66.70 Estructura del Computador

Circuitos secuenciales básicos:

Registros y contadores

Registros

Se caracterizan por:

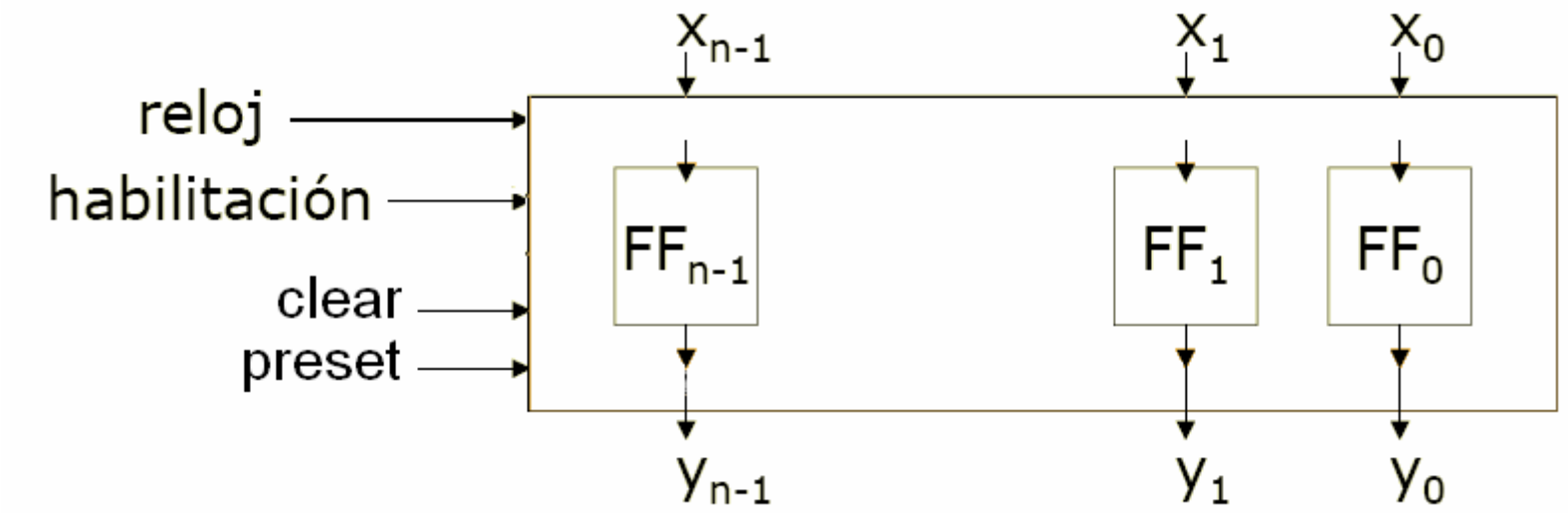
- Capacidad para guardar información
- Registro de **n** bits => ***n*** *FlipFlops operando sincrónicamente (igual clock)*
  + lógica de control (compuertas)

Aplicaciones:

- Almacenamiento (memoria)
- Desplazamiento (fines lógicos o aritméticos) (conversión de datos)

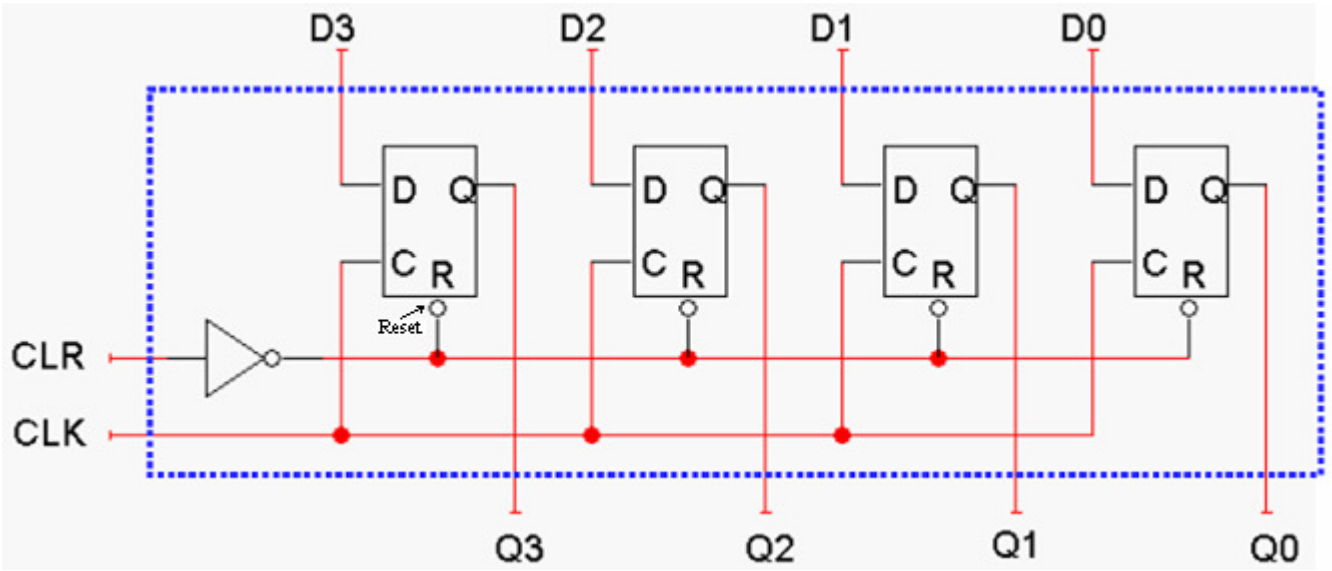
Entrada	Salida	Aplicación
Serie	Serie	Almacenamiento
Serie	Paralelo	Conversión
Paralelo	Serie	Conversión
Paralelo	Paralelo	Almacenamiento

