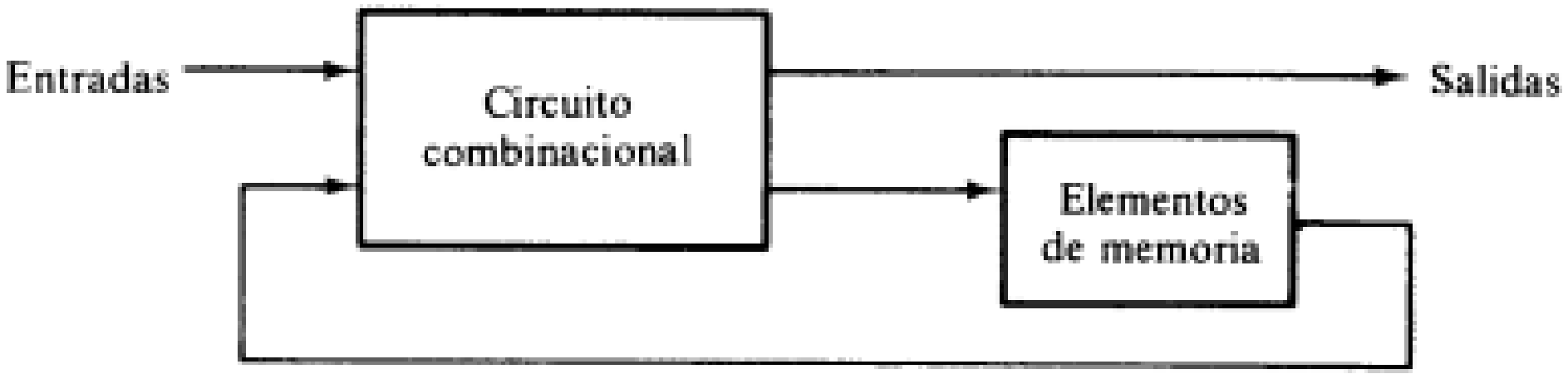
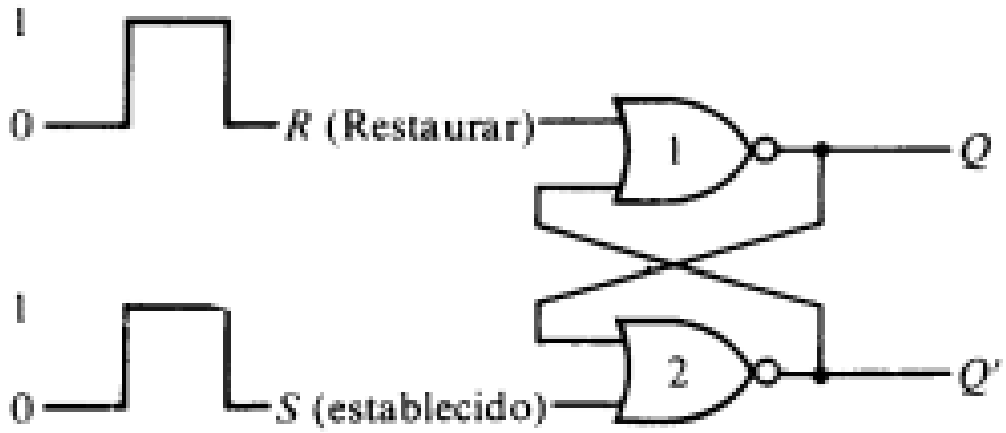
Sistemas Secuenciales

Diagrama de bloques de un sistema secuencial



LATCH (NOR)



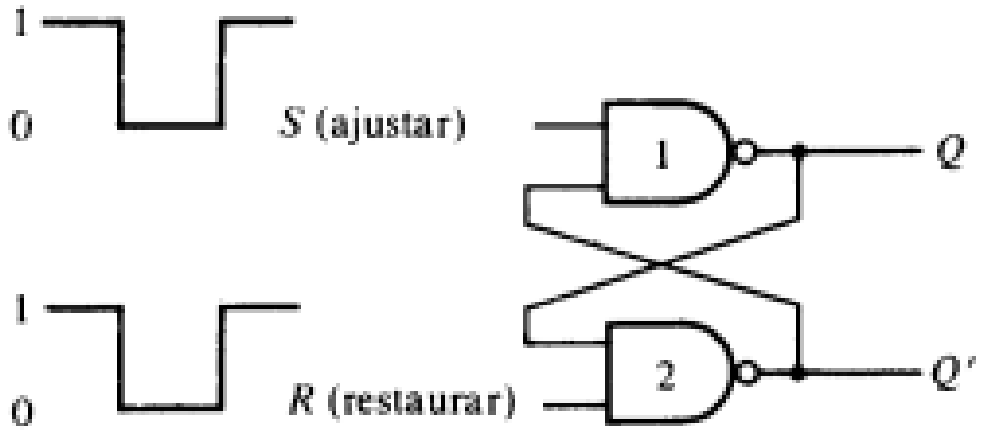
(a) Diagrama lógico

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(después que $S = 1$, $R = 0$)
0	1	0	1	
0	0	0	1	(después que $S = 0$, $R = 1$).
1	1	0	0	

(b) Tabla de verdad

Circuito flip-flop básico con compuertas NOR.

LATCH (NAND)

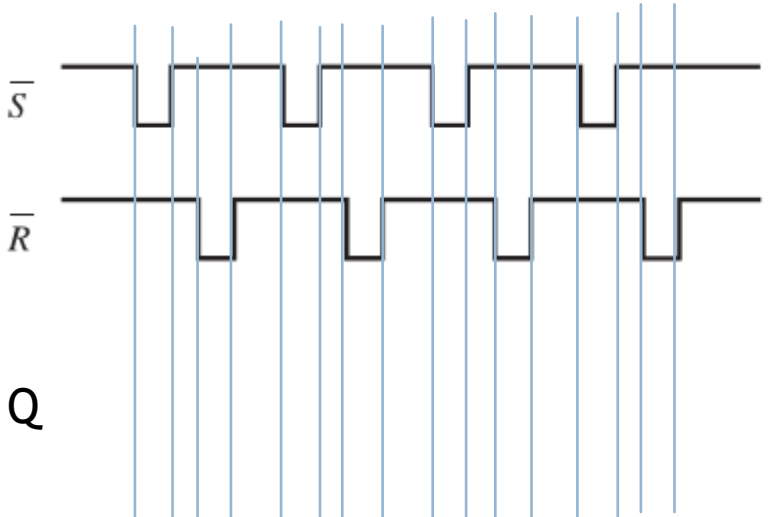


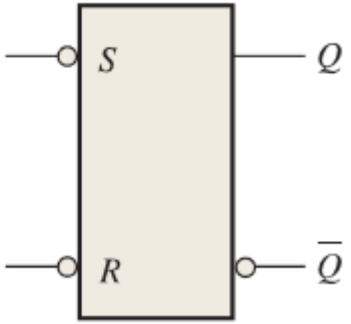
(a) Diagrama lógico

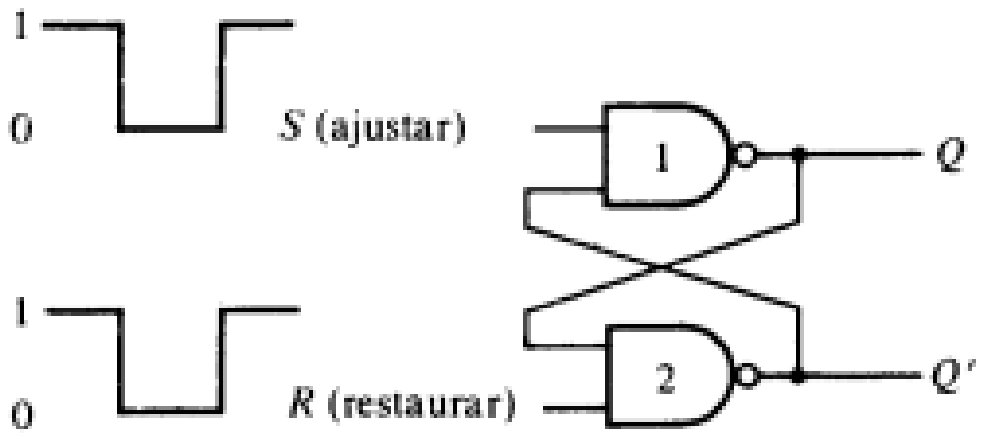
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(después que $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(después que $S = 0, R = 1$)
0	0	1	1	

(b) Tabla de verdad

Circuito flip-flop básico con compuertas NAND.





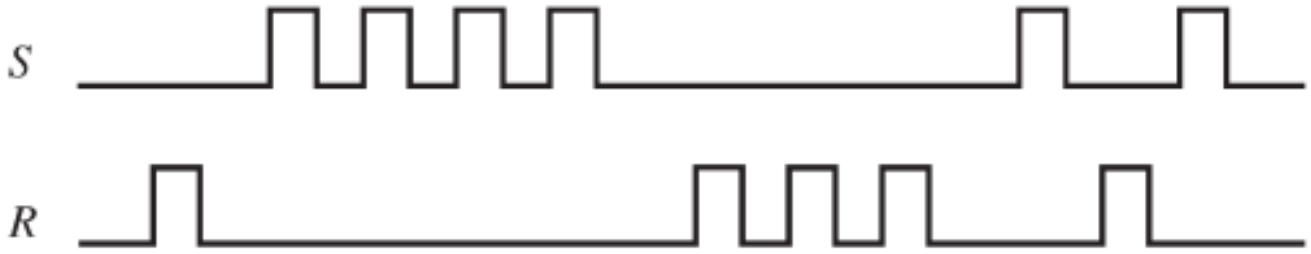


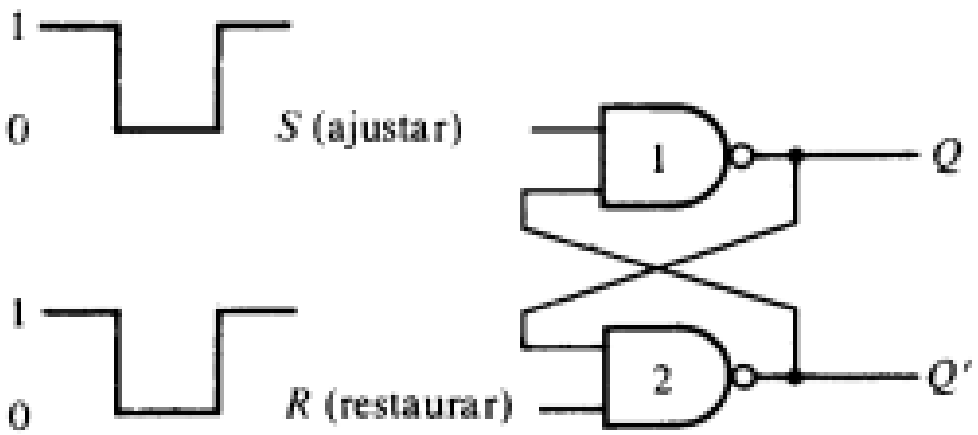
(a) Diagrama lógico

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(después que $S = 1$, $R = 0$)
0	1	1	0	
1	1	1	0	(después que $S = 0$, $R = 1$)
0	0	1	1	

(b) Tabla de verdad

Circuito flip-flop básico con compuertas NAND.



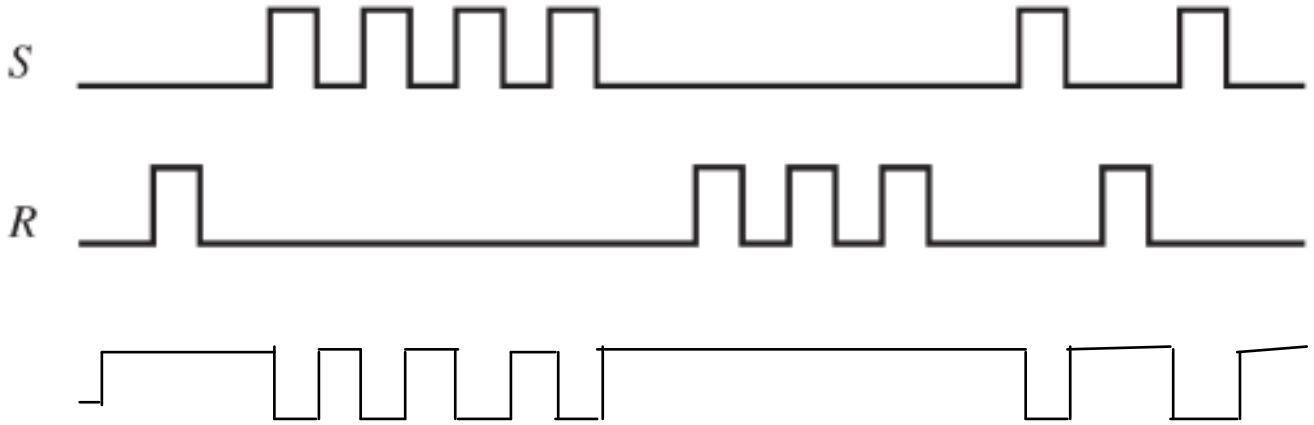


(a) Diagrama lógico

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(después que $S = 1$, $R = 0$)
0	1	1	0	
1	1	1	0	(después que $S = 0$, $R = 1$)
0	0	1	1	

(b) Tabla de verdad

Circuito flip-flop básico con compuertas NAND.



Flip flop

- Un **flip-flop** es un circuito digital secuencial que puede almacenar un bit de información (0 o 1). Es uno de los componentes fundamentales en el diseño de sistemas digitales, ya que permite mantener o "recordar" un estado hasta que se le ordene cambiar. Los flip-flops tienen dos estados estables, por lo que se les llama **biestables**.

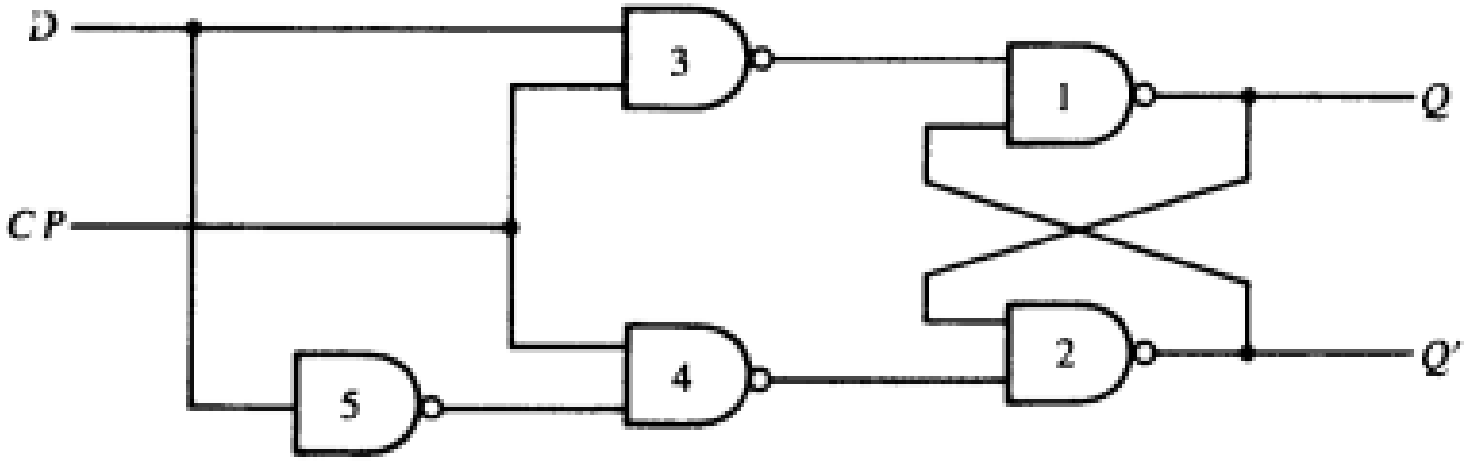
Aplicaciones

- Los flip-flops son esenciales en la creación de:
  - Registros
  - Contadores
  - Sistemas de almacenamiento temporal en circuitos digitales

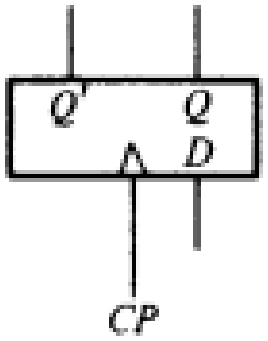
Existen varios tipos de flip-flops, pero los más comunes son:

1. **Flip-flop SR (Set-Reset)**: Tiene dos entradas, una para "Set" (establecer) y otra para "Reset" (reiniciar). Se utiliza para establecer un estado alto (1) o bajo (0) según las entradas.
2. **Flip-flop D (Data)**: Almacena el valor de una entrada de datos cuando recibe un pulso de reloj. El valor en la salida refleja el valor en la entrada cuando se activa el reloj.
3. **Flip-flop JK**: Es una variación del flip-flop SR que resuelve algunas ambigüedades del flip-flop SR. Permite alternar entre 0 y 1 cuando ambas entradas están activas.
4. **Flip-flop T (Toggle)**: Cambia de estado (de 0 a 1 o de 1 a 0) cada vez que recibe un pulso de reloj.

Flip Flop D



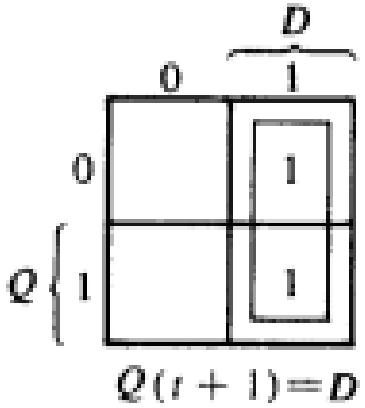
(a) Diagrama lógico con compuertas NAND



(b) Símbolo gráfico

Q	D	$Q(t+1)$
0	0	0
0	1	1
1	0	0
1	1	1

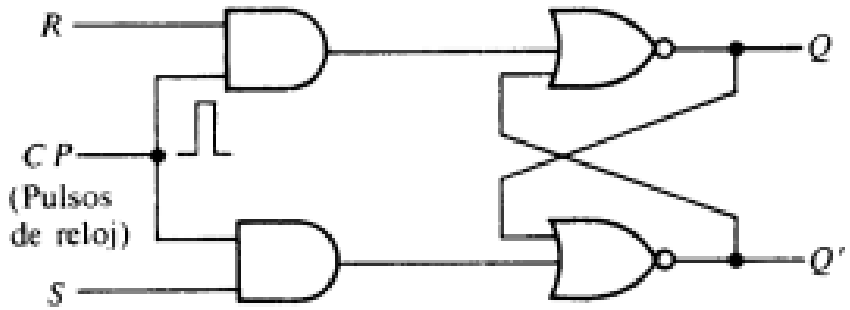
(c) Tabla característica



(d) Ecuación característica

Flip-flop D con pulsos de reloj.

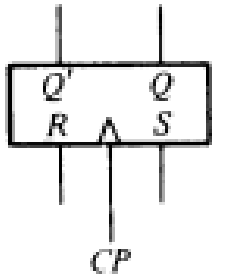
Flip Flop RS



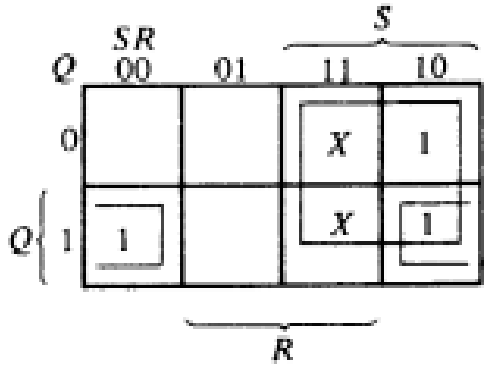
(a) Diagrama lógico

Q	S	R	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminado
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminado

(c) tabla característica



(b) símbolo gráfico



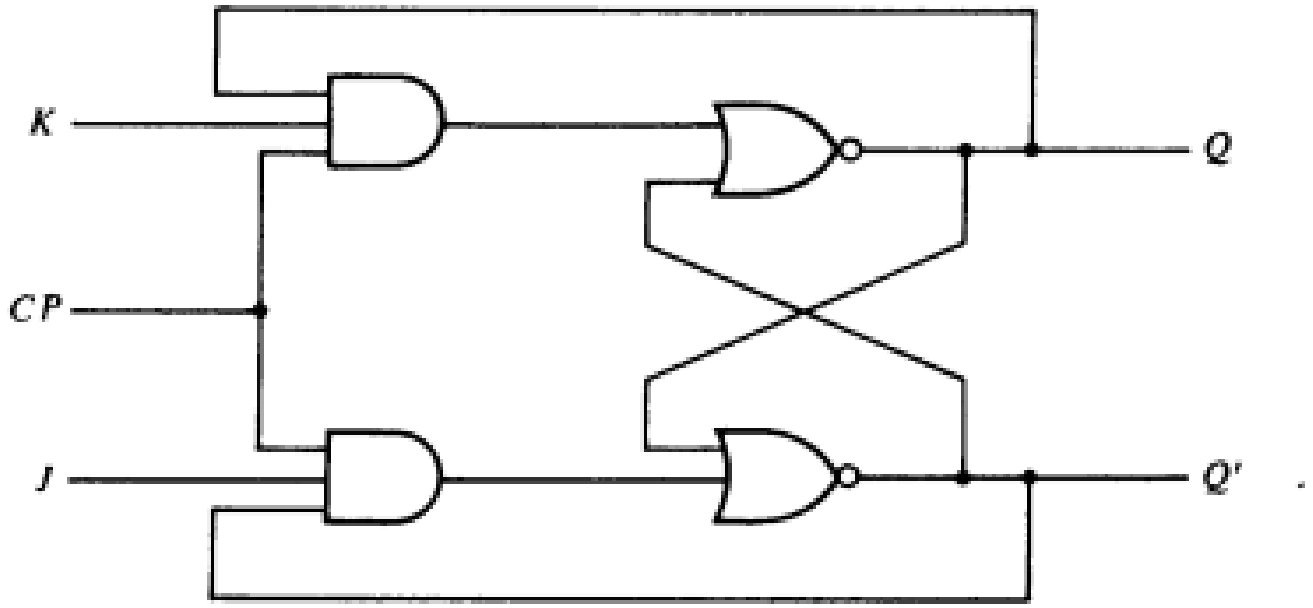
$$Q(t+1) = S + R'Q$$

$$SR = 0$$

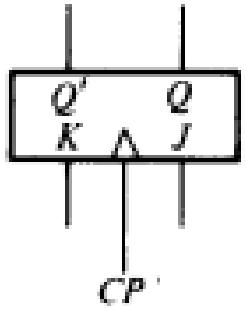
(d) ecuación característica

Flip-flop *RS* con pulsos de reloj.

Flip Flop JK



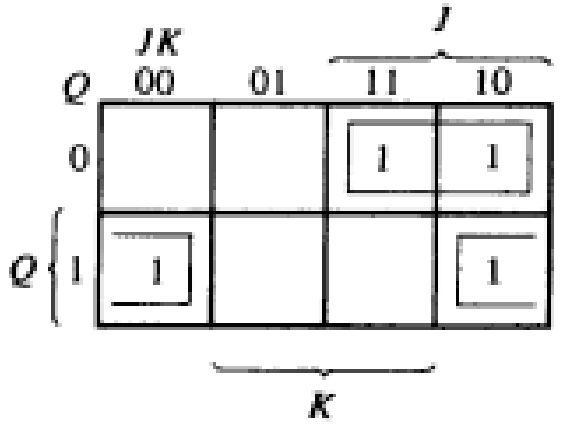
(a) Diagrama lógico



(b) Símbolo gráfico

Q	J	K	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

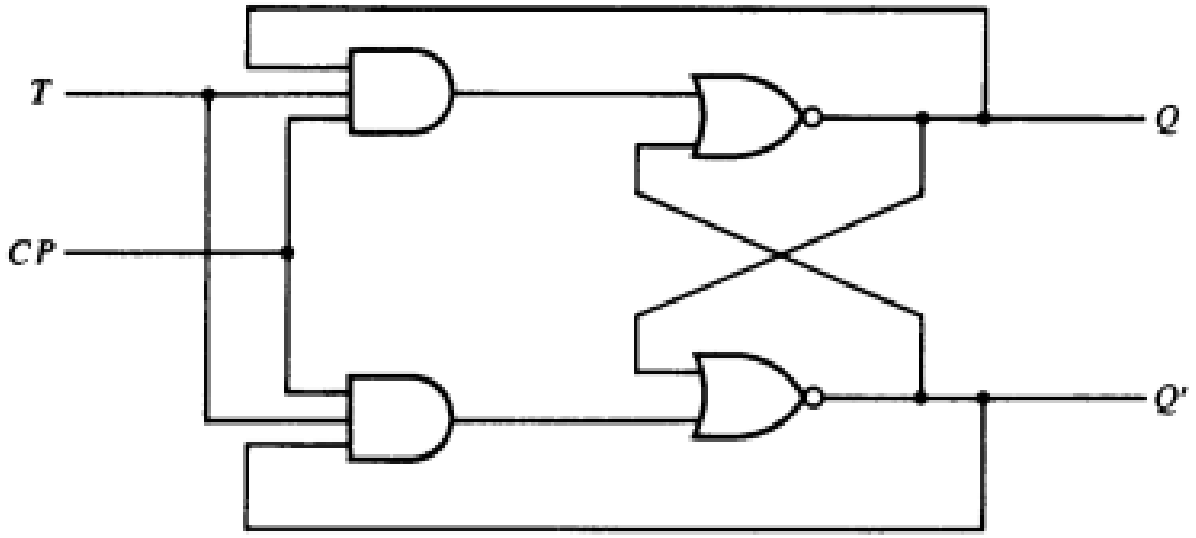
(c) Tabla característica



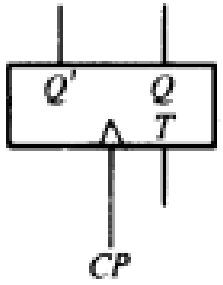
$$Q(t+1) = JQ' + K'Q$$

(d) Ecuación característica

Flip Flop T



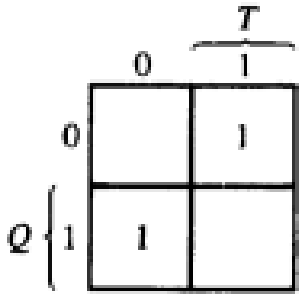
(a) Diagrama lógico



(b) Simbolo gráfico

Q	T	$Q(t+1)$
0	0	0
0	1	1
1	0	1
1	1	0

(c) Tabla característica

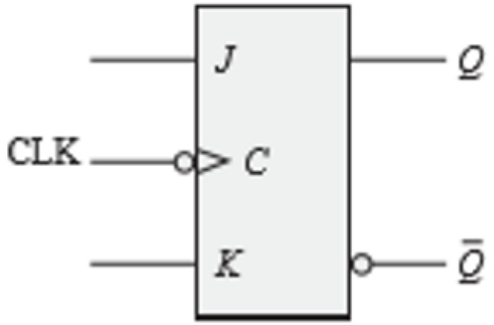


$$Q(t+1) = TQ' + T'Q$$

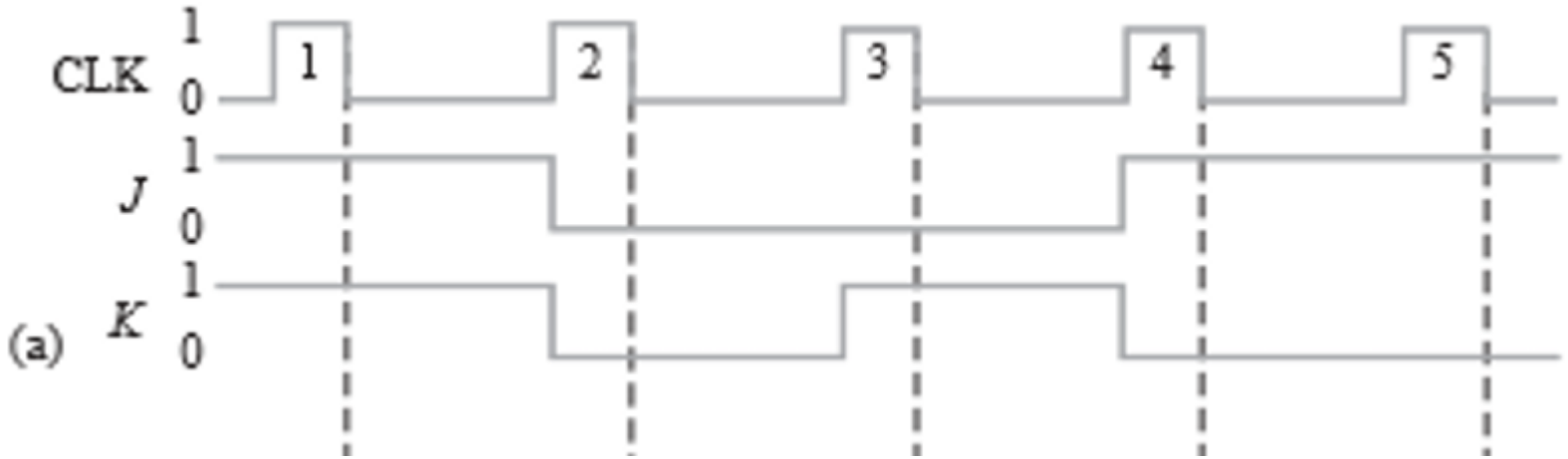
(d) Ecuación característica

Flip-flop T con pulsos de reloj.

Ejemplo



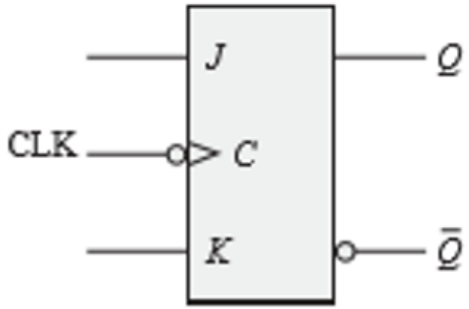
- Las formas de onda de la entrada de la figura (a) se aplican a las entradas J, K y de reloj, tal y como se muestra. Determinar Q suponiendo que el flip flop se encuentra inicialmente en estado de RESET.



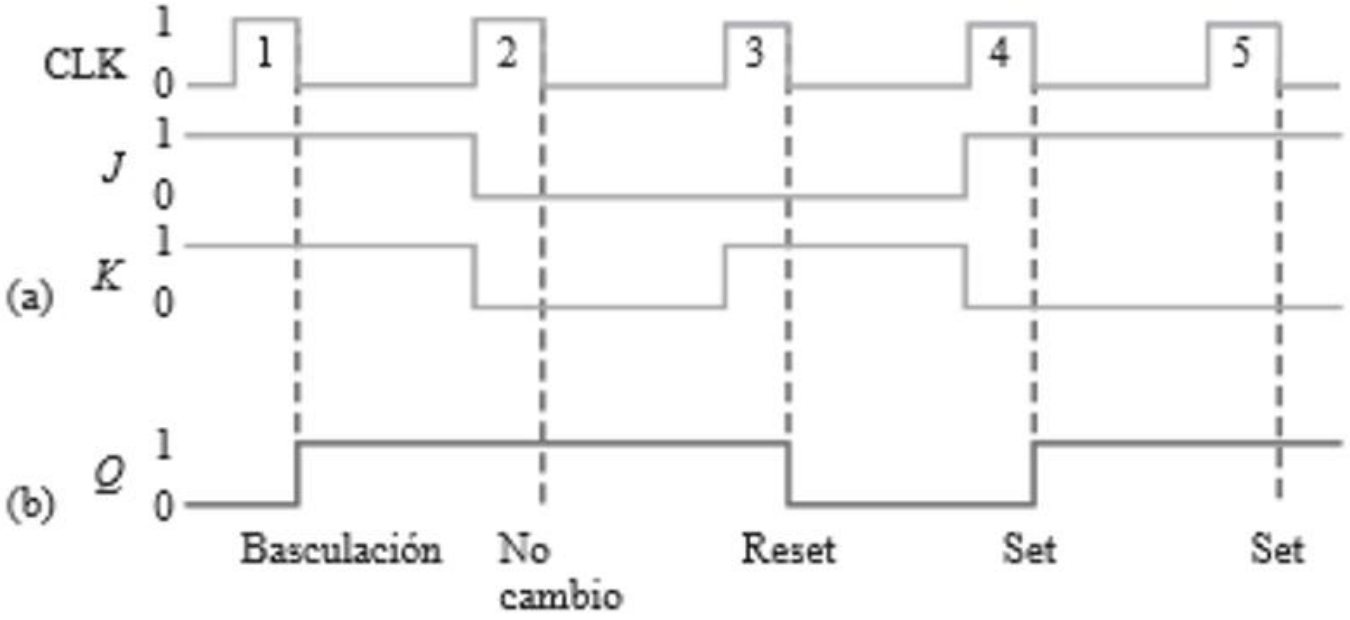
Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Ejemplo

- Las formas de onda de la entrada de la figura se aplican a las entradas J,K y de reloj, tal y como se muestra. Determinar Q suponiendo que el flip flop se encuentra inicialmente en estado de RESET.



Q	J	K	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

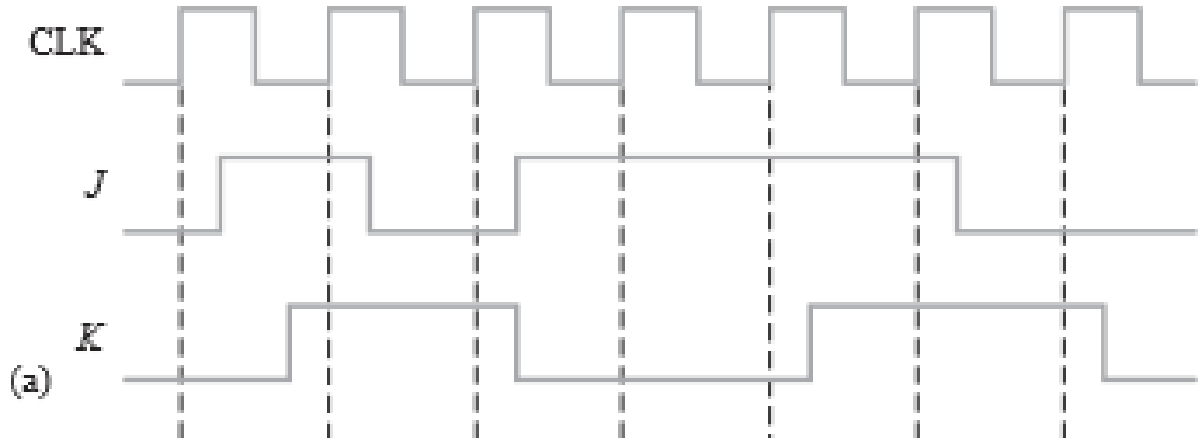


Tarea

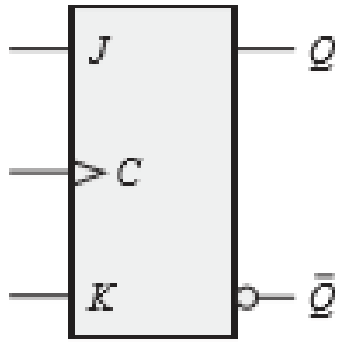
- Las formas de onda de la Figura (a) se aplican al flip-flop que se muestra. Determinar la salida Q, comenzado en el estado RESET.

Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Tabla característica



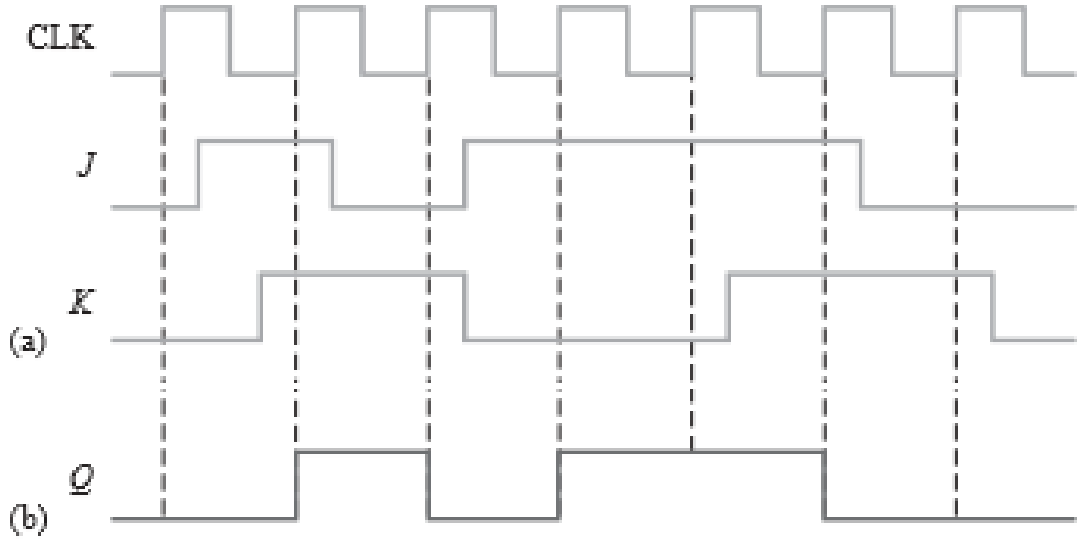
(a)

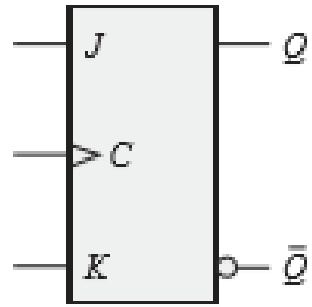


Solución

Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Tabla característica

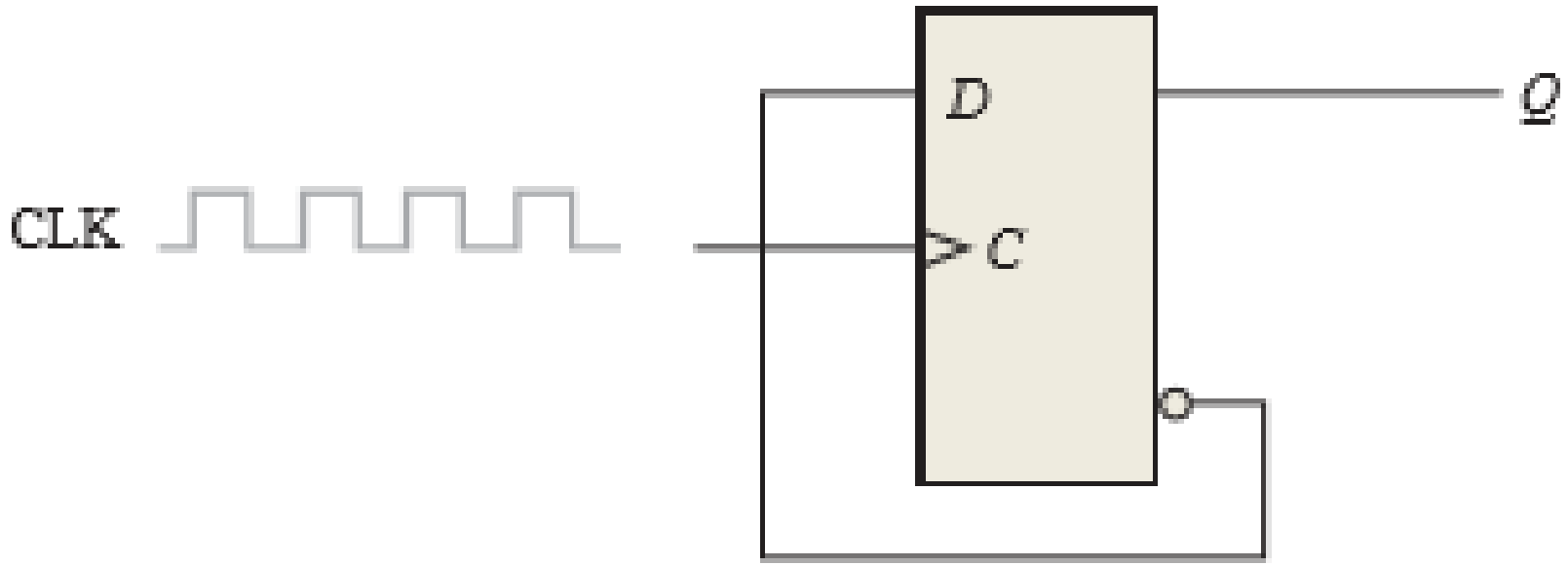




- La salida Q toma el estado determinado por los estados de las entradas J y K en el flanco positivo (flanco de disparo) del impulso de reloj. Si se produce un cambio en J o en K después del disparo del reloj éste no tiene efecto en la salida, como se muestra en la Figura (b).

