Unidad 3

3.A. Funciones lógicas.

Constantes y variables Booleanas.

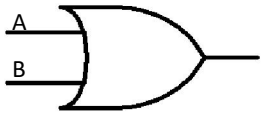
En el álgebra booleana las variables solo pueden tomar dos valores posibles: 0 y 1. Estos se conocen como **niveles lógicos**. Al nivel lógico 0 se lo denomina a también: falso, apagado, abierto, bajo, y al nivel 1: verdadero, encendido, cerrado, alto. Electrónicamente se representan mediante niveles de voltaje, tradicionalmente los "niveles TTL" 0 V y 5 V respectivamente, aunque con la evolución de la tecnología digital es común circuitos de 3,3V, 2,7V, 1.8V. Los niveles tienen un margen de tolerancia, según la tecnología, por ejemplo para TTL, de 0 a 0,8 es un '0' lógico y mayor de 2V es un '1' lógico. Para tecnología CMOS, de 0 a 0,3xVdd es un '0' y mayor de 0,7xVdd es un '1' lógico (siendo **Vdd** el voltaje de alimentación).

Se utiliza el álgebra booleana como medio para representar la entrada y la salida en un circuito lógico. Las entradas suelen ser designadas por letras.

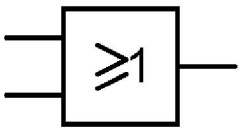
Cualquier circuito lógico puede describirse por completo mediante el uso de las tres operaciones básicas.

Operación OR

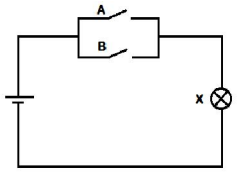
La tabla de verdad muestra la operación OR: "x" es un 1 lógico cuando una o más entrada vale 1. "x" se vuelve 0 únicamente cuando tanto "A" como "B" están en 0. Se puede decir que basta que una de las entradas sea verdadera para que la salida sea verdadera.
La expresión booleana para la operación OR es: $\boldsymbol{x = A + B}$



Símbolo IEEE



Símbolo IEC

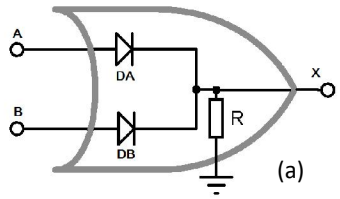


Lógica de llaves

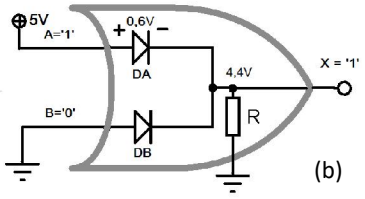
B	A	X= A+B
0	0	0
0	1	1
1	0	1
1	1	1

Tabla de verdad

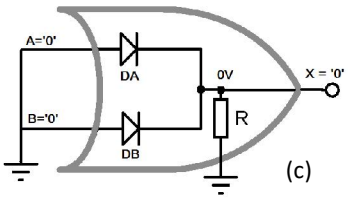
La implementación electrónica más sencilla de una compuerta OR es con diodos, como muestra la figura que siguiente (a). Basta con que a una de las entradas se le aplique un '1' (5 volts) para que en la salida haya un '1' (4,4 volts). La resistencia en la salida se denomina *pull down* ya que "tira hacia abajo" el potencial. Suele ser de entre 1000 y 10.000 ohms.



(a)



(b)



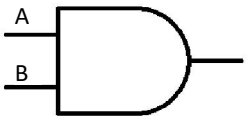
(c)

Nota: en la tabla de verdad, las combinaciones de las posibles entradas **deben seguir un orden determinado** ya que el patrón de salida es la identidad de la función (por ejemplo en la OR de 2 entradas, el patrón de salida es 0111). Las entradas deben agregarse de derecha a izquierda, quedando el mayor subíndice o última letra en la primera columna a la izquierda. Las entradas se nombran letras, y cuando son varias del mismo tipo se las denomina con la misma letra y subíndice de 0 a n-1 para n bits, por ejemplo para **3** bits $\mathbf{A_2, A_1, A_0}$. Esto se comprenderá mejor en próximos circuitos. Se debe tener en cuenta que la cantidad de combinaciones posibles es 2^E siendo E la cantidad de entradas. Para completar la tabla, la lógica es la siguiente: La columna de la primera entrada se completa alternando 0 y 1. La de la segunda entrada se completa alternando a la mitad de la frecuencia, es decir 0, 0, 1, 1 etc. La columna de la tercera entrada a la mitad de la frecuencia anterior, es decir 0, 0, 0, 0, 1, 1, 1, 1. Y así sucesivamente.

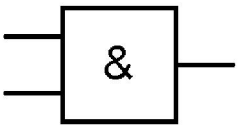
Operación AND

La tabla de verdad muestra la operación AND. "x" es un 1 lógico cuando tanto "A" como "B" se encuentran en nivel 1. Para cualquier caso en donde "A" o "B" valgan 0 entonces "x" valdrá 0. Se puede decir que, para que la salida sea verdadera, todas las entradas deben ser verdaderas. O, que basta que una de las entradas sea falsa para que la salida sea falsa.

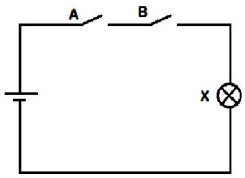
La expresión booleana para la operación AND es: $\boldsymbol{x = A \,.\, B}$



Símbolo IEEE



Símbolo IEC

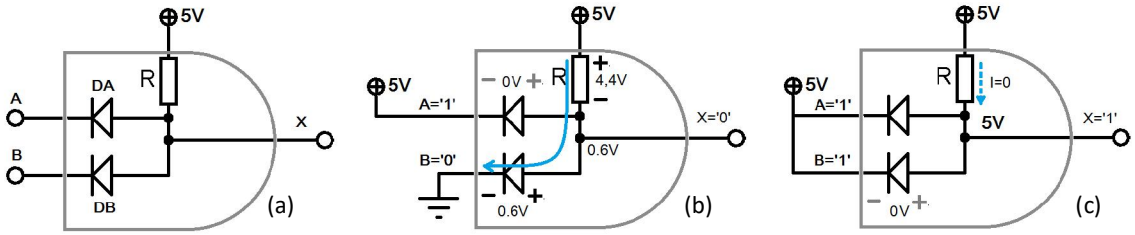


Lógica de llaves

B	A	X = A . B
0	0	0
0	1	0
1	0	0
1	1	1

Tabla de verdad

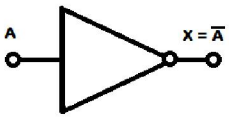
La implementación electrónica más sencilla de una compuerta AND es con diodos, como la mostrada en la figura siguiente (a). Cuando cualquiera de las entradas está en '0', es decir conectada a 0 volts, el diodo correspondiente, a través de R, estará polarizado en directo. Así en **(b)**, con A=1 y B=0, DB queda polarizado en directo y circula una corriente desde los 5V de alimentación, a través de R y DB. Hay entonces una caída de tensión de unos 4,4 volts en R, y el ánodo de DB queda a unos 0,6 volts. En este caso el diodo DA queda polarizado en inverso – impide que '1' de la entrada A pase a la salida. Situación similar se daría con A=0 y B=1.



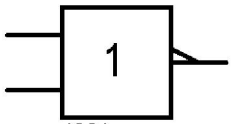
Cuando ambas entradas están en '1', es decir en 5 volts, **caso (c)**, ninguno de los dos diodos conduce y no hay circulación de corriente por R, por lo que los 5 volts de alimentación aparecerán en la salida. Es importante señalar que aplicar 0V a una entrada, es conectarla a 0V, no dejarla abierta. La resistencia R de la salida se denomina *pull up* ya que "tira hacia arriba" el potencial.

Operación NOT

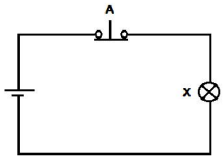
La operación NOT se realiza sobre una sola variable: "x" será el inverso, el complemento o el valor opuesto de "A". **X= A*. Se deonimina "A negado"** (usaremos una línea sobre la letra, o un asterisco como superíndice)



Símbolo IEEE



Símbolo IEC

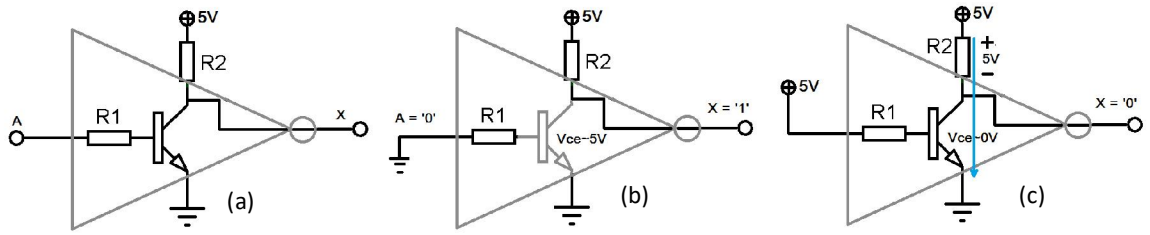


Lógica de llaves

A	X = $\bar{A}$
0	1
1	0

Tabla de verdad

En la compuerta NOT también es necesaria una fuente de alimentación. Cuando se aplican 5V en la entrada, es decir un 1, el transistor se satura y actúa como llave cerrada por lo que la tensión se descarga por la resistencia a masa. Esto se observa en el camino azul de la imagen. Al aplicar 0V, el transistor actúa como llave abierta y la tensión llega a la salida. Esto se observa en el camino rojo de la imagen.

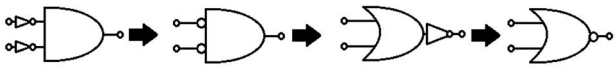


Propiedades de las funciones lógicas.

En circuitos combinacionales, cuando dos tablas de verdad (con las combinaciones de las entradas construidas en el mismo orden) son idénticas, es decir, poseen salidas idénticas, las funciones son **equivalentes** aunque estén implementadas por compuertas distintas. Veremos las principales propiedades de las funciones que nos permitirán operar y simplificar para obtener circuitos más simples o prácticos.

1. $A + 0 = A$ el '0' es el elemento neutro de la suma lógica
2. $A + 1 = 1$ el '1' es el elemento absorbente de la suma lógica
3. $A \,.\, 0 = 0$ el '0' es el elemento absorbente de producto lógico
4. $A \,.\, 1 = A$ el '1' es el elemento neutro del producto lógico
5. $A + \bar{A} = 1$ la suma lógica de una variable y su negación da por resultado '1'
6. $A \,.\, \bar{A} = 0$ el producto lógico de una variable y su negación da por resultado '0'
7. $\overline{(A \,.\, B)} = \bar{A} + \bar{B}$ **1ra Ley de De Morgan**
8. $\overline{(A + B)} = \bar{A} \,.\, \bar{B}$ **2da Ley de De Morgan**

Estas dos últimas propiedades se utilizan para sustitución de compuertas. Por ejemplo, en el caso de la propiedad 8 sería:



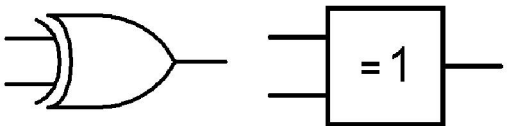
Esta compuerta es la compuerta NOR. Es la composición de una compuerta OR con un inversor en la salida y es equivalente a una compuerta AND con un inversor en cada entrada.

Lo mismo ocurre con la propiedad 7: una compuerta NAND y una OR con entradas negadas son equivalentes.

Las funciones OR, AND y NOT son funciones *primitivas*, mientras que la NAND y la NOR son funciones derivadas.

