Arquitectura de Computadoras para Ingeniería

(Cód. 7526)
1° Cuatrimestre 2016

Dra. Dana K. Urribarri
DCIC - UNS

Circuitos Secuenciales (continuación)

Asignación de estados

En un sistema tipo pulso, para *r* estados se necesitan *n* elementos de memoria tal que:

$$2^{n-1} < r \leq 2^n$$

Para $r = 3$, $n = 2$

Para $r = 6$, $n = 3$

Los *r* estados se pueden nombrar de

$$P\left(2^n, r\right) = \frac{2^n!}{\left(2^n - r\right)!}$$

maneras (aparentemente) diferentes.

Asignación de estados

Del total de posibilidades hay algunas que aparentan ser diferentes, pero los circuitos resultantes son iguales.

A) Si difieren en columnas complementadas.

y_1	y_2	y_3	y_1	y_2	y_3
0	0	1	0	1	1
1	0	1	1	1	1
1	1	0	1	0	0

En vez de Q se toma $\overline{Q}$.

B) Si difieren en el orden de las columnas.

y_1	y_2	y_3	y_1	y_2	y_3
0	0	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0

Sólo cambia el nombre.

Asignación de estados

De la cantidad inicial de permutaciones

$$P\left(2^n, r\right) = \frac{2^n!}{\left(2^n - r\right)!}$$

A) Si difieren no en columnas complementadas, se reduce en 2^n

B) Si no difieren en el orden de las columnas, se reduce en $n!$

El total de posibilidades realmente diferentes será:

$$\frac{\left(2^n - 1\right)!}{\left(2^n - r\right)! n!}$$

Asignación de estados

- En un circuito pulso, aunque asignaciones diferentes resultan en circuitos diferentes, cualquier asignación funciona correctamente.

  No nos preocupamos por cuál es la mejor asignación.

- Observación: En un circuito por nivel, no cualquier asignación funciona correctamente (condición de carrera)

Expresiones del próximo estado

- *B* bits (y_1, y_2,...y_B) para los estados
- *E* pulsos de entrada (x_1,x_2,…,x_E)
- *B* FF, uno por cada bit de los estados.
- Para cada bit *i* hay que hallar las funciones de entrada

$$f(y_1, y_2,...y_B,x_1,x_2,\ldots,x_E) =$$

$$= g_1(y_{1..b})\, x_1 + \ldots + g_E(y_{1..b})\, x_E$$

del FF *i* para las cuáles el próximo estado de y_i sea y^+_i.

Expresiones del próximo estado

- Ejemplo pulsos:
  - 4 estados
  - 2 entradas pulso (x_1 y x_2)
  - Salida pulso.

y_1	y_2	x_1	x_2
0	0	11/0	00/0
1	1	01/0	00/0
0	1	10/0	00/1
1	0	10/0	00/0

Expresiones del próximo estado

Implementación con FF SR

y_1	y_2	x_1	x_2
0	0	11/0	00/0
1	1	01/0	00/0
0	1	10/0	00/1
1	0	10/0	00/0

$Q \rightarrow Q^+$	S	R
0 → 0	0	*
0 → 1	1	0
1 → 0	0	1
1 → 1	*	0

Para y^+_1

S_1:

y_1 \ y_2	0	1
0	1	1
1	*	0

R_1:

y_1 \ y_2	0	1
0	0	0
1	0	1

$S_1 = \bar{y}_1$

$R_1 = y_1 y_2$

Para y^+_2

S_2:

y_1 \ y_2	0	1
0	0	0
1	0	0

R_2:

y_1 \ y_2	0	1
0	*	*
1	1	1

$S_2 = 0$

$R_2 = 1$

Expresiones del próximo estado

Implementación con FF SR

y_1	y_2	x_1	x_2
0	0	11/0	00/0
1	1	01/0	00/0
0	1	10/0	00/1
1	0	10/0	00/0

$Q \rightarrow Q^+$	S	R
0 → 0	0	*
0 → 1	1	0
1 → 0	0	1
1 → 1	*	0

Para y^+_2

S_1:

y_1 \ y_2	0	1
0	1	0
1	0	*

R_1:

y_1 \ y_2	0	1
0	0	1
1	*	0

S_2:

y_1 \ y_2	0	1
0	0	0
1	0	0

R_2:

y_1 \ y_2	0	1
0	*	*
1	1	1

$$S_1 = \bar{y}_1 \bar{y}_2$$
$$R_1 = \bar{y}_1 y_2$$

$$S_2 = 0$$
$$R_2 = 1$$

Expresiones del próximo estado

- Para el bit 1 del próximo estado, se tiene que:
  - $S_1 = \overline{y}_1$
  - $R_1 = y_1 y_2$

  } x_1

  - $S_2 = 0$
  - $R_2 = 1$

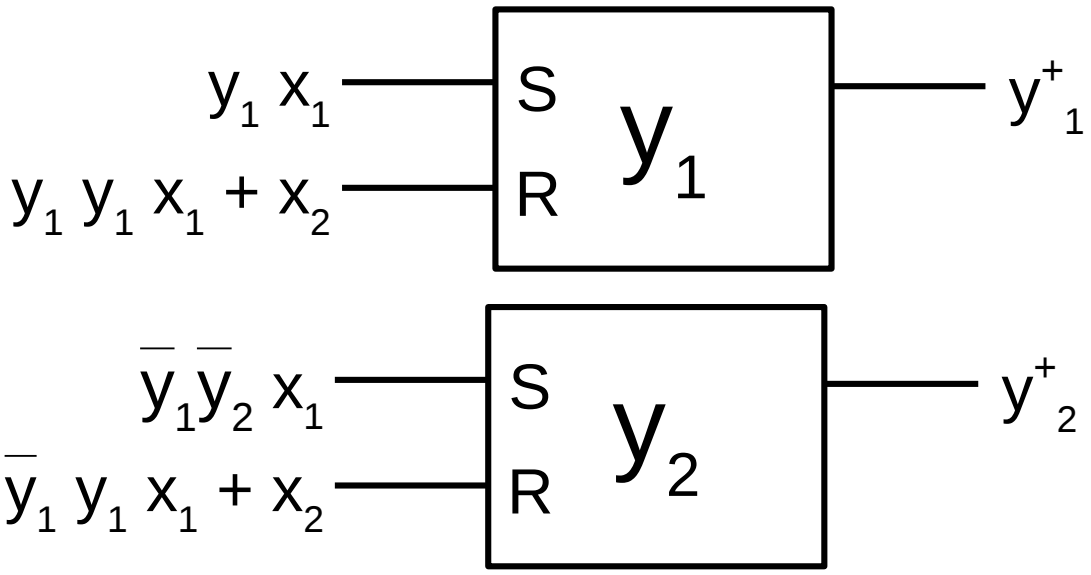
  } x_2

- Para el bit 2 del próximo estado, se tiene que:
  - $S_1 = \overline{y}_1 \overline{y}_2$
  - $R_1 = \overline{y}_1 y_2$

  } x_1

  - $S_2 = 0$
  - $R_2 = 1$

  } x_2



Expresiones del próximo estado

- Ejemplo niveles + clock (ej. formulación):
  - 3 estados
  - 2 entradas nivel (A y B) y una señal de reloj
  - 1 salida nivel z.

y_1y_2	x_1x_2	$x_1\bar{x}_2$	$\bar{x}_1x_2$	$\bar{x}_1\bar{x}_2$	Z
01	01	01	01	11	0
10	10	01	01	11	1
11	10	01	01	11	0

Expresiones del próximo estado

Implementación con FF T

y_1y_2	x_1x_2	$x_1\bar{x}_2$	$\bar{x}_1x_2$	$\bar{x}_1\bar{x}_2$	Z
01	01	01	01	11	0
10	10	01	01	11	1
11	10	01	01	11	0

$Q \rightarrow Q^+$	T
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	1
$1 \rightarrow 1$	0

