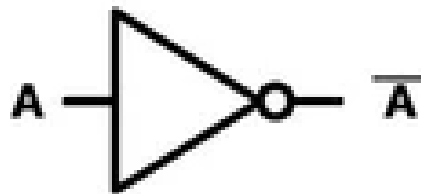


Ilustración 6 Compuerta NOT

Compuerta NAND

En la compuerta NAND es la contradicción de la compuerta AND, cuando en la entrada de la compuerta sus valores son verdaderos su valor será falso, mientras tanto será verdadero [27].

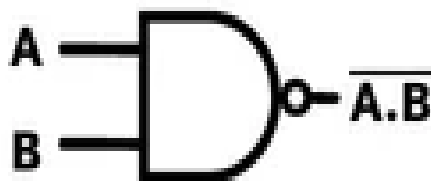


Ilustración 7 Compuerta NAND

Compuerta NOR

En la compuerta NOR es la negación de la compuerta OR, la salida se vuelve verdadera cuando en sus 2 entradas son falsas, mientras tanto su salida será falsa [28].



Ilustración 8 Compuerta NOR

Compuerta XOR

En la compuerta XOR cuando se quiere llegar a la salida alta ósea verdadera las entradas tienen que ser de valores diferentes no iguales, mientras tanto cuando son valores iguales serán falsos [25].

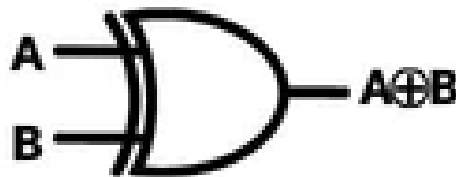


Ilustración 9 Compuerta XOR

Compuerta XNOR

La compuerta lógica NXOR es la negación de la compuerta XOR, cuando existe 2 entrada con el mismo valor será la salida verdadera, mientras tanto cuando sus valores sean diferentes serán falsos [29].



Ilustración 10 Compuerta XNOR

- [1] H. Winther, “SPIN REPRESENTATIONS AND BINARY NUMBERS,” *Archivum Mathematicum*, vol. 60, no. 4, pp. 231–241, 2024, doi: 10.5817/AM2024-4-231.
- [2] T. Ndjountche, “Digital Electronics,” in *Digital Electronics*, Robert Baptist, Ed., Great Britain United States: ISTE Ltd and John Wiley & Sons, Inc., 2016, ch. 1. doi: 10.1002/9781119318620.fmatter.

- [3] X. D. Li *et al.*, “Classification feature selection and dimensionality reduction based on logical binary sine-cosine function arithmetic optimization algorithm,” *Egyptian Informatics Journal*, vol. 26, Jun. 2024, doi: 10.1016/j.eij.2024.100472.
- [4] M. P. Véstias and H. C. Neto, “Decimal multiplication in fpga with a novel decimal adder/subtractor,” Jul. 01, 2021, *MDPI AG, Lisboa, Portugal*. doi: 10.3390/a14070198.
- [5] M. F. Mudawar, “Exact Versus Inexact Decimal Floating-Point Numbers and Arithmetic,” *IEEE Access*, vol. 11, pp. 17891–17905, 2023, doi: 10.1109/ACCESS.2023.3244891.
- [6] S. R. Sahu and M. Pradhan, “Mixture of Decimal and Binary Arithmetic Units for FPGA Based Architecture,” in *2020 4th International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/IEMENTech51367.2020.9270100.
- [7] *2018 IEEE Asia Pacific Conference on Circuits and Systems : 26-30 October 2018, Chengdu, China*. Institute of Electrical and Electronics Engineers, 2018.
- [8] A. Al Share, O. Al-Khaleel, F. N. Zghoul, M. Al-Khaleel, and C. Papachristou, “Design and Implementation of High Speed Carry Look-Ahead Decimal Adder (CLDA) Using CMOS Technology,” *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.3540836.
- [9] R. Malik, “Circuit Optimization using Arithmetic Table Lookups,” 2025, doi: 10.1145/3729258.
- [10] O. Ufuoma, “Boolean Subtraction and Division with Application in the Design of Digital Circuits,” *Journal of Engineering Research and Reports*, pp. 95–117, Apr. 2021, doi: 10.9734/jerr/2021/v20i517316.
- [11] S. Zhao, L. Yu, S. Yang, X. Tang, K. Chang, and M. Chen, “This journal is Cite this: Nanoscale Horiz,” vol. 6, p. 298, 2021, doi: 10.1039/d0nh00587h.
- [12] H. Xiao, R. Zhao, Y. Liu, Y. Liu, and J. Chen, “Bi-Directional and Operand-Controllable In-Memory Computing for Boolean Logic and Search Operations with Row and Column Directional SRAM (RC-SRAM),” *Micromachines (Basel)*, vol. 15, no. 8, Aug. 2024, doi: 10.3390/mi15081056.

