Electrónica digital

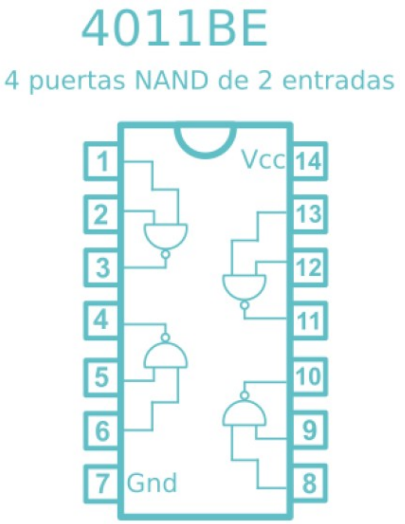
Electrónica digital

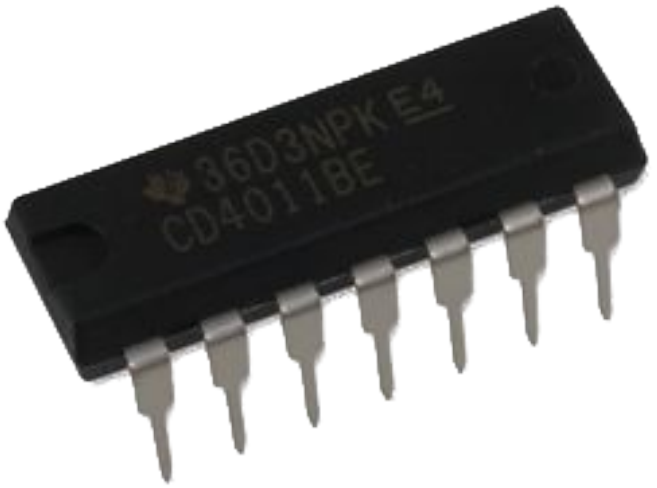
- En la **electrónica digital** la información está codificada en un **conjunto de estado discretos** que por lo general son dos estados también denominados **niveles lógicos que se representan con dos valores de voltaje**:
  - Cercano al que da la fuente de alimentación del circuito → También denominado **1 o verdadero**
  - Cercano al valor de referencia del circuito (0 voltios o tierra) → También denominado **0 o falso**
- Estos valores permite el uso del **álgebra de Boole**, lo cual nos proporciona herramientas muy potentes para realizar cálculo sobre las señales de entrada
- En estos circuitos es habitual transformar señales en principio de naturaleza analógica (audio, temperatura, luz, …) en números codificados en binario mediante **conversores analógico/digitales (A/D)**

Tipos de sistemas digitales I

Sistemas combinacionales

- Utilizan funciones lógicas elementales
- En estos los valores de salida dependen exclusivamente de la entradas
- No tienen memoria, por lo que la salida del circuito no depende de lo que haya pasado anteriormente
- Utilizan elementos como:
  - Puertas lógicas
  - Codificadores/descodificadores
  - multiplexor/demultiplexor
  - Comparador

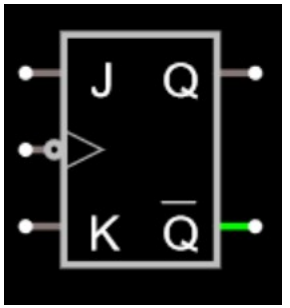


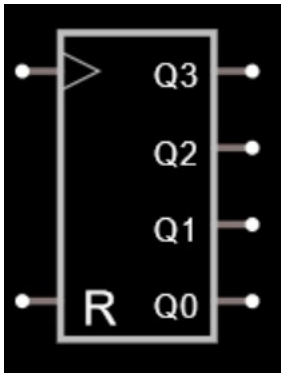


Tipos de sistemas digitales II

Sistemas secuenciales

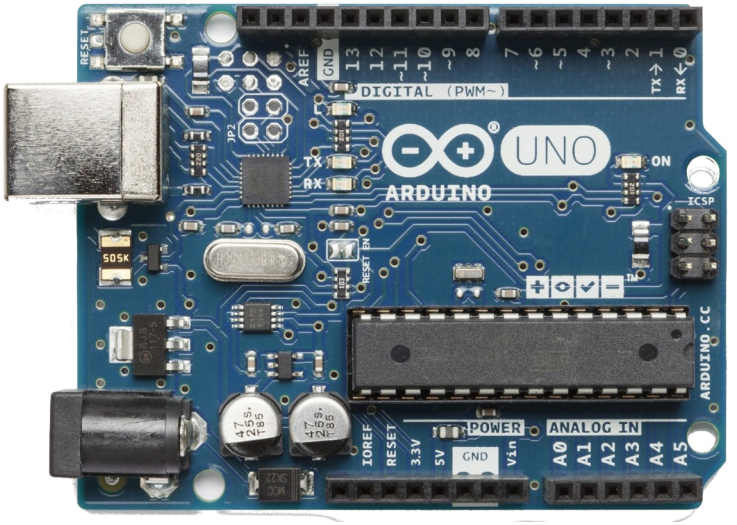
- El valor de salida no solo depende de la entrada, sino también de valores y estados almacenados anteriormente
- Son circuitos con memoria
- Utilizan elementos como:
  - Biestables
  - Contadores





Sistemas microprogramables

- Capaces de ejecutar las instrucciones de un programa
- Utiliza entre otros puertas lógicas, comparadores, contadores, conversores analógico-digital, memorias, decodificadores, ...
- Un ejemplo de estos es la placa arduino



Puertas lógicas

Álgebra de Boole

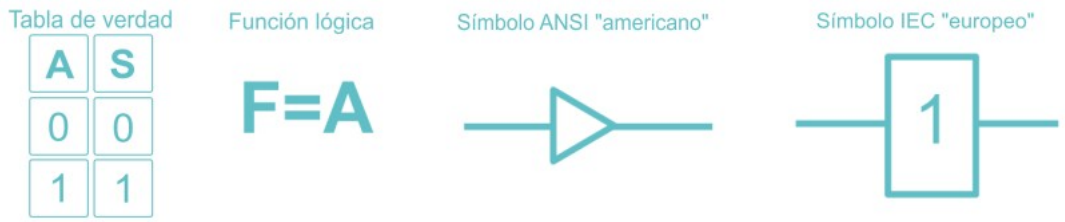
El álgebra de Boole entre otros se compone de dos elementos:

- **Variables lógicas**: Variables que sólo pueden tomar los valores binarios 0 y 1
- **Operaciones**: Que permiten combinar variables lógicas para obtener un resultado. Estas son:
  - **Suma lógica**: Actúa sobre varias variables y es también equivalente a un O lógico, devuelve 1 siempre que cualquiera de las variables es 1
  - **Producto lógico**: Actúa sobre varias variables y es también un Y lógico, devuelve 1 siempre que todas las variables sean 1
  - **Negación**: Actúa sobre una única variable e invierte su valor
  - **Igualdad**: Actúa sobre una única variable devolviendo su mismo valor

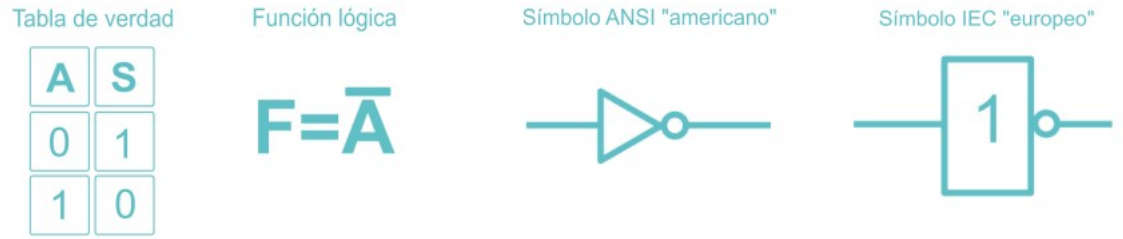
Vamos a ver cómo se implementan estas operaciones del álgebra de boole con las **puertas lógicas** que son las que nos van a permitir aplicarla en la electrónica digital

