

Los circuitos lógicos secuenciales

Montse Peiron Guàrdia
Fermín Sánchez Carracedo

PID_00215619

Índice

Introducción	5
Objetivos	6
1. Caracterización de los circuitos lógicos secuenciales	7
1.1. Necesidad de memoria en los circuitos lógicos	7
1.2. Reloj. Sincronización	7
2. El biestable D	10
2.1. Dispositivo elemental de memoria. El biestable D	10
2.2. Señal de carga	13
2.3. Entradas asíncronas	14
3. Bloques secuenciales	18
3.1. Registro	18
3.2. Banco de registros	23
3.3. Memoria RAM	25
4. El modelo de Moore	29
4.1. Estado. Transiciones	29
4.2. Representación gráfica: grafos de estado	34
4.2.1. Mecánica de diseño	35
4.2.2. Notación	37
4.2.3. Circuitos sin entradas	38
4.3. Sincronización	41
4.4. Implementación	45
Resumen	50
Ejercicios de autoevaluación	51
Solucionario	56
Bibliografía	75

Introducción

En el módulo “Los circuitos lógicos combinacionales” se ha visto que los circuitos computan funciones lógicas de las señales de entrada: el valor de las señales de salida en un instante determinado depende del valor de las señales de entrada en ese mismo momento. Cuando las señales de entrada varían, entonces, como consecuencia, también variarán las de salida (después del retraso producido por las puertas y bloques, que en este curso no tenemos en cuenta).

Ahora bien, en algunas aplicaciones es preciso que el valor de las señales de salida no dependa sólo del valor de las entradas en el mismo momento, sino que también tenga en cuenta los valores que las entradas han tomado anteriormente. En los circuitos que hemos conocido hasta ahora, esto no es posible: son necesarios elementos que tengan alguna “capacidad de recordar”, que son los que conformen los circuitos lógicos secuenciales.

En este módulo conoceremos el concepto de *sincronización* y se estudiarán los *biestables*, que son los dispositivos secuenciales más básicos, y los bloques secuenciales, que se construyen a partir de los biestables y tienen una funcionalidad determinada.

Después se presentará una de las maneras de formalizar el funcionamiento de un circuito temporal, el llamado *modelo de Moore*. El modelo de Moore utiliza los conceptos de *estado* y de *transiciones entre estados* para describir la evolución temporal del funcionamiento de un circuito temporal, que se representa gráficamente mediante un *grafo de estados*.

Este módulo permite describir el comportamiento de muchos circuitos secuenciales.

Objetivos

El objetivo fundamental de este módulo es conocer los circuitos lógicos secuenciales, es decir, saber cómo están formados y utilizarlos con agilidad. Para llegar a este punto se tendrán que haber satisfecho los siguientes objetivos:


1. Saber discernir, a partir de la funcionalidad que se quiere que tenga un circuito lógico, si el circuito tiene que ser de tipo secuencial o combinacional.
2. Entender el concepto de memoria, la necesidad de una sincronización en los circuitos lógicos secuenciales y el funcionamiento de la señal de reloj.
3. Conocer el funcionamiento del biestable D y de todas las entradas de control que puede tener.
4. Conocer la funcionalidad de los diferentes bloques secuenciales, y saberlos utilizar en el diseño de circuitos.
5. Comprender los conceptos de *estado* y de *transición entre estados*. Entender todos los elementos de un grafo de estados, y ser capaces de construir el grafo de estados de un circuito cualquiera a partir de la descripción de su funcionalidad.
6. Saber deducir la evolución temporal de un circuito a partir del grafo de estados que describe el funcionamiento.

1. Caracterización de los circuitos lógicos secuenciales

1.1. Necesidad de memoria en los circuitos lógicos

Sea un circuito con una señal de entrada X y una de salida Z , ambos de n bits, que interpretamos como números representados en complemento a dos. Supongamos que queremos que $Z = X + 2$. Con los elementos estudiados en el módulo “Los circuitos lógicos combinacionales” sabemos cómo se tiene que hacer, incluso de muchas formas diferentes. Cuando el valor en la entrada X varíe, entonces Z también cambiará consecuentemente de valor.

Supongamos que queremos que el valor Z corresponda a la suma de todos los valores que han estado presentes en la entrada X durante un intervalo de tiempo determinado (durante el cual el valor de X ha variado). Con los dispositivos lógicos que conocemos hasta ahora no podemos conseguirlo, porque cuando cambiamos el valor de X , el valor anterior ha “desaparecido”, por lo que ya no podemos utilizarlo para calcular la suma.

Es preciso que este circuito sea capaz de recordar o retener los valores anteriores de algunas señales, es decir, debe tener memoria. Ésta es la funcionalidad que distingue los circuitos lógicos secuenciales de los combinacionales. 

Secuencial

La denominación “secuencial” deriva justamente de la capacidad de recordar la secuencia de valores que toman las señales.

1.2. Reloj. Sincronización

En los circuitos combinacionales, la única noción temporal que interviene es el presente. En cambio, en los circuitos secuenciales se tiene en cuenta la evolución temporal de las señales (y aparece, como se verá más adelante, la noción de futuro).

Ahora bien, en la descripción del circuito del ejemplo anterior, ¿qué quiere decir con exactitud que “todos los valores que han estado presentes en la entrada X durante un intervalo de tiempo determinado”? La señal X puede ir cambiando de valor de forma aleatoria en el tiempo: puede valer 13 durante cuatro nanosegundos, después 25 durante diez nanosegundos, después 0 durante un nanosegundo, etc. ¿Cómo puede determinar el circuito en qué momento “ X ha dejado de tener el valor antiguo” y “empieza a tener el valor nuevo”? Para poder determinarlo, el circuito debe tener un **mecanismo de sincronización**. En los circuitos secuenciales que estudiaremos en este módulo se utiliza una señal de reloj como forma de sincronización.

El **reloj** es una señal que sirve para determinar los instantes en que un circuito secuencial “ve” el valor de las señales, o “es sensible”, y responde en consecuencia.

Discretización

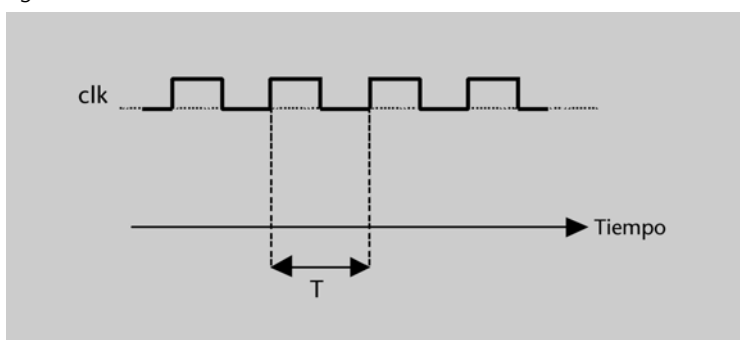
El reloj discretiza el tiempo: en lugar de verlo como una dimensión continua, los circuitos lo ven como una secuencia de instantes.

Esta tarea que lleva a cabo la señal de reloj se llama **sincronización de los circuitos**.

Concretamente, la señal del reloj toma los valores 0 y 1 de forma cíclica y continua desde la puesta en marcha de un circuito hasta que éste se detiene. De forma habitual, se utiliza la noción *clk* para hacer referencia a la señal del reloj (deriva del inglés *clock*).

La figura 1 muestra el cronograma de la señal de reloj. El ciclo que forma la secuencia de valores 0 y 1 tiene una duración determinada y constante, T , que se llama **periodo**. Se puede medir en segundos o, más habitualmente, en nanosegundos (mil millonésimas de segundo).

Figura 1



Los instantes en que la señal de reloj pasa de 0 a 1 se llaman **flancos ascendentes**. El intervalo que hay entre un flanco ascendente y el siguiente se llama **ciclo** o **ciclo de reloj**. Por tanto, la duración de un ciclo es un periodo de T segundos.

La **frecuencia** del reloj es la inversa del periodo, es decir, es el número de ciclos de reloj que ocurren durante un segundo. Se mide en hercios (ciclos por segundo); lo más habitual es usar el múltiplo gigahercios (mil millones de ciclos por segundo), que se abrevia GHz. Por ejemplo, si tenemos un reloj con un periodo de 0,75 nanosegundos, su frecuencia es la siguiente:

$$1,33 \cdot 10^9 \text{ ciclos/segundo} = 1,33 \text{ GHz.}$$

La señal de reloj puede sincronizar los circuitos de varias formas. En este curso sólo se verá la que se usa de forma más habitual, llamada **sincronización por flanco ascendente**. Esta forma de sincronización establece que los dispositivos secuenciales de un circuito serán sensibles a los valores de la señal en los instantes de los flancos ascendentes, tal como veremos en el siguiente apartado. En el resto de los apuntes, si escribimos sólo *flanco* nos referimos a *flanco ascendente*. !

Tal como se ha visto en el módulo "Los circuitos lógicos combinacionales", las transiciones entre los valores 0 y 1 de una señal tienen un cierto retraso. Sin embargo, nosotros consideramos que los cambios de valor, de la señal de reloj o de cualquier otro son instantáneos.

Generación de una señal de reloj

Físicamente, la señal de reloj se genera a partir de cristales de cuarzo, un mineral que tiene la propiedad llamada *piezoelectricidad*: cuando recibe corriente eléctrica vibra con una frecuencia extremadamente grande y regular.

Valores típicos de las frecuencias

A principios del año 2010, los procesadores comerciales más rápidos tienen frecuencias de reloj de 3,5 GHz, con periodos cercanos a 0,3 nanosegundos. Sin embargo, la frecuencia base se puede multiplicar utilizando la técnica denominada *overclocking*.

Otras formas de sincronización

Otras formas de sincronización son por nivel 0, por nivel 1 y por flanco descendente.

Actividades

1. Se quiere diseñar un sistema que reconozca si se produce la combinación 1010 en una entrada de cuatro bits. Indicad si el sistema es secuencial o combinacional y por qué.
2. Se quiere diseñar un sistema que reconozca una secuencia de cuatro dígitos decimales para identificar el número secreto de una tarjeta de crédito. El sistema tiene una entrada de datos única de cuatro bits, que codifican cada dígito. Indicad si el sistema es secuencial o combinacional y por qué.
3. ¿Cuál es el periodo del reloj de un procesador Intel Celeron E1200 que funciona a 1,6 GHz?

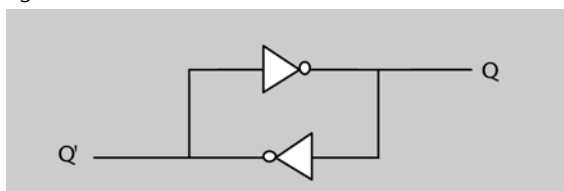
2. El biestable D

2.1. Dispositivo elemental de memoria. El biestable D

En el apartado anterior hemos visto la necesidad de que los circuitos lógicos tengan capacidad de memoria. En este apartado veremos cómo se construyen los dispositivos que pueden “recordar” los valores señalados.

Si examinamos el circuito que muestra la figura 2, vemos que el valor que hay en los puntos Q y Q' (0 ó 1) se mantendrá indefinidamente, ya que la salida de cada inversor está conectada con la entrada del otro. Por tanto, podemos decir que este circuito es capaz de “recordar”, o mantener en el tiempo, un valor lógico.

Figura 2



Ahora bien, este circuito no es muy útil, porque no admite la posibilidad de modificar el valor recordado. Interesa diseñar un circuito que tenga esta misma capacidad de memoria, pero que además permita que el valor en el punto Q pueda cambiar según los requerimientos del usuario. Un circuito con estas características se llama *biestable*.

Los **biestables** son los dispositivos de memoria más elementales: permiten guardar un bit de información.

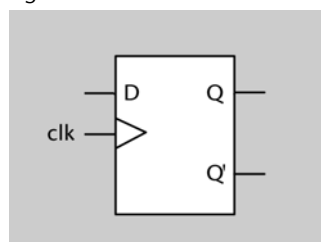
Un biestable tiene dos salidas, Q y Q' . Se dice que Q es “el valor que guarda el biestable” en cada momento, o “el valor almacenado en el biestable”, y Q' es su negación.

La denominación *biestable*

Proviene del hecho de que el biestable puede estar “en dos estados”: $Q = 0$ o $Q = 1$.

Hay diferentes tipos de biestables. En esta asignatura sólo veremos uno, el **biestable D**. La figura 3 muestra su representación gráfica. Podemos observar que tiene una entrada de reloj, ya que se trata de un dispositivo secuencial.

Figura 3



La entrada de reloj de un dispositivo secuencial sincronizado por flanco ascendente se identifica con el símbolo \triangleright .

Implementación interna

La implementación interna de un biestable D se basa en el circuito de la figura 2, pero no es objetivo de esta asignatura conocerla.

El biestable D funciona de la siguiente forma:

La salida Q toma el valor que haya en la entrada D en cada flanco ascendente de reloj. Durante el resto del ciclo, el valor de Q no cambia.