- [13] S. Byeok Jo, J. Kang, and J. Ho Cho, “REVIEW www.advancedscience.com Recent Advances on Multivalued Logic Gates: A Materials Perspective,” 2021, doi: 10.1002/advs.202004216.
- [14] R. Malik, V. Paranjape, and M. Kulkarni, “Circuit Optimization using Arithmetic Table Lookups,” *Proceedings of the ACM on Programming Languages*, vol. 9, Jun. 2025, doi: 10.1145/3729258.
- [15] E. Kubaczka *et al.*, “Energy Aware Technology Mapping of Genetic Logic Circuits,” *ACS Synth Biol*, vol. 13, no. 10, pp. 3295–3311, Oct. 2024, doi: 10.1021/acssynbio.4c00395.
- [16] B. Steinbach and C. Posthoff, “Derivative Operations for Lattices of Boolean Functions,” *Trinidad y Tobago*, Sep. 2013. doi: 10.13140/2.1.2398.6568.
- [17] B. Sampathkumar, R. Das, B. Martin, F. Enescu, and P. Kalla, “An Algebraic Approach to Partial Synthesis of Arithmetic Circuits,” in *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, Institute of Electrical and Electronics Engineers Inc., Mar. 2025, pp. 1097–1103. doi: 10.1145/3658617.3697724.
- [18] T. Higuchi *et al.*, “Real-world applications of analog and digital evolvable hardware,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 220–234, Sep. 1999, doi: 10.1109/4235.788492.
- [19] G. Liu, S. Shen, P. Jin, G. Wang, and Y. Liang, “Design of Memristor-Based Combinational Logic Circuits,” *Circuits Syst Signal Process*, vol. 40, no. 12, pp. 5825–5846, Dec. 2021, doi: 10.1007/s00034-021-01770-1.
- [20] X. Yin, X. Chen, M. Niemier, and X. S. Hu, “Ferroelectric FETs-Based nonvolatile logic-in-memory circuits,” *IEEE Trans Very Large Scale Integr VLSI Syst*, vol. 27, no. 1, pp. 159–172, Jan. 2019, doi: 10.1109/TVLSI.2018.2871119.
- [21] Z. Beiki and A. Jahanian, “RTL2DNA: An automatic ow of large-scale DNA-based logic circuit design,” *Scientia Iranica*, vol. 30, no. 4 D, pp. 1279–1295, 2023, doi: 10.24200/sci.2022.54993.4017.
- [22] A. P. Lapteva, N. Sarraf, and L. Qian, “DNA Strand-Displacement Temporal Logic Circuits,” *J Am Chem Soc*, vol. 144, no. 27, pp. 12443–12449, Jul. 2022, doi: 10.1021/jacs.2c04325.

