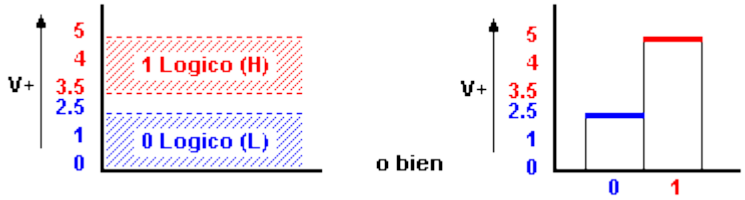
QUÉ ES ELECTRÓNICA DIGITAL...?

Obviamente es una ciencia que estudia las señales eléctricas, pero en este caso son señales discretas, es decir, están bien identificadas, razón por la cual a un determinado nivel de tensión se lo llama estado alto (High) o Uno lógico; y a otro, estado bajo (Low) o Cero lógico.

Suponte que las señales eléctricas con que trabaja un sistema digital son 0V y 5V. Es obvio que 5V será el estado alto o uno lógico, pero bueno, habrá que tener en cuenta que existe la Lógica Positiva y la Lógica Negativa, veamos cada una de ellas.

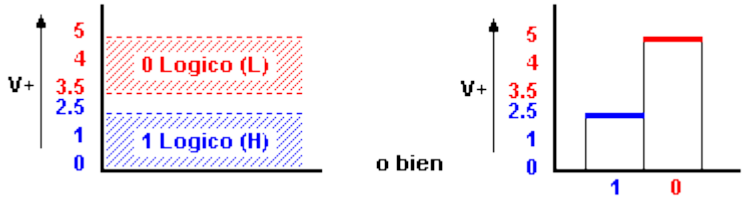
LÓGICA POSITIVA

En esta notación al 1 lógico le corresponde el nivel más alto de tensión (positivo, si quieres llamarlo así) y al 0 lógico el nivel mas bajo (que bien podría ser negativo), pero que ocurre cuando la señal no está bien definida...?. Entonces habrá que conocer cuales son los límites para cada tipo de señal (conocido como tensión de histéresis), en este gráfico se puede ver con mayor claridad cada estado lógico y su nivel de tensión.

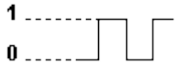


LÓGICA NEGATIVA

Aquí ocurre todo lo contrario, es decir, se representa al estado "1" con los niveles más bajos de tensión y al "0" con los niveles más altos.



Por lo general se suele trabajar con lógica positiva, y así lo haremos en este tutorial, la forma más sencilla de representar estos estados es como se puede ver en el siguiente gráfico.



De ahora en más ya sabrás a que nos referimos con estados lógicos 1 y 0, de todos modos no viene nada mal saber un poco más... ;-)

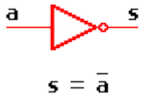
COMPUERTAS LÓGICAS

Las compuertas lógicas son dispositivos que operan con aquellos estados lógicos mencionados en la página anterior y funcionan igual que una calculadora, de un lado ingresas los datos, ésta realiza una operación, y finalmente, te muestra el resultado.

Cada una de las compuertas lógicas se las representa mediante un Símbolo, y la operación que realiza (Operación lógica) se corresponde con una tabla, llamada Tabla de Verdad, vamos con la primera...

COMPUERTA NOT

Se trata de un inversor, es decir, invierte el dato de entrada, por ejemplo; si pones su entrada a 1 (nivel alto) obtendrás en su salida un 0 (o nivel bajo), y viceversa. Esta compuerta dispone de una sola entrada. Su operación lógica es s igual a a invertida



a	s
0	1
1	0

COMPUERTA AND

Una compuerta AND tiene dos entradas como mínimo y su operación lógica es un producto entre ambas, no es un producto aritmético, aunque en este caso coincidan. **Observa que su salida será alta si sus dos entradas están a nivel alto**

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

COMPUERTA OR

Al igual que la anterior posee dos entradas como mínimo y la operación lógica, será una suma entre ambas... *Bueno, todo va bien hasta que 1 + 1 = 1*, el tema es que se trata de una compuerta O Inclusiva es como a y/o b
Es decir, basta que una de ellas sea 1 para que su salida sea también 1

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

COMPUERTA OR-EX O XOR

Es OR EXclusiva en este caso con dos entradas (puede tener mas, claro...!) y lo que hará con ellas será una suma lógica entre a por b invertida y a invertida por b.
Al ser O Exclusiva su salida será 1 si una y sólo una de sus entradas es 1

$s = a.\bar{b} + \bar{a}.b$

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

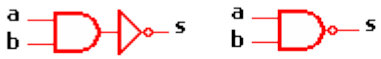
Estas serían básicamente las compuertas mas sencillas. Es momento de complicar esto un poco más...