Es decir, el biestable sólo es sensible al valor presente en la entrada D en los instantes de los flancos ascendentes.

La figura 4 muestra la tabla de verdad que describe el comportamiento del biestable D (no ponemos la columna correspondiente a Q' porque es la negación de Q). En esta figura se introducen algunas notaciones que se usarán de ahora en adelante:

- El símbolo \uparrow representa un flanco ascendente de reloj.
- El símbolo $^+$ a la derecha del nombre de una señal se refiere al valor que tomará esta señal cuando se produzca el próximo flanco ascendente de reloj.

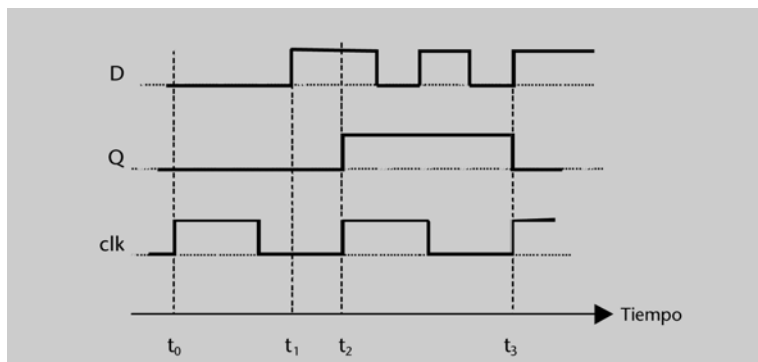
Por tanto, Q^+ no identifica ninguna señal del circuito, sino el valor que tendrá la señal Q en un instante futuro: a partir del momento en que se produce el próximo flanco. Esta notación, pues, nos permite describir con precisión la evolución temporal de las señales en un circuito lógico secuencial.

Figura 4

D	clk	Q^+
0	\uparrow	0
1	\uparrow	1

La figura 5 muestra el cronograma del comportamiento de un biestable D durante dos ciclos, el que va del instante t_0 al instante t_2 y el que va de t_2 a t_3 .

Figura 5



Nota

Notemos que no tendría sentido el hecho de dibujar una línea con el nombre Q^+ en un cronograma, ya que Q^+ no corresponde a ninguna señal.

En la figura 5 podemos observar que D toma diferentes valores durante un ciclo. En el ciclo que va de t_0 a t_2 vale primero 0 y después 1, y en un ciclo que va de t_2 a t_3 toma valores 1, 0, 1 y 0, y pasa de nuevo a 1 en el instante t_3 (coincidiendo con el tercer flanco ascendente del reloj). Nos podríamos preguntar cuál de estos valores es el que se carga en el biestable cuando llega el flanco ascendente, en los instantes t_2 y t_3 .

Para responder a esta pregunta, usaremos la siguiente convención: el valor que se carga en un biestable en el instante de un flanco es el que tiene la entrada D en el momento en que llega el flanco.

Valor que se carga en el biestable

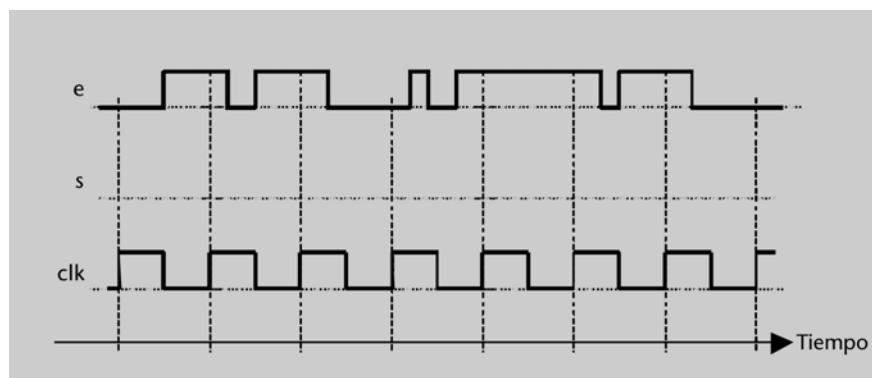
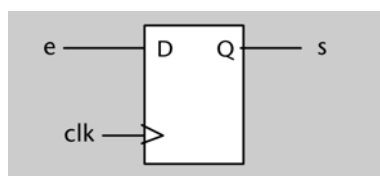
En los circuitos reales, el valor que se carga en el biestable es el que está presente de forma estable en la entrada D un cierto intervalo de tiempo anterior al flanco, que depende del retraso en la subida y bajada de tensión de las señales y de la tecnología utilizada para construir el biestable.

En un cronograma, esta idea se traduce en lo siguiente: el valor que se carga en el biestable en un flanco determinado es el que tiene la línea correspondiente a la entrada D cuando toca la línea vertical correspondiente a este flanco por la izquierda (ya que, en un cronograma, el tiempo transcurre de izquierda a derecha).

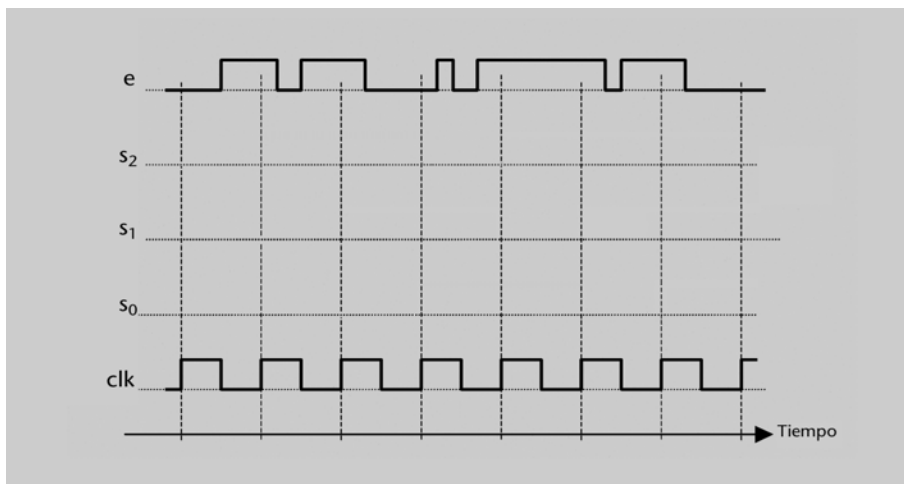
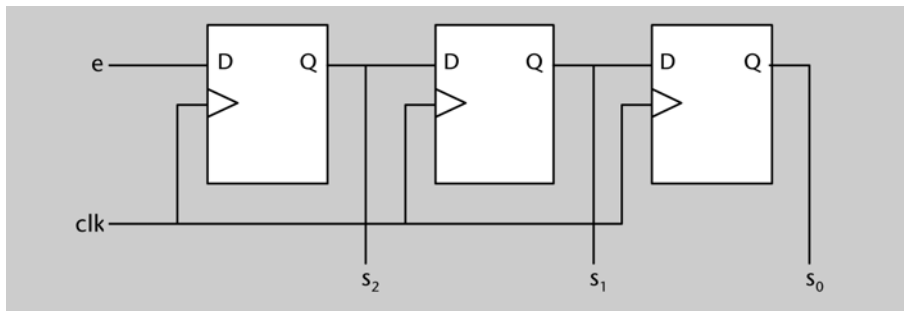
En la figura 5 podemos ver que en el instante t_2 , Q toma el valor 1 (en este caso no hay duda), y en el instante t_3 , Q toma el valor 0. La convención anterior nos dice que el valor que toma Q es 0, porque es el que hay en la línea correspondiente en D cuando ésta toca el flanco por la izquierda.

Actividades

4. Completad el cronograma que corresponde al circuito de la figura.



5. Completad el cronograma que corresponde al circuito de la figura.



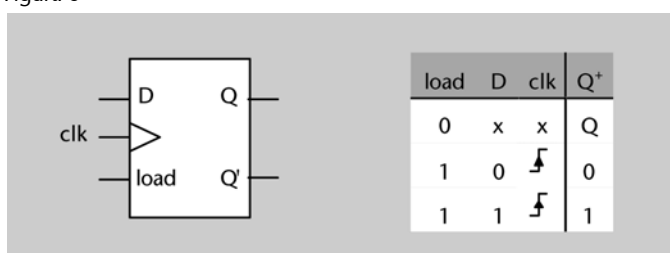
2.2. Señal de carga

Cuando se pone un biestable dentro de un circuito con otros componentes, a veces nos interesará que el contenido del biestable no cambie, ni siquiera en los instantes de los flancos, aunque varíe el valor de D ; queremos que el biestable sea “insensible” a las variaciones de D cuando así lo requiramos.

Con este fin se añade al biestable una **señal de carga** que funciona de la siguiente forma: si vale 0, el valor del biestable no cambia. Si vale 1, el biestable funciona tal como se ha explicado en el apartado anterior.

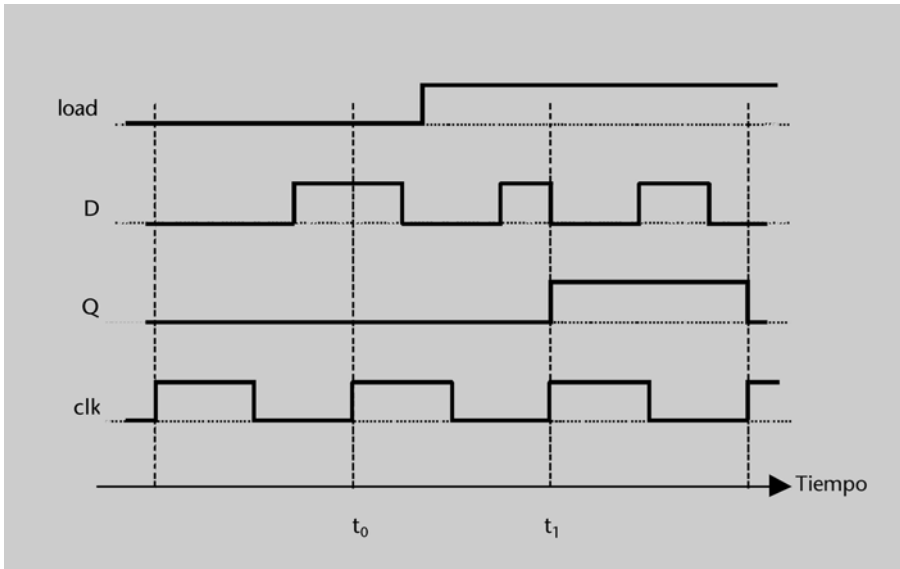
La figura 6 muestra la representación gráfica de un biestable D con señal de carga (se identifica por la palabra *load*), y la tabla de verdad de su funcionamiento. Como es habitual, las x indican valores cualesquiera de las señales.

Figura 6



La figura 7 muestra el cronograma del comportamiento de un biestable D con señal de carga. Podemos ver que en el instante t_0 el valor de Q no varía, aunque $D = 1$, porque la señal *load* está en 0. Cuando *load* = 1, entonces el biestable funciona tal como se ha estudiado en el apartado anterior.

Figura 7



2.3. Entradas asíncronas

El valor de un biestable D puede variar en los instantes de flancos ascendentes según el valor que hay en las entradas D y *load*. Ahora bien, se debe tener la capacidad de darle un valor inicial: el valor que tomará cuando se ponga en marcha un circuito.

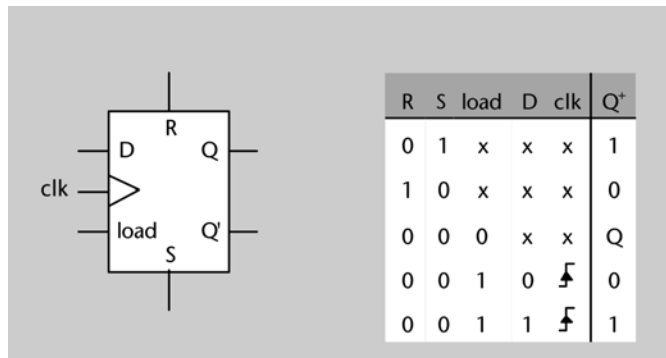
Las **entradas asíncronas** de un biestable permiten modificar su valor de forma instantánea, de forma independiente al valor de la señal de reloj y de las entradas D y *load*. Se dice que las entradas asíncronas tienen más prioridad que el resto de las entradas.

Los biestables suelen tener dos entradas asíncronas:

- R (del inglés *reset*): en el momento en que se pone a 1, el biestable se pone a 0.
- S (del inglés *set*): en el momento en que se pone a 1, el biestable se pone a 1.

La figura 8 muestra la representación gráfica de un biestable D con entradas asíncronas y señal de carga, y la tabla de verdad que describe su comportamiento.