$T_1 (y_1,y_2,x_1,x_2)$

y_1y_2 \ x_1x_2	00	01	11	10
00	*	*	*	*
01	1	0	0	0
11	0	1	0	1
10	0	1	0	1

$$T_1 = \bar{y}_1\bar{x}_1\bar{x}_2 + y_1\bar{x}_1x_2 + y_1x_1\bar{x}_2$$

Expresiones del próximo estado

Implementación con FF T

y_1y_2	x_1x_2	$x_1\bar{x}_2$	$\bar{x}_1x_2$	$\bar{x}_1\bar{x}_2$	Z
01	01	01	01	11	0
10	10	01	01	11	1
11	10	01	01	11	0

$Q \rightarrow Q^+$	T
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	1
$1 \rightarrow 1$	0

$T_2\ (y_1, y_2, x_1, x_2)$

y_1y_2 \ x_1x_2	00	01	11	10
00	*	*	*	*
01	0	0	0	0
11	0	0	1	0
10	1	1	0	1

$$T_2 = y_1\bar{y}_2\bar{x}_2 + y_1\bar{y}_2\bar{x}_1 + y_1y_2x_1x_2$$

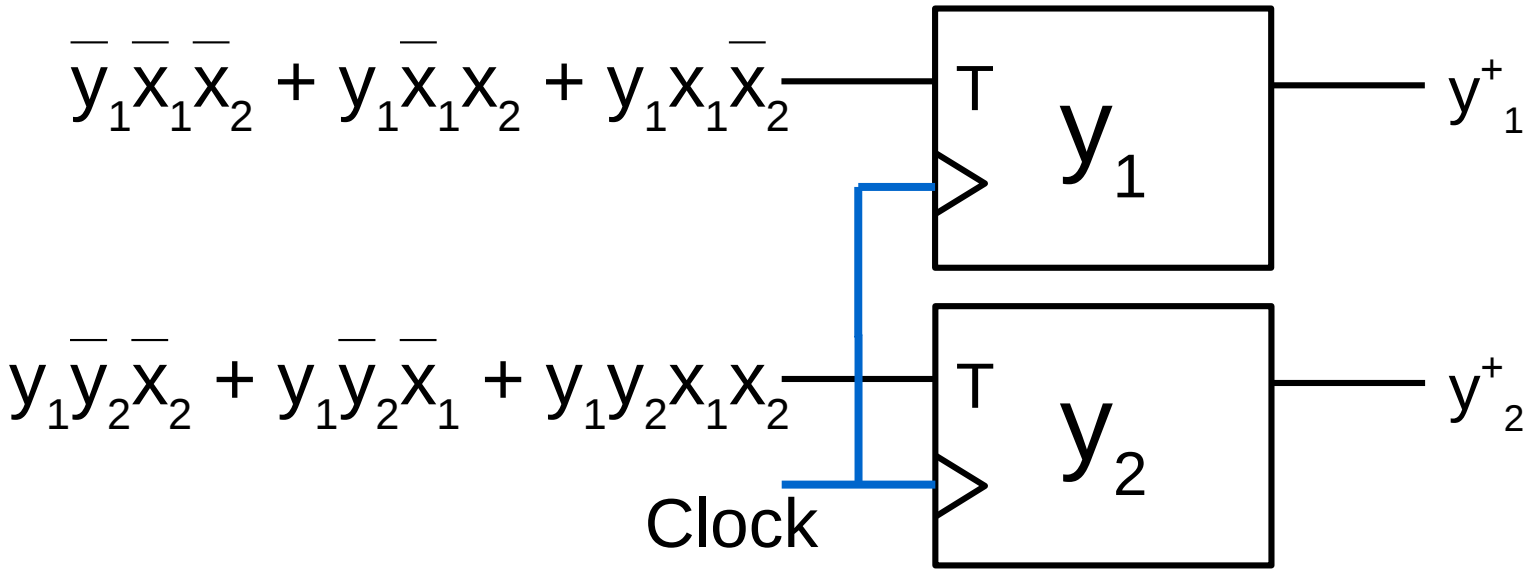
Expresiones del próximo estado

Para el bit 1 del próximo estado, se tiene que:

$$T_1 = \bar{y}_1\bar{x}_1\bar{x}_2 + y_1\bar{x}_1x_2 + y_1x_1\bar{x}_2$$

Para el bit 2 del próximo estado, se tiene que:

$$T_2 = y_1\bar{y}_2\bar{x}_2 + y_1\bar{y}_2\bar{x}_1 + y_1y_2x_1x_2$$



Expresiones de salida

- Derivar la salida a partir de la tabla de estados

y_1	y_2	x_1	x_2
0	0	11/0	00/0
1	1	01/0	00/0
0	1	10/0	00/1
1	0	10/0	00/0

→

y_1	y_2	x_1	x_2
0	0	0	0
1	1	0	0
0	1	0	1
1	0	0	0

$z(x_1)$:

y_1 \ y_2	0	1
0	0	0
1	0	0

$z(x_2)$:

y_1 \ y_2	0	1
0	0	1
1	0	0

$$z(x_1,x_2) = 0\, x_1 + \bar{y}_1 y_2\, x_2 = \bar{y}_1 y_2\, x_2$$

Implementación

Última Etapas del diseño.

Implementación con compuertas, FF, multiplexores, decoders, dispositivos programables, etc.

Bibliografía

- *Capítulo 4.* Morris Mano, Kime & Martin. *Logic and computer design fundamentals.* Prentice Hall (5ta Ed. 2015)
- *Capítulo suplementario "Design and Analysis using JK and T flip-flops".* Morris Mano, Kime & Martin. *Logic and computer design fundamentals.* http://wps.pearsoned.com/ecs_mano_lcdf_5/248/63706/16308896.cw/index.html