Puertas SI y NOT

- La **puerta SI** implementa la **función de igualdad**. Tiene una única entrada y una única salida siendo siempre iguales



- La **puerta NOT** implementa la **función de negación**. Tiene una única entrada y una única salida siendo siempre distintas



Puertas OR y NOR

- La **puerta OR** Realiza la **función suma lógica** o **función OR**. La función toma valor lógico 1 cuando alguna de las entradas vale 1 y toma el valor 0 cuando todas las entradas valen 0

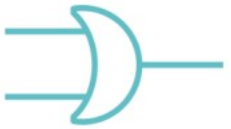
Tabla de verdad

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

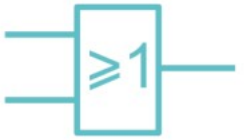
Función lógica

$F=A+B$

Símbolo ANSI "americano"



Símbolo IEC "europeo"



- La **puerta NOR** realiza la función **suma lógica negada** o **función NOR**. Es la función contraria a la OR, por lo tanto la salida siempre es 0 a no ser que las entradas sean todas 0

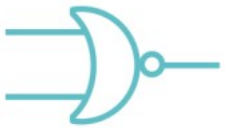
Tabla de verdad

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

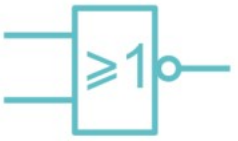
Función lógica

$F=\overline{A+B}$

Símbolo ANSI "americano"



Símbolo IEC "europeo"



Puertas AND y NAND

- La **puerta AND** realiza la función **producto lógico** o **función AND**. La función toma valor lógico 1 cuando todas las entradas valen 1 y toma el valor 0 cuando alguna de las entradas vale 0

Tabla de verdad

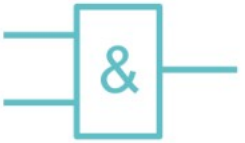
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Función lógica

$F=A.B$

Símbolo ANSI "americano"

Símbolo IEC "europeo"



- La **puerta NAND** realiza la **función producto lógico negado** o **función NAND**. Es la función contraria a la AND. La función toma valor lógico 0 cuando todas las entradas valen 1 y toma el valor 1en el resto de los casos

Tabla de verdad

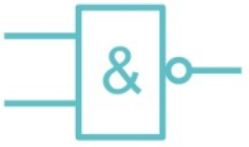
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Función lógica

$F=\overline{A.B}$

Símbolo ANSI "americano"

Símbolo IEC "europeo"



Puertas XOR y XNOR

- La **puerta XOR** realiza la función OR EXCLUSIVA. La función toma valor lógico 1 cuando las entradas tienen distinto valor y toma el valor 0 cuando las entradas son iguales

Tabla de verdad

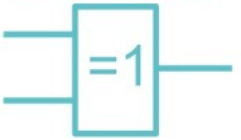
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Función lógica

$F=A \oplus B$

Símbolo ANSI "americano"

Símbolo IEC "europeo"



- La **puerta XNOR** realiza la función NOR EXCLUSIVA. Es la función contraria a la XOR. La función toma valor lógico 1 cuando las entradas tienen el mismo valor y toma el valor 0 cuando las entradas son distintas.

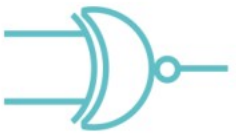
Tabla de verdad

A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

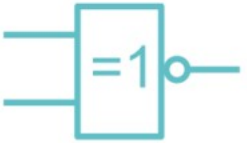
Función lógica

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

Símbolo ANSI "americano"



Símbolo IEC "europeo"

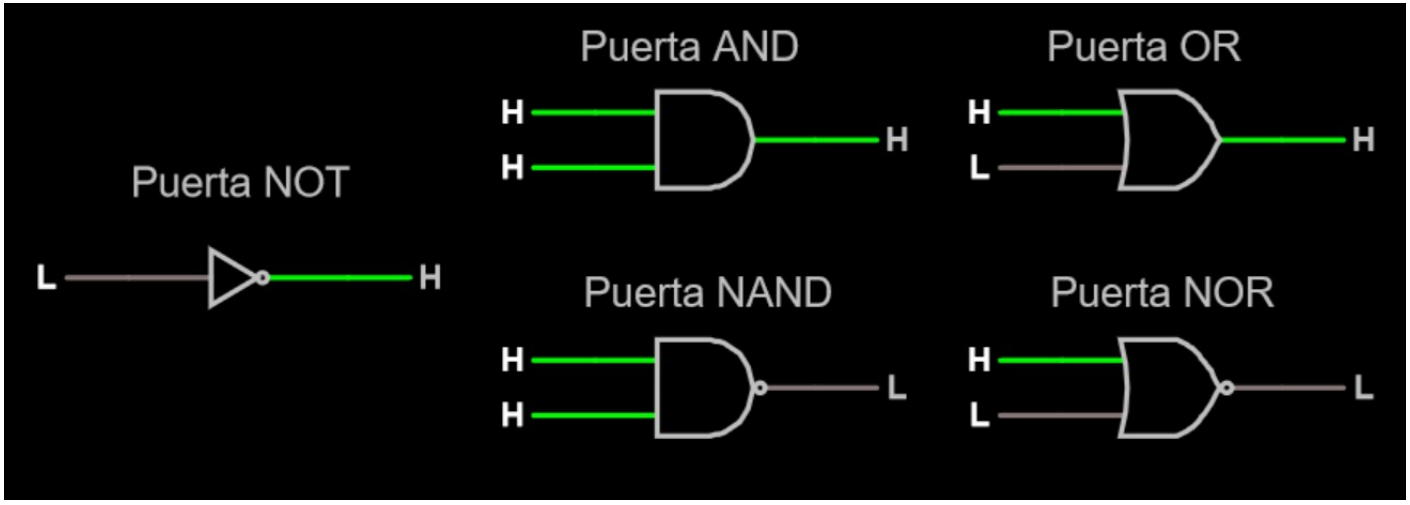


Puertas lógicas en el simulador

En el simulador todos los elementos para trabajar con puertas lógicas los tenemos en la categoría “Puertas Lógicas, Entrada y Salida”. Aquí además de un componente por cada tipo de puerta tenemos:

- **Entrada lógica (i de input)**: Lo utilizaremos para darle valores de entrada a nuestro circuito digital, que podrán ser 0 o 1, aunque por defecto vienen representadados con una H (High) y una L (Low) para el 1 y el 0 respectivamente
- **Salida lógica (o de output)**: Lo utilizaremos para ver el valor de la salida de un circuito digital, que también se representará con una H o una L

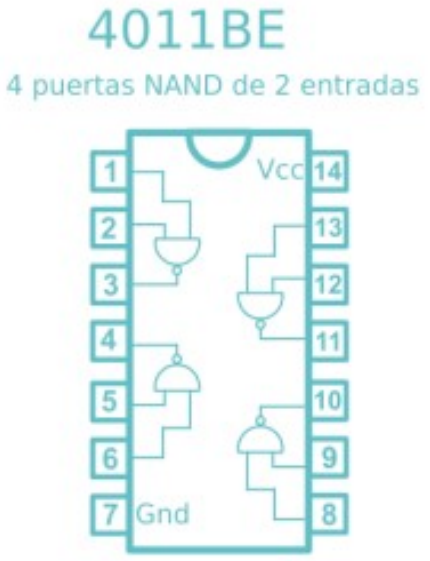
Entrada Lógica	i
Salida Lógica	o
Inversor	1
Puerta NAND	@
Puerta NOR	#
Puerta AND	2
Puerta OR	3
Puerta XOR	4