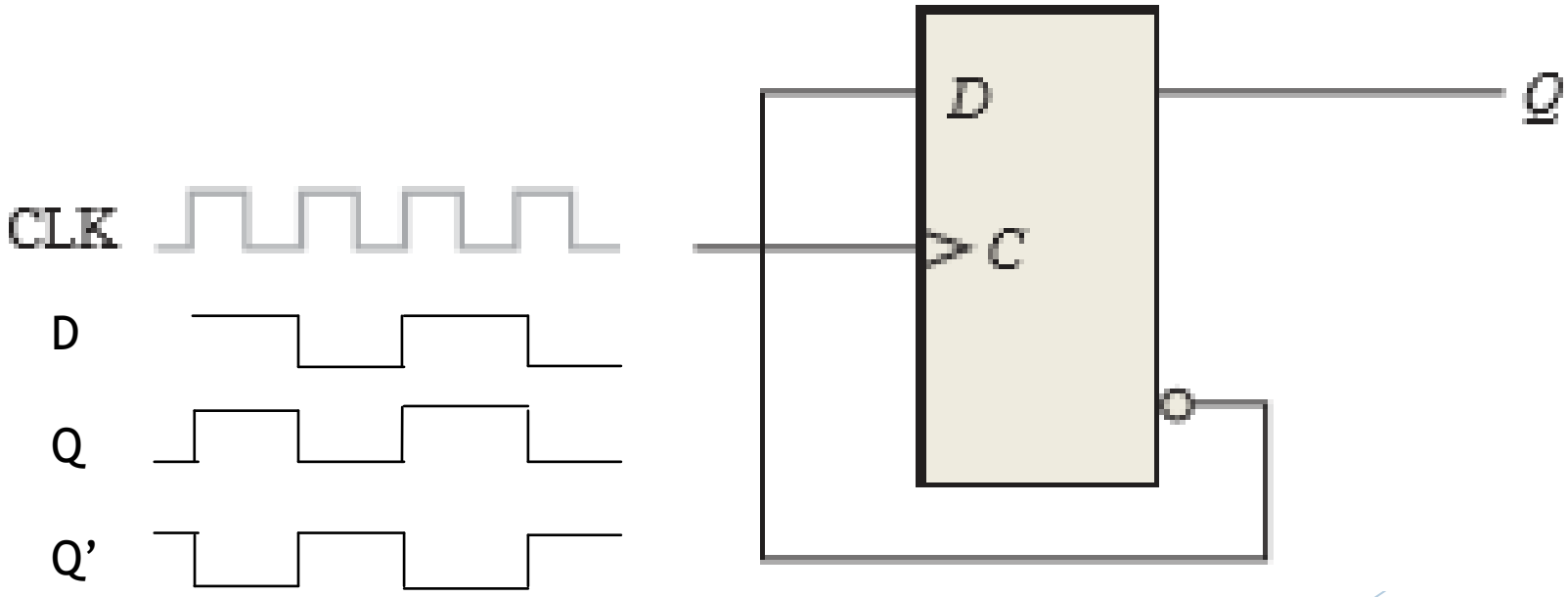
Ejercicio

- Un flip-flop D se encuentra conectado como se muestra en la Figura. Determinar la salida Q en función del reloj. ¿Cuál es la función que realiza este dispositivo?



Solución

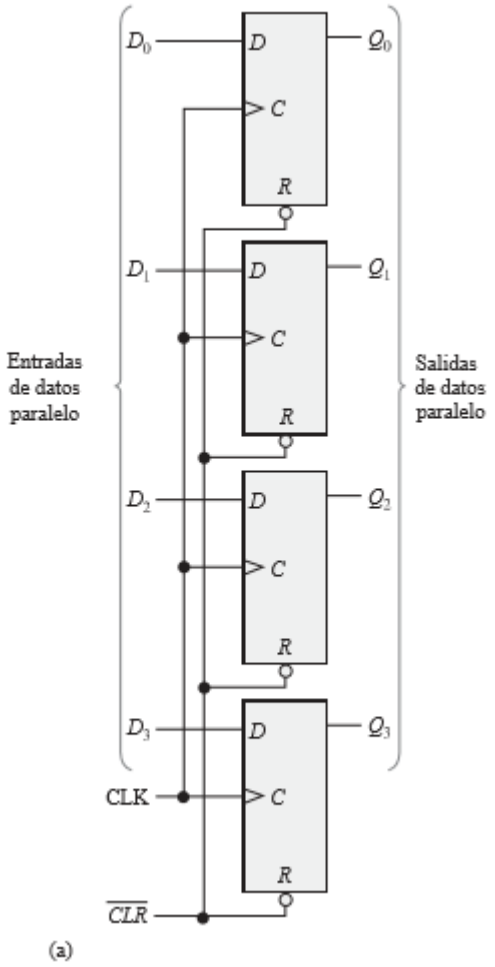
- Un flip-flop D se encuentra conectado como se muestra en la Figura. Determinar la salida Q en función del reloj. ¿Cuál es la función que realiza este dispositivo?



Aplicaciones

Almacenamiento de datos

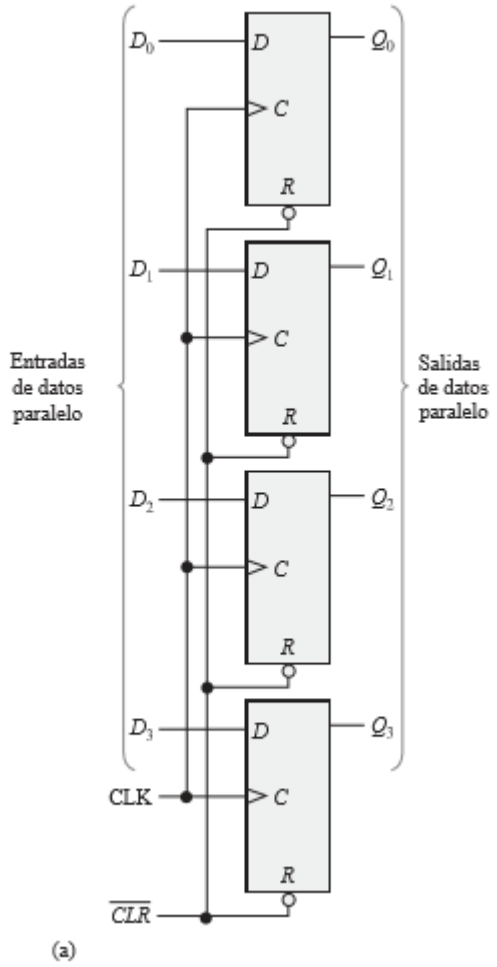
- Uno de los requisitos más comunes de los sistemas digitales consiste en almacenar de forma simultánea una serie de bits de datos, procedentes de varias líneas paralelas, en un grupo de flip-flops.

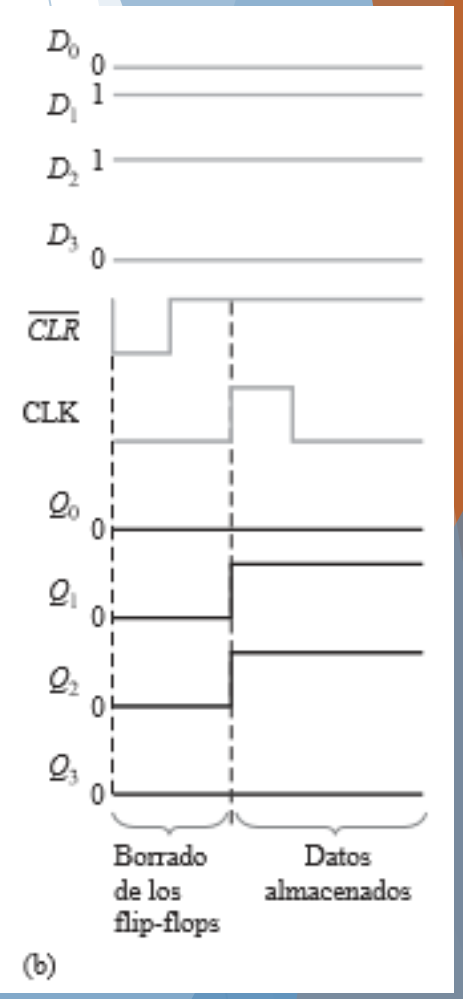


- Este proceso se ilustra en la Figura (a), utilizando mediante el mismo impulso del reloj.

- En este ejemplo, se utilizan cuatro flip-flops. Cada una de las cuatro líneas paralelas de datos se conecta a la entrada D de un flip-flop.

- Las entradas de reloj de los flip-flops se conectan juntas, de forma que los flip-flops son disparados por flanco positivo, por lo que los datos de las entradas D se almacenan simultáneamente en los flip-flops con el flanco positivo de reloj, como se indica en el diagrama de tiempos de la Figura(b). Además, las entradas de puesta a cero asíncronas (R) se conectan a una línea común, que inicialmente pone a cero a todos los flip-flops.

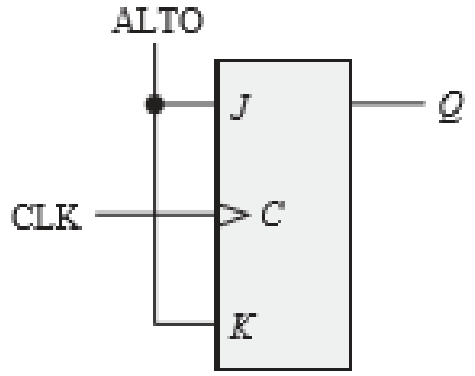


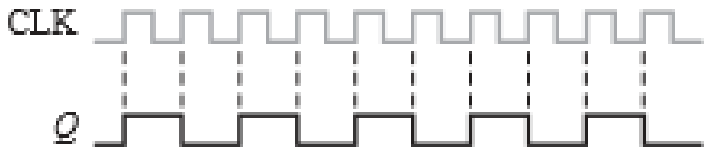


- Este grupo de cuatro flip-flops es un ejemplo de un registro básico utilizado para almacenamiento de datos. En los sistemas digitales, los datos se almacenan normalmente en grupos de bits (usualmente ocho o múltiplos de ocho), que representa números, códigos u otras informaciones.

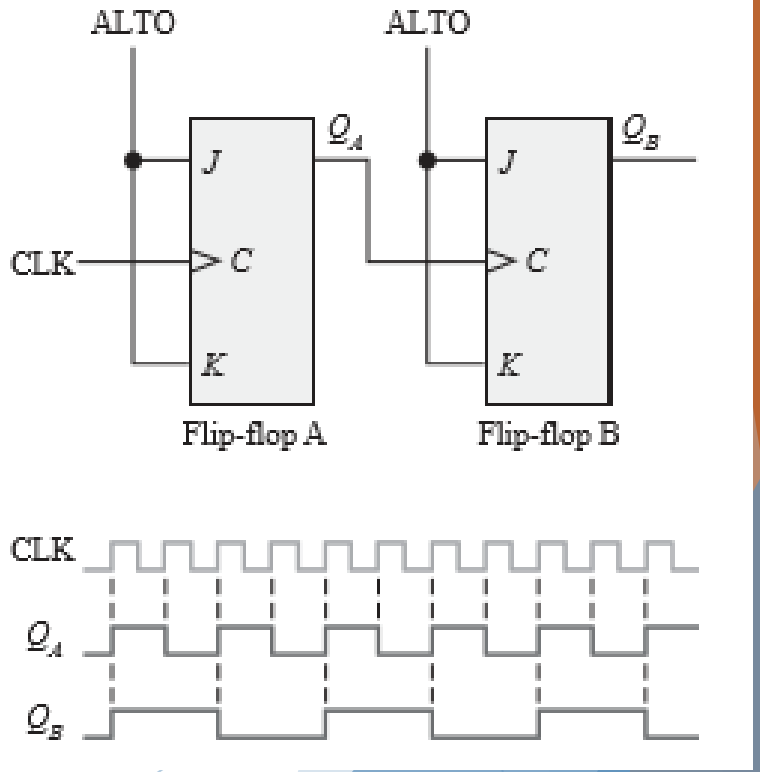
División de frecuencia

- Otra de las aplicaciones de un flip-flop es la división (reducción) de frecuencia de una señal periódica.
- Cuando se aplica un tren de impulsos a la entrada de reloj de un flip-flop J-K conectado en modo de basculación (J = K = 1), la salida Q es una señal cuadrada que tiene una frecuencia igual a la mitad de la que tiene la señal de reloj. Por tanto, se puede utilizar un único flip-flop como un divisor por 2, como muestra la Figura.
- Como puede verse, el flip-flop cambia de estado en cada flanco de disparo del impulso de reloj (flancos positivos en este caso). Esto da lugar a una salida que varía a la frecuencia mitad de la señal de reloj.



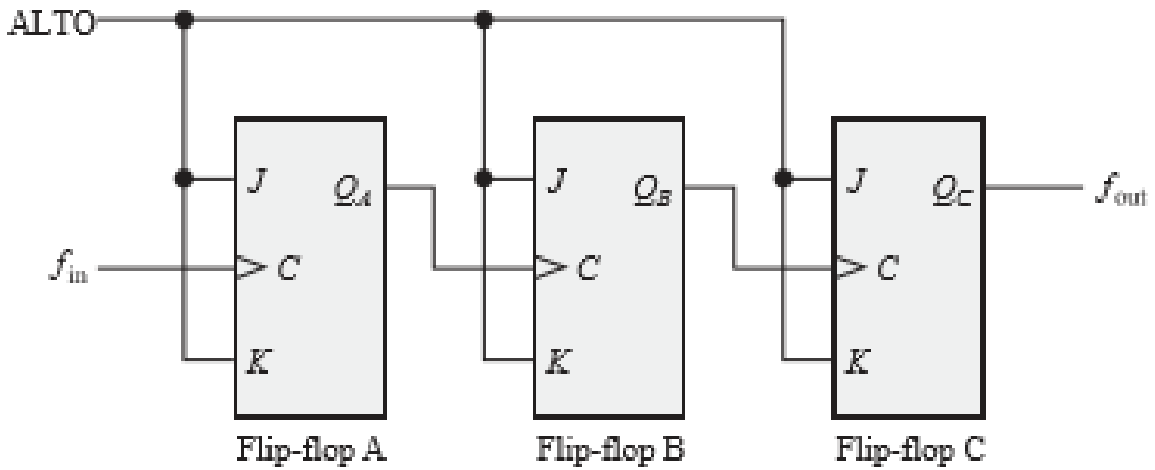


- Se pueden conseguir divisiones sucesivas de la frecuencia del reloj conectando la salida de un flip-flop a la entrada de reloj de un segundo flip-flop, como se muestra en la Figura. El flip-flop B divide la frecuencia de la salida QA por 2. La salida QB es, por tanto, un cuarto de la frecuencia de la señal de reloj original. En estos diagramas de tiempo no se muestran los retardos de propagación. Si se conectan varios flip-flops de esta manera, se puede conseguir una división de frecuencias de 2^n, donde n es el número de flip-flops. Por ejemplo, tres flip-flops dividen la frecuencia de reloj por $2^3 = 8$; cuatro flip-flops dividen la frecuencia de reloj por $2^4 = 16$, y así sucesivamente.

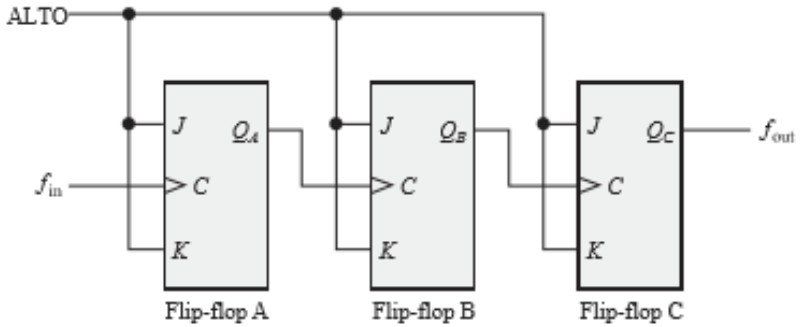


Ejercicio

- Desarrollar la forma de onda fout para el circuito de la Figura, cuando se aplica una señal cuadrada de 8 kHz en la entrada de reloj del flip-flop A.



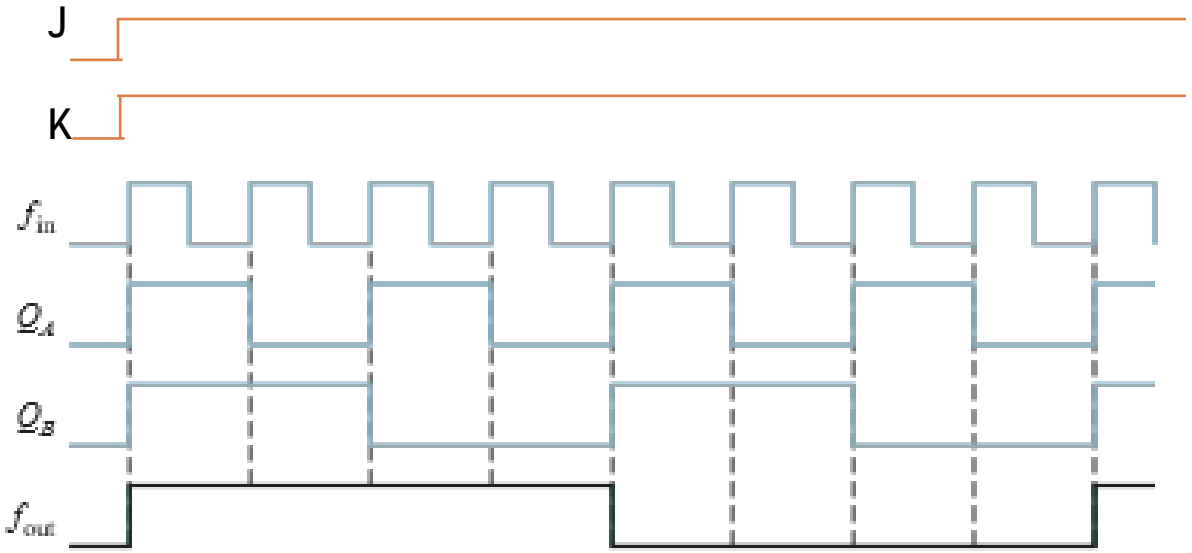
Solución



Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Tabla característica

- Los tres flip-flops están conectados para dividir la frecuencia de entrada por ocho ($2^3 = 8$) y la señal fout se muestra en la Figura. Dado que se trata de flip-flops disparados por flanco positivo, las salidas cambian durante el flanco positivo del reloj. Hay un impulso de salida por cada ocho impulsos de entrada, de forma que la frecuencia de salida es 1 kHz. Las señales QA y QB también se muestran.



Los cálculos

- El divisor de frecuencia es un componente simple, cuyo objetivo es reducir la frecuencia de ent50. Éste se implementa con ayuda del factor de escalamiento y un contador. Primeramente, el factor de escalamiento es la relación entre la frecuencia de ent50 y la frecuencia de sal200 deseada:

$$Escala = fent50/fsal200$$

$$Escala = 50MHz/200Hz = 250,000$$

- Asumiendo que tenemos una frecuencia de *50MHz* y deseamos una sal200 de *200Hz*, tenemos que:

Escala = 50MHz/200Hz = 250000

- Por lo tanto, el contador para el divisor de frecuencia tiene como función generar la señal de sal200 de *200Hz* cada 250000 ciclos.
- Dividir entre dos escala

250000/2=125000

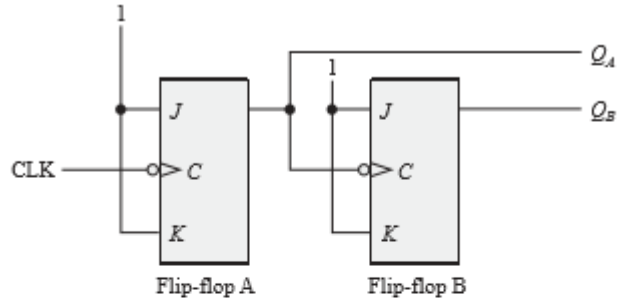
125000-1=124999

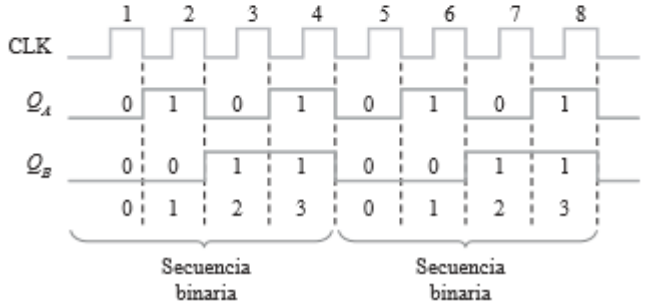
Divisor de frecuencia con rst asíncrono.1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clk200Hz is
    Port (
        ent50: in  STD_LOGIC;
        rst  : in  STD_LOGIC;
        sal200 : out STD_LOGIC
    );
end clk200Hz;

architecture Behavioral of clk200Hz is
    signal temporal: STD_LOGIC;
    signal contador: integer range 0 to
    124999 := 0;
begin
    process (rst, ent50) begin
        if (rst = '1') then
            temporal <= '0';
            contador <= 0;
        elsif rising_edge(ent50) then
            if (contador = 124999) then
                temporal <= NOT(temporal);
                contador <= 0;
            else
                contador <= contador+1;
            end if;
        end if;
    end process;