- [23] Y. Q. Aguiar *et al.*, “Mitigation and predictive assessment of SET immunity of digital logic circuits for space missions,” *Aerospace*, vol. 7, no. 2, Feb. 2020, doi: 10.3390/aerospace7020012.
- [24] O. J. Quintana-Romero and A. Ariza-Castolo, “Complex molecular logic gates from simple molecules †,” 2021, doi: 10.1039/d1ra00930c.
- [25] F. Sonmez, S. Ardali, B. Arpayay, U. Serincan, and E. Tiras, “Design and realization of XOR, OR, and NAND light logic gates using GaAs heterostructure,” *Appl Phys A Mater Sci Process*, vol. 131, no. 2, Feb. 2025, doi: 10.1007/s00339-024-08213-z.
- [26] N. Zenbaa *et al.*, “Realization of inverse-design magnonic logic gates,” 2025. [Online]. Available: <https://www.science.org>
- [27] A. Kotb, A. Hatziefremidis, G. Said, and K. E. Zoiros, “High-Speed and Cost-Efficient NAND Logic Gate Using a Single SOA-DI Configuration,” *Photonics*, vol. 11, no. 12, p. 1182, Dec. 2024, doi: 10.3390/photonics11121182.
- [28] M. Frické, “Boolean Logic 177 Frické, Martin. 2021,” *Knowledge Organization*, vol. 48, no. 2, pp. 177–191, 2021, doi: 10.5771/0943-7444-2021-2-177.
- [29] A. Kotb, K. E. Zoiros, and W. Chen, “All-Optical XOR, AND, OR, NOT, NOR, NAND, and XNOR Logic Operations Based on M-Shaped Silicon Waveguides at 1.55 μm ,” *Micromachines (Basel)*, vol. 15, no. 3, Mar. 2024, doi: 10.3390/mi15030392.

Anexos

Git-Hub

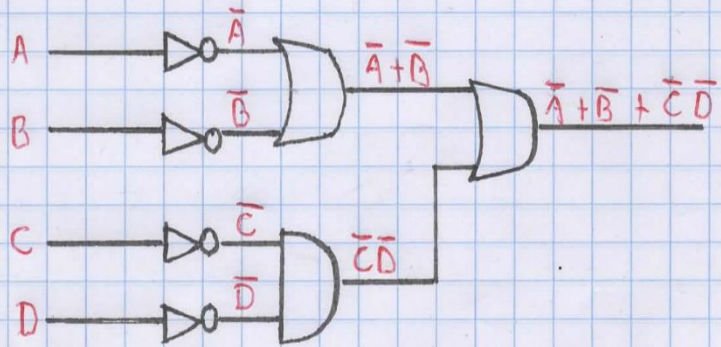
<https://github.com/MiloSaurio4kHD/GrupoE>

EJERCICIOS PLANTEADOS POR EL DOCENTE

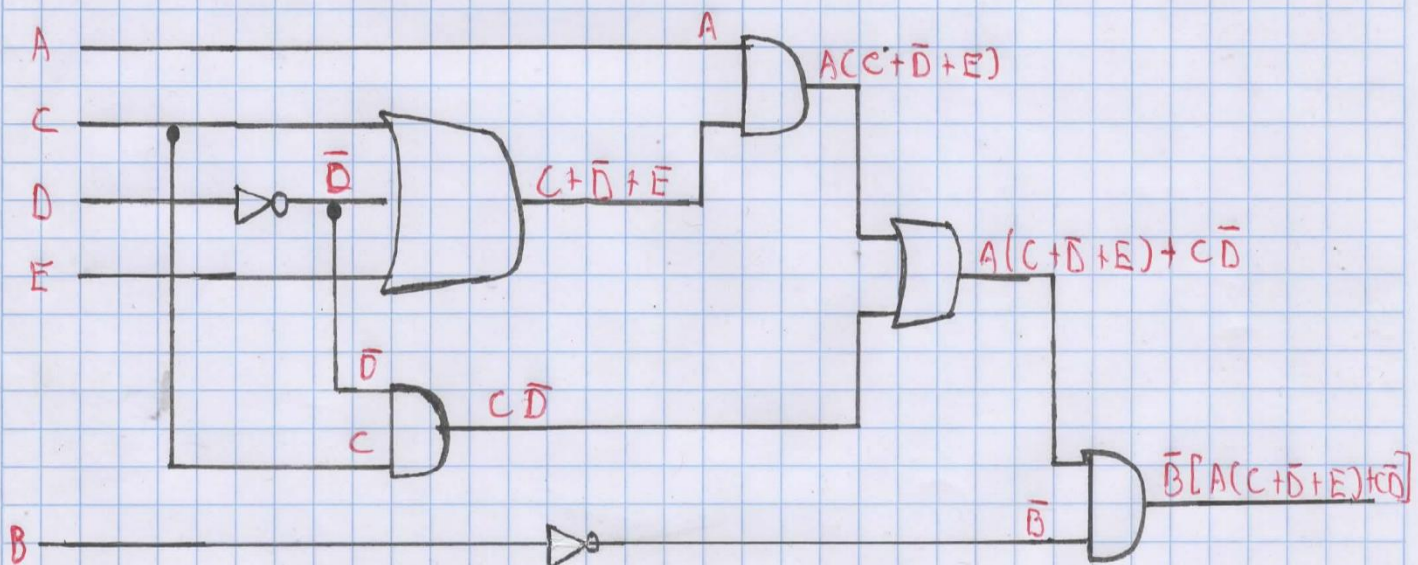
3-16

Para cada una de las siguientes expresiones, construya el circuito lógico correspondiente, utilizando puertas AND y OR e INVERSORES.