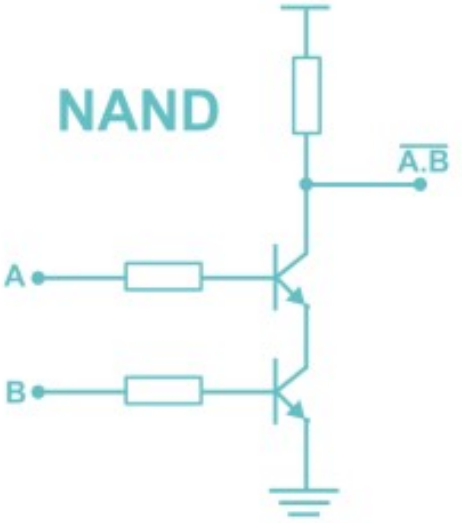


Enlace

Las puertas lógicas como circuitos integrados

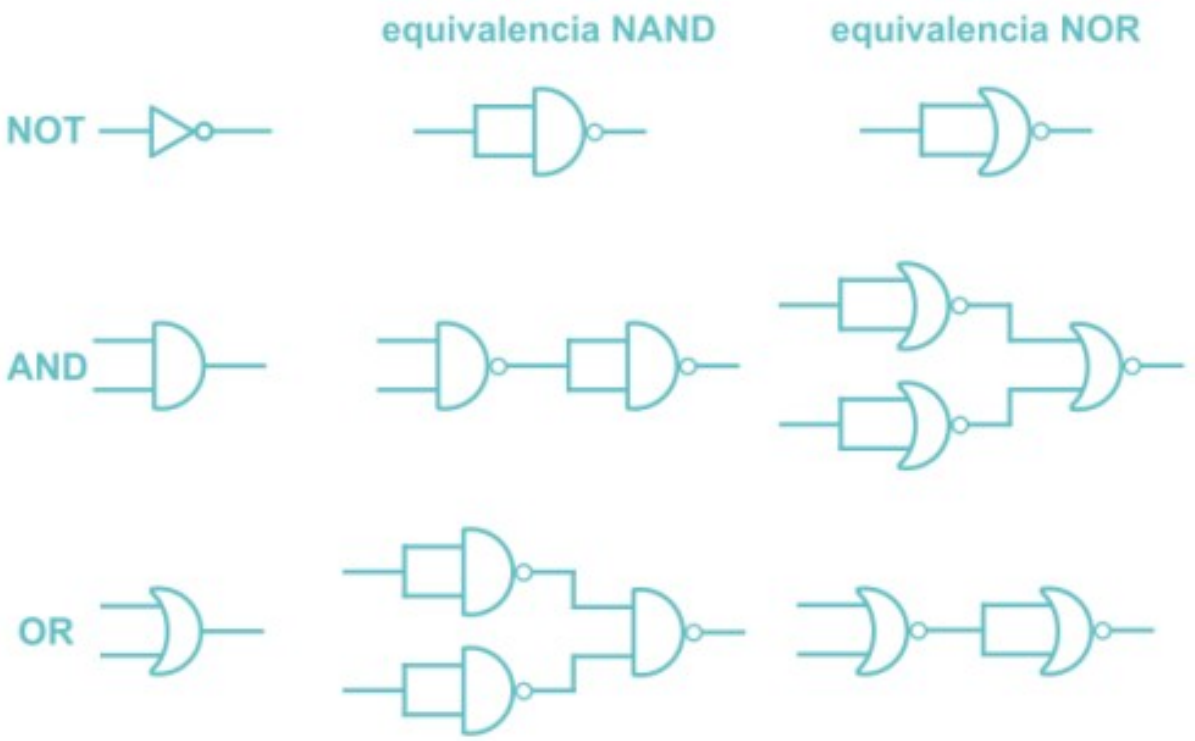
- Las puertas lógicas se usan en forma de circuitos integrados. El de la imagen, el 4011BE, está formado por cuatro puertas NAND de dos entradas.
- Interiormente este chip para cada puerta contiene **resistencias y transistores** conectados de tal modo que se cumplen eléctricamente la función lógica correspondiente





Equivalencias puertas lógicas

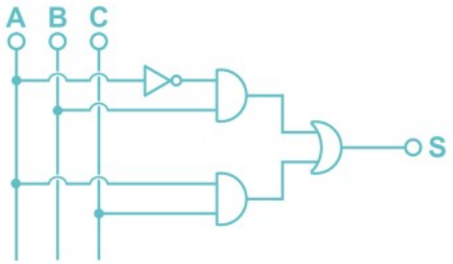
Las puertas lógicas NAND y NOR se denominan **puertas universales**, ya que pueden sustituir al resto de puertas. Esto es algo muy útil a la hora de montar los circuitos, ya que los simplifica mucho, y nos permite reducir el número de circuitos integrados necesarios para el montaje

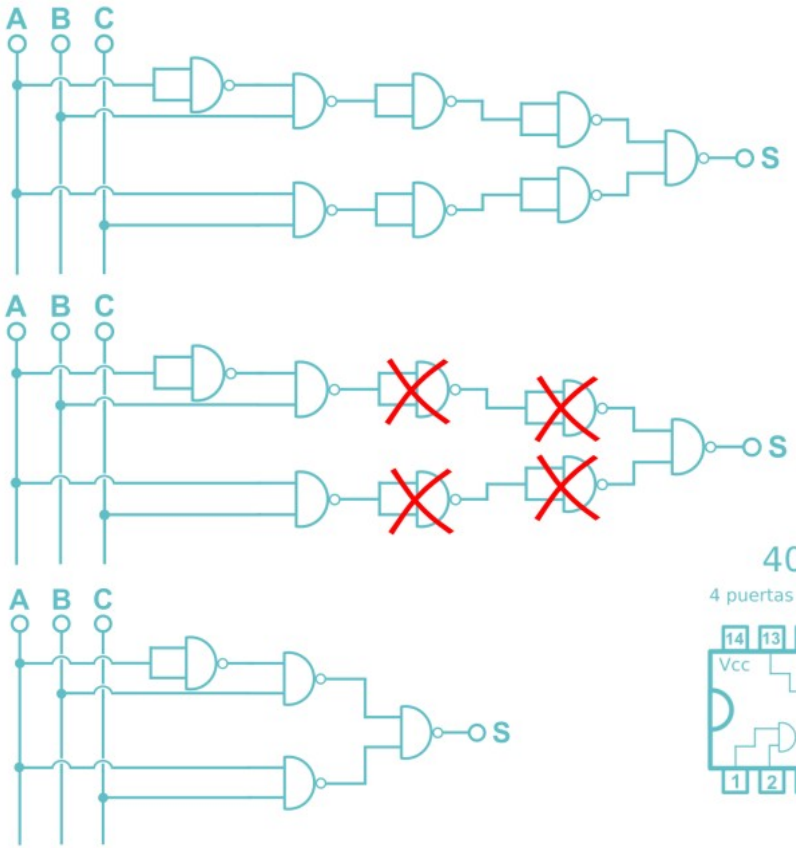


Simplificación de circuitos con equivalencias

El uso de puertas universales simplifica mucho la cantidad de circuitos integrados necesarios. Vamos a verlo con un ejemplo

- En el circuito de la izquierda necesitamos tres circuitos integrados, para los tres tipos de puertas que tiene
- En el de la derecha aplicamos la equivalencia con puertas NAND, simplificamos (dos negaciones es una afirmación) y al final nos quedamos con un circuito que necesitaría un único circuito integrado





Usos de las puertas lógicas

Como puedes ver las puertas lógicas se construyen utilizando componentes eléctricos y electrónicos básicos como resistencias y transistores

Entre otras cosas la puertas lógicas nos permitirán realizar:

- **Circuitos de control**
- **Operaciones aritméticas** con números binarios, que son utilizados entre otros por la ALU de las CPUs de los ordenadores

Veremos ejemplos de uso de puertas lógicas en cada uno de estos casos

Tarea: Construir puertas lógicas con transistores

Antes pusimos el esquema eléctrico de una puerta NAND utilizando transistores NPN y resistencias. El resto de puertas se pueden implementar en circuitos que emplean los mismos componentes.