    sal200 <= temporal;
end Behavioral;
```

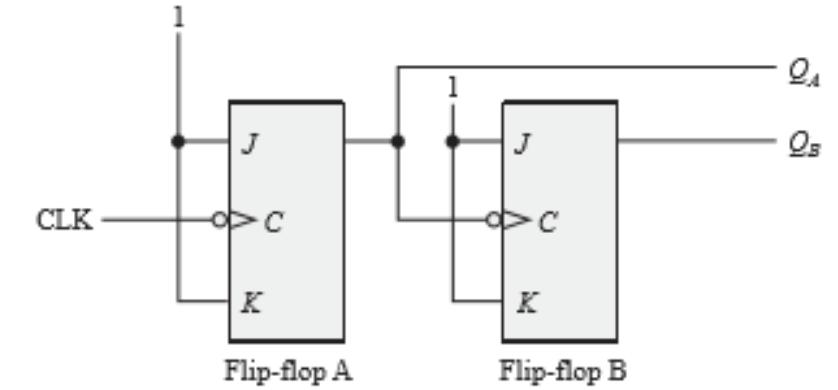
Contadores

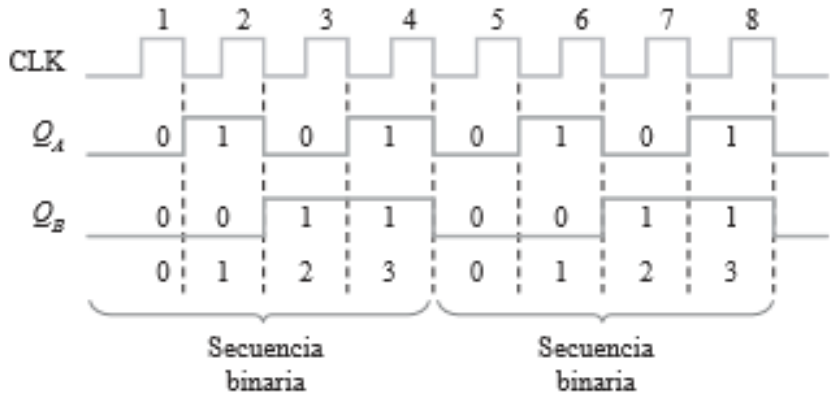
- El concepto se ilustra en la Figura.
- Los flip-flops son de tipo J-K disparados por flanco negativo. Ambos flip-flops se encuentran inicialmente en estado RESET.
- El flip-flop A bascula en las transiciones negativas de cada impulso de reloj. La salida Q del flip-flop Adispara el flip-flop B, de manera que siempre que QA realiza una transición de nivel ALTO a nivel BAJO, el flip-flop B bascula. Las señales resultantes QA y QB se muestran en la figura.





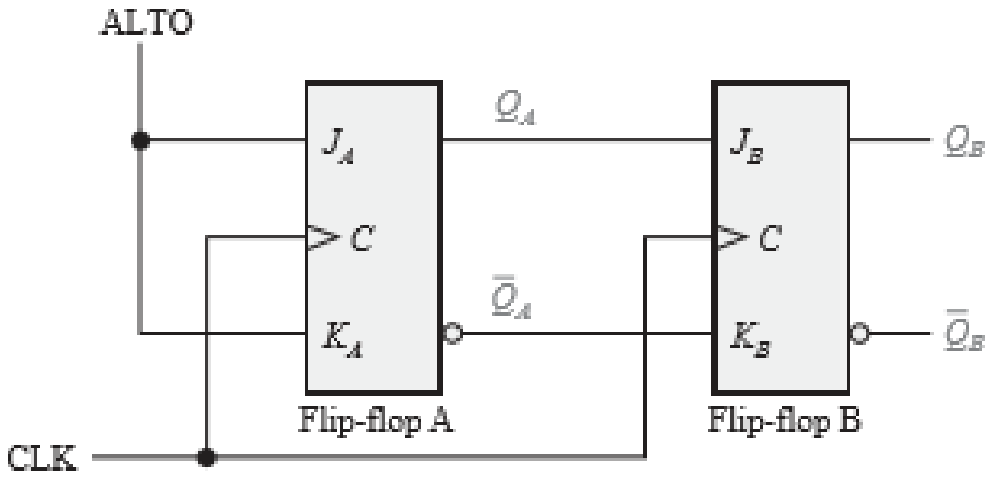
- Observe la secuencia de QA y QB en la Figura. Previamente al impulso de reloj 1, QA = 0 y QB = 0; tras el impulso de reloj 1, QA = 1 y QB = 0; después del impulso de reloj 2, QA = 0 y QB = 1, y tras el impulso de reloj 3, QA = 1 y QB = 1.
- Si se toma QA como el bit menos significativo, se produce una secuencia binaria de dos bits a medida que se disparan los flip-flops.
- Esta secuencia binaria se repite cada cuatro impulsos de reloj, como se muestra en el diagrama de tiempos. Por tanto, los flip-flops siguen una secuencia de 0 a 3 (00, 01, 10, 11) y luego vuelven a 0 para comenzar la misma secuencia de nuevo.



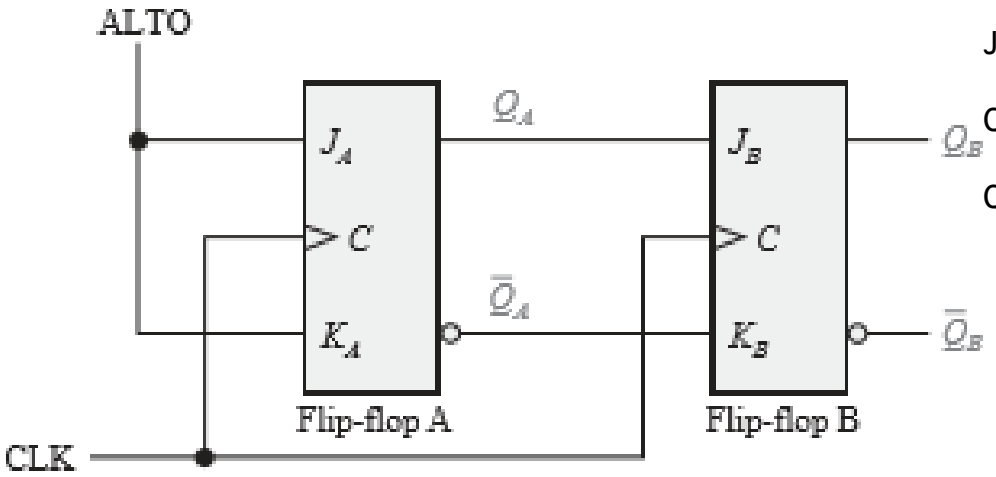


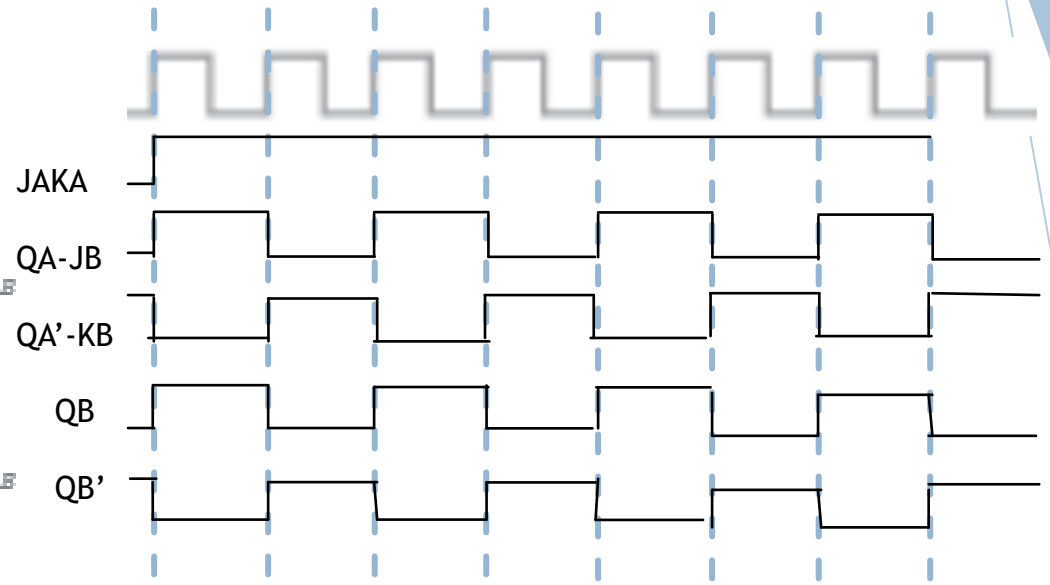
Ejercicio

- Para el circuito de la Figura, desarrollar un diagrama de tiempos para ocho impulsos de reloj, mostrando las salidas QA y QB en función del reloj.



Solución





Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Tabla caracteristica

SECCIÓN 7.2 Flip-flops disparados por flanco

8. En la Figura 7.75 se muestran dos flip-flops S-R disparados por flanco. Si las entradas son las que se indican, dibujar la salida Q de cada flip-flop en función de la señal de reloj y explicar la diferencia entre los dos. Los flip-flops se encuentran inicialmente en estado RESET.

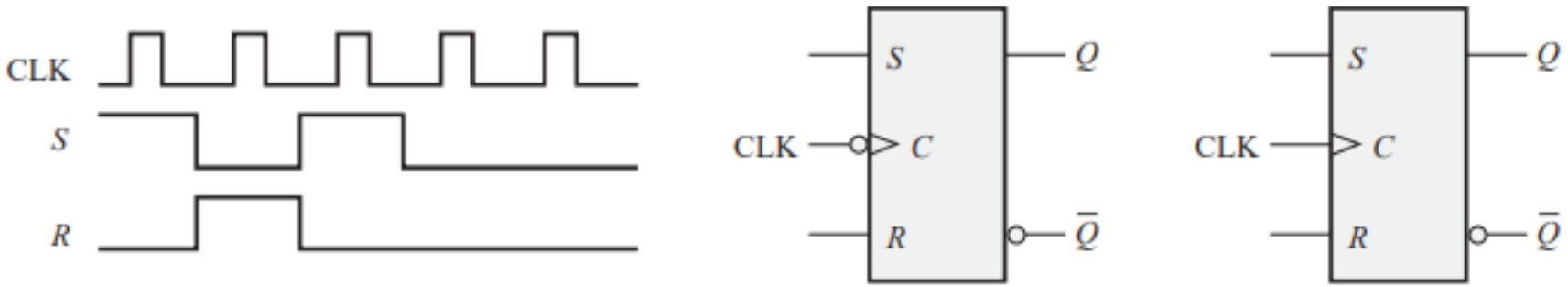
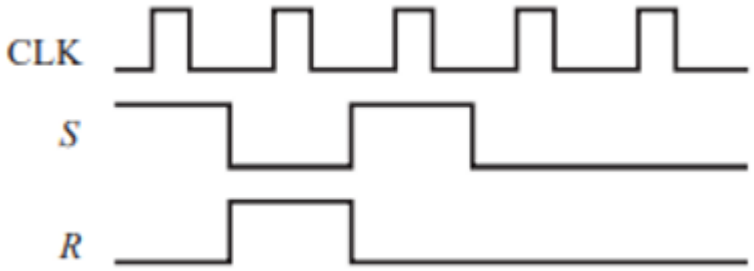
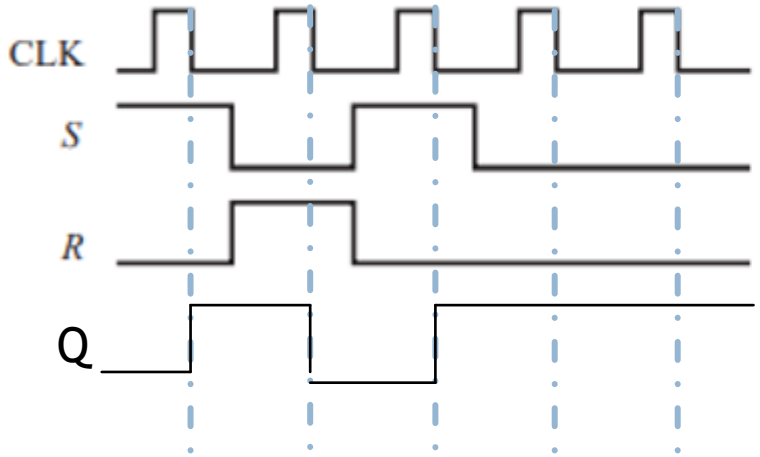
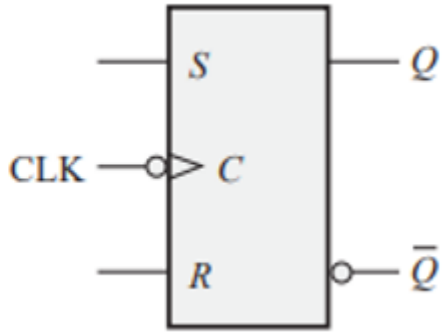


FIGURA 7.75

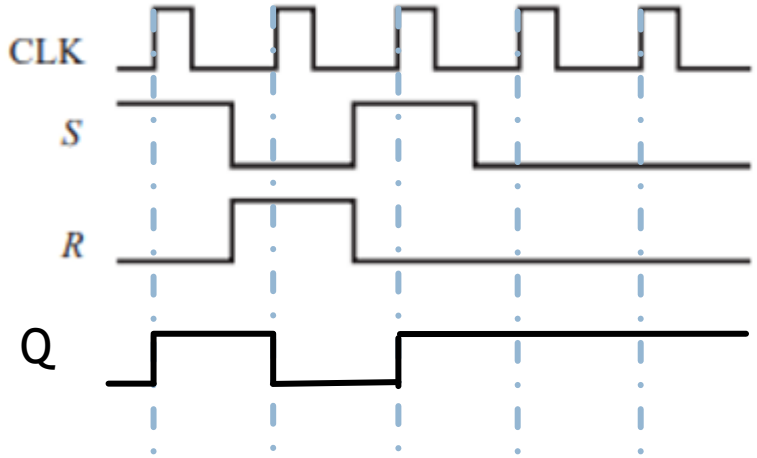
Q	S	R	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminado
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminado

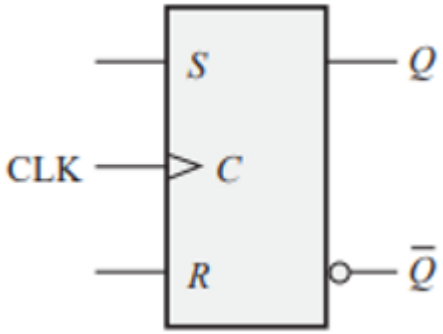






Q	S	R	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminado
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminado





Q	D	$Q(t+1)$
0	0	0
0	1	1
1	0	0
1	1	1

10. Dibujar la salida Q en función del reloj para un flip-flop D cuyas entradas son las que se muestran en la Figura 7.77. Suponer disparo por flanco positivo y que Q se encuentra inicialmente a nivel BAJO.

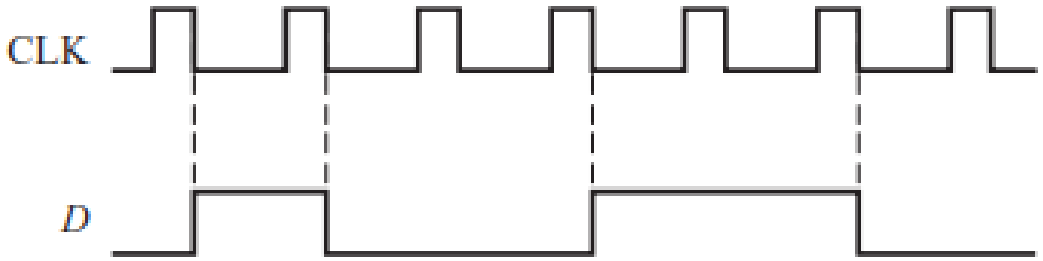


FIGURA 7.77

11. Resolver el Problema 10 para las entradas de la Figura 7.78.

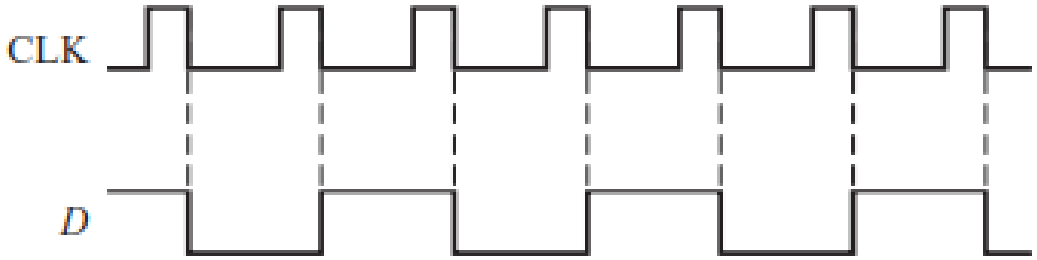


FIGURA 7.78

10. Dibujar la salida Q en función del reloj para un flip-flop D cuyas entradas son las que se muestran en la Figura 7.77. Suponer disparo por flanco positivo y que Q se encuentra inicialmente a nivel BAJO.

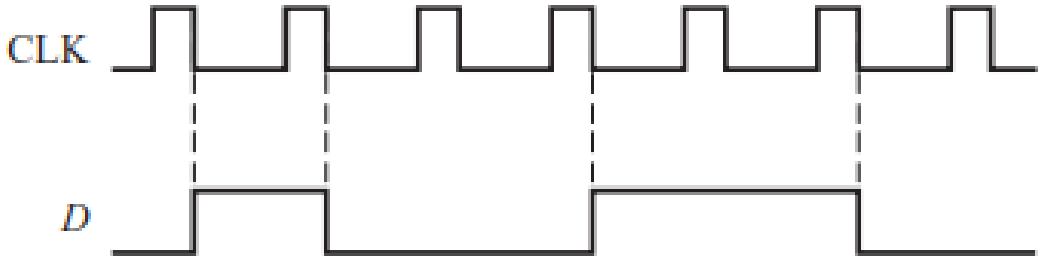


FIGURA 7.77

11. Resolver el Problema 10 para las entradas de la Figura 7.78.

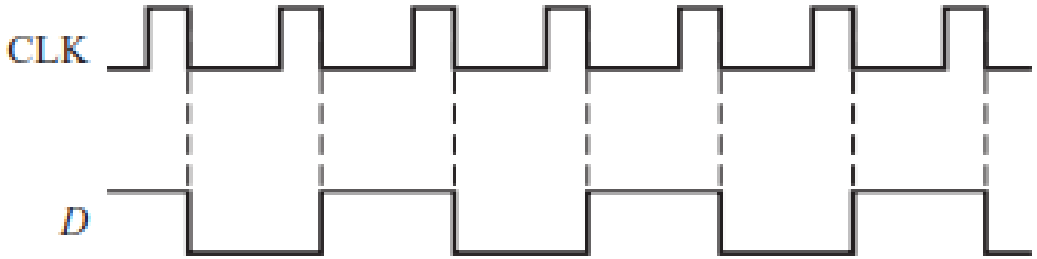
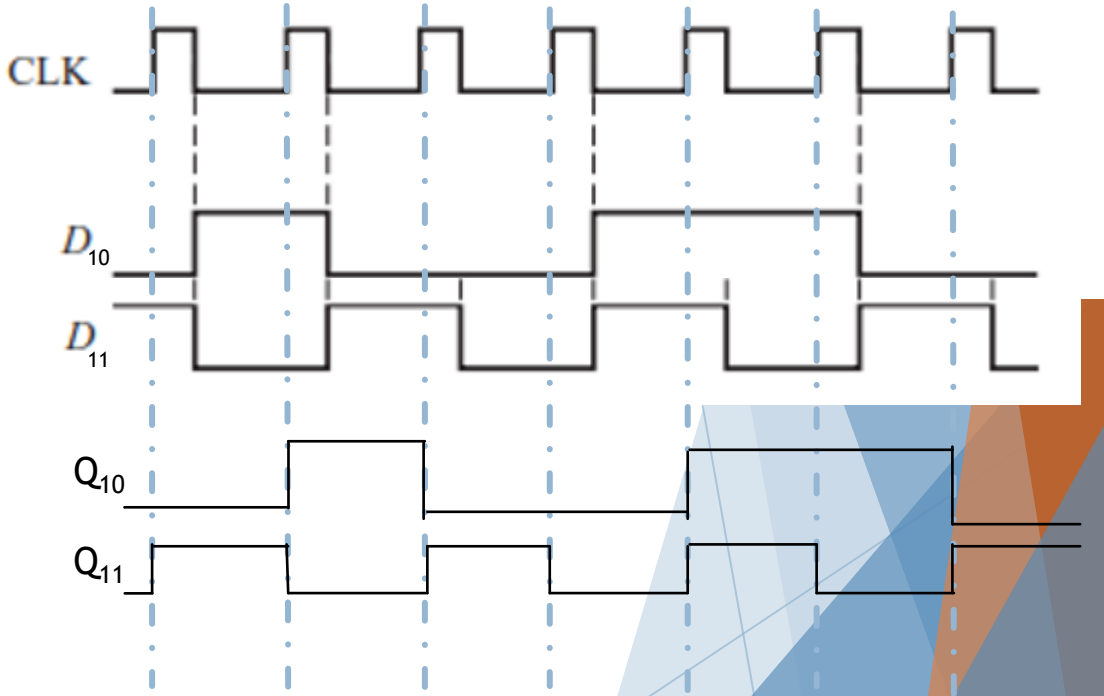


FIGURA 7.78

Q	D	$Q(t+1)$
0	0	0
0	1	1
1	0	0
1	1	1



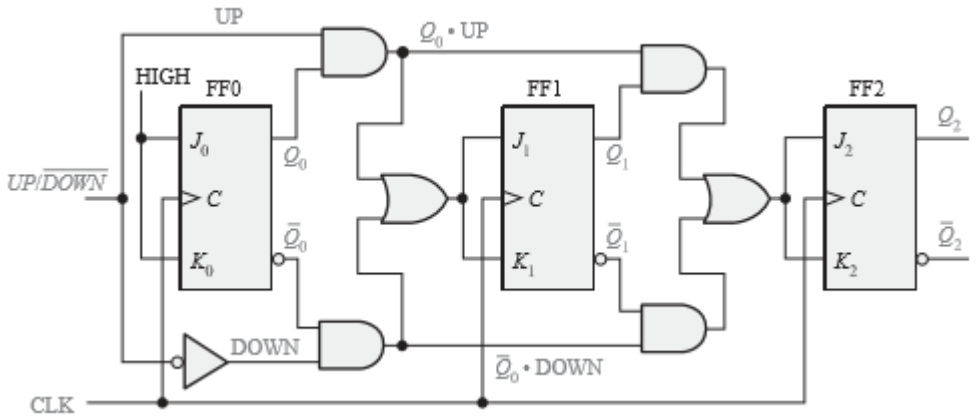
Contador ascendente descendente

- Es aquel capaz de progresar en cualquier dirección a lo largo de una cierta secuencia.
- Algunas veces también denominado contador bidireccional, puede tener cualquier secuencia de estados especificada.
- Un contador binario de 3 bits que avanza en modo ascendente a través de la secuencia (0, 1, 2, 3, 4, 5, 6, 7) y que luego puede invertirse para recorrer la secuencia en sentido contrario (7, 6, 5, 4, 3, 2, 1, 0) es un ejemplo de un modo de operación secuencial ascendente/descendente.

Contador ascendente descendente

- En general, la mayoría de los contadores ascendentes/descendentes pueden invertirse en cualquier punto de su secuencia.
- Por ejemplo, el contador binario de 3 bits se puede configurar para que realice la siguiente secuencia:

ASCENDENTE: 0, 1, 2, 3, 4, 5,
DESCENDENTE: 4, 3, 2,
ASCENDENTE: 3, 4, 5, 6, 7,
DESCENDENTE: 6, 5,
etc.



- La Tabla muestra la secuencia ascendente/descendente (up/down) completa de un contador binario de 3 bits. Las flechas indican los movimientos entre los estados del contador, tanto para el modo ASCENDENTE como para el modo Descendente.
- Un examen de Q0 para ambas secuencias, ascendente y descendente, muestra que FF0 bascula con cada impulso de reloj. Luego las entradas J0 y K0 de FF0 son:
- J0 = K0 =1
- Para la secuencia ascendente, Q1 cambia de estado en el siguiente impulso de reloj cuando Q0 = 1. Para la secuencia descendente, Q1 cambia en el siguiente impulso de reloj cuando Q0 = 0.

Impulso de reloj	ASC.	Q_2	Q_1	Q_0	DESC.
0	↑	0	0	0	↓
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

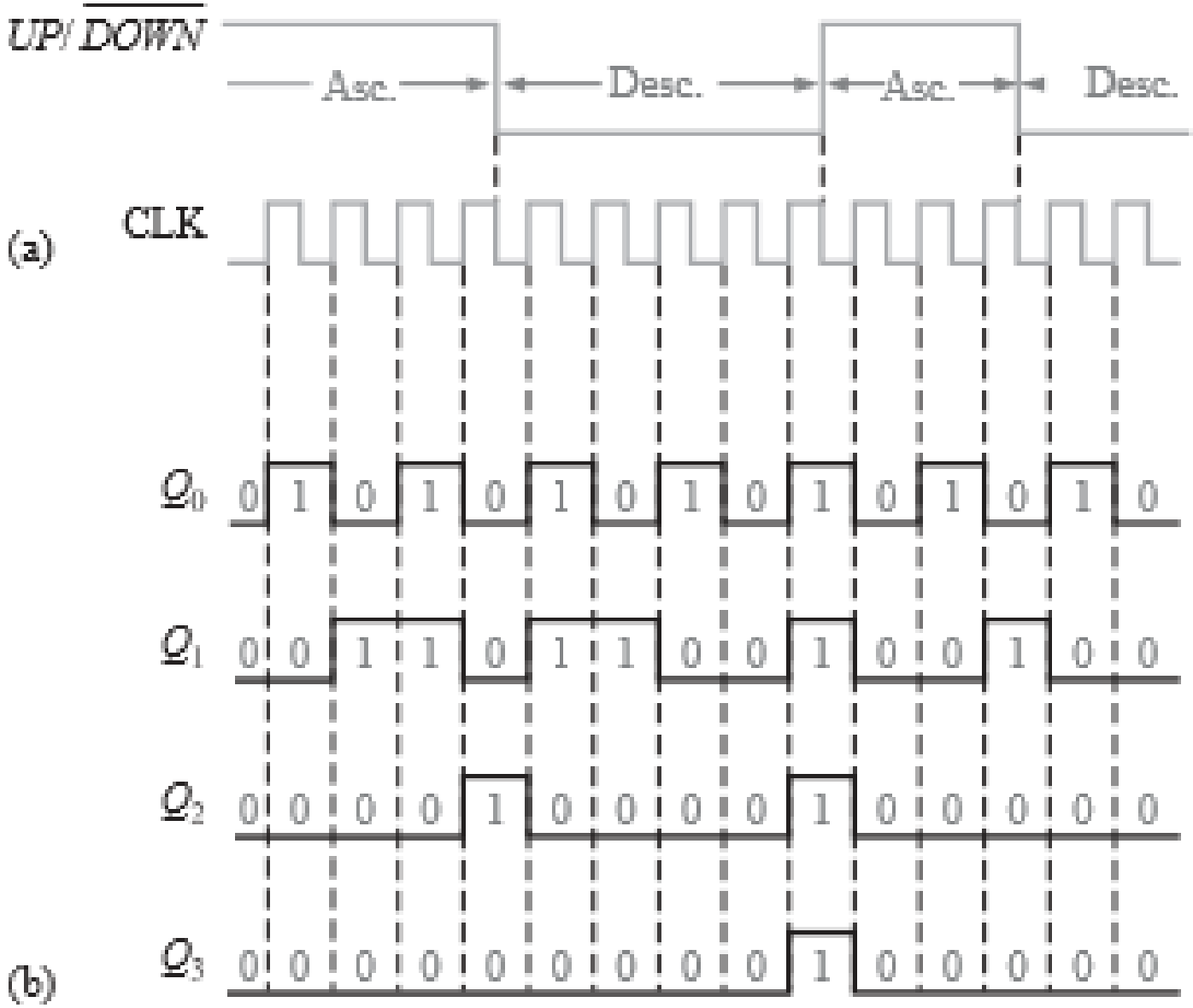
- Por tanto, las entradas J1 y K1 de FF1 tienen que ser igual a 1, para las condiciones expresadas en la siguiente ecuación:

$$J_1 = K_1 = (Q_0 \cdot \mathrm{UP}) + (\bar{Q}_0 \cdot \mathrm{DOWN})$$

- Para la secuencia ascendente, Q2 cambia de estado en el siguiente impulso de reloj cuando Q0 = Q1 = 1.
- Para la secuencia descendente, Q2 cambia en el siguiente impulso de reloj cuando Q0 = Q1 = 0. Por tanto, las entradas J2 y K2 de FF2 tienen que ser igual a 1, para las condiciones expresadas en la siguiente ecuación:

$$J_2 = K_2 = (Q_0 \cdot Q_1 \cdot \mathrm{UP}) + (\bar{Q}_0 \cdot \bar{Q}_1 \cdot \mathrm{DOWN})$$

- Cada una de las condiciones para las entradas J y K de cada flip-flop produce una basculación en el punto apropiado de la secuencia del contador. La Figura muestra una implementación básica de un contador binario de 3 bits ascendente/ descendente, utilizando las ecuaciones lógicas que acabamos de desarrollar para las entradas J y K de cada flip-flop. Observe que, la entrada de control está a nivel ALTO cuando trabaja en modo ascendente y a nivel BAJO cuando trabaja en modo descendente.


UP/ DOWN
Asc.
Desc.
Asc.
Desc.
(a)
CLK
Q_0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
Q_1 0 0 1 1 0 1 1 0 0 1 0 0 1 0 0
Q_2 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0
Q_3 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
(b)

Preset (PRE o PR)

- **Función**: La entrada Preset establece la salida **Q** en **1** de forma directa, independientemente del estado de la entrada de reloj o de las entradas J y K.
- **Modo de activación**: Generalmente, el Preset se activa cuando se le aplica un pulso bajo (lógica 0), en un flip-flop con entradas Preset y Clear activas en bajo. Sin embargo, en algunos flip-flops puede estar activo en alto, dependiendo del diseño del circuito.
- **Comportamiento**:
  - Si el Preset está activado (por ejemplo, en bajo si es activo en bajo), la salida Q se establece en **1**.
  - La salida complementaria **Q'**
  - se establecerá en **0**.

Clear (CLR o CL)

- **Función**: La entrada Clear pone la salida **Q** en **0** de forma directa, sin importar el estado de las otras entradas o el reloj.

- **Modo de activación**: Al igual que el Preset, generalmente se activa con un pulso bajo (lógica 0), aunque puede variar según el diseño.

- **Comportamiento**:
  - Si el Clear está activado (por ejemplo, en bajo si es activo en bajo), la salida Q se pone en **0**.
  - La salida **Q'** se establecerá en **1**.

Condiciones Especiales: Uso Simultáneo de Preset y Clear

- Si ambos están activados al mismo tiempo (por ejemplo, ambos en bajo), se genera una condición inválida o "indefinida" en muchos flip-flops JK. En esta situación, la salida no se define de forma confiable y puede quedar en un estado ambiguo.

- En la mayoría de los flip-flops JK, esta condición debe evitarse porque puede causar un mal funcionamiento en el circuito.

Tabla de comportamiento

Preset	Clear	Q	Q’
0	1	1	0
1	0	0	1
0	0	Indefinido	Indefinido
1	1	Depende de J,K y clk	

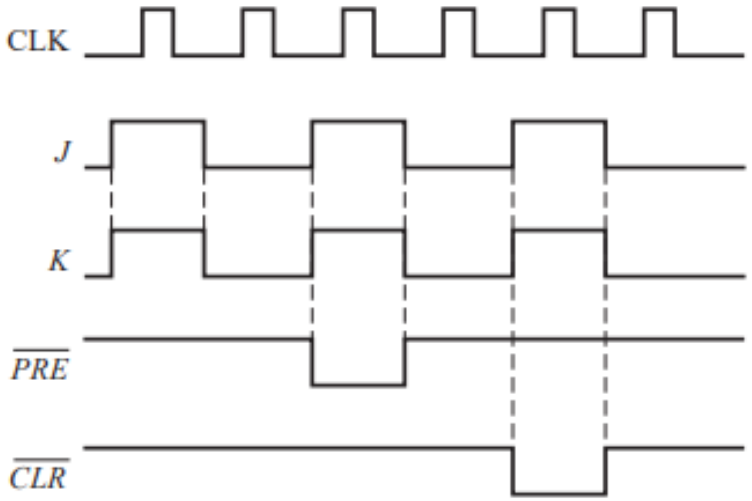
Aplicaciones del uso de Preset y Clear

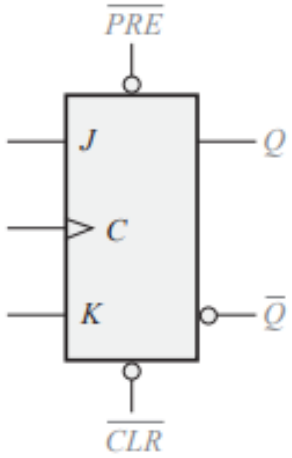
- **Inicialización de circuitos**: Los Preset y Clear permiten que el flip-flop se configure en un estado específico al encender el circuito, asegurando que los elementos de memoria tengan un valor definido.

- **Sincronización**: Son útiles en sistemas donde es necesario sincronizar todos los flip-flops a un estado conocido antes de iniciar operaciones regulares.

Ejercicio

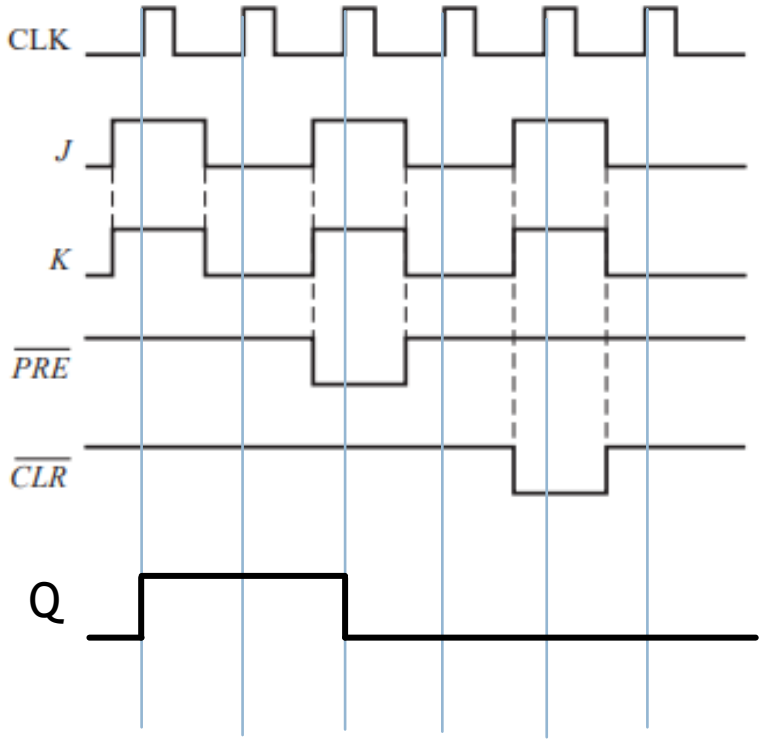
- Determinar la salida Q en función del reloj si las señales que se muestran en la Figura se aplican a las entradas de un flip-flop J-K. Suponer que Q se encuentra inicialmente a nivel BAJO.

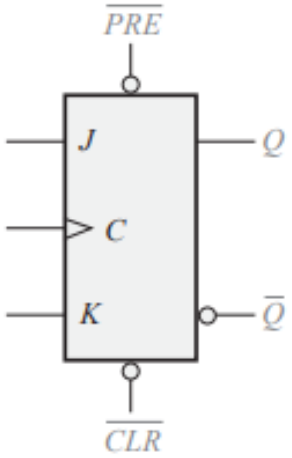




Solución

- Determinar la salida Q en función del reloj si las señales que se muestran en la Figura se aplican a las entradas de un flip-flop J-K. Suponer que Q se encuentra inicialmente a nivel BAJO.



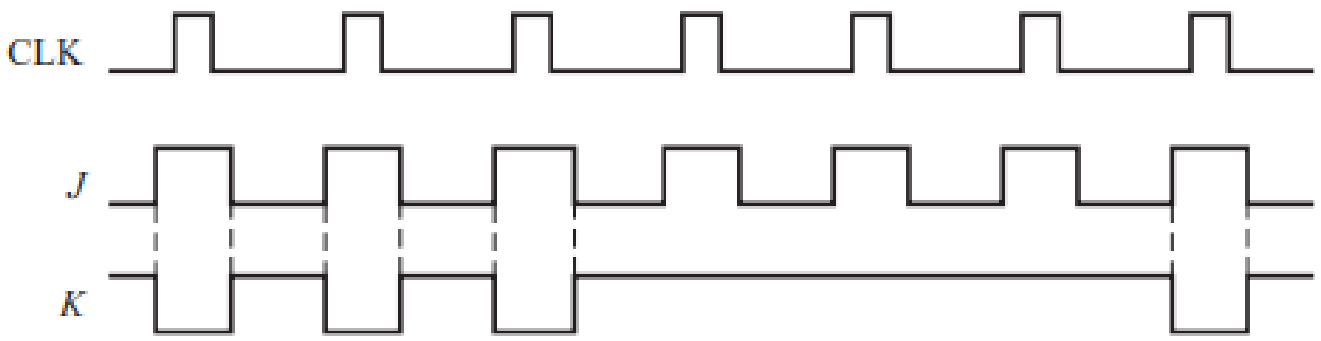


Q	J	K	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Tabla característica

Ejercicio

- desarrollar la forma de onda de salida Q en función del reloj. Suponer que Q se encuentra inicialmente a nivel BAJO



Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Tabla característica

16. Se aplican los siguientes datos serie a un flip-flop a través de puertas AND, como se indica en la Figura 7.83. Determinar los datos serie resultantes que aparecen en la salida Q. Hay un impulso de reloj por cada periodo de bit. Suponer que, inicialmente, Q es 0 y, $\overline{PRE}$ y $\overline{CLR}$ están a nivel ALTO. Los bits de más a la derecha son los primeros que se aplican.

J_1: 1 0 1 0 0 1 1

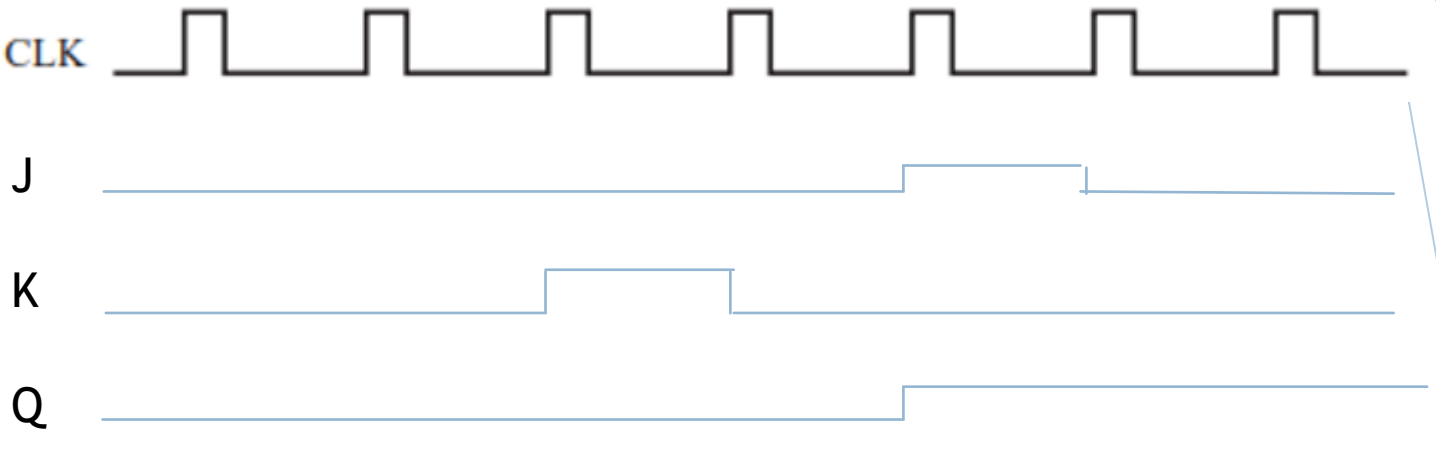
J_2: 0 1 1 1 0 1 0

J_3: 1 1 1 1 0 0 0

K_1: 0 0 0 1 1 1 0

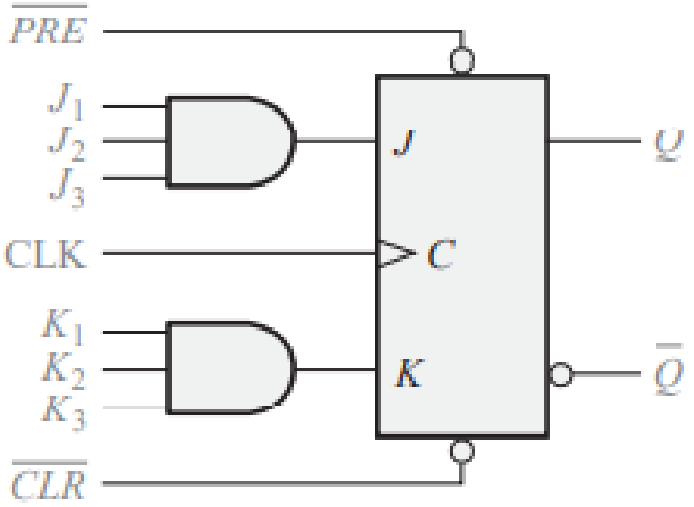
K_2: 1 1 0 1 1 0 0

K_3: 1 0 1 0 1 0 1

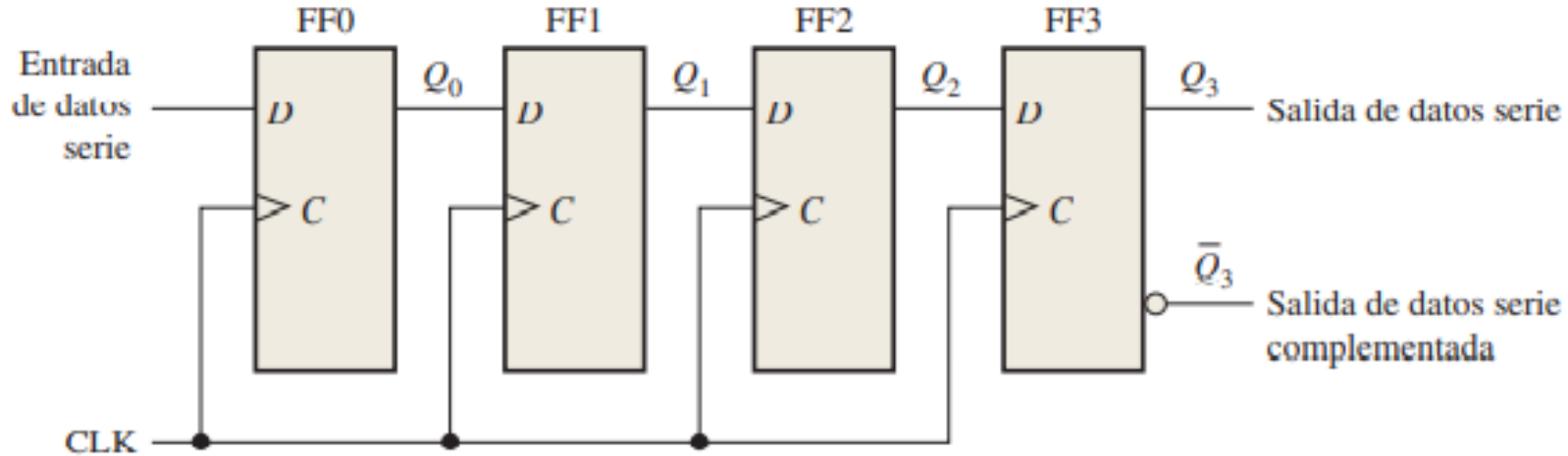


Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

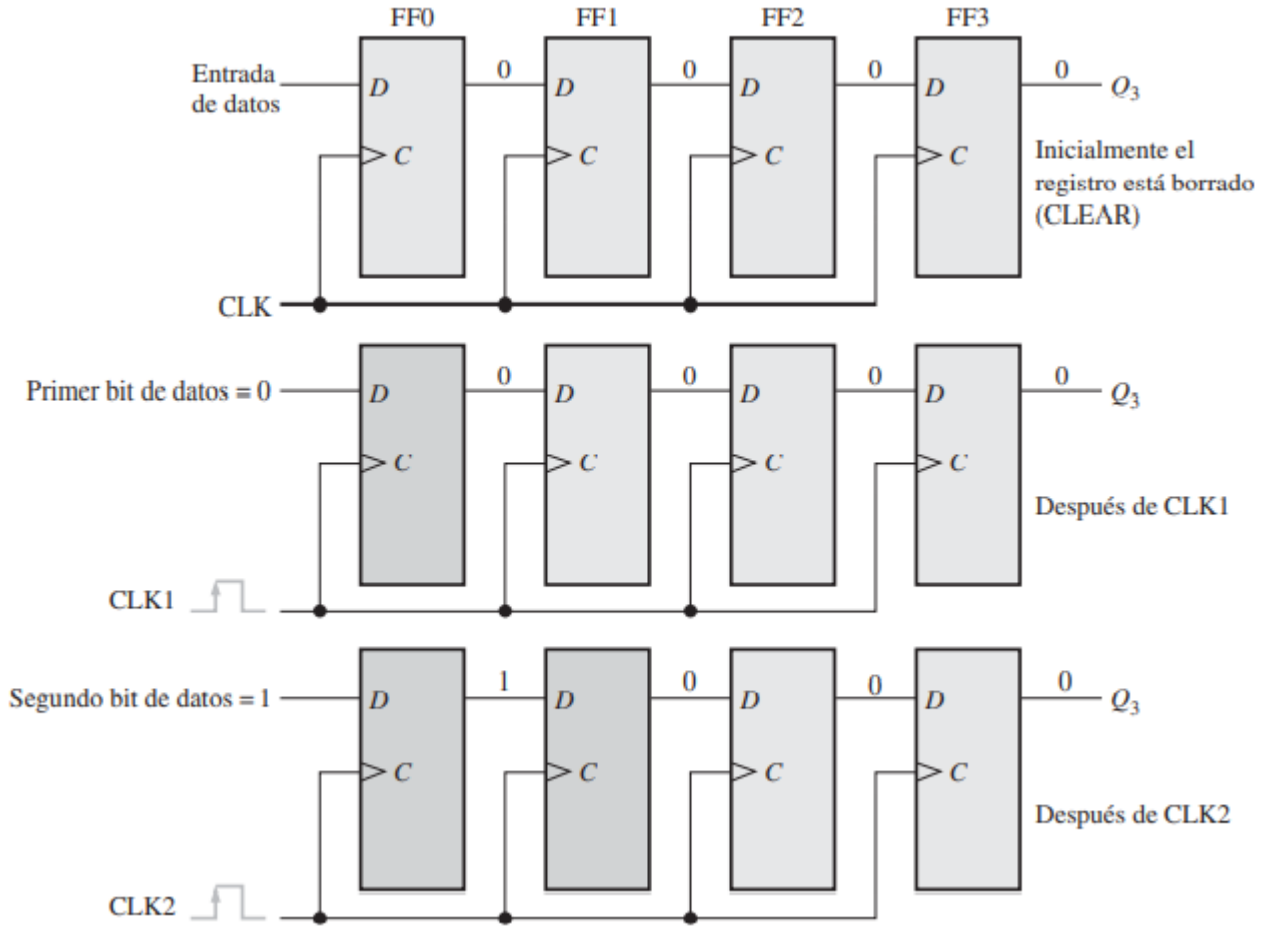
Tabla característica



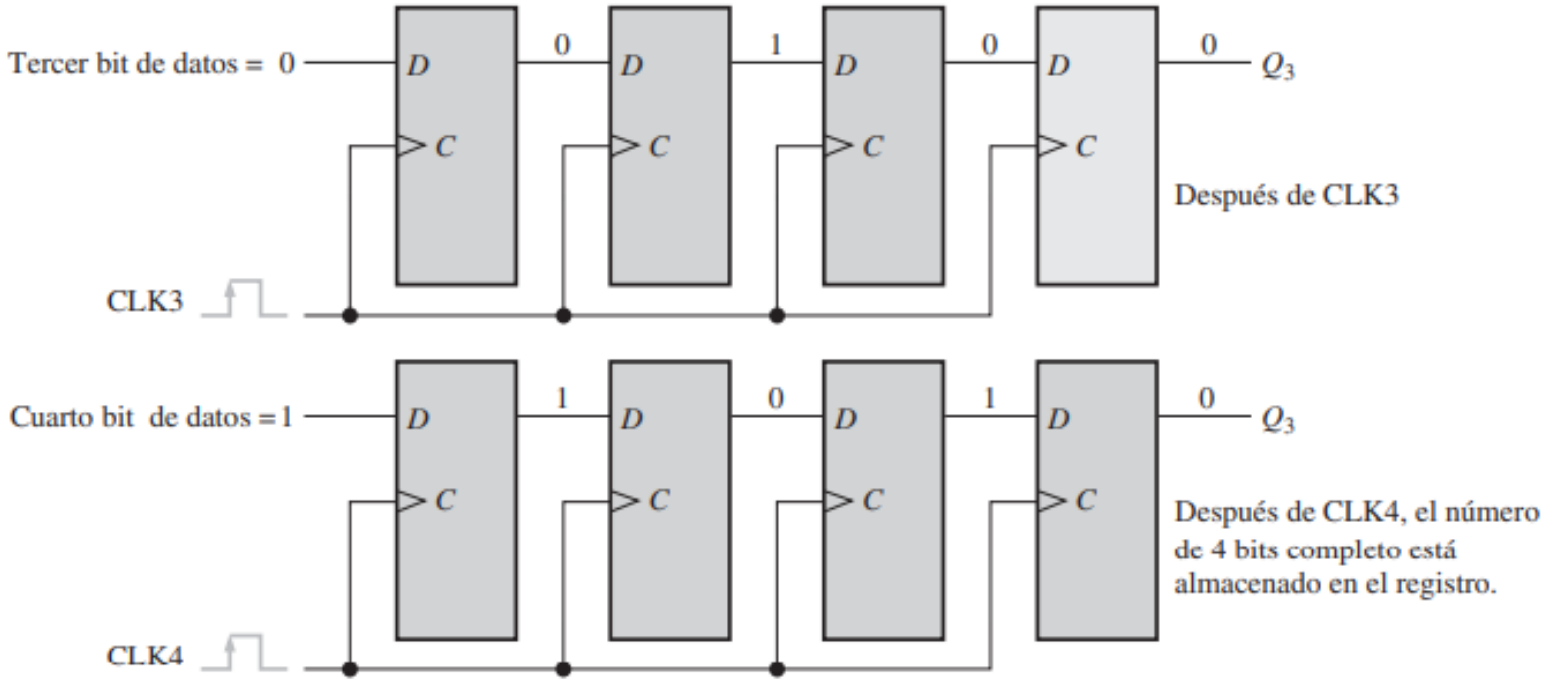
Registro de desplazamiento con entrada y salida serie.

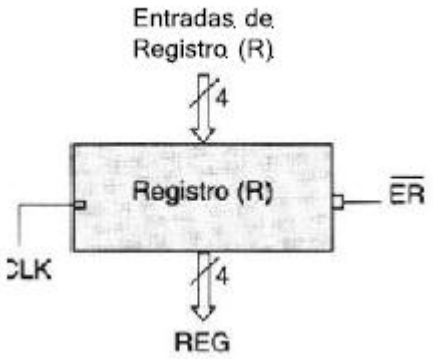


Introducción de cuatro bits en serie (1010) en el registro



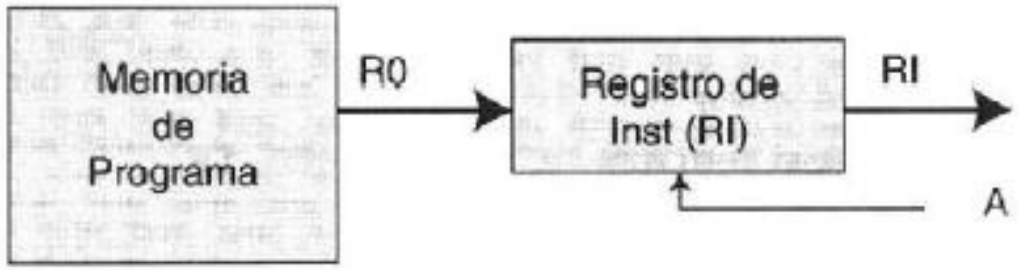
continuación






```
-- Diseño del registro R
library ieee;
use ieee.std_logic_1164.all;
entity registro is port(
        R:    in std_logic_vector(3 downto 0);
 ER, CLK:     in std_logic;
      REG:    inout std_logic_vector(3 downto 0));
end registro;

architecture arq_reg of registro is
begin
 process (CLK,RE,REG,R) begin
   if (CLK'event and CLK ='1')then
        if ER = '0' then
           REG <= R;
   else
           REG <= REG;
       end if;
   end if;
 end process;
end arq_reg;
```

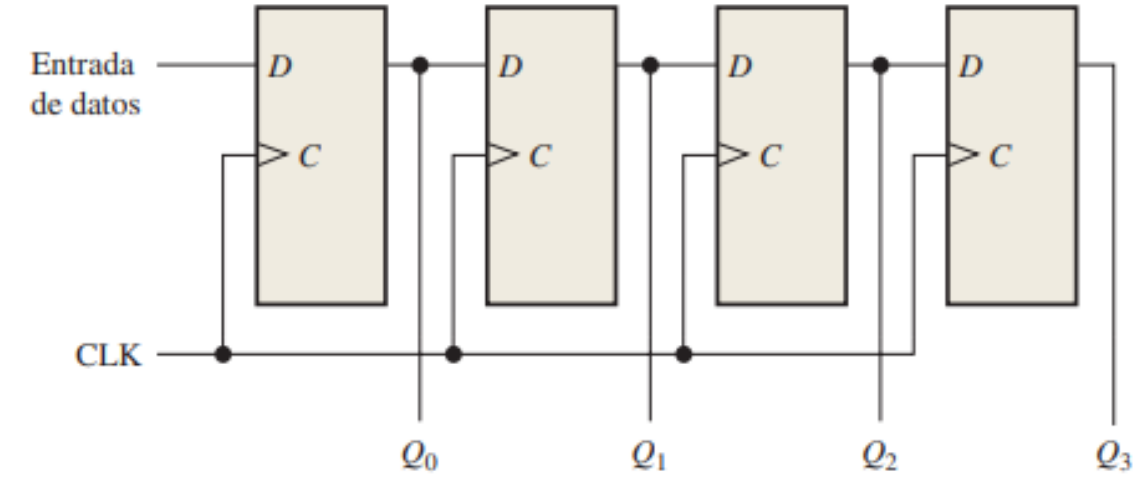
Programación del registro de instrucción (RI). Este módulo almacena temporalmente las instrucciones provenientes de la memoria de programa. Su función es guardar el código binario de la instrucción mediante la señal de habilitación (A) que envía el generador de ciclo de máquina.



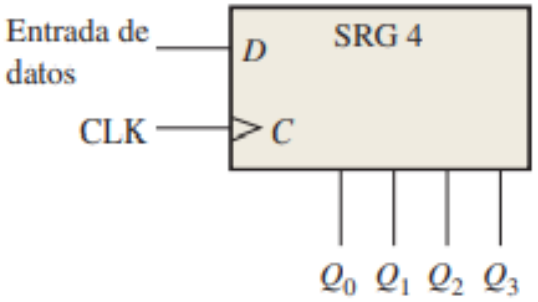
Registro de instrucción.