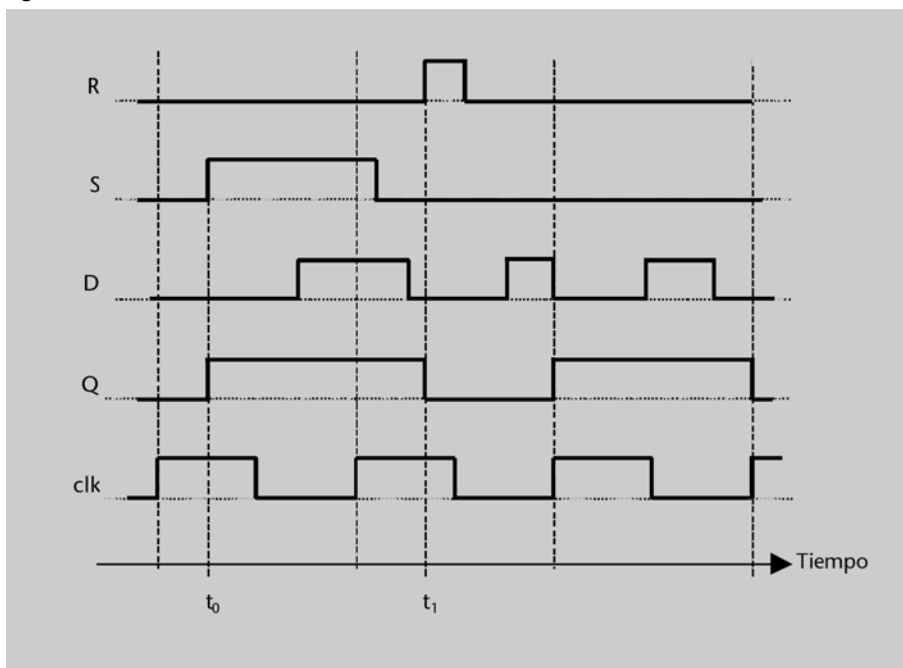
Figura 8



Si tanto R como S valen 1, el comportamiento del biestable no se puede predecir. En los circuitos reales siempre se garantiza que esta situación no se produzca nunca.

La figura 9 muestra el cronograma del comportamiento de un biestable D con entradas asíncronas y señal de carga (para simplificar el dibujo, supongamos que $load = 1$ todo el tiempo). Se puede observar que el biestable se pone a 1 en el instante t_0 aunque éste no coincida con un flanco ascendente, y se mantiene a 1 mientras $S = 1$, de forma independiente al valor de D . De igual forma, se pone a 0 en el instante t_1 . En cambio, mientras ambas entradas asíncronas están a 0, el biestable modifica su valor sólo en los momentos de un flanco ascendente, de acuerdo con el valor de D .

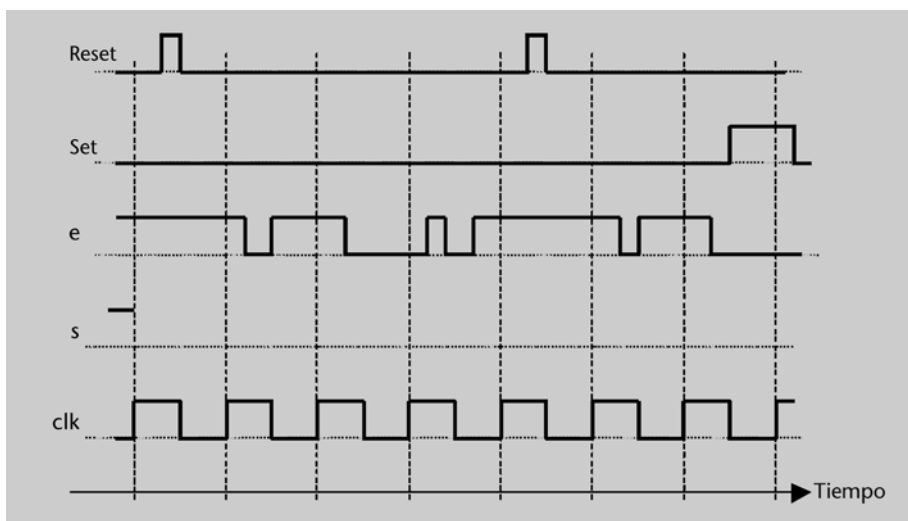
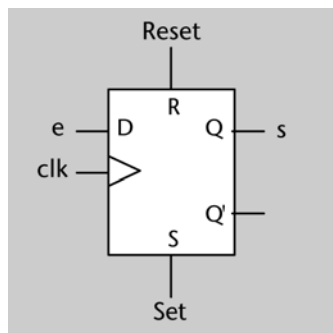
Figura 9



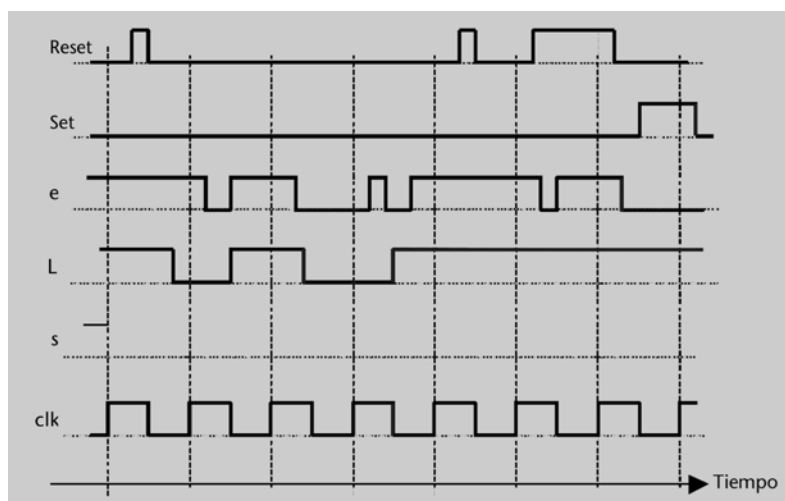
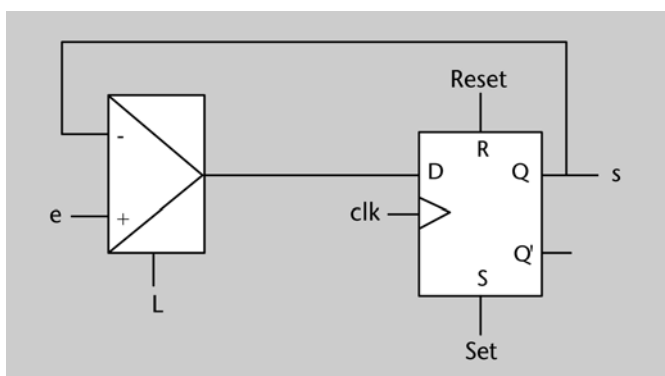
Cuando no dibujamos en un circuito las señales $load$, R y S de un biestable, asumiremos por defecto que valen 1, 0 y 0 respectivamente.

Actividades

6. Completad el cronograma que corresponde al circuito de la figura suponiendo que en su inicio la salida Q vale 1.



7. Completad el cronograma que corresponde al circuito de la figura, suponiendo que inicialmente la salida Q vale 1. ¿Cuál es el papel de la señal L en el circuito?

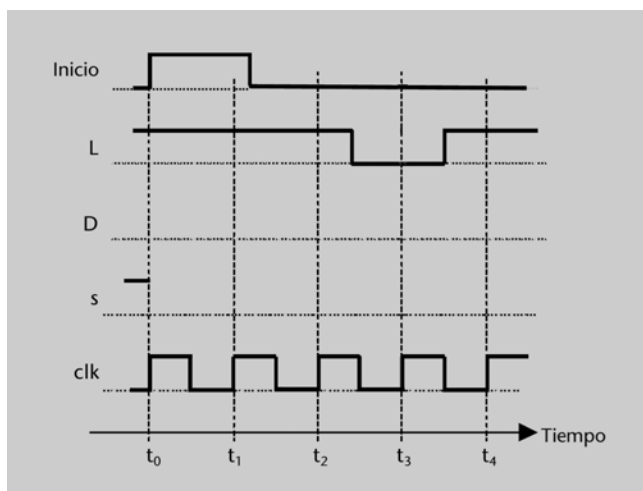
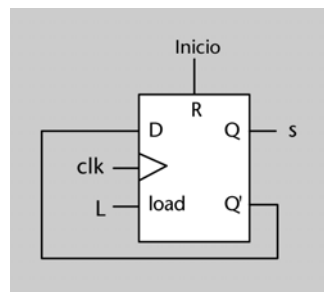


Inicialización de los circuitos

Los circuitos secuenciales suelen tener una señal que actúa de forma asíncrona y tiene por misión inicializar el circuito. Esta señal, que llamaremos *Inicio*, está conectada a las entradas asíncronas de los biestables (en *R* o *S* según si el valor inicial tiene que ser 0 ó 1). Cuando el circuito se pone en funcionamiento, la señal *Inicio* vale 1 durante un cierto intervalo de tiempo, y después baja a 0 (se dice que **genera un pulso a 1**); un pulso siempre dura más de un ciclo de reloj (es decir, siempre se produce al menos un flanco ascendente mientras *Inicio* vale 1). Durante el funcionamiento normal del circuito, *Inicio* permanece a 0. Si en algún otro momento *Inicio* hace otro pulso a 1, el circuito se reinicializa.

Actividades

8. Completad el cronograma que corresponde al circuito de la figura.



3. Bloques secuenciales

3.1. Registro

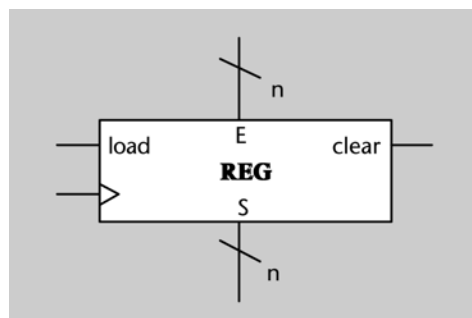
Hemos visto que un biestable permite guardar el valor de un bit. Para guardar el valor de una palabra de n bits serán necesarios n biestables D.

Un **registro** es un bloque secuencial formado por n biestables D, que permite guardar el valor de una palabra de n bits.

La figura 10 muestra la representación gráfica de un registro. Se puede ver que presenta las siguientes señales:

- Una entrada de datos de n bits, E . Cada uno de los bits de este bus está conectado con la entrada D de uno de los n biestables que forman el registro.
- Una salida de datos de n bits, S , que es un bus formado por las salidas Q de los n biestables que forman el registro.
- Dos entradas de control de un bit, *load* y *clear*. Estas dos señales están conectadas respectivamente a la señal *load* y a la entrada asíncrona R de cada uno de los biestables del registro.
- Una entrada de reloj, conectada a las entradas de reloj de todos los biestables.

Figura 10




El funcionamiento del registro es el siguiente: !

- La señal *clear* sirve para poner el contenido del registro a 0. Dado que se conecta con la entrada R de los biestables, es una señal asíncrona, es decir, actúa independientemente del reloj, y es el más prioritario, de forma que cuando está a 1, los n bits del registro se ponen a 0, independientemente del valor de las otras señales.

- Cuando *clear* está a 0, entonces los n biestables que forman el registro se comportan como n biestables D con señal de carga.

Este funcionamiento se puede expresar mediante la siguiente tabla de verdad:

clear	load	clk	S^+
1	x	x	0
0	0	x	S
0	1		E

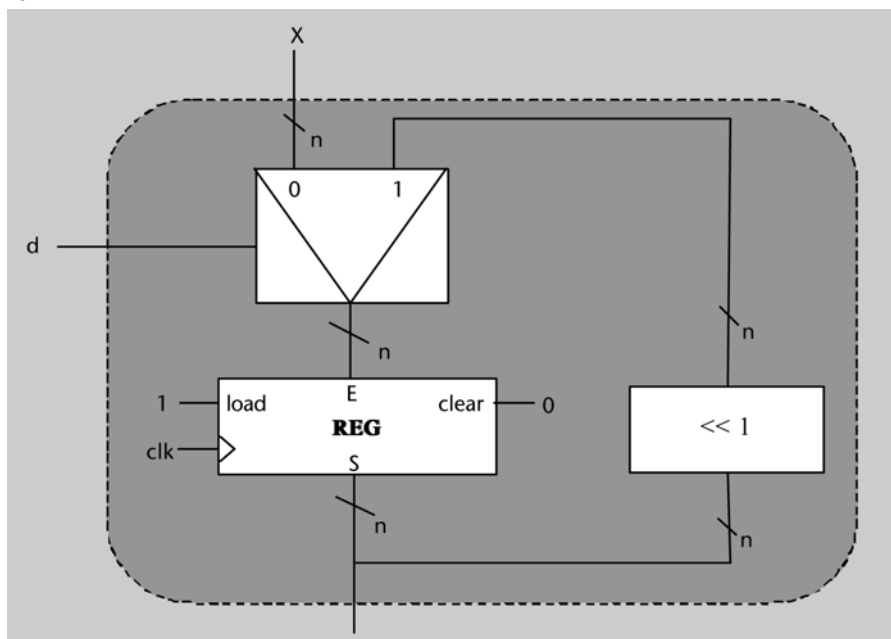
Si en un registro no dibujamos las señales *load* o *clear*, asumiremos que están a 1 y a 0, respectivamente.

