TEMA 5.3
SISTEMAS DIGITALES

CUD

Centro Universitario de la Defensa Zaragoza

TEMA 5
SISTEMAS DIGITALES

FUNDAMENTOS DE ELECTRÓNICA

08 de enero de 2015

TEMA 5.3 – SISTEMAS DIGITALES

- Introducción
- Sistemas combinacionales
- Sistemas secuenciales

Centro Universitario de la Defensa Zaragoza

TEMA 5.3 – SISTEMAS DIGITALES

- Introducción
- Sistemas combinacionales
- Sistemas secuenciales

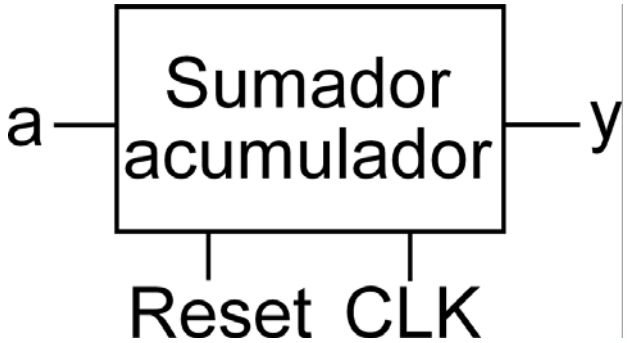
INTRODUCCIÓN

- Combinacionales:
  - Basados en puertas lógicas
  - Salida depende sólo de la entrada:
    - Sumadores (restadores)
    - Comparadores
    - Codificadores
    - Decodificadores
    - Conversores de Código
    - Demultiplexores
    - Multiplexores
    - Memorias ROM
  - Ejemplo: sumador

INTRODUCCIÓN

- Combinacionales:
  - Basados en puertas lógicas
  - Salida depende sólo de la entrada
  - Ejemplo: sumador

- Secuenciales:
  - Incluyen biestables
  - Salida depende de las entradas y del estado:
    - Latch/Flip Flop
    - Registros
    - Contadores
    - Memorias SRAM
  - Ejemplo: sumador - acumulador

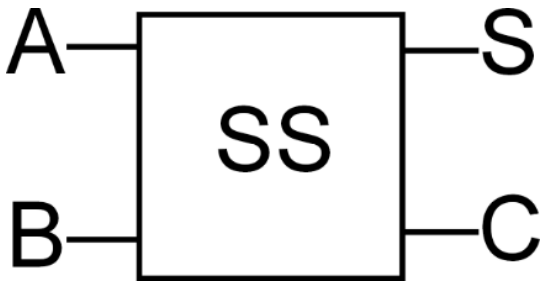


TEMA 5.3 – SISTEMAS DIGITALES

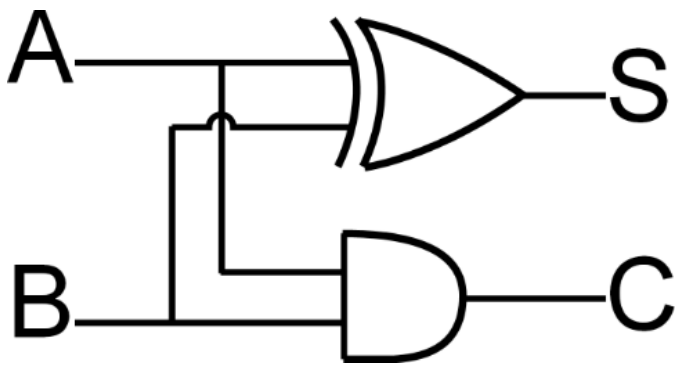
- Introducción
- Sistemas combinacionales
- Sistemas secuenciales

SEMISUMADOR (half adder)

- Suma binaria de dos bit
  - Resultado de la suma (S): XOR
  - Acarreo (C): AND



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

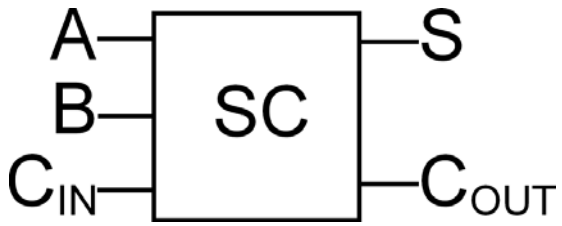


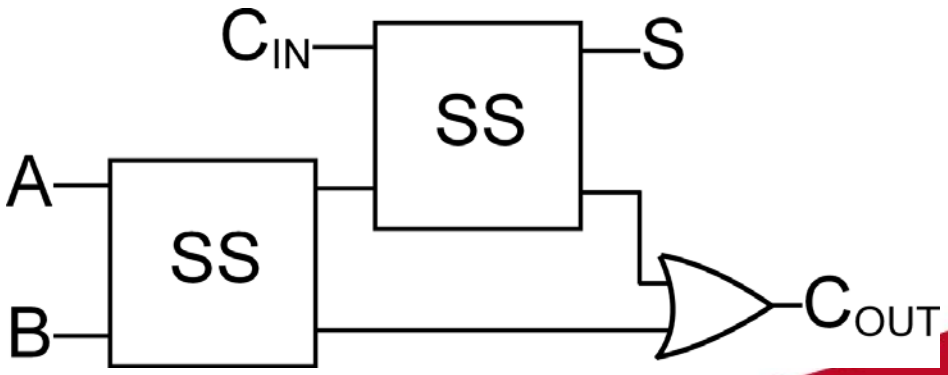
- El "semisumador" sólo es adecuado para sumar dos bits. No tiene en cuenta el bit de acarreo que se puede producir en la suma aritmética binaria

SUMADOR COMPLETO (full adder)

- El sumador completo tiene en cuenta el posible acarreo (C_{IN})
- Está formado por dos semisumadores conectados en cascada

A	B	C_{IN}	S	C_{OUT}
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

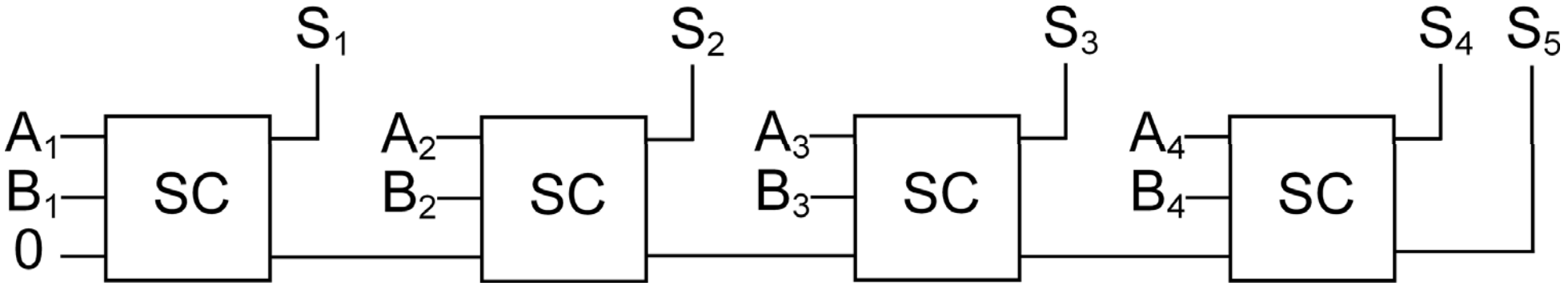




PROPAGACIÓN DE ACARREO SERIE

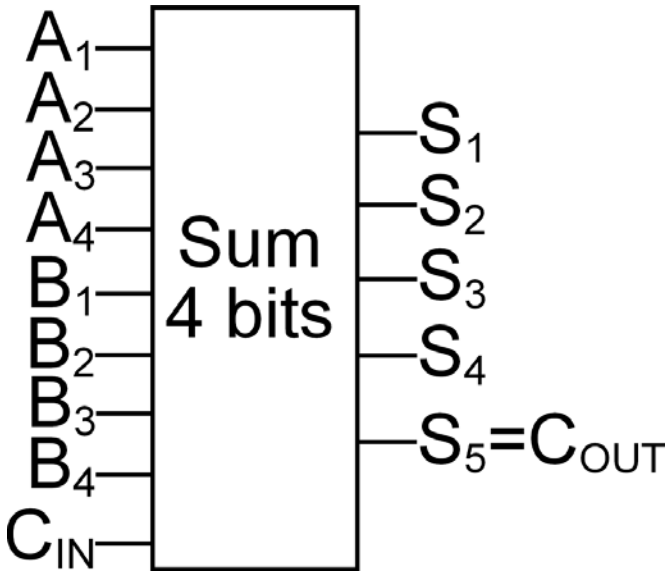
- Para formar un sumador de n bits, unimos varios sumadores completos en cascada. Cada uno tiene en cuenta el acarreo del anterior.
- Ejemplo: 4 bits

$$\begin{array}{r} A_4A_3A_2A_1 \\ +\ B_4B_3B_2B_1 \\ \hline S_5S_4S_3S_2S_1 \end{array}$$



ACARREO ANTICIPADO

- El problema es que el tiempo de retardo de cada circuito se va acumulando por la conexión en cascada.
- Solución: acarreo anticipado
- Integrado comercial: 74283
  - Sumador de 4 bits
  - Con acarreo anticipado
  - Permite conexión en cascada