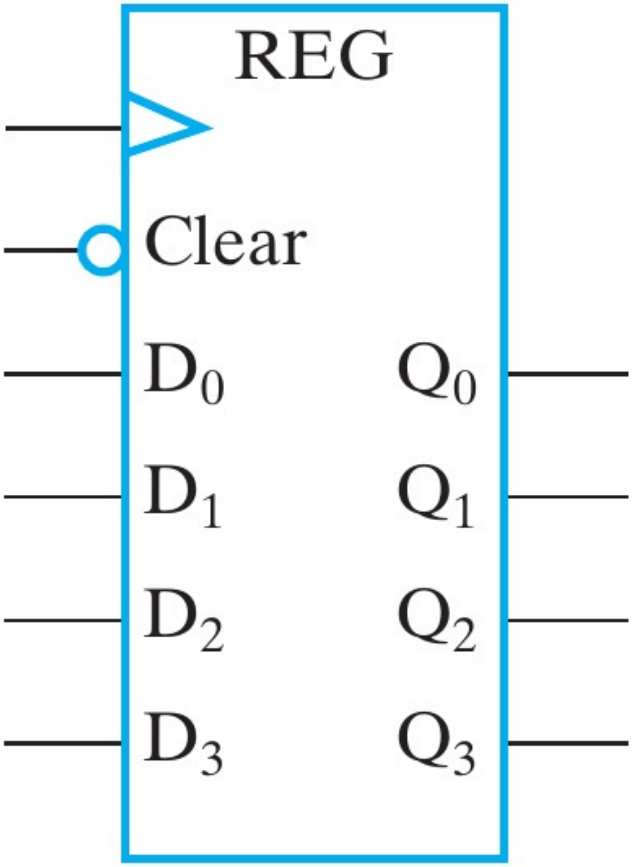
Registros y Contadores

Contadores y registros

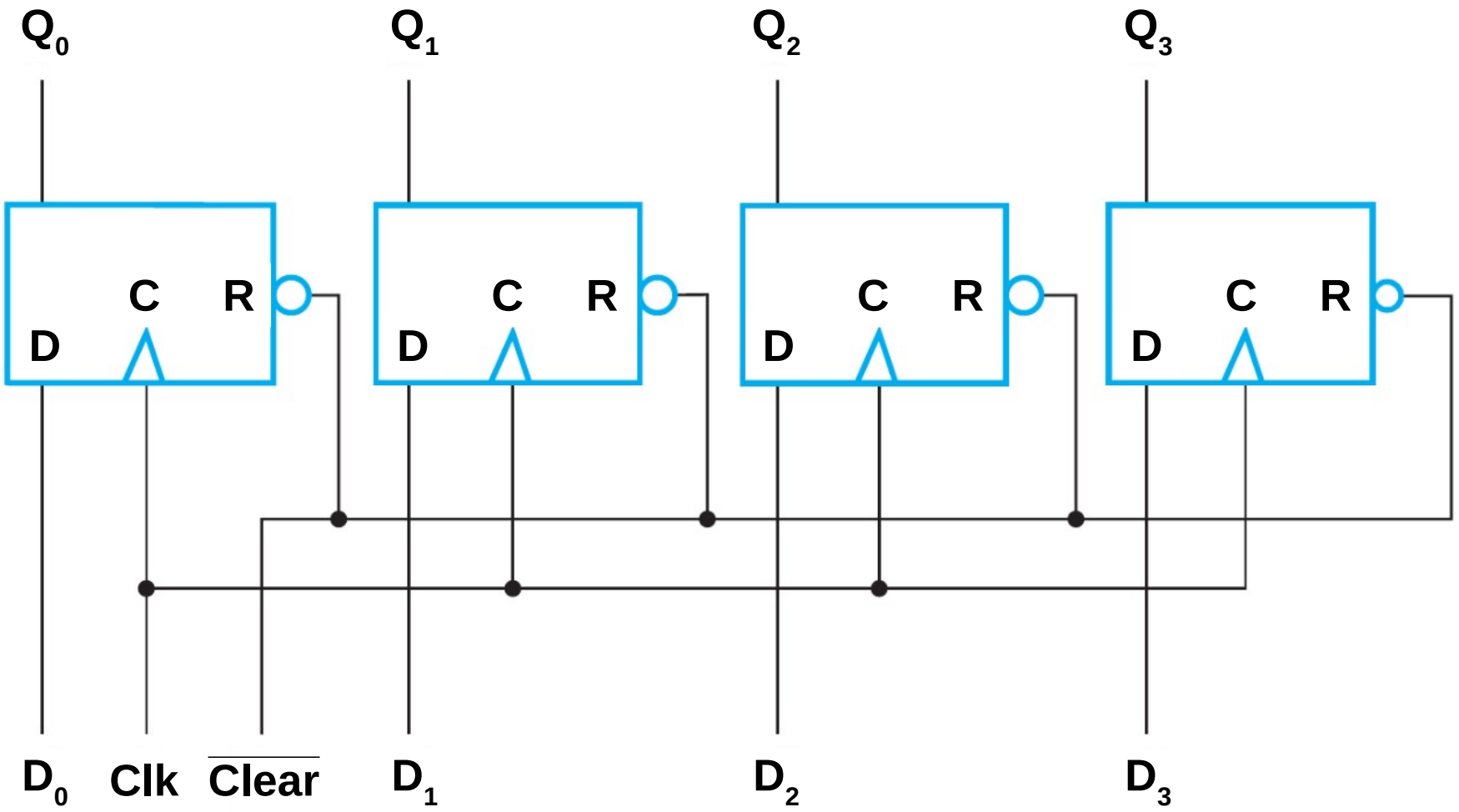
- Registro
  - Grupo de FF que comparten el reloj, cada uno almacena un bit de información. Un registro de *n*-bits consiste de *n* FF.
  - Además de los FF, puede haber compuertas que realicen alguna tarea sobre los datos.
- Contador:
  - Un grupo de FF que comparte el reloj y pasan por una secuencia predefinida de estados binarios.

Registro

- El más simple consiste solo de FF. No tiene compuertas.
- Las entradas D_i determinan las salidas Q_i con cada flanco positivo del reloj.
- La señal $\overline{\text{Clear}}$ resetea las salidas del registro.



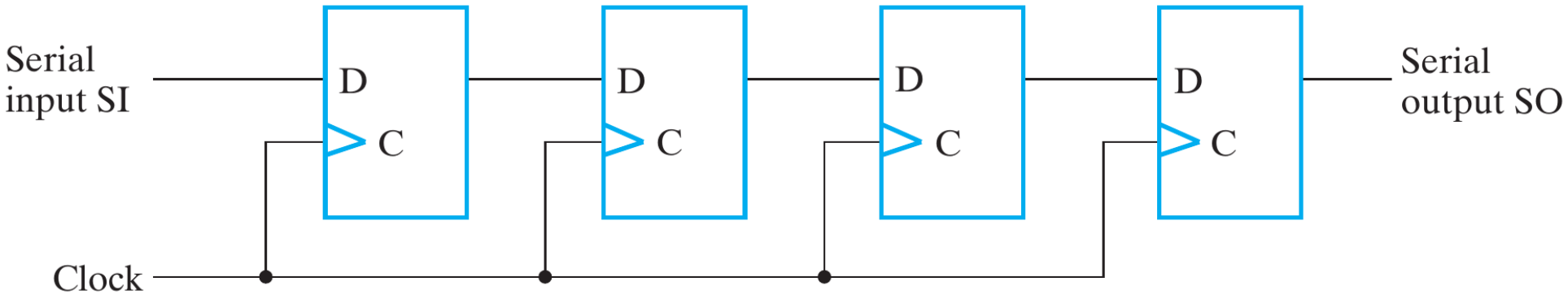
Registro



Un reloj común dispara todos los FF y el dato binario disponible en las entradas D se transfiere a las salidas Q.

Registro de desplazamiento (shift)

- Registro con la capacidad de desplazar, en una dirección seleccionada, la información binaria en cada FF al FF vecino.
  - Shift a derecha: $(z_0,z_1,\ldots,z_n) \rightarrow (0,z_0,z_1,\ldots,z_{n-1})$
  - Cadena de FF con reloj común.



