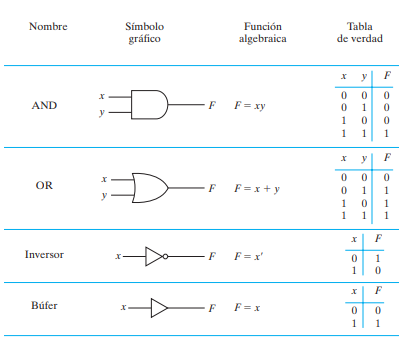
Arquitectura y organización de computadores - EIN079B

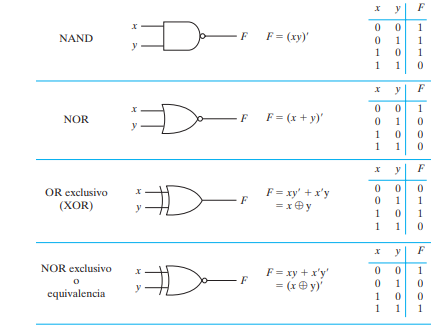
**Repositorio:** [**aoc.eajii.net**](http://aoc.eajii.net)**, usuario: aoc, clave: AOC$Ipsum.**

18-03-2024

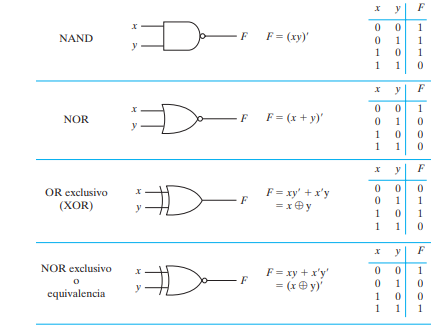
Puertas lógicas: Es un circuito compuesto por ENTRADAS (Inputs) y SALIDAS (Outputs), con una especificación funcional en base a un sistema de lógica TRUE / FALSE aplicada a la arquitectura de computadores.



* Sólo tendremos como resultado 1 cuando ambos (x,y) sean reales.
* Sólo tendremos resultado falso cuando ambas entradas sean falsas.
* Entrada unaria, el valor del bit de entrada se invierte (NOT).
* La entrada se “transfiere” o copia, por lo que su valor se mantiene.

\*  = compuerta OR.

\*Burbuja en el modelo de hardware = INVERSOR.



* Igual al circuito AND pero con un INVERSOR, por lo que los resultados son al revés.
* Mismo caso: igual al circuito OR pero con un INVERSOR.
* Útil en detección de errores, la “raya” simboliza una compuerta. Cuando ambas entradas son distintas, es 1.
* Similar al anterior, pero con un INVERSOR. Cuando ambas entradas son iguales, el resultado es 1.

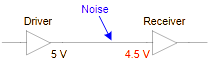
Iniciamos sesión en TinkerCAD, mis credenciales:

* [valentina.munozra@usm.cl](mailto:valentina.munozra@usm.cl)
* Thiare124

**Sobre el ruido.**

Se asocia al flujo de voltaje, pero en general el ruido se presenta ante cualquier cosa que *degrade*la señal: resistencia, ruido de la fuente de alimentación, acoplamiento a cables vecinos, etc.

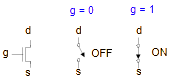
Ejemplo: una puerta (controlador) emite 5 V pero, debido a la resistencia en un cable largo, el receptor obtiene 4,5 V.



\*Esto se solucionaría añadiendo más *BÚFER* en el camino.

**Transitor.**

Dispositivo de 3 pines (patitas) que al interior contiene Silicio, y funciona de igual forma que un interruptor. Drain (d) y Source (s) están conectado cuando Gate (g) es 1.