```
Entity reg_instr is port(
A: in std_logic;
RO: in std_logic_vector (3downto 0);
RI: out std_logic_vector (3downto 0);
End reg_instr;

Architecture Behavioral of reg_instr is
Begin
Process (A,RO) begin
    if (A'event and A='1') then
        RI<=RO;
    end if;
End process;
End Behavioral;
```

REGISTROS DE DESPLAZAMIENTO CON ENTRADA SERIE Y SALIDA PARALELO



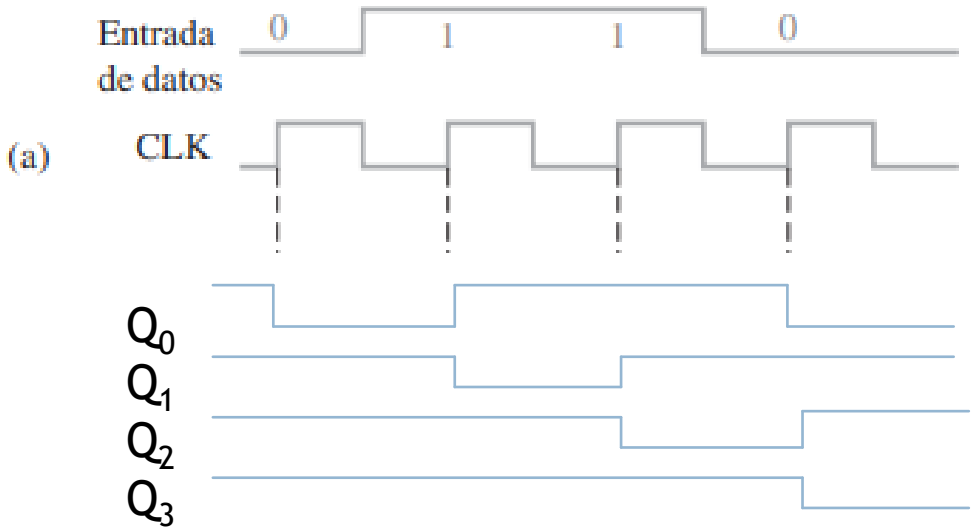
(a)



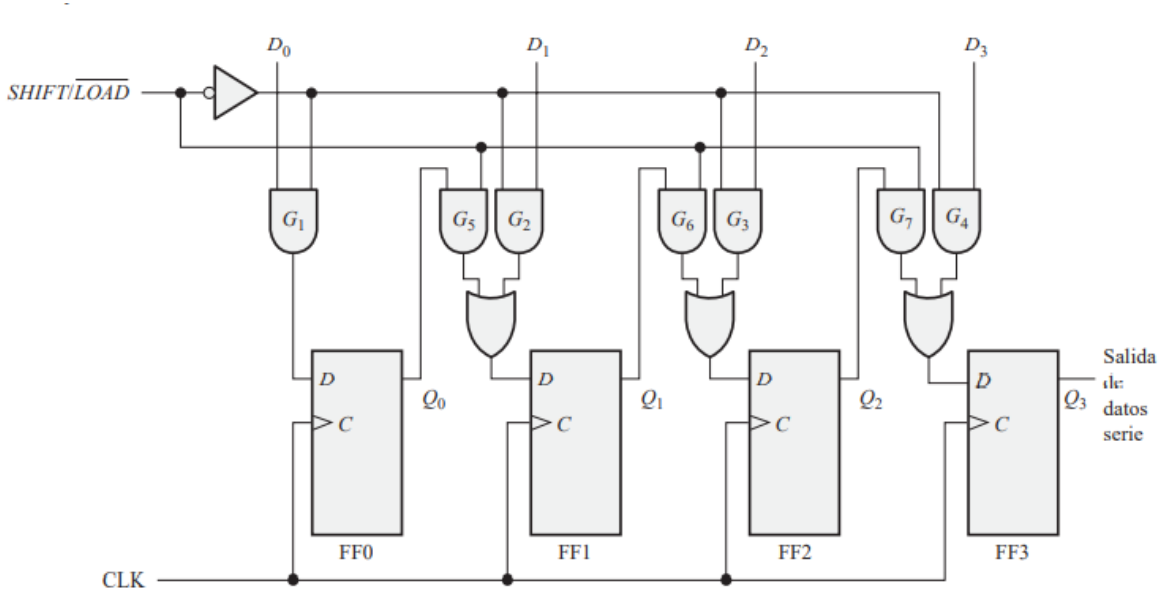
(b)

EJERCICIO

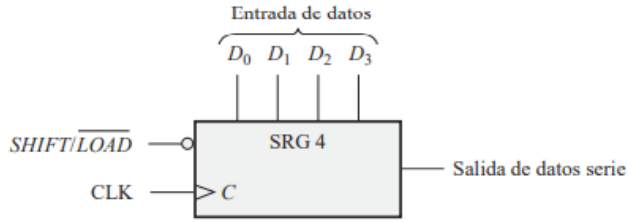
- Mostrar los estados del registro de 4 bits (SRG 4) para las formas de onda de entrada y de reloj de la Figura (a). Inicialmente, el contenido del registro es todo 1s.



Entrada paralelo salida serie



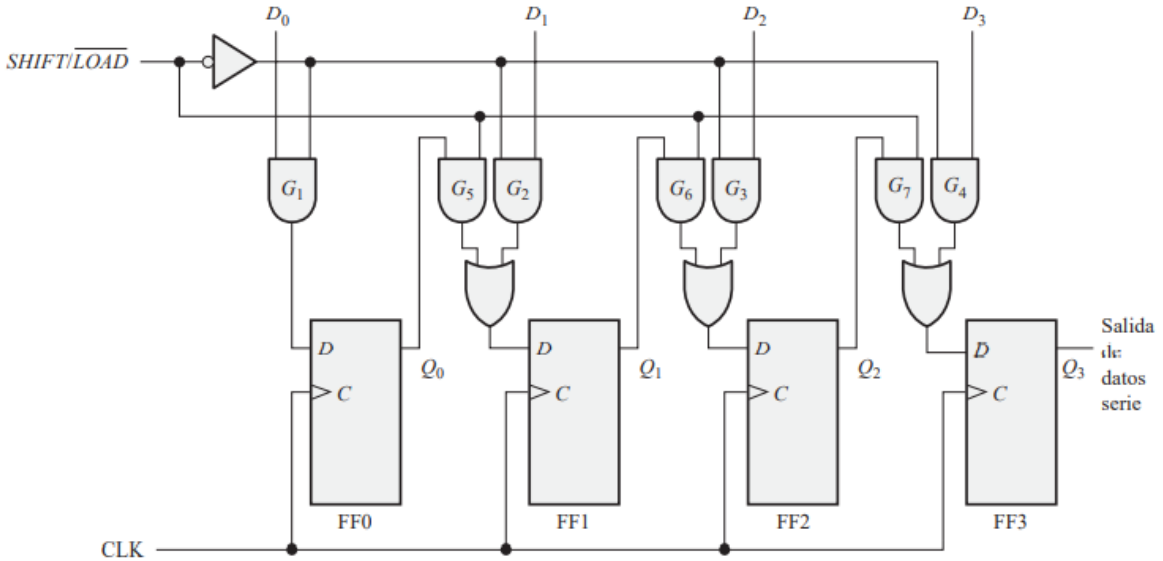
(a) Diagrama lógico



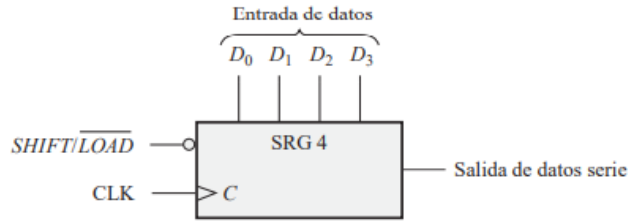
(b) Símbolo lógico

Ejercicio

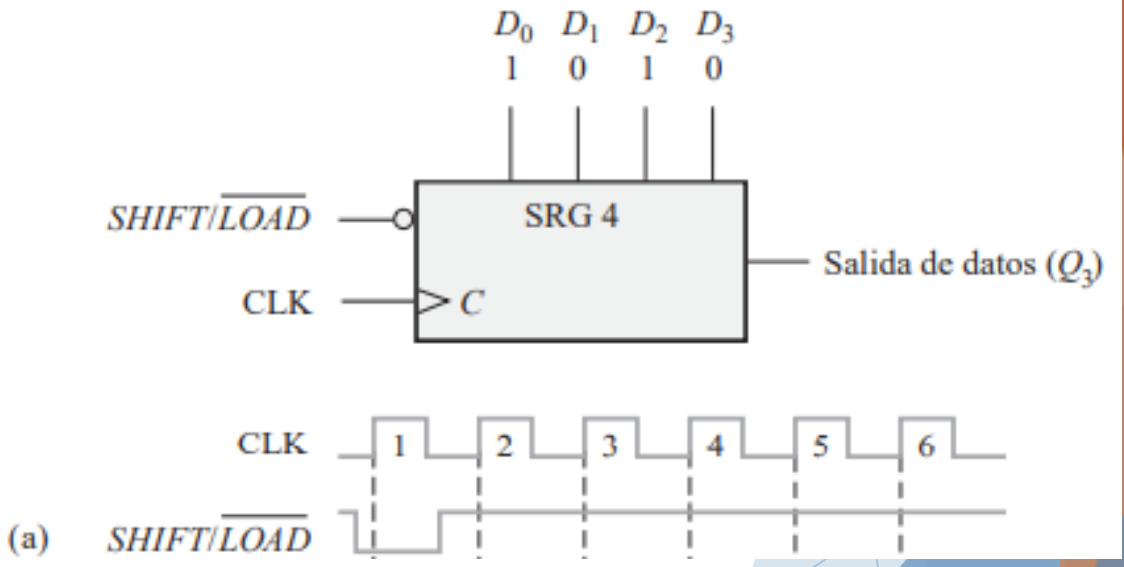
- Determinar la forma de onda de la salida de datos de un registro de 4 bits para las formas de onda de entrada paralelo de datos, de reloj y de la Figura (a). Utilizar el diagrama lógico de la Figura



(a) Diagrama lógico



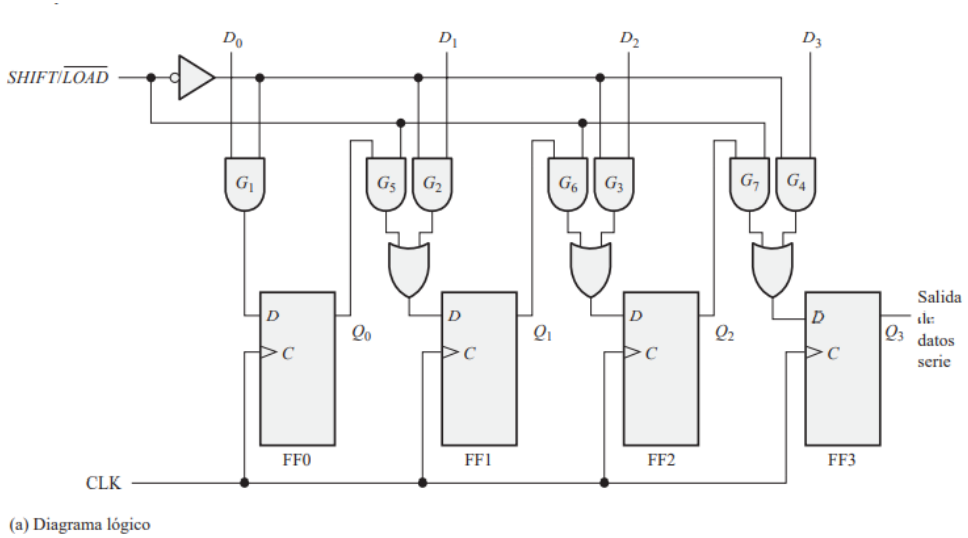
(b) Símbolo lógico



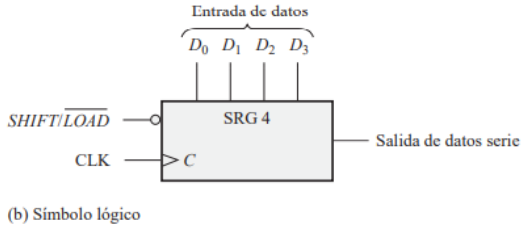
(a)

Ejercicio

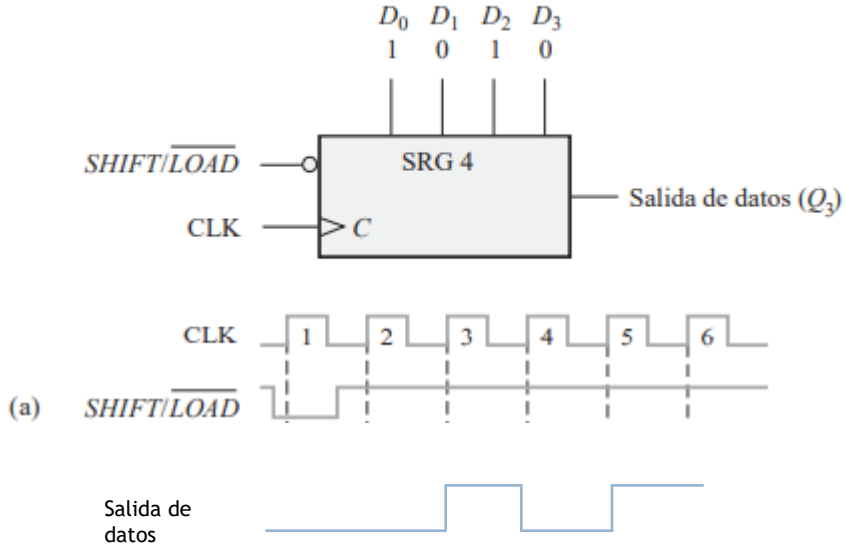
- Determinar la forma de onda de la salida de datos de un registro de 4 bits para las formas de onda de entrada paralelo de datos, de reloj y de la Figura (a). Utilizar el diagrama lógico de la Figura



(a) Diagrama lógico

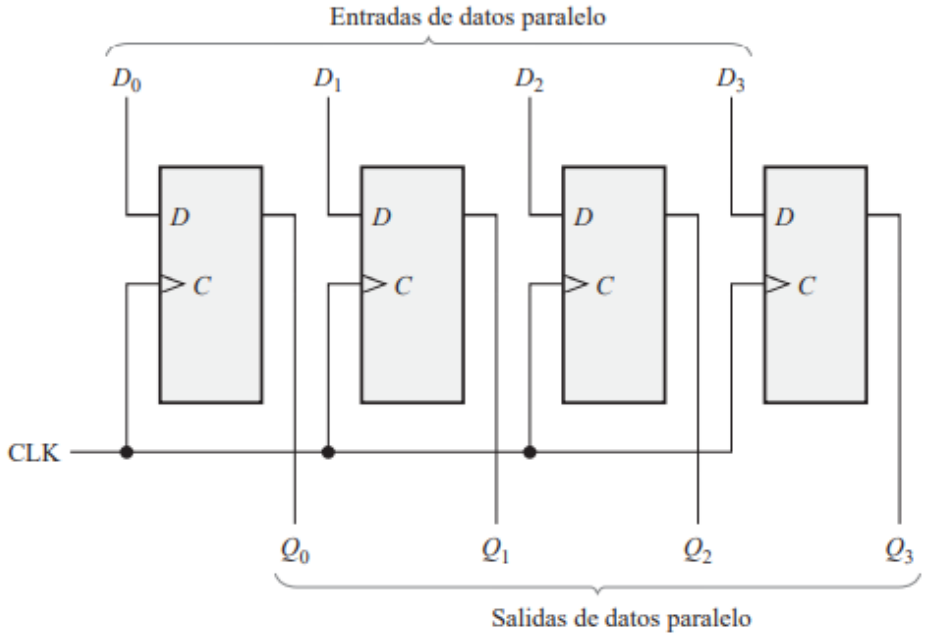


(b) Símbolo lógico

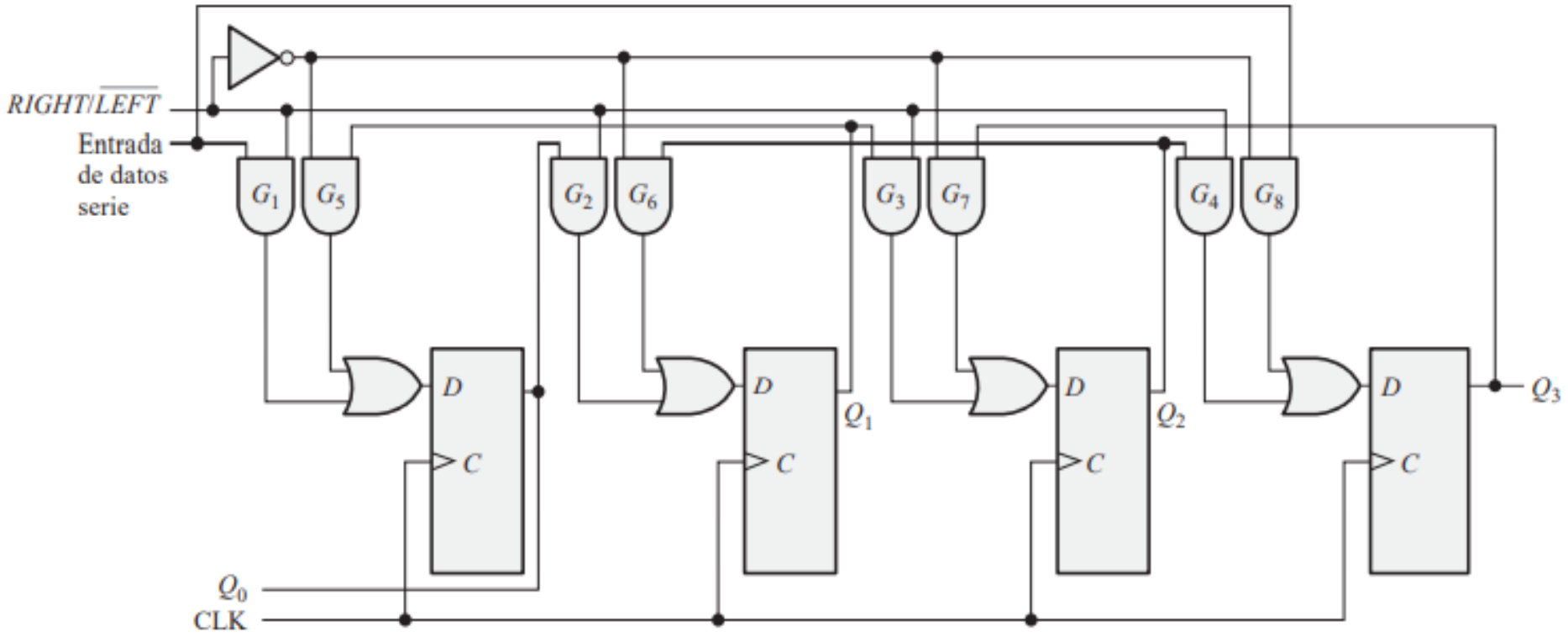


(a)

REGISTROS DE DESPLAZAMIENTO CON ENTRADA Y SALIDA PARALELO



REGISTROS DE DESPLAZAMIENTO BIDIRECCIONALES



Ejercicio

- Determinar el estado del registro de desplazamiento de la Figura 1 después de cada impulso de reloj para la forma de onda de la entrada de control indicada en la Figura 1(a). Suponer que Q0 = 1, Q1 = 1, Q2 = 0 y Q3 = 1, y que la línea de entrada de datos serie está a nivel BAJO.

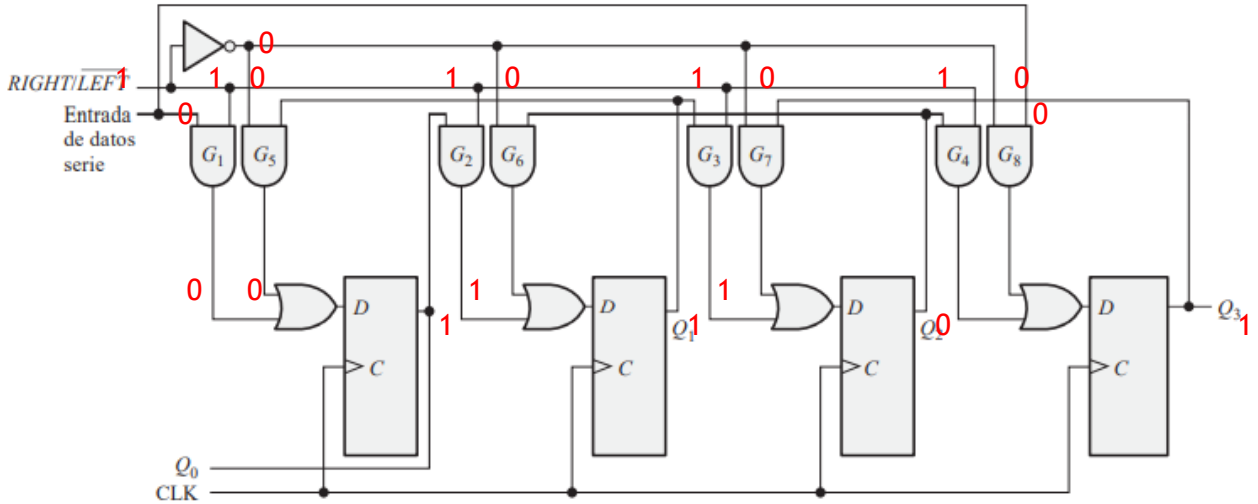


Figura 1

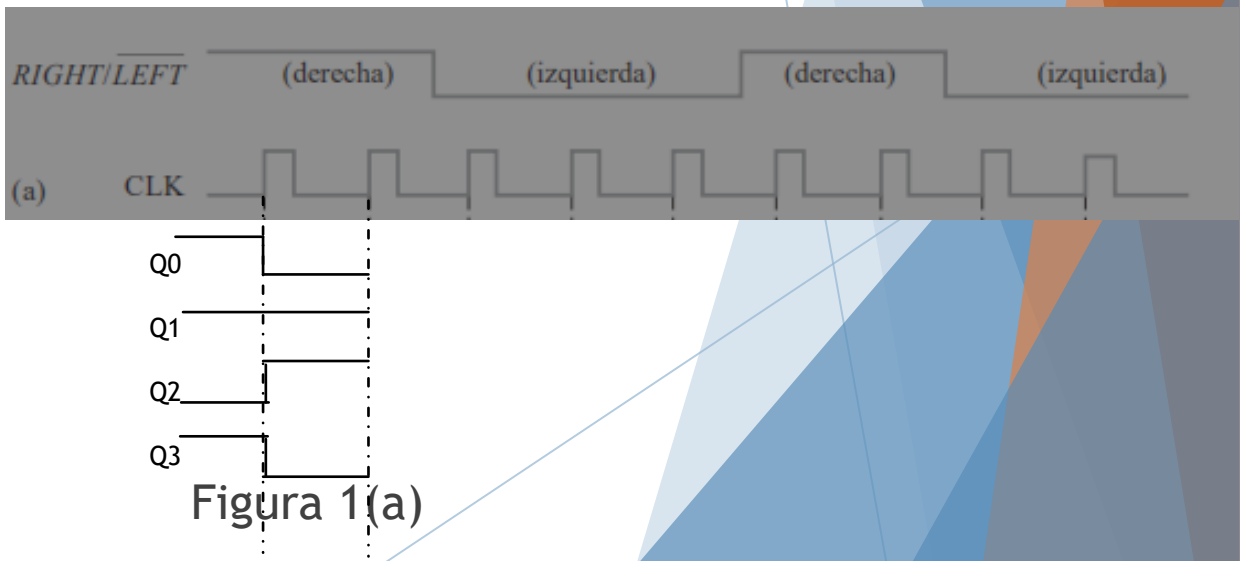
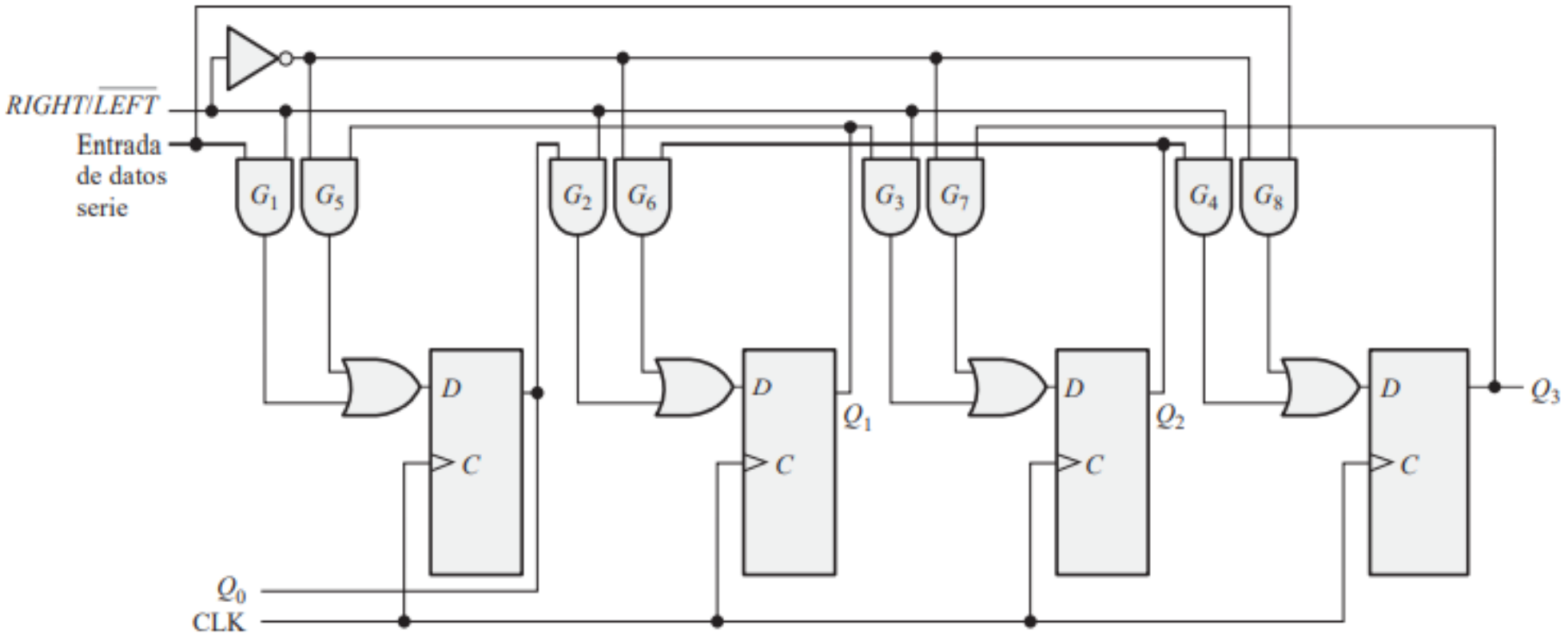
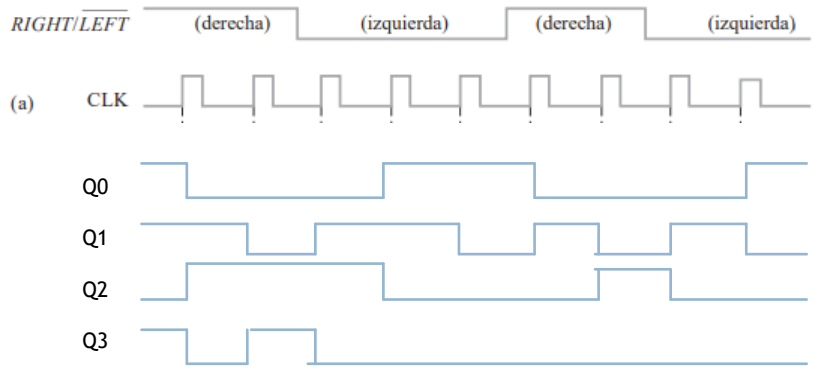


Figura 1(a)

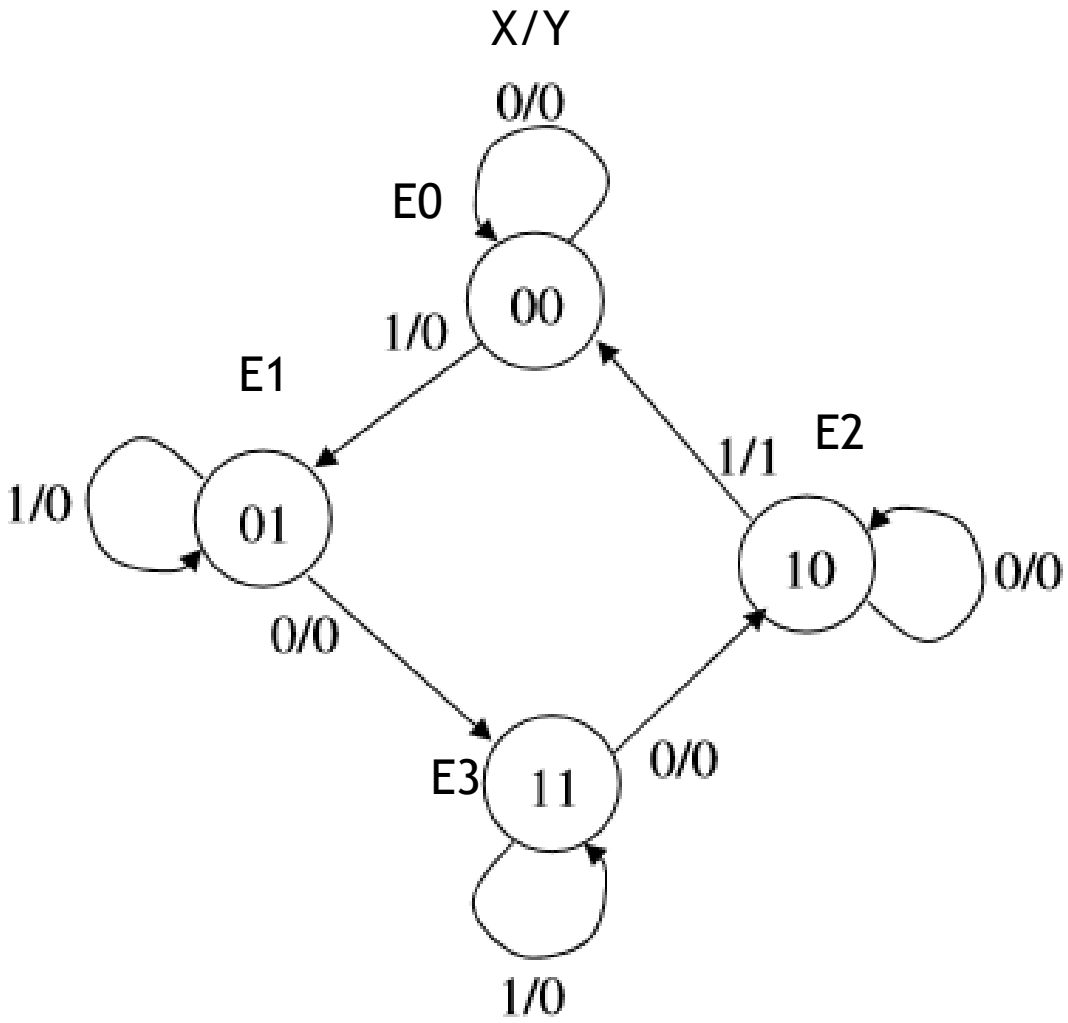

RIGHT/LEFT
Entrada de datos serie
G1
G5
G2
G6
G3
G7
G4
G8
D
C
Q1
Q2
Q3
Q0
CLK


RIGHT/LEFT
(derecha)
(izquierda)
(derecha)
(izquierda)
(a)
CLK
Q0
Q1
Q2
Q3

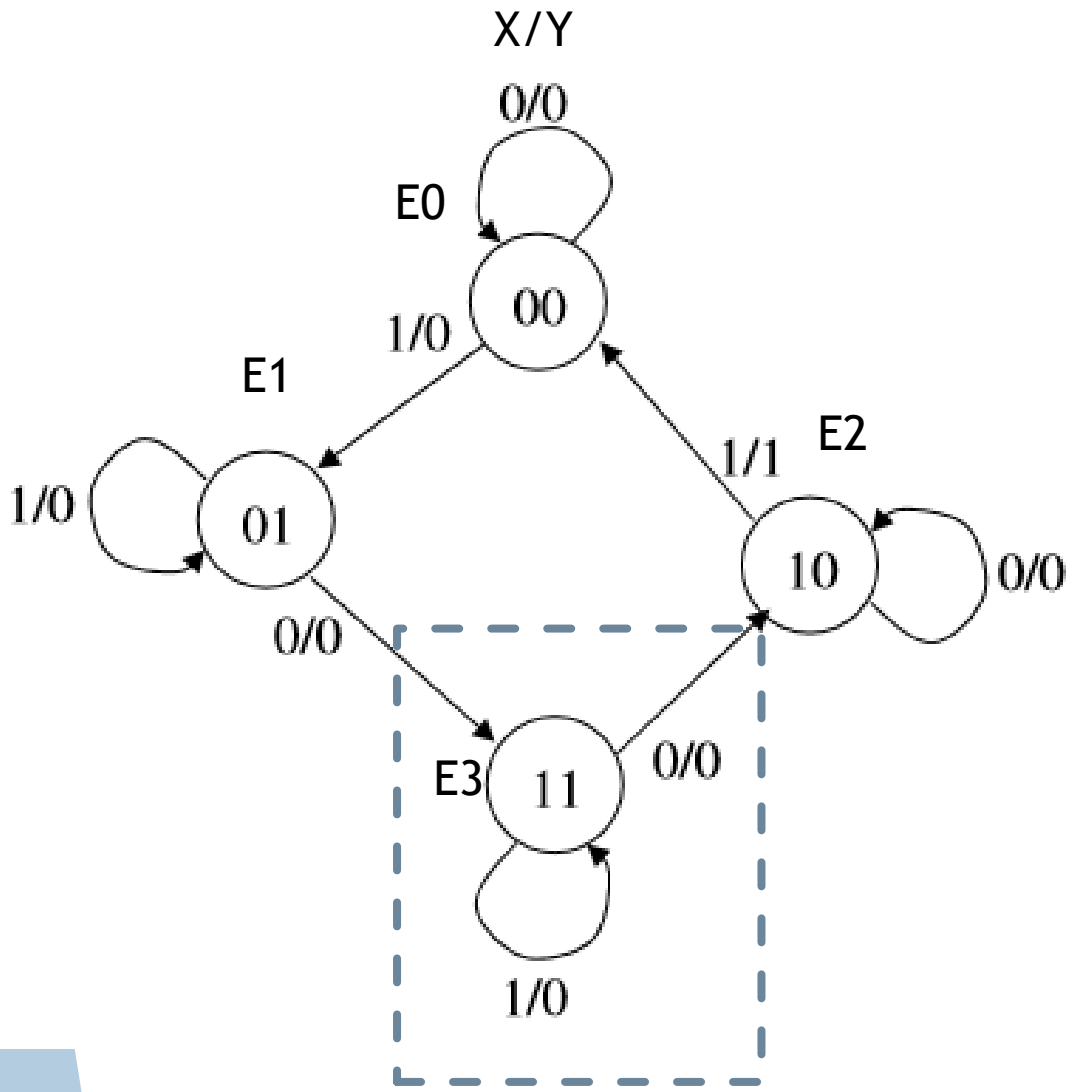
ANÁLISIS DE CIRCUITOS SECUENCIALES SÍNCRONOS

- El análisis consiste en obtener una tabla de estados (o tabla de transición) y/o un diagrama de flujo, de las secuencias de tiempo de las entradas, salidas y estados internos del sistema secuencial.
- También es posible escribir expresiones booleanas que describan su comportamiento.
- La tabla consta de 4 secciones principales: entradas, estado presente, estado siguiente y salida.
  - En la sección estado presente se indica los estados de los FF antes de la ocurrencia del pulso de reloj bajo las condiciones de entrada indicadas.
  - En la sección estado siguiente se muestra el estado de los FF después del pulso.
  - Y la sección de salida muestra los valores de las variables de salida durante el estado presente.

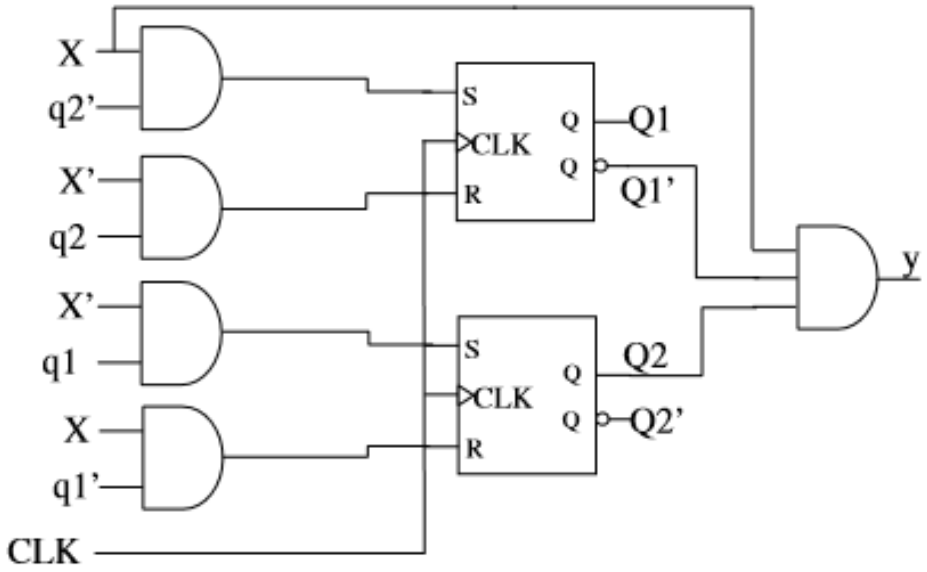
Diagrama de estados



Creación de tabla de transición



Entrada	Estado presente		Estado siguiente		Salida
X	Q2	Q1	Q2	Q1	Y
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	1	1	0



Ecuaciones:

$Q1 = S + R'q1 = Xq2' + (X'q2)'q1 = Xq2' + (X + q2')q1 = Xq2' + Xq1 + q2'q1$

$Q2 = S + R'q2 = X'q1 + (Xq1')'q2 = X'q1 + (X' + q1)q2 = X'q1 + X'q2 + q1q2$

$Y = XQ1'Q2$

TABLA de ESTADOS o de TRANSICIÓN

Entrada	Estado presente		Estado futuro		Salida
X	q2	q1	Q2	Q1	Y
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	1	1	0

Se pueden seguir una serie sencilla de pasos:

1. Se parte de las especificaciones de funcionamiento, que puede incluir un diagrama de flujo.

2. Se obtiene la tabla de estados.

3. Se procede a la reducción del número de estados, es opcional.

4. Se asignan valores binarios a cada estado de la tabla y obtenemos una tabla de transición.

5. Se determina el número de FF necesarios y se asigna un símbolo a cada uno.

6. Se escoge el tipo de FF que va a utilizarse

7. Mediante la tabla de transición se derivan las entradas de los biestables (tabla de excitación) y las salidas. Éstas suelen ser funciones combinacionales.

8. Mediante algún método (Karnaugh, McCluskey ...) se simplifican dichas funciones o se construyen usando dispositivos MSI o PLDs.

9. Dibujar el diagrama lógico

Cartas asm – maquinas de estados

MÁQUINAS DE ESTADOS

- El modelo de máquina de estados contiene los elementos necesarios para describir la conducta de un sistema en términos de entradas, salidas y del tiempo.
- El siguiente diagrama presenta el modelo general de una Máquina de Estados.

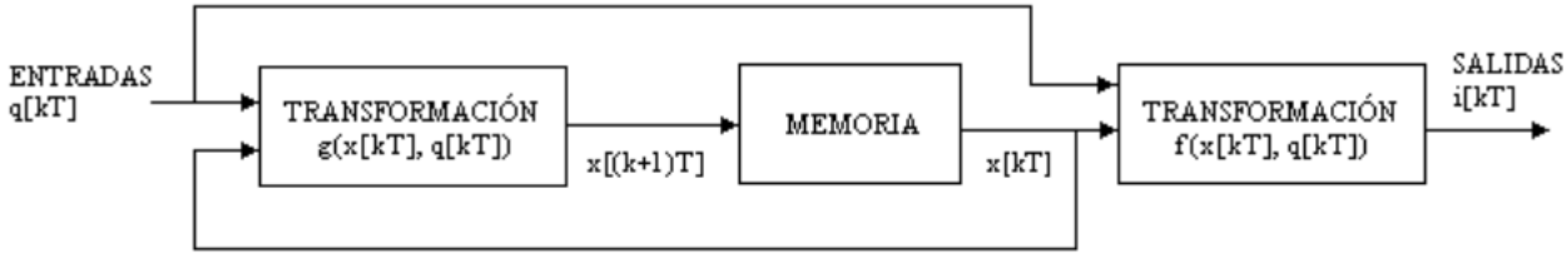


Figura. 1. Modelo general de una máquina de estados.

- x[kT], representa el estado en el tiempo kT.
- x[(k+1)T] = g(x[kT] , q[kT]), representa el siguiente estado.
- q[kT], representa las entradas en el tiempo kT.
- i[kT] = f(x[kT] , q[kT]), representa las salidas en el tiempo kT.
- En donde T es el período de duración de cada estado y k es un contador entero.

EL ALGORITMO DE LA MÁQUINA DE ESTADOS

- Juega un papel muy importante en el diseño de sistemas digitales.
- Para circuitos síncronos la técnica de la carta ASM (Algorithm State Machine / Algoritmo de la Máquina de Estados) es la mejor notación.

Diagramas de estados

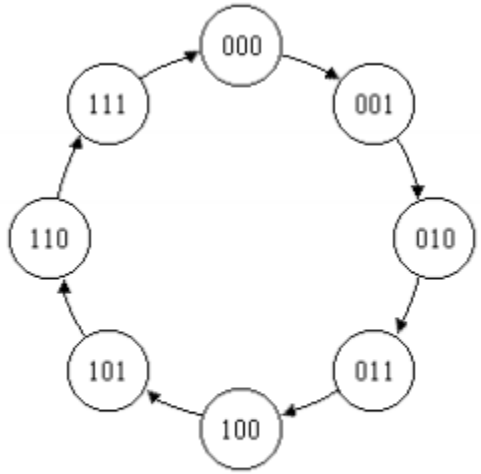


Figura. 2. Diagrama de estados para un contador binario de 3 bits.

- Existen otras técnicas como la de los diagramas de estados que son diagramas muy parecidos a las cartas ASM.
- Muestran gráficamente la secuencia de estados en un sistema y las condiciones de cada estado y de las transiciones entre cada uno de ellos. Como ejemplo, en la figura 2 se describe el comportamiento de un contador binario de 3 bits mediante un diagrama de estados.
- Este circuito en particular no tiene ninguna entrada aparte de la de reloj, y ninguna otra salida más que las que se toman en cada flip-flop del contador.

ESTADOS Y RELOJ

- El algoritmo de la máquina de estados se mueve a través de una secuencia de estados con base en la posición del estado presente y las variables de entrada. Los tiempos del estado están determinados por un reloj maestro.

NOTACIÓN DE LA CARTA ASM

- REPRESENTACIÓN DE ESTADOS
- El estado de una máquina de estados es la memoria de la historia pasada, suficiente para determinar las condiciones futuras.
- En la siguiente figura 3 se muestra la representación del estado.

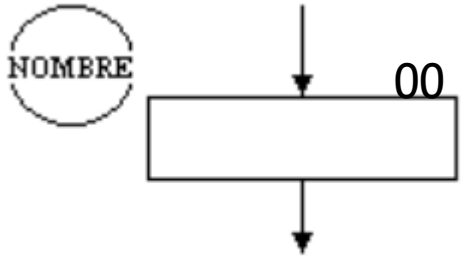


Figura 3. Representación del estado.

- Un estado se representa con un rectángulo y con su nombre simbólico en el extremo superior, encerrado en un círculo.

REPRESENTACIÓN DE DECISIONES

- Las decisiones permiten seleccionar el camino que el algoritmo de la máquina de estados debe tomar de acuerdo a la variable o variables de entrada evaluadas.
- Las decisiones se representan mediante un rombo con el nombre de la variable a probar o una función que evalúe varias variables.

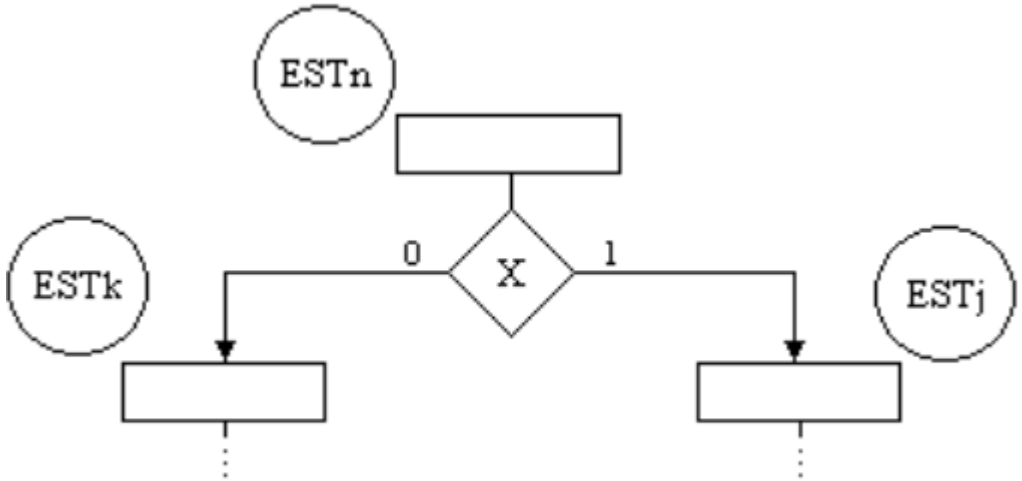


Figura 4. Representación de las decisiones.

REPRESENTACIÓN DE SALIDAS

- Salidas no condicionales.
- Sirven para indicar la activación de una variable de salida.
- Para representarlas, se escriben dentro del rectángulo de estado, los nombres de las variables de salida que se activan en ese estado.
- Las salidas no condicionales no dependen de las condiciones de entrada, sólo dependen del estado actual.
- La figura 5 muestra la activación de las salidas VAR1 y VAR2 en el estado EST1.

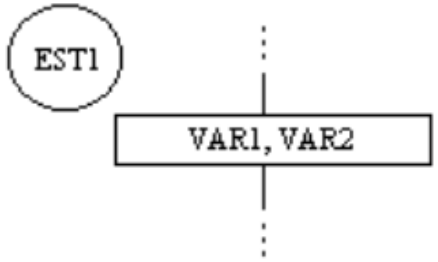


Figura 5. Representación de las salidas no condicionales.

Salidas Condicionales.

- Estas salidas se presentan solamente cuando ciertas condiciones de entrada existen. Se representan con un óvalo y los nombres de las salidas dentro de él.

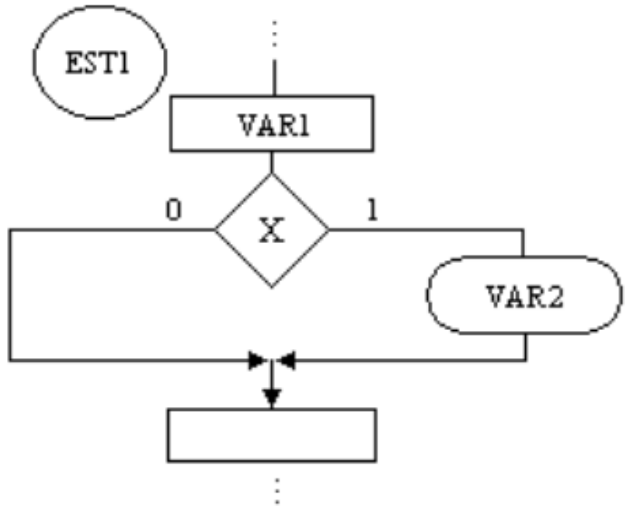


Figura 6. Representación de las salidas condicionales.

- Para este ejemplo, si en el estado EST1 la variable de entrada X vale uno, entonces las salidas VAR1 y VAR2 serán activadas. Si X es igual a cero solamente se activará VAR1.

EJEMPLOS DE CARTAS ASM

- 1) Diseñe un dispositivo que genere cierta secuencia binaria sólo cuando la variable INICIO sea igual a uno. Además esta secuencia dependerá del valor de la entrada X. Si X=0 la secuencia binaria que se genera es la siguiente: 11, 10, 01, por el contrario, si X=1 la secuencia es: 01, 10, 11. Considere que cada pareja binaria se genera con un ciclo de reloj de diferencia.

solución

- 1) Realizar un diagrama de bloques que clarifique cuáles son:
  - Las señales de entrada
  - Las señales de salida
  - Quién es el controlador
  - Qué se está controlando.
- Para este ejemplo, las señales de entrada son INICIO y X, y las señales de salida son las que representan la secuencia que se quiere generar, nombrémoslas VAR1 y VAR0.

- El diagrama de bloques queda de la siguiente manera.

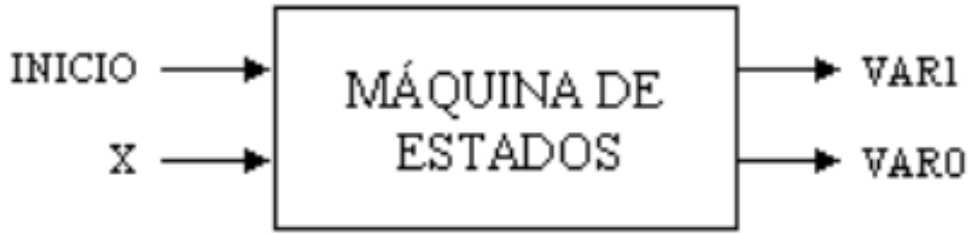


Figura 7. Diagrama de bloques para el ejemplo 1.

- Y la carta ASM para esta máquina de estados se muestra en la figura 8.

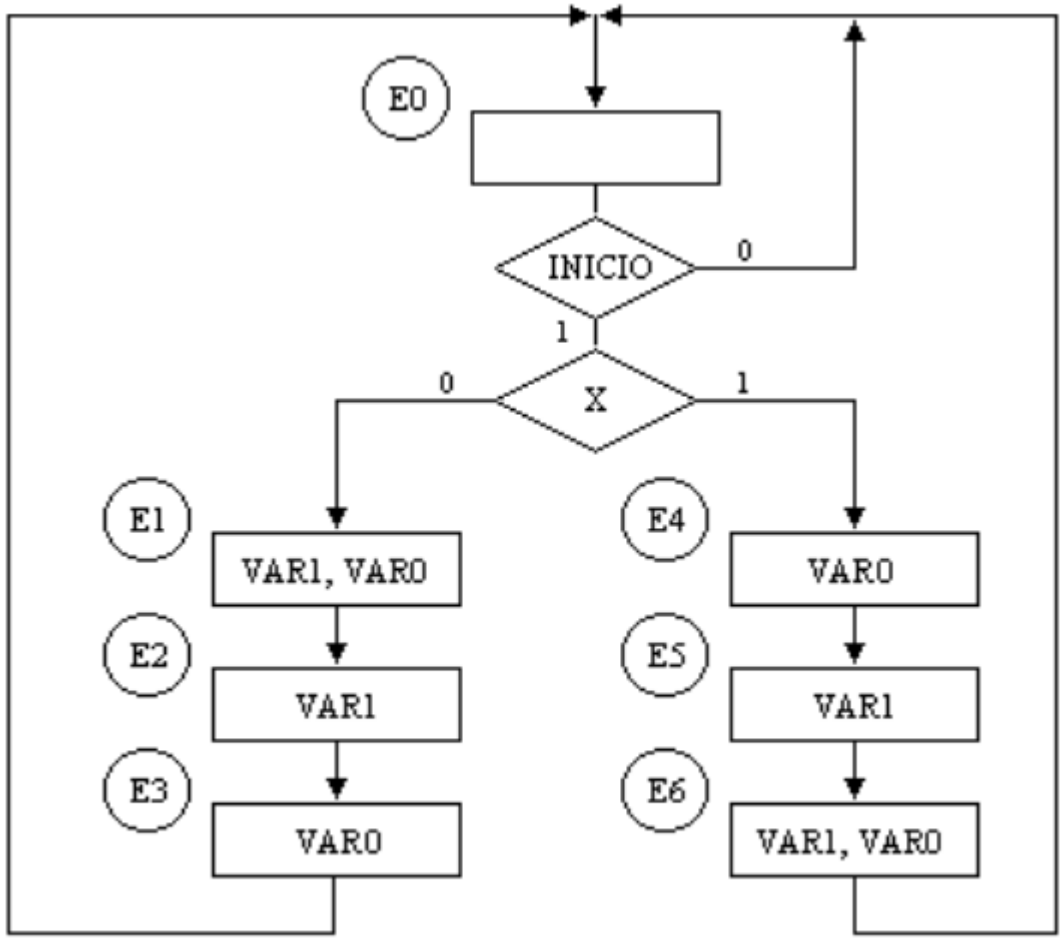


Figura 8. Carta ASM para el ejemplo 1.

- Para activar una señal de salida incondicional, en un estado en particular, es necesario colocar su nombre dentro del rectángulo del estado.

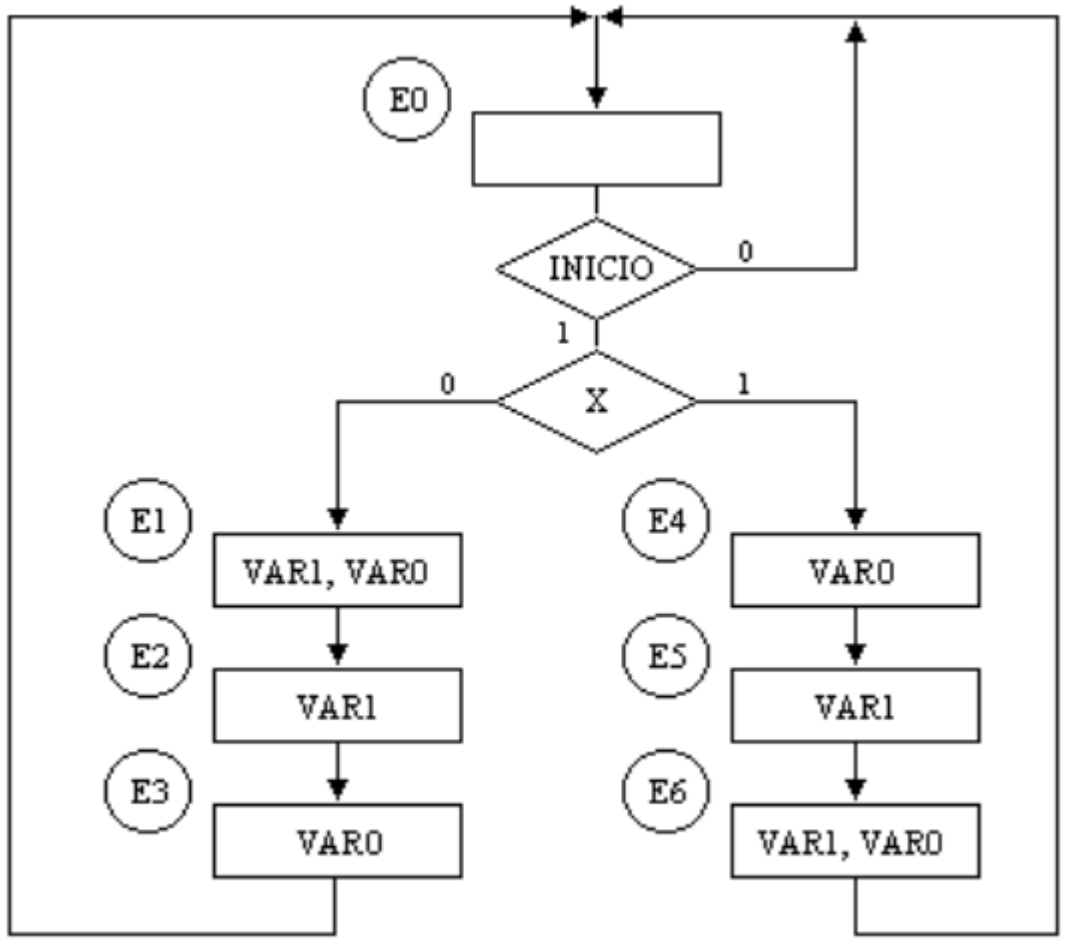
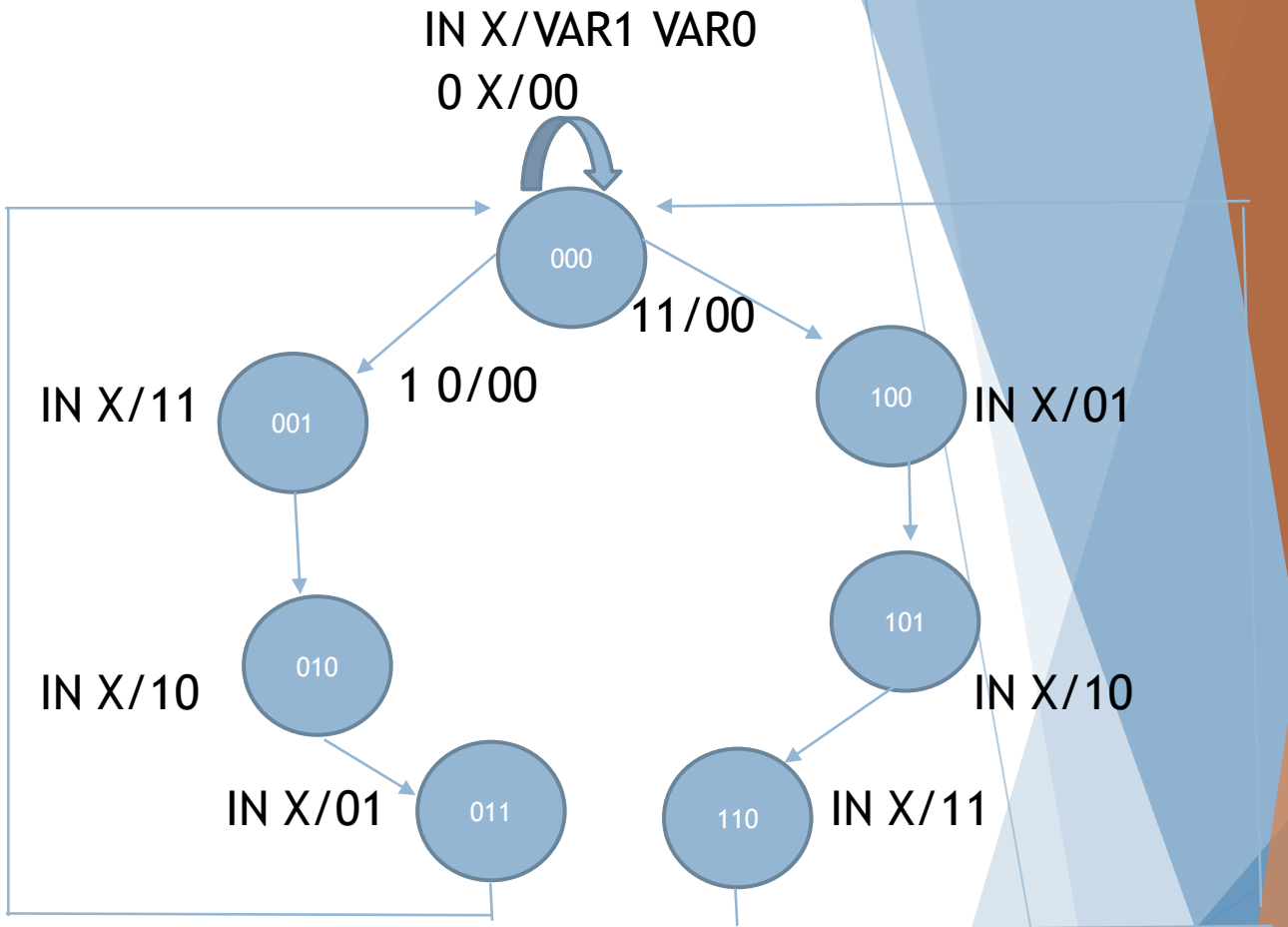


Figura 8. Carta ASM para el ejemplo 1.



Entradas		Estado presente			Estado siguiente			Salidas	
Inicio	X	q3	q2	q1	Q3	Q2	Q1	Var1	Var0
0	0/1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0
1	0	0	0	1	0	1	0	1	1
1	0	0	1	0	0	1	1	1	0
1	0	0	1	1	0	0	0	0	1
1	0	1	0	0	1	0	1	0	1
1	0	1	0	1	1	1	0	1	0
1	0	1	1	0	0	0	0	1	1
1	0	1	1	1	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0
1	1	0	0	1	0	1	0	1	1
1	1	0	1	0	0	1	1	1	0
1	1	0	1	1	0	0	0	0	1
1	1	1	0	0	1	0	1	0	1
1	1	1	0	1	1	1	0	1	0
1	1	1	1	0	0	0	0	1	1
1	1	1	1	1	0	0	0	0	0

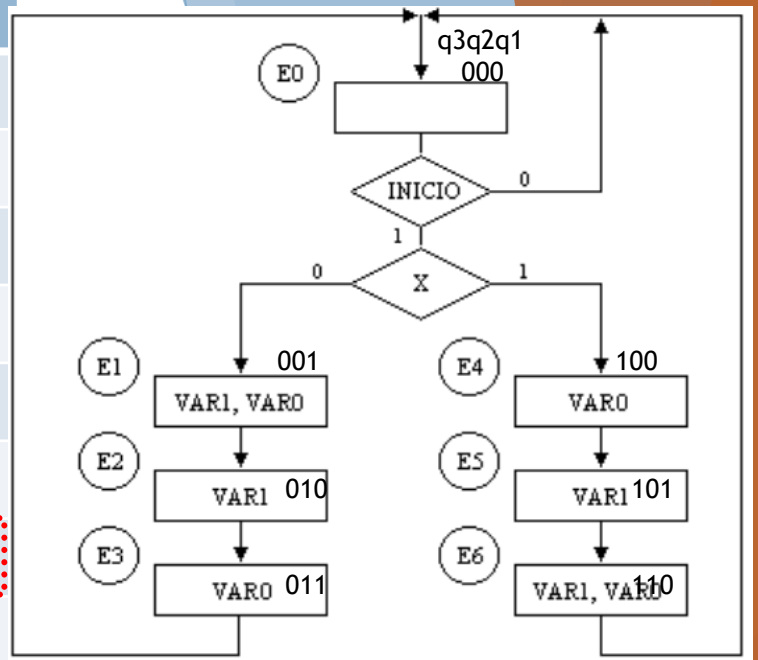


Figura 8. Carta ASM para el ejemplo 1.

X=0: 11, 10, 01
X=1 la secuencia es: 01, 10, 11.

Q3

	q2'q1'	q2'q1	q2q1	q2q1'
X'q3'	0	0	0	0
X'q3	1	1	0	0
Xq3	1	1	0	0
Xq3'	1	0	0	0

Q3=I(q3q2'+Xq2'q1')

Q2

	q2'q1'	q2'q1	q2q1	q2q1'
X'q3'	0	1	0	1
X'q3	0	1	0	0
Xq3	0	1	0	0
Xq3'	0	1	0	1

Q1

	q2'q1'	q2'q1	q2q1	q2q1'
X'q3'	1	0	0	1
X'q3	1	0	0	0
Xq3	1	0	0	0
Xq3'	0	0	0	1

Q2=I(q2'q1+q3'q2q1')

Q1=I(X'q2'q1'+q3q2'q1'+q3'q2q1')

Var1

	q2'q1'	q2'q1	q2q1	q2q1'
X'q3'	0	1	0	1
X'q3	0	1	0	1
Xq3	0	1	0	1
Xq3'	0	1	0	1

Var1=I(q2'q1+q2q1')

Var0

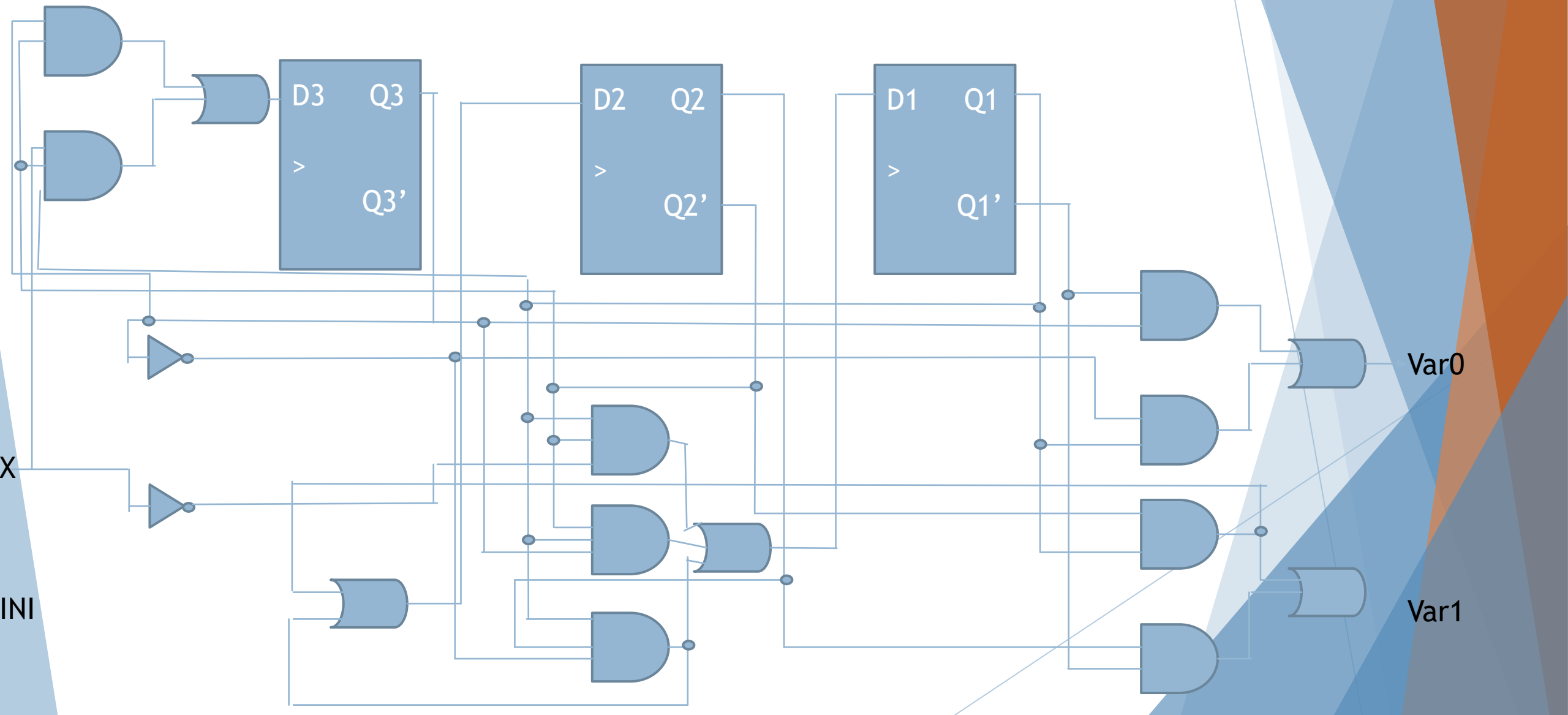
	q2'q1'	q2'q1	q2q1	q2q1'
X'q3'	0	1	1	0
X'q3	1	0	0	1
Xq3	1	0	0	1
Xq3'	0	1	1	0

Var0=I(q3q1'+q3'q1)

Estado siguiente			Salidas	
Q3	Q2	Q1	Var1	Var0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
0	0	0	0	1
1	0	1	0	1
1	1	0	1	0
0	0	0	1	1
0	0	0	0	0
1	0	0	0	0
0	1	0	1	1
0	1	1	1	0
0	0	0	0	1
1	0	1	0	1
1	1	0	1	0
0	0	0	1	1
0	0	0	0	0

D3=Q3=I(q3q2'+Xq2'q1') D2=Q2=I(q2'q1+q3'q2q1') D1=Q1=I(X'q2'q1'+q3q2'q1'+q3'q2q1')

Var1=I(q2'q1+q2q1') Var0=I(q3q1'+q3'q1)



Ejercicio

2) Convertir el siguiente código en lenguaje ‘C’ a una carta ASM.