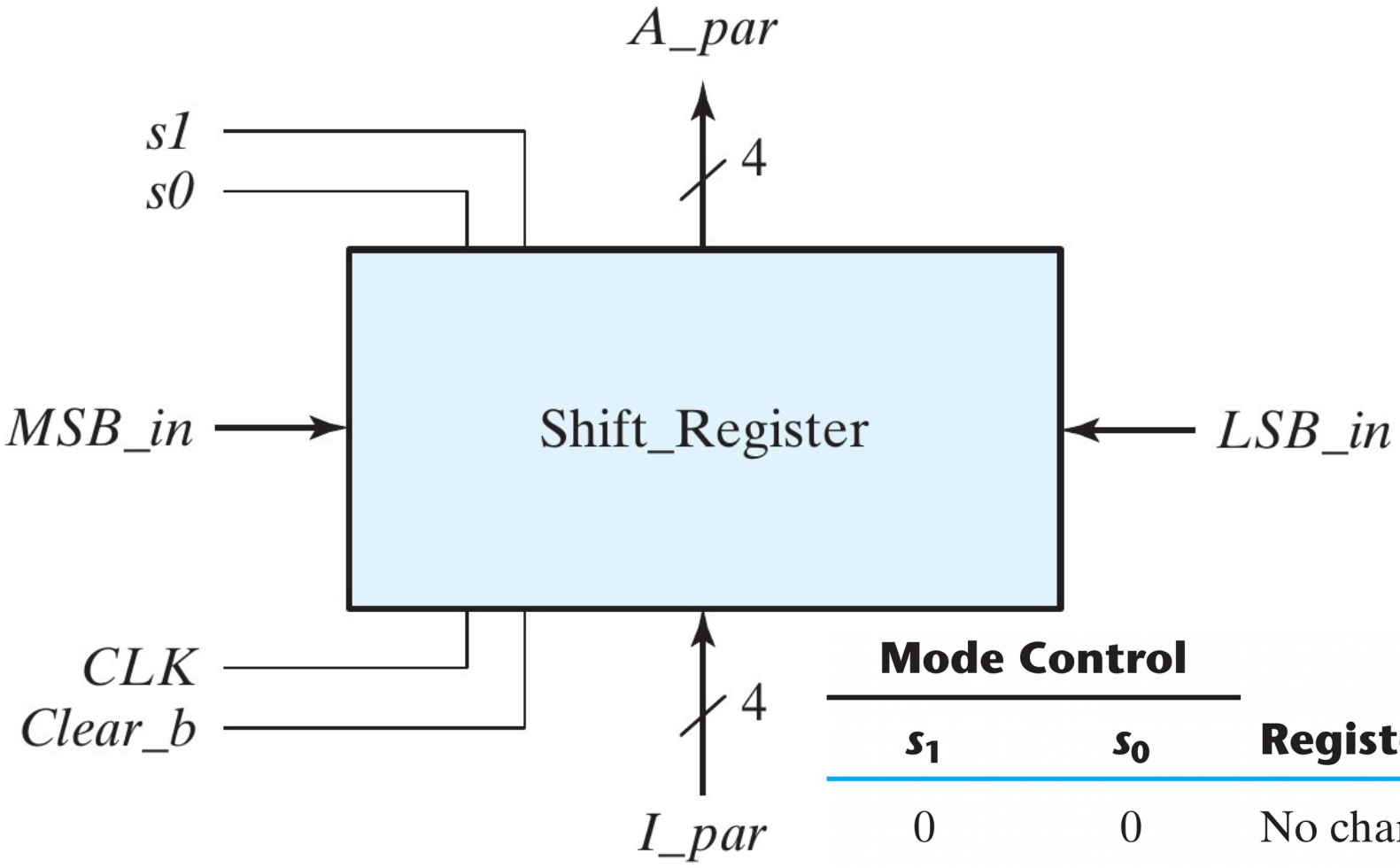
Registro de desplazamiento (shift)

El registro de desplazamiento más general permite:

- Clear para poner el registro en 0
- Entrada de reloj
- Control para desplazar a derecha, con líneas de entrada y salida seriales.
- Control para desplazar a izquierda, con líneas de entrada y salida seriales.
- Control de carga paralela y líneas de entradas asociadas.
- Líneas de salida paralela
- Control para habilitar/deshabilitar cambios en la salida en respuesta al reloj.

Utilidad: conversión serie en paralelo y viceversa.

Registro de desplazamiento (shift)



Mode Control

s_1	s_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Contador

- Registro que cicla a través de una secuencia predefinidas de estados en respuesta a un pulso de entrada.
- El pulso de entrada puede
  - ser un pulso de reloj u originado por una fuente externa
  - ocurrir a intervalos fijos de tiempo o random.
- Los contadores pueden diferir en módulo, sincrónicos o asincrónicos, hacia arriba (up) o hacia abajo (down).

Contadores asincrónicos

- Son contadores *ripple*
- El clock dispara los cambios en un FF.
- La salida de ese FF sirve como fuente de disparo de los demás FF.

  - Contador binario en ripple

Contador binario asincrónico

- Contador que cicla a través del a secuencia de números binarios.
- Un contador binario de *n*-bits consiste de *n* FF y puede contar, en binario, desde 0 hasta 2^n-1.

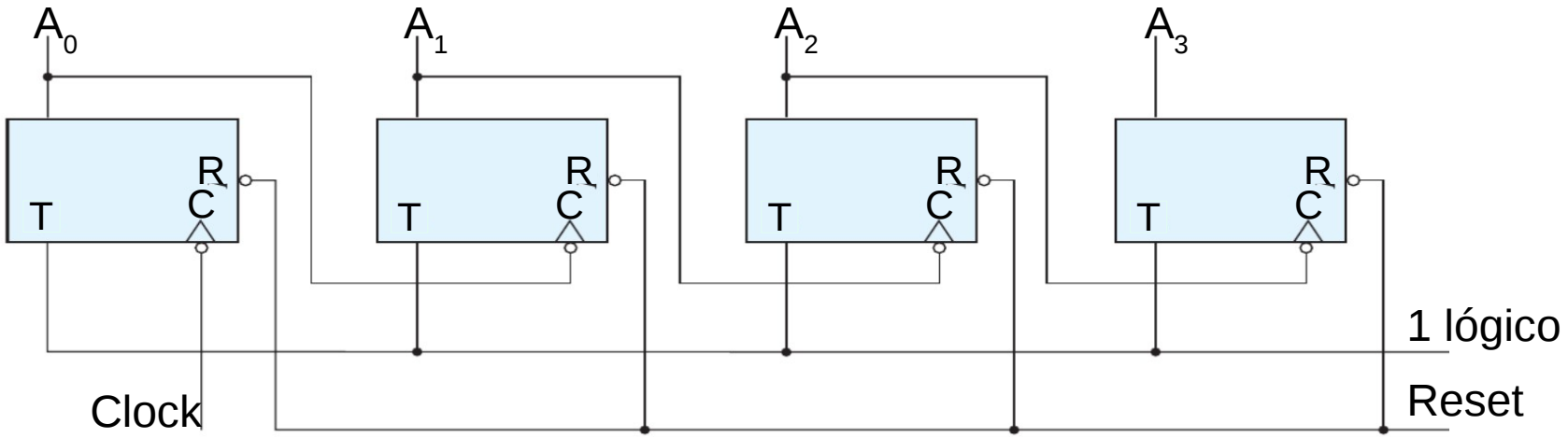
Binary Count Sequence

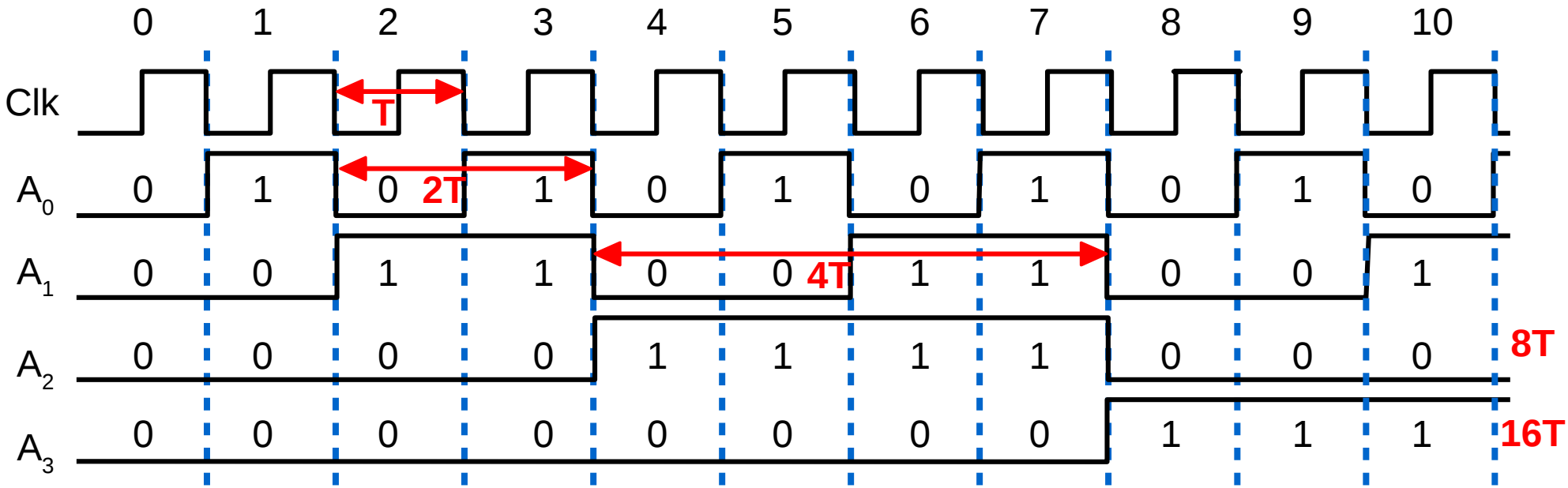
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

Contador binario asincrónico

- El contador binario en ripple consiste de una serie de FF complementados conectados.
  - La salida de cada FF se encuentra conectada al reloj del FF del bit siguiente.
  - El FF del bit menos significativo recibe el pulso de entrada.
- Puede implementarse con FF T, FF JK con ambas entradas en 1 o con FF D usando la salida complementada.