COMPUERTAS LÓGICAS COMBINADAS.

Al agregar una compuerta NOT a cada una de las compuertas anteriores, los resultados de sus respectivas tablas de verdad se invierten, y dan origen a tres nuevas compuertas llamadas NAND, NOR y NOR-EX... Veamos ahora como son y cual es el símbolo que las representa...

COMPUERTA NAND

Responde a la inversión del producto lógico de sus entradas, en su representación simbólica se reemplaza la compuerta NOT por un círculo a la salida de la compuerta AND.

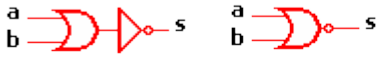


$s = \overline{a \cdot b}$

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

COMPUERTA NOR

El resultado que se obtiene a la salida de esta compuerta resulta de la inversión de la operación lógica o inclusiva es como un no a y/o b. Igual que antes, solo agregas un círculo a la compuerta OR y ya tienes una NOR.

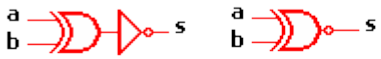


$s = \overline{a + b}$

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

COMPUERTA NOR-EX

Es simplemente la inversión de la compuerta OR-EX, los resultados se pueden apreciar en la tabla de verdad, que bien podrías compararla con la anterior y notar la diferencia, el símbolo que la representa lo tienes en el siguiente gráfico.

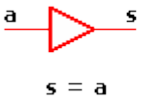


$s = \overline{a.\bar{b} + \bar{a}.b}$

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

BUFFER's

Ya la estaba dejando de lado..., no se si viene bien incluirla aquí pero de todos modos es bueno que la conozcas, en realidad no realiza ninguna operación lógica, su finalidad es amplificar un poco la señal (o refrescarla si se puede decir). Como puedes ver en el siguiente gráfico, la señal de salida es la misma que de entrada.



a	s
0	0
1	1

Hasta aquí de teoría, nos interesa más saber como se hacen evidente estos estados en la práctica, y en qué circuitos integrados se las puede encontrar y más adelante veremos unas cuantas leyes que se pueden aplicar a estas compuertas para obtener los resultados que deseas...

CIRCUITOS INTEGRADOS Y CIRCUITO DE PRUEBA.

Existen varias familias de Circuitos integrados, pero sólo mencionaré dos, los más comunes, que son los TTL y CMOS:

Estos Integrados los puedes caracterizar por el número que corresponde a cada familia según su composición. Por ejemplo;

Los TTL se corresponden con la serie 5400, 7400, 74LSXX, 74HCXX, 74HCTXX etc. algunos 3000 y 9000.

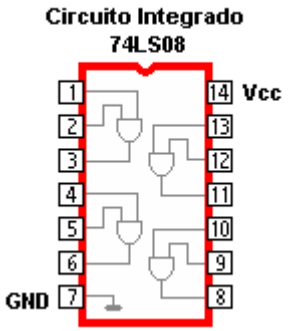
Los C-MOS y MOS se corresponde con la serie CD4000, CD4500, MC14000, 54C00 ó 74C00. en fin...

La pregunta de rigor... Cual es la diferencia entre uno y otro...?, veamos... yo comencé con los C-MOS, ya que disponía del manual de estos integrados, lo bueno es que el máximo nivel de tensión soportado llega en algunos casos a +15V, (especial para torpes...!!!), mientras que para los TTL el nivel superior de tensión alcanza en algunos casos a los +12V aproximadamente, pero claro estos son límites extremos, lo común en estos últimos es utilizar +5V y así son felices.

Otra característica es la velocidad de transmisión de datos, resulta ser, que los circuitos TTL son mas rápidos que los C-MOS, por eso su mayor uso en sistemas de computación.

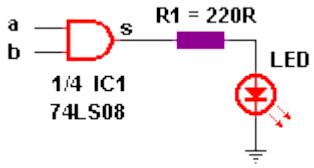
Suficiente... de todos modos es importante que busques la hoja de datos o datasheet del integrado en cuestión, distribuido de forma gratuita por cada fabricante y disponible en Internet... *donde más...?*

Veamos lo que encontramos en uno de ellos; en este caso un Circuito integrado 74LS08, un TTL, es una cuádruple compuerta AND. Es importante que notes el sentido en que están numerados los pines y esto es general, para todo tipo de integrado...

