

Circuitos Digitales


Los ***circuitos digitales***, también conocidos como ***circuitos lógicos***, son circuitos electrónicos que procesan señales discretizadas en dos niveles, que representan los valores binarios 0 y 1, correspondientes a los valores lógicos “falso” y “verdadero”. La forma de representación depende de la tecnología usada: por ejemplo, los circuitos que usan tecnología TTL (Transistor-Transistor Logic), es decir lógica de transistores, típicamente consideran una señal entre 0 y 0.8 voltios, relativamente al nivel de tierra, una señal “baja”, correspondiente al valor 0; y una señal entre 2 y 5 voltios una señal “alta”, correspondiente al valor 1. Noten el “hueco” entre 0.8 y 2 voltios: para que la discretización sea robusta, señales en ese rango son evitadas por ser ambiguas, y de hecho no ocurren, a menos que haya un defecto de manufactura o la fuente de corriente eléctrica no tenga capacidad suficiente para el circuito. La forma de representación es solamente un detalle de implementación, importante para el funcionamiento del circuito pero irrelevante para obtener la semántica deseada: el diseñador del circuito sólo se preocupa por el aspecto lógico del mismo, la composición funcional de las operaciones booleanas requeridas. Es a ese nivel que Uds. van a operar en este curso.

Los ***circuitos digitales*** tienen dos ingredientes fundamentales, que pueden ver en los diagramas de circuitos: las **compuertas digitales** que implementan las operaciones booleanas, y las líneas de comunicación que conectan las compuertas.

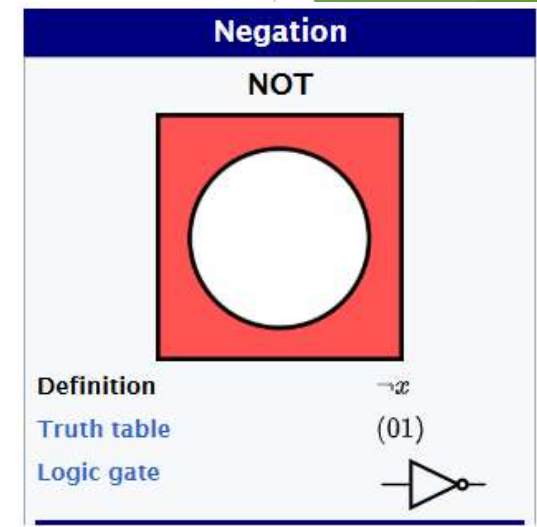
Los ***circuitos digitales*** son el fundamento de la gran mayoría de los dispositivos electrónicos actuales. Las computadoras, sistemas de comunicación, sistemas de control, y otros dispositivos electrónicos son “digitales” porque se basan en ***circuitos digitales***. Por eso decimos que vivimos en la “era digital”.

Circuitos Digitales: la compuerta NOT

Cómo interpretar el diagrama de Venn en el recuadro a la derecha

- Cada punto dentro del cuadrado representa un elemento de un dominio (conjunto) dado. También pueden imaginar cada punto como una muestra de una población. Los puntos dentro del círculo corresponden a los elementos que cumplen una proposición (propiedad) que llamamos x .
- Las áreas en rojo corresponden a los elementos que satisfacen la operación booleana que estamos describiendo. En este caso, la operación en cuestión es la negación (negation). Como pueden ver, los puntos dentro del círculo (los que cumplen x) **no** son rojos. En cambio, los puntos que no están dentro del círculo (no cumplen x) son los que satisfacen la negación, es decir satisfacen la proposición $NOT(x)$.
- La definición (definition) usando la sintaxis $\neg x$ es común para representar la negación de x en textos de lógica simbólica. Otra representación usada para la negación consiste en colocar una barra encima de la expresión que estamos negando. Notaciones como $NOT(x)$ o $not(x)$ son típicas en el mundo de la programación.
- La tabla de verdad (truth table) (0 1) corresponde a los **resultados** para valores de x en el orden (1 0). Esto puede confundirlos, ya que los valores de x , la variable independiente, no se muestran y, por razones que desconozco, el orden de esos valores “implícitos”, leyendo de izquierda a derecha, es descendiente: primero el 1 y luego el 0, como dicho antes. Por eso el resultado, que es lo único que se muestra, es (0 1). La tabla abajo muestra claramente los valores de la variable independiente y el resultado correspondiente. Como ven, la columna del resultado, leyendo de arriba para abajo, corresponde a la “tabla de verdad” en el recuadro.
- La representación de la compuerta lógica NOT, usada en diagramas de circuitos, es .
El valor de la x viene por la izquierda y el resultado $\neg x$ sale por la derecha, donde ven el círculo.
- Como verán más adelante, un círculo en la salida de una compuerta indica negación.
- <https://en.wikipedia.org/wiki/Negation>