Contador binario asincrónico





Funciona como un divisor de frecuencia.

Contadores sincrónicos

- En un contador sincrónico, todos los FF reciben la misma entrada de reloj.
- El reloj común dispara todos los FF simultáneamente.
- La evolución del contador está dada por las funciones de entrada.

  - Contador binario
  - Contador anillo (ring counter)
  - Contador Möbius (Contador Johnson)

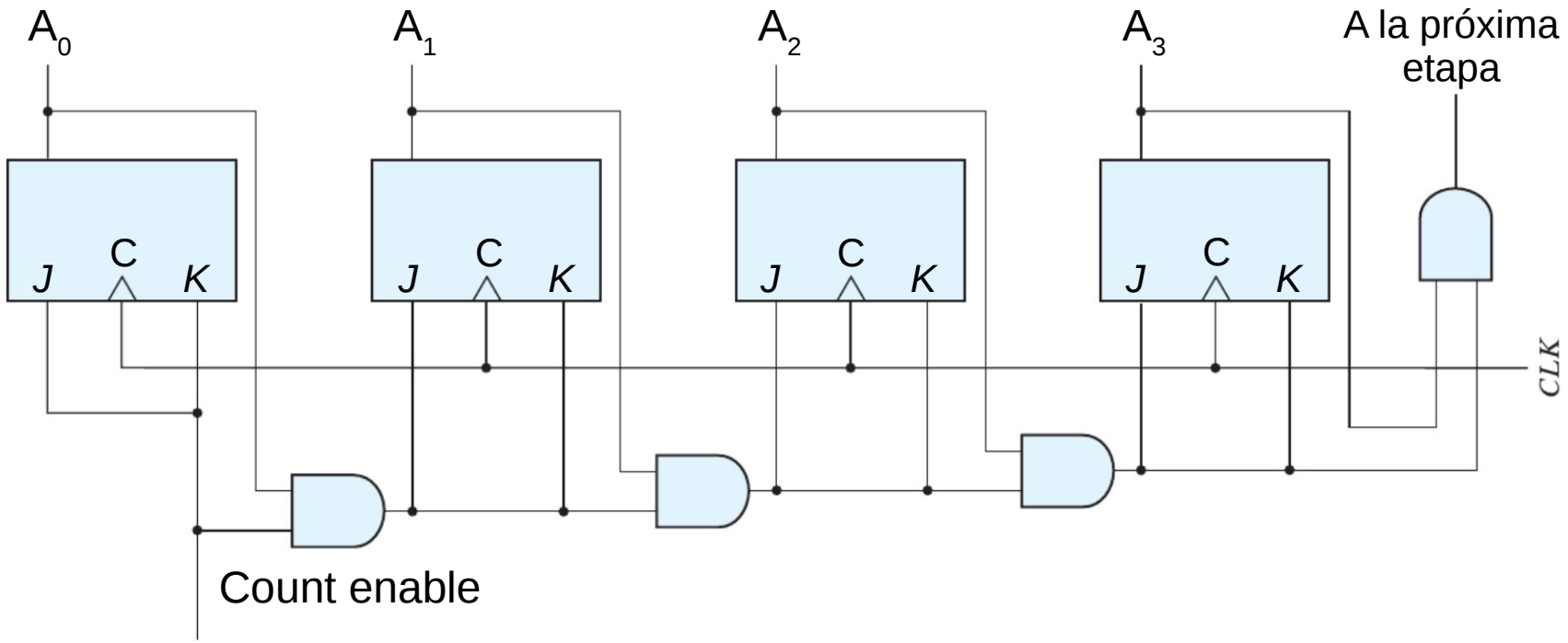
Contador binario

- Un contador binario de *n*-bits consiste de *n* FF y puede contar, en binario, desde 0 hasta 2^n-1.
- El bit menos significativo cambia siempre.
- El resto de los bits, cambian cuando todos los bits anteriores estaban en 1.

  Ó cuando el bit inmediato anterior, estaban en 1 y debe pasar a 0.
  - 0001 → 0010
  - 0011 → 0100
  - 0111 → 1000

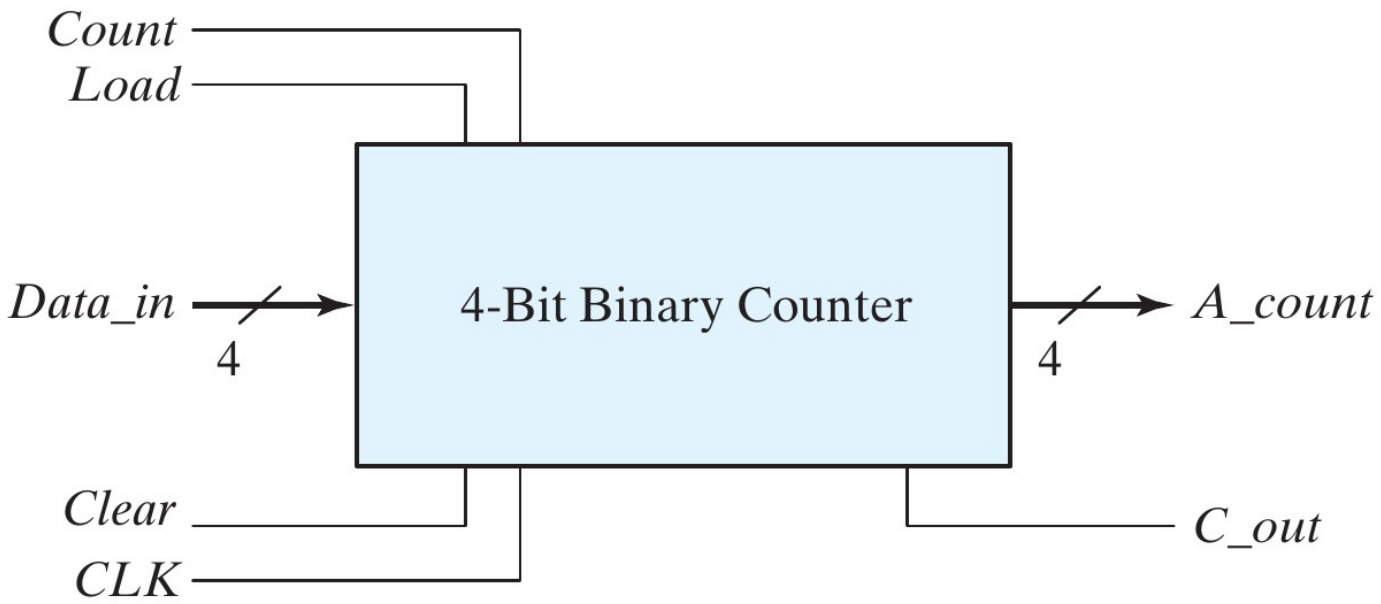
ABCD	$A^+B^+C^+D^+$
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	1010
1010	1011
1011	1100
1100	1101
1101	1111
1111	0000