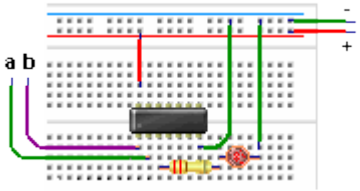


Comenzaremos con este integrado para verificar el comportamiento de las compuertas vistas anteriormente. El representado en el gráfico marca una de las compuertas que será puesta a prueba, para ello utilizaremos un fuente regulada de +5V, un LED una resistencia de 220 ohm, y por supuesto el IC que corresponda y la placa de prueba.

El esquema es el siguiente...



En el esquema está marcada la compuerta, como 1 de 4 disponibles en el Integrado 74LS08, los extremos a y b son las entradas que deberás llevar a un 1 lógico (+5V) ó 0 lógico (GND), el resultado en la salida s de la compuerta se verá reflejado en el LED, LED encendido (1 lógico) y LED apagado (0 lógico), no olvides conectar los terminales de alimentación que en este caso son el pin 7 a GND y el 14 a +5V. Montado en la placa de prueba te quedaría algo así...



Esto es a modo de ejemplo, *Sólo debes reemplazar IC1*, que es el Circuito Integrado que está a prueba para verificar su tabla de verdad.

¿Y en qué Circuito Integrado encuentro todas estas compuertas?...

Sabía que preguntarías eso... Para que puedas realizar las pruebas, te dejaré aquí los datos de algunos integrados.

UN POCO DE LEYES.

Antes de seguir... Lo primero y más importante es que trates de interpretar la forma en que realizan sus operaciones cada compuerta lógica, ya que a partir de ahora las lecciones se complican un poco más. Practica y verifica cada una de las tablas de verdad.

LEYES DE "DE MORGAN"

Se trata simplemente de una combinación de compuertas, de tal modo de encontrar una equivalencia entre ellas, esto viene a consecuencia de que en algunos casos no dispones del integrado que necesitas, pero si de otros que podrían producir los mismos resultados que estas buscando.

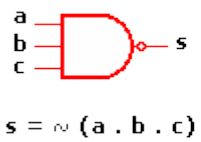
Para interpretar mejor lo que viene, considera a las señales de entrada como variables y al resultado como una función entre ellas. El símbolo de negación (operador NOT) lo representaré por "~", por ejemplo: a . ~ b significa a AND NOTb, se entendió...?

1º LEY:

El producto lógico negado de varias variables lógicas es igual a la suma lógica de cada una de dichas variables negadas. Si tomamos un ejemplo para 3 variables tendríamos..

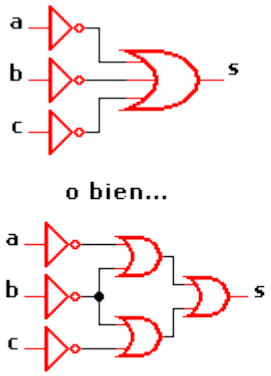
$$\sim (a.b.c) = \sim a + \sim b + \sim c$$

El primer miembro de esta ecuación equivale a una compuerta NAND de 3 entradas, representada en el siguiente gráfico y con su respectiva tabla de verdad.



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

El segundo miembro de la ecuación se lo puede obtener de dos formas...



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

$s = \sim a + \sim b + \sim c$

Fíjate que la tabla de verdad es la misma, ya que los resultados obtenidos son iguales. Acabamos de verificar la primera ley.

2º LEY:

La suma lógica negada de varias variables lógicas es igual al producto de cada una de dichas variables negadas...

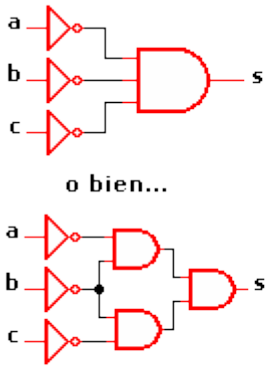
$$\sim (a + b + c) = \sim a \,.\, \sim b \,.\, \sim c$$

El primer miembro de esta ecuación equivale a una compuerta NOR de 3 entradas y la representamos con su tabla de verdad...

s = ∼(a + b + c)

a	b	c	s
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

El segundo miembro de la ecuación se lo puede obtener de diferentes forma, aquí cité solo dos...



a	b	c	s
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

s = ∼a . ∼b . ∼c

Nuevamente... Observa que la tabla de verdad es la misma que para el primer miembro en el gráfico anterior. Acabamos así de verificar la segunda ley de De Morgan.