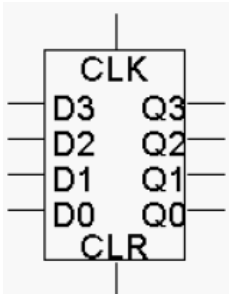
Registros



*Registro paralelo-paralelo de **n** bits*

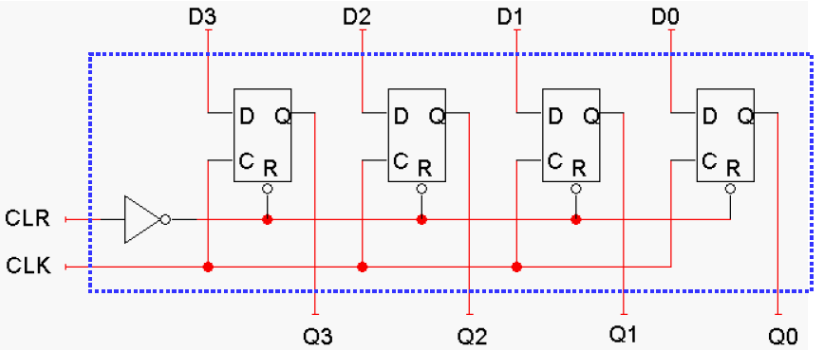
Registro paralelo-paralelo de 4 bits





- Con cada ciclo de reloj la entrada de 4 bits es copiada en la salida

Registro paralelo-paralelo de 4 bits



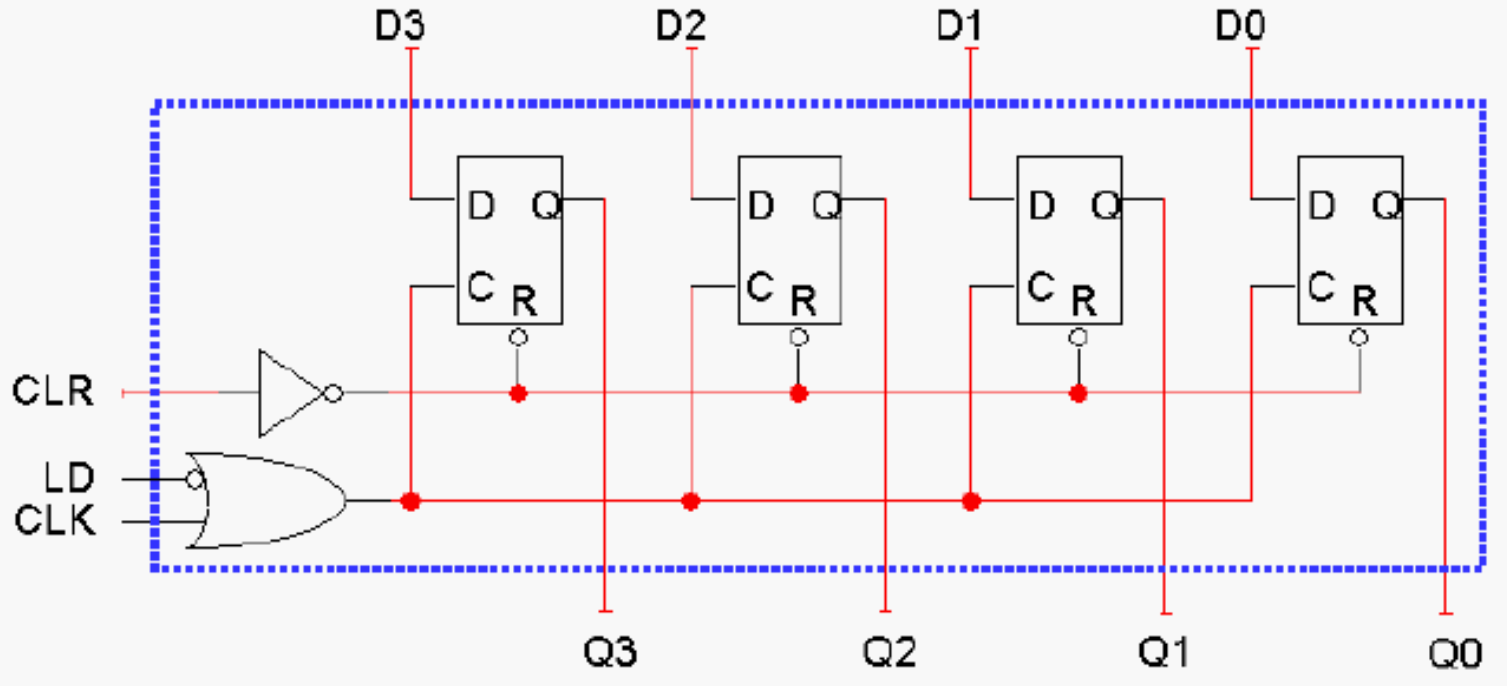
Cómo almacenar datos que duren más tiempo que 1 ciclo de reloj?

Agregamos una entrada para controlar la carga del registro (LD)
- Si LD = 0 el registro mantiene su valor actual
- Si LD = 1 el registro guarda el valor presenta a su entrada de datos (D0-D3)

Técnicas para implementarlo:

- ✓ Controlar la entrada de reloj (clock gating)
- ✓ Mantener sin cambios la entrada de datos a los FF

Clock gating



Modifica la entrada de clock de cada FF

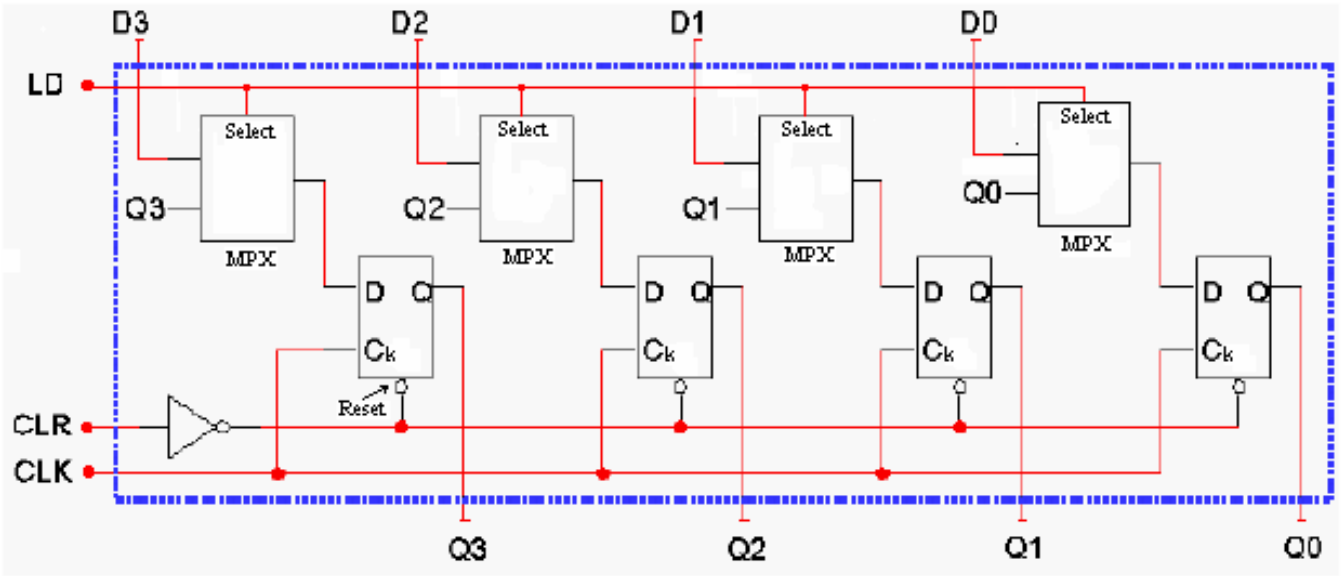
- LD=0 previene la aparición de flancos de reloj a la entrada de los FF
- LD=1 el reloj es aplicado a todos los FlipFlops

Inconveniente: En un sistema sincrónico (reloj general) dispositivos de este tipo no responden en forma sincrónica (retardos diferentes)

"Clock skew"

Otra solución: actuar sobre las entradas de los FF

Utiliza "multiplexores" ⟶ ¿?