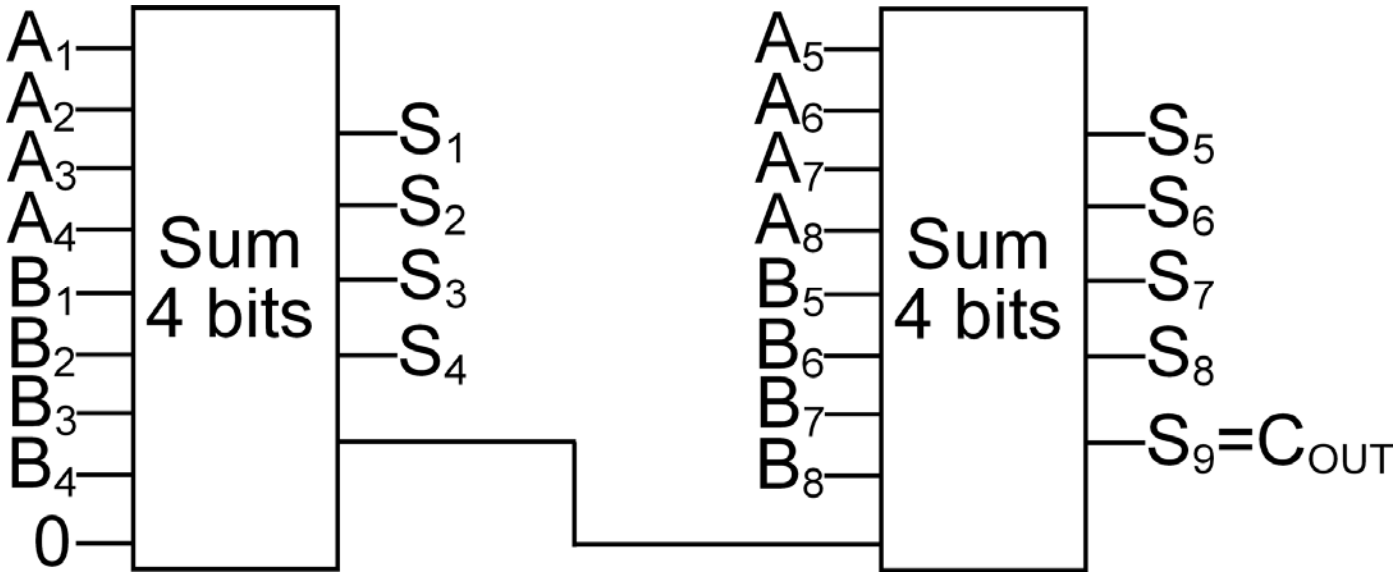


A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	C_{IN}	S_1	S_2	S_3	S_4	S_5
2^9 (512) filas													

CONEXIÓN EN CASCADA

- 2 sumadores de 4 bit permiten implementar un sumador de 8 bits

$$\begin{array}{r} A_8A_7A_6A_5A_4A_3A_2A_1 \\ +\ B_8B_7B_6B_5B_4B_3B_2B_1 \\ \hline S_9S_8S_7S_6S_5S_4S_3S_2S_1 \end{array}$$



SUMADOR/RESTADOR

- Restar dos números es equivalente a sumar al minuendo el complemento a 2 del sustraendo.
  - A - B = A + Ca2(B)
- Recordemos que Ca2 consiste en complementar todos los bits y sumarle 1 al resultado. Según esto podemos diseñar un sumador/restador de 4 bits basándonos en un sumador de 4 bits y puertas XOR, ya que:
  - Si a = 0:

$$y = a \oplus b = b$$

  - Si a = 1:

$$y = a \oplus b = \bar{b}$$

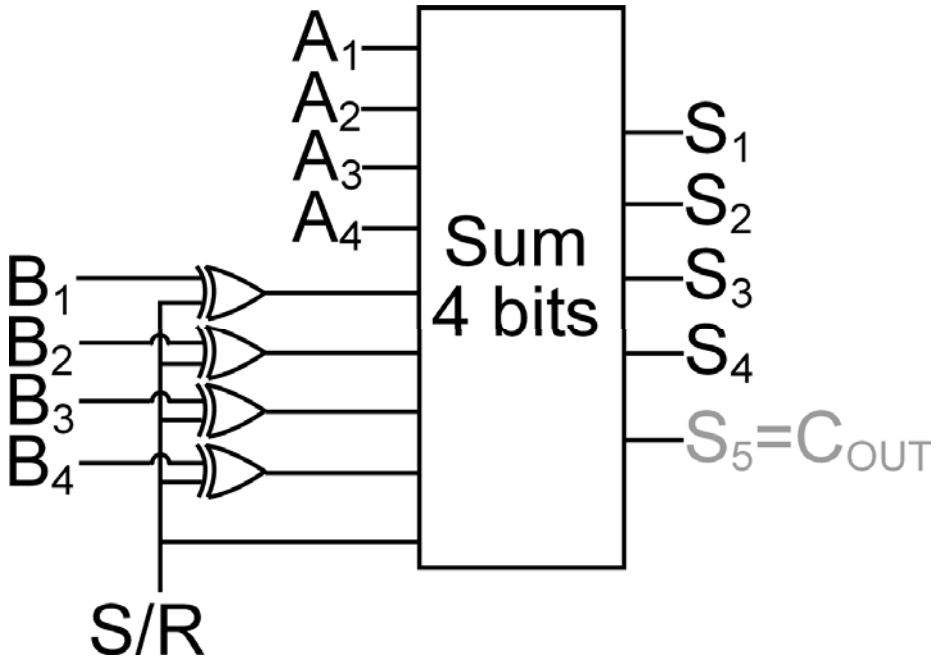
Complementamos el valor de la variable b (o no) en función de la variable a

Tabla de verdad de la función XOR

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

SUMADOR/RESTADOR

- Si S/R = 0
  - Los valores de las entradas B_1-B_4 no se complementan en las puertas XOR
  - El C_{IN} es 0
  - Por tanto, se realiza una suma
- Si S/R = 1
  - Los valores de las entradas B_1-B_4 sí se complementan en las puertas XOR
  - El C_{IN} es 1
  - Por tanto, se realiza una resta usando el Ca2



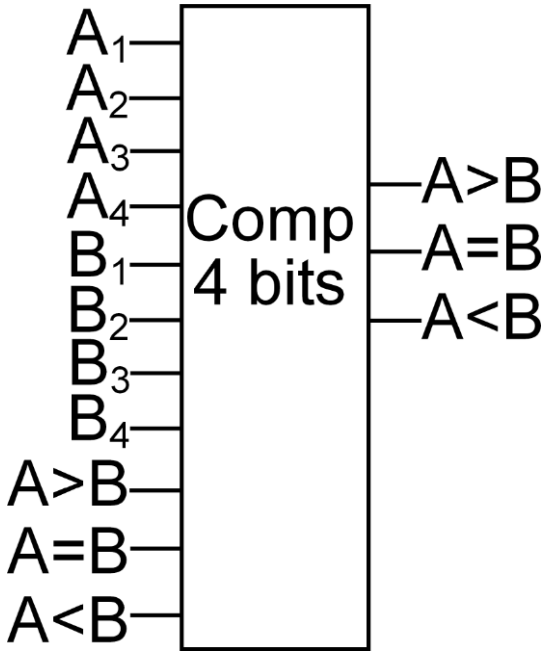
COMPARADORES

- Es un circuito que permite determinar si dos datos son iguales, o si uno es mayor que otro.

a	b	a>b	a=b	a<b
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

COMPARADORES

- El integrado comercia es el 7485. Compara dos números de 4 bits y permite conexión en cascada.

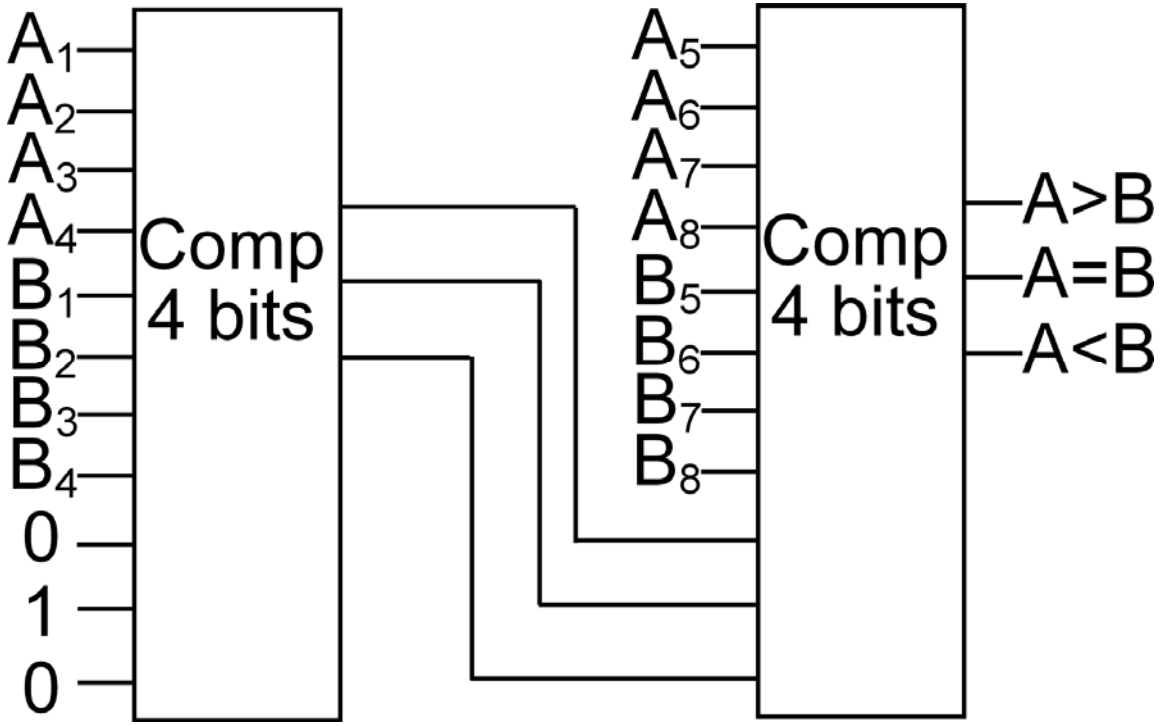


A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	A>B	A=B	A<B	A>B	A=B	A>B
2^{11} (2048) filas													