Para concluir... Con estas dos leyes puedes llegar a una gran variedad de conclusiones, por ejemplo...

Para obtener una compuerta AND puedes utilizar una compuerta NOR con sus entradas negadas, o sea...

$$a \,.\, b = \sim(\sim a + \sim b)$$

Para obtener una compuerta OR puedes utilizar una compuerta NAND con sus entradas negadas, es decir...

$$a + b = \sim(\sim a \,.\, \sim b)$$

Para obtener una compuerta NAND utiliza una compuerta OR con sus dos entradas negadas, como indica la primera ley de De Morgan...

$$\sim (a.b) = \sim a + \sim b$$

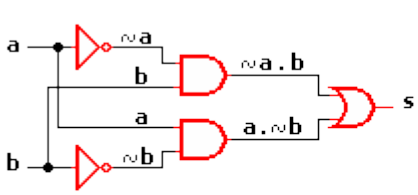
Para obtener una compuerta NOR utiliza una compuerta AND con sus entradas negadas, ...eso dice la 2º ley de De Morgan, así que... habrá que obedecer...

$$\sim(a + b) = \sim a \, . \, \sim b$$

La compuerta OR-EX tiene la particularidad de entregar un nivel alto cuando una y sólo una de sus entradas se encuentra en nivel alto. Si bien su función se puede representar como sigue...

$$s = a \, . \, \sim b + \sim a \, . \, b$$

te puedes dar cuenta que esta ecuación te indica las compuertas a utilizar, y terminarás en esto...



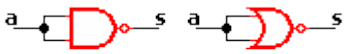
a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

$s = a.\sim b + \sim a.b$

Para obtener una compuerta NOR-EX agregas una compuerta NOT a la salida de la compuerta OR-EX vista anteriormente y ya la tendrás. Recuerda que su función es...

$$s = \sim(a \, . \, \sim b + \sim a \, . \, b)$$

Para obtener Inversores (NOT) puedes hacer uso de compuertas NOR o compuertas NAND, simplemente uniendo sus entradas.



$s = \sim a$

Existen muchas opciones más, pero bueno... ya las irás descubriendo, o las iremos citando a medida que vayan apareciendo, de todos modos valió la pena. *No crees...?*

MÁS SOBRE FUNCIONES Y OPERADORES LÓGICOS.

A estas alturas ya estamos muy familiarizados con las funciones de todos los operadores lógicos y sus tablas de verdad, todo vino bien..., pero... qué hago si dispongo de tres entradas (a, b y c) y deseo que los estados altos sólo se den en las combinaciones 0, 2, 4, 5 y 6 (decimal)...? Cómo combino las compuertas...? y lo peor, Qué compuertas utilizo...?. No te preocupes, yo tengo la solución, ...pégate un tiro... :o))

Bueno... NO...!!!, mejor no. Trataré de dar una solución verdadera a tu problema, preparado...?

MAPAS DE KARNAUGH

Podría definirlo como un método para encontrar la forma más sencilla de representar una función lógica.

Esto es... Encontrar la función que relaciona todas las variables disponibles, de tal modo que el resultado sea el que se está buscando.

Para esto vamos a aclarar tres conceptos que son fundamentales

a)- Minitérmino Es cada una de las combinaciones posibles entre todas las variables disponibles, por ejemplo con 2 variables obtienes 4 minitérminos; con 3 obtienes 8; con 4, 16 etc., como te darás cuenta se puede encontrar la cantidad de minitérminos haciendo 2^n donde n es el número de variables disponibles.

b)- Numeración de un minitérmino Cada minitérmino es numerado en decimal de acuerdo a la combinación de las variables y su equivalente en binario así...

a	b	Minit.
0	0	Minit. 0
0	1	Minit. 1
1	0	Minit. 2
1	1	Minit. 3

Bien... El Mapa de Karnaugh representa la misma tabla de verdad a través de una matriz, en la cual en la primer fila y la primer columna se indican las posibles combinaciones de las variables. Aquí tienes tres mapas para 2, 3 y 4 variables...

Para 2 Variables

a\b	0	1
0	0	1
1	2	3

Para 3 Variables

a\bc	00	01	11	10
0	0	1	3	2
1	4	5	7	6

Para 4 Variables

ab\cd	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Analicemos el mapa para cuatro variables, las dos primeras columnas (columnas adyacentes) difieren sólo en la variable d, y c permanece sin cambio, en la segunda y tercer columna (columnas adyacentes) cambia c, y d permanece sin cambio, ocurre lo mismo en las filas. En general se dice que...