Operación XOR (OR exclusiva)

Una de las compuertas derivadas más importantes es la OR EXCLUSIVA o XOR. La salida es la suma exclusiva de las entradas. La salida es cero cuando la cantidad de 1 es par.
La expresión booleana para la operación XOR es: $x = A \oplus B$



Símbolo IEEE Símbolo IEC

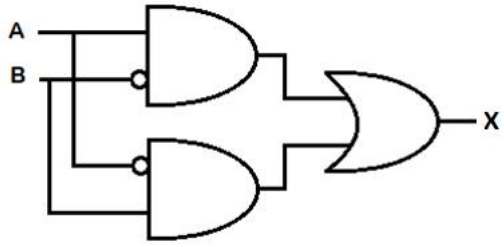
B	A	X = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Tabla de verdad

La OR EXCLUSIVA es una compuerta que puede armarse con compuertas AND y compuertas OR con lo que se llama SUMA DE PRODUCTOS.

La función estará formada por dos subfunciones:

$$X = A \,.\, \bar{B} + \bar{A} \,.\, B = A \oplus B$$



Obtengo una compuerta OR que conecta las salidas de dos compuertas AND con sus respectivos inversores.

3.B. Circuitos combinacionales

Las compuertas se pueden interconectar para que ante combinaciones de variables lógicas de entrada, se activen salidas siguiendo una lógica de funcionamiento deseada. Vemos a continuación algunas aplicaciones:

Síntesis de una función lógica a partir de la Tabla de Verdad:

Uno de los procedimientos para obtener la función a partir de la tabla de verdad es el siguiente: Se marcan aquellas combinaciones de entrada para las que la salida es **'1'**. Por cada combinación de entradas obtendré un término en la suma. Dentro de cada término, se colocan negadas aquellas entradas que valen cero y sin negar las que valen uno.

Ejemplo: Si queremos que la salida S se active en las combinaciones indicadas (1, 4, 5 y 7), se realiza una suma (OR) de 4 términos, que serán funciones AND de las entradas A, B y C.

	C	B	A	S
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

La función que se activa en la combinación 1 es $A.\bar{B}.\bar{C} = 1$ (es '1' cuando A=1, B=0 y C=0)
Para la combinación 4 es:
$\bar{A}.\bar{B}.C = 1$ (es '1' cuando A=0, B=0 y C=1)
Para la combinación 5 es:
$A.\bar{B}.C = 1$ (es '1' cuando A=1, B=0 y C=1)
Para la combinación 7 es:
$A.B.C = 1$ (es '1' cuando A=1, B=1 y C=1)
Luego para que S se active en 1,4, 5 y 7:
$S = A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.\bar{B}.C + A.B.C$

Si se observa, la función XOR se puede obtener de este modo.

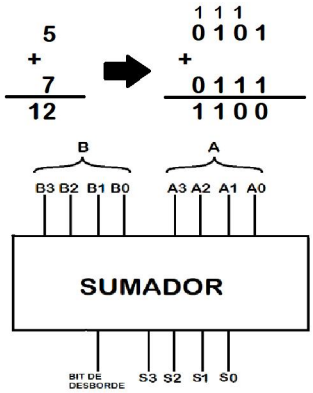
Generador de paridad.

La codificación de la unidad de información en las computadoras es el byte (1byte = 8 bits). Cuando la información se transmite en serie, se transmite de una computadora a otra bit por bit los paquetes de bytes. Una de las formas de verificar que la información llego correctamente es agregarle un bit extra al paquete de 8 bits, llamado bit de paridad. Entonces con este bit uno puede asegurarse de que la cantidad de '1' del byte sea por ejemplo, siempre par. Esta es una convención entre el equipo que recibe y el que transmite. Si, por ejemplo, por ruido eléctrico **uno** de los valores cambia durante la transmisión, la paridad con la que fue generado se va a romper, y el receptor podrá detectar el error.

El generador de paridad puede ser E (*even* - par), O (*odd* . impar) o N (sin paridad).

Sumador.

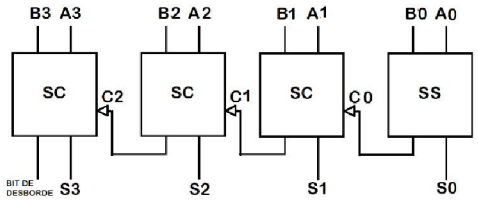
La idea central del sumador es, como su nombre lo indica, realizar la suma aritmética de números representados en binario.



La mecánica es igual que para decimales solo que en este caso el máximo número por columna es 1 y el excedente se acarrea.

Para la implementación del circuito tendremos dos conjuntos de entradas: Uno A y uno B para formar cada uno de los operandos. Si cada conjunto está formado por 4 bits podremos ingresar los números del 0 al 15. La salida tendrá un bit más, llamado bit de desborde, ya que el máximo resultado de la suma es 30.

Cada uno de los dígitos del resultado es la función suma de dos de los bits de las entradas. En la primera operación será la suma de dos bit pero a partir de la segunda deben sumarse 3 bit, ya que se agrega el acarreo de la operación anterior. Por lo que cada operación tendrá dos salidas, una de las cuales es el acarreo que estará conectado como entrada a la operación siguiente.



Estas operaciones se llevan a cabo en subsistemas: el sumador completo SC y el semisumador SS. El SS es un combinacional con 2 entradas y dos salidas y el SC posee 3 entradas y 2 salidas.

Implementación del semisumador

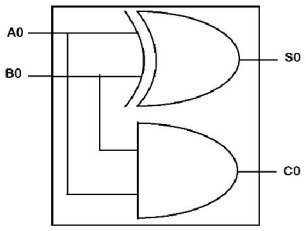
La tabla de verdad correspondiente es la siguiente.

B0	A0	S0	C0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Entonces:

$$S0 = \overline{B0}.A0 + B0.\overline{A0}$$

$$C0 = B0.A0$$



Podemos reconocer la función suma S0 como una OR Exclusiva y la función de acarreo (*carry*) C0 como una AND.

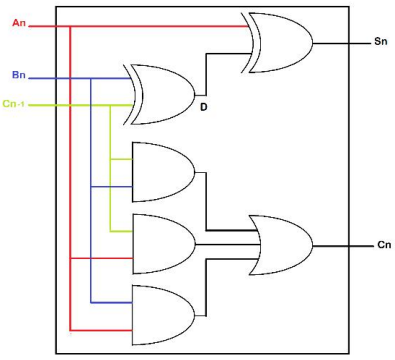
Implementación del sumador completo.

El sumador completo es un bloque con dos funciones de 3 entradas cada una.

La tabla de verdad correspondiente es la siguiente.

Cn-1	Bn	An	Sn	Cn
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

En principio, según la tabla, deberíamos usar 4 compuertas AND para cada salida S y C pero existen técnicas de minimización basadas en las propiedades de las funciones que permiten utilizar menos compuertas.

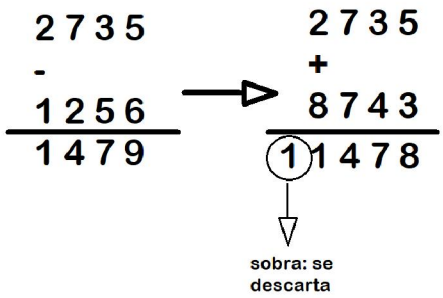


Nota: El circuito sumador de 4 bits visto se denomina *sumador de acarreo* y es una implementación sencilla pero lenta, ya que el resultado es válido luego de que se han propagado todos los acarreos desde el bit menos significativo al más significativo. En este caso son 4 bits, pero cuando los operandos son de muchos más bits (Ej 16, 32) se vuelve muy lento. Hay otras implementaciones más eficientes partiendo de los bloques sumadores completos, e incluso se puede utilizar una LUT (*look up table* o tabla de búsqueda) que es una memoria permanente con resultados precalculados.

Restador.

La resta, electrónicamente, sería más compleja que la suma. Si analizamos cómo resolvemos una resta, cuando "el de arriba" es más chico que "el de abajo", el primero "le pide uno" al de "al lado" y éste último debe "saber" que se le restó uno.

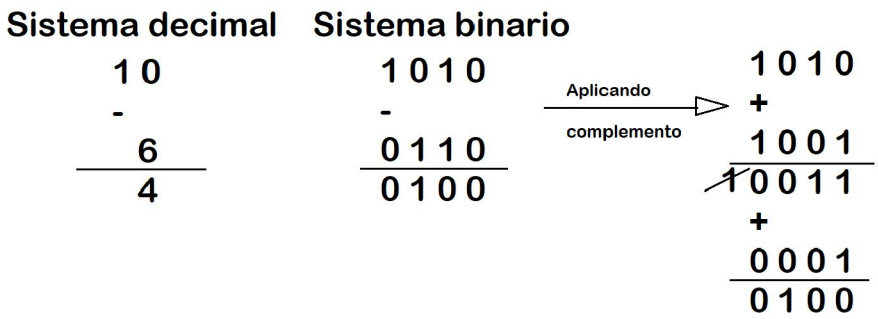
Para resolver esta situación se aplica una técnica que se llama complemento a la base.



El complemento a la base menos 1, en sistema decimal siendo la base 10, es el complemento a 9. Es lo que le falta a cada digito del número que resto para llegar a 9. Se reemplaza cada digito por su complemento a 9 y, en lugar de restarlo, se lo suma.

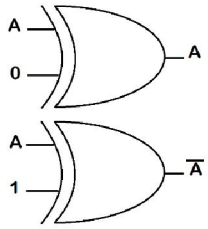
El resultado obtenido es muy similar pero no igual. Siempre va a faltar una unidad y esto se corrige sencillamente sumando uno. Además, el primer digito debe truncarse.

En números binarios esto es muy sencillo ya que el complemento a 1 de cualquier número es directamente el número inverso. Obsérvese el ejemplo.

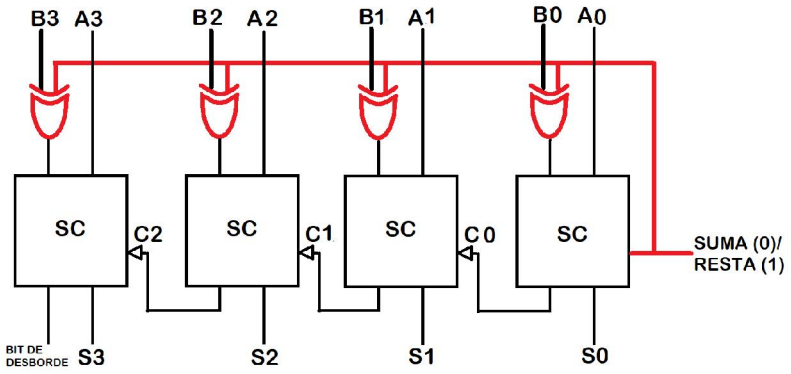


Implementación del circuito sumador/restador

Sencillamente se puede resolver agregando inversores en las entradas del número binario B y sumándole uno al resultado o directamente al primer semisumador se le asigna un uno volviéndolo un sumador completo.



Al mismo circuito se le puede agregar un inversor a B si se quiere restar y quitarlo si se quiere sumar. Se puede aprovechar la propiedad de la compuerta OR exclusiva de que, si una de las entradas es cero, en la salida tendré el valor de la otra variable, pero si es uno, tendré el inverso de la otra variable.



Comparador.

El circuito comparador revela cuál es la condición entre dos números (mayor, igual o menor). A continuación, se muestra su tabla de verdad para 2 bits.