Cuando modificamos el valor de un registro haciendo que se cargue con el valor que hay en la entrada E , decimos que hacemos una **escritura** en el registro.

Cuando analizamos el contenido de un registro a partir de la salida S , decimos que hacemos una **lectura**.

A partir de un registro y bloques combinacionales, se pueden diseñar circuitos con una funcionalidad determinada. Por ejemplo, el circuito de la figura 11 permite que el registro se pueda cargar con el valor de la entrada X o que pueda desplazar su contenido 1 bit a la izquierda, en función de la señal d .

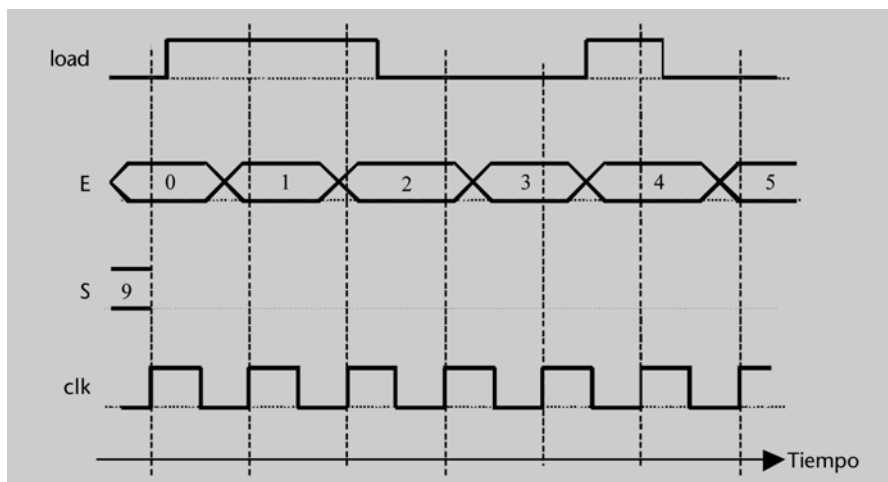
Figura 11



Actividades

9. La siguiente figura muestra los valores de las señales E y *load* de un registro de ocho bits durante cierto intervalo de tiempo. Indicad en la línea etiquetada como *tiempo* los

instantes en que el registro se carga con la entrada E y la secuencia de valores que tomará la salida S del circuito. Observad que inicialmente el valor de S es 9.

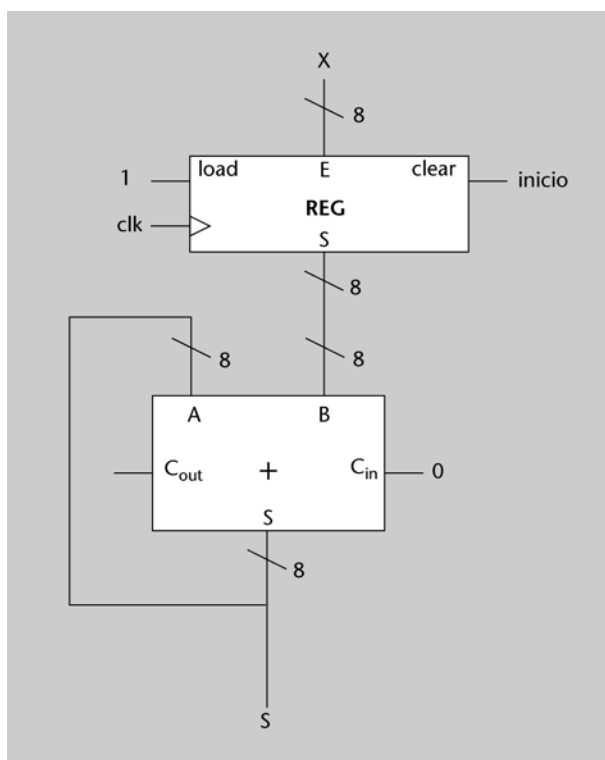


Interpretación de un cronograma

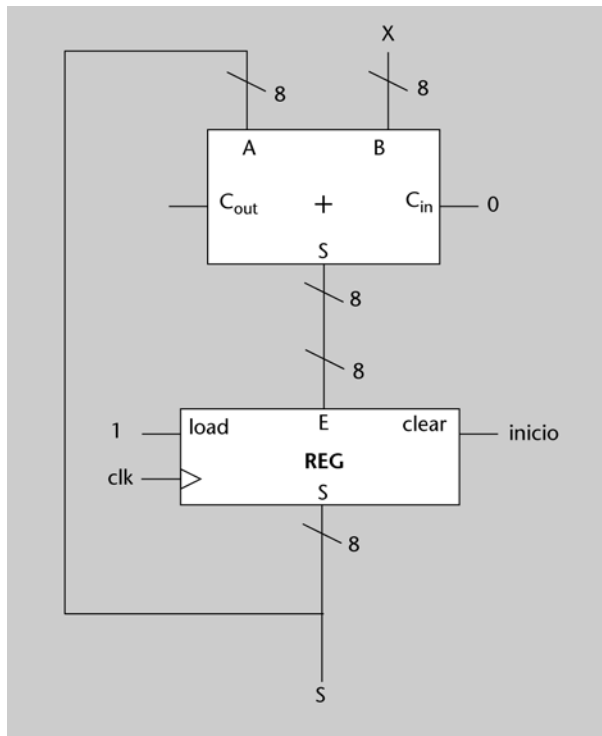
En un cronograma, los valores de palabras de n bits se dibujan mediante un hexágono, en cuyo interior se escribe el valor de la señal en decimal. Los puntos de contacto entre los extremos de los hexágonos indican el momento en que la palabra cambia de valor.

10. Se quiere diseñar un circuito con una entrada X de 8 bits que codifica un número natural en binario, y una salida S también de 8 bits que muestre en todo momento la suma acumulada de los valores que ha tenido X desde la inicialización del circuito y hasta el ciclo presente.

a) Un diseñador inexperto propone este circuito. ¿Por qué no es válido?



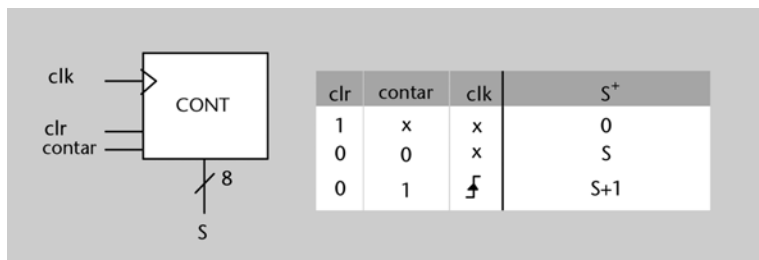
b) Un diseñador con un poco más de experiencia propone este otro circuito. ¿En qué mejora al anterior? Este circuito también tiene un inconveniente, ¿cuál es? ¿Es posible evitarlo?



11. Diseñad un bloque secuencial contador, **CONT**, con una salida S de 8 bits. Esta salida es un número natural (codificado en binario) que, siempre que la señal de entrada *contar* valga 1, se incrementará en cada flanco del reloj según esta expresión:

$$S^+ = (S + 1) \bmod 256$$

(por tanto, después de valer 255 pasa a valer 0). El bloque tiene, además, una entrada asíncrona, *clr*, que cuando vale 1 pone la salida a 0. El funcionamiento del circuito se muestra en la siguiente figura.



12. Diseñad un circuito secuencial con una entrada de datos E de 8 bits, dos entradas de control c_1 y c_0 de 1 bit y una salida S de 8 bits. E y S codifican números naturales en binario. El funcionamiento del circuito viene descrito por la tabla siguiente:

c_1	c_0	clk	S^+
0	0	\uparrow	0
0	1	\uparrow	E
1	0	\uparrow	$(S+1) \bmod 256$
1	1	\uparrow	$(S+E) \bmod 256$

Inicialmente, S debe tener el valor 0. El circuito no puede contener más de un bloque sumador.

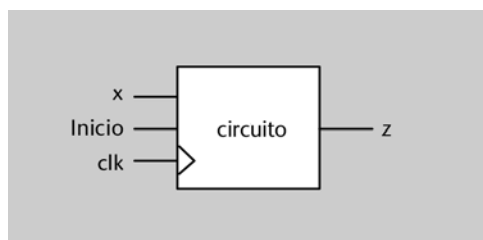
13. Se quiere diseñar un circuito que controle el estado de ocupación de un aparcamiento, en concreto poniendo en verde un semáforo mientras queden plazas libres y ponién-

dolo en rojo cuando esté lleno. El semáforo se pone verde cuando la señal *lleno* vale 0 y se pone rojo cuando vale 1.

El aparcamiento tiene capacidad para 500 vehículos. Cuando entra un coche, se produce un pulso en la señal *entra*, y cuando sale un coche se produce un pulso en la señal *sale*. Otros sistemas de control del aparcamiento garantizan que en un mismo ciclo nunca entrará y saldrá un coche al mismo tiempo, y generan un pulso en la señal *Inicio* cuando el aparcamiento abre las puertas, momento en el que no habrá ningún coche.

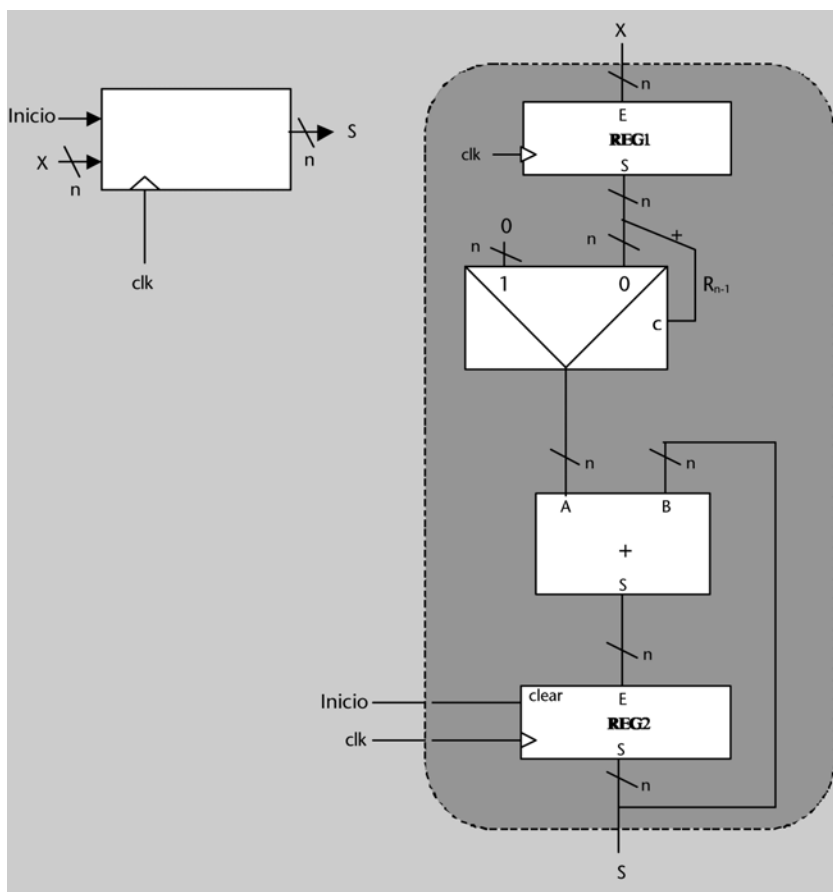
- a) ¿Qué señales de entrada y de salida debe tener el circuito? ¿De cuántos bits cada una?
- b) El circuito debe ser secuencial, ¿por qué?
- c) Diseñad el circuito e indicad el ancho de todos los buses.

14. Utilizando un registro de cuatro bits, bloques combinacionales y puertas, diseñad un circuito secuencial que reconozca si se ha producido la secuencia de valores 1010 en una entrada x de 1 bit. Los diferentes valores de x son los que tenga esta señal al llegar cada flanco de reloj. Cuando reconoce la secuencia, el circuito tiene que poner la señal de salida z a 1. El circuito tiene otra señal de entrada, *Inicio*, que genera un pulso a 1 para inicializar el circuito.



15. Analizad qué hace el circuito de la figura, teniendo en cuenta lo siguiente:

- X y S son números enteros representados en complemento a 2.
- La entrada X tiene un valor nuevo en cada ciclo de reloj.
- R_{n-1} se refiere al bit de más peso del registro REG1.
- La señal *Inicio* funciona de la forma habitual.