x	$\neg x$	$NOT(x)$
1	0	
0	1	

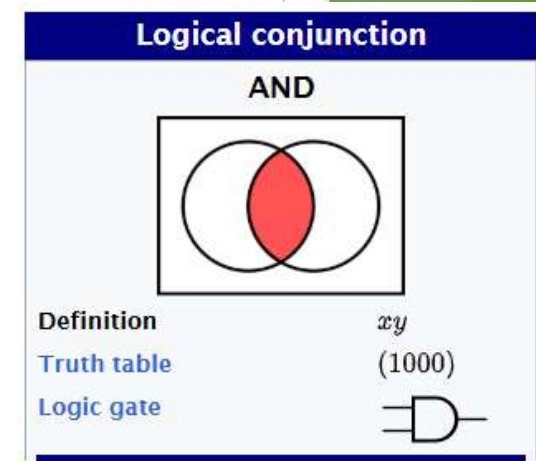


Circuitos Digitales: la compuerta AND

Como interpretar este diagrama de Venn

- La compuerta NOT, que vieron antes, computa la negación de un valor booleano. La compuerta AND computa la conjunción lógica de dos proposiciones. La palabra “and” traducida al Español es “y”: !Quiero una sopa sabrosa y caliente!
- La conjunción lógica de dos proposiciones requiere dos variables, podemos nombrarlas “x” y “y” para representar esas proposiciones. Por eso el diagrama tiene dos círculos: los puntos dentro del círculo a la izquierda satisfacen la proposición x (la sopa es sabrosa) y los puntos dentro del círculo a la derecha satisfacen la proposición y (la sopa está caliente) .. Intuitivo, ¿no?
- “mutatis mutandis”, es decir ajustando lo que hay que ajustar, las explicaciones para interpretar el recuadro del NOT aplican para el recuadro del AND. Lo que hay que tomar en cuenta para el “mutandis” es la diferencia entre la semántica del AND y el NOT. Lo mismo vale para las compuertas que siguen. Si prestaron atención, su redes neuronales ya aprendieron a hacer los ajustes necesarios.
- https://en.wikipedia.org/wiki/Logical_conjunction

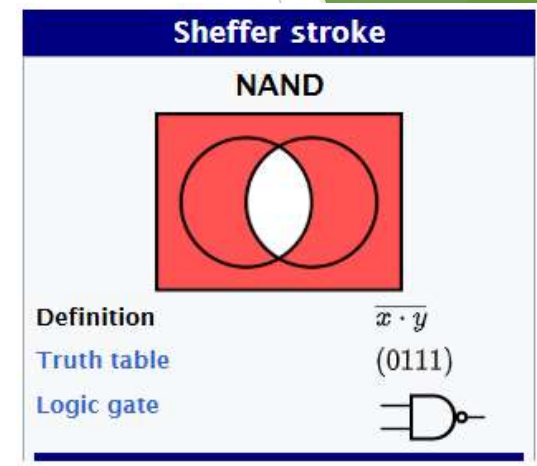
x	y	xy	$x \wedge y$	AND(x,y)
1	1	1		
1	0	0		
0	1	0		
0	0	0		



Circuitos Digitales: la compuerta NAND

- ▶ La compuerta NAND computa la negación de una conjunción lógica. Es así de fácil de entender.
- ▶ Noten el diagrama de la compuerta: es la compuerta AND con un circulito agregado en la salida .. !Es la negación del AND!
- ▶ Henry Maurice Sheffer popularizó la notación usando la barrita vertical para el operador NAND, y por eso es conocido como el “Sheffer stroke”, pero el operador y sus propiedades fueron descubiertas mucho antes.
- ▶ https://en.wikipedia.org/wiki/Sheffer_stroke

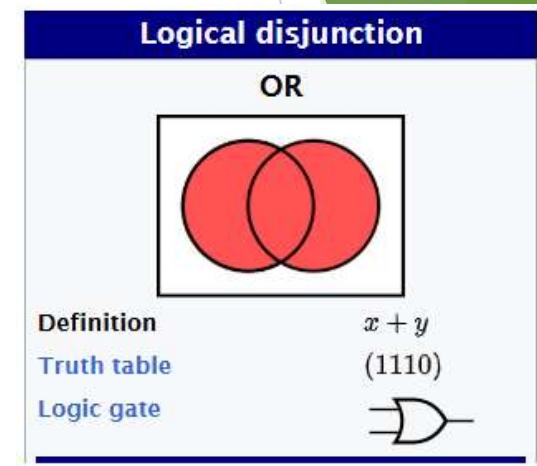
x	y	$x \uparrow y$	$x y$	$NAND(x,y)$
1	1	0		
1	0	1		
0	1	1		
0	0	1		