MPX=multiplexor

- LD selecciona cuál es la entrada de cada FF
  - LD=0 la entrada D es la salida Q del mismo FF
  - LD=1 la entrada D es el nuevo dato

Aplicaciones de los
Registros de almacenamiento

Los registros almacenan datos en un microprocesador

- ✓ Se usan para presentar los operadores a la Unid. Aritmético Lógica (ALU)
- ✓ Se usan para guardar los resultados de operaciones aritméticas-lógicas

Permiten guardar cualquier información en formato binario, entonces:

> Registros y la memoria en una computadora

Aplicaciones de los
Registros de almacenamiento

Los registros almacenan datos en un microprocesador

- ✓ Se usan para presentar los operadores a la Unid. Aritmético Lógica (ALU)
- ✓ Se usan para guardar los resultados de operaciones aritméticas-lógicas

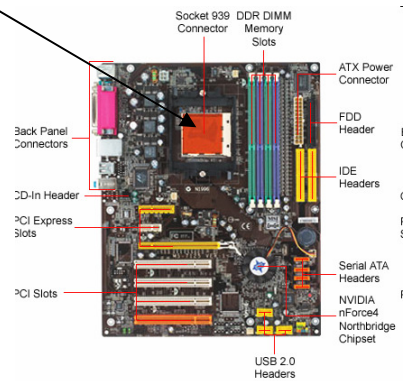
Permiten guardar cualquier información en formato binario, entonces:

> Registros y la memoria en una computadora

GPR= Reg de Propósito General

CPU	GPR's	Size	L1 Cache	L2 Cache
Pentium 4	8	32 bits	8 KB	512 KB
Athlon XP	8	32 bits	64 KB	512 KB
Athlon 64	16	64 bits	64 KB	1024 KB
PowerPC 970 (G5)	32	64 bits	64 KB	512 KB

Los registros ocupan la parte más cara del motherboard



Cache L1 y L2 son RAM muy rápida,
pero no tan rápida como los registros

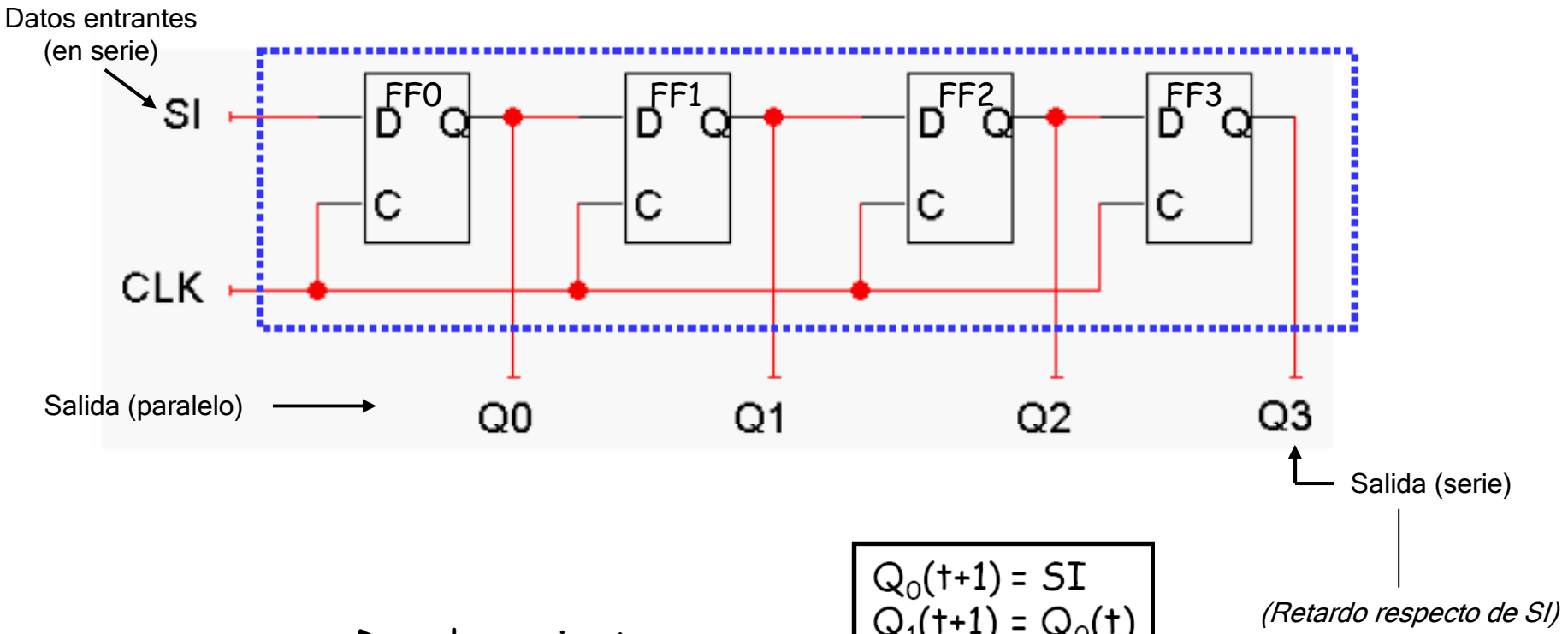
› Desventaja de los registros: son caros!
› La mayoría de los programas requieren más memoria de la que los registros pueden proveer => necesitamos RAM

Registros de desplazamiento

Entrada	Salida	Aplicación
Serie	Serie	Almacenamiento
Serie	Paralelo	Conversión
Paralelo	Serie	Conversión
Paralelo	Paralelo	Almacenamiento

Registros de desplazamiento

Carga de datos en serie

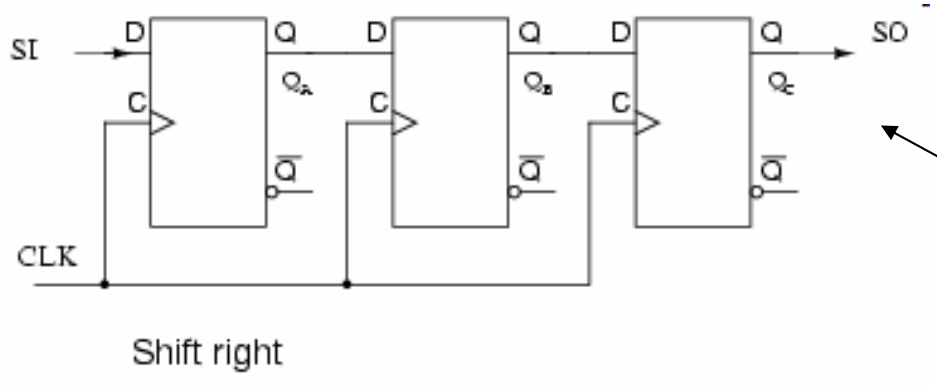


Desplazamiento →

$Q_0(t+1) = SI$
$Q_1(t+1) = Q_0(t)$
$Q_2(t+1) = Q_1(t)$
$Q_3(t+1) = Q_2(t)$

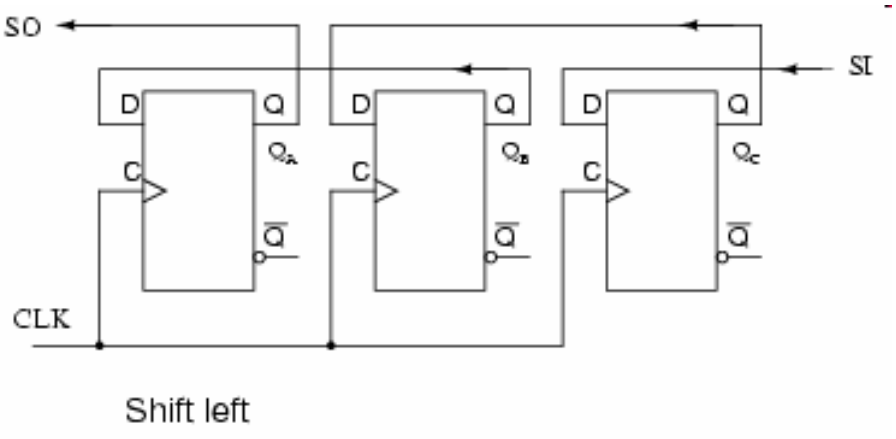
Registros de desplazamiento

Dirección del desplazamiento fija (cableada)