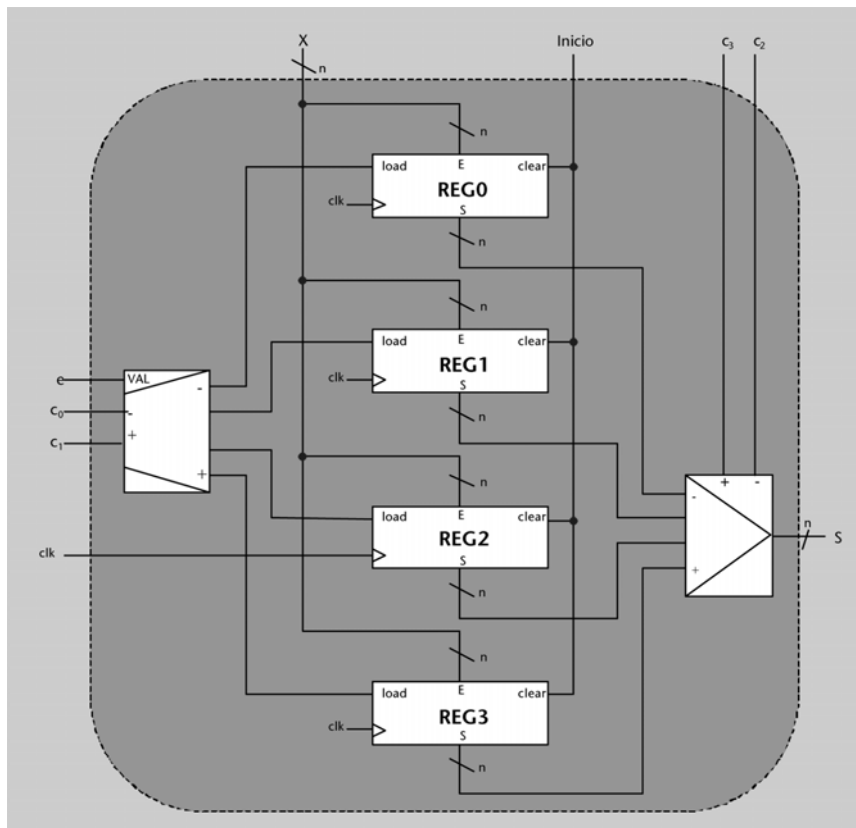


Precisiones sobre los gráficos

En los circuitos secuenciales, supondremos siempre que hay una señal única de reloj (*clk*). Sin embargo, en las figuras a veces no se conectan todas las entradas de reloj con una misma línea, con el fin de aclarar el dibujo.

En general, si en un circuito hay más de un punto identificado por un mismo nombre de señal, se entiende que los puntos están conectados, aunque no estén unidos por una línea. Por ejemplo, en la figura de la izquierda se escribe dos veces "clk", pero las dos corresponden a la misma señal.

16. Analizad qué hace el circuito de la figura siguiente:



Intersección de cables

A veces, en el diseño de circuitos se dibujan puntos en las intersecciones de cables para esclarecer el diseño.

17. Se quiere diseñar un circuito que controle la entrada a un edificio de acceso restringido, a cuya puerta hay un dispositivo con cinco teclas numéricas (del 0 al 4) y una tecla Entrar. Quien quiera entrar en el edificio debe teclear un código en el teclado numérico y pulsar Entrar; esto generará un pulso a 1 en la señal *entrar*, de un ciclo de reloj de duración, y pondrá en la señal *código* la representación en binario del código que se acaba de teclear. Se puede pulsar cualquier número de teclas numéricas, pero sólo se tendrán en cuenta las tres últimas que se hayan tecleado antes de pulsar *Entrar*.

Si el código coincide con la contraseña correcta, se enciende una luz verde (se consigue activando la señal *verde*) y la puerta se abre. Si el código es diferente de la contraseña, se enciende una luz amarilla (se consigue activando la señal *amarillo*) y la puerta no se abre. Si se teclea un código equivocado tres veces seguidas, se enciende una luz roja (se consigue activando la señal *rojo*) y la puerta se bloquea hasta que venga el portero y reinicie el sistema (generando un pulso en la señal *Inicio*).

Las luces se deben apagar un cierto número de segundos después de que se hayan encendido, y también se tienen que mantener apagadas mientras nadie haya tecleado ningún código. Sin embargo, de esto se ocupa otro subsistema (es decir, sólo os tenéis que preocupar de que cuando alguien haya tecleado un código se encienda la luz apropiada).

- ¿Qué entradas y salidas tiene el circuito? ¿De cuántos bits cada una?
- Diseñad el circuito, suponiendo que la contraseña correcta no varía nunca y que disponéis de un registro que estará siempre cargado con su codificación en binario. Indicad el ancho de todos los buses.
- ¿Qué elementos sería necesario añadirle para permitir que se pudiera cambiar la contraseña?

3.2. Banco de registros

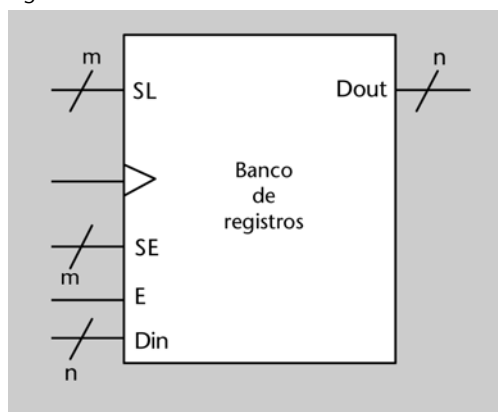
Un **banco de registros** es una agrupación de un cierto número de registros, todos del mismo número de bits. El contenido de los registros se puede leer y modificar gracias a los **puertos de lectura y de escritura**.

El número de registros de un banco siempre es una potencia de 2. Los 2^m registros están numerados desde 0 hasta $2^m - 1$.

La figura 12 muestra la representación gráfica de un banco de registros. Se puede ver que dispone de las siguientes señales:

- Una entrada de selección de lectura, SL , de m bits (si 2^m es el número de registros del banco).
- Una salida $Dout$, del mismo número de bits que los registros del banco. SL y $Dout$ forman el **puerto de lectura del banco**.
- Una entrada de selección de escritura, SE , de m bits.
- Una entrada de permiso de escritura, E , de un bit.
- Una entrada Din , del mismo número de bits que los registros del banco. SE , E y Din forman el **puerto de escritura del banco**.

Figura 12



Banco de registros

En general, un banco de registros puede tener varios puertos de lectura y de escritura. El número de puertos de cada tipo determina el número de operaciones de lectura y de escritura que se pueden hacer de forma simultánea. Por ejemplo, si tiene dos puertos de lectura y uno de escritura, se pueden leer dos registros y escribir otro de forma simultánea. En esta asignatura, siempre tendrán un puerto de escritura y uno de lectura.

El banco de registros funciona de la siguiente forma: !

- Para hacer una lectura, hay que poner en la entrada SL la codificación binaria del número de registro que se quiere leer. Entonces, el contenido de este registro estará presente en la salida $Dout$.
- Para hacer una escritura, hay que poner en SE la codificación binaria del número de registro que se quiere escribir y poner la entrada E a 1. Cuando se produzca el próximo flanco ascendente de reloj, el valor que haya en Din se escribirá en el registro indicado por SE .

Como se puede observar, la entrada E funciona como una señal de carga: si está a 0 no se puede modificar el contenido de ningún registro del banco.

Actividades

18. Se dispone de un banco de registros de 16 bits. A partir del cronograma que se muestra a continuación, haced lo siguiente:

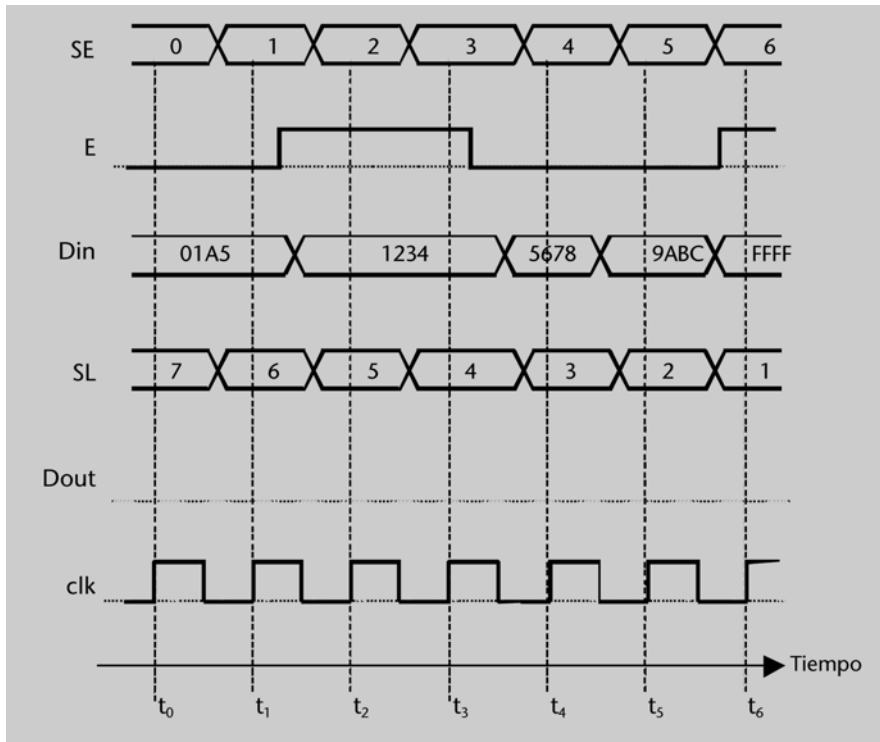
- a) Indicad qué registros se escriben, en qué momento y con qué valor.

Atención

Fijémonos en que en el banco “siempre se está leyendo”; dicho de otra forma, en $Dout$ hay en todo momento el contenido del registro indicado por SL . En cambio, sólo se escribe en los ciclos en que $E = 1$; la escritura se realizará al final del ciclo, coincidiendo con el próximo flanco de reloj.

b) Indicad en el cronograma el valor de la salida *Dout* suponiendo que el valor inicial de los registros, expresado en hexadecimal, sea el siguiente:

$R0 = 0000$,
 $R1 = 1111$,
 $R2 = 2222$,
 $R3 = 3333$,
 $R4 = 4444$,
 $R5 = 5555$,
 $R6 = 6666$,
 $R7 = 7777$.



c) Indicad el valor de todos los registros del banco después del instante t_6 .

19. Implementad un circuito con la misma funcionalidad que el de la actividad 16, pero sin la señal *Inicio*, utilizando sólo un banco de cuatro registros.

3.3. Memoria RAM

La **memoria RAM** es un bloque secuencial que permite guardar el valor de un cierto número de palabras (2^m) de un cierto número de bits (n).


La funcionalidad de una memoria RAM, pues, es similar a la de un banco de registros. Las diferencias entre ambos bloques son las siguientes:

- El tamaño: un banco de registros suele guardar unas cuantas decenas de palabras, mientras que una memoria RAM puede guardar varios millones.
- La velocidad: por cómo se implementan físicamente una y otra, el tiempo de respuesta (retraso) de una memoria RAM es mucho mayor que el de un banco de registros (y, por tanto, este último es más rápido).

RAM

La denominación RAM proviene del inglés *random access memory* ("memoria de acceso aleatorio"). Se le dio este nombre porque el tiempo que se tarda en hacer una lectura o una escritura no depende de la palabra a la que se acceda (a diferencia de lo que pasaba en otros dispositivos de memoria que se utilizan en los primeros computadores).

- La implementación interna de ambos bloques es muy diferente.
- En un banco de registros, las escrituras se hacen coincidiendo con los flancos ascendentes del reloj. En cambio, la memoria no tiene señal de reloj: las escrituras son efectivas un cierto intervalo de tiempo después de haber dado la orden de escribir.

En este curso no se estudiarán detalladamente estas cuestiones. Tenemos suficiente con la idea de que un banco de registros es pequeño y rápido, y una memoria RAM es grande y lenta. También asumiremos que la memoria RAM está sincronizada de la misma manera que los bancos de registros, con una señal de reloj. 

Como en el caso de la memoria ROM que hemos visto en el módulo “Los circuitos lógicos combinacionales”, la memoria RAM se puede ver como un archivador con cajones, numerados con una dirección. Cada cajón guarda una palabra. Llamamos $M[i]$ a la palabra guardada en el cajón con dirección i .

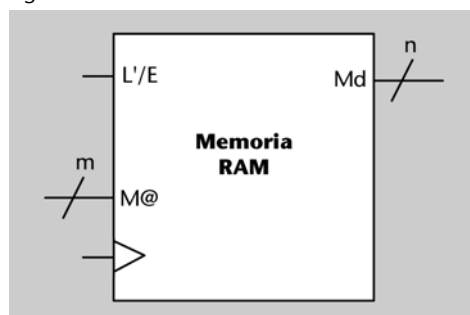
La figura 13 muestra la representación de una memoria RAM con un solo **puerto de lectura/escritura**. Se puede ver que dispone de las siguientes señales:

- Una entrada de direcciones, $M@$. Si la memoria tiene capacidad para 2^m palabras, la entrada tendrá m bits.
- Una entrada/salida de datos, Md , de n bits (si las palabras que guarda la memoria son de n bits).
- Una entrada de control, L'/E , que indica en todo momento si se tiene que hacer una lectura o una escritura.