Contador binario



Contador binario carga paralela

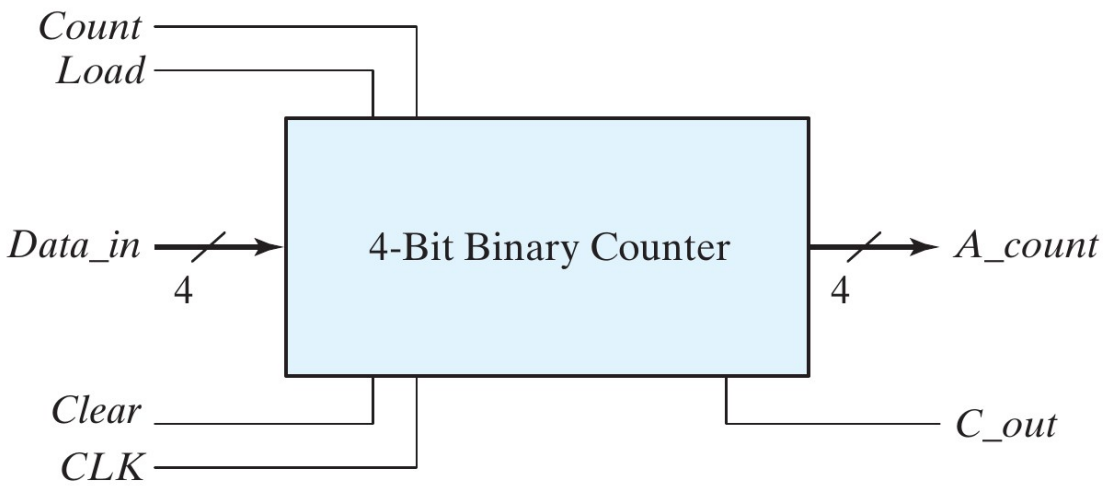
- Flexibiliza el uso de los contadores.
- La carga paralela permite modificar el estado del contador *antes* de la operación de cuenta.



Contador binario carga paralela

Cuando Load es 1, se deshabilita la cuenta y se transfiere el valor de las entradas al estado del contador.

Cuando Load y Count están en 0, los pulsos del reloj no cambian el estado del contador.



Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

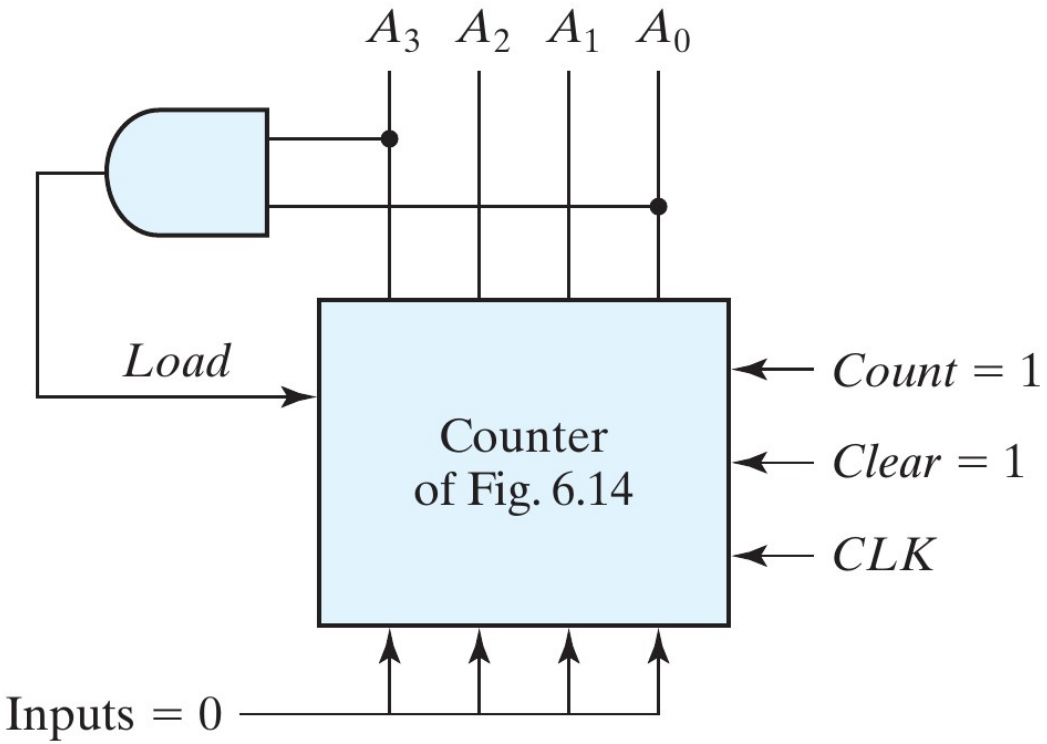
Contador BCD

- Se puede diseñar un circuito secuencial de 4 FF a través de la tabla de estados:

ABCD	$A^+B^+C^+D^+$
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	0000

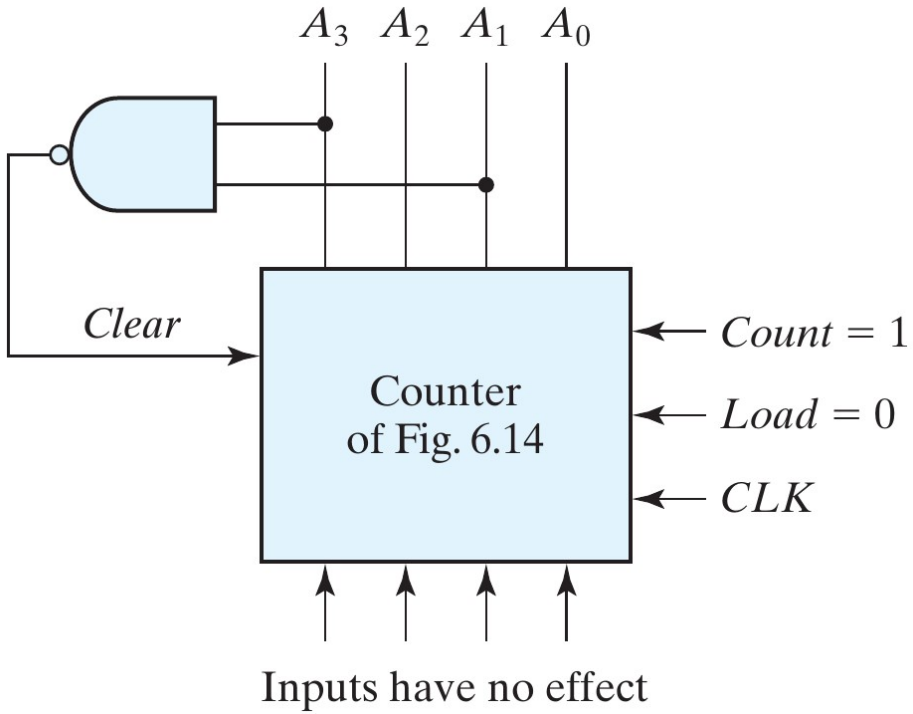
Contador BCD

- Se puede usar un contador de carga paralela y codificar el estado 9 para cargar un 0 en el próximo pulso de reloj.



Contador BCD

- También se *podría* usar la señal Clear del contador.

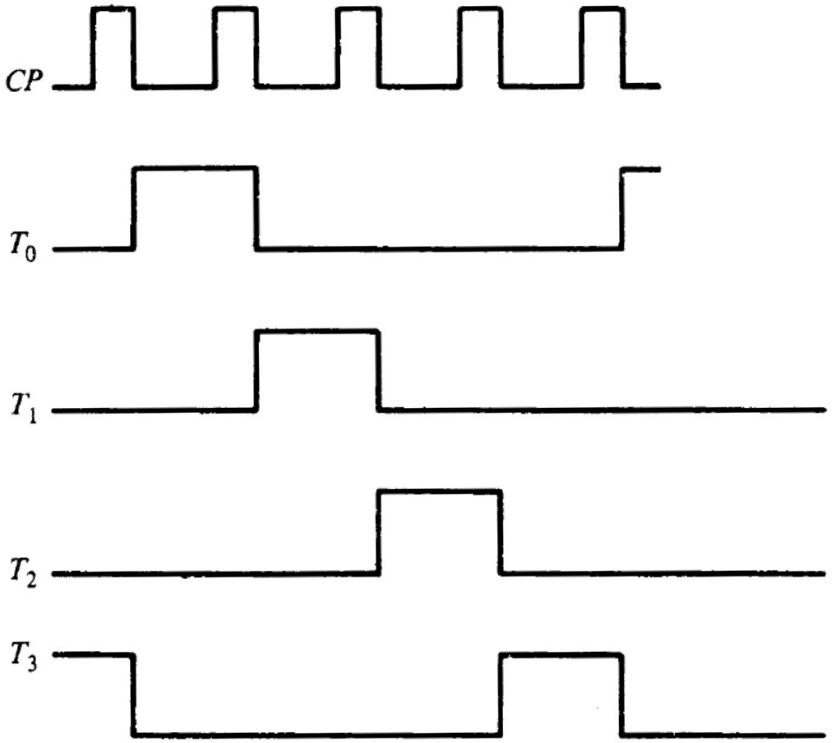


  - El Clear no dependen del reloj. Inmediatamente cuando la salida es 1010 el registro se resetea.
  - Tiempo despreciable, pero puede generar picos indeseables en la salida.