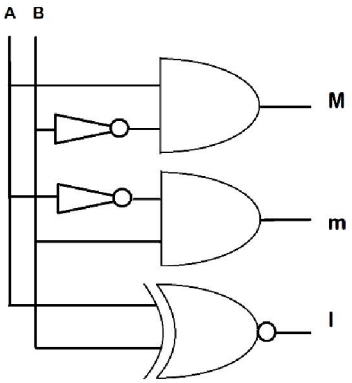
B	A	M	m	i
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Con el método de suma de productos, las salidas son:

$$M = A \,.\, \bar{B}$$

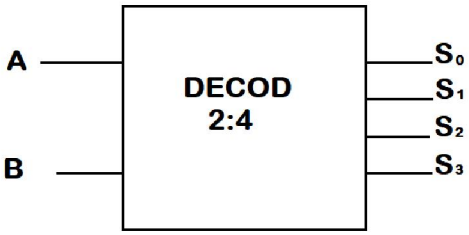
$$i = \bar{A}.\bar{B} + A \,.\, B$$

$$m = \bar{A} \,.\, B$$



Decodificador

Es un sistema que tiene n entradas (A, B, …) y 2^n salidas.
Esquematizamos aquí un Decodificador 2:4, con dos entradas (A y B) y 2^2 salidas (S0 a S3)



Bin	B	A	S0	S1	S2	S3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

Las combinaciones de entradas corresponden a un número binario y cada salida se activa cuando su número binario correspondiente es ingresado en la entrada, en tanto que las demás salidas permanecen desactivadas. Funciona como un selector de activación. Una de sus principales aplicaciones es en los circuitos de memoria.

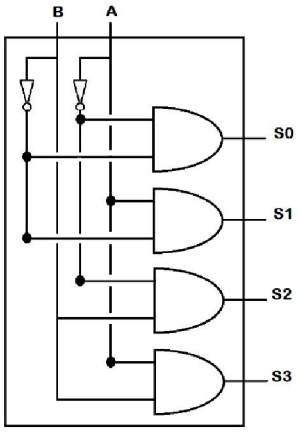
Para obtener la expresión algebraica y circuito correspondiente a esta tabla de verdad observamos que la combinación de salidas de la S3 es idéntica a la de la función AND, variando solamente que las entradas A y B estén negadas o sin negar de forma de conseguir la activación de la Sx correspondiente.

$$S_0 = \bar{A} \,.\, \bar{B}$$

$$S_1 = A.\bar{B}$$

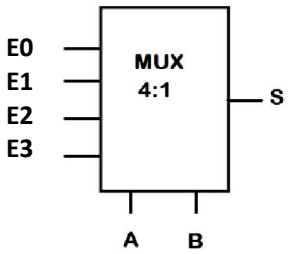
$$S_2 = \bar{A} \,.\, B$$

$$S_3 = A.B$$



Pregunta: ¿Cómo será el decodificador de 3:8?

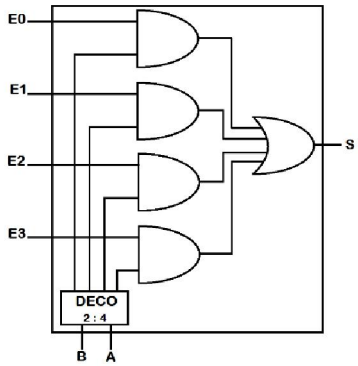
Multiplexor



B	A	S
0	0	E0
0	1	E1
1	0	E2
1	1	E3

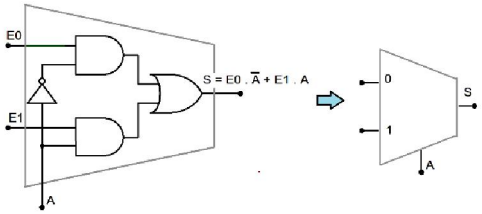
Es un circuito que permite que la salida S adopte el valor lógico de una entre 2^n entradas Ei, en función de los bits de selección A y B. Funciona como una llave selectora que permite elegir cuál de las entradas estará presentes en la salida dependiendo de la combinación de A y B.

Para confeccionar la tabla de verdad se debe tener en cuenta que, en este caso, las entradas son 6: E0, E1, E2, E3, A y B. es decir la cantidad de combinaciones posibles a la salida son 2^6=64. Confeccionar la tabla de verdad completa sería muy extenso y se perdería de vista la función del multiplexor. Por esto se usa una tabla de verdad condensada o reducida.



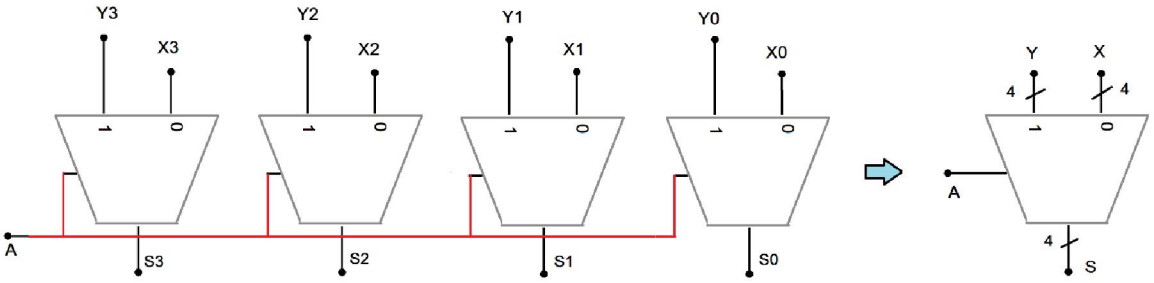
Para el selector usamos un decodificador.

Si analizamos el comportamiento de este circuito se observa que el decodificador activara solo la salida indicada por A y B, es decir, solo esa salida valdrá uno y las demás valdrán cero. Por las propiedades enunciadas anteriormente, sabemos que aquellas compuertas AND que tengan una entrada cero, sin importar el valor de la otra entrada, la salida será cero y la que tenga un 1 en una entrada, la salida reflejará el valor de la otra entrada. De esta manera en la salida se reflejara el valor de la entrada seleccionada con el decodificador.



El multiplexor más sencillo es el 2:1, que permite seleccionar una de dos entradas E1, E0 en función de un único bit de selección A.

Si agrupamos un conjunto de N multiplexores 2:1, podemos realizar un multiplexor que nos permita escoger entre 2 grupos de bits. En el ejemplo siguiente tenemos dos números X e Y, de 4 bits, y mediante la entrada selectora A podemos seleccionar: Cuando A = 0 S = X, y cuando A = 1 S = Y, siendo X, Y y S números binarios de 4 bits. También los podemos expresar como X[3..0], Y[3..0] y S[3..0]



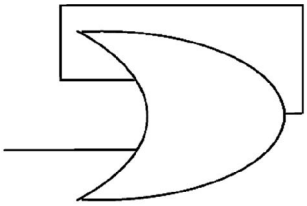
3.C. Circuitos secuenciales.

Los circuitos vistos hasta ahora tienen la característica de que, siempre que ingrese la misma **combinación** en las entradas, en la salida se presentará el mismo valor. Por eso se denominan **combinacionales**.

En el caso de los circuitos **secuenciales,** mediante el uso de realimentaciones desde una etapa hacia otra etapa anterior, se logra que la salida dependa no sólo de los datos ingresados sino también del estado previo del sistema. Esto permite almacenar estados que van cambiando con el tiempo.

Los biestables son dispositivos capaces de almacenar un estado que puede ser alto o bajo.

Biestable SR básico.

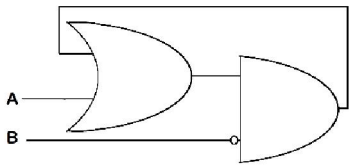


Se comenzará analizando un circuito muy simple: una compuerta OR con su salida realimentada hacia una de las entradas.

Momento 1: Suponiendo que la salida está en cero, la entrada superior toma el mismo valor. Al ingresar un cero por la entrada inferior, la salida continuará siendo cero y el estado se mantiene.

Momento 2: La salida y la entrada superior, como se dijo, están en cero. Se ingresa por la entrada inferior un uno, este se refleja en la salida y es realimentado a la entrada superior.

Momento 3: Al ingresar un cero por la entrada inferior nuevamente, como la entrada superior está dominada por la salida, la salida continúa valiendo uno. La compuerta "ya tiene historia".



Al circuito de salida se le intercala una llave que abra el circuito para que deje de enviar el uno y envíe un cero para poder volver a utilizar la compuerta. Para esto se utiliza una compuerta AND con la entrada inferior negada

De esta forma, si B=0, en la entrada de la AND se tendrá un 1 y lo que haya en la otra entrada va a pasar. Es decir que cuando B=0, el circuito es equivalente al anterior. Pero cuando B=1, por el inversor, la entrada valdrá cero, abriendo el circuito ya que, sin importar el valor que ingrese a la OR, en la salida siempre habrá un cero.

El estado normal de B será cero, es decir que estará apagado, dejando pasar la realimentación y al ingresar un uno por la entrada A este quedara memorizado. Cuando B=1, se cortará la realimentación. Lo primero se denomina **enclavamiento** y lo segundo **desenclavamiento**.

Los ceros y los unos se ingresan con pulsadores normal abierto (NA, como el de la imagen) o pulsadores normal cerrado (NC). El primero (NA) al pulsarlo **conecta** y el segundo (NC) **desconecta**. Una llave también se denomina pulsador con retención y tiene dos estados estables posibles. Es importante recordar que ingresar un cero no es dejar la entrada abierta ya que con esto lo que se logra es que ingrese ruido y no está garantizado ninguno de los dos valores. Para ingresar un cero se debe conectar con una resistencia a masa. Estas resistencias no son necesarias si la etapa previa es otra compuerta digital, que asegura los niveles de tensión de '0' y '1'.

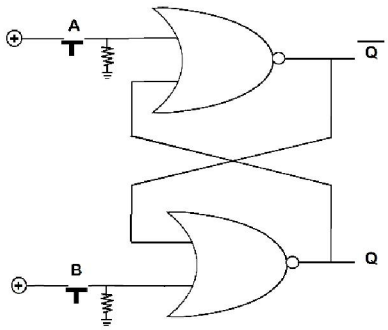
Este circuito es equivalente a dos compuertas NOR realimentadas mutuamente. Se demuestra de la siguiente forma. En la figura anterior:

1. Se colocan dos inversores seguidos entre la salida de la OR y la entrada de la AND. En principio el efecto es neutro.

2. Uno de los inversores se une a la compuerta OR formando una compuerta NOR.

3. Los dos inversores en la entrada de la AND, por la Ley de Morgan equivalen a una NOR.

4. Se observa que la salida de la NOR superior está conectada a la entrada de la inferior y viceversa.



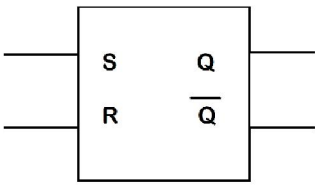
Al pulsar A, ingresa un 1 a la NOR superior, por lo que su salida se vuelve cero sin importar lo que haya en la otra entrada (al revés que con la OR). Si no se está pulsando B, entra un cero y por la otra entrada de retroalimenta el cero por lo que la salida de la NOR inferior será 1.

Este 1 se retroalimenta a la entrada de la NOR superior y mantiene la salida en cero aunque se suelte el pulsador.

Es decir: presionando A se logra que se encienda la salida de la NOR inferior (**Q**) y que la salida de la NOR superior (**!Q**) sea cero.

Si luego se acciona el pulsador B, de forma simétrica, la salida de la NOR inferior se vuelve cero y la NOR superior se vuelve uno.

Este biestable se denomina **SET-RESET** o SR o **SR Asíncrono**.



Es importante observar que las salidas están "cruzadas". Al presionar A o el SET se activa la salida inferior y al presionar B o el RESET se activa la superior. En el esquema las salidas se representan invertidas para que haya una correspondencia lógica. Lo que puede hacerse en el circuito es "cruzar los cables" por dentro.