Nota

Las memorias también pueden tener un cierto número de puertos de lectura y de escritura, que determinan el número de operaciones que se pueden hacer de forma simultánea. En una memoria con un solo puerto de lectura/escritura como la que presentamos en estos apuntes, en cada momento sólo se puede hacer o una lectura o una escritura.

Figura 13



El funcionamiento de la memoria es el siguiente: 

- Si $L'/E = 0$, entonces se hace una lectura: por el bus Md sale el valor de la palabra que está guardada en la dirección indicada por $M@$. Si $M@$ cambia mientras $L'/E = 0$, Md variará también inmediatamente (en este curso, asumimos que las lecturas a memoria tienen retraso 0).

- Si $L'/E = 1$, entonces se hace una escritura: la palabra indicada por $M@$ toma el valor que hay en Md en el primer flanco de reloj que se produzca después de activar L'/E (asumimos que las escrituras tampoco tienen retraso). Mientras $L'/E = 1$, el bus Md toma el valor 0 (a diferencia de lo que sucede en los bancos de registros, que “siempre están leyendo”).

La siguiente tabla de verdad resume el funcionamiento de la memoria RAM.

L'/E	
0	$Md := M[M@]$
1	$M[M@] := Md$

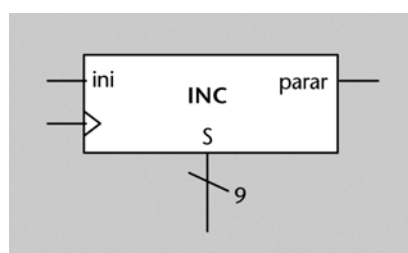
La capacidad de una memoria RAM se suele medir en *bytes* (palabras de ocho bits). Como hemos dicho, suele contener varios millones de palabras, y, por eso, para indicar su capacidad se suelen utilizar las letras k, M y G, que tienen los siguientes significados:

Letra	Significado	Ejemplo
k	$2^{10} = 1.024 \approx 10^3$	16 kb (16 kilobytes) = 2^{14} bytes
M	$2^{20} = k \cdot k \approx 10^6$	32 Mb (32 megabytes) = 2^{25} bytes
G	$2^{30} = k \cdot M \approx 10^9$	2 Gb (2 gigabytes) = 2^{31} bytes

Actividades

20.

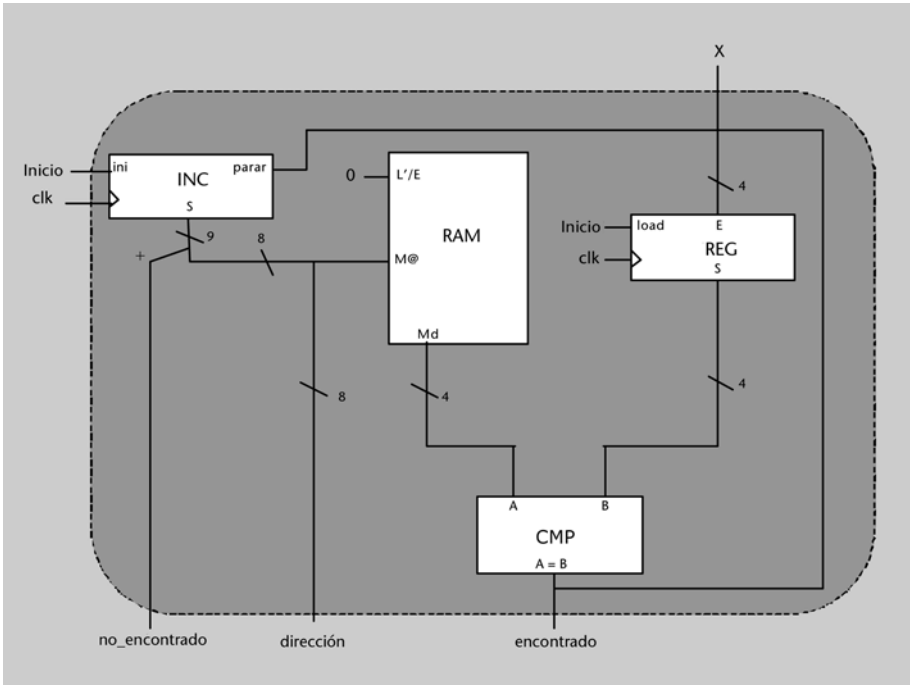
a) Diseñad un bloque incrementador, INC, con una salida S de nueve bits (parecido al que se ha diseñado en la actividad 11), que funciona de la siguiente forma:



- Cuando la entrada *ini* está en 1, S se pone a 0.
- En cada flanco de reloj, S se incrementa en 1.
- Si en algún momento la señal *parar* vale 1, entonces S deja de incrementarse y mantiene su valor en la salida.
- Cuando S llega a 256 también deja de incrementarse, y su salida se mantiene en 256 hasta que en la entrada *ini* vuelva a haber un 1.

A partir del circuito de la siguiente figura (el bloque INC es el diseñado en el apartado a), contestad estas preguntas:

- Indicad cuáles son las entradas y salidas del circuito, y cuántos bits tiene cada una.
- ¿Qué bloques del circuito son combinacionales y cuáles son secuenciales?
- ¿Cuál es el tamaño de la memoria RAM del circuito?
- Indicad qué función hace este circuito y razonad la respuesta (recordad que *Inicio* genera un pulso a 1 cuando el circuito se pone en marcha).



4. El modelo de Moore

4.1. Estado. Transiciones

El **modelo de Moore** es una forma de expresar o modelizar el funcionamiento de un circuito lógico secuencial. Es fundamental en los conceptos de *estado* y *transiciones entre estados*.

Otras modelizaciones

Otra forma muy usual de modelizar un circuito secuencial es el llamado **modelo de Mealy** (que no se estudia en esta asignatura).

Para introducir estos conceptos utilizaremos un ejemplo: 

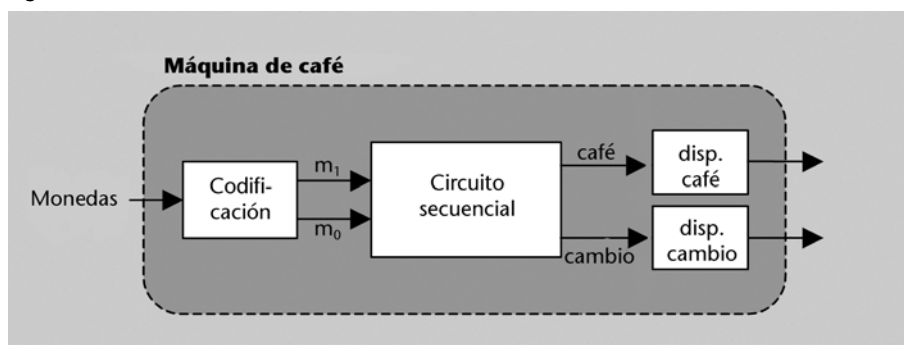
Imaginemos un circuito que controla el funcionamiento de una máquina expendedora de café. Para simplificar, asumiremos que la máquina sólo sirve un único tipo de café, y que sólo admite monedas de 0,5 euros y de 1 euro. La información sobre las monedas introducidas se codifica mediante dos señales lógicas m_1 y m_0 , tal como se muestra en la tabla al margen. Las señales m_1 y m_0 serán las entradas del circuito.

Tabla 1

Monedas introducidas	m_1	m_0
Ninguna moneda	0	0
Moneda de 0,5 euros	0	1
Moneda de 1 euro	1	0

El precio del café es de 1,5 euros. La máquina tiene dos dispositivos de salida, uno para servir el café y otro para dar el cambio. Estos dos dispositivos están controlados respectivamente por las señales lógicas *café* y *cambio*, de forma que la máquina dará café o cambio cuando la señal correspondiente esté a 1. La figura 14 muestra el esquema del funcionamiento de la máquina.

Figura 14



Para determinar en cada momento si tiene que servir café o dar cambio, la máquina necesita saber cuánto dinero se le ha introducido hasta el momento. Se pueden dar las siguientes situaciones:

- Se han introducido 0 euros.
- Se han introducido 0,5 euros.
- Se ha introducido 1 euro.
- Se han introducido 1,5 euros.
- Se han introducido 2 euros.

Se llama **estado** a cada situación diferente en la que se puede encontrar un circuito.

En nuestro ejemplo, la máquina de café puede encontrarse en cinco estados diferentes, los descritos por las cinco posibilidades anteriores.

Las señales de salida del circuito tomarán un valor u otro según en qué estado se encuentre el circuito. En el caso de la máquina de café, la señal *café* estará a 1 cuando se hayan introducido al menos 1,5 euros, y la señal *cambio* estará a 1 si se han introducido 2.

Una **tabla de salida** se expresa mediante el valor que toman las señales de salida en cada estado.


La tabla de salidas tiene a la izquierda los diferentes estados y a la derecha, el valor que toman las diferentes señales de salida en cada estado. La tabla de salidas del circuito de la máquina de café es la siguiente:

Estado	café	cambio
Se han introducido 0 euros.	0	0
Se han introducido 0,5 euros.	0	0
Se ha introducido 1 euro.	0	0
Se han introducido 1,5 euros.	1	0
Se han introducido 2 euros.	1	1

Explicación de la tabla de salidas


Cuando la señal *café* esté a 1, la máquina servirá un café. Cuando la señal *cambio* esté a 1 (es decir, cuando el estado sea “se han introducido 2 euros”), la máquina dará 0,5 euros de cambio (recordemos que un café vale 1,5 euros).

A medida que el tiempo avanza, el circuito cambia de estado en función de los valores que vayan llegando por las entradas. Mientras no se haya introducido ninguna moneda, la máquina de café se encontrará en el estado “se han introducido 0 euros”. Cuando se introduzca una moneda de 0,5 euros, pasará al estado “se han introducido 0,5 euros”; si la moneda es de 1 euro pasará al estado “se ha introducido 1 euro”.

El circuito no tiene que recordar necesariamente todos los valores que han llegado por las entradas, sino que los tiene que resumir en informaciones que sean relevantes para su funcionamiento. En el ejemplo de la máquina de café, se puede haber introducido 1 euro mediante una sola moneda de 1 o dos monedas de 0,5 euros. A la máquina le es indiferente cómo se haya hecho, ya que ambas acciones llevan a la misma situación: “se ha introducido 1 euro”. Por eso no están los estados “se han introducido dos monedas de 0,5 euros” y “se ha introducido una moneda de 1 euro”, sino que ambas informaciones se resumen en el estado “se ha introducido 1 euro”. 

Llamamos **estado actual** al estado en que se encuentra la máquina en un instante dado. El nuevo valor (moneda o ausencia de monedas) que llega por la entrada determinará cuál será el próximo estado en que se encontrará el circuito: el **estado futuro**.

Se llama **transición** al paso del estado actual a un estado futuro.

Una señal de reloj sincroniza los circuitos secuenciales. En cada flanco ascendente se produce una transición hacia un estado futuro u otro en función del valor de las entradas. Las consideraciones con respecto a la evolución temporal de los circuitos se tratan en otro subapartado. 

La sincronización de los circuitos secuenciales se estudia con detalle en el subapartado 4.3. de este módulo.

Todas las transiciones que se pueden dar en un circuito secuencial se especifican mediante la **tabla de transiciones**.

La tabla de transiciones tiene a la izquierda todas las combinaciones posibles de estados actuales y valores de las entradas, y a la derecha, el estado futuro al que lleva cada combinación. A continuación se muestra la tabla de transiciones del ejemplo de la máquina de café (para hacer la tabla más legible, en lugar de escribir “se han introducido 0 euros” escribimos “cero”, y de forma análoga para todos los estados).

Estado actual	Entrada	Estado futuro
cero	Ninguna moneda	cero
cero	Moneda de 0,5 euros	medio
cero	Moneda de 1 euro	uno
medio	Ninguna moneda	medio
medio	Moneda de 0,5 euros	uno
medio	Moneda de 1 euro	uno y medio
uno	Ninguna moneda	uno
uno	Moneda de 0,5 euros	uno y medio
uno	Moneda de 1 euro	dos
uno y medio	Ninguna moneda	cero
uno y medio	Moneda de 0,5 euros	medio
uno y medio	Moneda de 1 euro	uno
dos	Ninguna moneda	cero
dos	Moneda de 0,5 euros	medio
dos	Moneda de 1 euro	uno

En cada transición, el estado futuro puede coincidir o no con el estado actual.

Nota


Cuando ya se hayan introducido 1,5 euros o 2 euros, la máquina servirá un café y, si es el caso, dará el cambio. Si en una de estas situaciones se introduce otra moneda, el circuito entenderá que corresponde a una nueva petición de café.