CONEXIÓN EN CASCADA

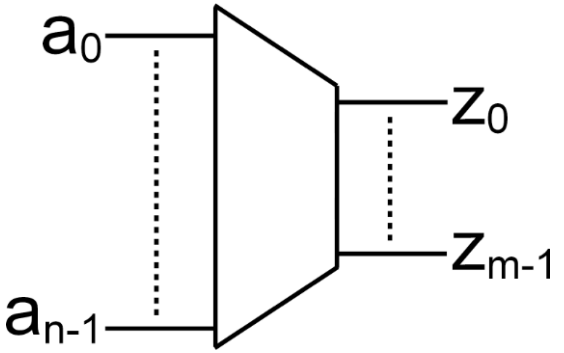
- Comparador de 8 bits = 2 comparadores de 4 bits en cascada

$$A_8A_7A_6A_5A_4A_3A_2A_1$$
$$B_8B_7B_6B_5B_4B_3B_2B_1$$



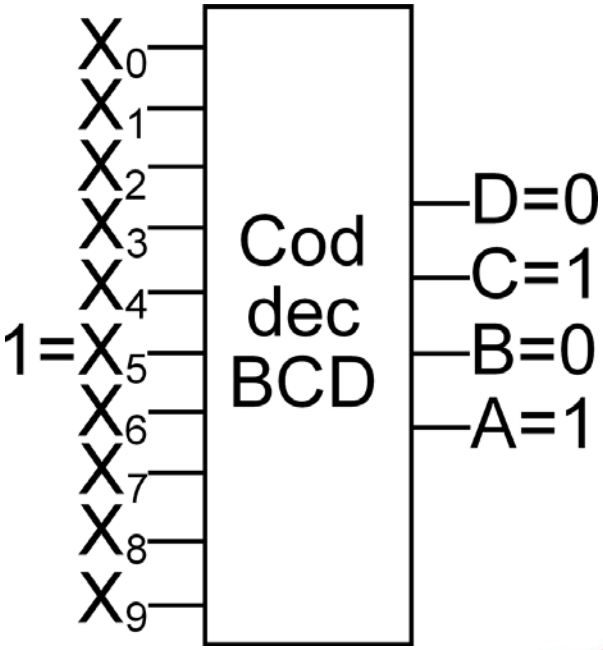
CODIFICADORES

- Es un circuito que permite transformar un nivel activo en una de sus entradas en un valor codificado



- De manera general, tiene n entradas y m salidas.

- Ejemplo: 74147 Decimal a BCD. 10 entradas y 4 salidas. Usado en los teclados numéricos

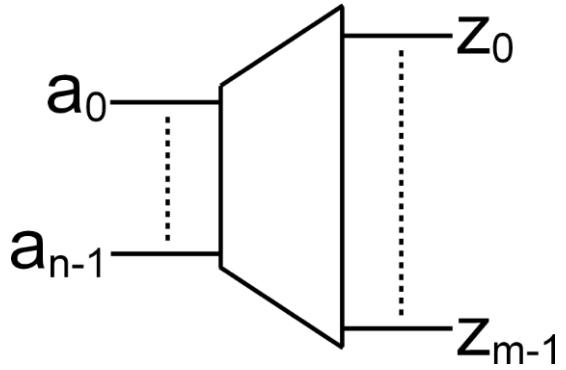


CODIFICADORES

- SIN PRIORIDAD:
  - Suponen que sólo una entrada puede estar activa.
  - Si se activan varias entradas a la vez, la salida puede ser errónea.
- CON PRIORIDAD:
  - Si se activan varias entradas a la vez, dan prioridad a una de ellas
  - Por ejemplo:
    - Al bit más significativo: se da prioridad a la entrada mayor
      - Si se activan In_1 y In_5, el resultado es 5
    - Al bit menos significativo: se da prioridad a la entrada menor
      - Si se activan In_1 y In_5, el resultado es 1

DECODIFICADORES

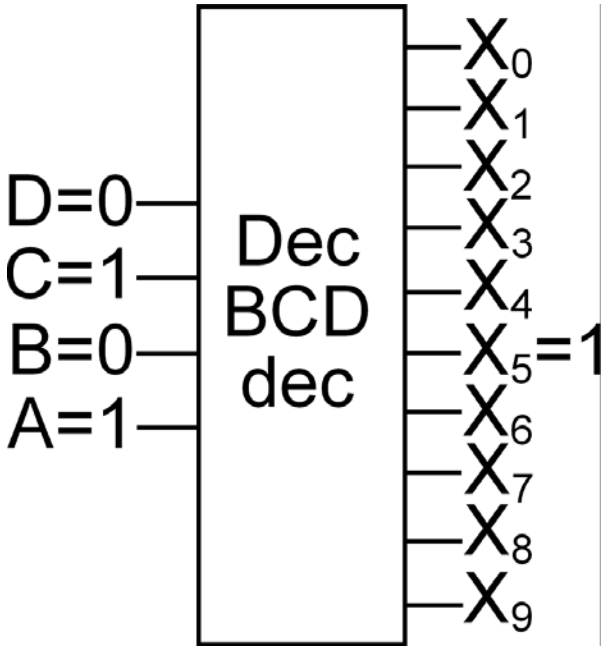
- Transforma un valor codificado en:
  - La activación de una única salida correspondiente a dicho valor. Realizan la función inversa a los codificadores.
  - Un valor codificado en otro código (conversores de código)



Símbolo genérico del decodificador con n entradas y m salidas

DECODIFICADOR

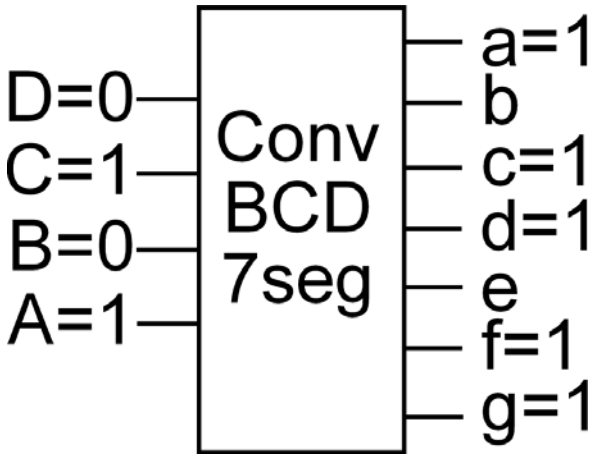
- Ejemplo: 7442
  BCD a decimal

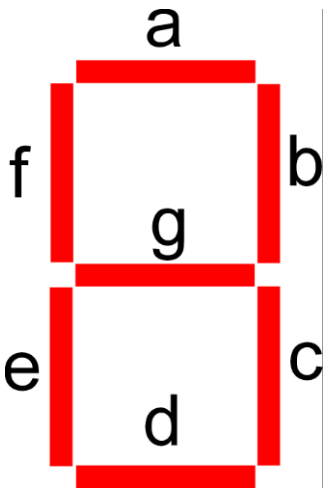


D	C	B	A	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1
resto													

CONVERSOR DE CÓDIGO

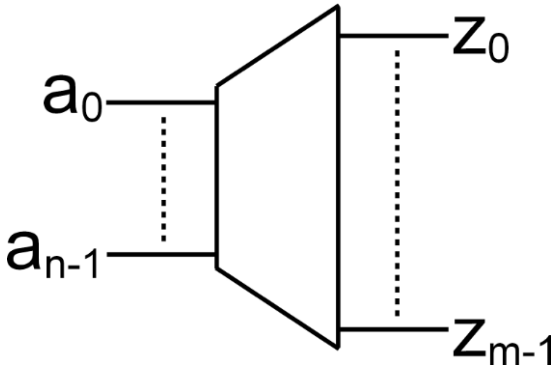
- Ejemplo: 7447 BCD a display 7 segmentos





- Una memoria ROM se puede interpretar como un conversor de código

Dirección de Memoria



Datos "almacenados"

DEMULTIPLEXORES

- Un demultiplexor es un circuito que copia el valor de la entrada de datos (d) en la salida (z) indicada por el valor de las señales de control (a).

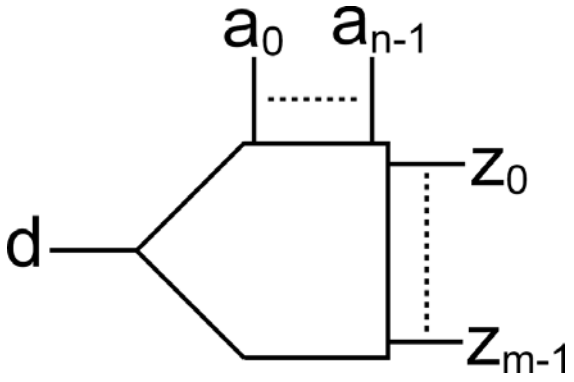


Tabla verdad para 3 bits de control y 8 salidas

a_2	a_1	a_0	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_0
0	0	0	0	0	0	0	0	0	0	d
0	0	1	0	0	0	0	0	0	d	0
0	1	0	0	0	0	0	0	d	0	0
0	1	1	0	0	0	0	d	0	0	0
1	0	0	0	0	0	d	0	0	0	0
1	0	1	0	0	d	0	0	0	0	0
1	1	0	0	d	0	0	0	0	0	0
1	1	1	d	0	0	0	0	0	0	0

MULTIPLEXOR

- El multiplexor es un circuito que permite seleccionar una de las entradas de datos (x) y copiar su valor a la salida (z). La entrada seleccionada depende del valor que se dé a las entradas de control (a).

a_2	a_1	a_0	z
0	0	0	x_0
0	0	1	x_1
0	1	0	x_2
0	1	1	x_3
1	0	0	x_4
1	0	1	x_5
1	1	0	x_6
1	1	1	x_7

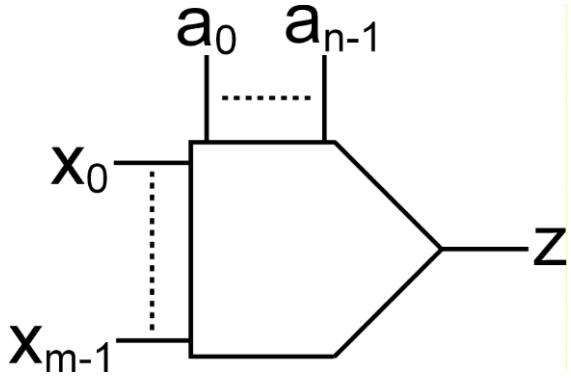


Tabla verdad para
3 bits de control
y 8 entradas

VARIABLES DE CONTROL

- Para un multiplexor (MUX), el número de entradas de datos (m) y el número de variables de control (n) están ligadas por

$$2^n = m$$

- Para un demultiplexor, existe la misma ligadura entre el número de salidas de datos (m) y el número de variables de control (n)

- Los MUX y DMUX se denominan por el número de entradas de datos: MUX2, DMUX4…

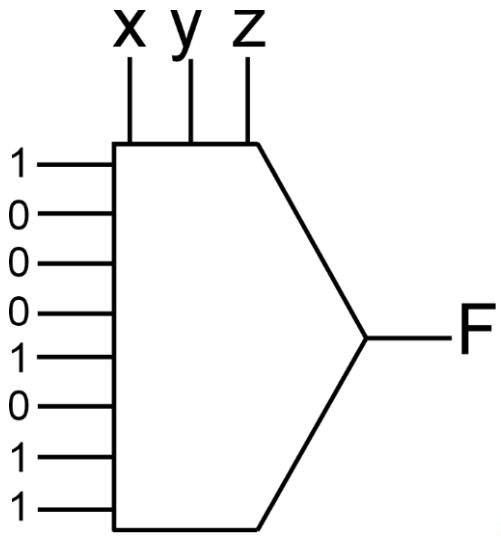
APLICACIONES DEL MULTIPLEXOR

- Un multiplexor de *n* entradas de control es un modulo lógico universal: permite sintetizar cualquier función lógica de *n* variables.