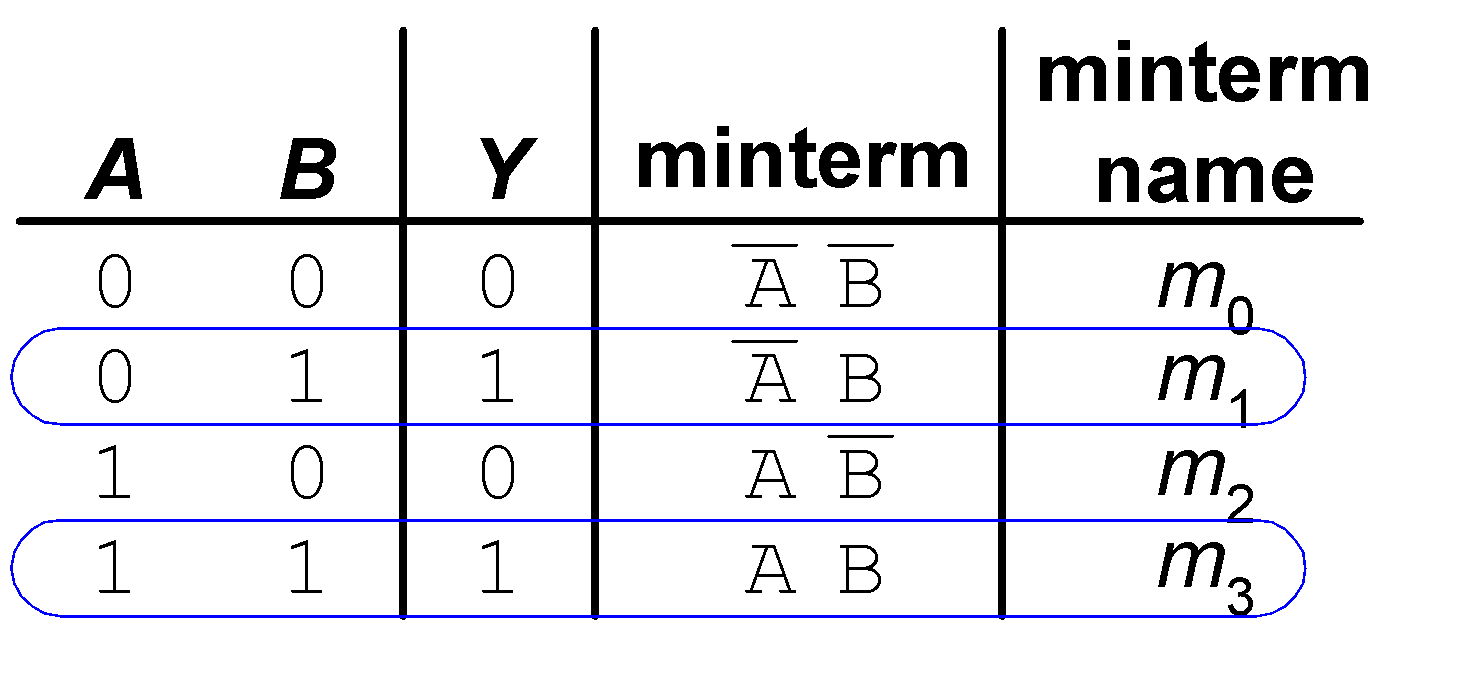


*Finaliza PPT 1.*

***Ponerse al día PPT 2 (0-7).***

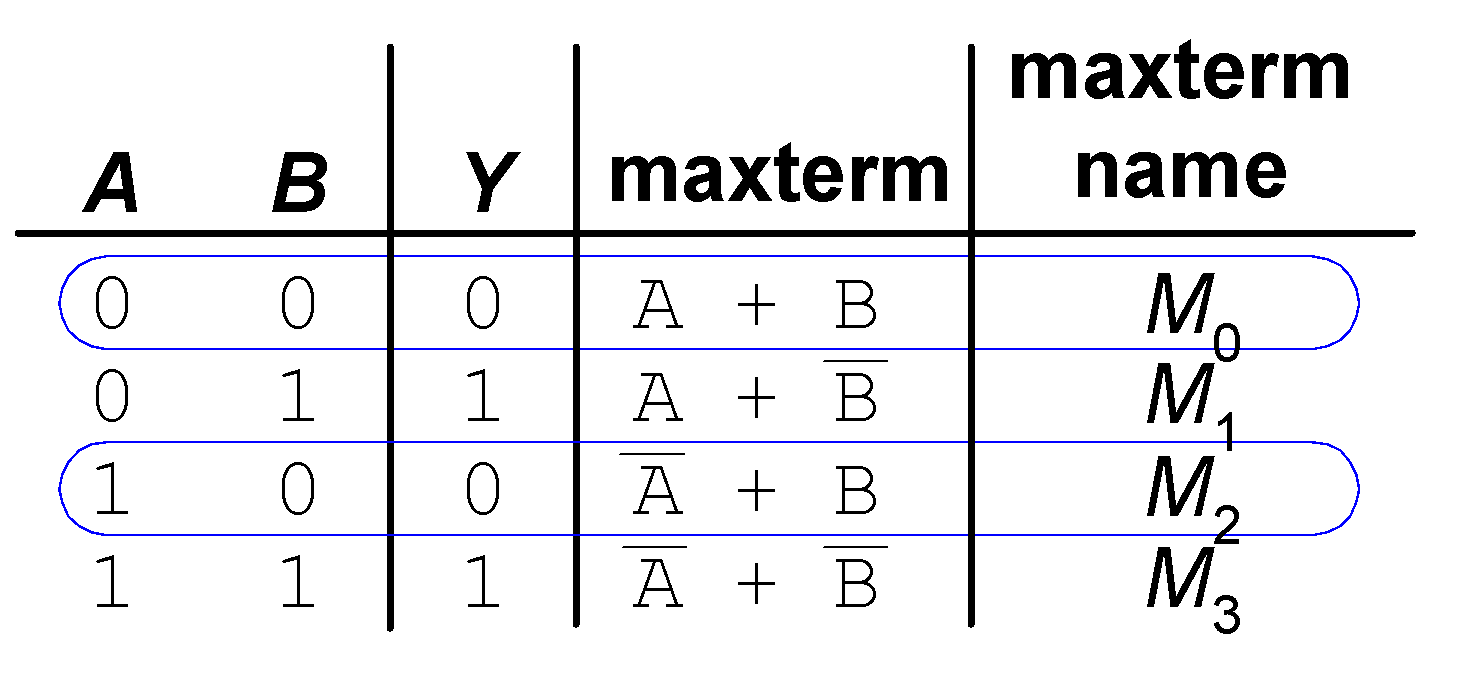
**Ecuaciones booleanas.**

* Variable negada/complementada: A’, ¬A, A.
* Variable literal: A, A’, B, B’, C’, C.
* Implicante: producto de literales. ABC, AC, BC.
* Minterm: producto de todas las variables. ABC.
* Maxterm: suma de todas las variables: (A+B+C).