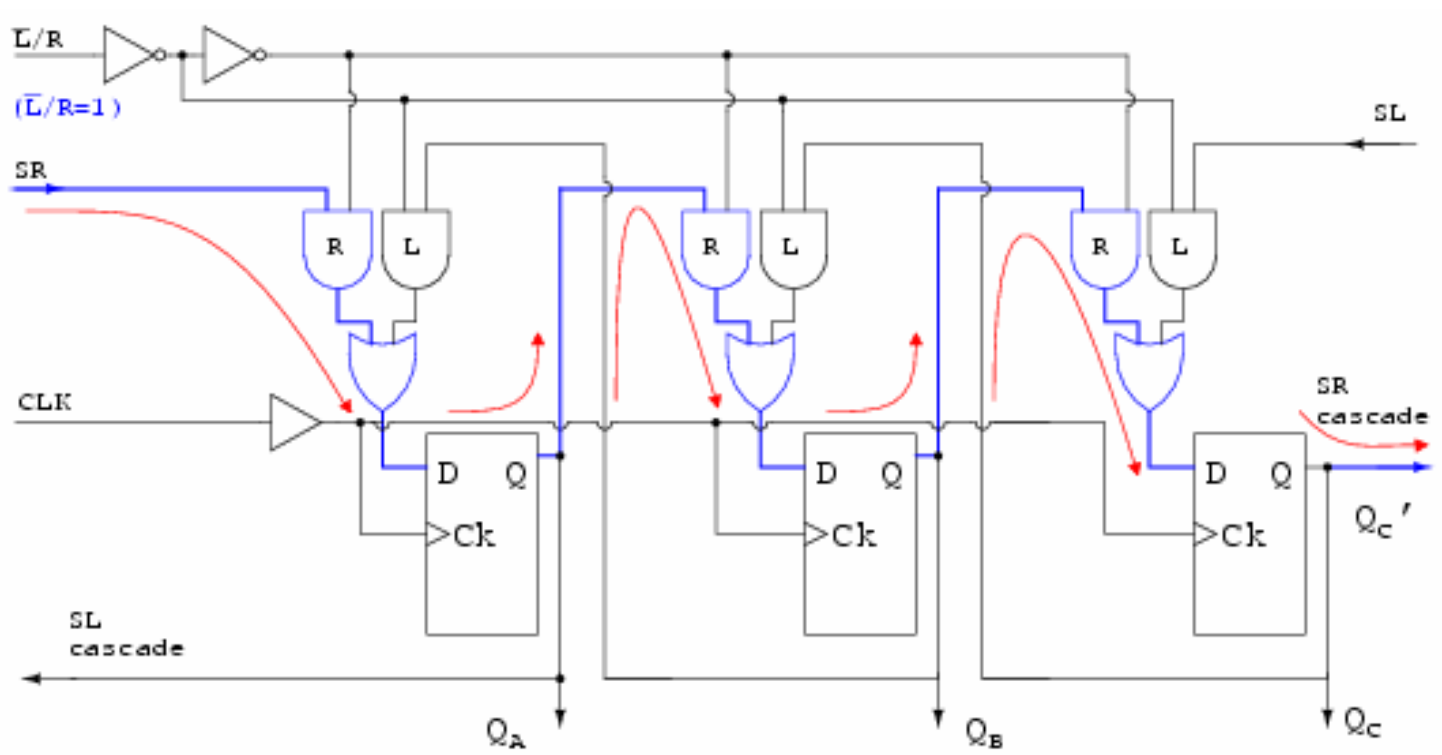
Multiplica o divide?

Multiplica o divide?



Registros de desplazamiento

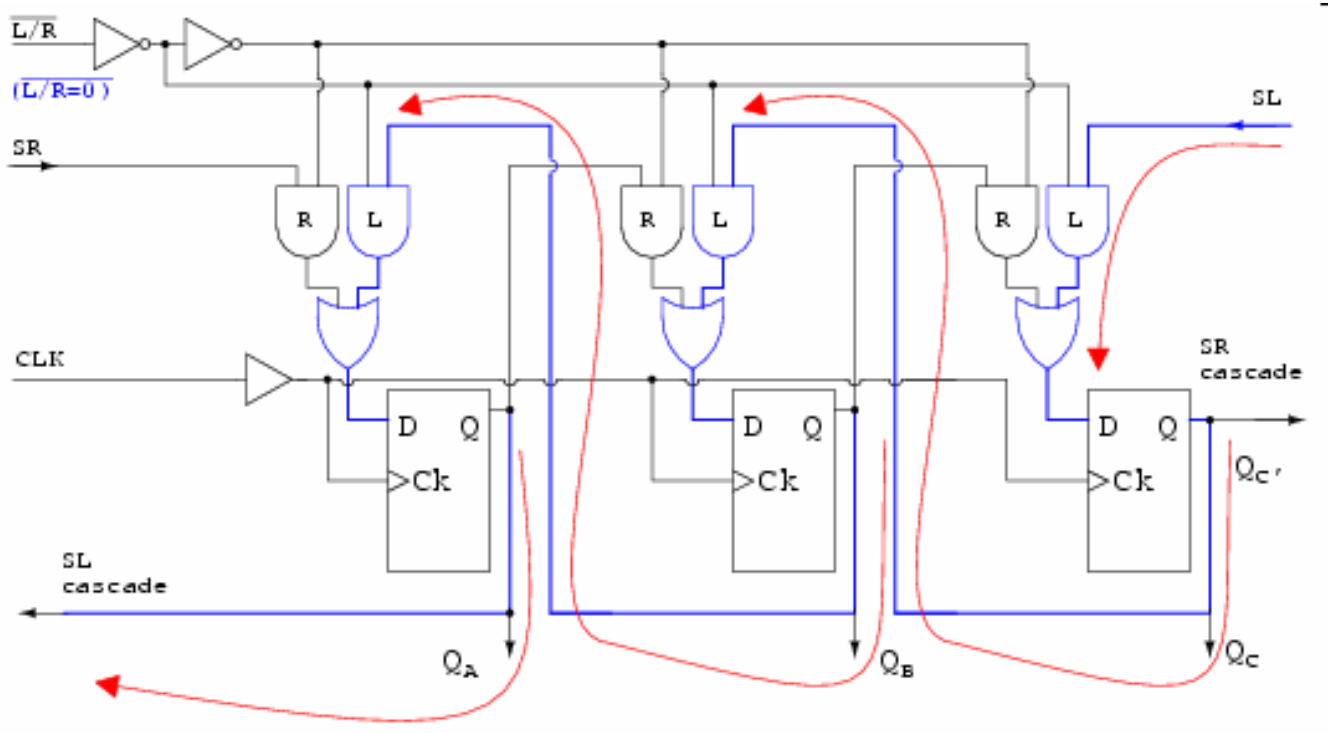
Control de la dirección del desplazamiento