Circuitos Digitales: la compuerta OR

- La compuerta OR computa la disyunción lógica.
- Noten el diagrama de la compuerta: se distingue de la compuerta AND porque la entrada es curveada. 😊
- https://en.wikipedia.org/wiki/Logical_disjunction

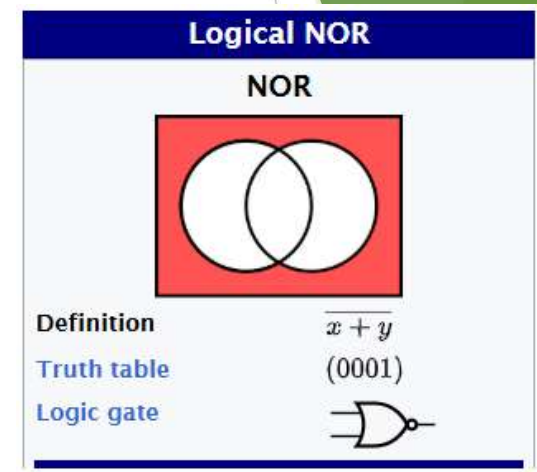
x	y	$x+y$	$x \vee y$	$OR(x,y)$
1	1	1		
1	0	1		
0	1	1		
0	0	0		



Circuitos Digitales: la compuerta NOR

- La compuerta NOR computa la negación de la disyunción lógica.
- Noten el diagrama: una compuerta OR con el circulito de negación en la salida .. ¡Es una simple negación del OR! 😊
- https://en.wikipedia.org/wiki/Logical_NOR

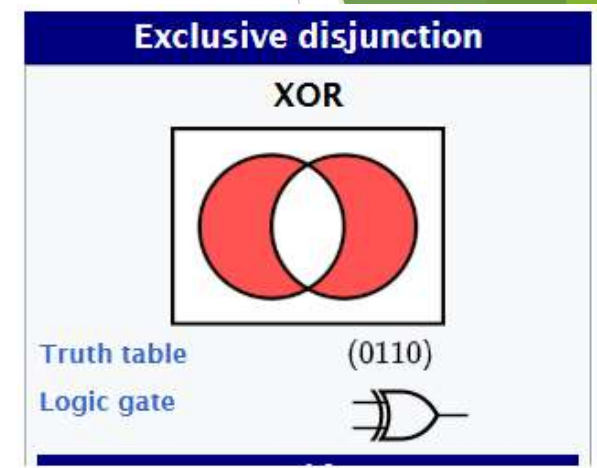
x	y	$x \downarrow y$	$x \bar{\vee} y$	$NOR(x,y)$
1	1	0		
1	0	0		
0	1	0		
0	0	1		



Circuitos Digitales: la compuerta XOR

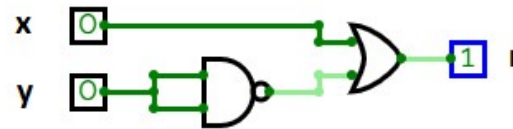
- La compuerta XOR computa la disyunción exclusiva, es decir x o y pero no ambos.
- El diagrama se parece al OR con una línea adicional en la entrada.
- https://en.wikipedia.org/wiki/Exclusive_or

x	y	$x \oplus y$	$x \underline{\vee} y$	$XOR(x,y)$
1	1	0		
1	0	1		
0	1	1		
0	0	0		



Ejercicio

- Considere este circuito digital **C**:



- Noten que tiene dos compuertas: un NAND y un OR
 - Como pueden ver, los datos de entrada son la x y la y .
 - La salida del NAND es una de las entradas del OR, y r es el resultado del circuito.
 - Ultima cucharita: el diagrama muestra lo que el circuito computa para **un** caso específico.
-
- 1) Construya la tabla de verdad de **C**
 - Muestre claramente los valores de entrada y el resultado correspondiente
 - Cubra todos los casos, siguiendo el orden descendiente que usamos en las láminas
 - 2) Implemente una función en Zilly isomorfa a **C**
 - Isomorfa significa que tiene la misma semántica
 - Sugerencia: implemente la función **nand** y la función **or**
 - Prueben la función para todos los casos - comprueben que es consistente con la tabla
 - 3) Implemente una función en C/C++ isomorfa a **C**
 - Implementen el **nand** y el **or** usando funciones de C/C++ que toman dos argumentos
 - Prueben la función para todos los casos - comprueben que es consistente con la tabla