* Suma de productos (SOP):



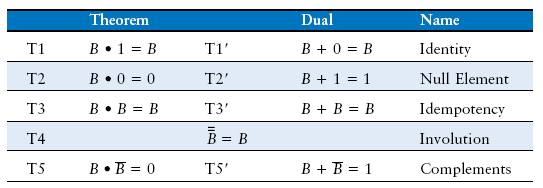
***Y* = F(*A*, *B*) = A'B + AB = Σ(1, 3)**

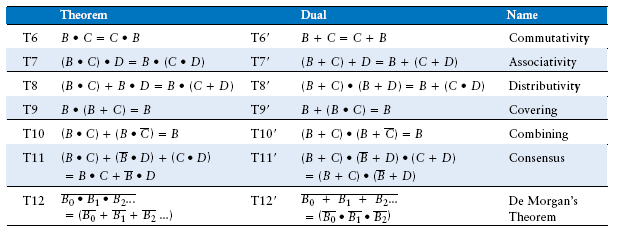
* Productos de suma (POS):

******

**Y = F(A, B) = (A + B)(A’ + B) = Π(0, 2)**

\*En este último los ceros se identifican con variable A,B y los 1 como compuesto.



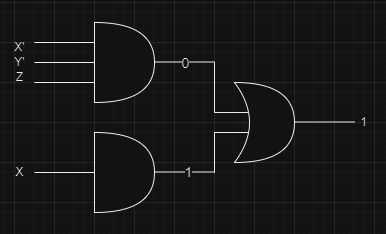


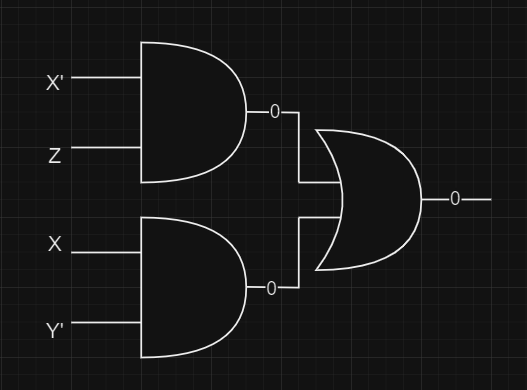
**EJERCICIOS:**

i.