En la tabla de verdad solo se representa la salida Q ya que se asume que la otra simplemente tomara el valor opuesto.

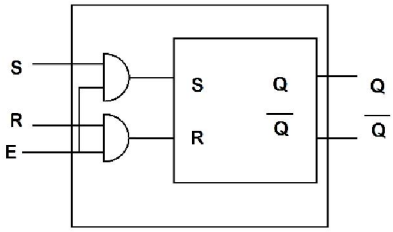
S	R	Q
0	0	Q t-1
0	1	0
1	0	1
1	1	X

Q t-1 es el estado anterior y X significa que hay una indeterminación: no puede saberse con seguridad que valor tomará la salida.

Este biestable permite guardar información: 1 bit. Esta es la unidad mínima de la memoria RAM. Al agrupar 8 se forma un byte. A diferencia de la memoria ROM, esta información se puede escribir y borrar. Además, es el núcleo de todos los biestables.

Biestable SR activado por nivel.

También se denomina SR Síncrono.



En este caso se tienen 2 salidas pero 3 entradas.

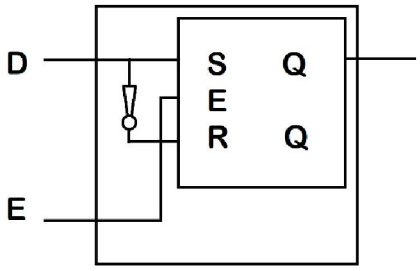
Si E=1, las AND dejan pasar los valores de S y R pero si E=0, sin importar los valores de S y R, no puede cambiarse la salida.

La entrada ENABLE se utiliza para sincronizar el dispositivo al momento de operarlo.

En la tabla de verdad se encuentran X en las entradas. Esto significa que no interesa el valor de la entrada.

E	**S**	**R**	**Q**
0	X	X	Q_{T-1}
1	0	0	Q_{T-1}
1	0	1	0
1	1	0	1
1	1	1	X

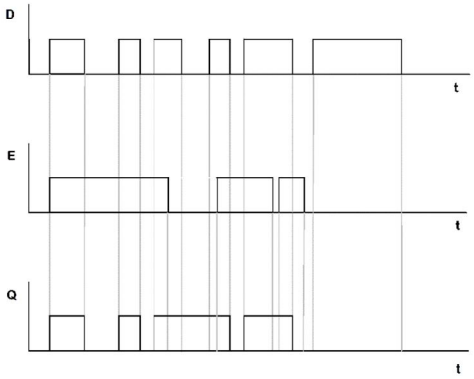
Biestable D activado por nivel.



Cuando E=0, mantiene el dato memorizado y no toma el dato de entrada. Cuando E=1, Q toma el valor de la entrada D, es decir: se puede escribir.

Una gran ventaja es que se anula la posibilidad de que exista la indeterminación de tener dos 1 en las entradas.

E	Q
0	Q t-1
1	D

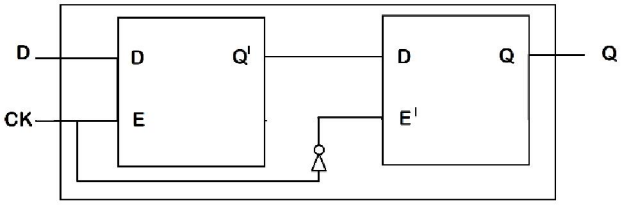


Se puede decir que D_E es un circuito **transparente** si E=1. Es decir, que, si D cambia, Q cambia de la misma manera. Pero, cuando E=0, el ultimo valor de D queda memorizado en Q.

Biestable D activado por flanco (*master-slave*)

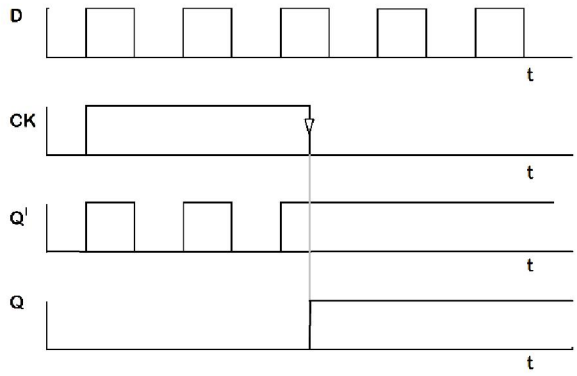
Este biestable sirve para aquellas aplicaciones en que se necesita que el dato se almacene en un instante único denominado flanco activo. Puede ser flanco de subida o flanco de bajada.

El biestable D_{MS} tiene dos entradas: el Clock y la entrada de datos.



Internamente está constituido por dos biestables D activados por nivel. Las habilitaciones de los biestables (E y E') están complementadas mediante el inversor. Cuando la del primero biestable vale 1, dejando pasar el dato, la del segundo vale cero y no permite que sea escrito permaneciendo en un estado de retención del dato anterior.

Cuando el clock pase a valer 0, será al revés. El primer biestable estará bloqueado y no se dejará escribir, pero el segundo sí.

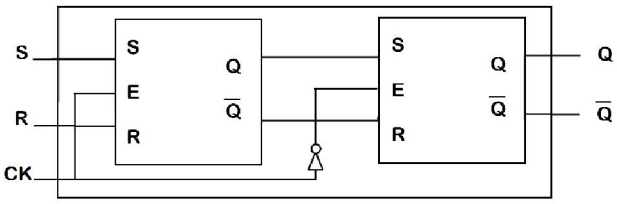


El objetivo es que mientras el clock valga 1, aunque la entrada D esté cambiando, la salida permanezca constante. En el momento en que el clock pase a valer cero, la salida Q' quedará con el ultimo valor de D retenido y el segundo enable E' habilitará que Q tome éste último valor.

CK	Q
↓	D
X	Q_{T-1}

En resumen: la transferencia de datos de la entrada a la salida ocurre en el momento en que el clock pasa de valer 1 a 0. Es decir, su activación es por flanco de bajada.
En un segundo momento, aunque el segundo biestable sea transparente, es decir, dejando pasar el dato, el primero estará bloqueado y no importará que varíe D.
De haber un nuevo flanco de subida, es decir que el CK pase a valer 1, el primer biestable volverá a ser transparente pero el segundo estará bloqueado sin que haya transferencia del dato.

Biestable SR *master-slave*



Éste no es un biestable que se utilice, si no que se estudia porque es un intermedio para analizar el biestable J_K.

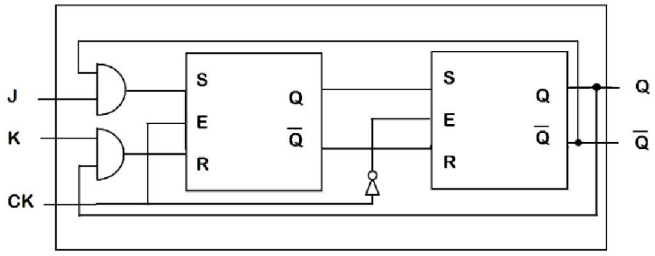
De la misma forma que para el biestable D_{MS}, se combinan dos biestables SR activados por nivel.

El comportamiento es el siguiente, si CK=1, el primer biestable es transparente y el dato que se ingresa en la entrada se transfiere a la salida del primer biestable y a la entrada del segundo. En el momento en que se ingresa un cero en el CK, el primer biestable se bloquea y el último valor que ingresó al mismo, es el que ingresa al segundo.

CK	S	R	Q
↓	0	0	Q_{T-1}
	0	1	0
	1	0	1
	1	1	X
X	X	X	X

En este caso se mantiene el problema de la indeterminación al ingresar dos 1.

Biestable JK master-slave.

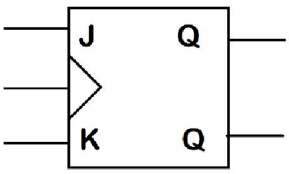


Esta configuración salva la indeterminación del biestable anterior SR_{MS}.

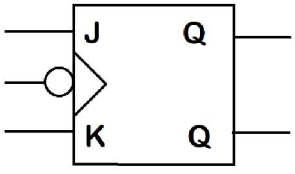
El objetivo es no permitir que ingresen dos unos en simultaneo. Para esto, se busca que, si Q está encendido, permita que se pueda apagar y que, si está apagado, que se pueda prender.

CK	J	K	Q
↓	0	0	Q_{T-1}
	0	1	0
	1	0	1
	1	1	$\overline{Q_{T-1}}$
X	X	X	Q_{T-1}

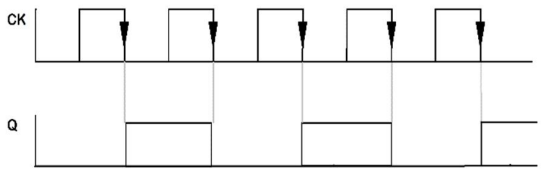
La idea es permitir mediante el uso de la realimentación con compuertas AND que se escriba 01 o 10 pero no ambas en simultáneo.
El comportamiento es el siguiente: si Q está encendido, K debe poder actuar para apagarlo y si está apagado, no es necesario que K actúe.
Cuando Q=0 y Q'=1, K no puede actuar sobre RESET, pero no importa porque ya está reseteado.
Al ingresar J=1 y K=1, en el flanco de bajada solo podrá pasar aquel que este apagado. Si Q=0, pasará J y lo encenderá y si Q=1, pasará K y lo apagará.
Los símbolos son los siguientes:



ACTIVADO POR FLANCO DE SUBIDA



ACTIVADO POR FLANCO DE BAJADA



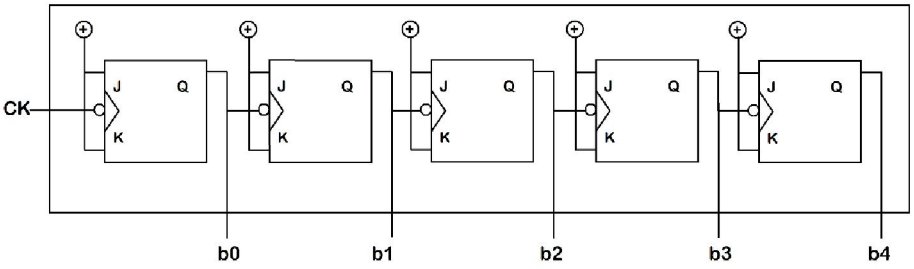
Como se observa en la tabla de verdad, en el flanco de bajada del CK, Q pasa a tener el valor opuesto al que tenía. El efecto obtenido es que a partir de una señal que tiene n pulsos/segundos, obtenemos una señal que tiene n/2 pulsos /segundos. Por lo que una de las aplicaciones de este biestable es como divisor de frecuencia.

Registros.

Los registros son agrupaciones de biestables. Se muestran a continuación los principales registros.

Contador asíncrono.

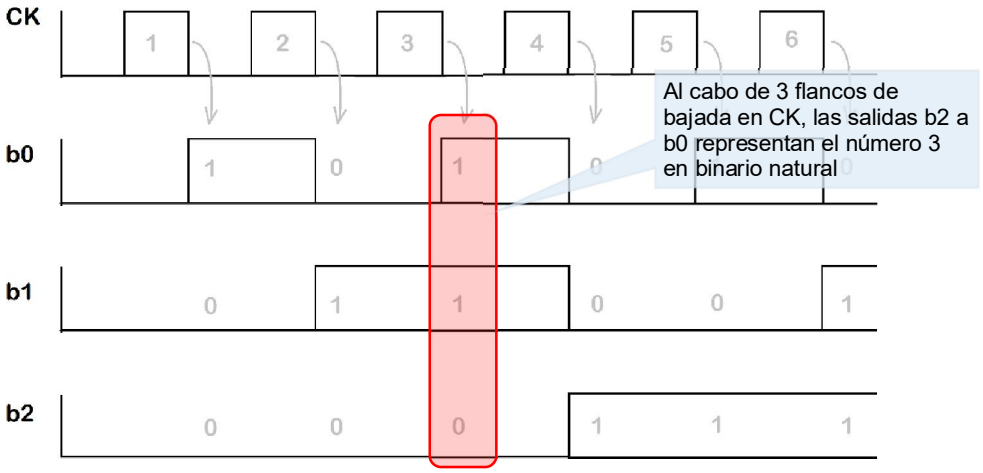
Es un contador binario. Consiste de una entrada donde ingresan pulsos y una serie de salidas denominadas bits y donde la onda de salida de un biestable funciona como el CK del siguiente biestable.