Ring counter

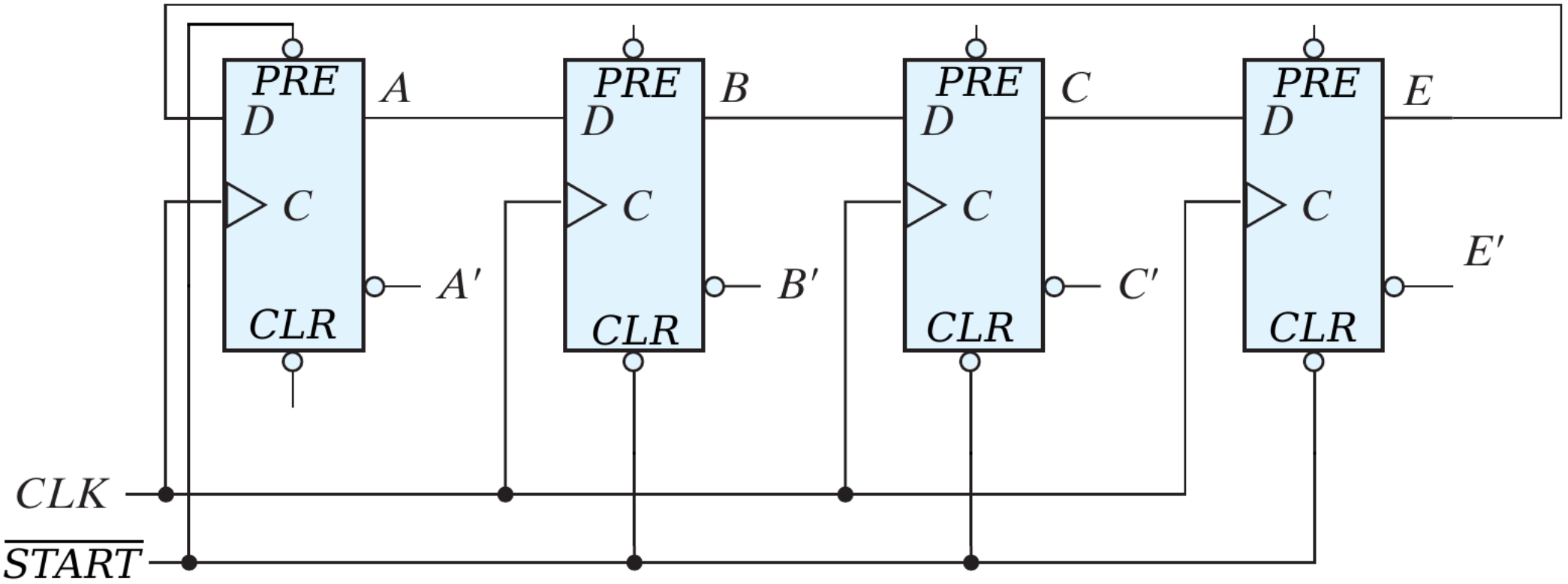
- Define una secuencia de estados con un único FF en 1 en cada momento. Todos los demás están en 0.
- Ese bit en 1 se desplaza de un FF al siguiente.
- Es costoso. Para un contador de *n* estados se necesitan *n* FF.
- Define *n* señales de temporizado.

ABCD	$A^+B^+C^+D^+$
0001	0010
0010	0100
0100	1000
1000	0001



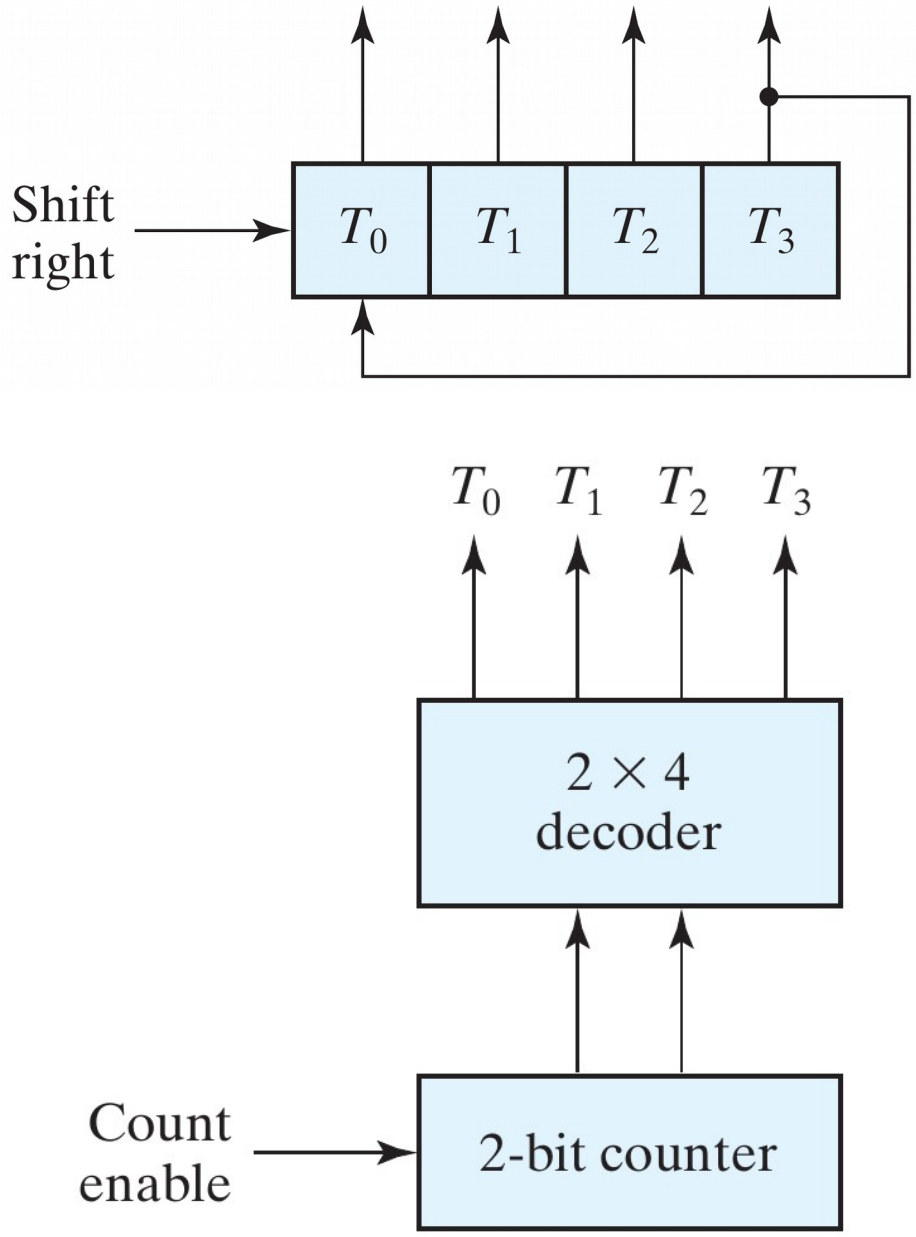
Ring counter

- La implementación con FF requiere de FF con la capacidad de Preset/Clear



Ring counter

- Puede implementarse como un registro de desplazamiento inicializado en 1000 y configurado de manera circular.
- Puede implementarse con contador binario y un decoder.