Lo que tienes que hacer en esta tarea es implementar la puerta NAND según el circuito de arriba y después buscar en Internet como serían los esquemas para algunas de las otras puertas (AND, OR, NOR y NOT) e implementarlas todas utilizando en un mismo diseño de circuito.

Es posible que encuentres varias alternativas para la implementación de la puerta lógica, entre ellas trata de utilizar siempre las que **utilicen diodos o transistores bipolares NPN**.

Para todas ellas:

- Verifica el correcto funcionamiento utilizando las tablas de verdad como referencia
- Utiliza etiquetas (Draw → Output and Labels → Add text) para ayudar en la identificación de la puerta que implementa cada circuito
- Para las entradas utilizarás las entradas lógicas y para la salida podrás utilizar un led o una salida lógica

Entregarás en el aula virtual:

- Una captura del circuito completo
- La exportación del circuito como URL para ponerlo en los comentarios de la entrega de la tarea

Circuito → Tabla de verdad y expresión lógica

Al igual que para cada puerta lógica tenemos una tabla de verdad y una función lógica, cualquier circuito que emplee varias puertas lógicas podemos expresarlo de la misma forma:

- Una tabla de verdad nos dará **para todas las combinaciones de entradas** el valor de la salida correspondiente
- Una expresión lógica resultado de combinar todas las funciones lógicas

Vemos un ejemplo para el siguiente circuito con tres entradas, una salida y dos puertas lógicas, una AND y una OR:

Circuito → Tabla de verdad y expresión lógica

Tabla de verdad:

Hacemos una tabla con todas las combinaciones de entradas. Como tenemos tres entradas que pueden tomar valores 0 o 1 tenemos 2^3 combinaciones.

Para cada combinación evaluamos el valor de salida que tendrá evaluando cada puerta lógica con la entrada que le corresponda en la combinación.

En este caso la salida de la primera puerta OR es la entrada de la segunda puerta AND

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Expresión lógica:

La expresión se construye utilizando los operador lógicos (AND → Producto lógico y OR → Suma lógica) de una forma similar a cualquier expresión matemática:

$$S = (A + B) \cdot C$$

Tabla de verdad → Expresión lógica I

Antes partimos de un circuito ya construido, pero cuando construimos un circuito en realidad se suele empezar por la tabla de verdad.

Vamos a ver como obtener la expresión lógica a partir de la tabla de verdad:

- Localizamos las combinaciones que dan salida 1
- A partir de estas combinaciones creamos una expresión como un producto lógico entre todas las entradas, en las que las entradas con valor 0 serán negadas
- La expresión lógica final será el resultado de la suma lógica de todas las anteriores

Vemos un ejemplo con la tabla de verdad anterior:

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$\overline{A} \cdot B \cdot C$

$A \cdot \overline{B} \cdot C$

$A \cdot B \cdot C$

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

Tabla de verdad → Expresión lógica II

Podemos ver que la expresión que hemos obtenido a partir de la tabla de verdad aparentemente no se parece en nada a la expresión lógica que sacamos a partir de las puertas lógicas, pero en realidad son equivalentes.

El álgebra de Boole nos proporciona una serie de reglas que aplicadas a una expresión lógica nos permite simplificarla. Vemos como las aplicaríamos para nuestro caso:

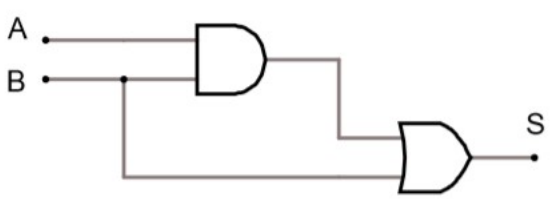
$\overline{A}BC + A\overline{B}C + ABC$ → Regla 6 → $\overline{A}BC + AC$ → Regla 12 → $(\overline{A}B + A)\,C$ → Regla 11 → **$(A + B)\cdot C$**

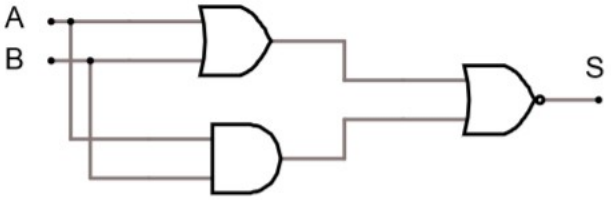
1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

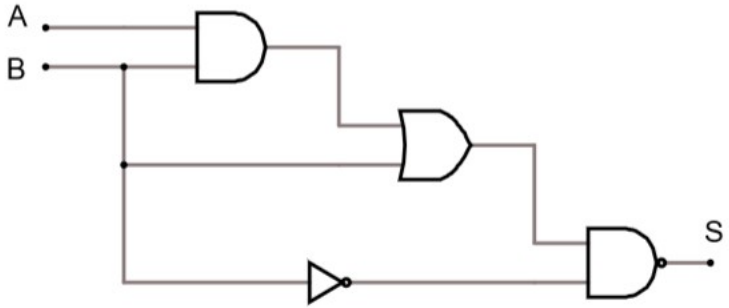
Tarea: Obtención de expresión y tabla de verdad

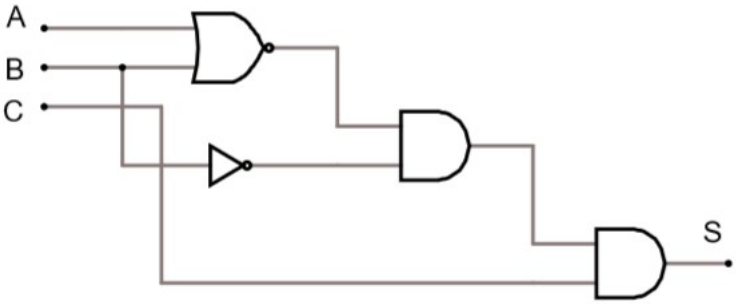
Para los siguientes circuitos con puertas lógicas obtén:

- Su tabla de verdad
- Su expresión lógica









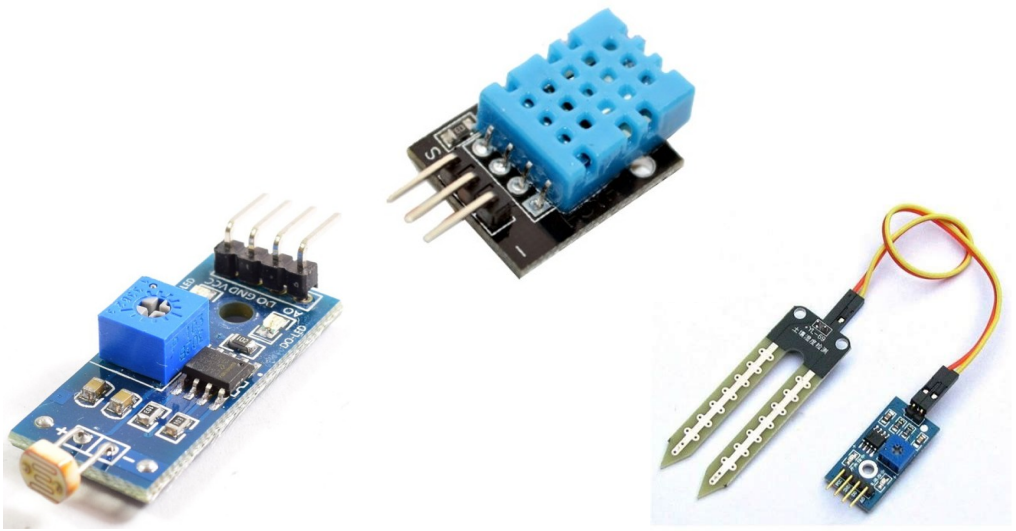
Circuitos de control con puertas lógicas

Sistema de control de riego

Vamos a implementar un circuito de control de un sistema de riego como ejemplo de aplicación de la electrónica digital y en concreto de las puertas lógicas

Nuestro circuito tiene que controlar el encendido y el apagado del sistema de riego de tal forma que se tiene que encender de acuerdo a las siguientes condiciones de temperatura, luz y humedad:

- Siempre que la tierra esté seca, a no ser que sea de noche y haga frío.
- Siempre que haga calor, si es de noche



Definición de variables y valores

Nuestro circuito tendrá por tanto tres entradas, la que nos dan los tres sensores, y un salida que será encender o apagar el riego.