La combinación binaria de la salida depende de la cantidad de pulsos ingresados por el clock.

Se realiza la suposición de que, al comenzar, todos los bits están en 0. Al ingresar un pulso por el CK, el b_0 pasa a valer 1: número 1 en binario. Si se observa la tabla de numeración binaria, b0 cambia con cada flanco de bajada; b1 cada dos flancos de bajada; b2 cada 4 y así sucesivamente. Si se lo interpreta en términos de pulsos, si b0 tiene una frecuencia; b1 tiene una frecuencia que es la mitad de la de b0 y así sucesivamente.

b4	b3	b2	b1	b0	Cantidad de pulsos
0	0	0	0	0	1
0	0	0	0	1	2
0	0	0	1	0	3
0	0	0	1	1	4
0	0	1	0	0	5
0	0	1	0	1	6

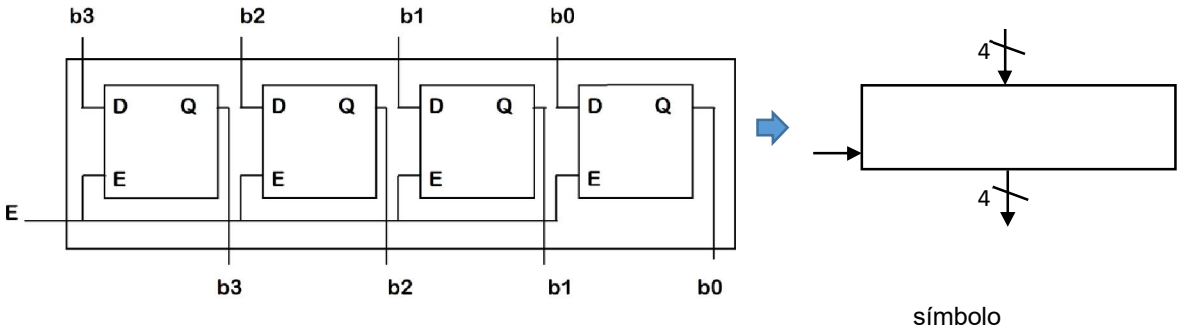


¿Qué ocurre cuando todos los bits coincidan en 1? El contador llega a su valor máximo posible y al siguiente pulso coinciden todos los flancos de bajada, reiniciándose en cero.

¿Se podría lograr que el contador cuente hacia atrás? Esto se logra utilizando JK activados por flanco de subida.

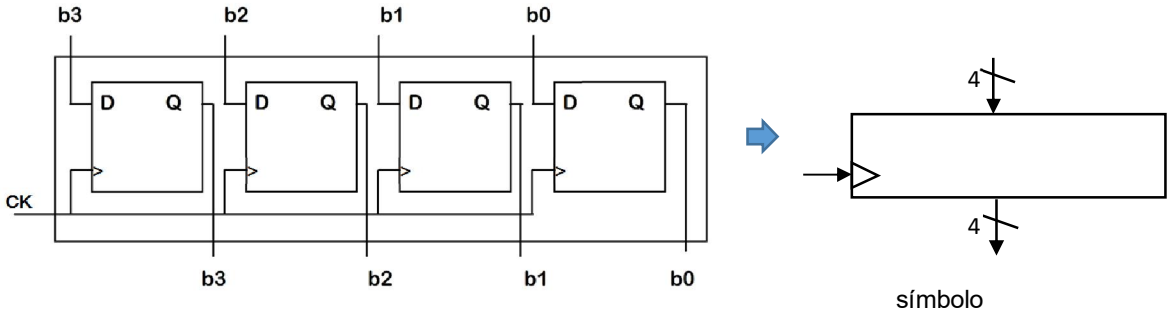
Registro paralelo-paralelo con habilitación por nivel (*latch* transparente).

Este registro consiste en una agrupación de biestables D activados por nivel, con entradas y salidas individuales, pero con una misma entrada de habilitación. De este modo es posible memorizar un grupo de bits (por ejemplo la salida del sumador, de N bits, podría memorizarse en este registro). Con este tipo de registros, un decodificador y algunos elementos auxiliares se construyen las memorias RAM estáticas.



símbolo

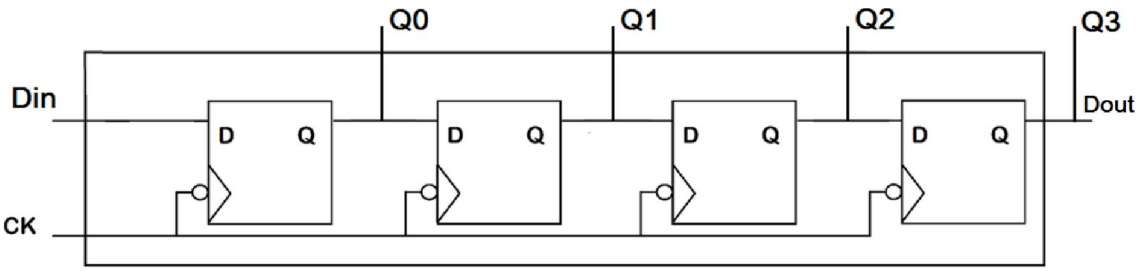
Registro paralelo-paralelo con activación por flanco



símbolo

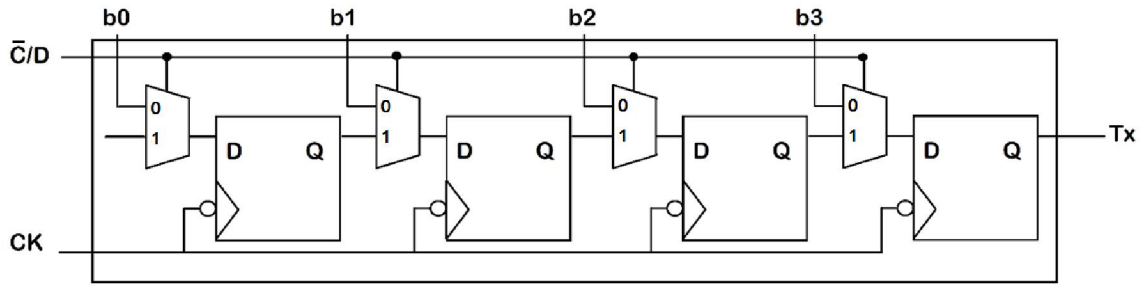
Este registro es muy similar al anterior, pero con biestables D activados por flanco, conectado a una misma activación CK. Este tipo de registros los veremos luego en interior de un microcontrolador.

Registro de desplazamiento serie-paralelo.



En este registro los biestables, **que deben ser de tipo activado por flanco,** se encadenan a través de sus salidas y entradas de datos (Q y D), como se muestra en la figura.
Ante cada flanco activo de CK se desplaza un lugar a la derecha el valor lógico que contengan los biestables. (**Nota:** *en este caso es flanco de bajada, pero podría ser de subida también*). El valor lógico de cada biestable se observa en las salidas Q0, Q1, Q2 y Q3. El primer biestable tiene su entrada D conectada a la entrada Din, por donde recibirá valores lógicos 1 y 0 en secuencia. La salida del último biestable, Q3 o Dout, podría encadenarse a otro circuito similar, para formar así un registro de más bits. Este circuito es la base de los **receptores** de comunicación serie de los sistemas digitales. A continuación veremos su contraparte, que es la base de los transmisores de comunicación serie de los sistemas digitales.

Registro de desplazamiento paralelo-serie.



En este registro los biestables también son de tipo activado por flanco. En este caso, en lugar de una conexión directa entre salidas Q y entradas D, se intercala entre ellas multiplexores 2:1, como se muestra en la figura. Estos multiplexores tienen un único bit de selección conectado a la entrada $\bar{\mathbf{C}}/\mathbf{D}$ (carga/desplazamiento).
Observe que si $\bar{\mathbf{C}}/\mathbf{D}$ = 0 el circuito se comporta como un registro paralelo/paralelo activado por flanco, es decir permitiría almacenar un valor de N bits proveniente de las entradas b0, b1, b2, b3, mientras que si $\bar{\mathbf{C}}/\mathbf{D}$ = 1 el circuito se comporta como un registro de desplazamiento, permitiendo así que los bits almacenados en los biestables vayan saliendo ordenadamente a través de la salida Tx.

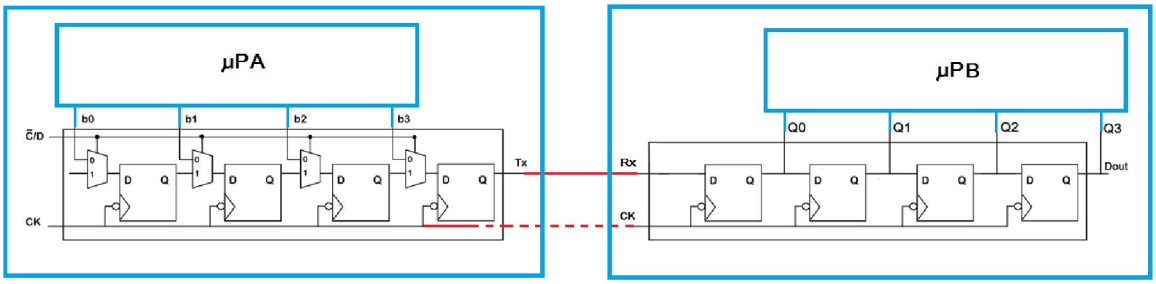
Aplicación en comunicación serie.

En el siguiente esquema se representa una implementación básica de una comunicación serie entre un equipo transmisor (por ejemplo una computadora, un PLC etc.) y otro equipo receptor.

El microprocesador del equipo transmisor (uPA) carga los bits b0 a b3 en el registro paralelo-serie, que luego son transmitidos en serie por la salida **Tx** dando pulsos a CK (no se muestran estas conexiones por simplicidad).

El registro serie/paralelo del equipo receptor recibe estos pulsos por la entrada **Rx** y – luego de N pulsos de CK, el microprocesador uPB lee los bits recibidos en paralelo (Q0 a Q3).

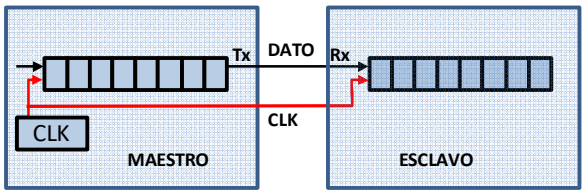
La línea en rojo, que conecta Tx y Rx, es la línea de datos. Puede ser un cable simple, o tener algún tipo de adaptación eléctrica para poder atravesar grandes distancias, e incluso podría ser una conexión inalámbrica. La línea de CK se ha dibujado en línea de puntos porque, dependiendo de la aplicación, podrá estar presente o no. Obviamente el registro serie/paralelo del receptor necesita pulsos en CK para ir desplazando los bits recibidos en Rx, pero existe la posibilidad de que el receptor tenga su propio generador de pulsos de CK, que deberá estar sincronizado con el generador de pulsos de CK del transmisor.