```
for (x = a; x ≤ b; x = x + c)
{ var1 = 1; var2 = 0; }
 var1 = 0;
```

Solución

- El diagrama de bloques de este sistema es el siguiente.

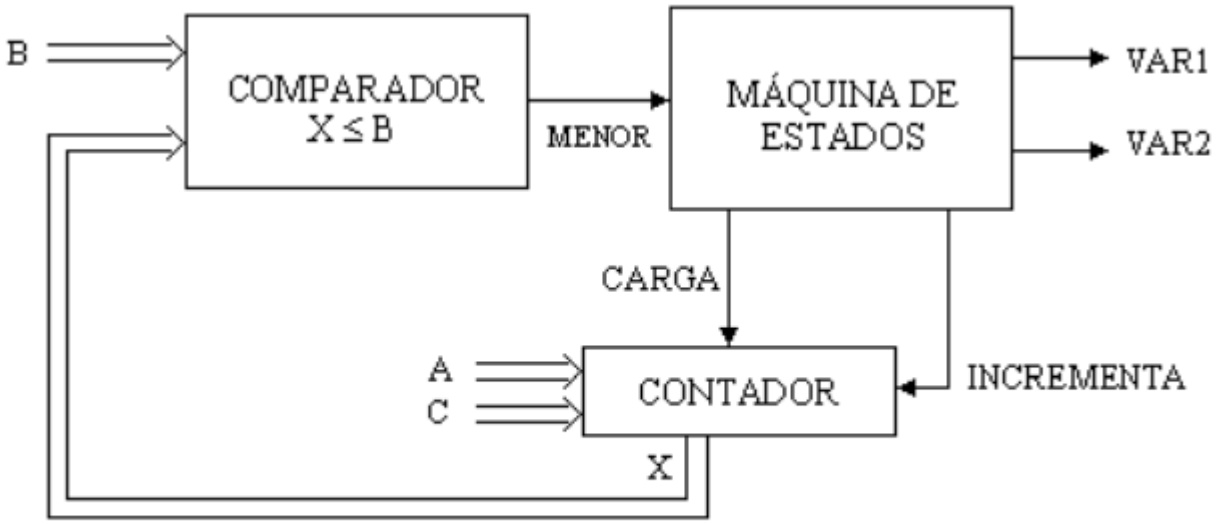


Figura 9. Diagrama de bloques para el ejemplo 2.

- Se utiliza un contador para cargar el valor inicial de X o incrementar su valor en C unidades.
- La activación de la señal CARGA inicializará el valor de X con A, mientras que la activación de la señal INCREMENTA incrementará el valor de X en C unidades.
- También se cuenta con un comparador que evalúa la condición $X \leq B$. Si X es menor o igual a B, el resultado es la activación de la señal MENOR, en caso contrario, la señal MENOR permanece en cero.

- El algoritmo de la máquina de estados es el siguiente.

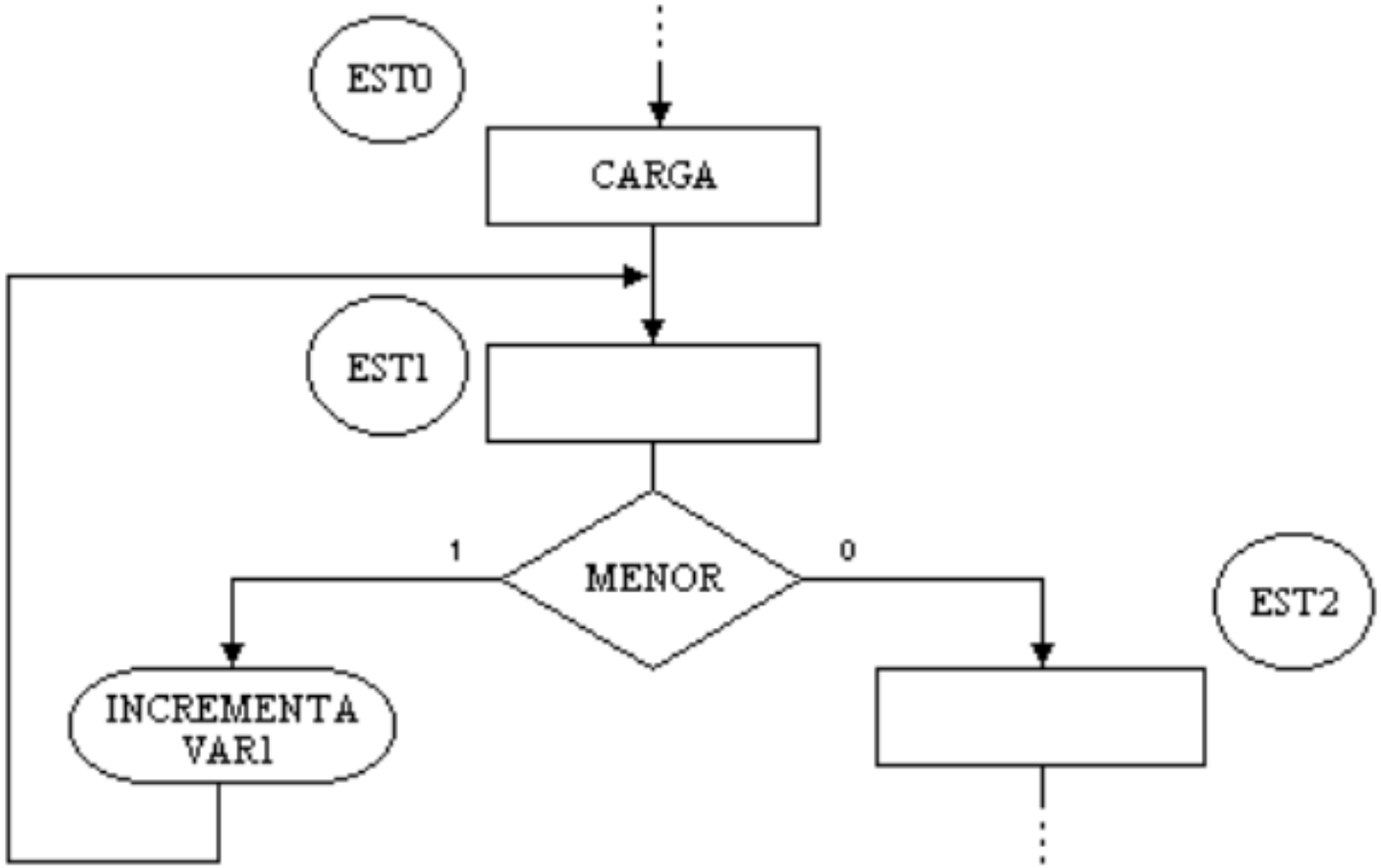
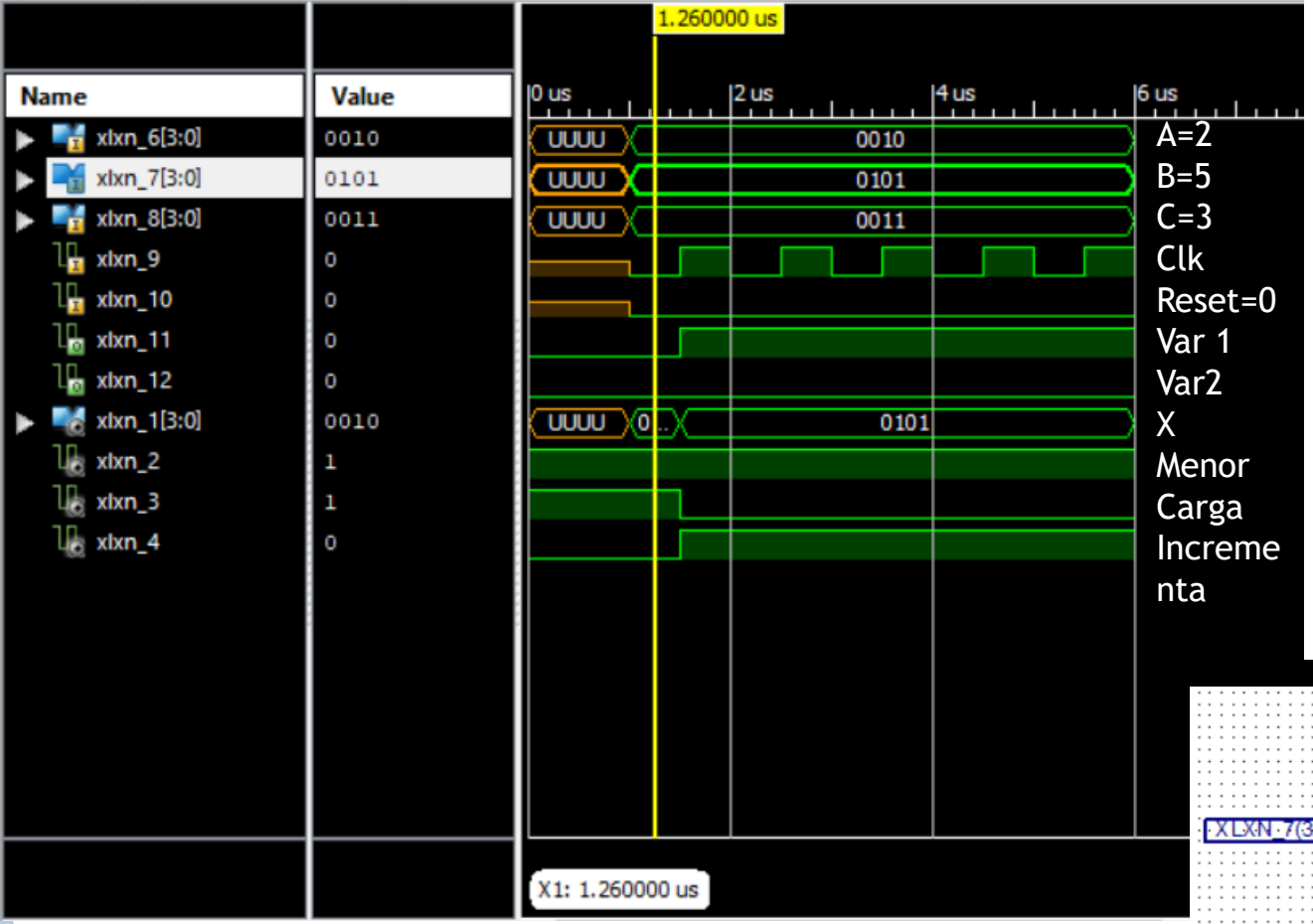


Figura 10. Carta ASM para el ejemplo 2.

- En el estado EST0 se activa la señal CARGA con el fin de cargar en el contador el valor inicial de X.
- En el estado EST1 se pregunta por la variable de entrada MENOR, si ésta es igual a cero, la condición $X \leq B$ es falsa.
- Si MENOR es igual a uno, la condición es verdadera y por tanto, son activadas las señales INCREMENTA y VAR1 como salidas condicionales.



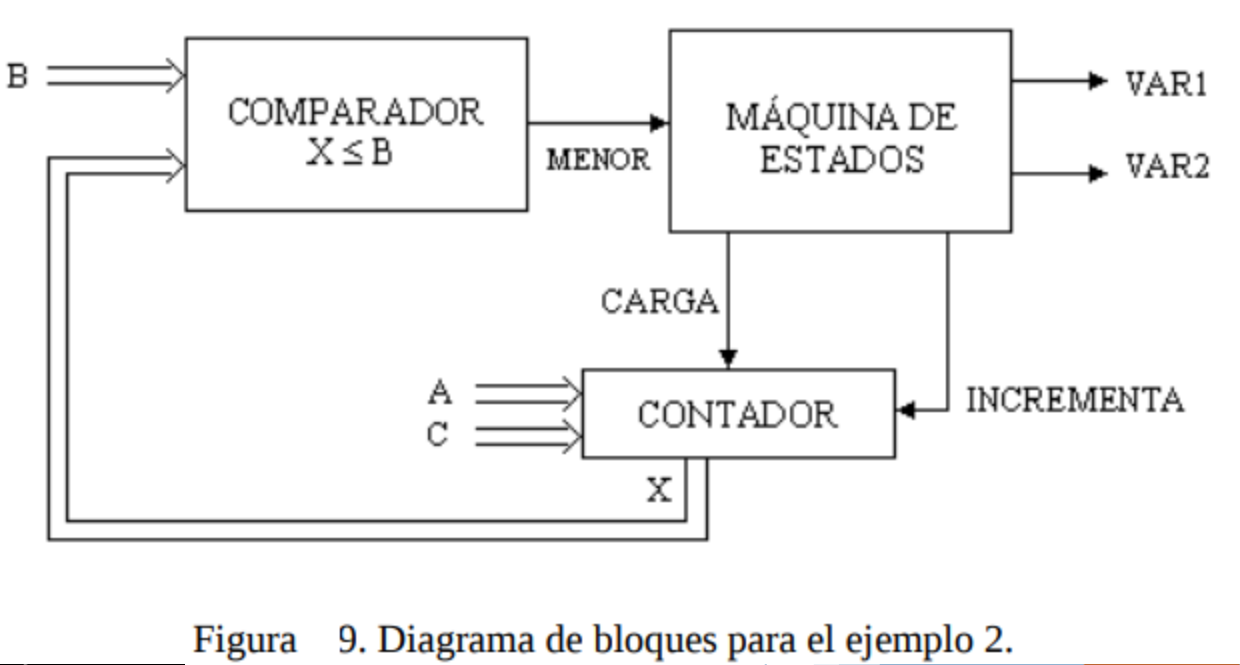
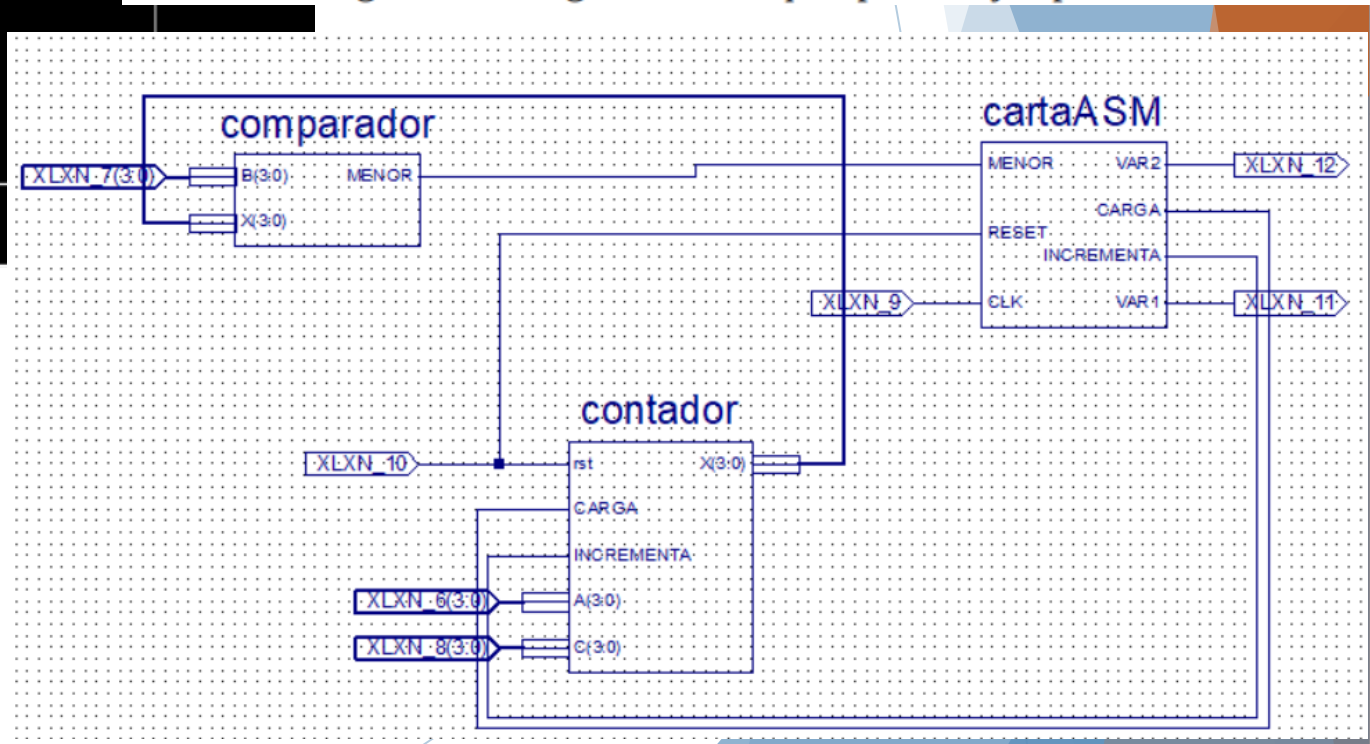


Figura 9. Diagrama de bloques para el ejemplo 2.