$$(a) x = \frac{\overline{AB(C+D)}}{\overline{AB} + \overline{(C+D)}} \\ \boxed{\overline{A+B} + \overline{C} \overline{D}}$$

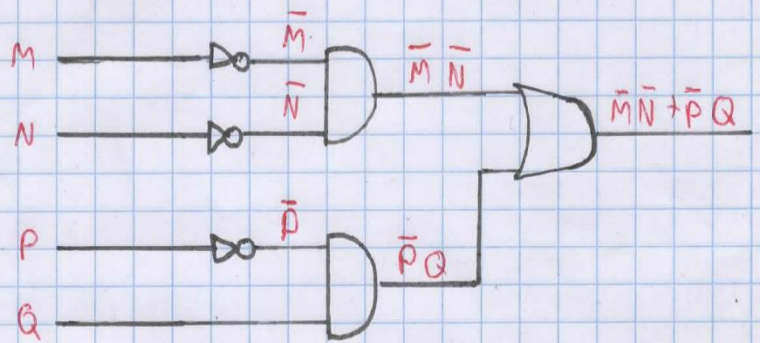


$$(b) z = \overline{(A+B+\overline{C} \overline{D} \overline{E})} + \overline{B} C \overline{D} \\ (\overline{A+B})(\overline{\overline{C} \overline{D} \overline{E}}) + \overline{B} C \overline{D} \\ (\overline{A} \overline{B})(\overline{\overline{C} + \overline{D} + \overline{E}}) + \overline{B} C \overline{D} \\ (\overline{A} \overline{B})(C + \overline{D} + E) + \overline{B} C \overline{D} \\ \overline{A} \overline{B} C + \overline{A} \overline{B} \overline{D} + \overline{A} \overline{B} E + \overline{B} C \overline{D} \\ \overline{B}(\overline{A} C + \overline{A} \overline{D} + \overline{A} E + C \overline{D}) \\ \boxed{\overline{B}[\overline{A}(C + \overline{D} + E) + C \overline{D}]}$$