| X | Y | Z | F1 | F2 |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

F1 en mintérminos:

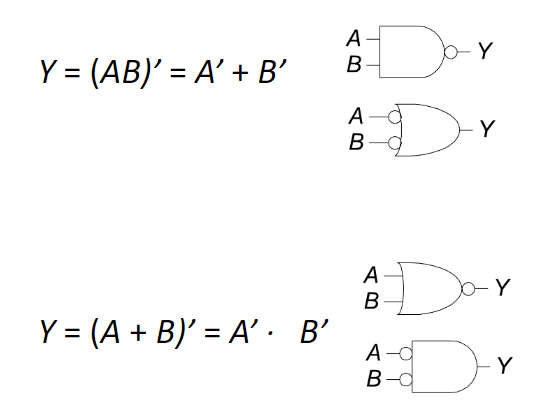
* X’ Y’ Z + X Y’ Z’ + X Y’ Z + X Y Z’ + X Y Z
* X’ Y’ Z + XY’(Z’+Z) + XY(Z’+Z)
* X’ Y’ Z + XY’(0+1) + XY(0+1)
* X’ Y’ Z + XY’(1) + XY(1)
* X’ Y’ Z + XY’ + XY
* X’ Y’ Z + X(Y’+Y)
* X’ Y’ Z + X(0+1)
* X’ Y’ Z + X(1)
* X’ Y’ Z + X

F2 en mintérminos:

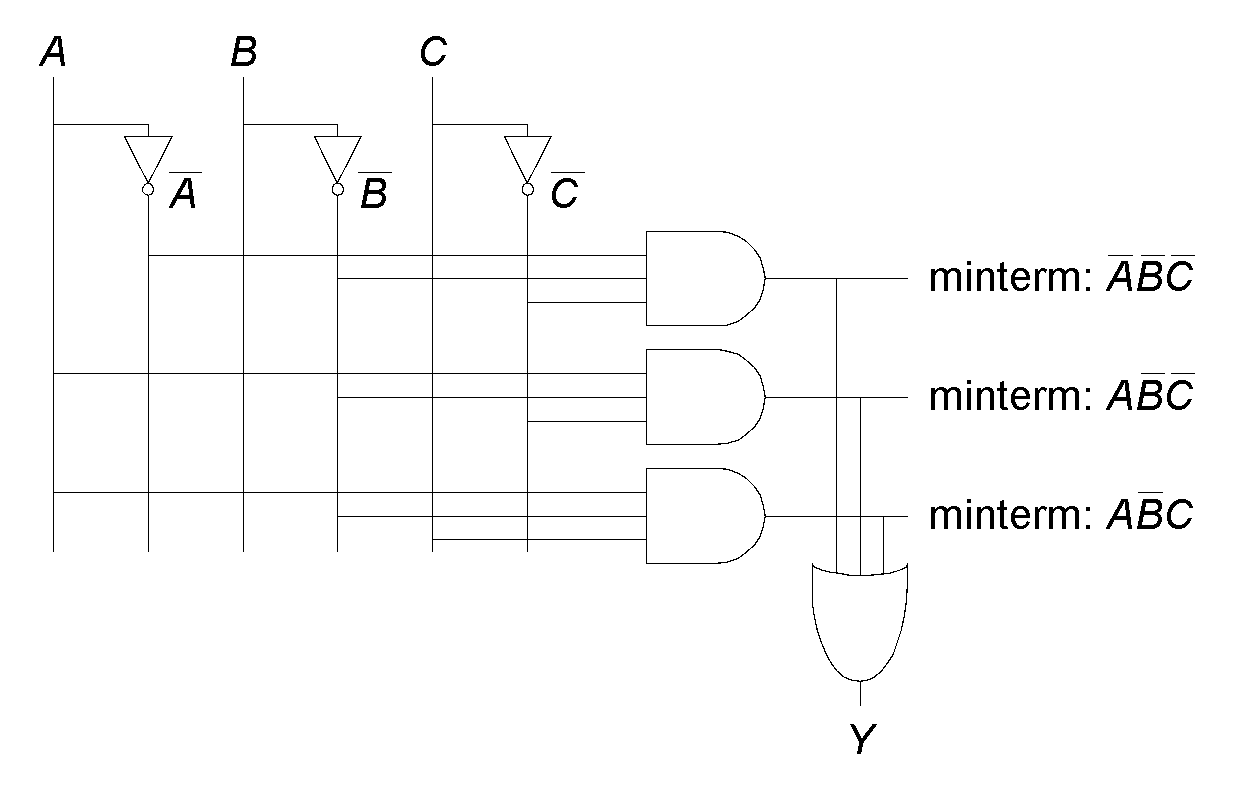
* X’ Y’ Z + X’ Y Z + X Y’ Z’ + X Y’ Z
* X’Z(Y’+Y) + XY’(Z’+Z)
* X’Z(0+1) + XY’(0+1)
* X’Z(1) + XY’(1)
* X’Z + XY’