```
for (x = a; x ≤ b; x = x + c)
{ var1 = 1; var2 = 0; }
 var1 = 0;
```



Maquinas de estados en vhdl

Estilo de diseño 1: Plantilla

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------------
ENTITY <entity_name> IS
   PORT ( input: IN <data_type>;
          reset, clock: IN STD_LOGIC;
          output: OUT <data_type>);
END <entity_name>;
-------------------------------------------------------
ARCHITECTURE <arch_name> OF <entity_name> IS
   TYPE state IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: state;
BEGIN
   ---------- Lower section: -----------------------
   PROCESS (reset, clock)
   BEGIN
      IF (reset='1') THEN
         pr_state <= state0;
      ELSIF (clock'EVENT AND clock='1') THEN
         pr_state <= nx_state;
      END IF;
   END PROCESS;
```

Tipo enumerado: contiene todos los estados.

Sección secuencial.

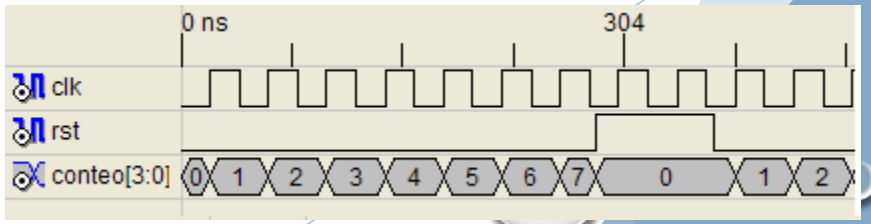
Sección Combinacional.

```
   ---------- Upper section: -------
   PROCESS (input, pr_state)
   BEGIN
      CASE pr_state IS
         WHEN state0 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state1;
            ELSE ...
            END IF;
         WHEN state1 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state2;
            ELSE ...
            END IF;
         WHEN state2 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state3;
            ELSE ...
            END IF;
         ...
      END CASE;
   END PROCESS;
END <arch_name>;
```

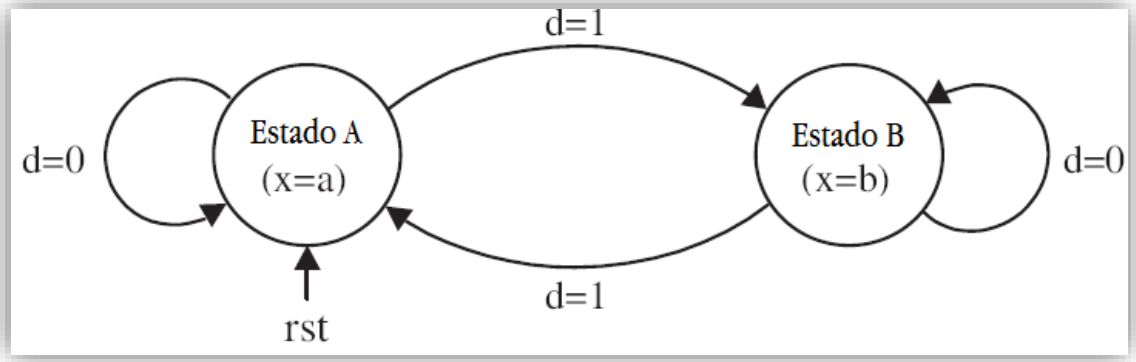
Ejemplo 1: contador BCD.

- Un contador es un ejemplo de una máquina de Moore, ya que la salida depende solo del estado presente.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
4  entity contador is
5      Port ( clk, rst : in  STD_LOGIC;
6             conteo : out  STD_LOGIC_VECTOR (3 downto 0));
7  end contador;
9  architecture maquina_d_estado of contador is
10 type estados is (cero, uno, dos, tres, cuatro,
11 cinco, seis, siete, ocho, nueve);
12 signal estado_pr, sig_estado: estados;
13 begin
14 ------- Sección inferior ---------
15    process(rst,clk) begin
16       if (rst='1') then
17          estado_pr<=cero;
18       elsif (clk'event and clk='1') then
19          estado_pr<=sig_estado;
20       end if;
21    end process;
22 ---- Sección superior ------
23 process(estado_pr) begin
24    case estado_pr is
25       when cero =>
26          conteo<="0000";
27          sig_estado<=uno;
28       when uno =>
29          conteo<="0001";
30          sig_estado<=dos;
34       when tres =>
35          conteo<="0011";
36          sig_estado<=cuatro;
37       when cuatro =>
38          conteo<="0100";
39          sig_estado<=cinco;
40       when cinco =>
41          conteo<="0101";
42          sig_estado<=seis;
43       when seis =>
44          conteo<="0110";
45          sig_estado<=siete;
46       when siete =>
47          conteo<="0111";
48          sig_estado<=ocho;
49       when ocho =>
50          conteo<="1000";
51          sig_estado<=nueve;
52       when nueve =>
53          conteo<="1001";
54          sig_estado<=cero;
55    end case;
56 end process;
57 end maquina_d_estado;
```



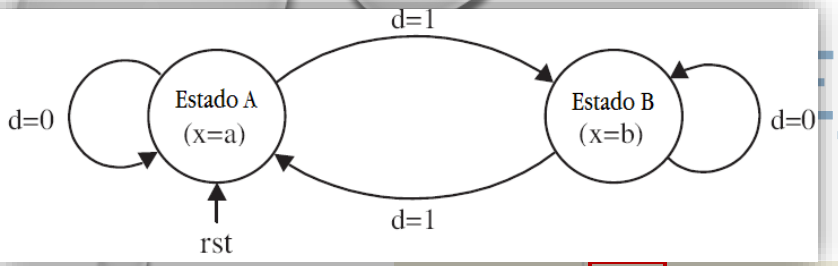
Ejemplo 2: FSM simple.



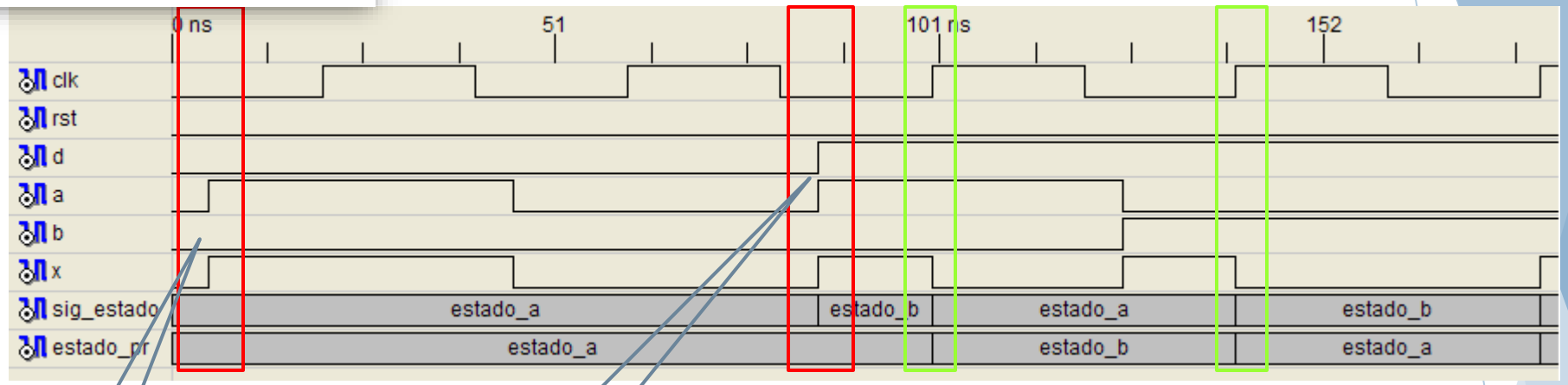

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.std_logic_arith.all;
 4 use ieee.std_logic_unsigned.all;
 6 entity simple_fsm is
 7     Port (a,b,d,clk,rst: in  std_logic;
 8           x: out  std_logic);
 9 end simple_fsm;
11 architecture fsm_sencilla of simple_fsm is
12 type estado is (estado_A,estado_B);
13 signal estado_pr, sig_estado: estado;
14 begin
15 --------- Sección inferior ---------
16    process (clk, rst) begin
17       if rst='1' then
18          estado_pr<=estado_A;
19       elsif (clk'event and clk='1') then
20          estado_pr<=sig_estado;
21       end if;
22    end process;
```

```
23 --------- Sección superior ---------
24    process (a,b,d,estado_pr) begin
25       case (estado_pr) is
26          when estado_A =>
27             x<=a;
28             if d='1' then
29                sig_estado<=estado_B;
30             else
31                sig_estado<=estado_A;
32             end if;
33          when estado_B =>
34             x<=b;
35             if d='1' then
36                sig_estado<=estado_A;
37             else
38                sig_estado<=estado_B;
39             end if;
40       end case;
41    end process;
42 end fsm_sencilla;
```

Ejemplo 2: FSM simple.




```
case (estado_pr) is
```

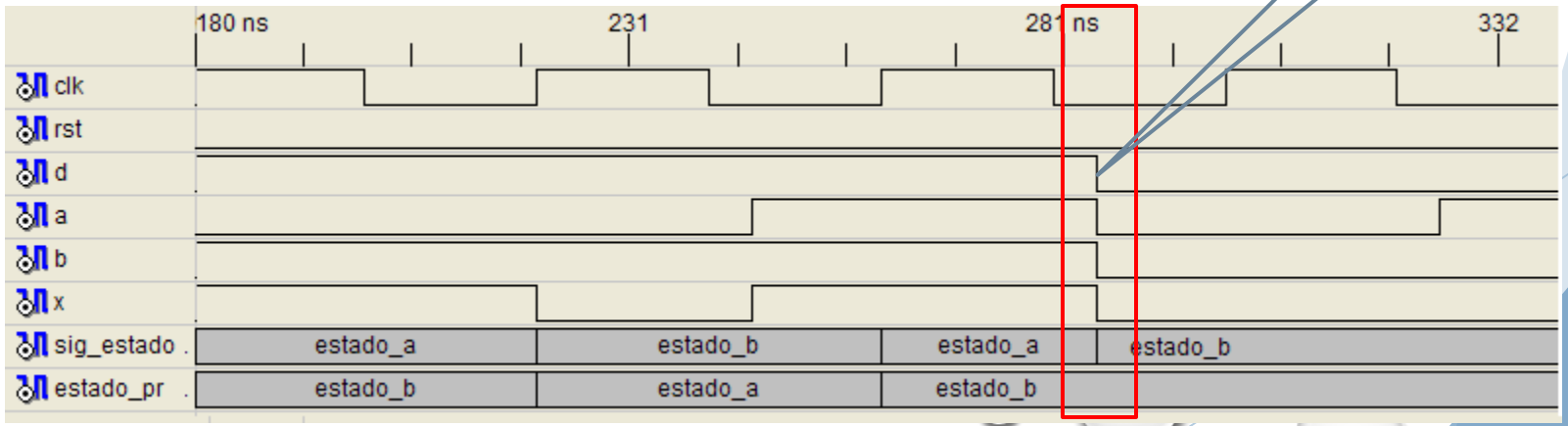


x = a

D = 1

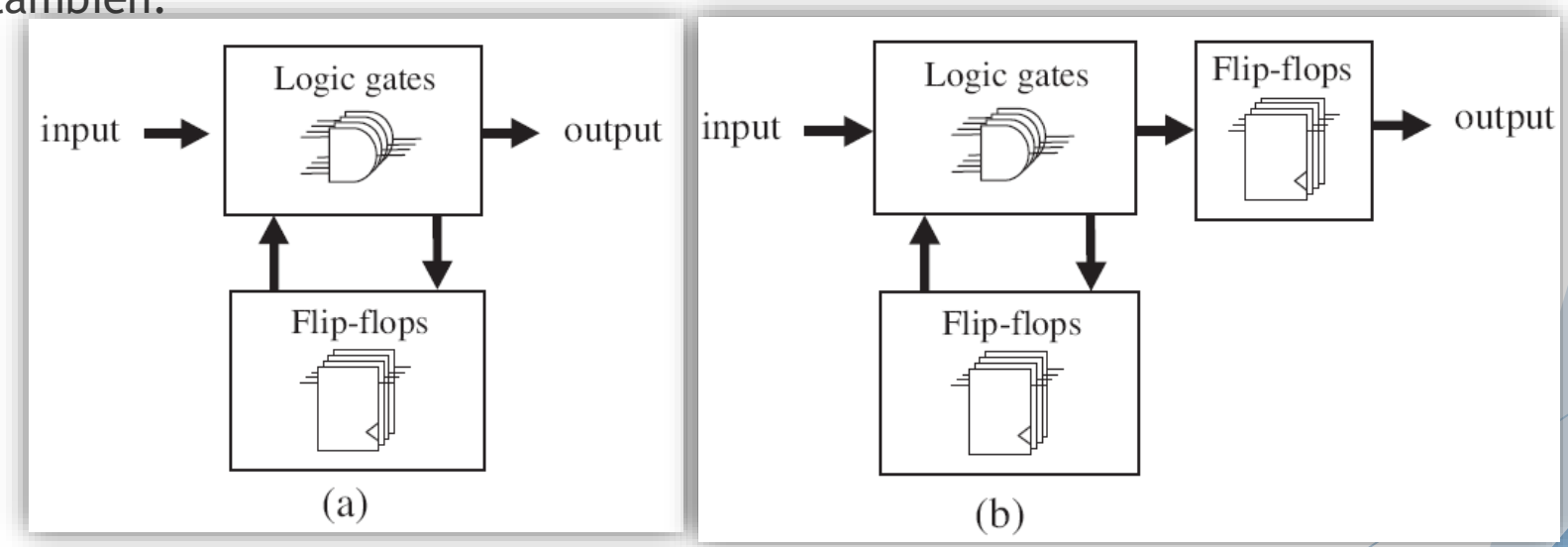
```
if d='1' then
   sig_estado<=estado_B;
```

D = 0



Estilo de diseño 2

- En el primer estilo de diseño, solo el estado presente (estado_pr) es almacenado.
- En este caso, si se usa una máquina de Mealy (la salida depende de la entrada actual), la salida cambia cuando la entrada cambia (salida asíncrona).
- En muchas aplicaciones, las señales deben ser síncronas, de modo que la salida debe actualizarse solo cuando existe un pulso de reloj.
- Para hacer una máquina de Mealy síncrona, la salida debe ser almacenada también.



El estilo de diseño 2, usa una señal adicional para contener el valor de la salida (en la sección superior), pero solo pasa este valor a la salida cuando un evento de reloj ocurre (sección inferior).

Estilo de diseño 2: Plantilla

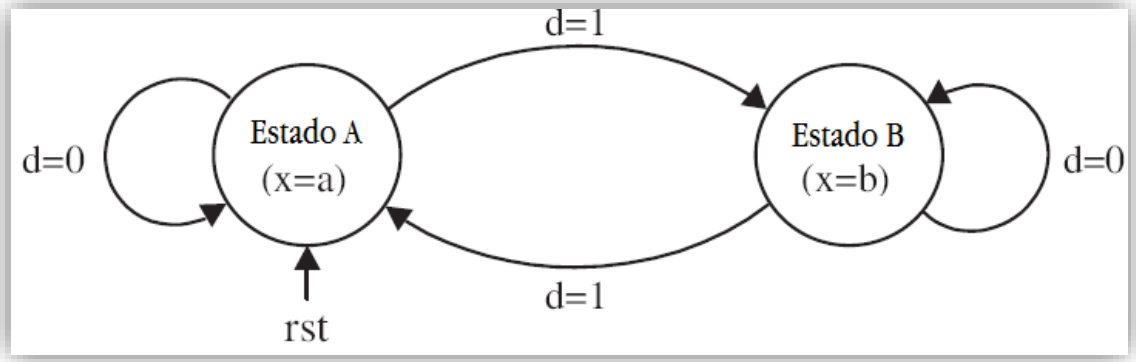
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------------
ENTITY <ent_name> IS
   PORT (input: IN <data_type>;
         reset, clock: IN STD_LOGIC;
         output: OUT <data_type>);
END <ent_name>;
-------------------------------------------------------
ARCHITECTURE <arch_name> OF <ent_name> IS
   TYPE states IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: states;
   SIGNAL temp: <data_type>;
BEGIN
   ---------- Lower section: ------------------------
   PROCESS (reset, clock)
   BEGIN
      IF (reset='1') THEN
         pr_state <= state0;
      ELSIF (clock'EVENT AND clock='1') THEN
         output <= temp;
         pr_state <= nx_state;
      END IF;
   END PROCESS;
```

La asignación de la salida solo se ejecuta al pulso de reloj.

El estilo de diseño 2, usa una señal adicional para contener el valor de la salida (en la sección superior), pero solo pasa este valor a la salida cuando un evento de reloj ocurre (sección inferior).

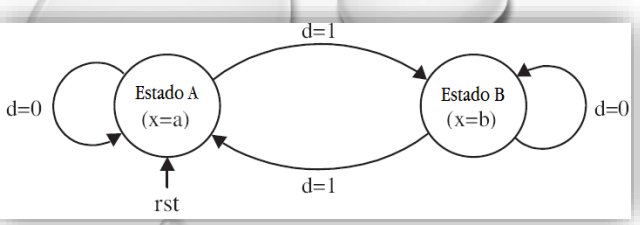
```
---------- Upper section: ----------------------
PROCESS (pr_state)
BEGIN
   CASE pr_state IS
      WHEN state0 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state1;
         ...
         END IF;
      WHEN state1 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state2;
         ...
         END IF;
      WHEN state2 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state3;
         ...
         END IF;
      ...
   END CASE;
END PROCESS;
END <arch_name>;
```

Estilo de diseño 2: FSM simple.

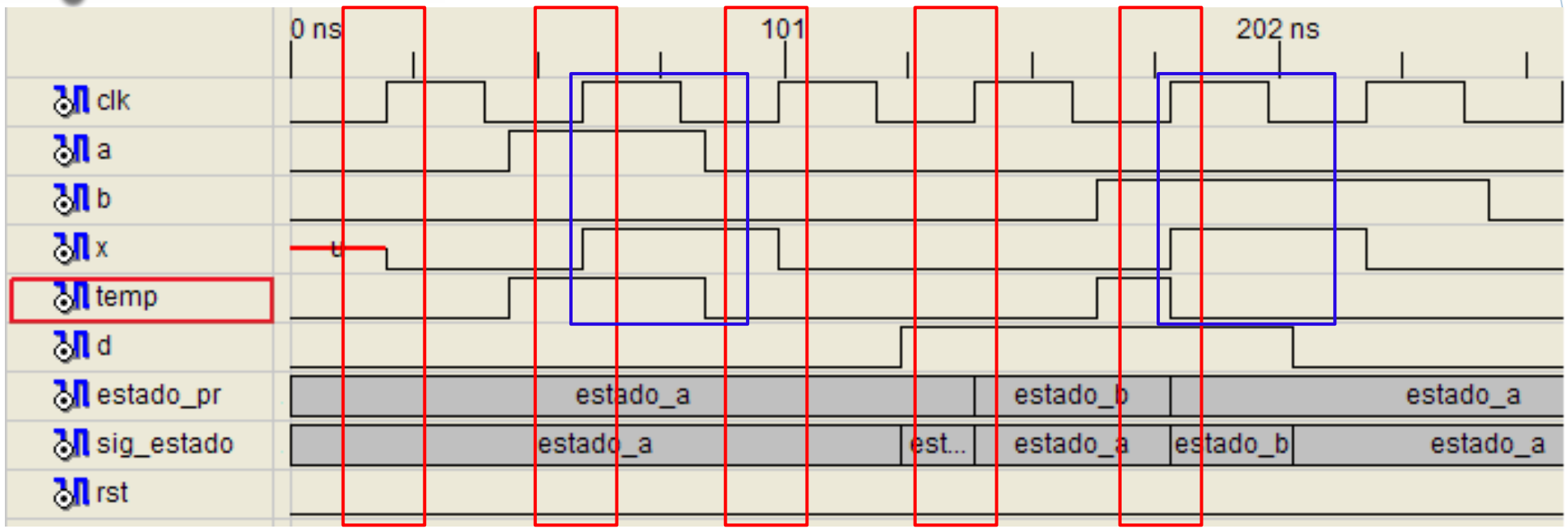



```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  use ieee.std_logic_arith.all;
 4  use ieee.std_logic_unsigned.all;
 6  entity fsm_dos is
 7      Port (a,b,d,clk,rst: in  std_logic;
 8            x: out  std_logic);
 9  end fsm dos;
11  architecture fsm_d2 of fsm_dos is
12  type estado is (estado_A,estado_B);
13  signal estado_pr, sig_estado: estado;
14  signal temp: std_logic;
15  begin
16  --------- Sección inferior ---------
17     process (clk, rst) begin
18        if rst='1' then
19           estado_pr<=estado_A;
20        elsif (clk'event and clk='1') then
21           x<=temp;
22           estado_pr<=sig_estado;
23        end if;
24     end process;
```

```
25 --------- Sección superior ---------
26     process (a,b,d,estado_pr) begin
27        case (estado_pr) is
28           when estado_A =>
29              temp<=a;
30              if d='1' then
31                 sig_estado<=estado_B;
32              else
33                 sig_estado<=estado_A;
34              end if;
35           when estado_B =>
36              temp<=b;
37              if d='1' then
38                 sig_estado<=estado_A;
39              else
40                 sig_estado<=estado_B;
41              end if;
42        end case;
43     end process;
44 end fsm_d2;
```



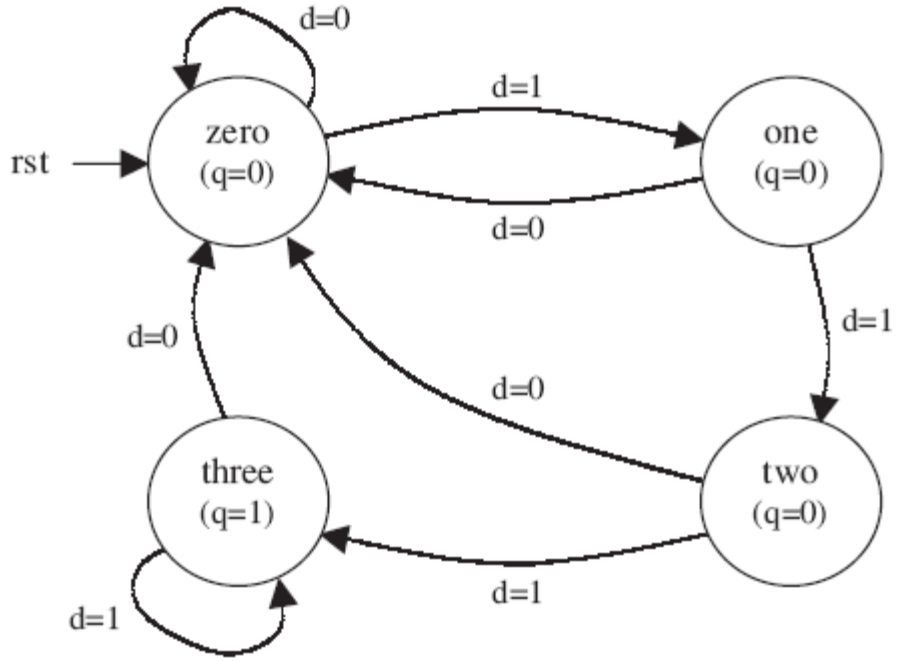
Estilo de diseño 2: FSM simple.



- Cuando una señal es almacenada, su valor permanece estático entre dos flancos de reloj. Por lo tanto, si la entrada (a o b) cambia durante este intervalo, el cambio no se observará por el circuito.

Ejemplo: Detector de secuencia.

- El circuito tiene como entrada una cadena serial de bits y su salida será 1 cuando la secuencia "111" ocurra. En caso de solapamiento, esto es, una secuencia 0111110 suceda, la salida debe mantenerse activa por tres ciclos consecutivos de reloj.



Ejemplo: Detector de secuencia.

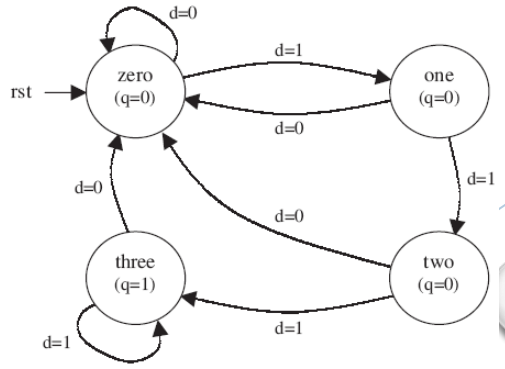
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
6  entity detec_cadena is
7      Port ( d,clk,reset : in  STD_LOGIC;
8             q : out  STD_LOGIC);
9  end detec_cadena;
11 architecture arq of detec_cadena is
12 type estados is (cero, uno, dos, tres);
13 signal estado_pr, sig_estado: estados;
14 begin
15 ---------- Sección inferior ----------
16    process (clk, reset) begin
17       if reset='1' then
18          estado_pr<=cero;
19       elsif (clk'event and clk='1') then
20          estado_pr<=sig_estado;
21       end if;
22    end process;
```

```
23 ---------- Sección superior -----
24    process (d, estado_pr) begin
25       case (estado_pr) is
26          when cero =>
27             q<='0';
28             if d='1' then
29                sig_estado<=uno;
30             else
31                sig_estado<=cero;
32             end if;
33          when uno =>
34             q<='0';
35             if d='1' then
36                sig_estado<=dos;
37             else
38                sig_estado<=cero;
39             end if;
40          when dos =>
41             q<='0';
42             if d='1' then
43                sig_estado<=tres;
44             else
45                sig_estado<=cero;
46             end if;
47          when tres =>
48             q<='1';
49             if d='1' then
50                sig_estado<=tres;
51             else
52                sig_estado<=cero;
53             end if;
54       end case;
55    end process;
56 end arq;
```

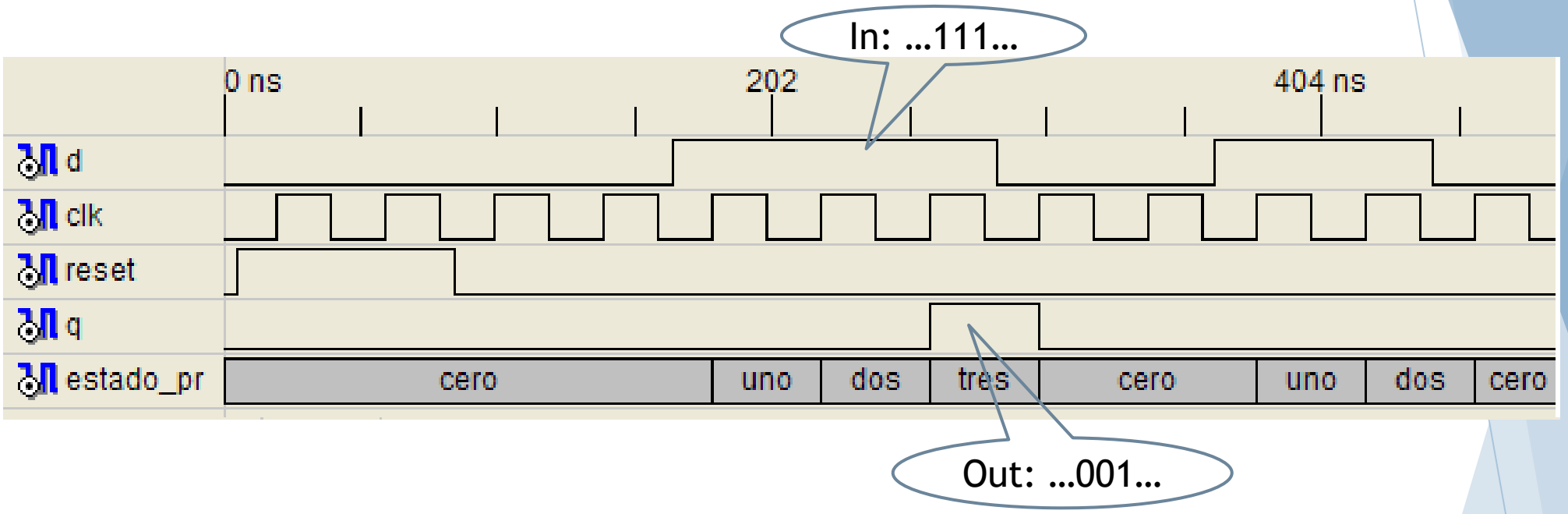
Solapamiento, la salida sigue siendo 1.

```
23 ---------- Sección superior -----
24     process (d, estado_pr) begin
25         case (estado_pr) is
26             when cero =>
27                 q<='0';
28                 if d='1' then
29                     sig_estado<=uno;
30                 else
31                     sig_estado<=cero;
32                 end if;
33             when uno =>
34                 q<='0';
35                 if d='1' then
36                     sig_estado<=dos;
37                 else
38                     sig_estado<=cero;
39                 end if;
```

```
40                 when dos =>
41                     q<='0';
42                     if d='1' then
43                         sig_estado<=tres;
44                     else
45                         sig_estado<=cero;
46                     end if;
47                 when tres =>
48                     q<='1';
49                     if d='1' then
50                         sig_estado<=tres;
51                     else
52                         sig_estado<=cero;
53                     end if;
54         end case;
55     end process;
56 end arq;
```

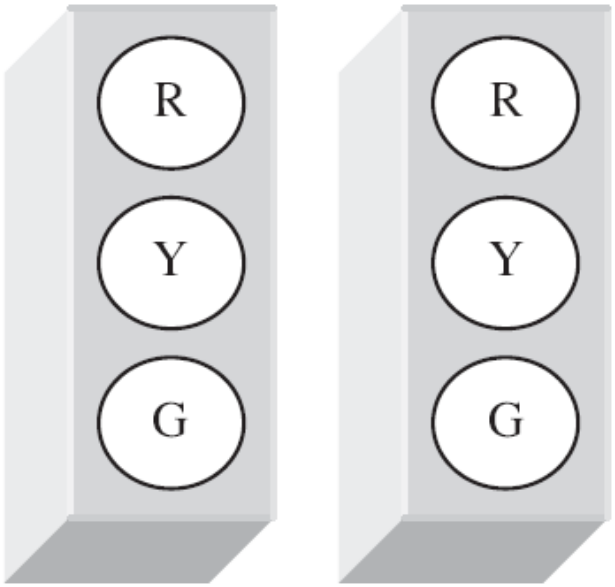


Ejemplo: Detector de secuencia (TB)



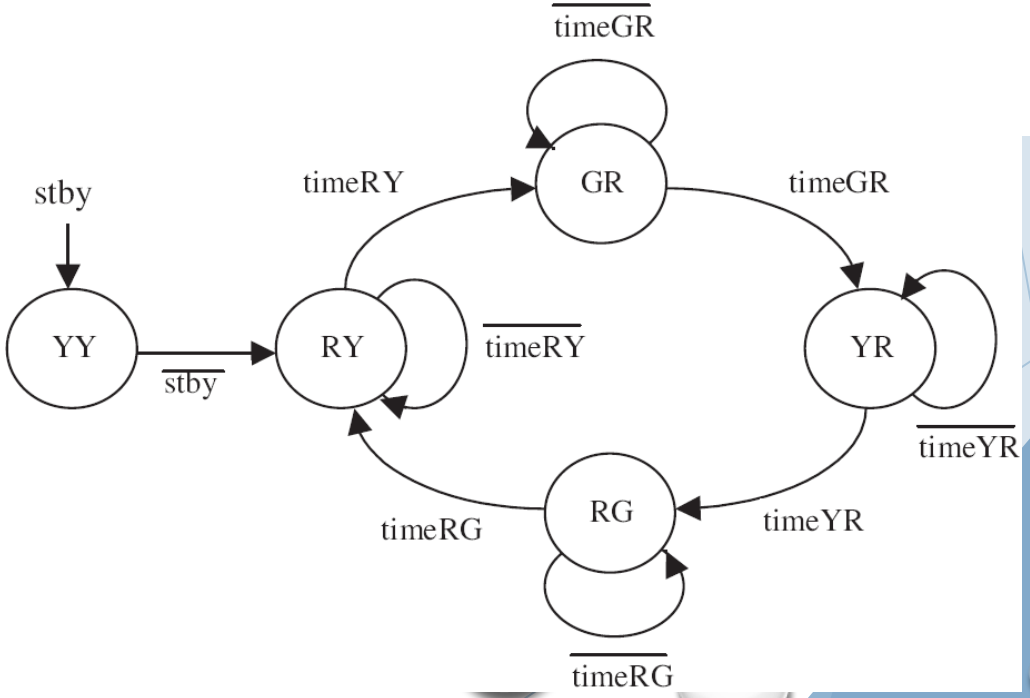
Notar que la salida no depende de la entrada actual. Todas las asignaciones a q son incondicionales (esto es, no dependen de d). Por lo tanto, la salida es automáticamente sincronizada.

Ejemplo: controlador de semáforo.



Estado	Modo de operación		
	REGULAR	TEST	STANDBY
	Tiempo	Tiempo	Tiempo
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinido

Se asume que la frecuencia de reloj es 60Hz (tomada de la misma fuente de poder).