$$(c) y = (\overline{M+N} + \overline{P}Q)$$

$$\boxed{\overline{M}\overline{N} + \overline{P}Q}$$

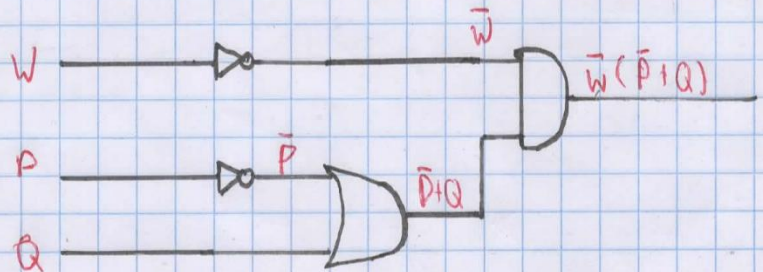


$$(d) x = \overline{W + P\overline{Q}}$$

$$\overline{W} \overline{P\overline{Q}}$$

$$\overline{W} (\overline{P} + \overline{\overline{Q}})$$

$$\boxed{\overline{W} (\overline{P} + Q)}$$

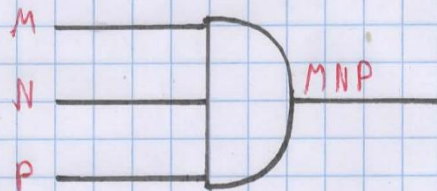


$$(e) z = MN(P + \overline{N})$$

$$MNP + MN\overline{N}$$

$$\boxed{MNP}$$

$$N\overline{N} = 0$$



$$(f) x = (A+B)(\overline{A} + \overline{B})$$

$$A\overline{A} + A\overline{B} + \overline{A}B + B\overline{B}$$

$$A\overline{A} = 0$$

$$B\overline{B} = 0$$

$$A\overline{B} + \overline{A}B$$

$$\boxed{A \oplus B}$$

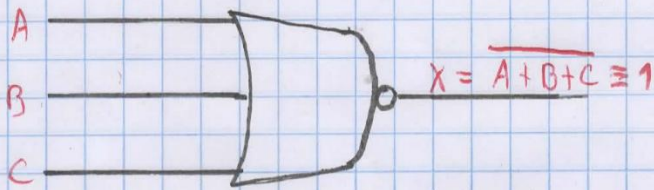


3-17 a) Aplique las formas de ondas de entrada de la Figura 3-64 a una puerta NOR y dibuje la forma de onda de salida.

b) Repita con C en posición bajo.

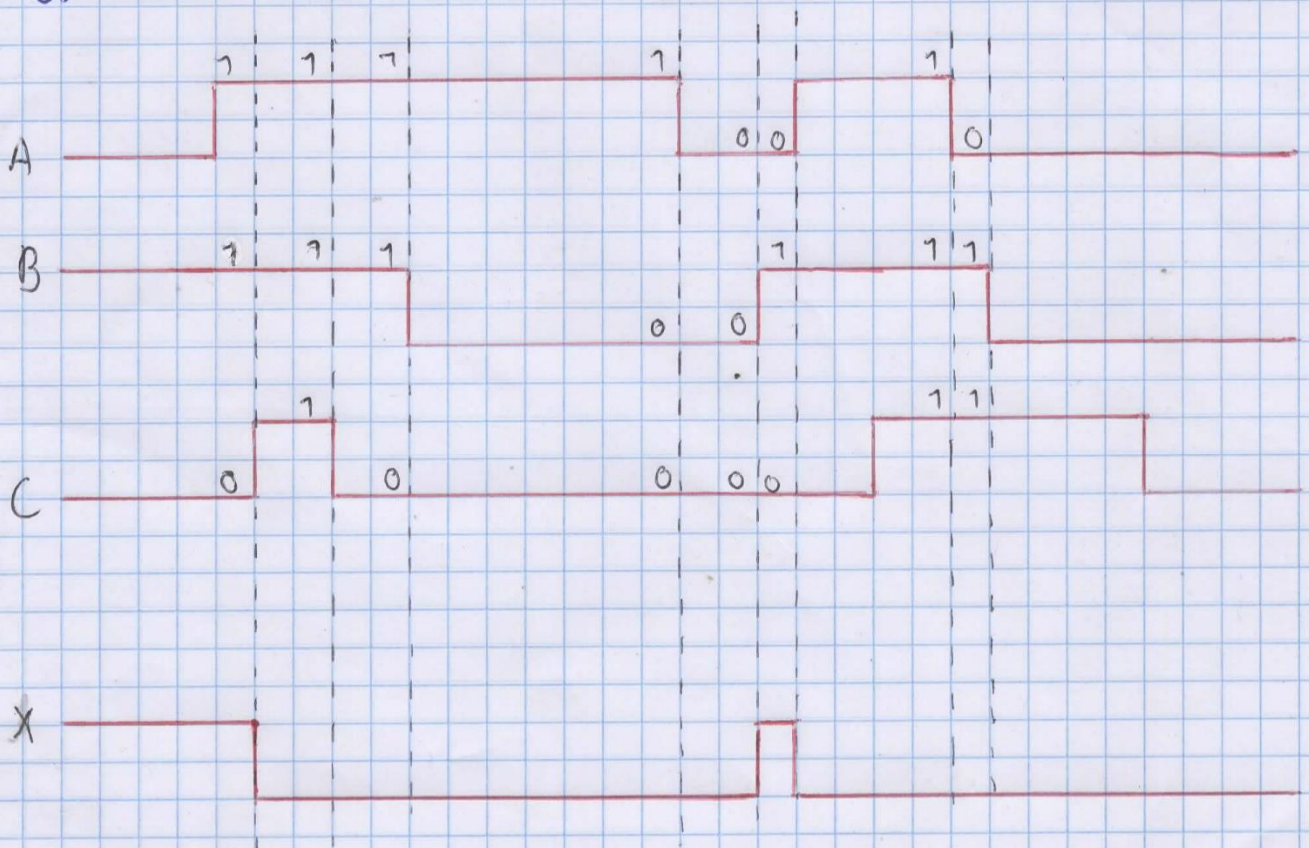
c) Repita con C en posición alto.