Dos columnas o filas adyacentes sólo pueden diferir en el estado de una de sus variables

Observa también que según lo dicho anteriormente la primer columna con la última serían adyacentes, al igual que la primer fila y la última, ya que sólo difieren en una de sus variables

c)- Valor lógico de un minitérmino (esos que estaban escritos en rojo), bien, estos deben tener un valor lógico, y es el que resulta de la operación que se realiza entre las variables. Lógicamente 0 ó 1

Listo... Lo que haremos ahora será colocar el valor de cada Minitérmino según la tabla de verdad que estamos buscando... diablos...!!! En este momento no se me ocurre nada, bueno si, trabajemos con esta...

a\bc	00	01	11	10
0	1	1	0	0
1	1	0	0	1

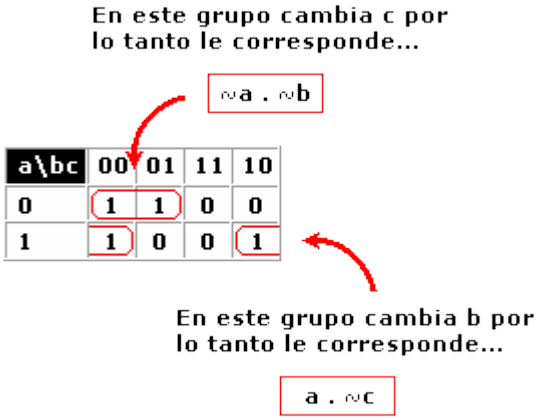
a	b	c	s
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

El siguiente paso, es agrupar los unos adyacentes (horizontal o verticalmente) en grupos de potencias de 2, es decir, en grupos de 2, de 4, de 8 etc... y nos quedaría así...

a\bc	00	01	11	10
0	1	1	0	0
1	1	0	0	1

Te preguntarás que pasó con la fila de abajo... bueno, es porque no estas atento...!!! Recuerda que la primera columna y la última son adyacentes, por lo tanto sus minitérminos también lo son.

De ahora en más a cada grupo de unos se le asigna la unión (producto lógico) de las variables que se mantienen constante (ya sea uno o cero) ignorando aquellas que cambian, tal como se puede ver en esta imagen...



Para terminar, simplemente se realiza la suma lógica entre los términos obtenidos dando como resultado la función que estamos buscando, es decir...

$$f = (\sim a \,.\, \sim b) + (a \,.\, \sim c)$$

Puedes plantear tu problema como una función de variables, en nuestro ejemplo quedaría de esta forma...

$$f(a, b, c) = S(0, 1, 4, 6)$$

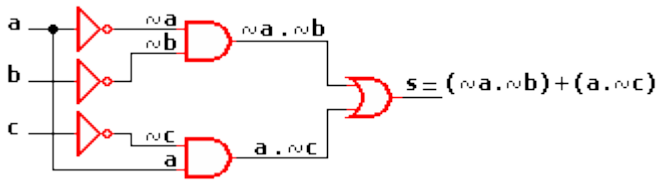
f: es la función buscada
(a, b, c): son las variables utilizadas
(0, 1, 4, 6): son los minitérminos que dan como resultado 1 o un nivel alto.
S: es La sumatoria de las funciones que producen el estado alto en dichos minitérminos.

Sólo resta convertir esa función en su circuito eléctrico correspondiente. Veamos, si la función es...

f = (~a . ~b) + (a . ~c) o sea...

(NOT a AND NOT b) OR (a AND NOT c)

El esquema eléctrico que le corresponde es el que viene a continuación...



El resultado de todo este lío, es un circuito con la menor cantidad de compuertas posibles, lo cual lo hace más económico, por otro lado cumple totalmente con la tabla de verdad planteada al inicio del problema, y a demás recuerda que al tener menor cantidad de compuertas la transmisión de datos se hace más rápida.

En fin... Solucionado el problema...!!!

Por cierto, un día, mientras merodeaba por la red me encontré con un pequeño programa que hace todo este trabajo por tu cuenta, El programa se llama Karma Creado por Pablo Fernández Fraga, mis saludos Pablo...!!! está muy, pero muy bueno...!!!

Basta por hoy, muy pronto utilizaremos toda esta teoría y el programa de pablo (Karma) para diseñar una tarjeta controladora de motores paso a paso, mientras tanto averigua como funcionan estos motores.

Saludos lógicos para todos...!!!