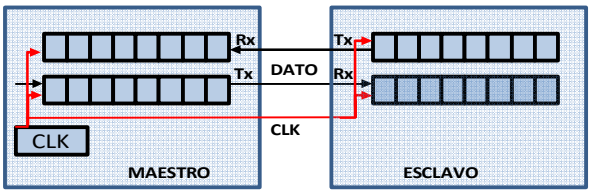
En el siguiente esquema vemos ambas implementaciones. Hemos representado simplificadamente los registros de desplazamiento como celdas rectangulares contiguas 

La primera se denomina **comunicación síncrona**, y se utiliza en distancias muy cortas, por lo general menores a 1 metro (por ejemplo para comunicar dispositivos dentro de un mismo sistema (microprocesadores entre sí, o microprocesadores con sensores, memorias serie, conversores A/D y D/A serie etc). Ejemplos de esta comunicación síncrona son los buses **SPI** e **I2C**.

En esta comunicación, el equipo que maneja el CLK se denomina MAESTRO. Observe que es posible también realizar una comunicación bidireccional colocando un registro Paralelo/serie en el esclavo y un serie/paralelo en el Maestro, para así poder establecer una "conversación" entre ambos equipos.

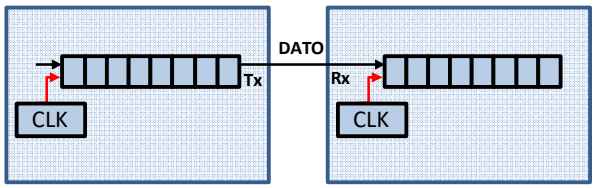


Comunicación síncrona unidireccional (simplex)

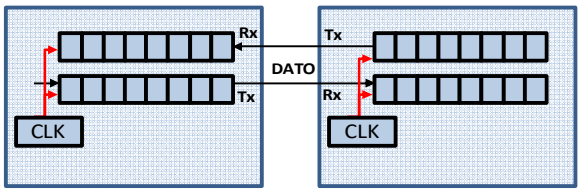


Comunicación síncrona bidireccional (duplex)

La otra posibilidad es que cada equipo tenga su propio generador de pulsos (CLK). Por supuesto es necesario que ambos generadores estén sincronizados, es decir que sus frecuencias sean prácticamente iguales. En las comunicaciones de este tipo hay que "consensuar" la velocidad de trasmisión, que se denomina bps (bits por segundo). Se utiliza para distancias mayores, incluso permitiendo una comunicación inalámbrica (*wireless*). Ejemplos de esta comunicación son las UART (que son cableadas), CAN y otras. Nota: En el caso de otras como USB y Ethernet, aunque tampoco comparten la línea de CLK, no se consideran asíncronas puras porque utilizan técnicas de sincronización dentro de la propia trama de datos (algo se verá en la Unidad 7).



Comunicación asíncrona unidireccional (simplex)

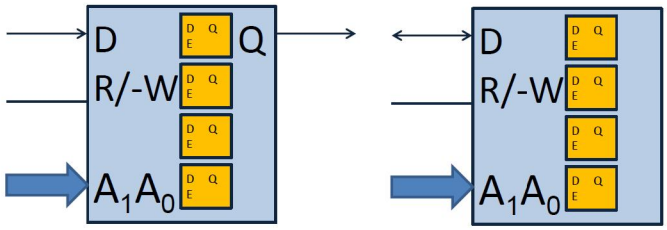


Comunicación asíncrona bidireccional (dúplex)

3.D. Memorias.

Memoria RAM estática (SRAM)

La idea básica es poder almacenar o "escribir" un bit en un biestable determinado, seleccionado mediante una combinación binaria A1A0 denominada "dirección", y luego poder recuperarlo. Con la entrada de control R/-W=0 se realiza la escritura, y con R/-W=1 la lectura.
En un esquema con entradas y salidas separadas el bit a escribir se presenta en la entrada D, y luego se lee por la salida Q.
En un esquema con entrada y salida única (bidireccional), el bit se presenta para escribir o se lee por el mismo terminal. Es lo más habitual.



RAM 4 bits con E/S separadas

RAM 4 bits con E/S bidireccional

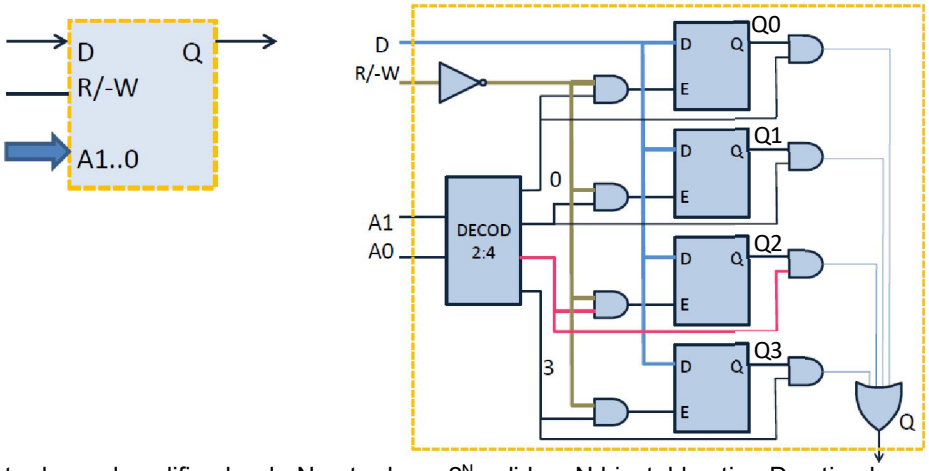
Operación de Escritura: 1) Poner dirección en A1A0 y dato en D.
2) Aplicar pulso negativo en R/-W
3) Ya se puede quitar el dato

Operación de Lectura: 1) Con R/-W en 1, poner dirección en A1A0. En Q se tendrá el dato direccionado. (En la memoria E/S bidireccional, estará en D)

Comenzaremos a analizar el funcionamiento de distintas implementaciones, desde las más sencillas para entender las ideas básicas, hasta otras más cercanas a los circuitos utilizados en la práctica.

RAM con E/S separadas

Esta no es una implementación habitual, pero permite entender la idea principal.

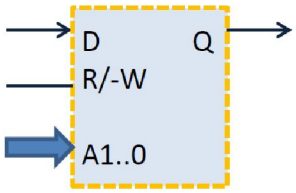


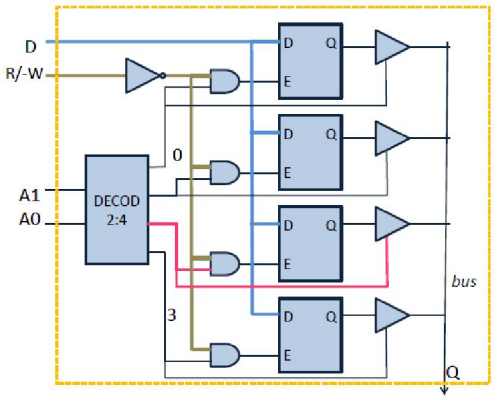
Consta de un decodificador de N entradas y 2^N salidas, N biestables tipo D activados por nivel, y puertas auxiliares.

Al poner un dato en la entrada D, éste se presenta en las entradas D de los 4 biestables. Las salidas del decodificador combinadas en las AND de entrada junto con la señal R/-W=0 habilitarán la escritura de uno de ellos, aquel que sea direccionado por A1 A0. En el ejemplo supone seleccionado el 3er biestable.

Observe que el mismo decodificador – junto con las AND de salida y la OR de salida – **forma un multiplexor** que selecciona una de las salidas Q según los bits A1A0. En este caso Q2

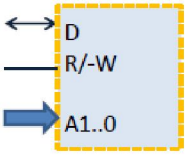
RAM con E/S separadas y buffers de Tercer Estado (*Triestate*)

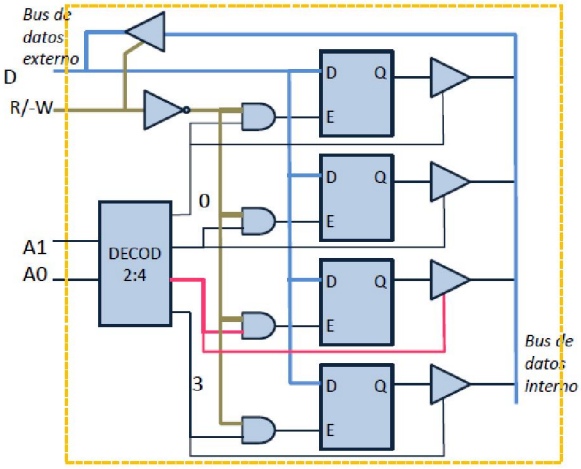




La escritura es igual que antes.
En la salida, en vez de un multiplexor, el decodificador habilita uno de los buffers *Triestate* permitiendo que la salida Q correspondiente tome el control del bus.

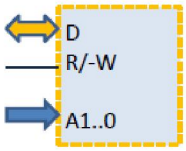
RAM con E/S bidireccional

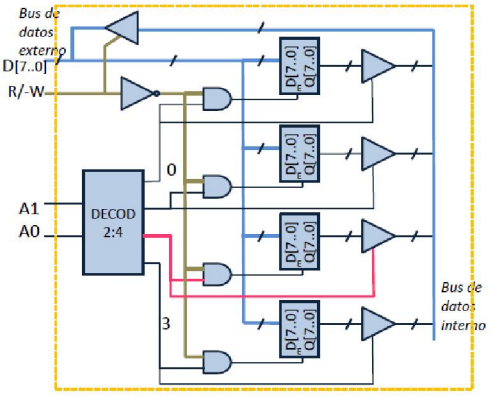




La escritura es igual que antes.
La misma entrada R/-W se utiliza para arbitrar qué señal toma el control del bus de datos externo.
Con R/-W=1, los datos salen de la RAM (lectura), con R/-W=0 (escritura) el buffer agregado impide la colisión.

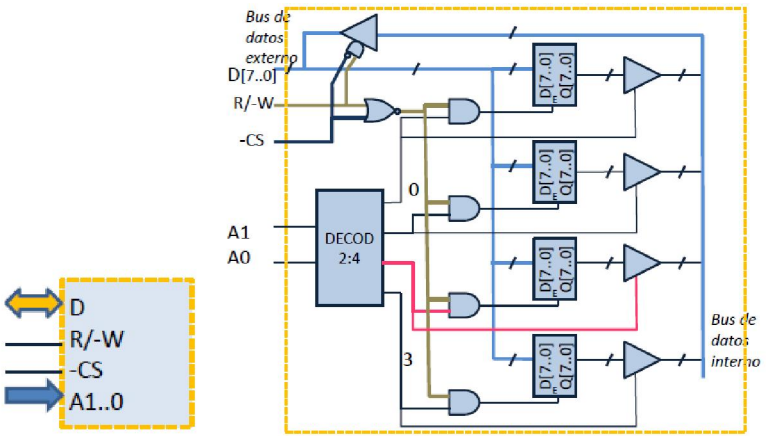
RAM estática de 4 Bytes con E/S bidireccional





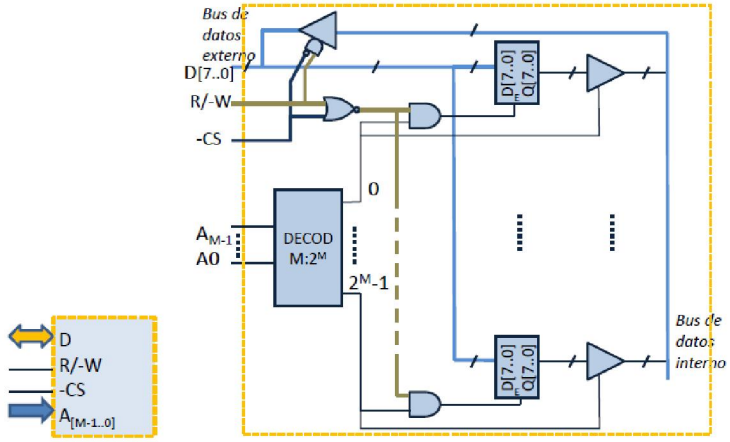
La estructura de control es la misma de antes, pero se ha reemplazado los biestables D por registros paralelo/paralelo de 8 bits. En la salida de cada registro hay 8 buffers *Triestate*, representados como uno solo, pero que debe interpretarse como un grupo de 8 buffers que son controlados simultáneamente.