Puerta NOR

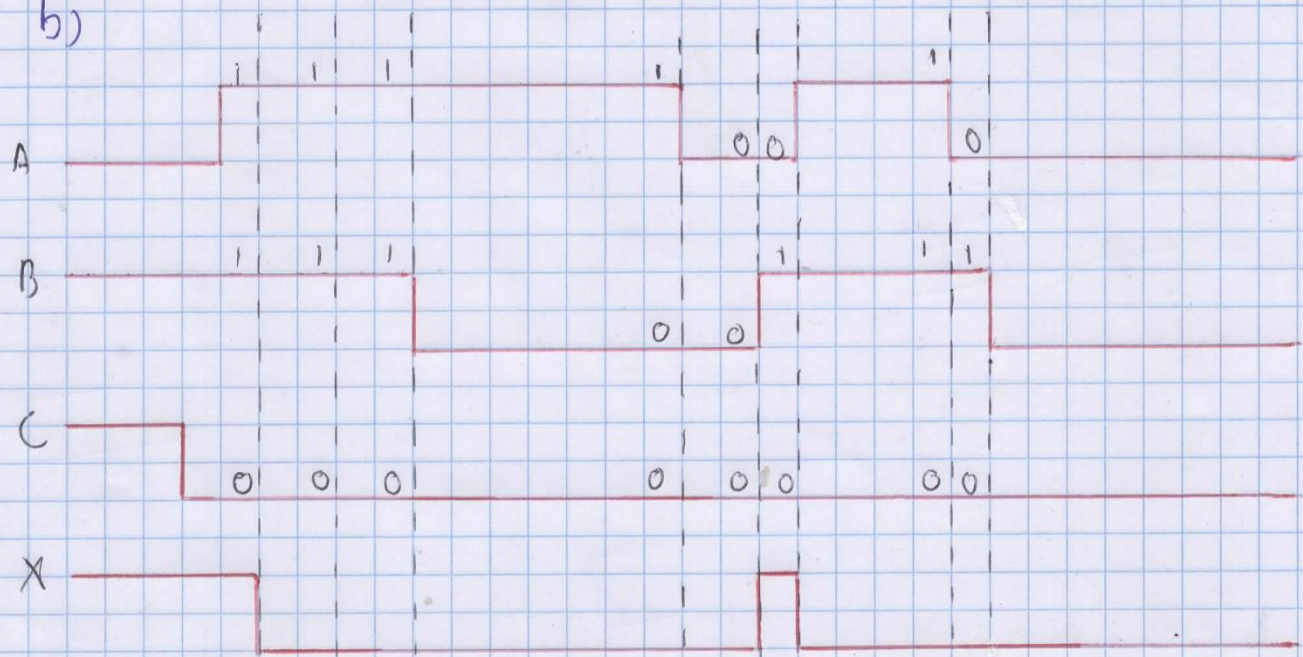


A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

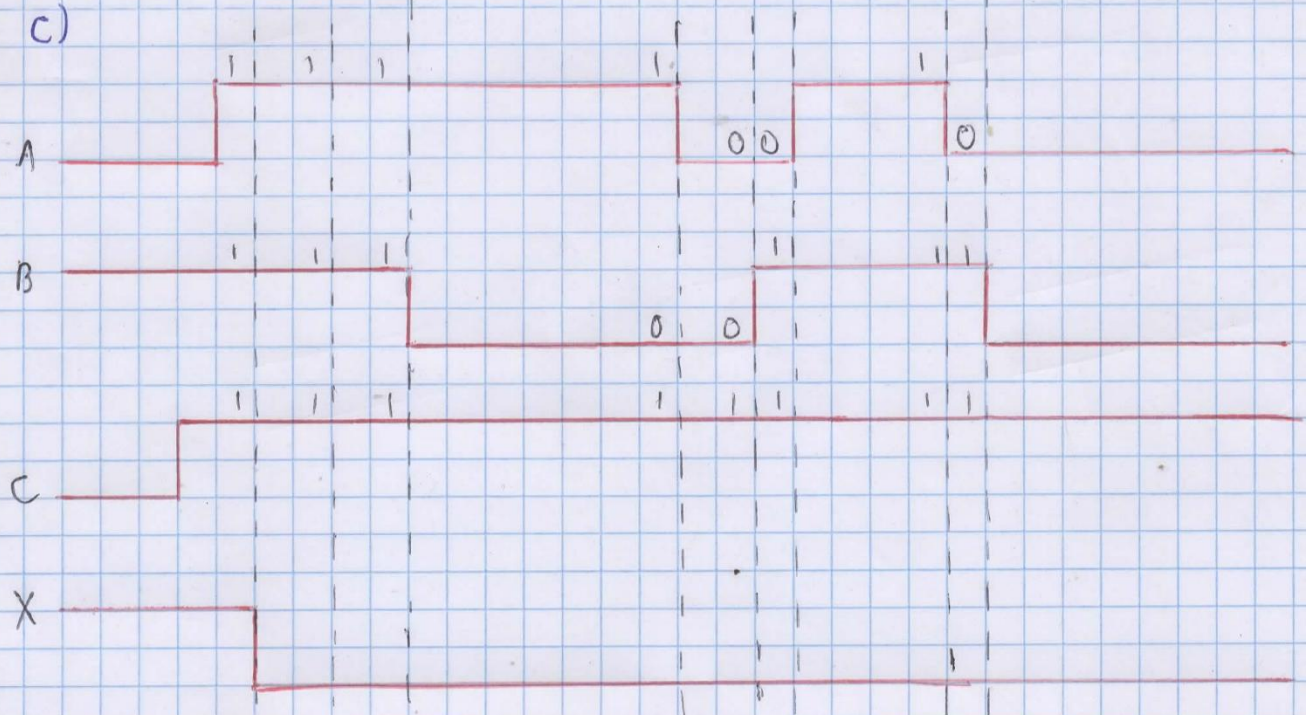
a)



b)



c)



3-26 Simplifique cada una de las siguientes expresiones utilizando los teoremas de De Morgan

$$\begin{aligned} a) \quad & \overline{\overline{A} \overline{B} \overline{C}} \\ & \overline{\overline{A} + \overline{B} + \overline{C}} \\ & \boxed{A + B + C} \end{aligned}$$

$$\begin{aligned} b) \quad & \overline{A + \overline{B}} \\ & \overline{\overline{A} \overline{B}} \\ & \boxed{\overline{A} \overline{B}} \end{aligned}$$

$$\begin{aligned} g) \quad & \overline{A(\overline{B + \overline{C}})D} \\ & \overline{\overline{A} + (\overline{B + \overline{C}}) + \overline{D}} \\ & \boxed{\overline{A} + B + \overline{C} + \overline{D}} \end{aligned}$$

$$\begin{aligned} b) \quad & \overline{\overline{A} + \overline{B} C} \\ & \overline{\overline{A} \overline{B} + \overline{C}} \\ & \boxed{AB + C} \end{aligned}$$

$$\begin{aligned} e) \quad & \overline{\overline{A} \overline{B}} \\ & \boxed{AB} \end{aligned}$$