Vamos a suponer que ya tenemos resuelta la parte la que hemos digitalizado las señales de los sensores y que nos devuelven 0 o 1 dependiendo del valor leído, y que a partir de ahí nosotros construimos nuestro circuito de control.

Asía asignamos a cada entrada y salida un nombre de variable y un valor digital 0 o 1.

Tipo	Nombre descriptivo	Nombre variable	Valor digital
Entrada	Luz	A	Si hay luz el valor 1 y oscuridad 0
Entrada	Temperatura	B	Calor es 1 y frío 0
Entrada	Humedad	C	Tierra seca es 1 y húmeda 0
Salida	Riego	S	Regar es 1 y no regar es 0

Tabla de verdad

Con las variables identificada y sus valores hacemos la **tabla de verdad**, en la que representamos todas las combinaciones de entradas y su salida correspondiente. Recordamos las condiciones de encendido:

- Siempre que la tierra esté seca, a no ser que sea de noche y haga frío.
- Siempre que haga calor, si es de noche

Nombre descriptivo	Nombre variable	Valores
Luz	A	Si hay luz el valor 1 y oscuridad 0
Temperatura	B	Calor es 1 y frío 0
Humedad	C	Tierra seca es 1 y húmeda 0
Riego	S	Regar es 1 y no regar es 0

A	B	C	S	
0	0	0	0	
0	0	1	0	
0	1	0	1	Es de noche, hace calor y tierra húmeda
0	1	1	1	Es de noche, hace calor y tierra seca
1	0	0	0	
1	0	1	1	Es de día, hace frío y tierra seca
1	1	0	0	
1	1	1	1	Es de día, hace calor y tierra seca

Obtener la expresión booleana

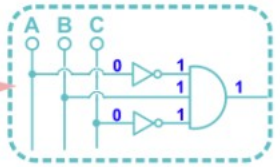
A partir de la tabla podemos obtener la expresión booleana que nos devolverá la salida. Para eso cogiendo sólo las combinaciones que activan la salida:

- Agrupamos las entradas con operaciones AND
- Agrupamos todas las combinaciones con operaciones OR

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Cuando la entrada vale 0 debemos negarla para que sea 1 en la puerta AND

$\overline{A}.B.\overline{C}$



$\overline{A}.B.C$

$A.\overline{B}.C$

$A.B.C$

La condición "y" se expresa como un producto

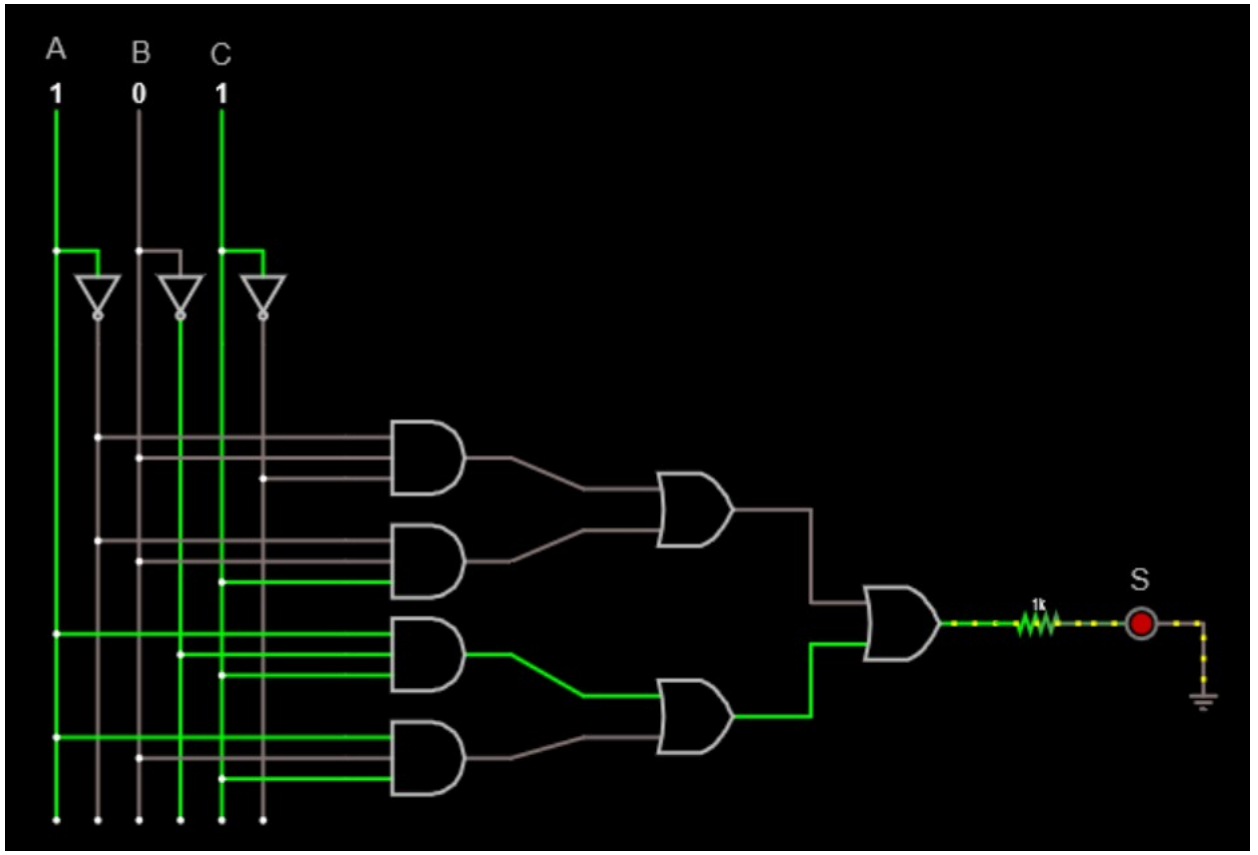
$$S= \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$$

La condición "o" se expresa como suma

Implementación de la expresión con puertas lógicas

Implementamos la expresión con puertas lógicas. Para simplificar la representación agrupamos dos puertas AND en una sola con tres entradas, pero por cada una de estas serían dos AND de 2 entradas, por lo que nos quedaría en 3 NOT, 8 AND y 3 OR:

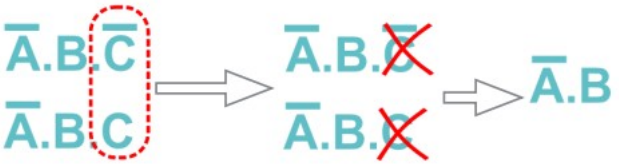
$$S= \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$$



Enlace

Simplificación de expresiones booleanas

- Podemos deducir intuitivamente algunas simplificaciones. Por observación vemos que el riego se activa siempre que sea de noche y hace calor, independientemente de la humedad, por lo que podemos hacer una simplificación