Tabla verdad de la función F que se quiere implementar

x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

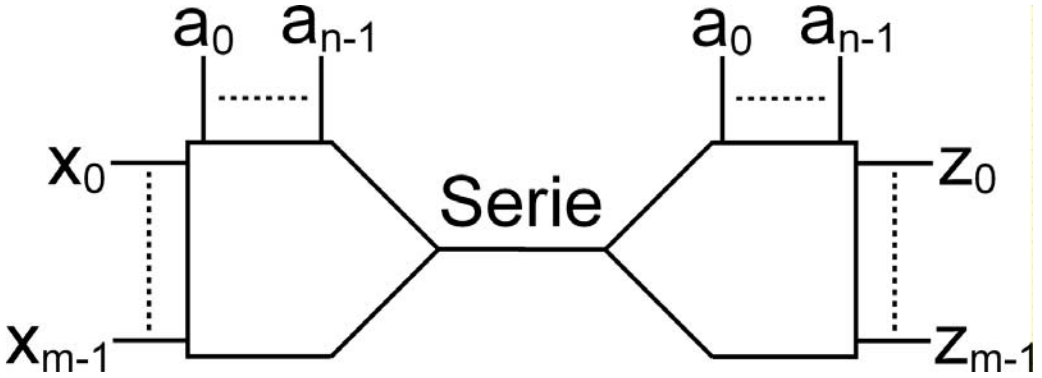
Las variables de control del multiplexor son las variables de la función a sintetizar y las entradas de datos tienen los valores 0 ó 1 correspondientes al valor de la función para cada combinación de variables.

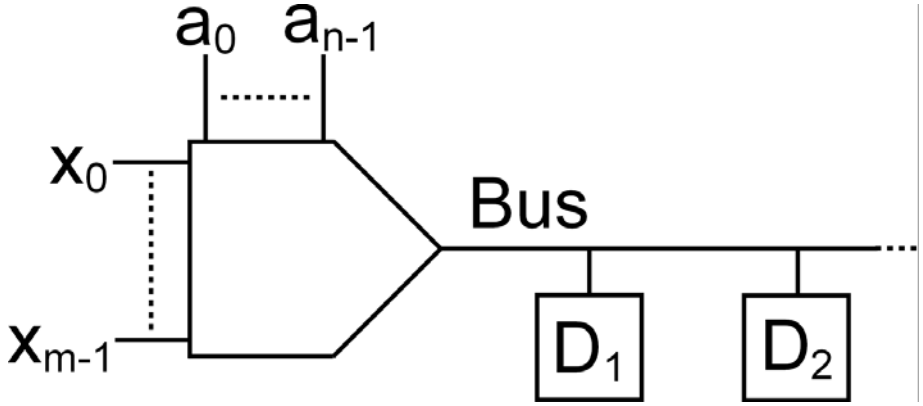


Implementación de la función F con un multiplexor

APLICACIONES DEL MULTIPLEXOR

- Selector de datos. Sirve para convertir información en paralelo en información serie.
- Acceso a buses. El control del acceso a un bus para enviar información al mismo se puede hacer de forma cómoda mediante un multiplexor a través del cual pasen todas las entradas.





TEMA 5.3 – SISTEMAS DIGITALES

- Introducción
- Sistemas combinacionales
- Sistemas secuenciales

BIESTABLES

- Es un circuito capaz de almacenar un bit de información, gracias a un lazo de realimentación
- El más sencillo se basa en dos puertas NOT en serie

- El circuito se puede encontrar en dos situaciones estables (de ahí el nombre de biestable):

Estado 0 o Reset: Q = 0

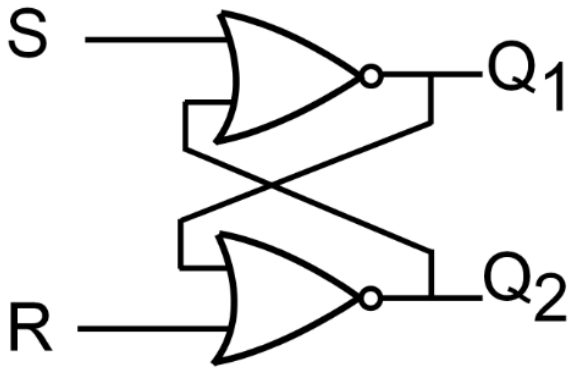
Estado 1 o Set: Q = 1

TIPOS DE BIESTABLES

- Para incluir entradas que nos permitan llevar al biestable a uno de sus estados, vamos a cambiar las puertas NOT por NAND o NOR
- La filosofía es que el valor almacenado (Q= 0 ó 1) se mantiene hasta que las entradas provoquen un cambio.
- Tipos de biestables:
  - Asíncronos (latch SR o D):
    - Q puede cambiar al cambiar cualquier entrada
  - Síncronos (flip-flop D, JK o T):
    - tienen una señal de control (Clock) que indica cuándo pueden cambiar el valor de Q

LATCH SR ASÍNCRONO

- Existe versión basada en dos puertas NOR y en dos puertas NAND. Veamos la NOR:



- Si analizamos el circuito nos encontramos con tres posibilidades para las entradas que determinan las salidas y una posibilidad (S=R=0) para la que las salidas no están influenciadas por las entradas.

Tabla verdad

S	R	Q_2	Q_1
0	0	$\overline{Q_1}$	$\overline{Q_2}$
1	0	1	0
0	1	0	1
1	1	0	0

LATCH SR ASÍNCRONO

- Denominamos
  - Estado 0 cuando $Q_2 = 0$ y $Q_1 = 1$
  - Estado 1 cuando $Q_2 = 1$ y $Q_1 = 0$
- Entonces, el circuito presenta un tercer estado ($Q_1 = Q_2 = 0$). No nos interesa, por lo que evitaremos que el sistema vaya a ese estado. Es el **estado prohibido**.
- Así, las salidas Q_2 y Q_1 son siempre una la negada de la otra, por lo que las denominamos Q y $\overline{Q}$. Y lo que es más importante, la combinación S = R = 0, mantiene el estado por lo que el biestable mantendrá el último estado escrito.

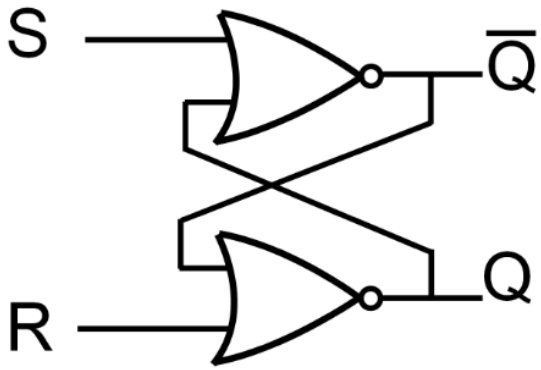


Tabla verdad

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

DINÁMICA

- El sistema permanece estable mientras S=R=0. Se dice que ambas entradas están inactivas.
- Si activamos S (S=1, R=0) el sistema irá al estado 1. Se denomina S porque lleva al estado Set.
- Si activamos R (R=1, S=0) el sistema irá al estado 0. Se denomina R porque lleva al estado Reset.
- El circuito recuerda la última activación en S o R.
- Las dos entradas no pueden activarse a la vez, ya que el sistema iría al estado prohibido.