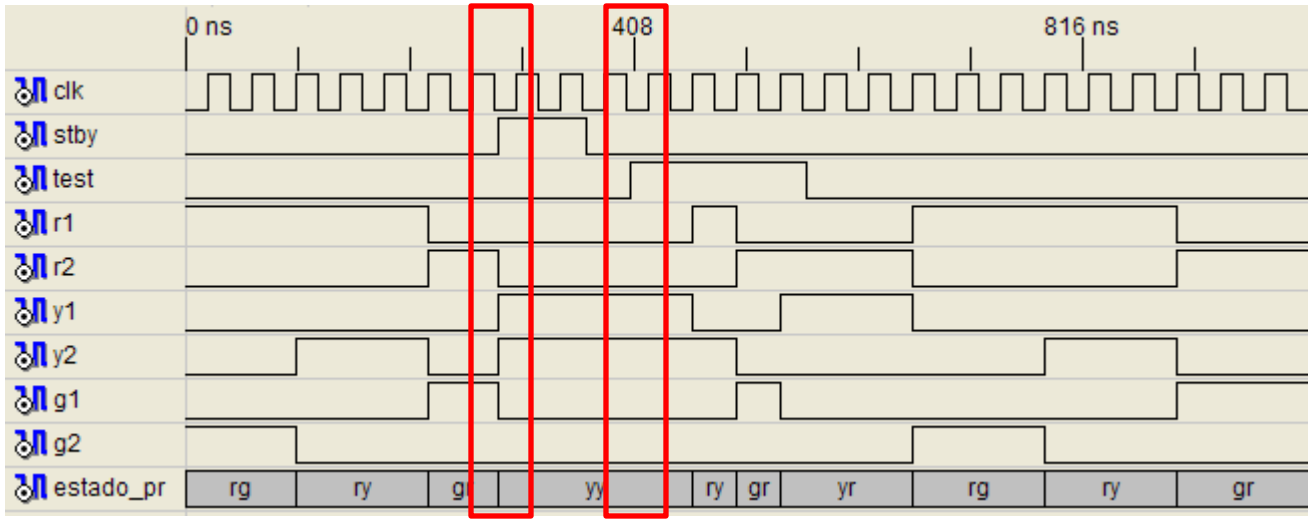


Ejemplo: controlador de semáforo

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
6  entity semaforo is
7      Port ( clk,stby,test : in  STD_LOGIC;
8             r1,r2,y1,y2,g1,g2 : out  STD_LOGIC);
9  end semaforo;
11 architecture Behavioral of semaforo is
12   constant timeMax : integer:=2700;--2700/60=45s
13   constant timeRG  : integer:=1800;--1800/60=30s
14   constant timeRY  : integer:= 300;-- 300/60= 5s
15   constant timeGR  : integer:=2700;--2700/4 = 5s
16   constant timeYR  : integer:= 300;-- 300/60= 5s
17   constant timeTest: integer:=  60;--  60/60= 1s
18   type estados is (RG,RY,GR,YR,YY);
19   signal estado_pr, sig_estado: estados;
20   signal tempo: integer range 0 to timeMax;
21 begin
22 ---- Sección inferior ----
23    process (clk, stby)
24      variable cont: integer range 0 to timeMax;
25    begin
26       if (stby='1') then
27          estado_pr<=YY;
28          cont:=0;
29       elsif (clk'event and clk='1') then
30          cont:=cont + 1;
31          if cont=tempo then
32             estado_pr<=sig_estado;
33             cont:=0;
34          end if;
35       end if;
36    end process;
```

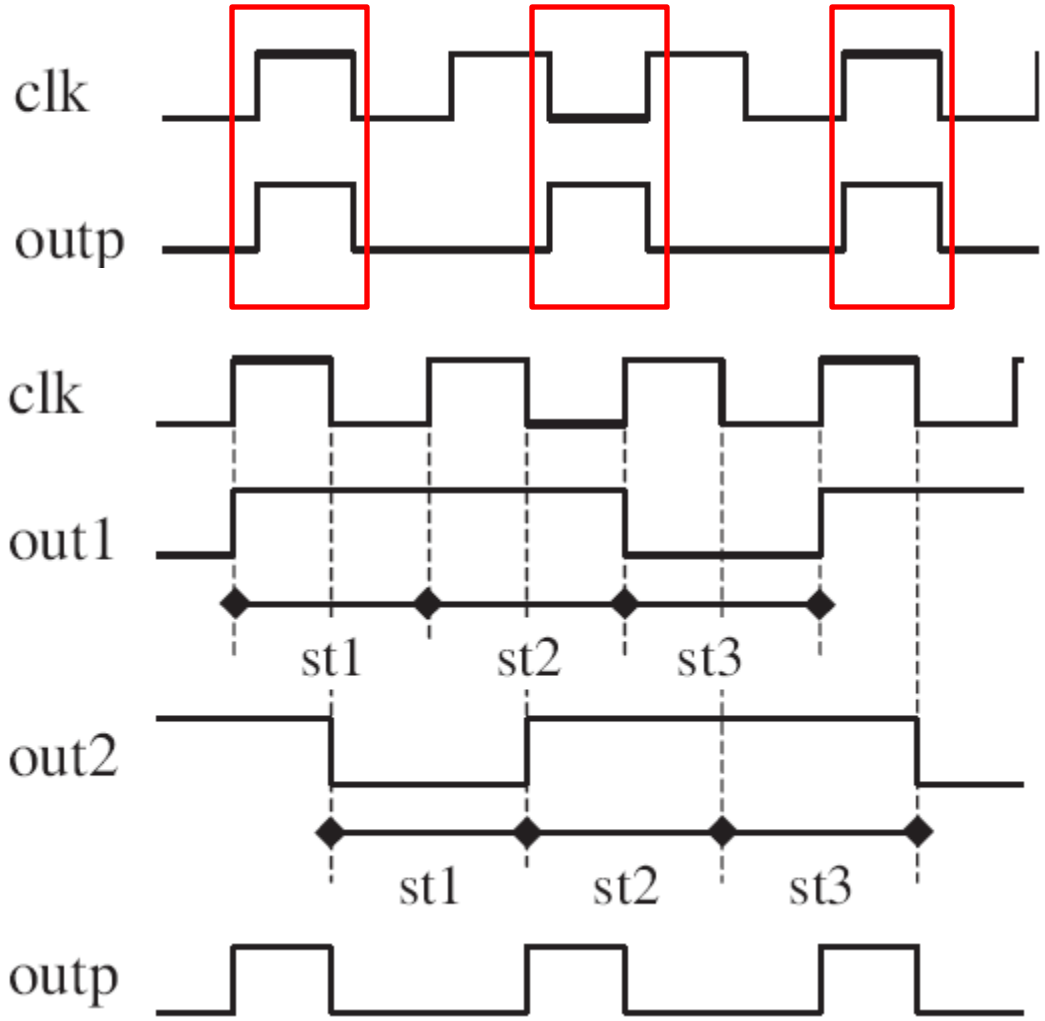
```
37 ---- Sección superior ----
38    process(estado_pr,test) begin
39       case estado_pr is
40          when RG =>
41             r1<='1';r2<='0';y1<='0';
42          y2<='0';g1<='0';g2<='1';
43             sig_estado<=RY;
44             if test='0' then tempo <=timeRG;
45             else tempo <=timeTest;
46             end if;
47          when RY =>
48             r1<='1';r2<='0';y1<='0';
49             y2<='1';g1<='0';g2<='0';
50             sig_estado<=GR;
51             if test='0' then tempo <=timeRY;
52             else tempo <=timeTest;
53             end if;
54          when GR =>
55             r1<='0';r2<='1';y1<='0';
56             y2<='0';g1<='1';g2<='0';
57             sig_estado<=YR;
58             if test='0' then tempo <=timeGR;
59             else tempo <=timeTest;
60             end if;
61          when YR =>
62             r1<='0';r2<='1';y1<='1';
63             y2<='0';g1<='0';g2<='0';
64             sig_estado<=RG;
65             if test='0' then tempo <=timeYR;
66             else tempo <=timeTest;
67             end if;
68          when YY =>
69             r1<='0';r2<='0';y1<='1';
70             y2<='1';g1<='0';g2<='0';
71             sig_estado<=RY;
72       end case;
73    end process;
74 end Behavioral;
```

Ejemplo: controlador de semáforo.



- Para visualizar la salida de forma óptima, se ha modificado el valor de todas las contantes a 3, excepto el valor de timeTest, que vale 1. Se espera que el sistema cambie de estado cada tres ciclos de reloj cuando está en modo de operación regular, o cada ciclo de reloj si está en mode Test.

Ejemplo: generador de funciones.

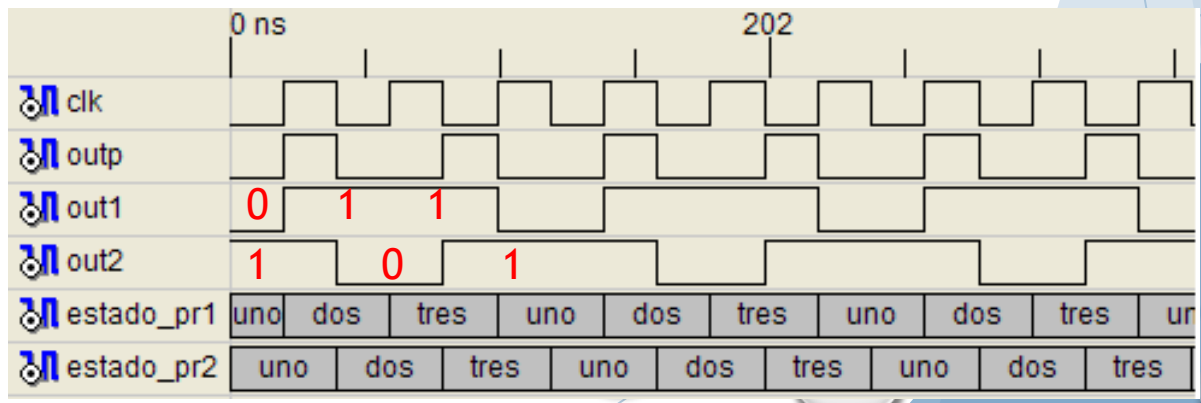


- Se diseñará dos máquinas de estado, una que operará exclusivamente en la transición positiva del reloj y otra que operará exclusivamente en el flanco negativo. Esto generará la señal alternativa out 1 y out2 . Estas señales serán tratadas con la operación *and* para obtener la señal deseada.
- Notar que el circuito no posee entradas externas (excepto la señal de reloj), de modo que la salida cambiará solo cuando la señal de reloj cambie (salida síncrona).

Ejemplo: generador de funciones.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
6  entity gen_senal is
7      Port ( clk : in  STD_LOGIC;
8             outp : out  STD_LOGIC);
9  end gen_senal;
11 architecture Behavioral of gen_senal is
12     type estados is (uno, dos, tres);
13     signal estado_pr1, sig_estado1: estados;
14     signal estado_pr2, sig_estado2: estados;
15     signal out1, out2: std_logic;
16 begin
17 ---- Sección inferior máquina 1 ----
18     process (clk) begin
19         if clk'event and clk='1' then
20             estado_pr1<=sig_estado1;
21         end if;
22     end process;
23 ---- sección superior máquina 1 ----
24     process (estado_pr1)begin
25         case estado_pr1 is
26             when uno =>
27                 out1<='0';
28                 sig_estado1<=dos;
29             when dos=>
30                 out1<='1';
31                 sig_estado1<=tres;
32             when tres=>
33                 out1<='1';
34                 sig_estado1<=uno;
35         end case;
36     end process;
```

```
37 ---- Sección inferior máquina 2 ----
38     process (clk) begin
39         if clk'event and clk='0' then
40             estado_pr2<=sig_estado2;
41         end if;
42     end process;
43 ---- sección superior máquina 2 ----
44     process (estado_pr2)begin
45         case estado_pr2 is
46             when uno =>
47                 out2<='1';
48                 sig_estado2<=dos;
49             when dos=>
50                 out2<='0';
51                 sig_estado2<=tres;
52             when tres=>
53                 out2<='1';
54                 sig_estado2<=uno;
55         end case;
56     end process;
57 outp<= out1 and out2;
58 end Behavioral;
```



Ejercicio

3) Convertir el siguiente código en lenguaje ‘C’ a un sistema secuencial.

```
while( x==0 ) {
        var5 = 1;
        var2 = 1;
        if( z==0 ){
                x = 1;
                var5 = 0;
} }
var5 = 0;
var2 = 0;
```

- 4) Convertir el siguiente código en lenguaje ‘C’ a una carta ASM.

```
if ( x==n ) {
        var1 = 1;
        var2 = 0; }
else {
        var1 = 0;
        var2 = 1; }
var1 = 0;
var2 = 0;
```

- 4) Convertir el siguiente código en lenguaje ‘C’ a una carta ASM.

```
if ( x==n ) {
      var1 = 1;
      var2 = 0; }
else {
      var1 = 0;
      var2 = 1; }
var1 = 0;
var2 = 0;
```

x y n son de un sólo bit.
Para hacer la comparación de las variables x y n se usa la función lógica XOR.
La tabla XOR se presenta a continuación.

Entradas		Salida
x	n	XOR
0	0	0
0	1	1
1	0	1
1	1	0

El diagrama de bloques que ejecuta el código anterior en C se presenta enseguida.

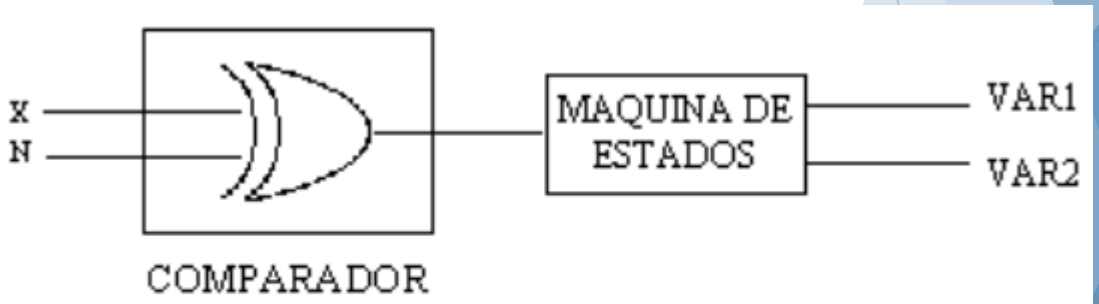


Figura 13. Diagrama de bloques para el ejemplo 4.

- Y el algoritmo de esta máquina de estados es el siguiente.

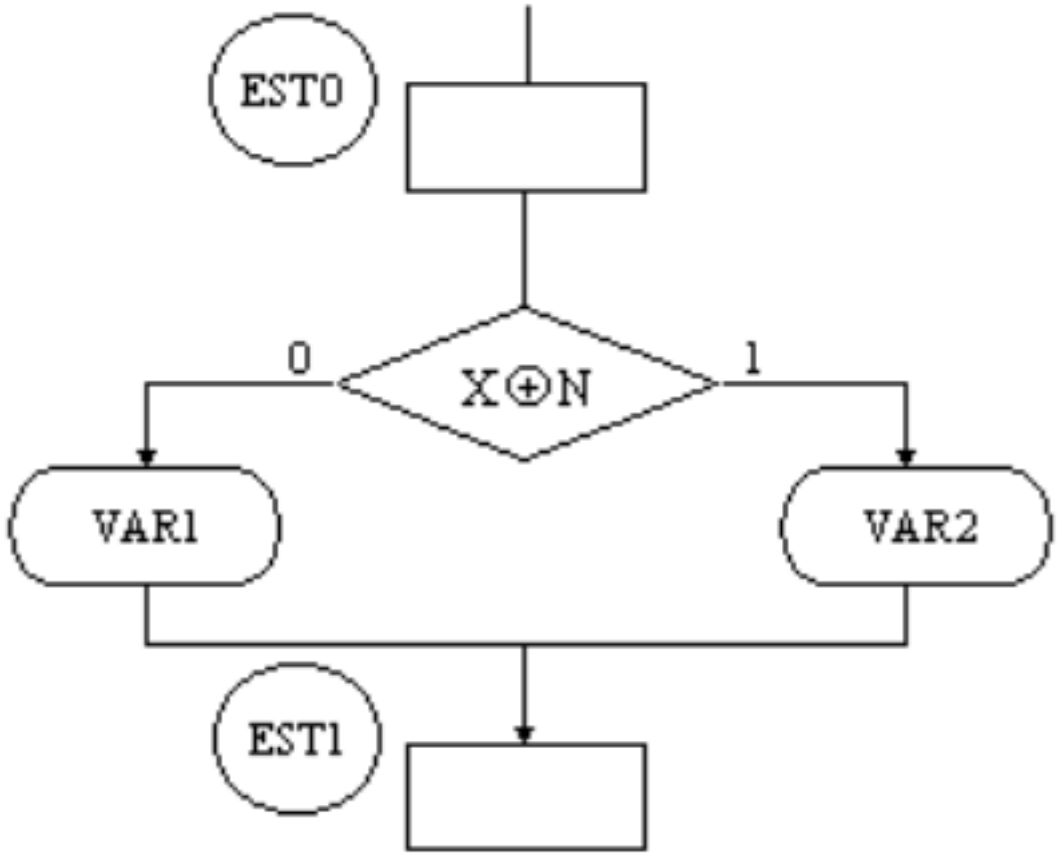


Figura 14. Carta ASM para el ejemplo 4.

En el diagrama se observa que si deseamos activar una señal de salida condicional, en un estado en particular, es necesario colocar su nombre dentro de un óvalo.

5) Usando un diagrama de tiempos mostrar la diferencia entre las cartas ASM de la figura, en donde la variable de salida VAR2 está como salida condicional en la carta ASM1 y como salida incondicional en la carta ASM2.

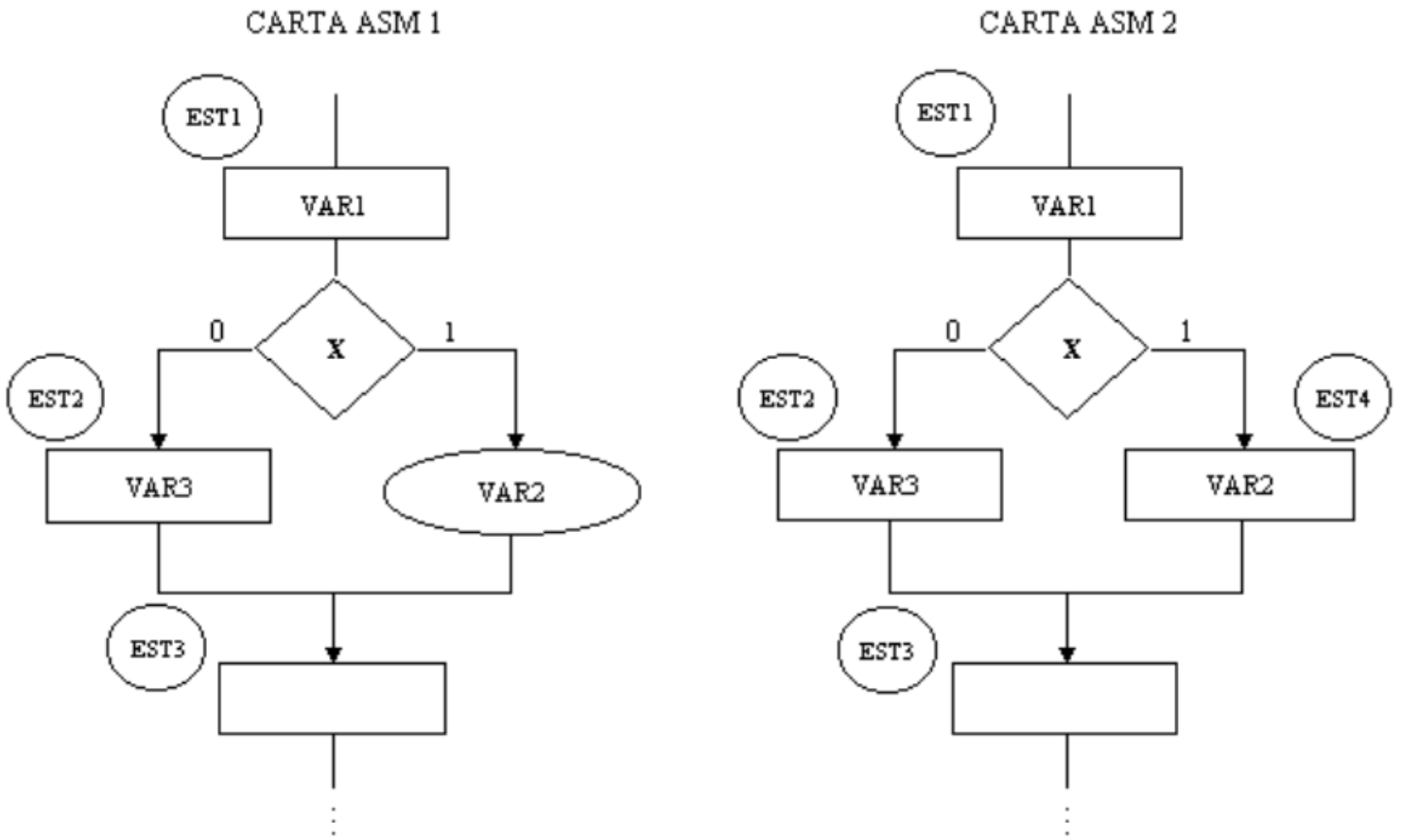
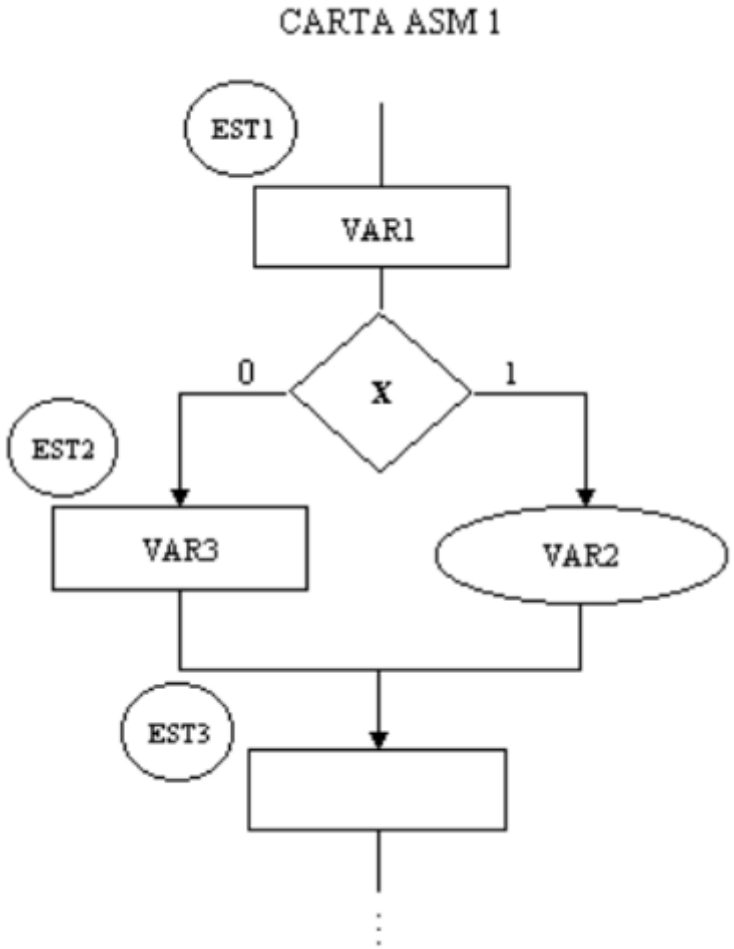
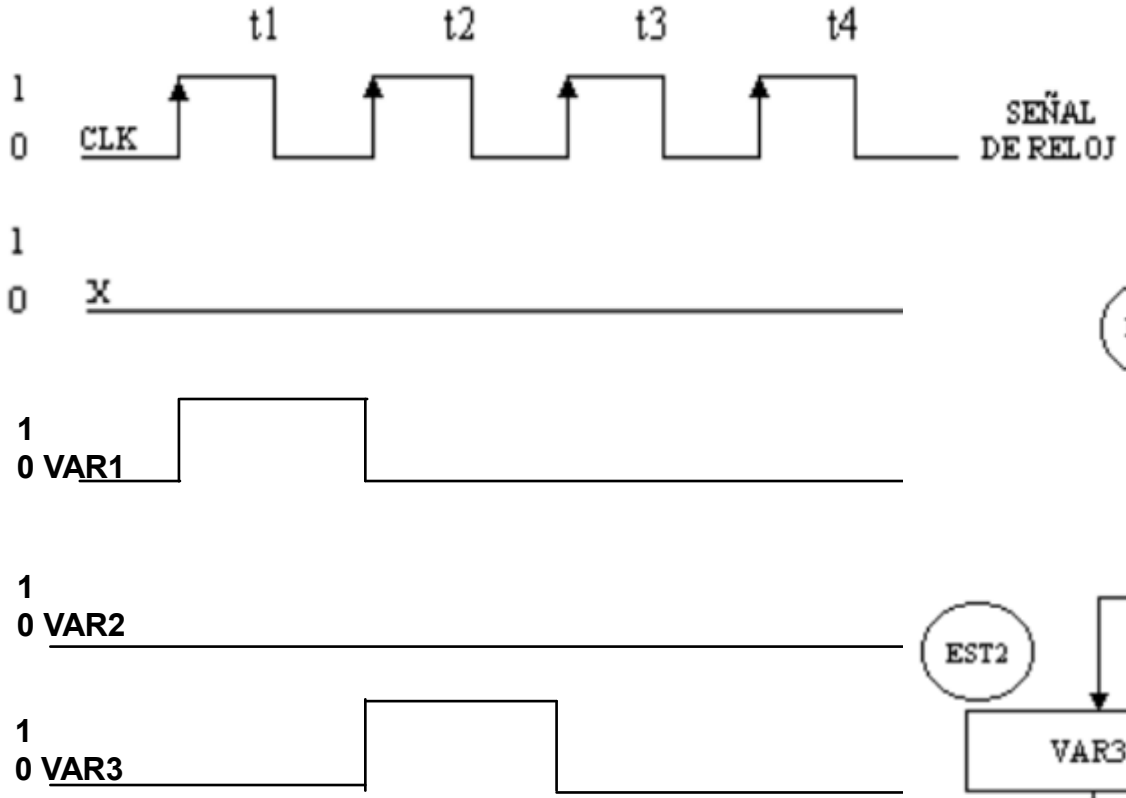
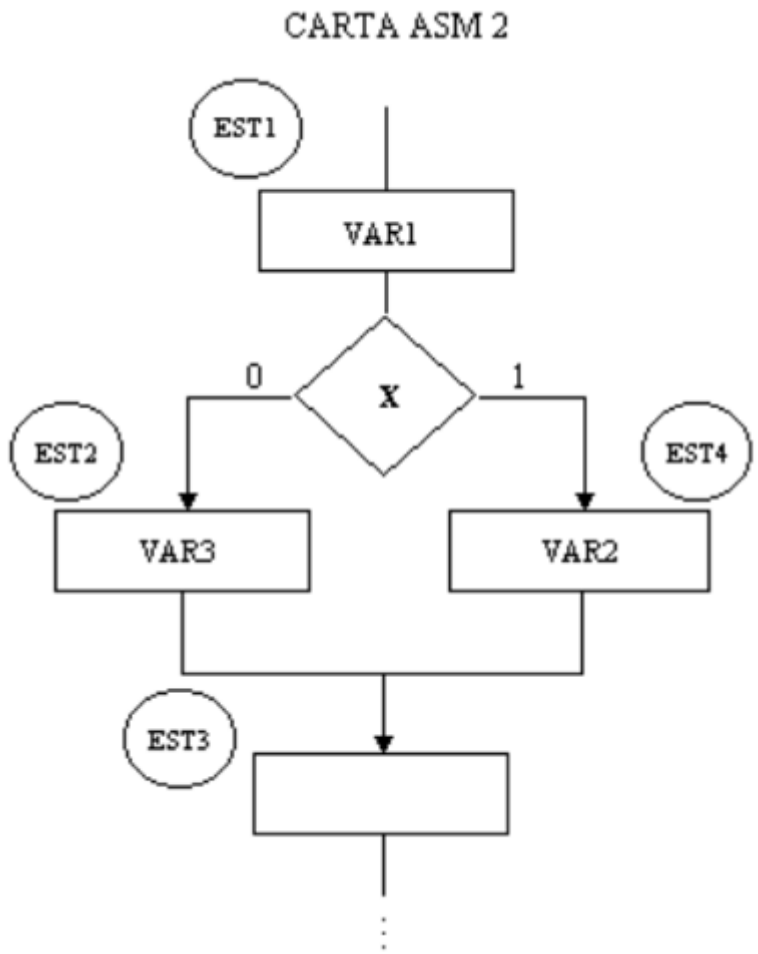


Figura 15. La salida VAR2 en la carta ASM1 se presenta como salida condicional, mientras que en la carta ASM2 se presenta como no condicional.

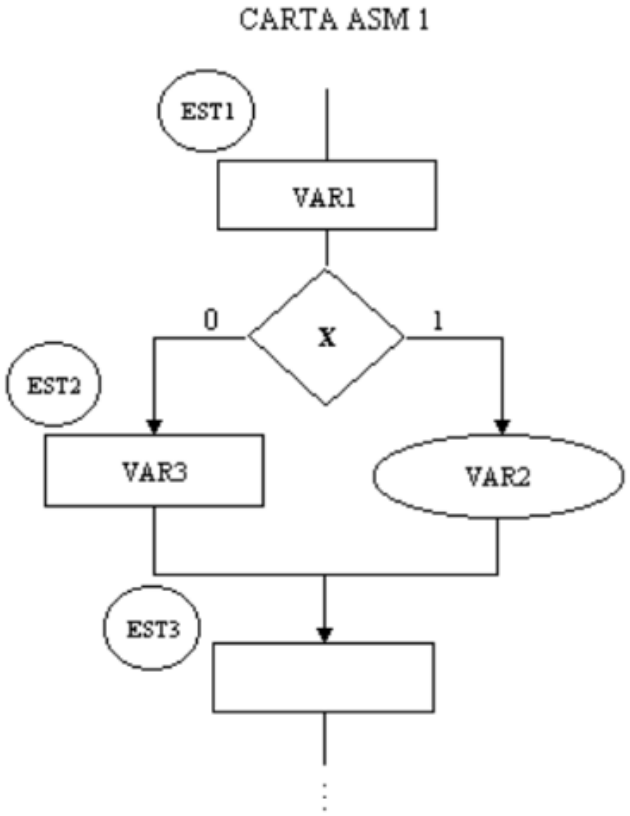
Para x=0

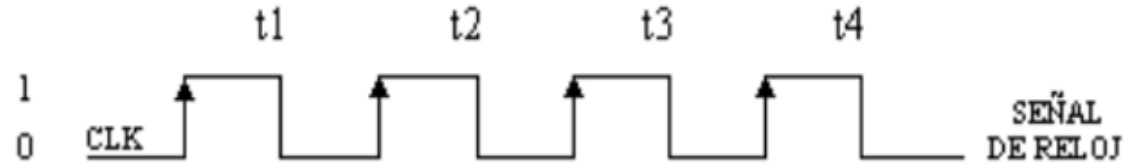

CARTA ASM 1
EST1
VAR1
0
X
1
EST2
VAR3
VAR2
EST3


t1
t2
t3
t4
1
0
CLK
SEÑAL DE RELOJ
1
0
X
1
0 VAR1
1
0 VAR2
1
0 VAR3

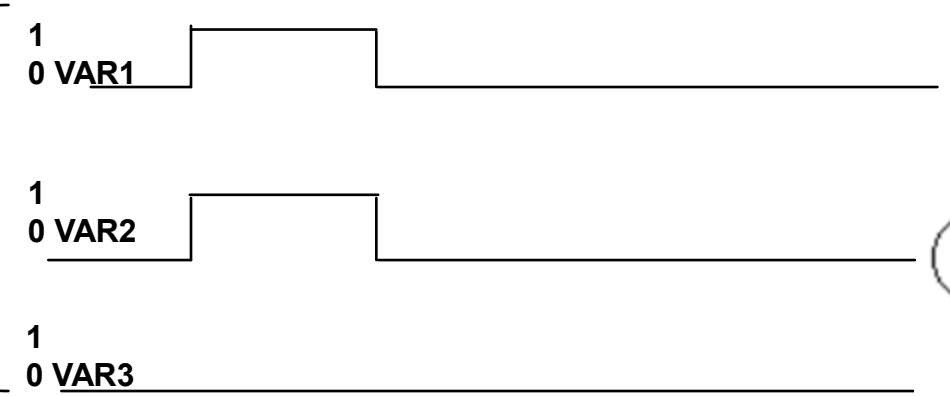

CARTA ASM 2
EST1
VAR1
0
X
1
EST2
EST4
VAR3
VAR2
EST3

Para x=1

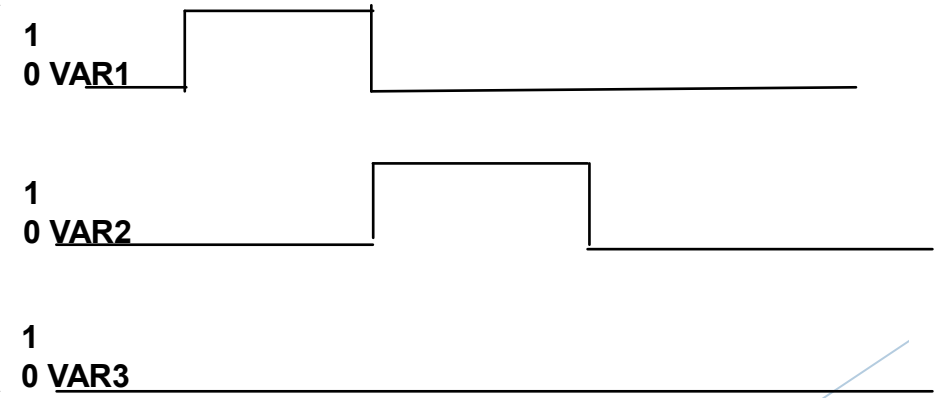


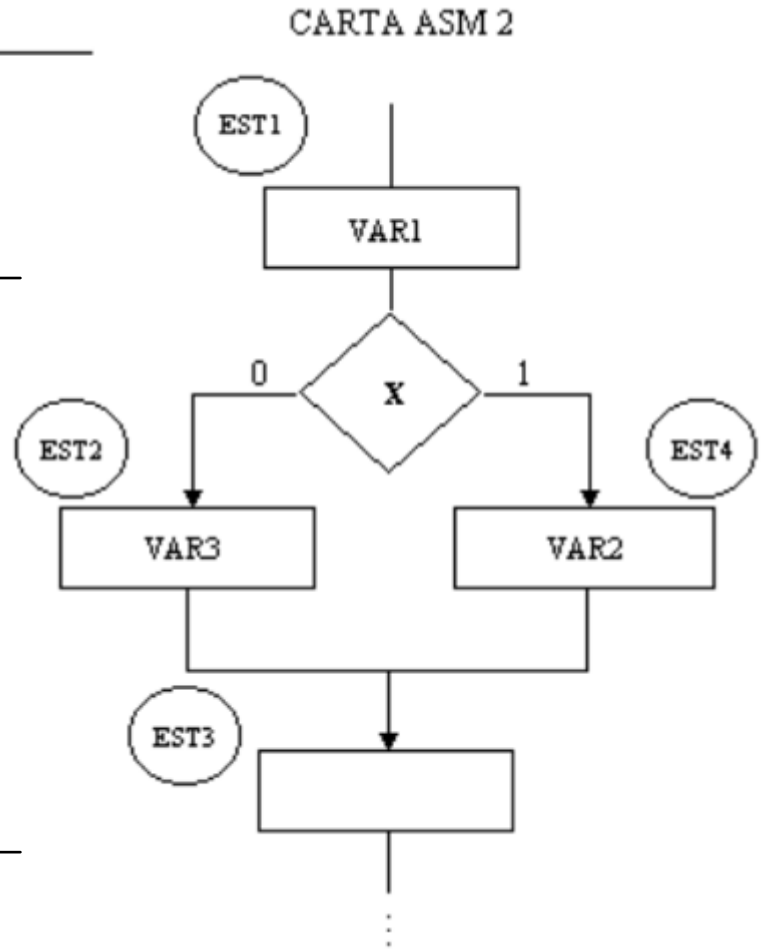


Carta ASM1



Carta ASM2

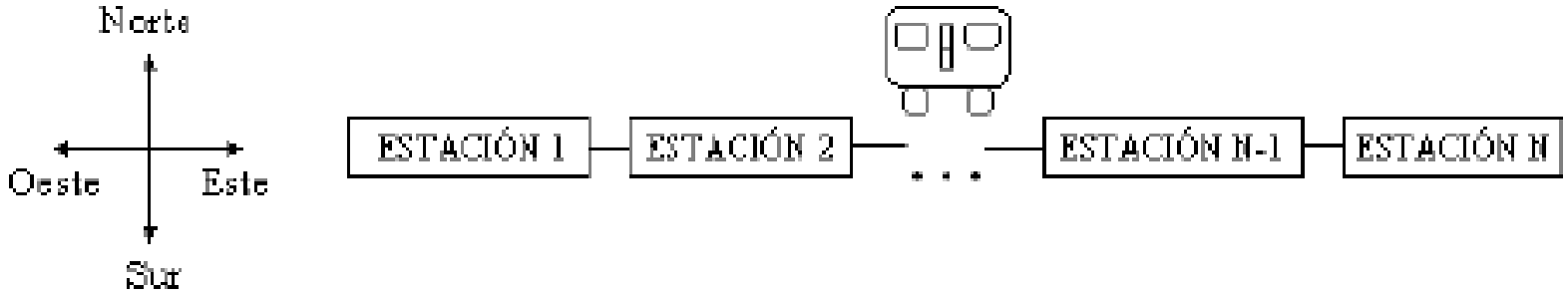




Tarea

- Diseñe el algoritmo de una máquina de estados, carta ASM, que genere la siguiente secuencia binaria: 000, 010, 101, 111, 110, 011, 100, 001.

Ejercicio



- El objetivo es diseñar un sistema digital, usando maquinas de estados, que mueva al tren de derecha a izquierda sobre la linea. En cada estacion hay unos sensores que detectan la entrada de un tren, de manera que cuando arriba a una de ellas, hace una parada de dos minutos.
- Ademas, existe un boton de emergencia en los vagones que hace que el tren se detenga un minuto extra en la estacion, si asi se requiriera.

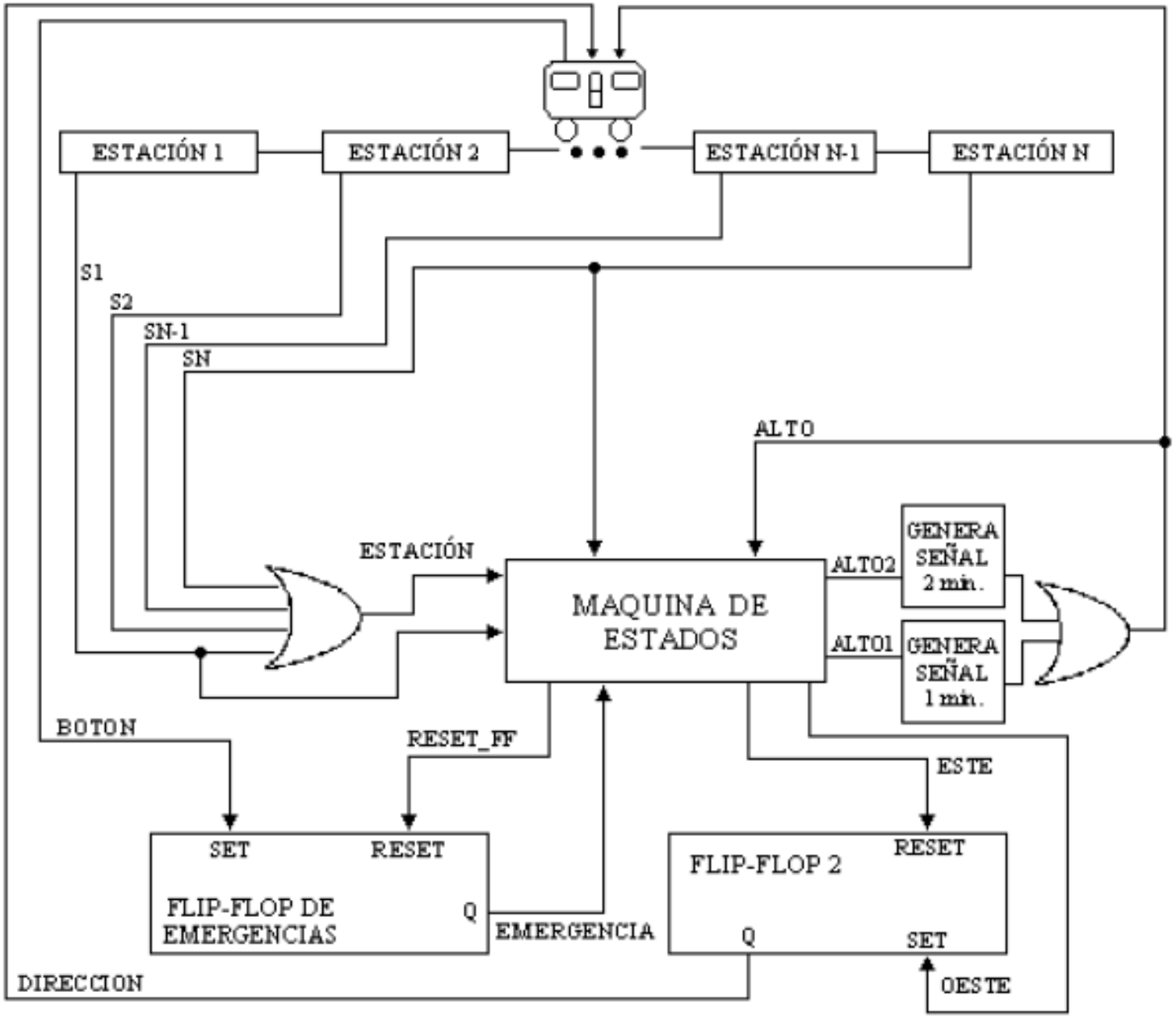
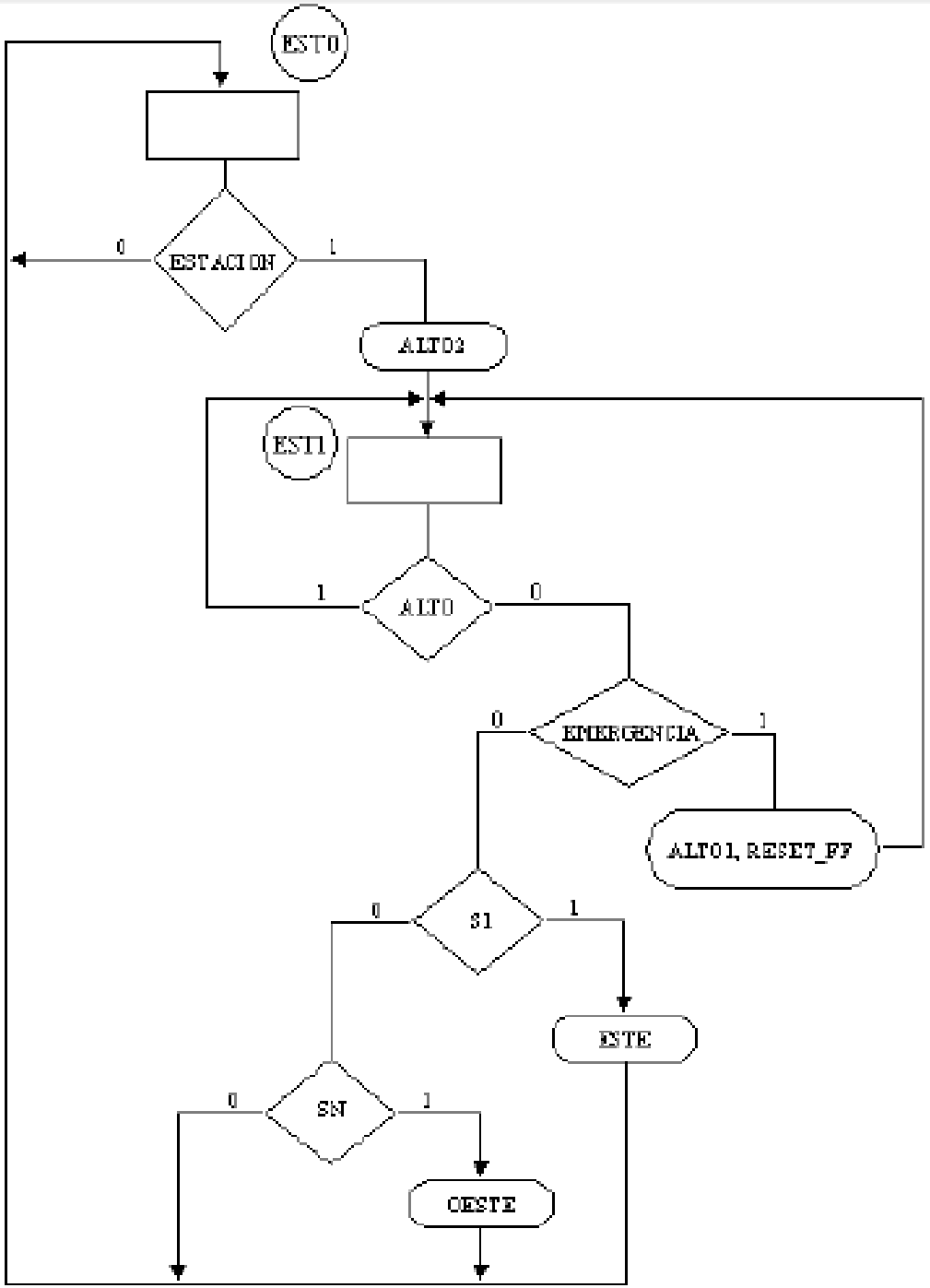


Diagrama de bloques de este sistema:

- En el diagrama podemos observar que los sensores que detectan la presencia del tren en la estacion estan conectados a una compuerta OR. La salida de la compuerta OR, llamada ESTACION, indica si un tren ha entrado en una estacion, sin importar a que estacion entro. Solo interesa saber a que estacion entra si se trata de las estaciones terminales S1 y SN, con el fin de cambiar la dirección de movimiento del tren.
- Ademas, se tiene un modulo que genera una senal de salida de 2 minutos cuando la senal ALTO2 es activada. De manera similar, se tiene otro modulo que genera una senal de 1 minuto cuando ocurre una emergencia, es decir, se activa la senal ALTO1. Ambas senales de espera están conectadas a una compuerta OR. La salida de esta compuerta, llamada ALTO, se encarga de detener al tren durante el tiempo necesario. La salida ALTO tambien se retroalimenta a la maquina de estados para saber si el tren continua parado.