Desplazamiento a derecha

Registros de desplazamiento

Control de la dirección del desplazamiento

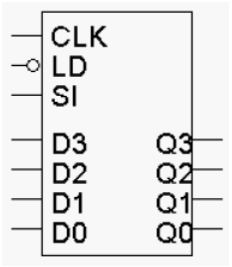


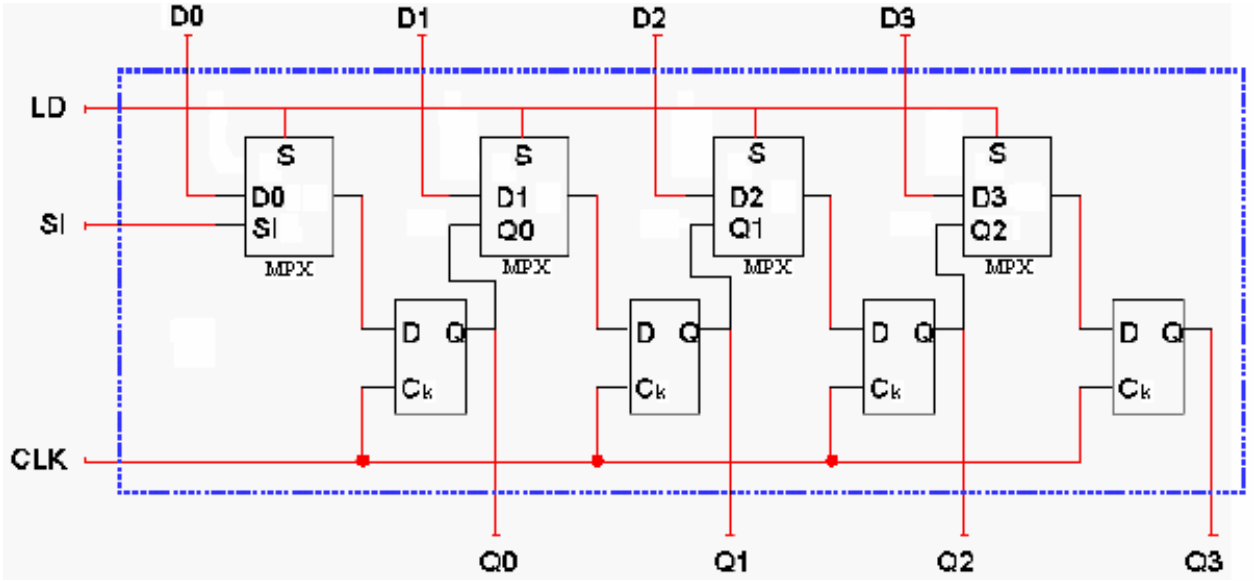
Desplazamiento a izquierda

Registros de desplazamiento

Carga de datos en paralelo

SI: bit de entrada serie
LD= 0 carga en paralelo
LD=1 desplaza





Conversión serie/paralelo
con registros de desplazamiento

- **Las computadoras trabajan con palabras (conjunto de bits)**
  - Números: Enteros, punto flotante (32 bits, 64 bits)
  - Caracteres ASCII (8 bits)

- **La transmisión muchas veces debe hacerse de a un bit por vez**
  - Mouse y teclado
  - Impresoras
  - Puertos serie, USB o Firewire.
  - Discos rígidos: Serial ATA

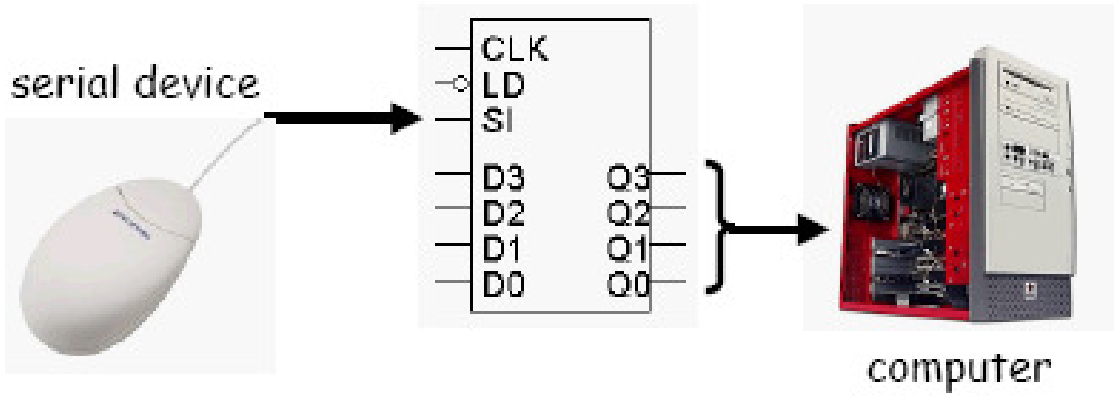
Recepción de datos desde un dispositivo serie

- Dispositivo serie conectado a la entrada SI
- Salidas Q3-Q0 conectadas a la computadora

✓ El dispositivo serie transmite 1 bit por ciclo de reloj

✓ Luego de 4 ciclos de reloj una palabra de 4 bits completa esta disponible.

✓ En un solo paso la computadora lee Q3-Q0 a la salida del Reg. de Despl.



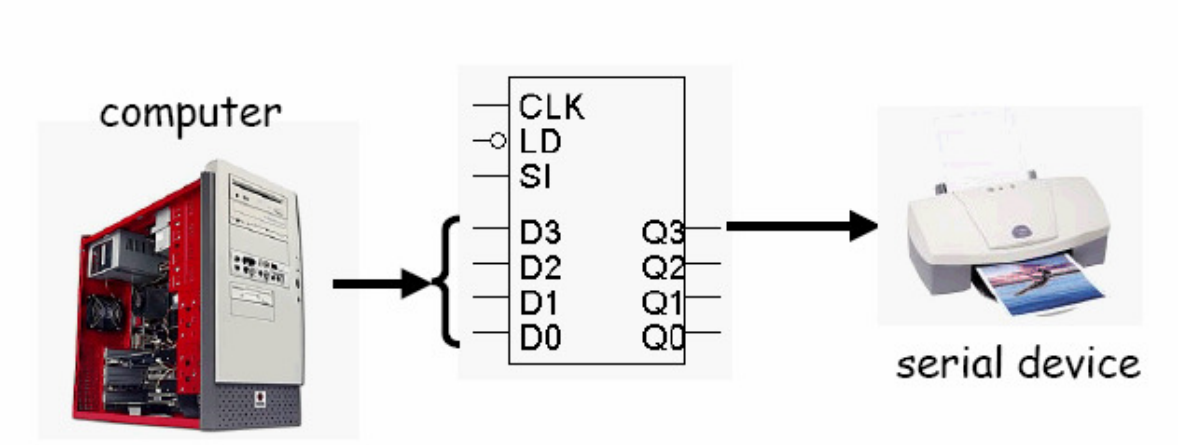
Envío de datos a un dispositivo serie

- La CPU conectada a las entradas D del registro
- La salida SO (Q3) conectada al dispositivo serie

✓ En un solo ciclo de reloj la computadora almacena los 4 bits en el reg.