ii. X (X’+Y) = XX’ + XY = XY

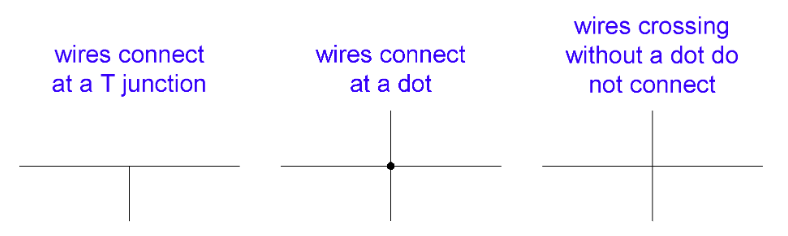
**TEOREMA DE MORGAN:**



* Graficando:
  + *Y* = *A’B’C’* + *AB’C’* + *AB’C*

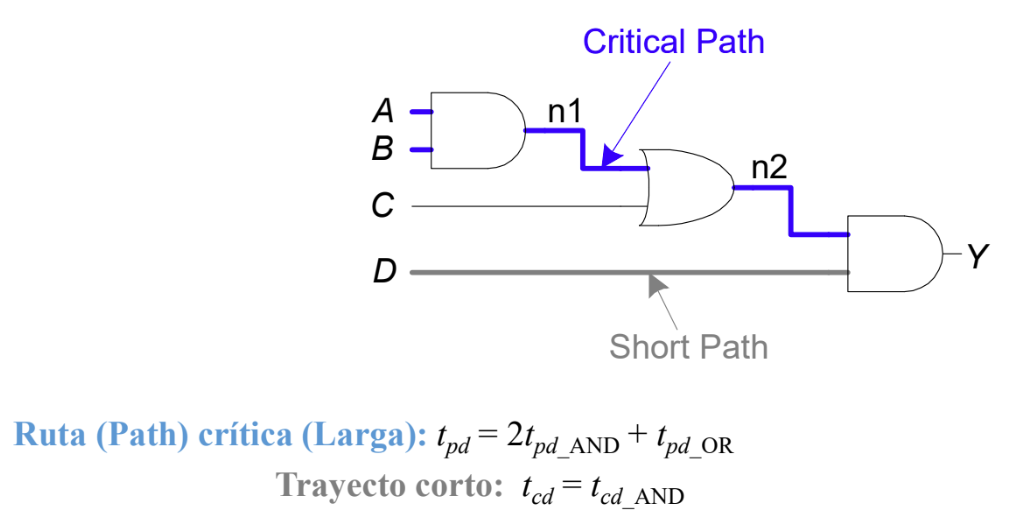


Otras definiciones:

* Los cables siempre se conectan en una unión en T
* Un punto donde se cruzan los cables indica una conexión entre los cables
* Los cables que se cruzan sin un punto no hacen conexión

**Tiempos de demora.**

1. Tiempo de Propagación (tpd): Cuánto demora la ruta crítica de un circuito.
2. Tiempo de Contaminación (tcd): Cuánto demora el trayecto corto.



Falla (Glitches):

Los fallos se solucionan mediante un diseño síncrono, que utiliza un reloj para coordinar las secuencias y evitar glitches.

**Hardware Description Language (HDL).**

Sirve para representar circuitos, escribir un pequeño programa y someterlo a una aplicación que permita una prueba virtual: se puede crear una simulación que interpreta el HDL. problemas técnicos por tiempos de propagación, identificar glitches, ó generar “síntesis” (crear un circuito a partir de una tabla de verdad).