RAM estática de 4 Bytes con *chip select*.



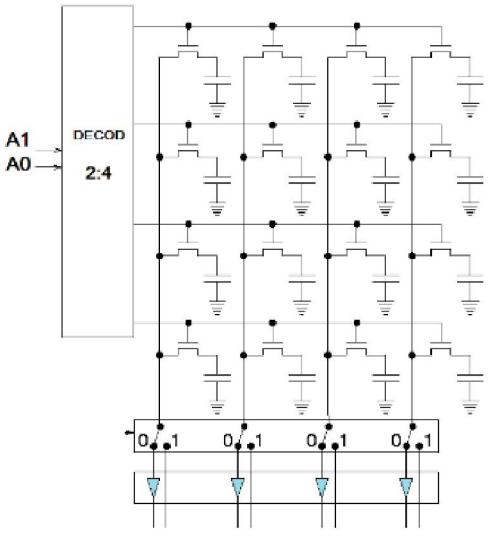
Sobre el esquema anterior se ha agregado una entrada de control *Chip Select* . Esta entrada permite anular completamente las operaciones de lectura escritura, permitiendo aislar el chip completo. En el esquema, con -CS=1 se anulan las operaciones de lectura y escritura, con -CS=0 se habilitan. Esto es fundamental para poder armar bancos de memoria más grandes.

Ejemplo más general: RAM estática de 2^M Bytes



Con un bus de direcciones de M bits se direccionan 2M bytes.

Memorias RAM Dinámicas (DRAM y SDRAM)



En las memorias RAM estáticas recién vistas el almacenamiento de cada bit se realiza en un biestable tipo D. Esto requiere un gran número de transistores por cada bit. Una alternativa para lograr dispositivos de gran densidad de almacenamiento es utilizar una capacitancia como celda de almacenamiento.
En el circuito de la figura se muestra un fragmento de una memoria RAM dinámica. Se observa en la salida un conjunto de llaves. Estas llaves son electrónicas (transistores), y permiten seleccionar la operación de lectura (posición 0) o escritura (posición 1). También se observan unos *buffers* (en color celeste).
Durante la escritura se cargan o descargan las capacitancias de las celdas seleccionadas por el decodificador, mientras que durante la lectura el voltaje de las celdas pasa a la salida a través de los *buffers*. Los

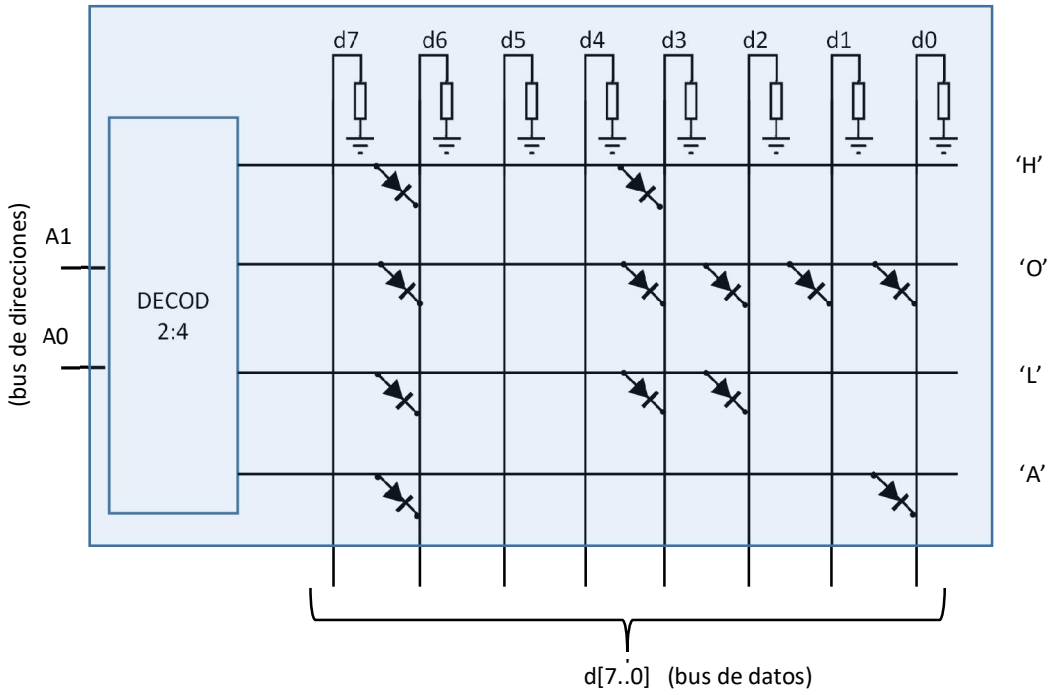
buffers (en este contexto) son compuertas que replican en la salida el valor lógico de la entrada, pero que reconstituyen el voltaje, piden mínima corriente en la entrada y pueden entregar mayor corriente a la salida (así se evita que se descarguen los capacitores durante la lectura).
Los capacitores de las celdas tienden a perder el valor lógico almacenado al cabo de unos milisegundos, debido a corrientes de fuga. Por este motivo necesitan un refresco periódico, esto es una lectura y recarga del valor lógico en el capacitor, para lo que debe disponerse de un mecanismo de barrido cíclico de todas las direcciones, lo que dificulta su uso.
A diferencia de las **DRAM** más antiguas, las memorias **SDRAM** (*Synchronous Dynamic RAM*) incorporan internamente el circuito de barrido, lo que simplifica su uso.

Memoria ROM (read Only Memory) o MROM (Mask ROM)

La base de todas las memorias tipo ROM (MROM, PROM, EPROM, EEPROM, FLASH) es similar. Consta de un Decodificador, de forma similar a las RAM vistas, pero en vez de biestables habrá una especie de matriz de interconexión, que conecta cada una de las salidas del decodificador, con cada bit de salida de datos de la memoria. El almacenamiento de unos y ceros consistirá simplemente en la presencia o ausencia de la conexión.

En el ejemplo de la figura, se representa una **MROM** (*Mask ROM*) de 4 bytes. En la salida 0 del decodificador se ha conectado (durante la fabricación del chip) los diodos correspondientes al número binario 01001000, que en código ASCII es el carácter 'H'. De forma similar se ha procedido con las filas 1, 2, 3, en las que se ha conectado los binarios correspondientes a los caracteres 'O', 'L', 'A' respectivamente.

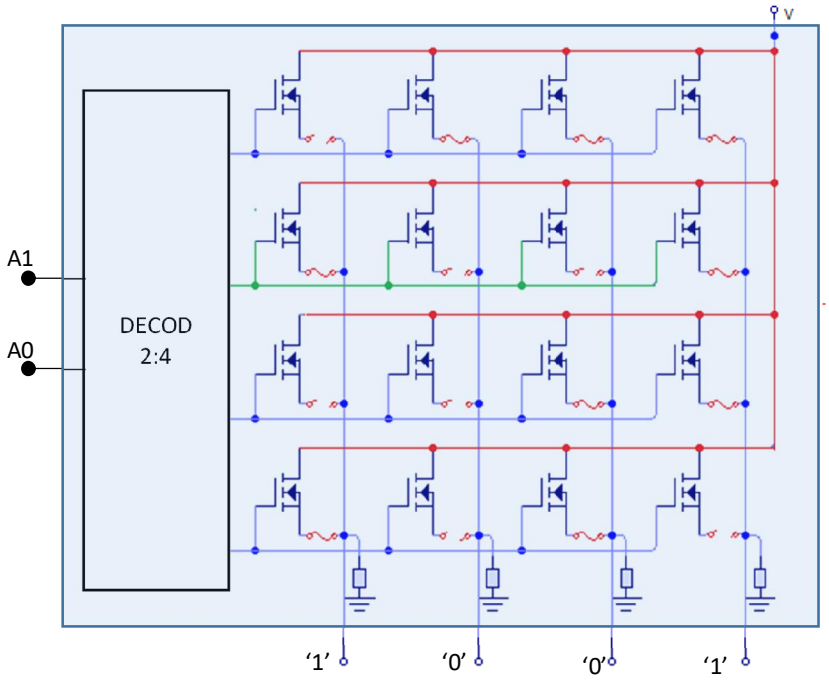
H-O-L-A = 0x48-0x4f-0x4C-0x41 = 01001000 – 01001111 – 01001100 - 01000001



Como se ve, directamente en el proceso de fabricación se ha predefinido los contenidos de cada posición de la memoria. Esto podría es económicamente viable si se fabrican grandes cantidades, digamos más de unas 10.000 unidades. Además será una memoria sumamente estable, permitiendo su uso en ambientes hostiles sin perder su contenido.

Memoria PROM (Programmable Read Only Memory) u OTP (One Time Programmable)

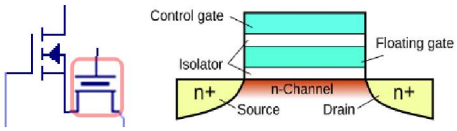
En este caso en cada cruce de fila/columna hay un transistor MOSFET y un fusible. La memoria viene inicialmente con todas las conexiones, es decir en todas las posiciones todos los bits están en '1'. Desde el exterior el usuario puede programarla quemando los fusibles. Este proceso es **irreversible**.



El principio de quemado es simple: durante la programación se presenta en el bus de direcciones la dirección a escribir, y en el bus de datos se presenta la combinación de unos y ceros que se quiere grabar. En los transistores de la fila seleccionada, y cuyos terminales 'source' estén conectados a '0' se producirá una circulación de corriente que quemará el fusible. En los demás transistores no habrá circulación de corriente. Este proceso se repite hasta grabar todas las posiciones deseadas. Luego la memoria se utilizará en su modo normal, es decir su bus de datos será solamente salida.

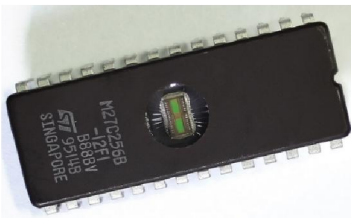
Memoria EPROM (Erasable-Programmable Read Only Memory)

En estas memorias, en lugar de un fusible, cada celda lleva un transistor MOSFET especial, que cuenta con una **puerta flotante** adicional *(floating gate)*. Esta puerta flotante es una isla metálica (normalmente silicio policristalino dopado N) interpuesta entre la **Puerta de Control** (Control Gate) y el canal, rodeada de aislante (dióxido de silicio) puesta a muy corta distancia del canal.



Mediante una combinación adecuada de tensiones en *Source, Drain* y *Control Gate* (Fuente, Drenaje, Puerta de control) es posible inyectar (y extraer) electrones de la puerta flotante. Los fenómenos físicos que permiten esta migración de portadores a través de una barrera aislante delgada son el denominado **Inyección de Portador Caliente** (*hot carrier injection*), que implica un portador de alta energía cinética que atraviesa el aislante, y el **Efecto Túnel**, que es un fenómeno cuántico.