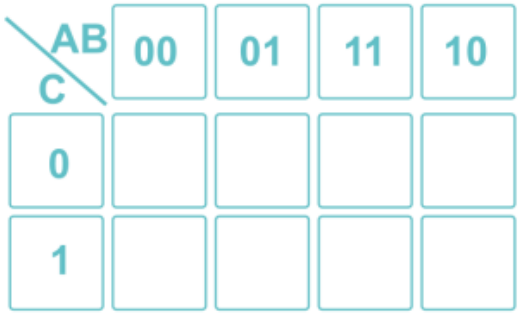
- Pero para esto el álgebra de Boole también nos proporciona una serie de reglas que podemos aplicar para simplificarla:

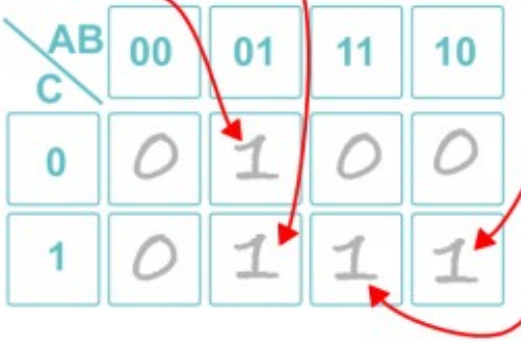
1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

Simplificación con mapas de Karnaugh I

- La aplicación de las reglas anteriores puede resultar difícil y que nos podamos saltar alguna, por lo que no siempre obtendríamos la expresión más simple
- La solución es utilizar los **mapas de Karnaugh**, que es un método gráfico en el que utilizaremos unas tablas y siguiendo una serie de reglas, nos permitirá simplificar nuestra función lógica
- En un mapa de karnaugh entre dos celdas adyacentes sólo varía el valor de una variable, por lo que tenemos que cambiar el orden de columnas o filas según el caso
- Teniendo esto en cuenta dibujamos la tabla y en esta repartimos las entradas entre los dos ejes y cubriendo con 1's o 0's según nuestras combinaciones:

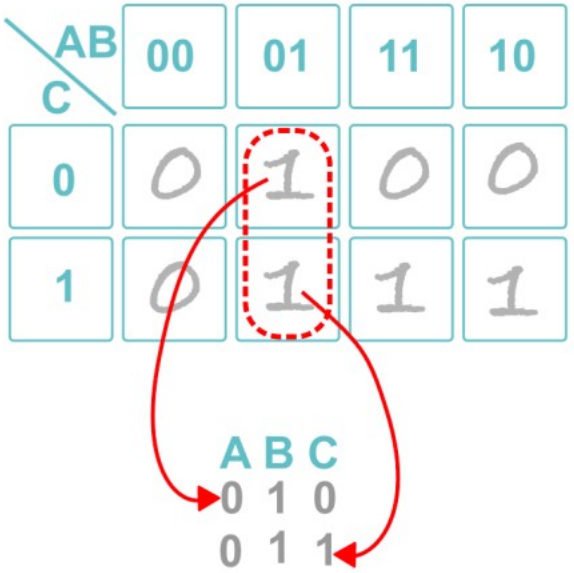
$$S = \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$$

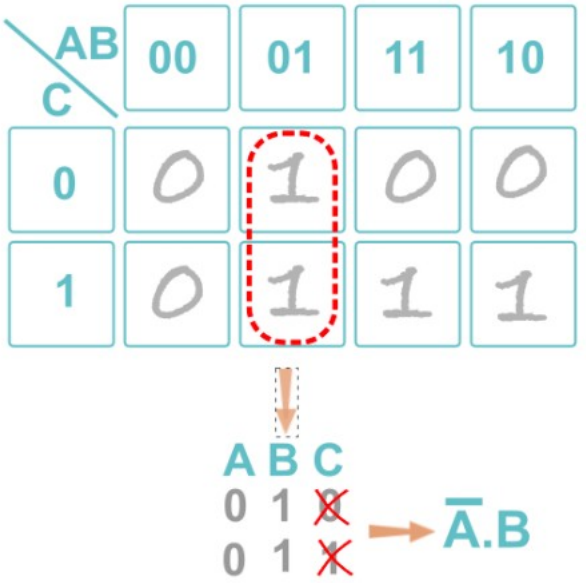




Simplificación con mapas de Karnaugh II

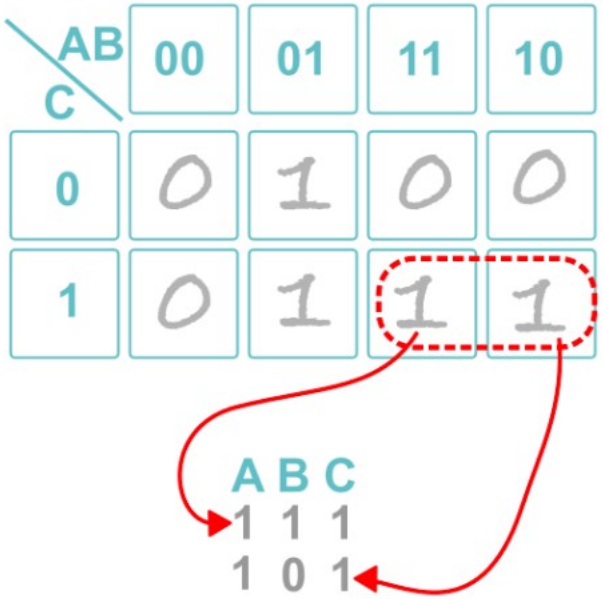
- Asociamos los 1 que sean adyacentes en horizontal o vertical, e incluso en esquinas opuestas en la misma fila o columna, siempre en grupos de potencias de 2, es decir, grupos de dos, cuatro u ocho:
- Sobre estas eliminamos la variable que varía de valor, quedándonos con una expresión más simple que auna estas combinaciones

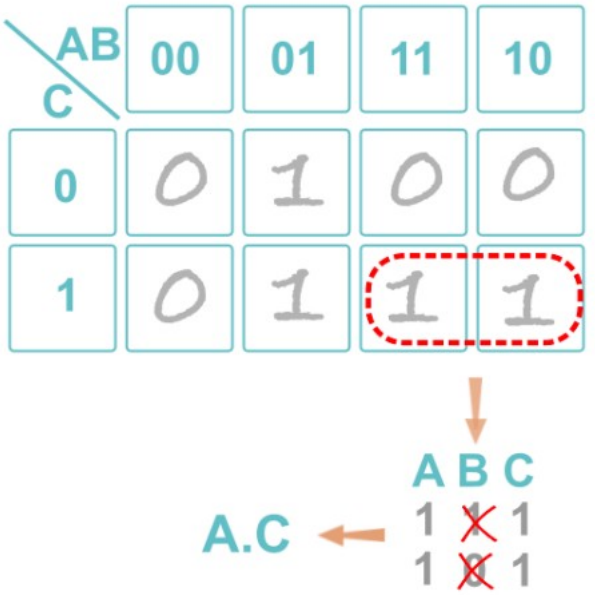




Simplificación con mapas de Karnaugh III

- También podemos agrupar los otros dos unos que aparecen en horizontal





- Reescribimos la expresión en su forma simplificada:

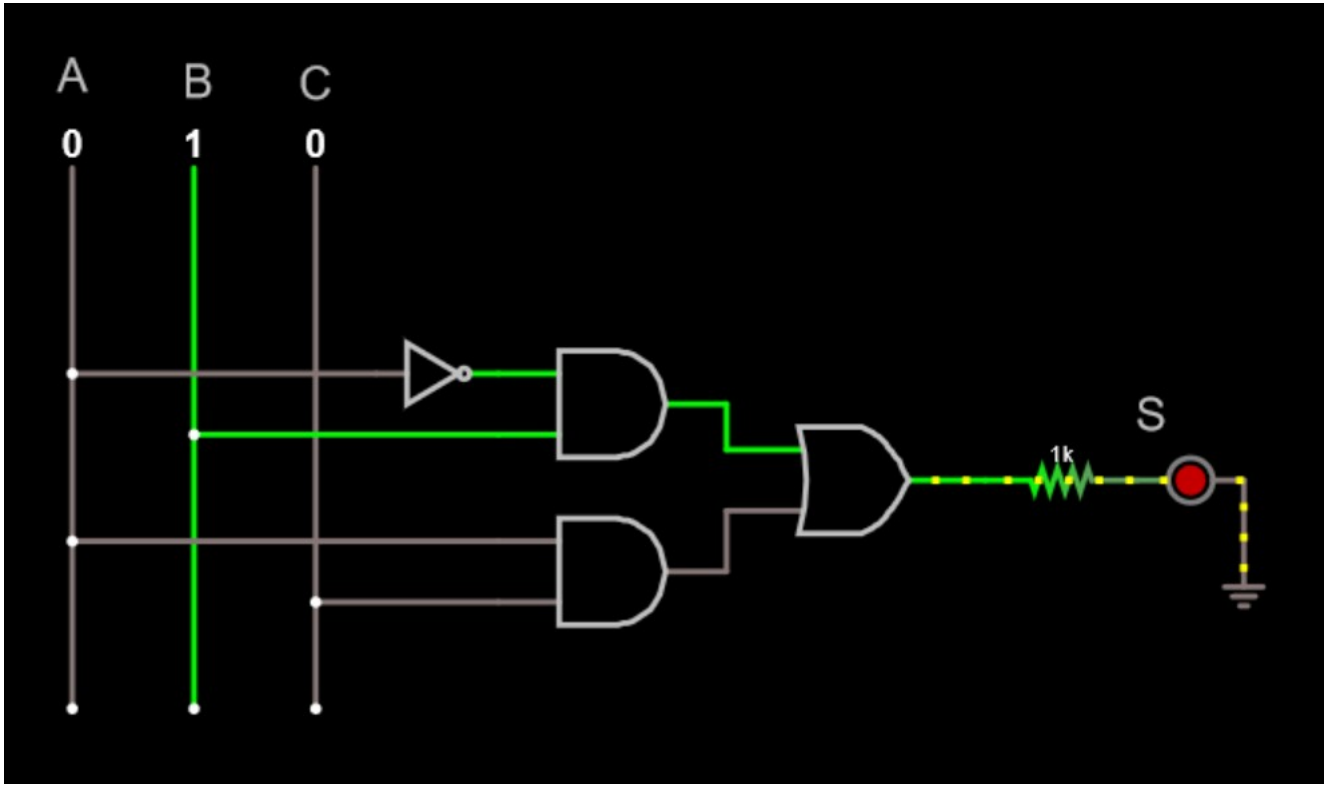
$$S= \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$$

$$S=\overline{A}.B+A.C$$

Circuito simplificado

Volvemos a implementar el circuito con la fórmula simplificada y podemos comprobar que es tan operativo como el anterior

$$S=\overline{A}.B+A.C$$

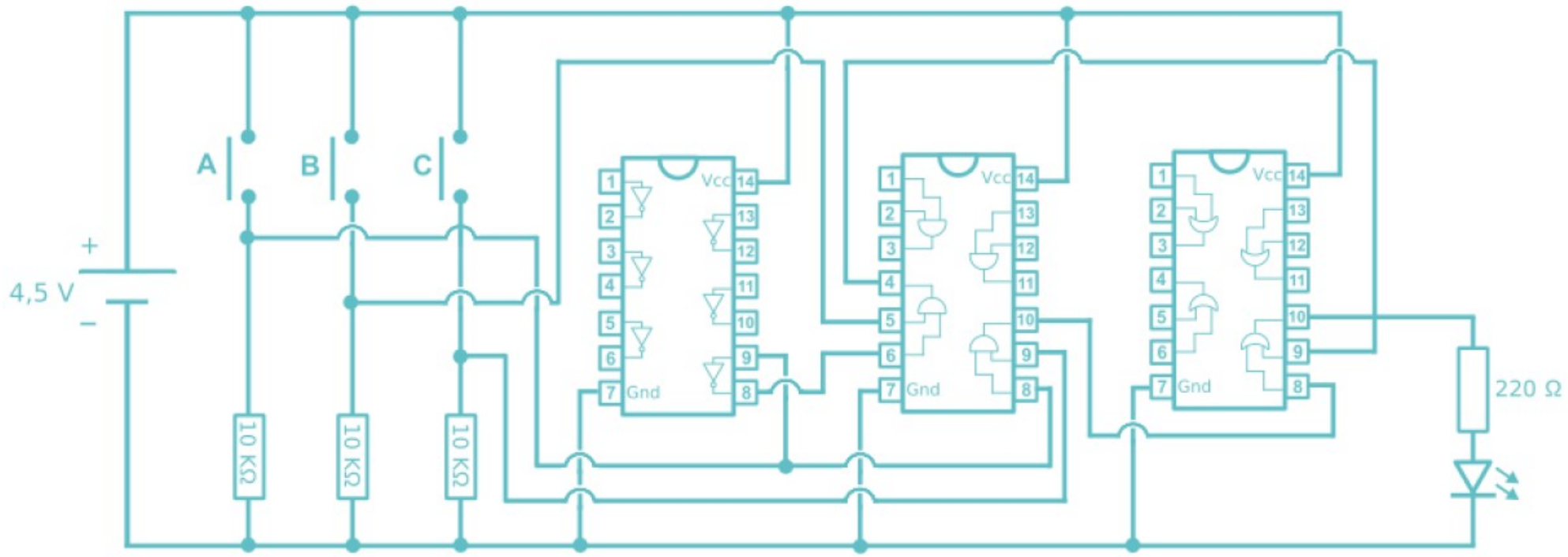


Enlace

Implementación con circuitos integrados

Para llevarlo a la realidad tendríamos que utilizar los circuitos integrados con las puertas lógicas que necesitáramos y cablearlo correspondientemente.

Opcionalmente también podríamos implementarlo todo con puertas NAND o NOR haciendo las traducciones correspondientes.



Tarea: Aplicación mapas de Karnaugh

Para la tabla de verdad dada:

- Obtén su **expresión lógica reducida** aplicando el método de los mapas de Karnaugh
- A partir de la expresión lógica generada **construye con el simulador de Falstad el circuito** que se comporte tal como se indica en la tabla

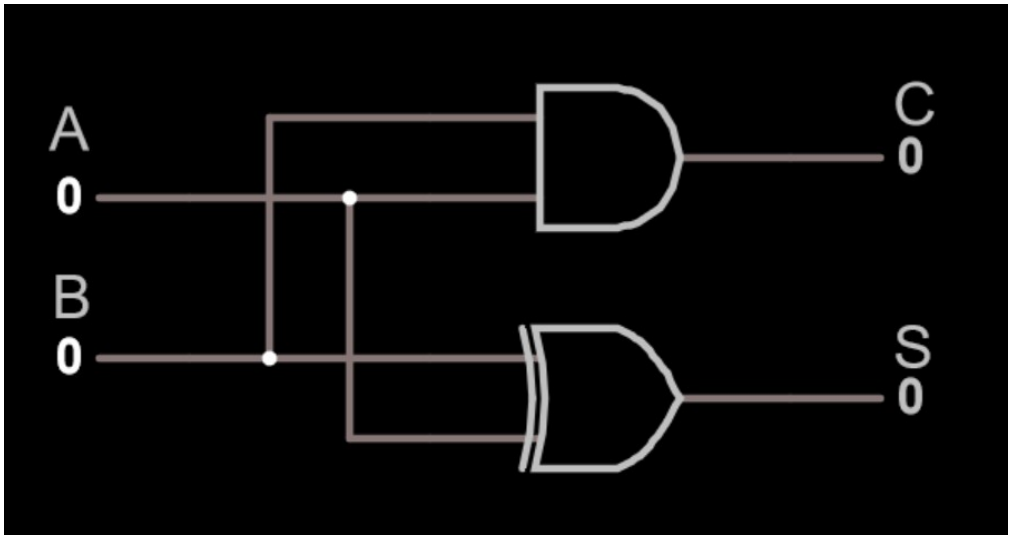
A	B	C	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Operaciones aritméticas con puertas lógicas