* La simulación lógica es la representación de la estructura y el comportamiento de un sistema lógico digital empleando una computadora.
* El simulador interpreta la descripción en HDL y produce una salida comprensible, digamos un diagrama de temporización, que predice la forma en que se comportará el hardware antes de que se fabrique físicamente.
* La simulación permite detectar errores funcionales de un diseño sin tener que crear el circuito físico.
* Los errores detectados durante la simulación se corrigen modificando los enunciados HDL apropiados. El estímulo que prueba la funcionalidad del diseño se denomina conjunto de pruebas.
* Así, para simular un sistema digital, el diseño se describe primero en HDL y luego se verifica simulando el diseño y verificándolo con un conjunto de pruebas, que también está escrito en HDL.
* La síntesis lógica es el proceso de deducir una lista de componentes y sus interconexiones (lo que se conoce como lista del circuito) a partir del modelo de un sistema digital descrito en HDL.
* La lista del circuito en el nivel de compuertas sirve para fabricar un circuito integrado o para diagramar una tarjeta de circuitos impresos.

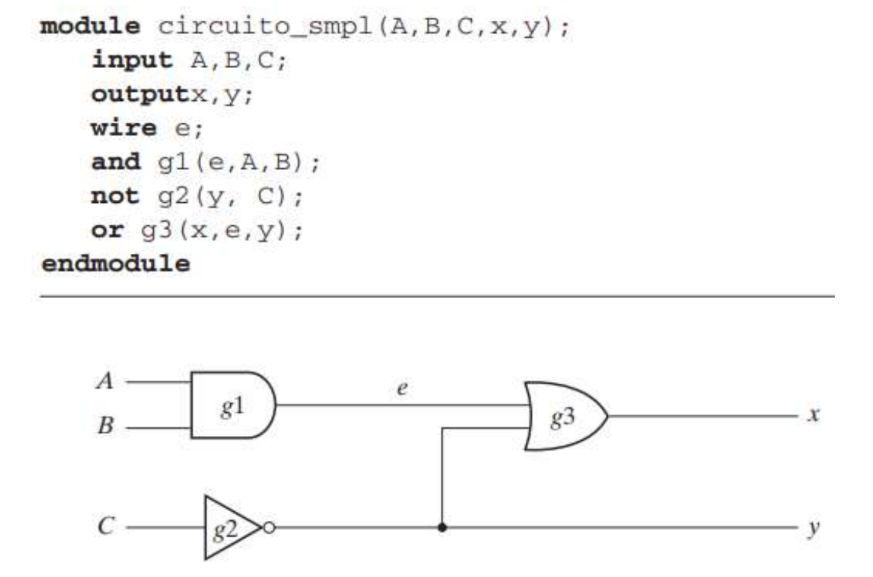
**Verilog HDL** es uno de los dos lenguajes de descripción de hardware (HDL, del inglés Hardware Description Language) más comunes que utilizan los diseñadores. Verilog utiliza cerca de 100 palabras clave: identificadores predefinidos, en minúsculas, que definen las construcciones del lenguaje.

• Como ejemplos de palabras clave podemos citar module, endmodule, input, output, wire, and, or, not, etcétera.

• Todo texto comprendido entre dos diagonales (// ) y el fin de la línea se interpreta como un comentario.

• Se hace caso omiso de los espacios en blanco y se hace distinción entre mayúsculas y minúsculas. El bloque de construcción en Verilog es el módulo.

Ejemplo HDL:



Tiempos de retardo:

* Si no se especifica una escala de tiempo, el simulador utilizará por omisión cierta unidad de tiempo, por lo regular 1 ns (1 ns = 10-9 s).
* En los análisis supondremos la unidad de tiempo por omisión.
* Las compuertas AND, OR y NOT tienen un retardo de 30, 20 y 10 ns, respectivamente.

Las expresiones booleanas se especifican en Verilog HDL con un enunciado de asignación

continuo que consiste en la palabra clave assign seguida de una expresión booleana.