Inicialmente, cuando la memoria está borrada, no hay electrones inyectados en la puerta flotante de cada celda de memoria, y el transistor se comportará de forma normal, es decir formando el canal y conduciendo cuando se le aplique voltaje en la Puerta de Control, con lo cual esa celda se leerá como un '1' (del mismo modo que en la PROM cuando el fusible está sano). En cambio, cuando se han inyectado electrones en la puerta flotante, esta carga negativa se interpone a la puerta de control, impidiendo que se forme el canal, y esta celda se leerá como un '0'.



En las memorias EPROM, el borrado no se realiza extrayendo los electrones eléctricamente, sino mediante la exposición a radiación UV. Para esto, los chips de memoria EPROM cuentan con una ventana de cuarzo. El borrado se realiza con una lámpara UV similar a las utilizadas

para esterilizar, ubicada a unos pocos centímetros del chip, y dura aproximadamente unos 20 minutos. Luego de este borrado todos los bits vuelven a estar en '1'.
Para evitar el borrado accidental o paulatino por luz ambiente, la ventana debe cubrirse con una película opaca a los rayos UV, normalmente un papel autoadhesivo de aluminio.

Memorias EEPROM y FLASH

En las memorias EPROM el borrado es muy lento, y normalmente requiere quitar el chip de la placa en la que está conectado.
Para agilizar el proceso de borrado se crearon las memorias **EEPROM** (*Electrically Erasable-Programmable Read Only Memory*) y las memorias **FLASH**. En estas memorias la extracción de portadores de la puerta flotante (es decir el borrado) se realiza **eléctricamente**. Este proceso es mucho más rápido, pudiendo realizarse en un tiempo de 1 a 5 ms.
En las EEPROM el proceso de borrado se realiza en cada dirección **en forma individual**, mientras que en las FLASH el borrado se realiza **por bloques**, por ejemplo de 64 direcciones o más.

Consideraciones prácticas

- Se han mostrado los principales tipos de memorias, aunque hay otros mecanismos de almacenamiento y es un campo en constante evolución.
- Cada tipo de memoria tiene su campo de aplicación en circuitos comerciales e industriales. **Lo invitamos a averiguar y discutir estos campos de aplicación.**
- Hemos visto que las memorias tienen un **Bus de direcciones** que corresponde a las entradas de un decodificador, un **Bus de Datos** y líneas de control (como *Read/Write, Chip Select, Output Enable*). Hemos mostrado un **acceso paralelo** a estos buses,es decir que el chip de memoria debería tener al menos tantos pines como bits tengamos en estos buses, más los bits de control y de alimentación. Este acceso paralelo es el utilizado cuando se requiere muy alta velocidad de acceso. Sin embargo existen memorias en las que – utilizando registros de desplazamiento – se consigue reducir drásticamente el número de pines. Estas memorias se denominan **memorias serie**, y existen en gran variedad de modelos de *Serial RAM*, *Serial EEPROM*, *Serial FLASH*.

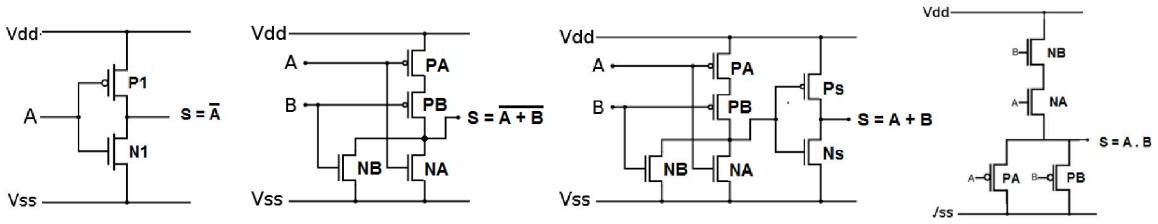
3.E. Tecnología: Esquemas de salida en puertas digitales.

Cuando analizamos el funcionamiento de las compuertas lógicas, nos centramos en su comportamiento lógico.

En cuanto a su implementación, vimos la OR y la AND realizadas con diodos (tecnología ***Diode Logic*** *o* **DL**), y la NOT realizada con un transistor y resistencias en base y colector (tecnología ***Resistor Transistor Logic*** o **RTL**). Estas tecnologías son sencillas, pero presentan muchas limitaciones. No nos adentraremos en las diferentes tecnologías que permiten superar estas limitaciones, pero sí vamos a explicar brevemente qué características son deseables en una puerta digital para obtener altas prestaciones en los circuitos digitales.
Consumo: Es la corriente que se extrae de la fuente de alimentación para funcionar. Debe ser **muy bajo** en régimen estático (cuando se mantiene el estado lógico) y dinámico (cuando ocurren cambios de estado lógico). Normalmente en régimen dinámico se consume más corriente.
Velocidad de conmutación: Debe ser alta para permitir altas velocidades de procesamiento.
Corriente de entrada: Debe ser **muy baja** para no cargar a la puerta precedente (es decir, debe tener alta impedancia de entrada). Normalmente es mayor en régimen dinámico debido a la carga y descarga de la capacitancia parásita de entrada. Se relaciona con el FAN-IN de una compuerta..
Corriente de salida: Debe ser **suficiente** para que el nivel lógico no se degrade al conectar una o más compuertas en la salida, y también **para acelerar** la carga y descarga de las capacitancias parásitas de dichas compuertas. Se relaciona con el FAN-OUT de una compuerta.

Aunque hay muchas, las dos tecnologías principales para implementar compuertas digitales son las denominadas **CMOS** (*Complementary* MOS), basada en MOSFETs de enriquecimiento Canal N y Canal P, y la **TTL** (*Transistor-Transistor Logic*), basada en transitores bipolares. Ambas tecnologías han seguido evolucionando para mejorar sus prestaciones, dando lugar a algunas variantes, que escapan a los alcances de la asignatura.
La tecnología **CMOS** presenta ventajas frente a la **TTL** en cuanto a bajo consumo y bajas corrientes de entrada, lo que la hace ideal para lograr altas escalas de integración. Todos los circuitos de procesadores son CMOS.

Ejemplos *(Nota: Vdd es el voltaje positivo, Vss es 0 volts o masa)*



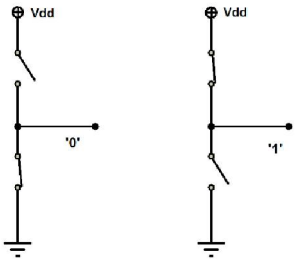
El circuito (a) es un inversor CMOS, con P1 MOSFET de canal P y N1 MOSFET de canal N. Cuando A=1, N1 conduce a masa y P1 está en corte, con lo que la salida es S=0. Cuando A=0, P1 conduce a Vdd y N1 está en corte, con lo que la salida resulta S=1.

El circuito (b) es una NOR CMOS. Cuando A=1 **o** B=1, la rama serie PA-PB no conducirá, y el paralelo NB-NA conducirá a masa, con lo cual la salida será S=0. Solo cuando A=0 y B=0, la rama serie PA-PB conducirá a Vdd y el paralelo NB-NA no conducirá, con lo que la salida será S=1.

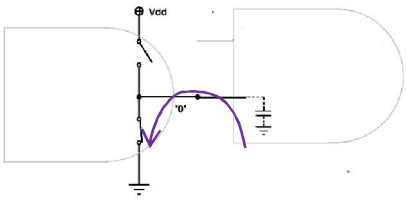
El circuito (c) es una combinación del (b) con el (a) en la salida, de modo que es una OR: S = A+B

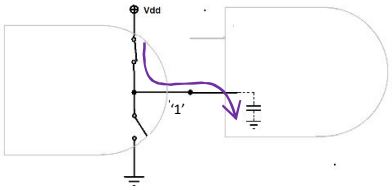
El circuito (d) es una AND CMOS. Cuando A=0 o B=0, la rama serie NA-NB no conducirá, y el paralelo PA-PB conducirá a masa, resultando S=0. Cuando A=1 y B=1, la rama serie NA-NB conduce a Vdd, y el paralelo PA-PB no conduce, resultando S=1. (ver similitud con el circuito b y recordar leyes de De Morgan).

Salida complementaria o *Push-Pull*

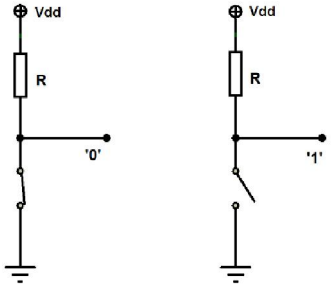


En todos estos circuitos se ve que la salida siempre es conducción a Vdd o conducción a masa (o Vss). Decimos que es una salida tipo **complementaria**. La podemos representar simplificadamente como dos llaves que trabajan en forma complementaria, es decir, cuando una abre la otra cierra y viceversa. De esta forma se logra una conexión directa a Vdd y a masa, sin pérdidas de tensión. Esto va a permitir cargar y descargar rápidamente la capacitancia parásita de la puerta subsiguiente

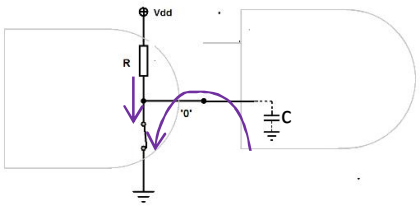


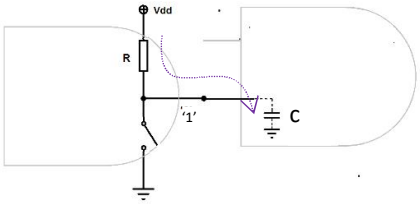


Salida colector abierto o drenaje abierto (*open Collector*, *open Drain*)



Si consideramos por ejemplo la compuerta NOT realizada con un transistor (ver tema anterior), observamos que el circuito trabaja con un transistor como llave y una resistencia R, de modo que el '0' es una conexión directa a masa, y el '1' es una conexión a Vdd pero a través de R. Podemos decir que el '0' es fuerte y el '1' es débil, y si se le pide corriente habrá caída de tensión en R, que si es excesiva puede degradar el valor lógico. Además se limitará la velocidad de conmutación, pues cuando la salida es un '1' lógico tardará en establecerse el voltaje en la capacitancia parásita (hay una constante de tiempo R.C)





Buffers de Tercer estado (*Triestate*)

En forma simplificada podemos representar el modelo eléctrico del Buffer de Tercer Estado como una llave conectada en serie, de modo tal que cuando la llave está cerrada conecta el circuito interno de la puerta lógica con la salida, y cuando está lo desconecta. Esta llave está controlada por una entrada de habilitación E (*enable*).
Al estado de desconexión se lo denomina de **alta impedancia**, y se lo representa por la letra Z. La

E	X
0	Z
1	A

E	X
0	A
1	Z

En el tema de memorias RAM estáticas vimos la necesidad de utilizar *buffers* de Tercer Estado o *Triestate*. Hay muchas más aplicaciones, como en microprocesadores que veremos en la Unidad 4, buses de comunicaciones que veremos en las unidades 7 y 8. Esta necesidad surge para conectar muchas salidas a un mismo nodo, de modo que se pueda elegir en un momento qué salida maneja el nodo. Si bien sería factible utilizar un multiplexor, es mucho más sencillo mediante buffers *Triestate* (ver por ejemplo en el tema de memoria RAM estática).

Las salidas de tipo complementaria o *Push-Pull* vistas anteriormente no pueden conectarse directamente entre sí, pues habría **colisión de datos** (cortocircuito) cuando una compuerta entregue un '1' (con su transistor de arriba conectado a Vdd) y otra compuerta entregue un '0' (con su transistor de abajo conectado a masa), lo que puede destruir estos transistores. Este inconveniente no lo tienen las compuertas de colector o drenaje abierto, pues ante diferente valor lógico de las salidas, la resistencia R limita la corriente del lado positivo. Así, el '0' es dominante sobre el '1'.

El *buffer* de Tercer Estado soluciona este inconveniente de las salidas complementarias.