El semisumador (Half-adder)

- Vamos a utilizar las puertas lógicas para implementar la operación aritmética suma tal como se podría hacer en la ALU
- Vamos a ir empezando con un circuito llamado **semisumador**, que es un **sumador de números de 1 bit** que tiene **dos salidas**:
  - S → El resultado de la suma como número de 1 bit, que es pasar las dos entradas por una puerta XOR (O exclusivo)
  - C → El acarreo, que es una salida que se pone a 1 cuando la suma de estos dos números no se puede almacenar en la salida ("la que me llevo") , que será en los casos en los que intentamos sumar 1+1 en binario. Esta es pasar las dos entradas por una puerta AND

Entradas		Salidas	
A	**B**	**S**	**C**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



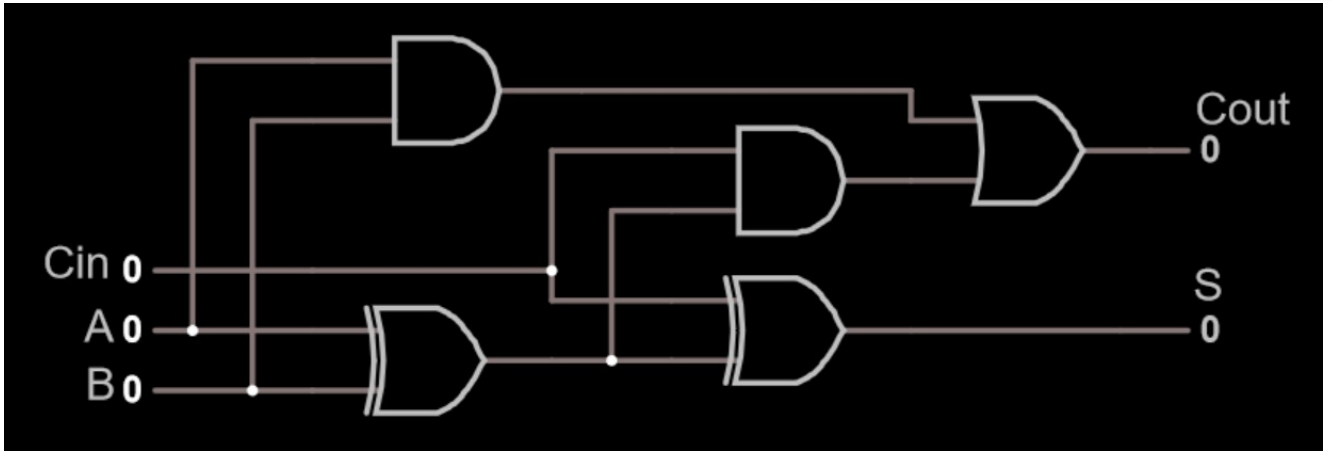
Enlace

El sumador completo (Full-adder)

Vamos a complicar un poco el semisumador añadiendo una entrada adicional, que sea el bit de acarreo que venga de una suma anterior

Vemos su tabla de verdad con las dos salidas y su implementación con puertas lógicas

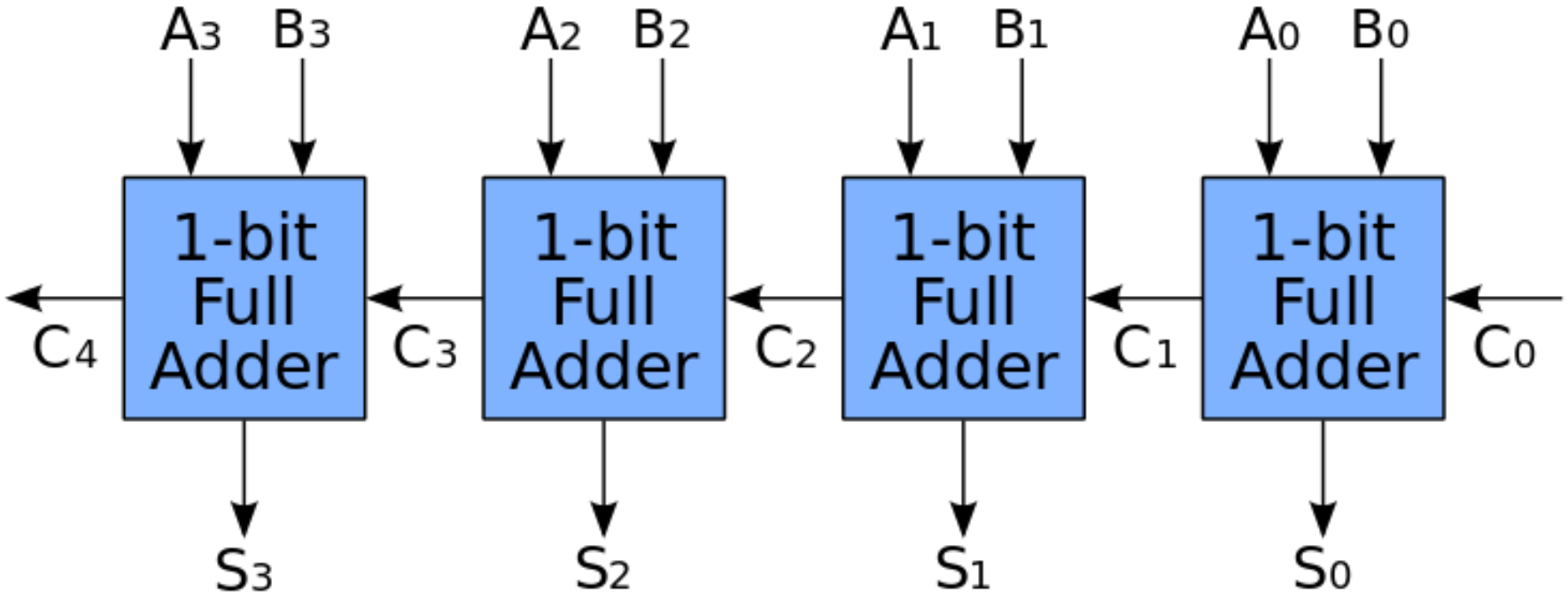
Entradas			Salidas	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Enlace

El sumador de n-bits

Si utilizamos varios sumadores completos enlazando el acarreo de salida con el acarreo de entrada del siguiente implementamos la suma de dos números con tantos bits como sumadores completos combinemos



Tarea: Implementa un sumador de 4 bits

- Vamos a probar a implementar con puertas lógicas la suma de números de 4 bits
- Para esto utilizaremos el sumador completo, pero en vez de construirlo desde cero utilizaremos el que el simulador de falstad nos proporciona en el menú Circuit
- Etiqueta cada una de las entradas con la forma A0, A1, A2, A3, B0, B1, B2 y B3 para distinguir los 4 bits del operador A y los 4 del operador B
- Etiqueta igualmente las salidas en la forma S0, S1, S2 y S3 y uno adicional C para el acarreo
- Coloca las entradas y salidas de forma que sean los más legibles posibles para facilitar su lectura y conversión a decimal

Entregarás en el aula virtual:

- Una captura del circuito completo
- La exportación del circuito como URL para ponerlo en los comentarios de la entrega de la tarea