Verilog HDL usa los símbolos:

(&) para AND

| para OR

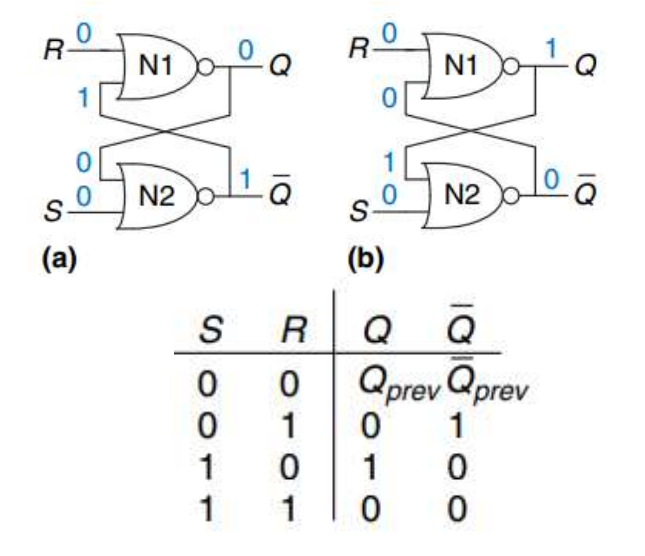
~ para NOT (complemento)

**Probamos edaplayground.com**

**Circuitos secuenciales**

**LATCHes / Flip flops**

El componente fundamental de la memoria es un elemento biestable, un elemento con dos estados estables. La Figura muestra un elemento biestable simple compuesto por un par de inversores conectados en bucle.

****

Para evitar GLITCHES o errores, los circuitos secuenciales se ejecutan bajo un sistema sincrónico, es decir, se les agrega un reloj que regula su funcionamiento mediante “tramos”.

Cada elemento del circuito es un registro o un circuito combinacional.

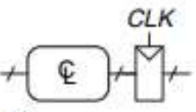
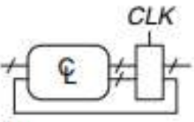
Al menos un elemento del circuito es un registro.

Todos los registros reciben la misma señal de reloj.

Cada ruta cíclica contiene al menos un registro.

El reloj se grafica como “CLK”, habiendo 2 opciones:

a) FLANCO ascendente b) FLANCO positivo

**Máquinas de estados finitos (FSM)**

Obtienen su nombre porque un circuito con k registros puede estar en uno

de un número número finito (2k) de estados únicos.

Una FSM tiene M entradas, N salidas y k bits de Estado. También recibe un

reloj y, opcionalmente, una señal de reinicio.

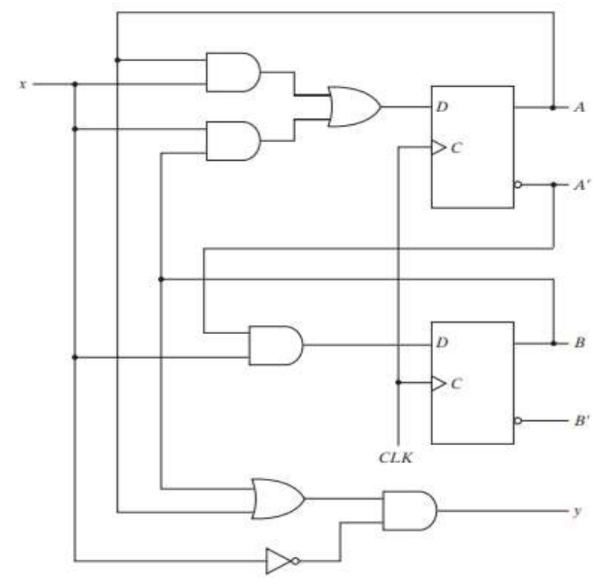
Una FSM consta de dos bloques de lógica combinacional, lógica de siguiente estado y salida lógica, y un registro que almacena el estado.

En cada borde del reloj, el FSM avanza al siguiente estado, que se calculó en función del estado actual y entradas.

Hay dos clases generales de máquinas de estados finitos, caracterizadas por sus especificaciones funcionales. En las máquinas de Moore, las salidas dependen únicamente del estado actual de la máquina. En las máquinas Mealy, las salidas dependen tanto del estado actual como de las entradas actuales. Las máquinas FSM proporcionan una forma sistemática para diseñar circuitos secuenciales sincrónicos, dada una especificación funcional.

**Análisis de un circuito secuencial sincrónico.**

**Circuito: Ecuaciones de entrada y salida:**



* DA = Ax + Bx
* DB = A’x
* y = Ax’ + Bx’

**Tabla de estados:  
**