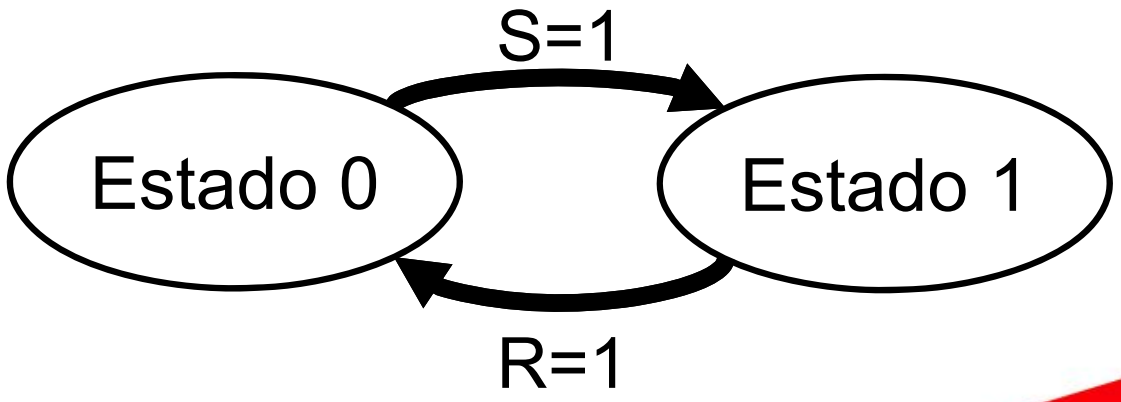
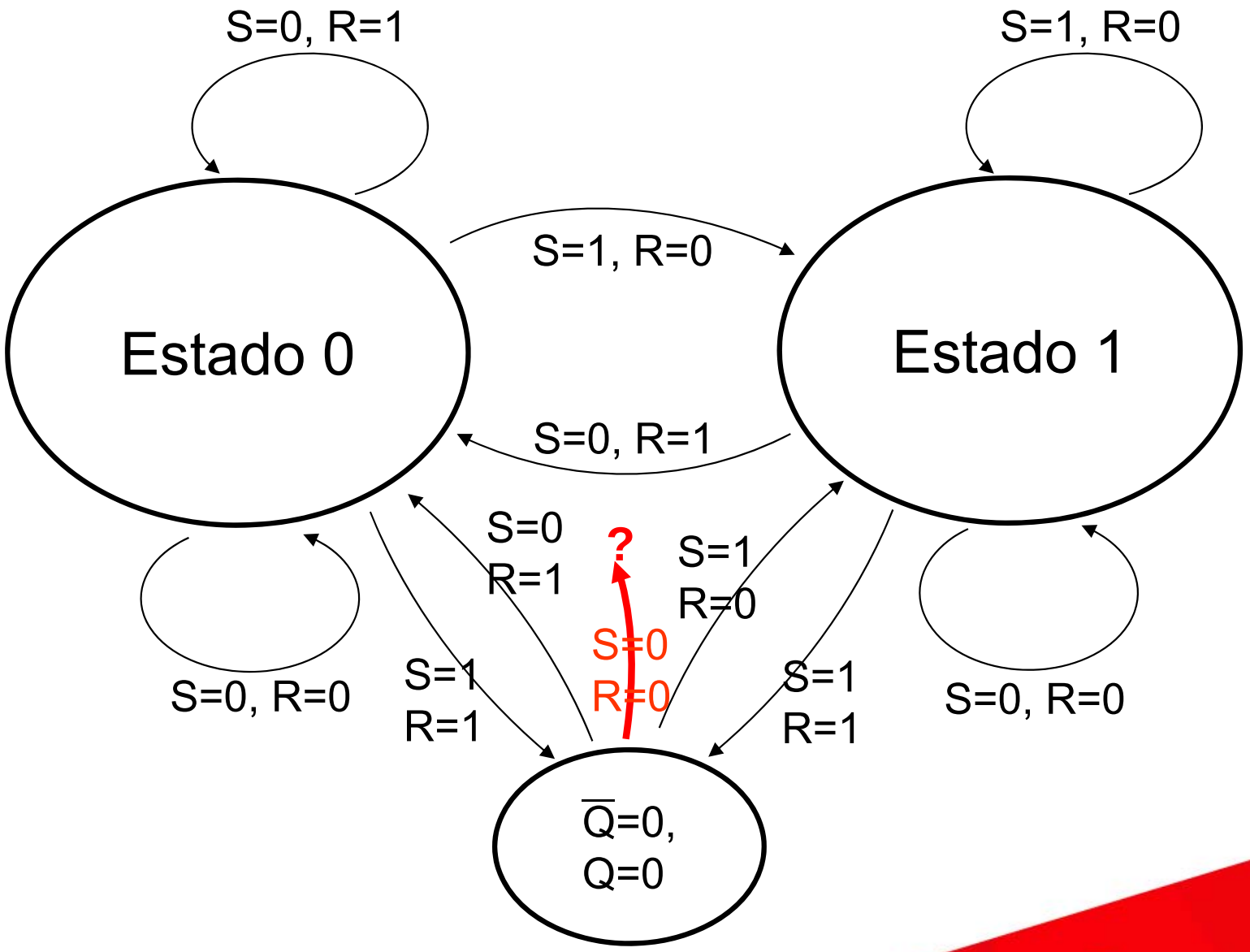
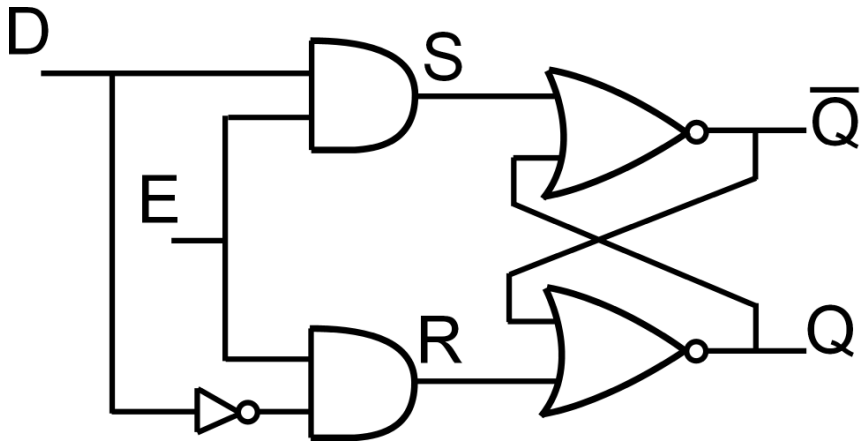


DIAGRAMA DE ESTADOS



LATCH D

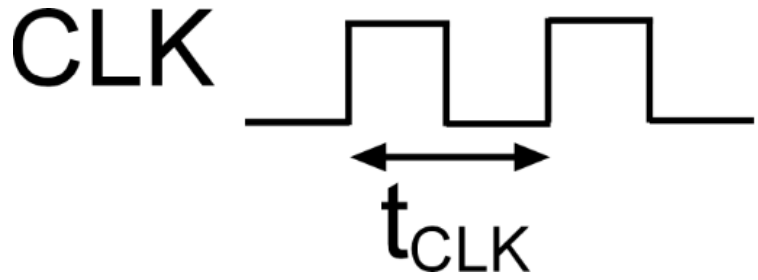
- Está formado por un latch SR y lógica adicional para evitar el estado prohibido
- Por definición, si la señal de control “Enable” (E) está inactiva, el biestable mantiene el estado y si está active el valor de la entrada D se escribe en la salida Q.
- El Enable puede ser activo en bajo o en alto. Si el Enable es activo en alto:



E	D	S	R	Q	$\overline{Q}$
0	-	0	0	Q	$\overline{Q}$
1	1	1	0	1	0
1	0	0	1	0	1

CLOCK (RELOJ)

- Clock: Señal formada por una serie de pulsos intermitentes con un ancho específico.

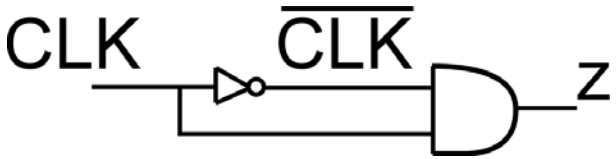


- Tiempo de ciclo del reloj t_{CLK}: intervalo entre los flancos correspondientes de dos pulsos consecutivos.

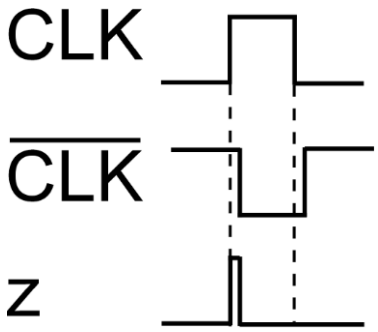
- Para marcar un instante temporal para sincronizar varios biestables, usamos los flancos (de subida o bajada)

DETECTOR DE FLANCO

- Circuito que tiene salida activa (en alto o bajo) solo cuando se produce un flanco (de subida o bajada)
- Versión activa en alto para flanco de subida:

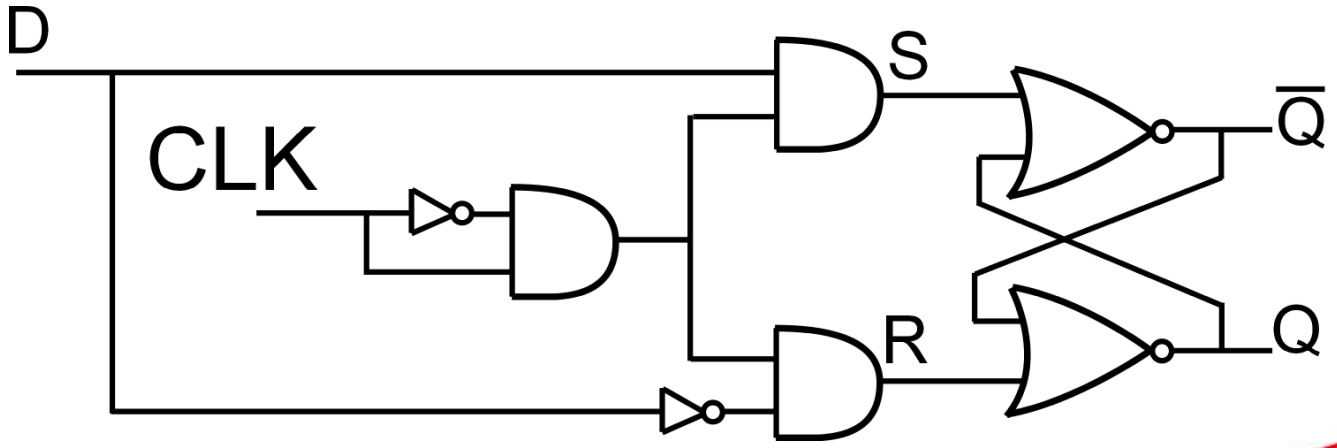


- La salida z será 1 solo cuando se produce el flanco de subida, debido al retardo que se produce en el inversor



FLIP FLOP D

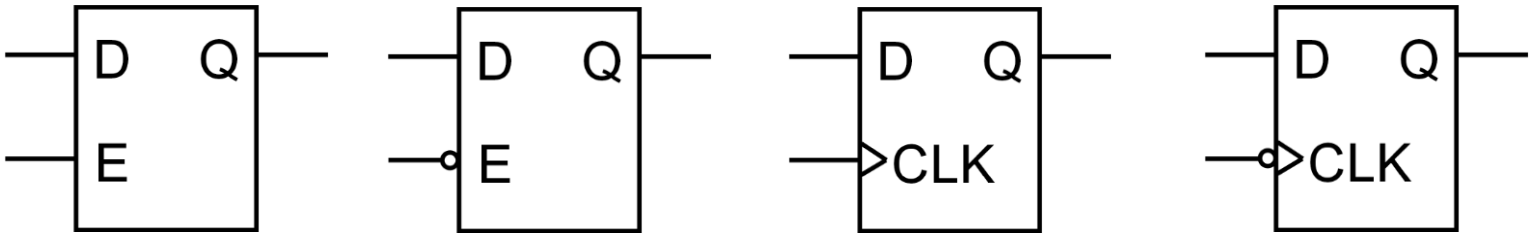
- Formado por un latch D y un detector de flanco
- El biestable solo es activo (Q toma el valor de D), cuando el reloj tiene una transición. Así se pueden sincronizar los cambios de varios flip flop.
- Versión activa con flanco de subida (latch D activo en alto y detector de flanco con salida en alto con flanco de subida de reloj)