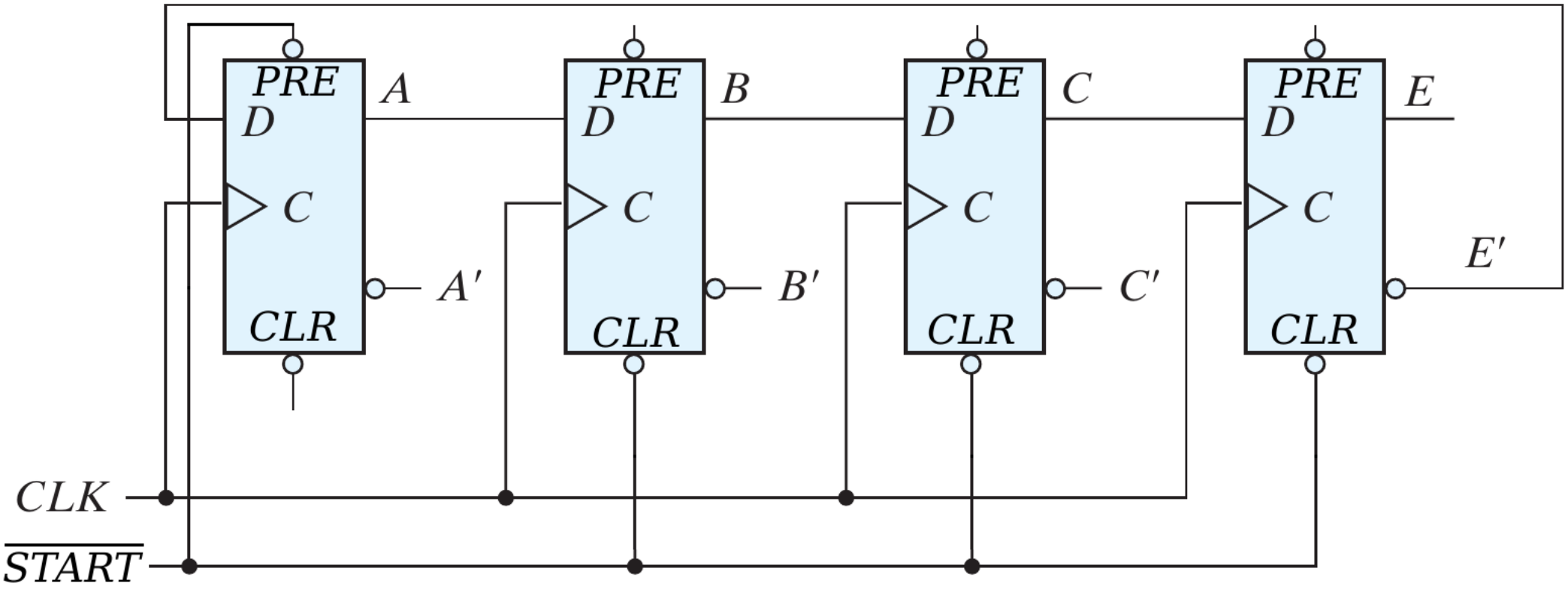
Contador Möbius

- También llamado Contador Johnson.
- Similar al Ring counter.
- Con *n* FF logra 2*n* estados.

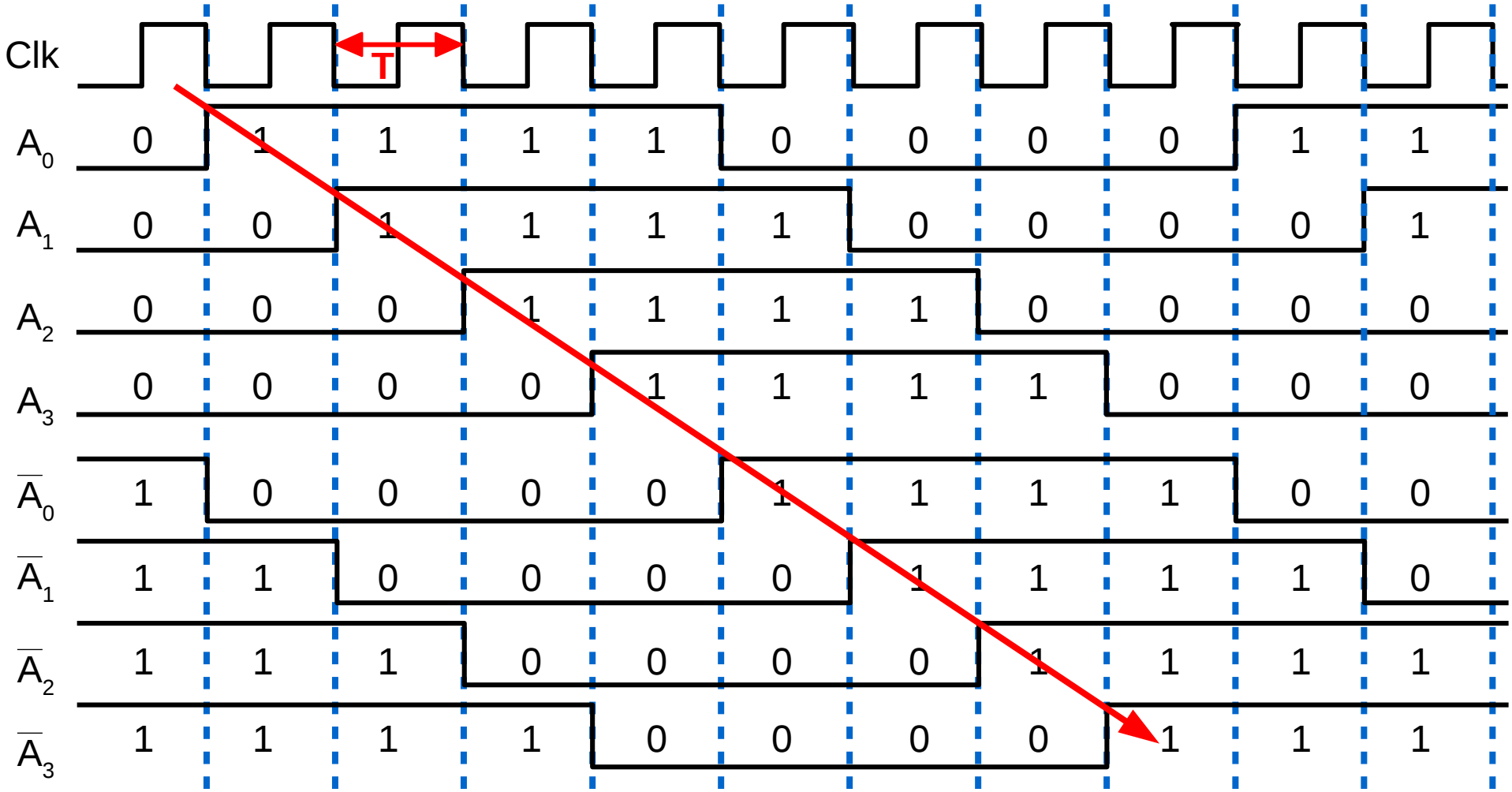
A	*B*	*C*	*E*
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1

Contador Möbius

Duplica la cantidad de estados conectando la salida negada del último FF a la entrada del primero.



Contador binario asincrónico



Para *n* FF: tenemos 2*n* señales, de período *n*T, desfasadas en T.

Bibliografía

- *Capítulo 6.* M. Morris Mano & Michael D. Celetti. *Digital Design: With an Introduction to the Verilog HDL*. Pearson. (2015, 5ta Ed.)