✓ En cada uno de los ciclos de reloj siguientes el dispositivo lee la salida serie

✓ Después de 4 ciclos la palabra de 4 bits fue transmitida



Contadores

- El igual que en los registros las salidas Q (estado) son las salidas del contador
- Con cada ciclo de reloj la salida (n bits) del contador se incrementa en 1
- Vuelve al primer estado una vez agotada la capacidad de cuenta
- A diferencia de los registros, la secuencia de los estados recorridos es fija

Estado actual	Próximo estado
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	0

Cantidad de estado <=> Cantidad de bits

Cantidad de bits <=> Cantidad de flipflops

Para qué necesitamos contadores?

- ✓ Contar cantidad de veces que algún evento ha ocurrido
- ✓ Medir tiempo
- ✓ Contar la cantidad de bits que fueron enviados o recibidos desde o hacia un dispositivo
- ✓ Todos los procesadores tienen un PC (Program Counter)
  - – Los programas consisten en una lista de instrucciones que se ejecutan secuencialmente (en general)
  - – El PC lleva la cuenta de cual es la instrucción actualmente en ejecución
  - – El PC se incrementa una vez por cada ciclo de reloj indicando la próxima instrucción a ser ejecutada
- ✓ … y más

Descripción formal de un contador

- Tabla de estados
- Diagrama de estados
- Módulo y código de cuenta
- Diagrama de tiempos

"Analizar" un contador = a partir del circuito determinar todo lo anterior

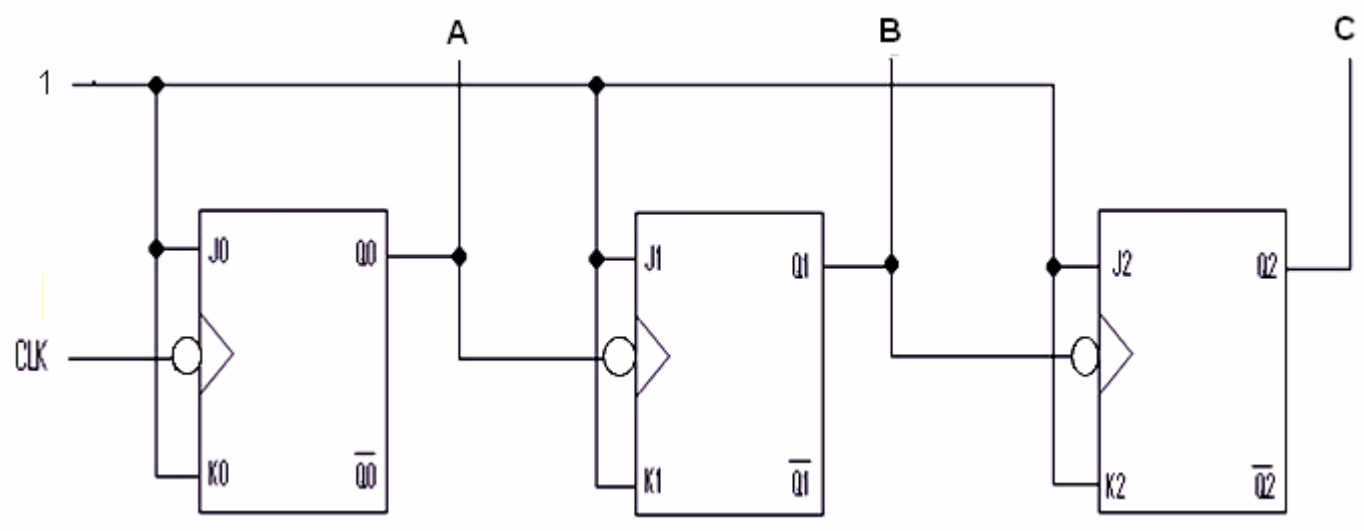
Algunas **clasificaciones**

- Sincrónicos y asincrónicos
- Módulo 2^n y de módulo distinto de 2^n *(**n** = cantidad de FF)*
- Sincrónicos de transporte serie, transporte paralelo o mixto
- De cuenta ascendente y de cuenta descendente
- Contadores en anillo y de cadena abierta

Algunas **definiciones**

- módulo
- código
- estados prohibidos
- secuencia cerrada
- secuencia prohibida

Contador **a**sincrónico de 3 bits

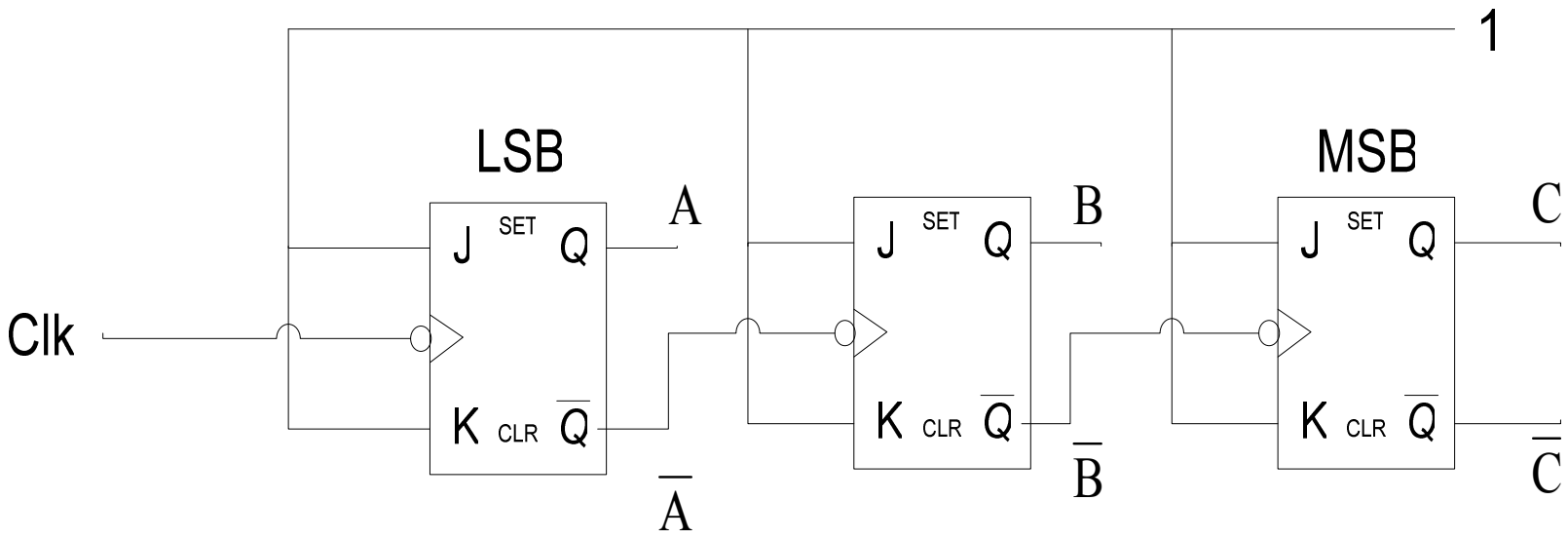


… Analizar este contador

… Estudiar como convertirlo en un contador de módulo 6

… Como sería un contador de décadas (módulo 10) asincrónico?