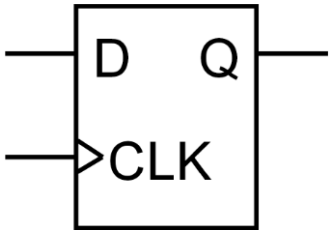
BIESTABLES TIPO D

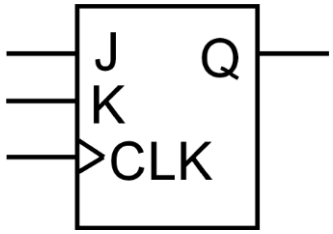
- Un latch es activado por nivel en el sentido de que se pueden producir transiciones de estado mientras la señal de control (enable) esté en nivel 1 (o 0) (level triggered)
- Un flip-flop es activado por flanco de subida en el sentido de que la transición de estado se produce sólo cuando la señal de control (reloj) pasa de 0 a 1 (o de 1 a 0) (edge triggered)

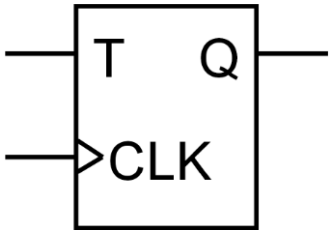
- Latch D activado con nivel alto (a) o con nivel bajo (b)
- Flip-Flop D activado con flanco de subida (c) o de bajada (d)



TIPOS DE FLIP FLOP: D, JK Y T







CLK	D	Q	$\overline{Q}$
0	-	Q	$\overline{Q}$
1	-	Q	$\overline{Q}$
↓	-	Q	$\overline{Q}$
↑	0	0	1
↑	1	1	0

CLK	J	K	Q	$\overline{Q}$
0	-	-	Q	$\overline{Q}$
1	-	-	Q	$\overline{Q}$
↓	-	-	Q	$\overline{Q}$
↑	0	0	Q	$\overline{Q}$
↑	1	0	1	0
↑	0	1	0	1
↑	1	1	$\overline{Q}$	Q

CLK	T	Q	$\overline{Q}$
0	-	Q	$\overline{Q}$
1	-	Q	$\overline{Q}$
↓	-	Q	$\overline{Q}$
↑	0	Q	$\overline{Q}$
↑	1	$\overline{Q}$	Q

AGRUPACIONES DE FLIP FLOPS

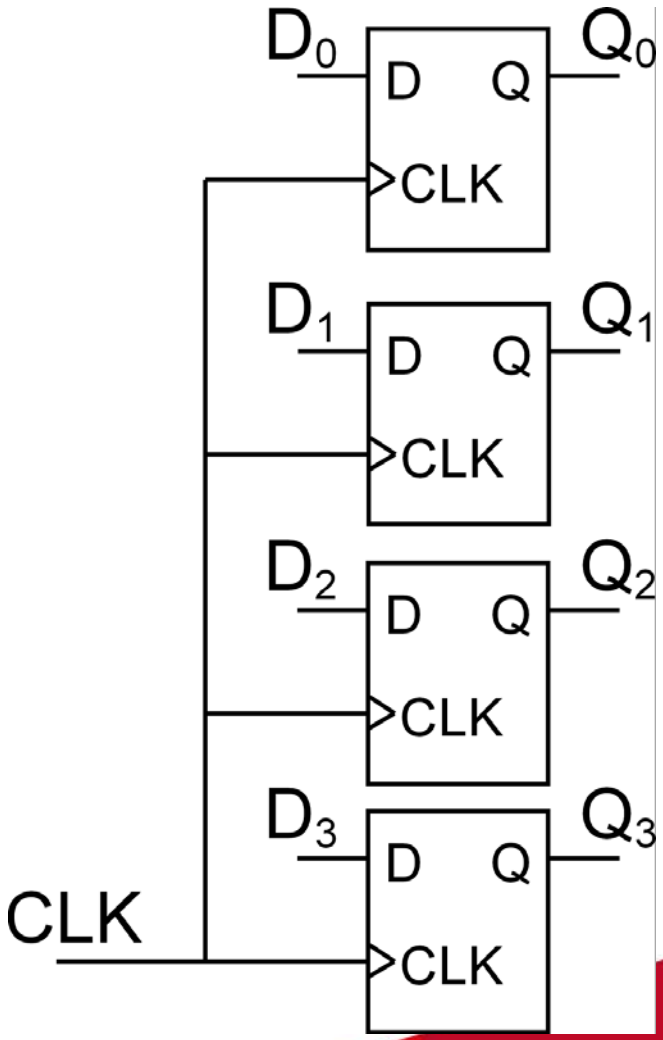
- Flip flops D en paralelo: Registros de almacenamiento
- Flip flops D en serie: Registros de desplazamiento

- Flip flops T en serie: contador asíncrono
- Flip flops T en paralelo: contador síncrono

- Matriz de flip flops D: Memoria SRAM

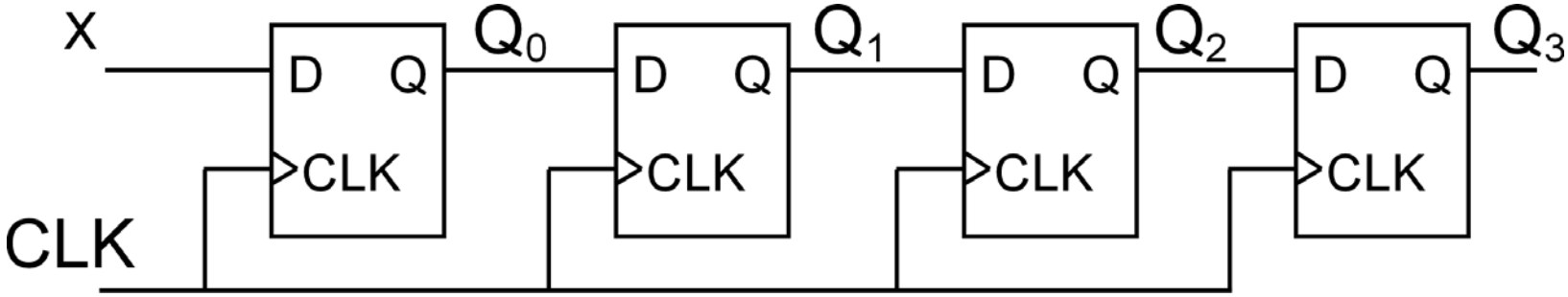
REGISTRO DE ALMACENAMIENTO

- Conexión en paralelo de flip-flops D
- Se produce una carga síncrona de los datos D_i en la salida correspondiente Q_i
- El dato queda almacenado hasta la siguiente activación del reloj



REGISTRO DE DESPLAZAMIENTO

- Conexión en serie de flip-flops D



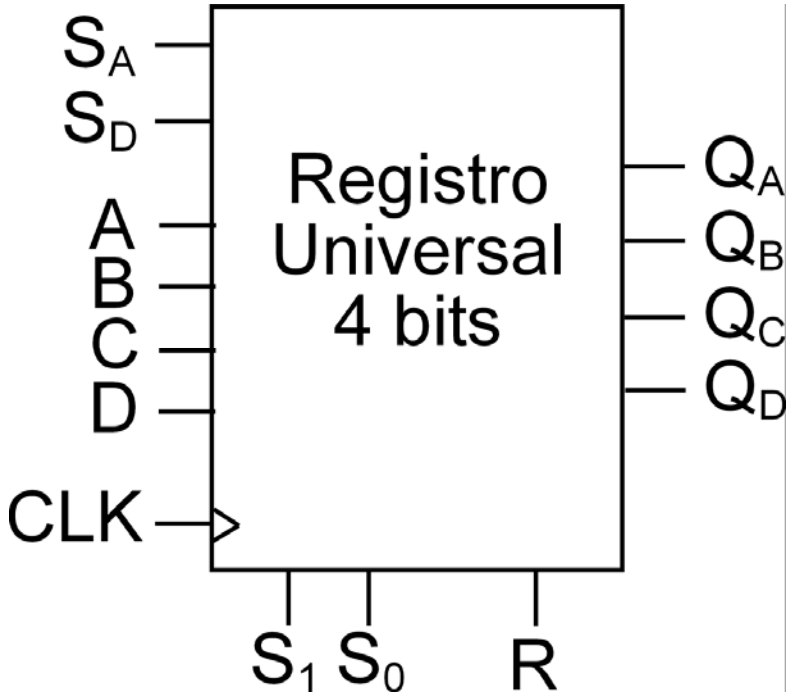
- En cada activación del reloj se produce un desplazamiento de los datos hacia la derecha, ya que cada biestable toma el valor que tiene en su entrada

$$Q_{i+1} = Q_i \ (Q_0 = X)$$

- No se produce un desplazamiento en cadena porque el tiempo de retardo de cada biestable es superior al de activación del reloj

REGISTRO UNIVERSAL

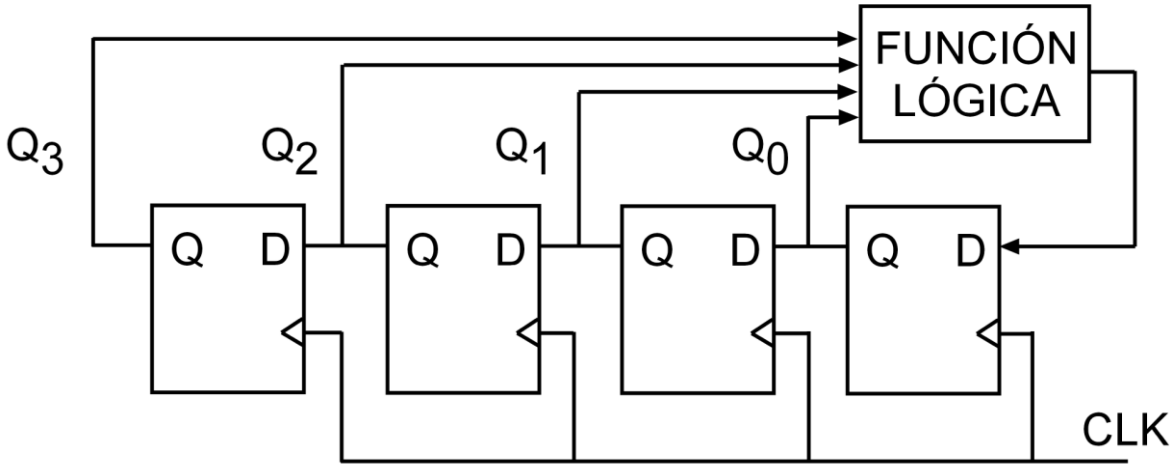
- Integrado 74194: Reg. Universal de 4 bits
- Desplazamiento bidireccional y carga de datos en paralelo síncronos
- Reset asíncrono
- Selección de operación: S_1, S_0