Un circuito secuencial siempre tiene un estado que refleja la situación “aún no ha pasado nada”, es decir, “no es necesario recordar ninguno de los valores que ha llegado por la entrada”. Este estado se denomina **estado inicial**.

Cuando un circuito se pone en funcionamiento, está en el estado inicial. Ahora bien, también puede estar en otros momentos, si la funcionalidad del circuito así lo requiere. En el caso de la máquina de café, el estado inicial es “se han introducido 0 euros”. Cuando la máquina haya servido un café y mientras no se introduzcan más monedas, volverá al estado inicial.

Una vez especificado cuál es el estado inicial, la tabla de transiciones y la tabla de salidas describen completamente el comportamiento de un circuito lógico secuencial de acuerdo con el modelo de Moore.

La tabla de transiciones también se puede escribir con las señales de entrada codificadas en binario (ya que todo el circuito trabaja sólo con señales lógicas, como ya sabemos). En el ejemplo de la máquina de café, las entradas se pueden codificar en binario mediante las señales m_1 y m_0 , tal como se ha visto en la tabla 1.

Cuando escribimos una tabla de transiciones con las entradas codificadas en binario, ponemos todas las combinaciones de las variables de entrada posibles, aunque algunas no se produzcan nunca. 

En nuestro ejemplo, las variables m_1 y m_0 no tomarán nunca los valores [1 1]; pese a todo, ponemos estas combinaciones en la tabla de transiciones que se muestra a continuación. El valor del estado futuro en estos casos será x , ya que se trata de combinaciones *don't care*.

Estado actual	m_1	m_0	Estado futuro
cero	0	0	cero
cero	0	1	medio
cero	1	0	uno
cero	1	1	\times
medio	0	0	medio
medio	0	1	uno
medio	1	0	uno y medio
medio	1	1	\times
uno	0	0	uno
uno	0	1	uno y medio
uno	1	0	dos
uno	1	1	\times
uno y medio	0	0	cero
uno y medio	0	1	medio
uno y medio	1	0	uno
uno y medio	1	1	\times
dos	0	0	cero
dos	0	1	medio
dos	1	0	uno
dos	1	1	\times

Ejemplo de especificación de un circuito secuencial con el modelo Moore

Veamos otro ejemplo de especificación de un circuito secuencial mediante el modelo de Moore.

Sea un circuito con una señal lógica de entrada, x , y una de salida, z . La señal de salida tiene que valer 1 siempre que se cumpla que por la entrada hayan llegado un número par de unos y un número par de ceros (recordemos que cero es un número par).

El circuito se puede encontrar en los siguientes estados:

- Ha llegado un número par de unos y un número par de ceros.
- Ha llegado un número par de unos y un número impar de ceros.
- Ha llegado un número impar de unos y un número par de ceros.
- Ha llegado un número impar de unos y un número impar de ceros.

El valor de la señal de salida en cada estado viene dado por la siguiente tabla de salidas:

Estado Ha llegado un número...	Salida z
... par de unos y par de ceros	1
... par de unos e impar de ceros	0
... impar de unos y par de ceros	0
... impar de unos e impar de ceros	0

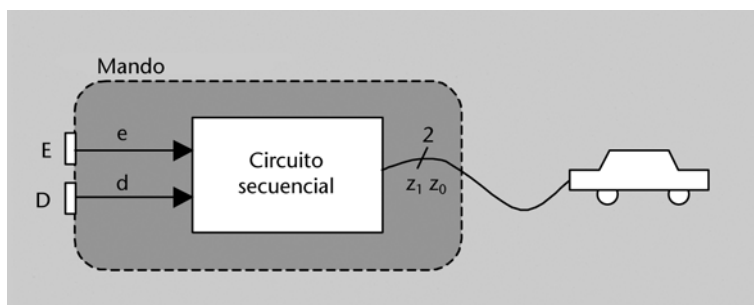
El circuito irá pasando por un estado u otro según si el valor de la entrada vale 0 ó 1. En concreto, la tabla de transiciones será la siguiente:

Estado actual Ha llegado un número...	Entrada x	Estado futuro Ha llegado un número...
... par de unos y par de ceros	0	... par de unos e impar de ceros
... par de unos y par de ceros	1	... impar de unos y par de ceros
... par de unos e impar de ceros	0	... par de unos y par de ceros
... par de unos e impar de ceros	1	... impar de unos e impar de ceros
... impar de unos y par de ceros	0	... impar de unos e impar de ceros
... impar de unos y par de ceros	1	... par de unos y par de ceros
... impar de unos e impar de ceros	0	... impar de unos y par de ceros
... impar de unos e impar de ceros	1	... par de unos e impar de ceros

Para referirnos a *estado actual* y *estado futuro*, también podemos escribir Estado y Estado⁺.

Actividades

21. Se quiere diseñar el mando de control remoto de un coche de juguete. El mando tiene dos botones: E y D .



Si el coche está parado, al pulsar cualquier botón se pone en movimiento: gira a la izquierda si se pulsa E , gira a la derecha si se pulsa D y va adelante si se pulsan los dos botones a la vez. Mientras el coche está en movimiento, si se pulsa E hará lo siguiente:

- girará a la izquierda si iba recto,
- irá recto si giraba a la derecha,
- continuará girando a la izquierda si ya lo hacía.

Sucedará de forma análoga cuando se pulse D . Cuando se pulsen los dos botones a la vez, se parará.

El mando dispondrá de un circuito secuencial que recibe como entrada dos señales e y d conectadas a los botones E y D respectivamente (1: pulsar; 0: no pulsar), y genera dos señales, z_1 y z_0 , que gobernarán el coche según la tabla que vemos al margen.

- ¿Qué estados tiene el circuito?
- ¿Cuál es el estado inicial?
- Escribid la tabla de salidas y la de transiciones.

z_1	z_0	Acción del coche
0	0	Girar a la derecha
0	1	Girar a la izquierda
1	0	Detenerse
1	1	Moverse adelante

22. Sea un circuito secuencial con dos señales de entrada x e y , ambas de un bit, y una señal de salida z también de un bit. La señal de salida se tiene que poner a 1 cuando en al menos tres ocasiones los valores de x e y se hayan igualado.

- ¿Qué estados tiene el circuito?
- ¿Cuál es el estado inicial?
- Escribid la tabla de salidas y la de transiciones.

4.2. Representación gráfica: grafos de estado

La especificación del funcionamiento de un sistema secuencial mediante el modelo de Moore se representa gráficamente con un **grafo de estados**, de la forma siguiente:

1) Para cada estado se dibuja un círculo, con el nombre del estado en la parte superior. El círculo correspondiente al estado inicial se señala con una flecha y la palabra *Inicio*.

Recordemos

Los circuitos secuenciales suelen tener una señal de entrada *Inicio* que genera un pulso a 1 para indicar al circuito que se ponga en funcionamiento.

2) En la parte inferior de cada círculo se escribe el valor que toman las señales de salida cuando el circuito se encuentra en el estado correspondiente a este círculo.

3) Las transiciones se representan mediante flechas o **arcos**, que tienen su origen en el círculo correspondiente al estado actual y la punta en el círculo correspondiente al estado futuro. El valor de las señales de entrada asociado con la transición se escribe al lado del arco.

Los valores escritos al lado de un arco también se llaman *etiquetas del arco*.

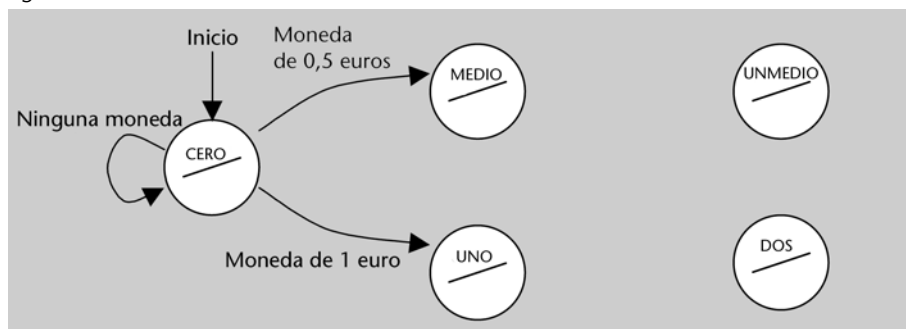
La figura 15 muestra la representación gráfica de los estados de la máquina de café, y las transiciones que parten del estado “se han introducido 0 euros”. En esta figura hemos dado a los diferentes estados estos nombres:

Estado	Nombre
Se han introducido 0 euros.	CERO
Se han introducido 0,5 euros.	MEDIO
Se ha introducido 1 euro.	UNO
Se han introducido 1,5 euros.	UNMEDIO
Se han introducido 2 euros.	DOS

Consejo práctico

Al especificar un circuito secuencial mediante un grafo de estados, se suele dar a cada estado un nombre corto, ya que así se puede escribir cómodamente. El nombre que se dé a cada estado es indiferente, pero resulta práctico que sea un mnemotécnico que nos remita de alguna forma a la situación que refleja cada estado.

Figura 15



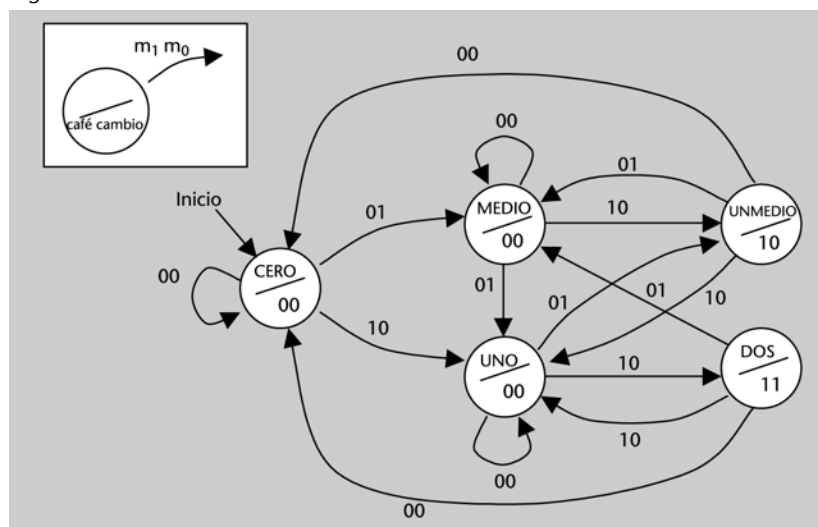
El grafo completo se muestra en la figura 16, con las entradas codificadas en binario (de acuerdo con la tabla 1). En el recuadro de la parte superior izquierda encontramos la leyenda del grafo.

La **leyenda** del grafo de estados indica el orden en que se escriben las señales de entrada en la etiquetas de los arcos y las señales de salida dentro de los círculos.

La leyenda de un grafo

Si un grafo con más de una señal de entrada o de salida no dispone de leyenda, es imposible descifrar su significado.

Figura 16



Combinaciones imposibles

En general, en un grafo no aparecen las combinaciones de las variables de entrada que no se darán nunca. Por este motivo, en la figura 16 no hay ninguna etiqueta [1 1].

Gracias a la leyenda sabemos que, en las parejas de valores de las etiquetas de la figura 16, el de la izquierda corresponde a m_1 y el de la derecha a m_0 (y no al revés). La leyenda nos indica también que, de los valores de salida, el de la izquierda corresponde a la señal *café* y el de la derecha, a la señal *cambio*.

4.2.1. Mecánica de diseño

A partir del enunciado del comportamiento de un circuito secuencial, para encontrar el grafo de estados podemos seguir al algoritmo que presentamos a continuación:

- 1) Analizar qué entradas y salidas tiene el circuito y determinar la leyenda del grafo.

- 2) Dibujar un círculo para el estado inicial, darle un nombre y escribir el valor de las salidas en este estado.
- 3) Hacer una lista de todas las combinaciones de valores que pueden tomar las señales de entrada en este estado. Para cada una, deducir qué transición provoca. Si la transición comporta la aparición de un estado inexistente hasta el momento, incorporarlo al grafo, darle un nombre y escribir el valor adecuado para las salidas. Dibujar el arco correspondiente a la transición.
- 4) Repetir el paso 3 para todos los estados nuevos que hayan aparecido, hasta que no aparezca ninguno nuevo.

Circuitos reconocedores de secuencia

Veamos un ejemplo de construcción del grafo a partir de la especificación del funcionamiento del circuito.

Se quiere dibujar el grafo de estados de un circuito con una señal de entrada x y una de salida z , ambas de un bit. Inicialmente, la salida tiene que valer 0. Cuando en la entrada se haya producido la secuencia de valores 101, la salida se tiene que poner a 1. La salida se tiene que volver a poner a 0 cuando por la entrada haya llegado la secuencia de valores 001.

Los circuitos que generan un 1 en la salida cuando se ha producido una secuencia de valores determinada en la entrada se llaman *reconocedores de secuencia*.