- Tambien se tienen dos flip-flops, uno indica la direccion de movimiento del tren, y el otro la activacion de la senal de emergencia. El flip-flop de emergencias es puesto a uno cuando se oprime el boton de emergencia en el tren, por ello, se conecta la linea del boton de emergencia en el SET del flip-flop. La salida de este flip-flop, denominada EMERGENCIA, entra a la maquina de estados.
- El flip-flop de emergencias debe ser puesto a cero nuevamente para permitir otra emergencia, esto se hace por medio de la linea RESET-FF.
- Por otra parte, el flip-flop de direcciones le indica al tren la direccion a seguir. Si la salida del flip-flop es igual a cero el tren ira hacia el este, si es uno ira hacia el oeste. El tren estara en movimiento todo el tiempo a menos que la senal de ALTO este activada.

Carta ASM


ESTO
0
ESTACION
1
ALTO2
ES11
1
ALTO
0
0
EMERGENCIA
1
ALTO1, RESET_FF
0
S1
1
ESTE
0
SN
1
OESTE

Consideraciones

- En el estado EST0 se revisa el valor de la variable ESTACION, si esta es igual a cero indica que el tren no ha entrado a la estacion, es decir, continua avanzando. Si ESTACION vale uno, entonces se activa la senal ALTO2, la cual genera la senal de 2 minutos que detiene al tren.
- En el estado EST1 se revisa la variable ALTO, si esta vale uno, el algoritmo permanece en ese estado hasta que ALTO valga cero. Cuando ALTO vale cero se revisa la senal de EMERGENCIA.
- Si el valor de EMERGENCIA es uno, se debe detener el tren por un minuto adicional. Para ello se activa la senal ALTO1 y se limpia el flip-flop de emergencias utilizando la senal de salida RESET_FF, de esta manera nuevas peticiones de emergencia seran permitidas.
- Si no hay emergencias, se revisa en que estacion se encuentra el tren para saber si se ha alcanzado una de las estaciones terminales. Si S1 vale uno entonces el tren debe ir al este, por lo tanto, se activa la senal ESTE que pone en cero al flip-flop de direccion. Si SN es igual a uno, entonces se activa la senal OESTE para colocar en uno al flip-flop. Si no se ha alcanzado alguna de las estaciones terminales, el tren continuara avanzando siguiendo la direccion indicada por el flip-flop de direcciones.

Ejercicio

Diseñe un sistema digital que reconozca la siguiente secuencia binaria.

0 1 0 1 1 1 0 1 1

1er ciclo 1

2do ciclo 1

3er ciclo 0

4to ciclo 1

5to ciclo 1

6to ciclo 1

7mo ciclo 0

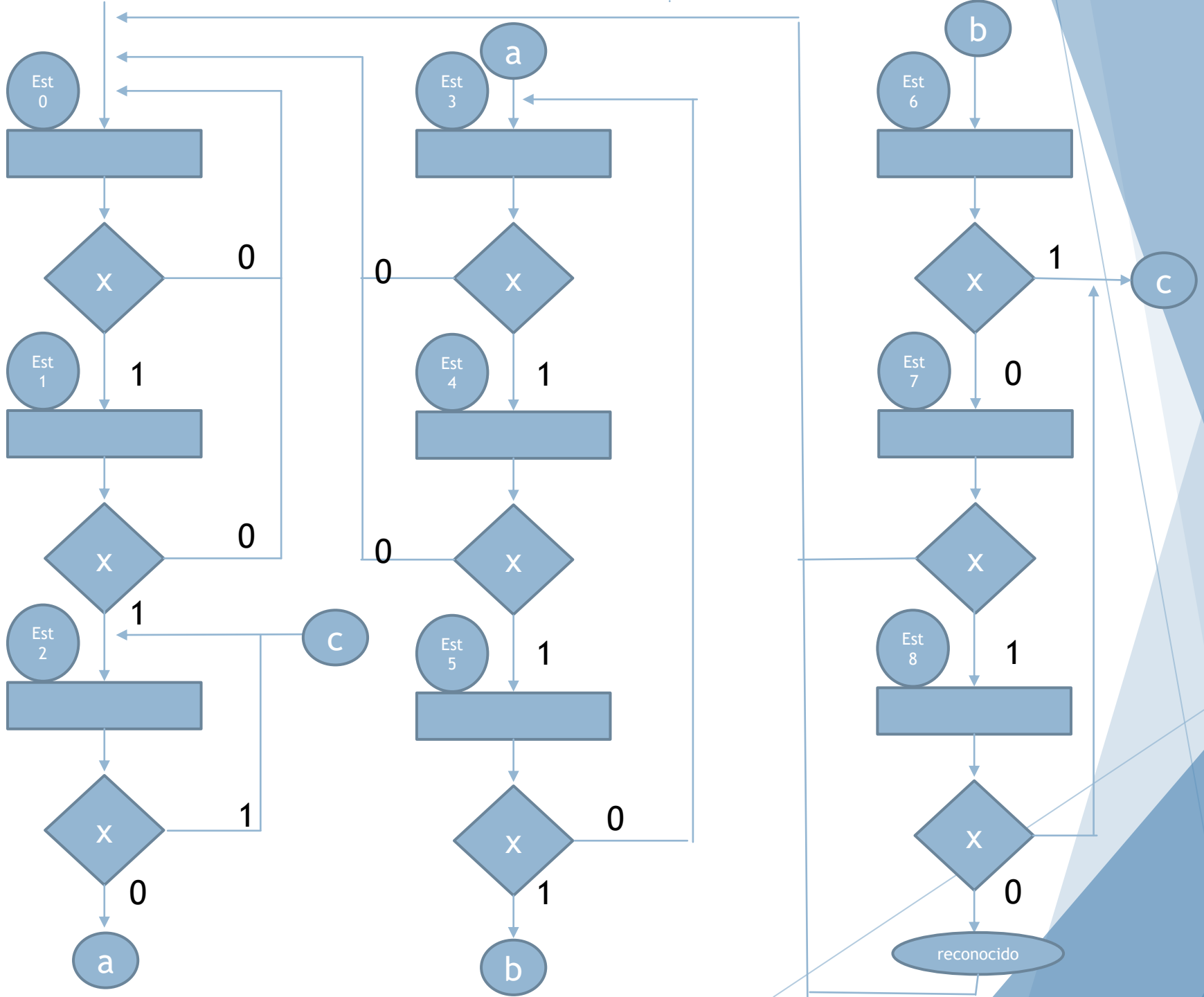
8vo ciclo 1

9no ciclo 0

Cuando haya llegado una secuencia igual, la señal de salida de reconocimiento deberá activarse. Tenga en cuenta que el primer bit de la secuencia es el que se localiza a la derecha, mientras que el último bit de la secuencia es el que se localiza a la izquierda.

Diagrama de bloques

1er ciclo 1
2do ciclo 1
3er ciclo 0
4to ciclo 1
5to ciclo 1
6to ciclo 1
7mo ciclo 0
8vo ciclo 1
9no ciclo 0



PROCEDIMIENTO PARA EL DISEÑO DE CIRCUITOS SECUENCIALES

- El método tradicional que se utilizará plantea los siguientes pasos para el diseño de un circuito secuencial:
- 1. Construir un Diagrama de Estados. Un diagrama de estados muestra la progresión de estados por los que el diseño avanza cuando se aplica una señal de reloj. Recuerde que en lugar de un diagrama de estados podemos usar a la carta ASM.
- 2. A partir del diagrama de estados o la carta ASM, desarrollar una tabla con las transiciones entre estados y las salidas para cada estado. El número de flip-flops necesarios para representar todos los estados de la red secuencial está dado por:
- Número de Flip-flops = $n = \log_2 S$
- donde S es el número total de estados.

- 3. Transferir los valores para cada entrada D de los flip-flops a un mapa de Karnaugh. A partir de los mapas de Karnaugh podremos determinar las expresiones lógicas para las entradas de los flip-flops.
- 4. Transferir cada salida a un mapa de Karnaugh y obtener sus expresiones lógicas.
- 5. Implantación del circuito secuencial. Las expresiones lógicas obtenidas nos permitirán construir el diagrama lógico del circuito.

EJEMPLO

Para ilustrar este procedimiento se desarrollará un ejemplo: Diseñe un circuito secuencial a partir de la siguiente carta ASM.

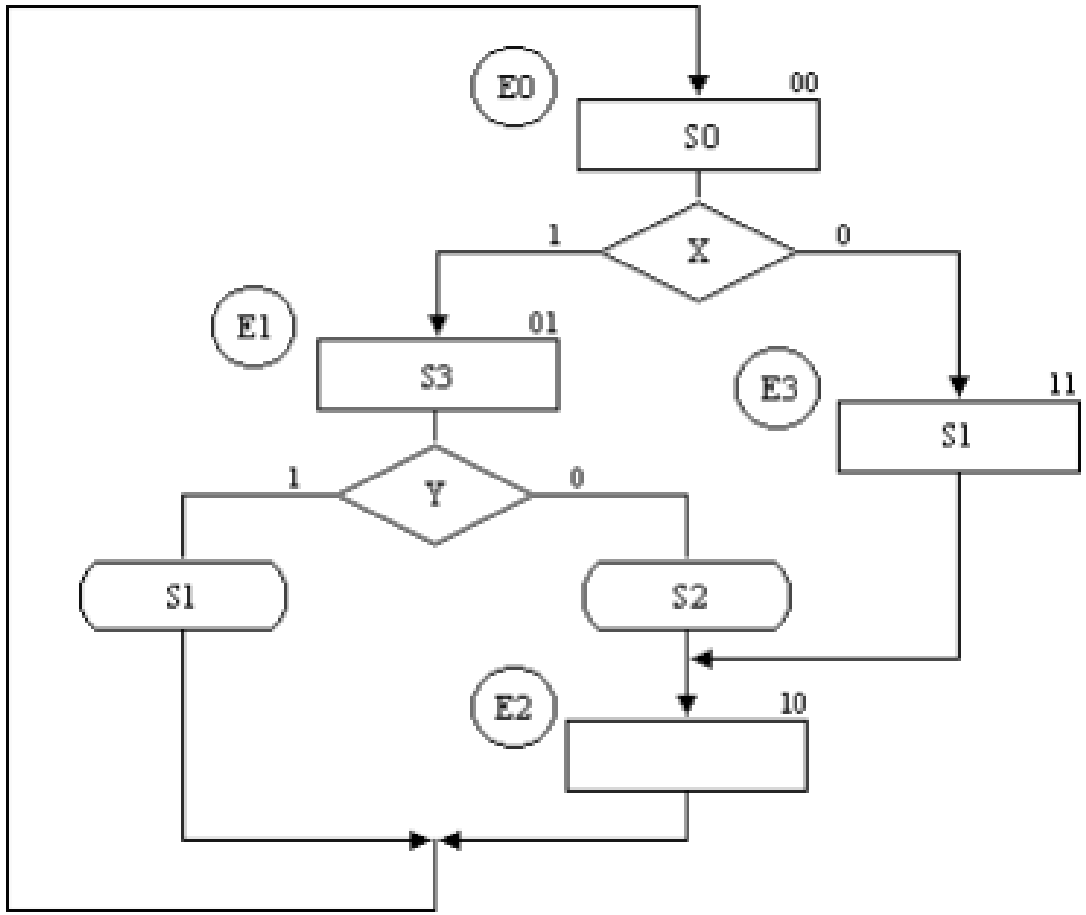


Figura Carta ASM.

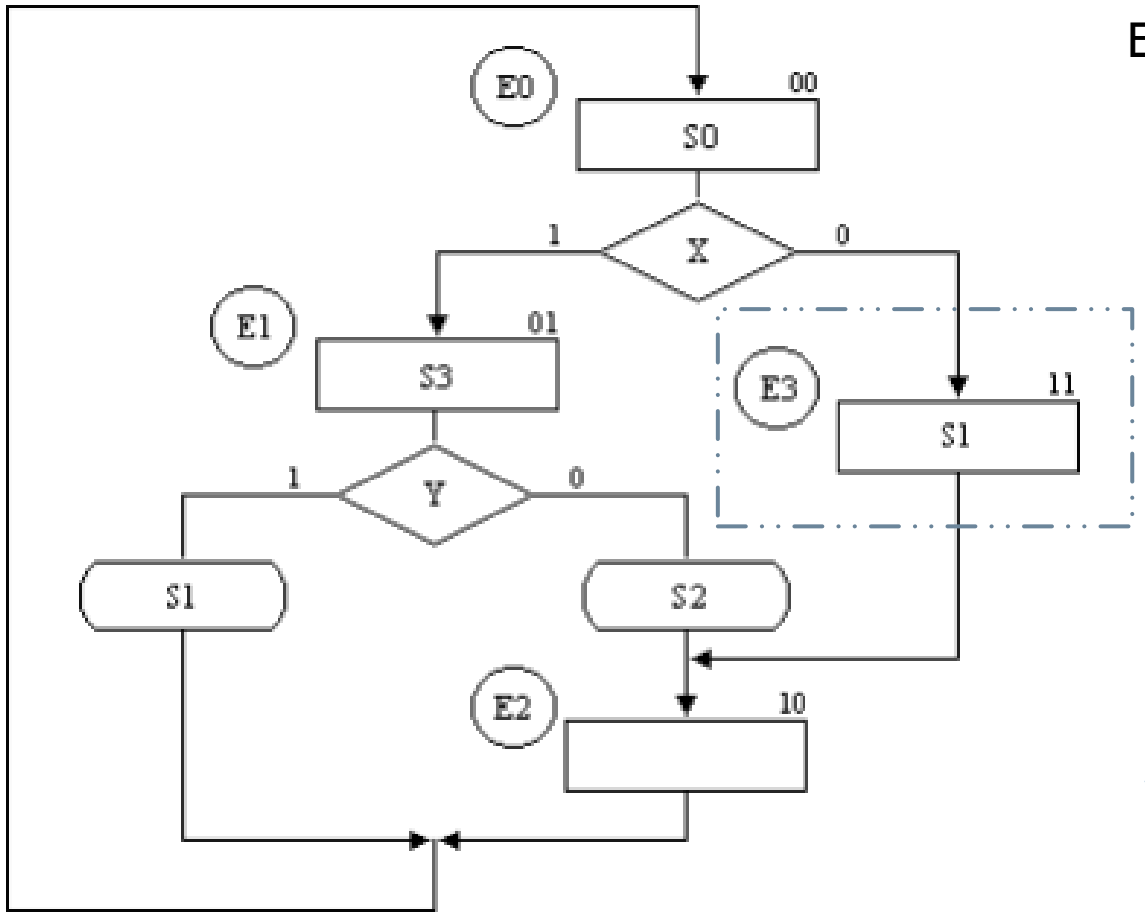
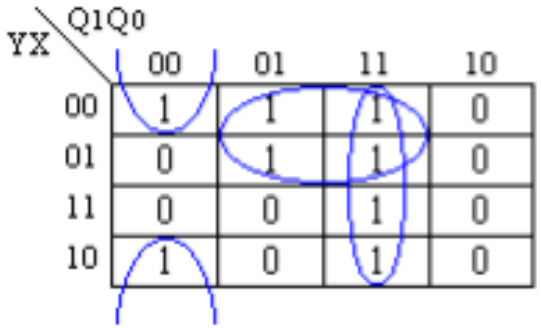


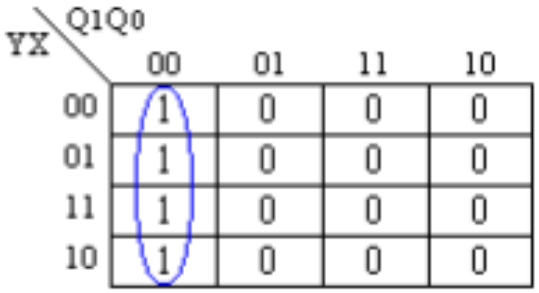
Figura Carta ASM.

E0= 00
E1= 01
E2= 10
E3= 11

Edo presente		Entradas		Edo siguiente		Salidas			
D1	D0	Y	X	Q1	Q0	S3	S2	S1	S0
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	0	1
0	0	1	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1
0	1	0	0	1	0	1	1	0	0
0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	0	1	0
0	1	1	1	0	0	1	0	1	0
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0	1	0
1	1	0	1	1	0	0	0	1	0
1	1	1	0	1	0	0	0	1	0
1	1	1	1	1	0	0	0	1	0

- Se emplearon dos flip-flops tipo D para representar las transiciones de los cuatro estados que componen a la carta ASM. Los mapas de Karnaugh y las expresiones lógicas para las entradas de los flip-flops son los siguientes.





$$D_1 = Q_1^+ = Q_1 Q_0 + Q_0 \overline{Y} + \overline{Q_1}\,\overline{Q_0}\,\overline{X}$$

$$D_0 = Q_0^+ = \overline{Q_1}\,\overline{Q_0}$$

Estado Presente		*Entradas*		*Estado Siguiente*		*Salidas*			
Q_1	Q_0	Y	X	Q_1^+	Q_0^+	S_3	S_2	S_1	S_0
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	0	1
0	0	1	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1
0	1	0	0	1	0	1	1	0	0
0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	0	1	0
0	1	1	1	0	0	1	0	1	0
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0	1	0
1	1	0	1	1	0	0	0	1	0
1	1	1	0	1	0	0	0	1	0
1	1	1	1	1	0	0	0	1	0

Tabla de transiciones de estados y de salidas.

- El mapa de Karnaugh para la entrada D1 se construye con los valores del estado Q1 + , mientras que el mapa para la entrada D0 se construye con los valores del estado Q0 + .

1	**1**	**1**	**0**
0	1	1	0
0	0	1	0
1	0	1	0

- De manera similar, se obtienen las expresiones lógicas para las señales de salida.

YX \ Q1Q0	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	1	0	0
10	0	1	0	0

$$S_3 = \overline{Q_1}\, Q_0$$

YX \ Q1Q0	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	0	0	0
10	0	0	0	0

$$S_2 = \overline{Q_1}\, Q_0\, \overline{Y}$$

YX \ Q1Q0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	1	1	0
10	0	1	1	0

$$S_1 = Q_1\, Q_0 + Q_0\, Y$$

YX \ Q1Q0	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	0	0	0
10	1	0	0	0

$$S_0 = \overline{Q_1}\, \overline{Q_0}$$

Estado Presente		*Entradas*		*Estado Siguiente*		*Salidas*			
Q_1	Q_0	Y	X	Q_1^+	Q_0^+	S_3	S_2	S_1	S_0
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	0	1
0	0	1	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1
0	1	0	0	1	0	1	1	0	0
0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	0	1	0
0	1	1	1	0	0	1	0	1	0
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0	1	0
1	1	0	1	1	0	0	0	1	0
1	1	1	0	1	0	0	0	1	0
1	1	1	1	1	0	0	0	1	0

Tabla de transiciones de estados y de salidas.

- Finalmente, el diagrama lógico de esta máquina de estados es,

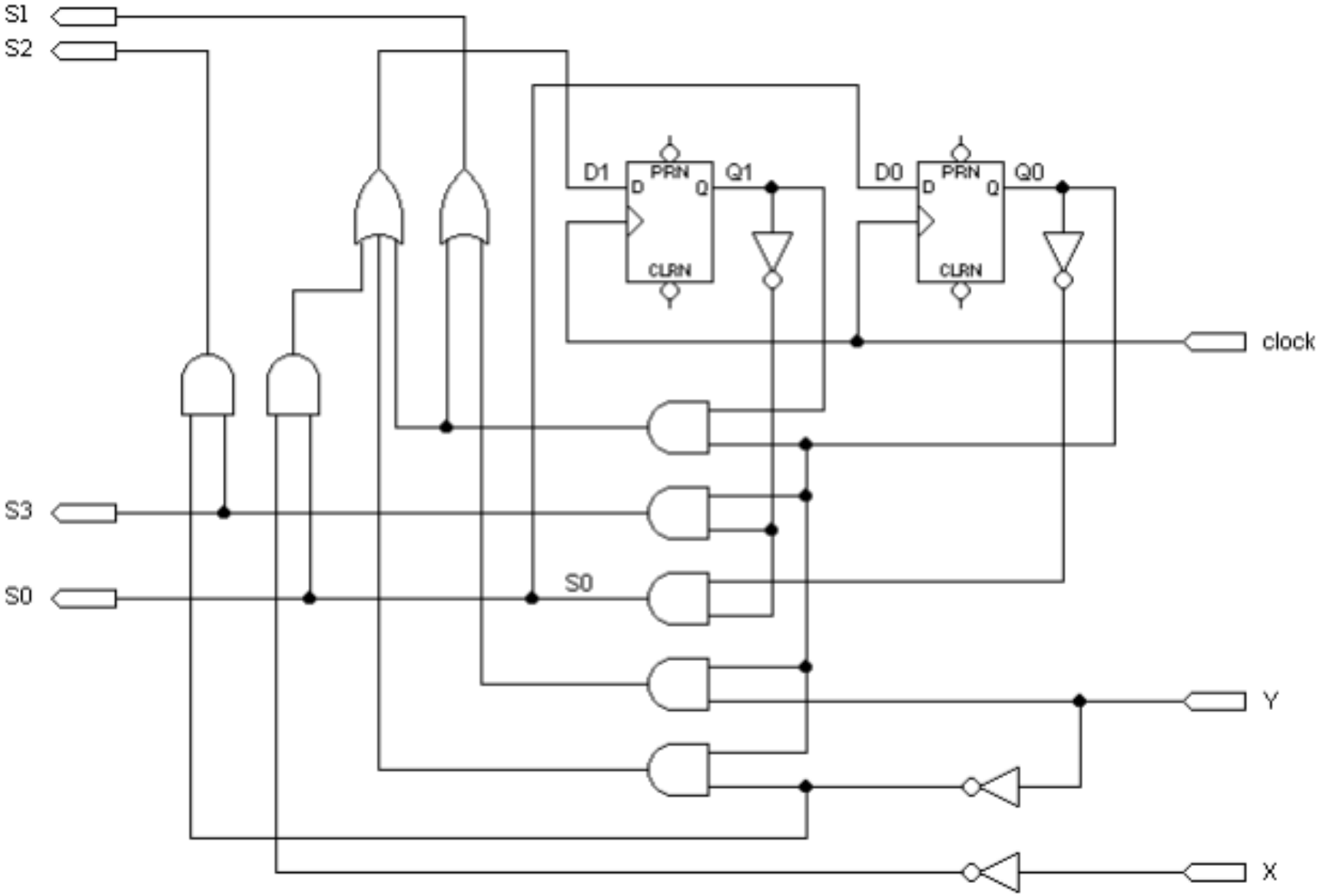


Figura Diagrama lógico.

Diseño de un semáforo

- Requisitos de la lógica secuencial
- La lógica secuencial controla el secuenciamiento de las luces de los semáforos basándose en las entradas procedentes de los circuitos de temporización y del sensor de vehículos. La lógica secuencial generará una secuencia de código Gray de 2 bits para los cuatro estados del sistema indicados en la Figura

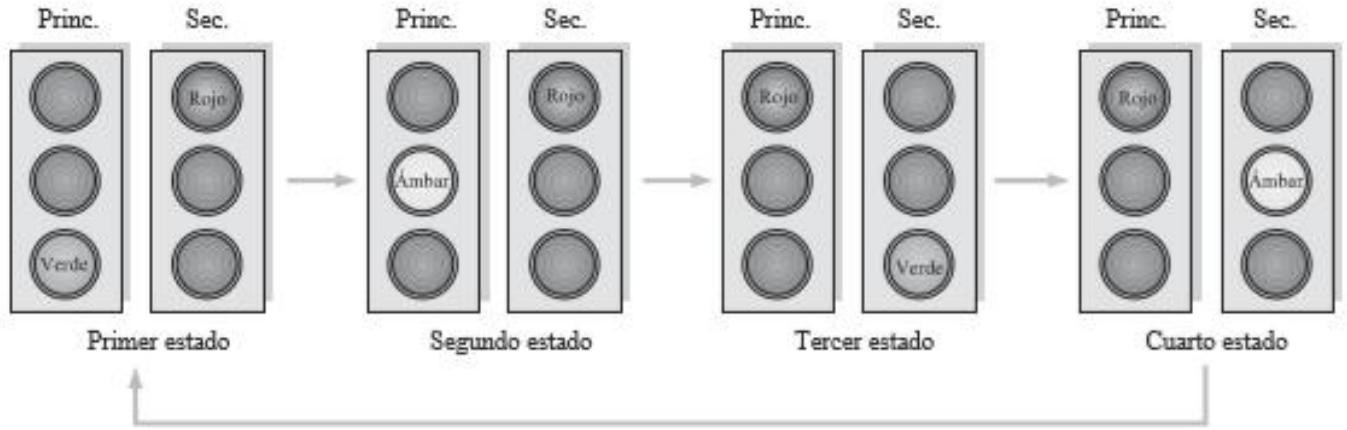
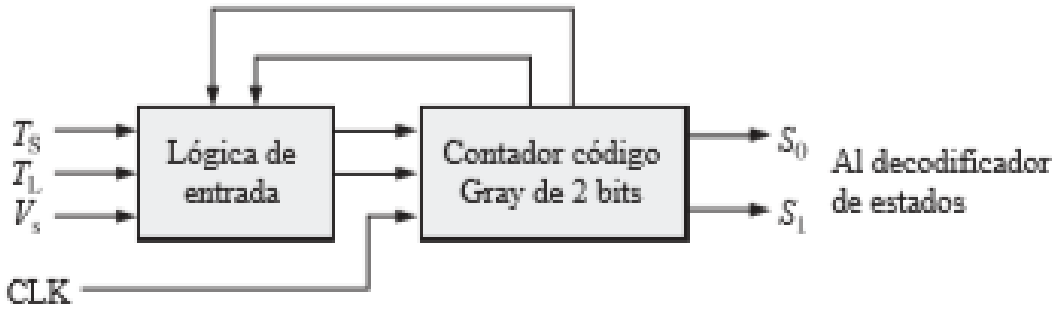


Diagrama de bloques

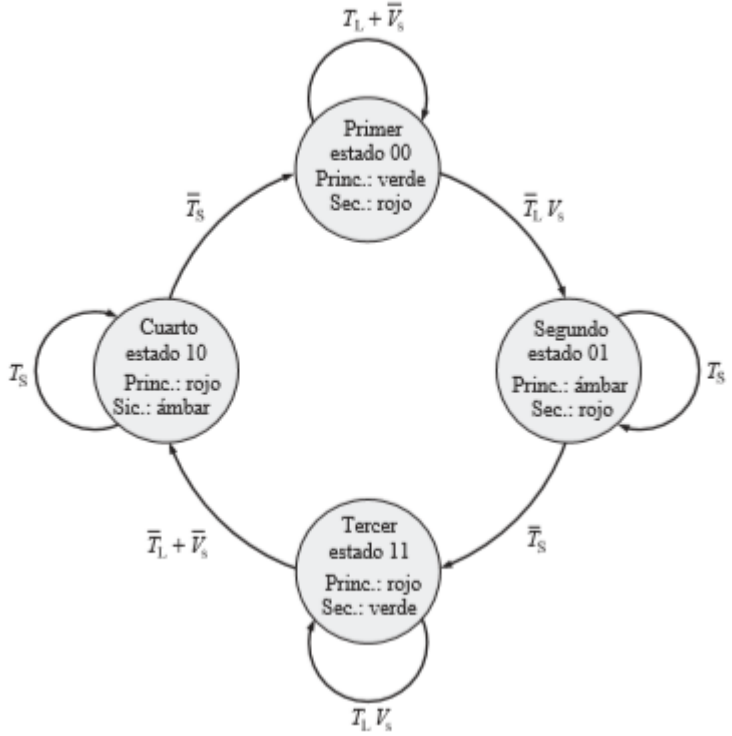
- La lógica secuencial consiste en un contador en código Gray de 2 bits y la lógica de entrada asociada, como se muestra en la Figura. El contador genera una secuencia de cuatro estados. Las transiciones desde un estado al siguiente están determinadas por el temporizador de 4 s, por el temporizador de 25 s y por la entrada del sensor de vehículos. El reloj del contador es la señal de 10kHz producida por el oscilador de los circuitos de temporización.



T_S : Temporizador corto (4 s)
T_L : Temporizador largo (25 s)
V_s : Sensor de vehículos de la calle secundaria

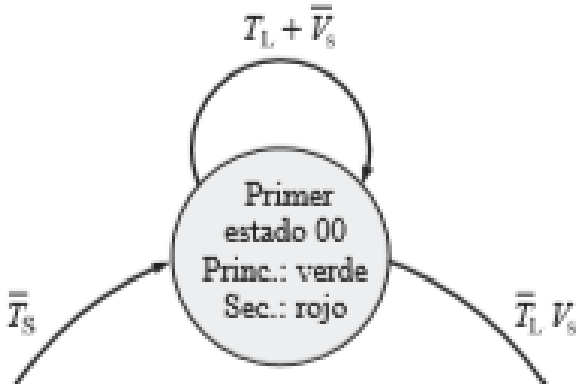
Diagrama de estados

- El diagrama de estados se muestra en la Figura. En función de este diagrama de estados, se describe a continuación el funcionamiento de la lógica secuencial.



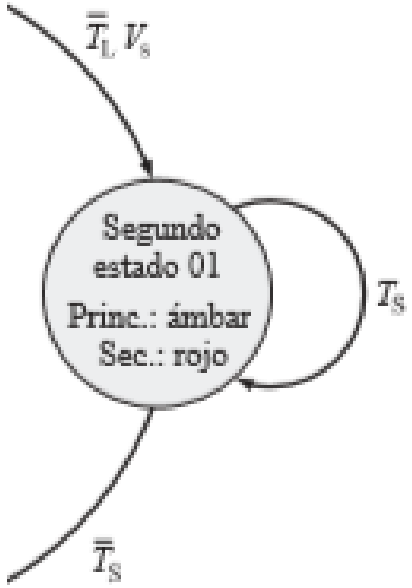
Primer estado

- El código Gray para este estado es 00. La luz del semáforo de la calle principal está en verde y la de la calle secundaria está en rojo. El sistema permanece en este estado al menos durante 25 s cuando el temporizador largo está activado mientras que no haya vehículos en la calle secundaria. Esto se expresa como El sistema pasa al siguiente estado cuando el temporizador largo está desactivado y hay un vehículo en la calle secundaria. Esto se expresa como (T1 Vs).



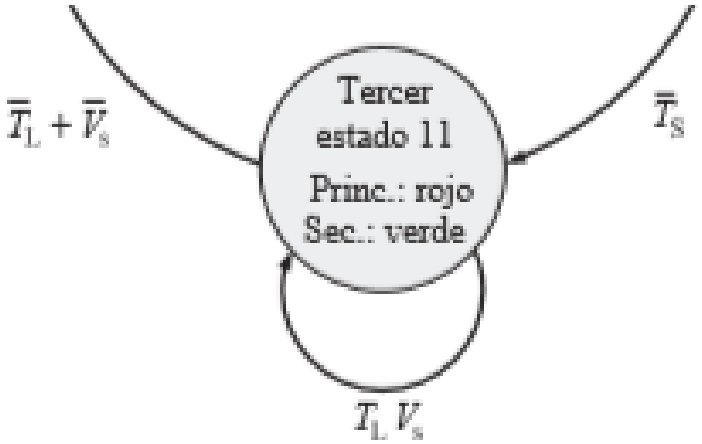
Segundo estado

- El código Gray para este estado es 01. La luz del semáforo de la calle principal está en ámbar y la de la calle secundaria está en rojo. El sistema permanece en este estado durante 4 s cuando el temporizador corto está activado (TS) y pasa al siguiente estado cuando este mismo temporizador se desactiva (Ts).



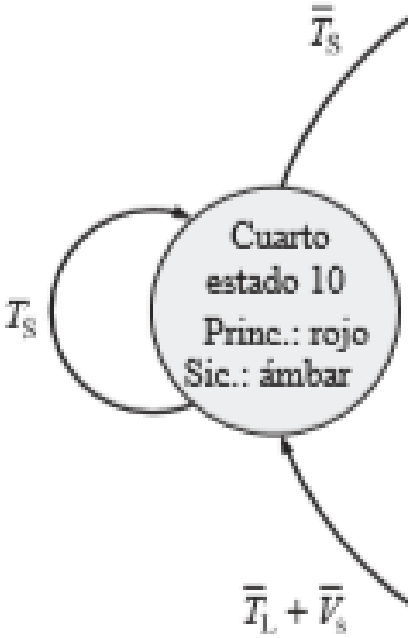
Tercer estado

- El código Gray para este estado es 11. La luz del semáforo de la calle principal está en rojo y la de la calle secundaria está en verde. El sistema permanece en este estado cuando el temporizador largo está activado y hay un vehículo en la calle secundaria. Esto se expresa como TLVS. El sistema pasa al estado siguiente cuando se desactiva el temporizador largo o cuando no hay vehículos en la calle secundaria, lo que se indica como TL+VS



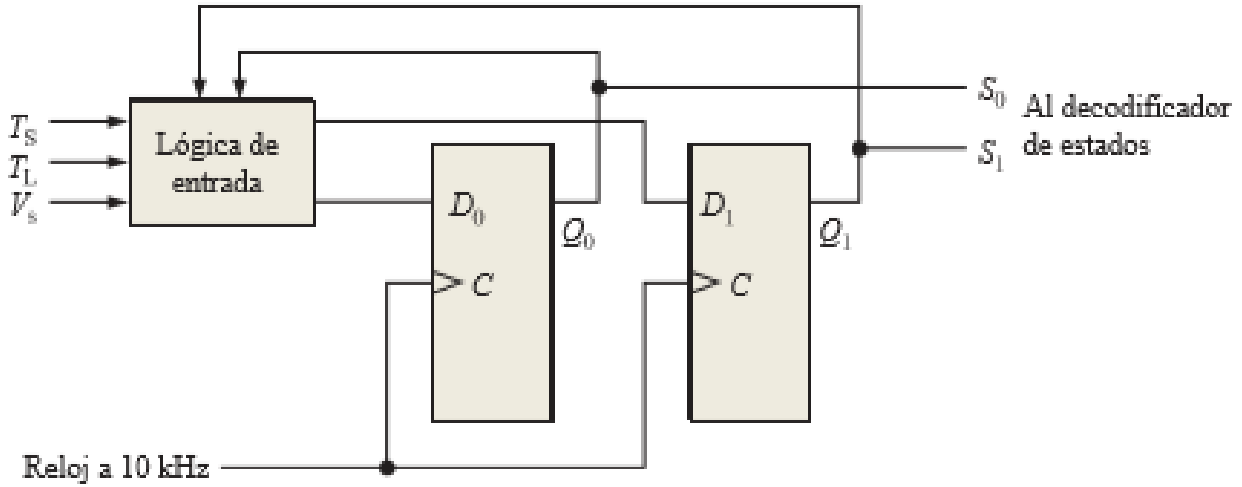
Cuarto estado

- El código Gray para este estado es 10. La luz del semáforo de la calle principal está en rojo y la de la calle secundaria está en ámbar. El sistema permanece en este estado durante 4 s cuando el temporizador corto está activado(TS) y vuelve al primer estado cuando el temporizador corto se desactiva (TS).



Implementación de la lógica secuencial

- El diagrama de la Figura muestra que se emplean dos flip-flops D para implementar el contador Gray. Las salidas de la lógica de entrada proporcionan las entradas D a los biestables y el contador se sincroniza mediante el reloj de 10 kHz oscilador. La lógica de entrada tiene cinco variables de entrada: Q0, Q1, TL, TS y VS.



- En la Tabla 1 se muestra la tabla de transiciones del flip-flop D. A partir del diagrama de estados, puede desarrollarse la tabla del estado siguiente, como se muestra en la Tabla 2. Las condiciones de entrada para TL, TS y VS para cada combinación de estado actual/estado siguiente se enumeran en la tabla.

Transiciones de salida Q_N		Q_{N+1}	Entradas del flip-flop D
0	→	0	0
0	→	1	1
1	→	0	0
1	→	1	1

Tabla 1

Estado actual Q_1	Q_0	Estado siguiente Q_1	Q_0	Condiciones de entrada	Entradas FF D_1	D_0
0	0	0	0	$T_L + \overline{V}_s$	0	0
0	0	0	1	$\overline{T}_L V_s$	0	1
0	1	0	1	T_s	0	1
0	1	1	1	$\overline{T}_s$	1	1
1	1	1	1	$T_L V_s$	1	1
1	1	1	0	$\overline{T}_L + \overline{V}_s$	1	0
1	0	1	0	T_s	1	0
1	0	0	0	$\overline{T}_s$	0	0

Tabla 2