R	S_1	S_0	CLK	Q_A	Q_B	Q_C	Q_D	
0	-	-	-	0	0	0	0	Reset
1	1	1	↑	A	B	C	D	Carga en paralelo
1	0	1	↑	S_A	Q_A	Q_B	Q_C	Desplazamiento a la derecha
1	1	0	↑	Q_B	Q_C	Q_D	S_D	Desplazamiento a la izquierda
1	0	0	-	Q_A	Q_B	Q_C	Q_D	Sin cambio

REGISTROS CON REALIMENTACIÓN

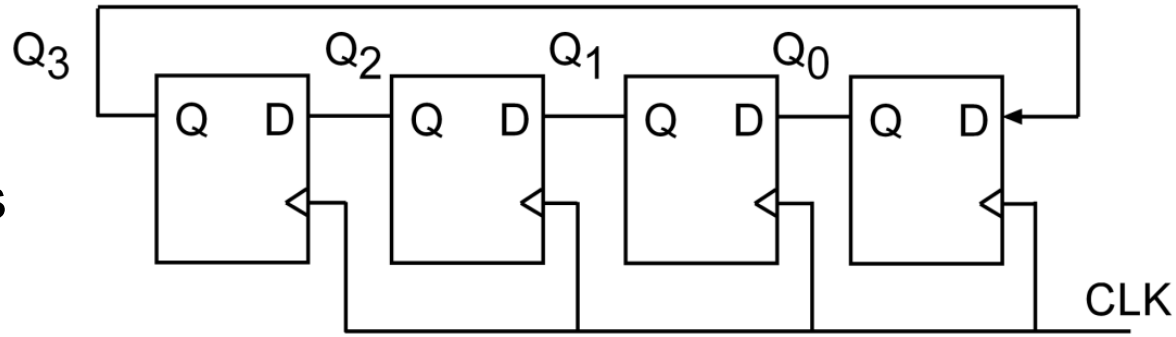
- Registro de desplazamiento (a la izquierda) realimentado

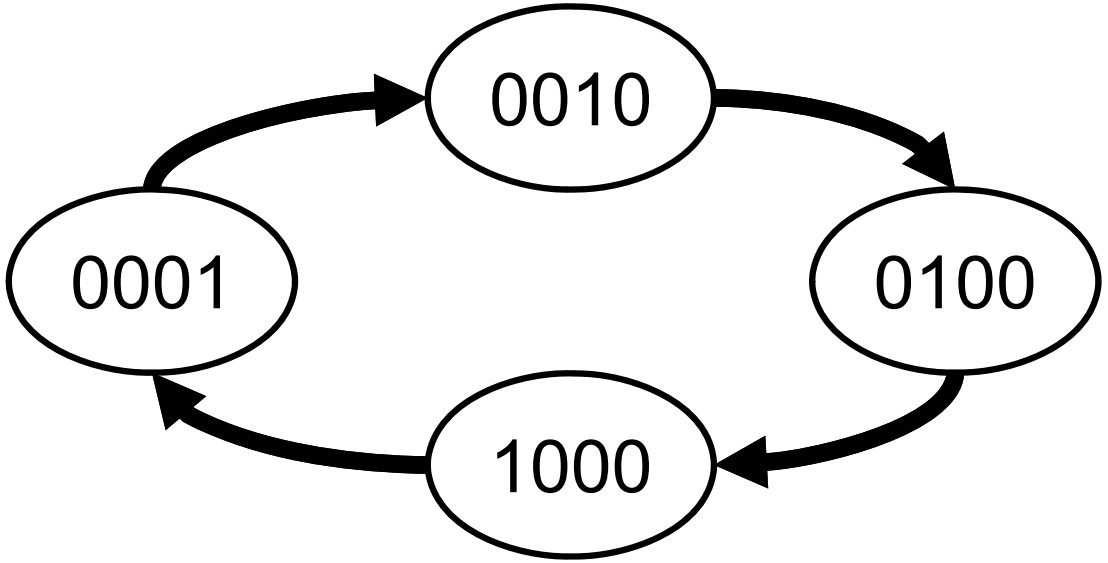


- Estado definido por el valor de las salidas: $Q_3Q_2Q_1Q_0$
- Número de estados limitado ($\leq 2^N$) que se repiten cíclicamente
- Suelen requerir inicialización (o corrección de estados no permitidos)
- Ejemplo: contadores de anillo y doble anillo

CONTADOR EN ANILLO

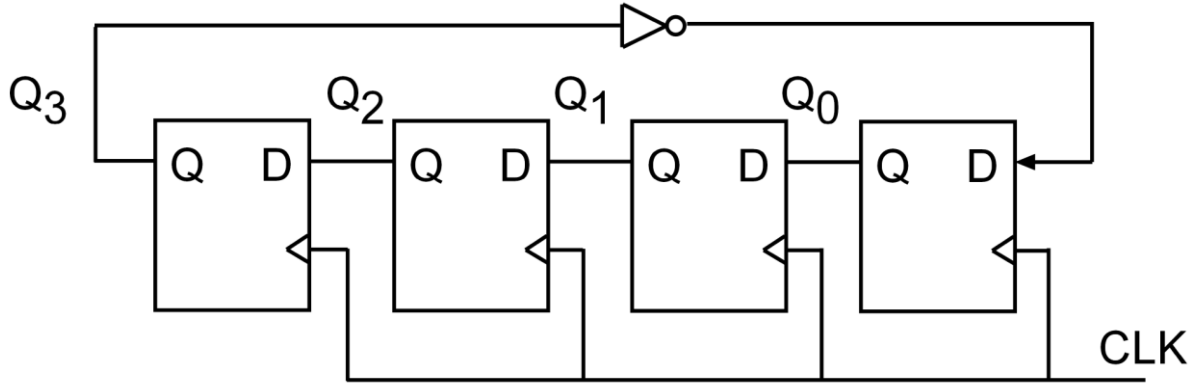
- Contador de anillo
  - Salida serie → Entrada serie
  - N estados permitidos

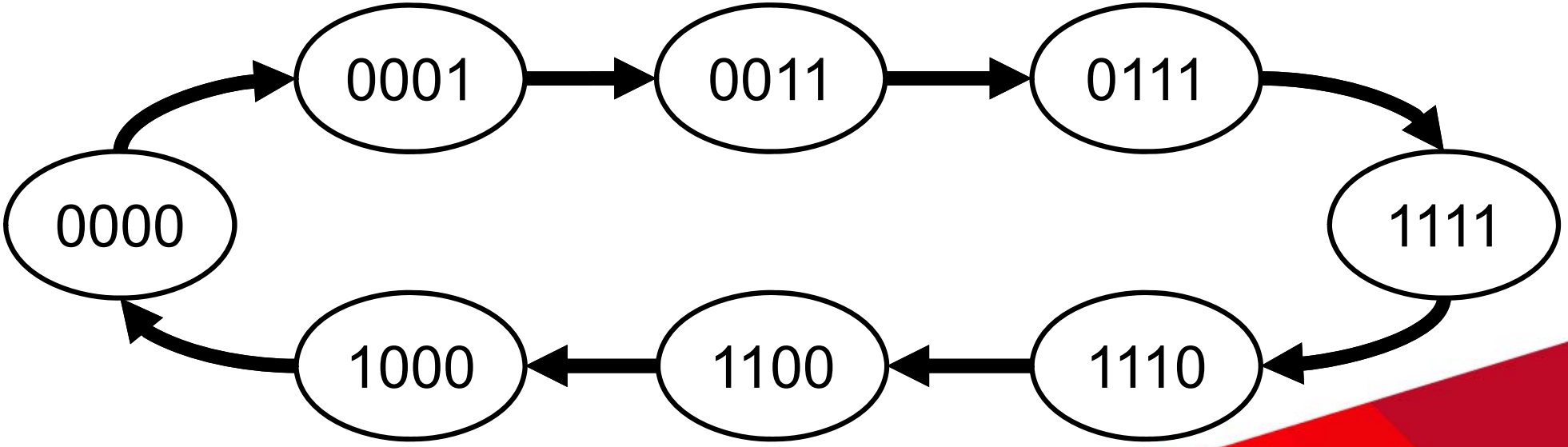




CONTADOR DE DOBLE ANILLO

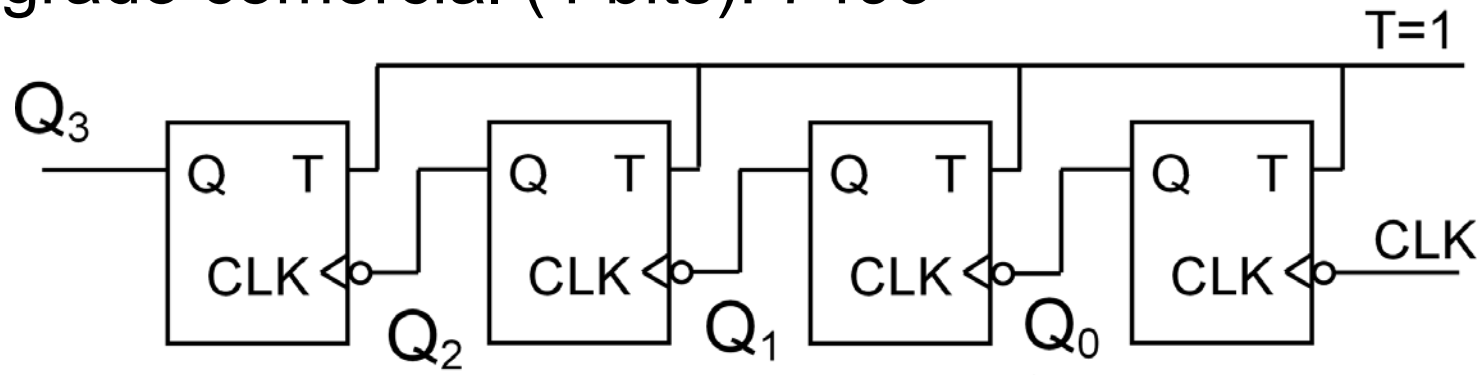
- Contador de doble anillo
  - Salida serie negada → Entrada serie
  - 2N estados permitidos