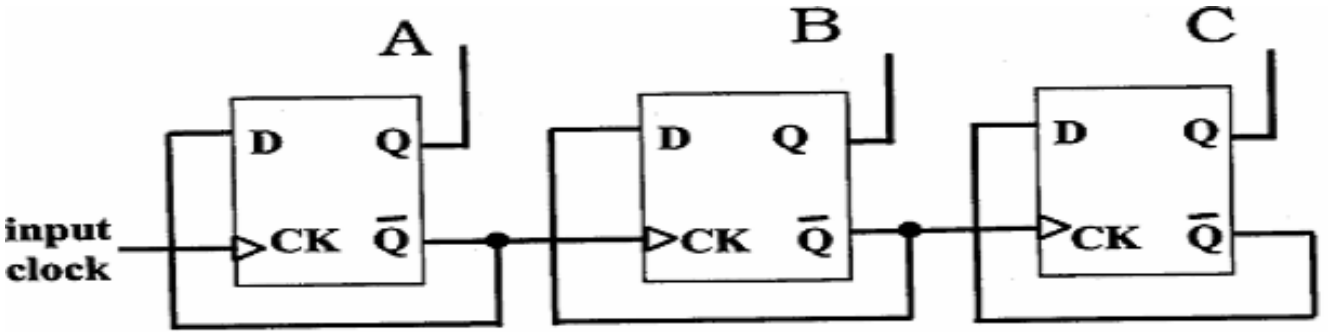
Otro contador **a**sincrónico de 3 bits



… Determinar tabla de estados, diagr..de estados, módulo y código de cuenta

Otro contador **a**sincrónico de 3 bits

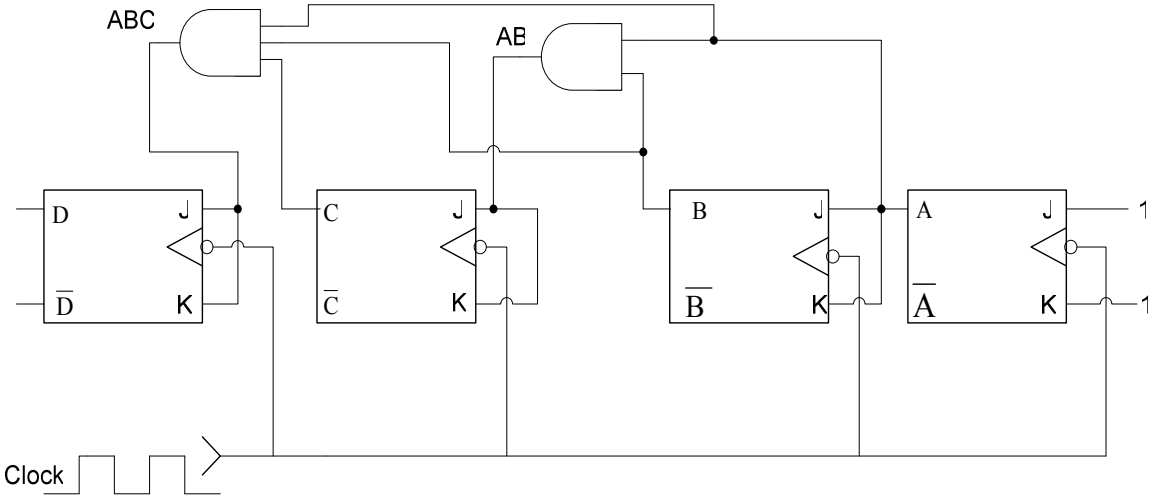
Ahora con FF tipo D



… Determinar tabla de estados, diagr. de estados, módulo y código de cuenta

... Qué tipos de FF pueden utilizarse para construir contadores?

Contadores sincrónicos
contador de 4 bits



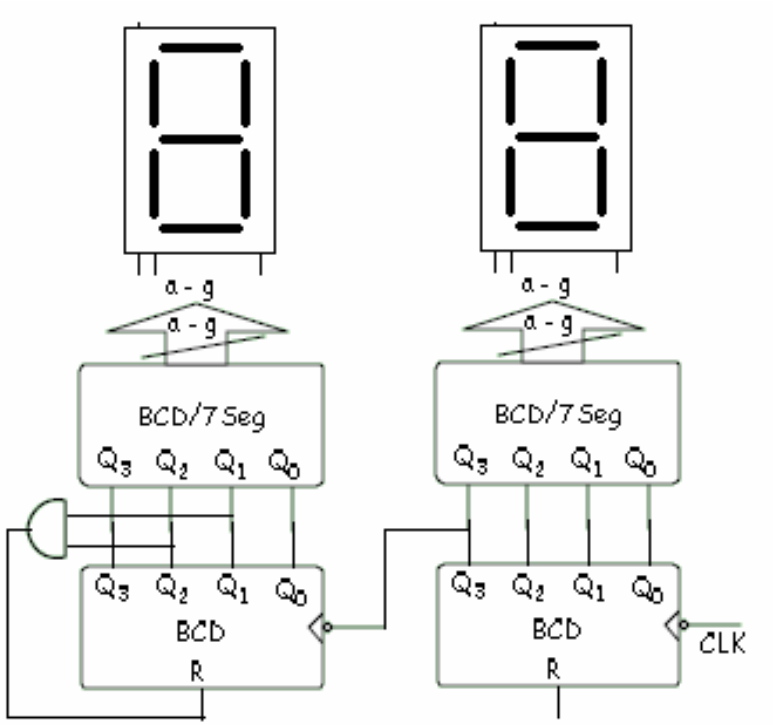
… *Analizar*

… *Estimar frecuencia máxima con retardos dados: FFs 22 ηseg, ANDs: 8 ηseg*

… *Comparar con la velocidad esperable de un contador asincrónico similar*

Ejemplo de aplicación:

Segundero Digital



- *Dar las características requeridas para los contadores*
- *Cuál es la función de la compuerta AND?*
- *Cómo conectar a un módulo de horas?*

Diseño
de contadores sincrónicos

1) Elijo el módulo y el código de cuenta (=> diagr. de estados)
2) Módulo => cantidad de FlipFlops necesarios
3) Elegir un tipo dado de FF
4) Diseñar la lógica de compuertas en base al código de cuenta

El contador puede hacerse con cualquier tipo de FF (D, T, JK) pero la lógica de compuertas será diferente en cada caso

Para este tipo de diseños conviene representar el funcionamiento de los FF en base a su "tabla de transiciones" en vez de su tabla de estados.

Diseño
de contadores sincrónicos

TABLA DE TRANSICIONES DEL FLIP-FLOP T

Qn	Q^{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Variables independientes

Variables dependientes

TABLA DE TRANSICIONES DEL FLIP-FLOP JK

Qn	Q^{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Variables independientes

Variables dependientes

Introducción al diseño de circuitos secuenciales sincrónicos
por el método de las transiciones

Ejemplo:

Contador módulo 4 con entrada de habilitación

***X* : Entrada de habilitación**

X=1 responde a los pulsos de entrada

X=0 no responde, permanece en el mismo estado

Módulo: 4
Código de cuenta: binario

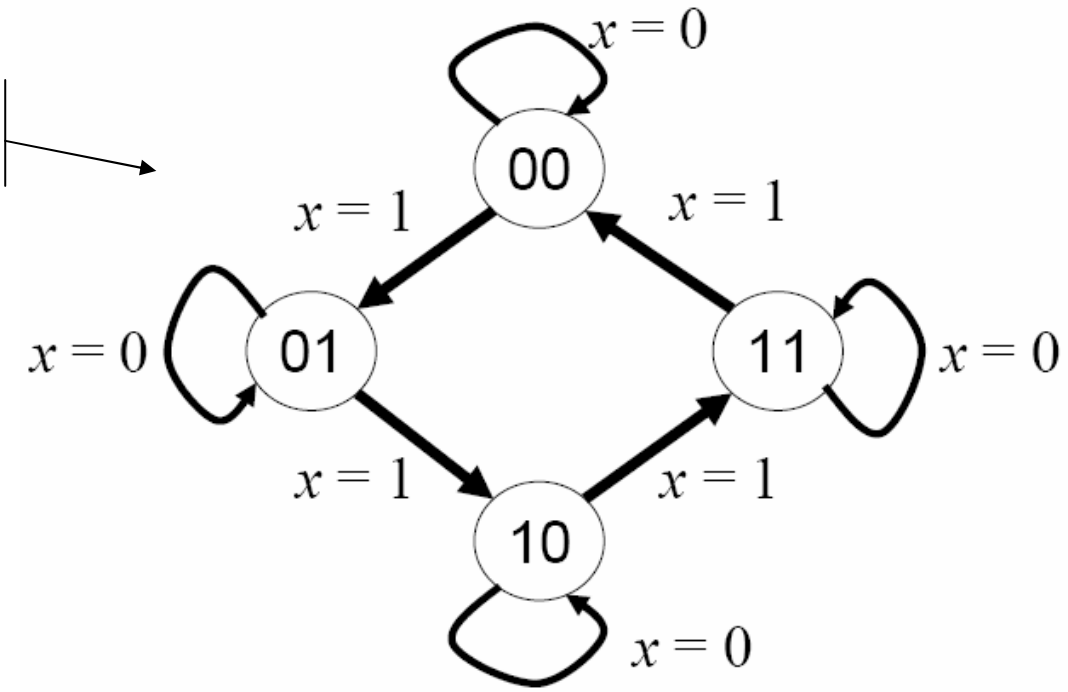


Diagrama de estados

Ejemplo:

Contador módulo 4 con entrada de habilitación

TABLA DE TRANSICIONES DEL CONTADOR

Habil.	Estado actual		Estado siguiente		Entradas FF	
X	Q_0	Q_1	Q_0	Q_1	T_0	T_1
1	**0**	**0**	**0**	**1**		
1	**0**	**1**	**1**	**0**		
1	**1**	**0**	**1**	**1**		
1	**1**	**1**	**0**	**0**		
0	**0**	**0**	**0**	**0**		
0	**0**	**1**	**0**	**1**		
0	**1**	**0**	**1**	**0**		
0	**1**	**1**	**1**	**1**		

Ejemplo:

Contador módulo 4 con entrada de habilitación

TABLA DE TRANSICIONES DEL CONTADOR

Habil.	Estado actual		Estado siguiente		Entradas FF	
X	Q_0	Q_1	Q_0	Q_1	T_0	T_1
1	**0**	**0**	**0**	**1**		
1	**0**	**1**	**1**	**0**		
1	**1**	**0**	**1**	**1**		
1	**1**	**1**	**0**	**0**		
0	**0**	**0**	**0**	**0**		
0	**0**	**1**	**0**	**1**		
0	**1**	**0**	**1**	**0**		
0	**1**	**1**	**1**	**1**		

n	n+1	T
0	0	0
0	1	1
1	0	1
1	1	0

T. de transiciones del FF-T

Ejemplo:

Contador módulo 4 con entrada de habilitación

TABLA DE TRANSICIONES DEL CONTADOR

Habil.	Estado actual		Estado siguiente		Entradas FF	
X	Q_0	Q_1	Q_0	Q_1	T_0	T_1
1	**0**	**0**	**0**	**1**	**0**	**1**
1	**0**	**1**	**1**	**0**	**1**	**1**
1	**1**	**0**	**1**	**1**	**0**	**1**
1	**1**	**1**	**0**	**0**	**1**	**1**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**
0	**1**	**1**	**1**	**1**	**0**	**0**

n	n+1	T
0	0	0
0	1	1
1	0	1
1	1	0

T. de transiciones del FF-T

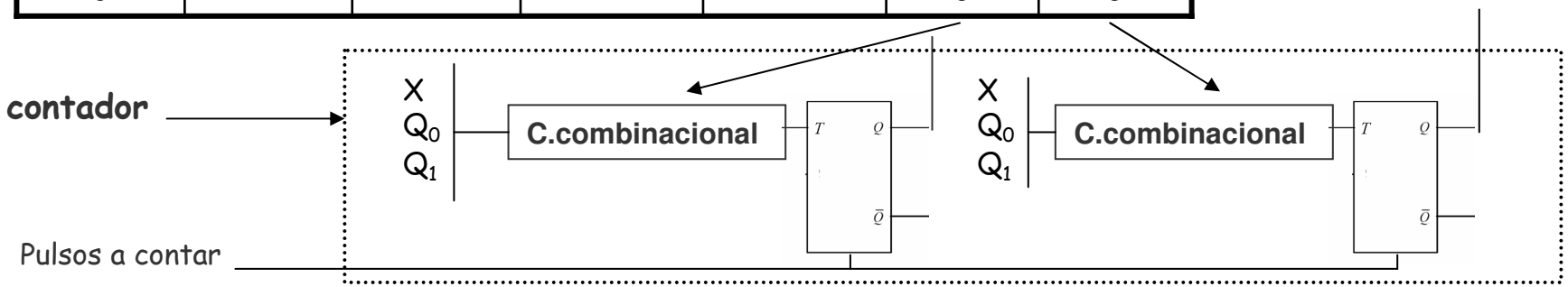
Ejemplo:

Contador módulo 4 con entrada de habilitación

Variables independientes

Variables dependientes

Habil.	Estado actual		Estado siguiente		Entradas FF	
X	Q_0	Q_1	Q_0	Q_1	T_0	T_1
1	**0**	**0**	**0**	**1**	**0**	**1**
1	**0**	**1**	**1**	**0**	**1**	**1**
1	**1**	**0**	**1**	**1**	**0**	**1**
1	**1**	**1**	**0**	**0**	**1**	**1**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**	**0**
0	**1**	**1**	**1**	**1**	**0**	**0**



Método de las transiciones

1. Formalizar el funcionamiento esperado por medio de un diagrama de estado. Identificar las variables entrada y de salida

2. El número de flip-flops necesarios para el circuito es el número de bits que tienen los estados.

3. Se realiza la tabla de estados y se agregan columnas con los 0's y 1's necesarios en las entradas de cada FF para que esa transición se produzca (tomada de la tabla de transiciones del FF).

4. Se diseña el circuito combinacional para cada entrada de cada flip-flop usando mapas de Karnaugh.

5. Se implementa el circuito en base a las ecuaciones de entrada a los FF

Método de las transiciones

Otros ejemplos en los que este método es aplicable:

1. Diseñar un contador sincrónico de décadas que cuente en binario

2. Idem con una entrada que permita elegir si la cuenta es ascendente o descendente.

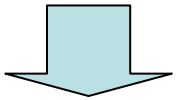
Método de las transiciones

Otros ejemplos en los que este método es aplicable:

1. Diseñar un contador sincrónico de décadas que cuente en binario

2. Idem con una entrada que permita elegir si la cuenta es ascendente o descendente.

Este método de diseño no está limitado a contadores.



Un circuito secuencial cualquiera, que podría incluir otras varias entradas y salidas, puede diseñarse en forma similar.