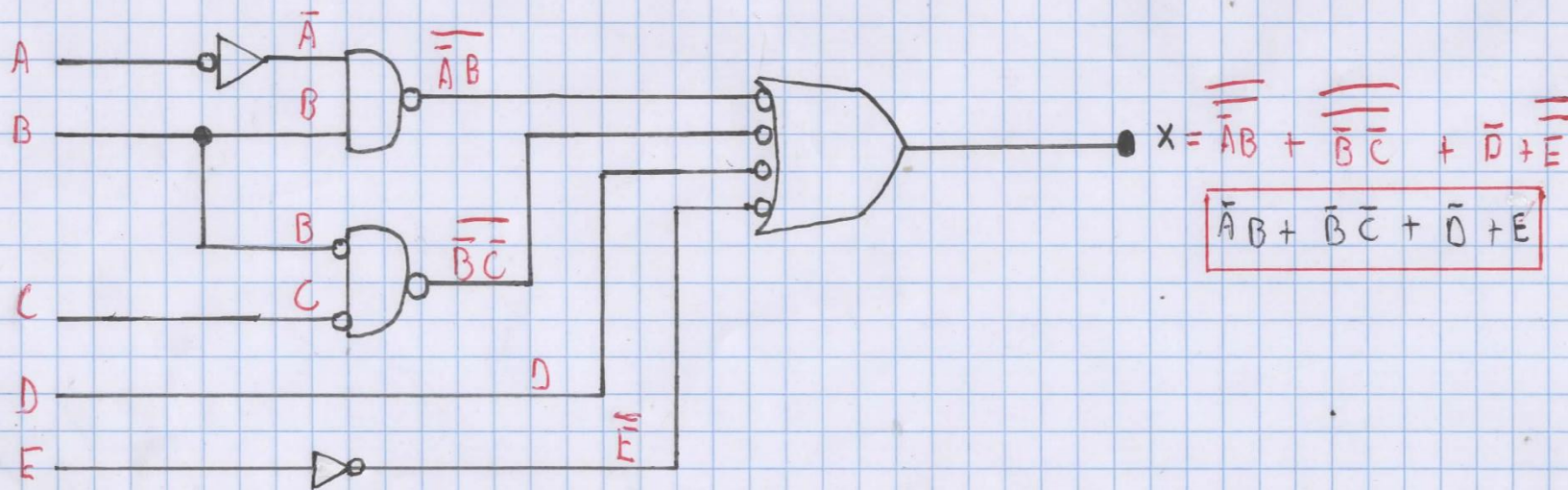
$$\begin{aligned} h) \quad & \overline{(M + \overline{N})(\overline{M} + N)} \\ & \overline{(\overline{M + \overline{N}}) + (\overline{\overline{M} + N})} \\ & (\overline{\overline{M}} \overline{\overline{N}}) + (\overline{\overline{M}} \overline{N}) \\ & \overline{M} N + M \overline{N} \\ & \boxed{M \oplus N} \end{aligned}$$

$$\begin{aligned} c) \quad & \overline{A \overline{B} \overline{C} D} \\ & \overline{\overline{A} \overline{B} + \overline{C} D} \\ & \boxed{\overline{A} + \overline{B} + C D} \end{aligned}$$

$$\begin{aligned} f) \quad & \overline{\overline{A} + \overline{C} + \overline{D}} \\ & \overline{\overline{A} \overline{C} \overline{D}} \\ & \boxed{ACD} \end{aligned}$$

$$\begin{aligned} i) \quad & \overline{\overline{A} \overline{B} C D} \\ & \overline{\overline{A} \overline{B} C + \overline{D}} \\ & \overline{\overline{A} \overline{B} C} + \overline{\overline{D}} \\ & \boxed{(\overline{A} + \overline{B})C + \overline{D}} \end{aligned}$$

3-38 * Determine las condiciones de entrada necesarias para que la salida de la Figura 3-57 sea en su estado activo



R// = El circuito estará en su estado activo cuando $\bar{A} y \bar{B}$, o $\bar{B} y \bar{C}$, o D , o E

4-1) Determine la expresión mínima para cada función K en la figura 4-67. Punt especial atención al par 5 para la función en el mapa (a)

a)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	1	1	0	0
AB	0	0	0	1
$A\bar{B}$	0	0	1	1

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

b)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	1	1
$\bar{A}B$	1	0	0	1
AB	0	0	0	0
$A\bar{B}$	1	0	1	1

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

c)

	\bar{C}	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	0	0
AB	1	0
$A\bar{B}$	1	x

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

4-12 Para la tabla de verdad a continuación cree un mapa K 2×2 , agrupe los términos y simplifique. Luego, mire la tabla de verdad nuevamente para ver si la expresión es verdadera para cada entrada en la tabla.

			$\bar{A}\bar{B} + \bar{A}B$	
A	B	F	\bar{A}	A
0	0	1	1	0
0	1	1	1	0
1	0	0		
1	1	0		

$\leftarrow \bar{A}\bar{B}$ \bar{B}

$\leftarrow \bar{A}B$ B

$$F \equiv \bar{A}$$

$$\bar{A}\bar{B} \equiv 1$$

$$\bar{A}B \equiv 1$$