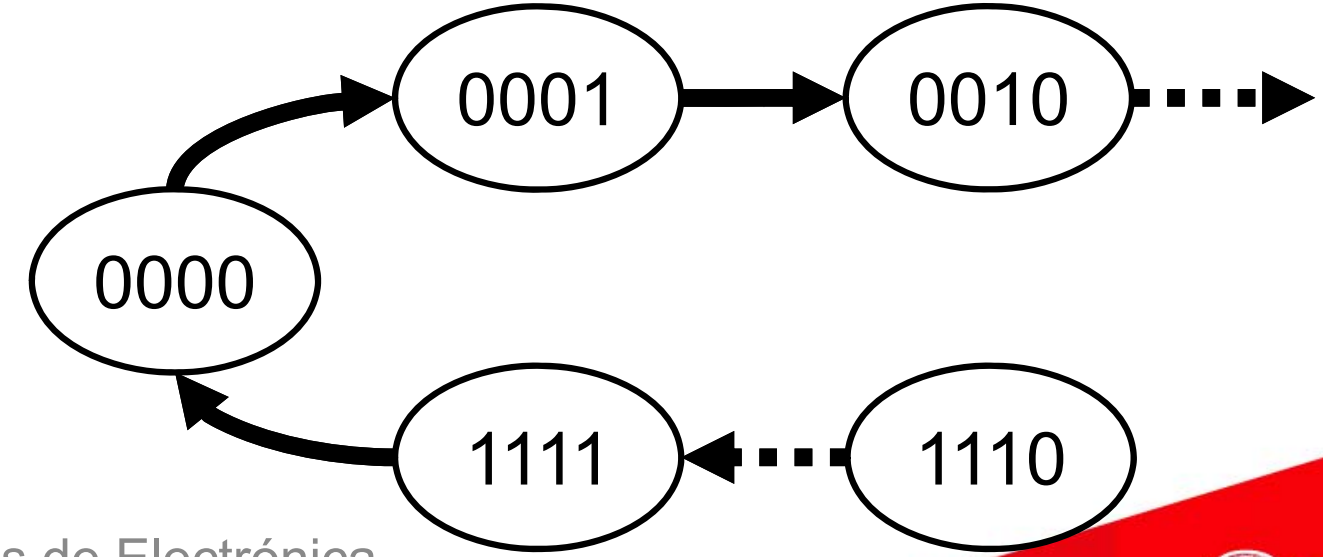




CONTADORES ASÍNCRONOS

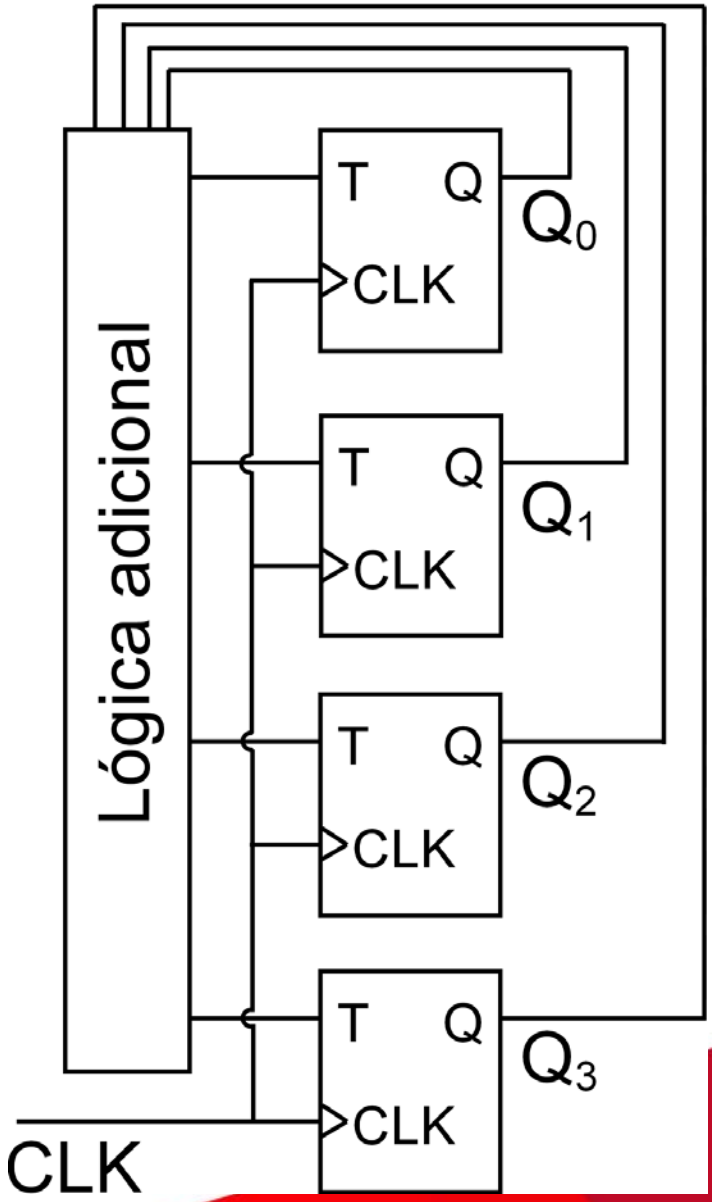
- Conexión “serie” de flip-flops T (todos activos con T=1)
- Reloj (activo en flanco de bajada) no común: **Asíncrono**
- Integrado comercial (4 bits): 7493





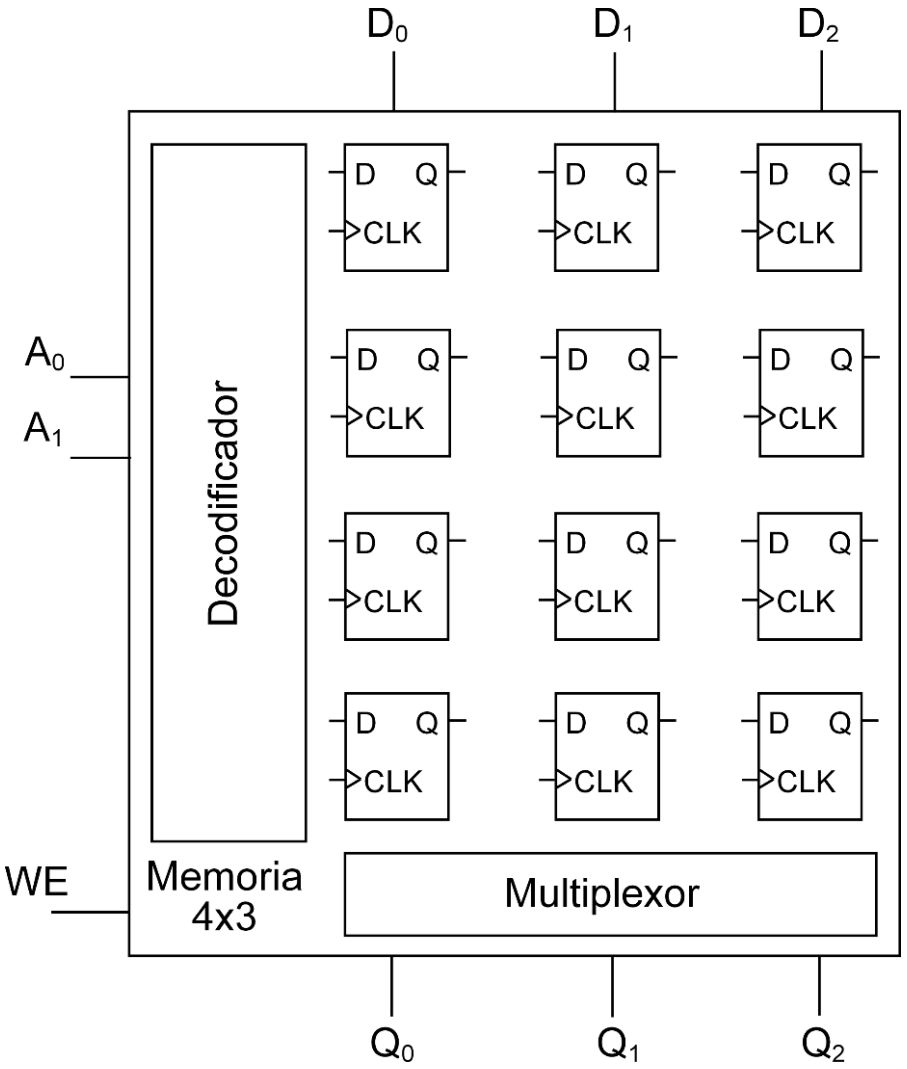
CONTADORES SÍNCRONOS

- Conexión “paralelo” de flip-flops T
- Requiere lógica adicional (mayor complejidad)
- Retardo no acumulativo (mayor velocidad)
- Reloj común: **síncrono**
- Integrado comercial (4 bits): 74163



MEMORIA SRAM

- Matriz m x n de Flip Flop D + Decodificador + Multiplexor
- Ejemplo 4x3:
  - 3 datos de entrada (D)
  - 2 para la dirección (A): el decodificador indica una única fila
  - 3 datos de salida (Q): el multiplexor lleva a la salida la fila indicada por el decodificador
  - 1 control de escritura (WE: write enable)
    - WE = 0: lectura
    - WE = 1: escritura



REFERENCIA

- Open Course Ware (Universidad Carlos III de Madrid)
  - http://ocw.uc3m.es/tecnologia-electronica/electronica-digital