En este caso, las señales de entrada y de salida son de un único bit; por tanto, no hay que determinar la orden de las señales en la leyenda, sólo podemos poner el nombre.

El circuito tendrá un estado inicial, llamado *INI* con salida 0. Puesto que la entrada es de un único bit, sólo puede tomar los valores 0 y 1.

Si estando en el estado *INI*, por la entrada llega un 1, tenemos que recordarlo, ya que puede ser el principio de la secuencia 101 que queremos reconocer. Aparece, pues, un nuevo estado, que llamamos “ha llegado un 1”; la salida en este estado continúa valiendo 0. En cambio, si estando en el estado inicial llega un 0, no es preciso recordarlo, y nos quedaremos en el mismo estado *INI*.

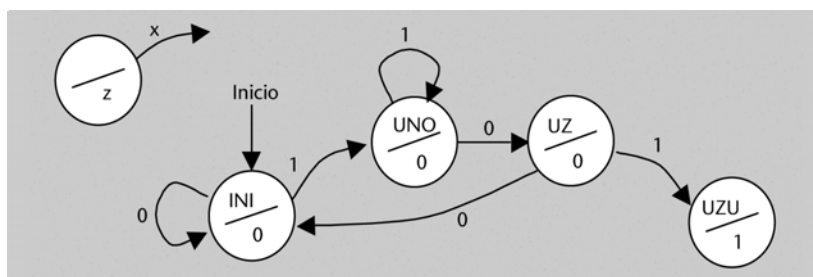
Estando en el estado “ha llegado un 1”, si llega un 0 tenemos que pasar a un estado nuevo, que llamaremos “ha llegado la subsecuencia 10”, con salida 0. Si llega un 1, se rompe la secuencia anterior, pero este nuevo 1 puede ser a su vez el inicio de una nueva secuencia 101. Por tanto, nos tenemos que quedar en el mismo estado “ha llegado un 1”.

Situémonos ahora en el nuevo estado que ha aparecido, “ha llegado la subsecuencia 10”. Si llega un 0, tenemos que los tres últimos valores que han llegado por la entrada son 100 y, por tanto, ninguno de ellos puede formar parte de la secuencia 101. Volvemos, pues, al estado *INI*. En cambio, si llega un 1, tenemos que los tres últimos valores que han llegado son 101, que es justamente la secuencia que el circuito tiene que reconocer. Pasaremos, pues, a un nuevo estado “ha llegado la secuencia 101”, en el que la salida vale 1.

Demos a los estados que han aparecido hasta ahora los siguientes nombres:

Estado	Nombre
Ha llegado un 1	UNO
Ha llegado la subsecuencia 10	UZ
Ha llegado la secuencia 101	UZU

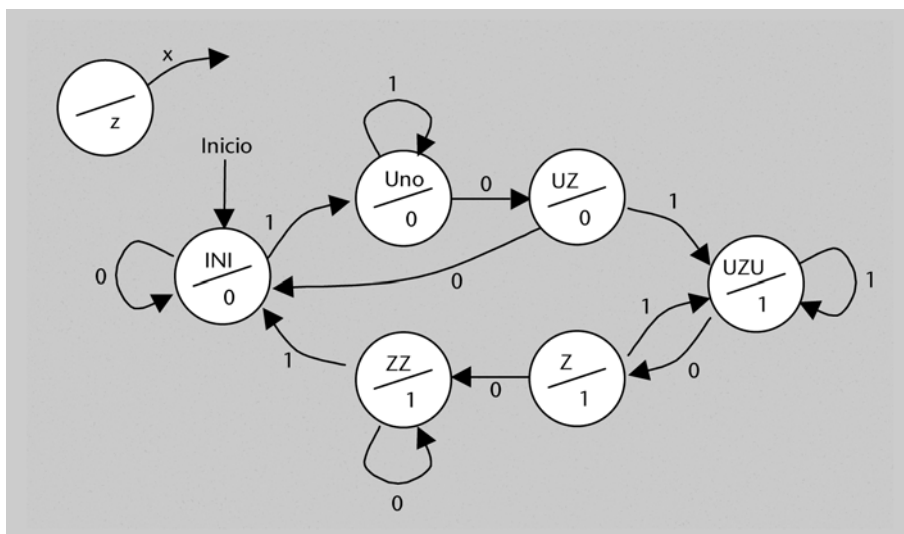
La parte del grafo que hemos construido hasta ahora se muestra a continuación:



Una vez ha llegado al estado *UZU*, el circuito tiene que reconocer la secuencia 001 para saber cuándo tiene que volver a poner la salida a 0. Razonemos de forma análoga al caso anterior y obtendremos que deberá presentar los siguientes estados:

Estado	Nombre
Ha llegado un 0	Z
Ha llegado la subsecuencia 00	ZZ
Ha llegado la secuencia 001	ZZU

Sin embargo, vemos que el estado *ZZU* coincide con el estado inicial, ya que la salida debe valer 0 y el circuito tiene que reconocer a partir de este momento la secuencia 101; es decir, el circuito se encuentra en la misma situación que al empezar a funcionar. Por último, obtenemos el siguiente grafo completo:

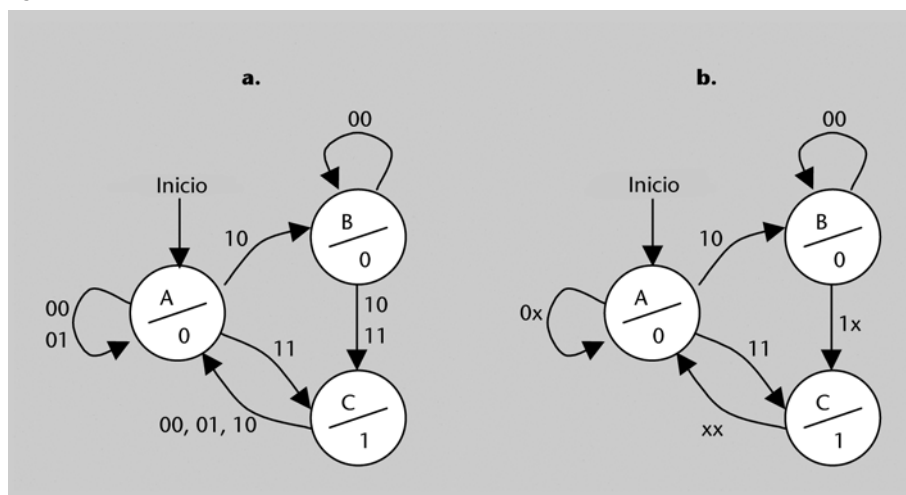


4.2.2. Notación

En un grafo, si se produce una transición de un estado determinado hacia el mismo estado futuro por más de una combinación de valores de las señales de entrada, escribiremos las etiquetas correspondientes una debajo de otra, o bien separadas por comas, tal como se muestra en el gráfico **a** de la figura 17. Observamos que, en el circuito correspondiente a este grafo, estando en el estado *B* no se dará nunca la combinación de entrada [0 1], y estando en el estado *C* no se dará nunca la combinación [1 1].

En una etiqueta podemos escribir también *x* para referirnos a un valor cualquiera de una señal de entrada (igual que se hace en las tablas de la verdad). Cuando de un estado se pasa siempre a un mismo estado de futuro, independientemente del valor de las entradas (como es el caso del estado *C* en el gráfico **a** de la figura 17), se puede poner una sola etiqueta *x* como valor de todas las variables, incluso si algunas combinaciones no se dan nunca. Así, los dos grafos de la figura 17 son equivalentes (observemos que, estando en el estado *C*, la combinación de entrada 11 no se puede producir).

Figura 17



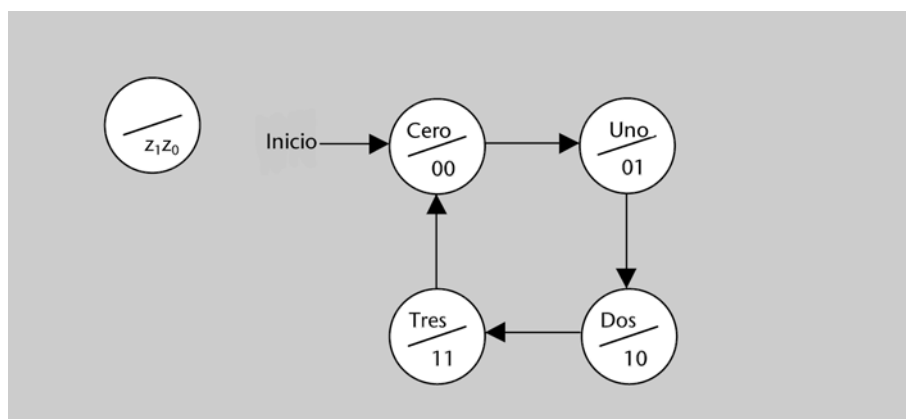
4.2.3. Circuitos sin entradas

Un circuito secuencial puede no tener ninguna señal. En este caso, siempre se producirán las mismas transiciones entre estados y, por tanto, la secuencia de valores en las salidas será siempre la misma.

Los contadores módulo n

En general, un contador módulo n es un circuito que genera cíclicamente la secuencia de valores $0, 1, \dots, n - 1$.

Imaginemos un circuito cuya misión es generar cíclicamente la secuencia de números $0, 1, 2$ y 3 , codificados en binario. Este circuito recibe el nombre de *contador módulo 4*, y su grafo de estados es el que se muestra en la siguiente figura.



Nota

Observad que cuando el circuito no tiene ninguna entrada, al diseñar su grafo de estados, la leyenda no tiene ningún arco ni ninguna etiqueta asociada al mismo.

Actividades

23. Dibujad el grafo de estados del circuito que se describe en la actividad 17.

24. Dibujad el grafo de estados del circuito secuencial que se comporta tal como describen las siguientes tablas:

a) Estado inicial: A

Tabla de transiciones			
Estado	x_1	x_0	Estado ⁺
A	0	0	A
A	0	1	C
A	1	0	B
A	1	1	B
B	0	0	B
B	0	1	D
B	1	0	A
B	1	1	A
C	0	0	A
C	0	1	C
C	1	0	D
C	1	1	D
D	0	0	D
D	0	1	B
D	1	0	C
D	1	1	C

Tabla de salidas			
Estado	y_2	y_1	y_0
A	1	0	0
B	0	1	0
C	0	0	1
D	1	1	1

b) Estado inicial: C

Tabla de transiciones			
Estado	e_1	e_0	Estado ⁺
A	0	x	B
A	1	0	C
A	1	1	A
B	0	0	B
B	0	1	C
B	1	0	A
B	1	1	x
C	0	x	A
C	1	0	A
C	1	1	x

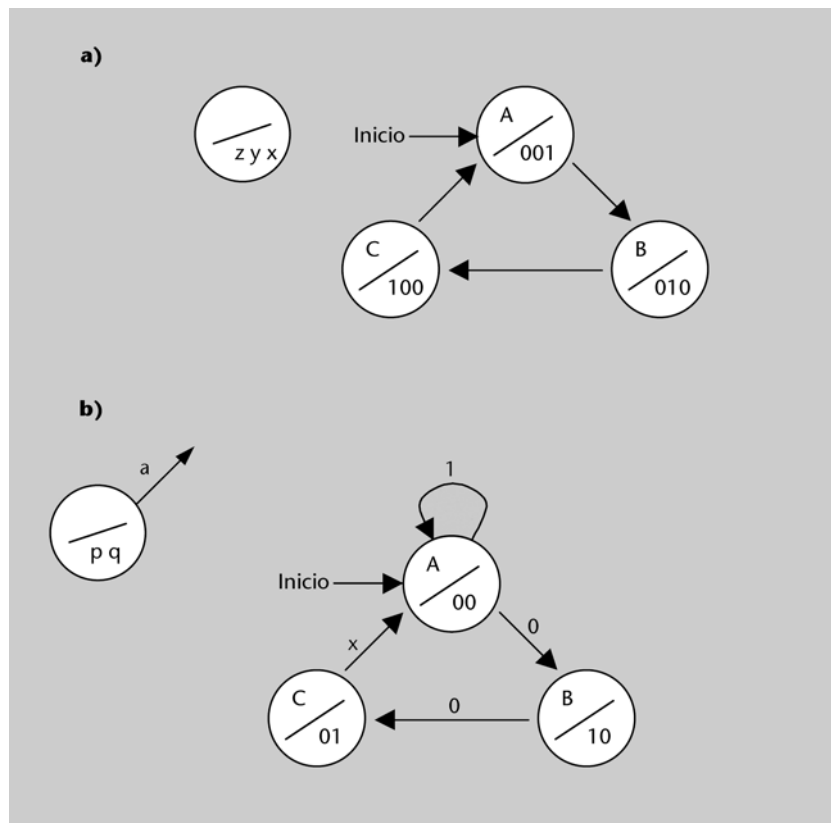
Tabla de salidas	
Estado	z
A	1
B	1
C	0

c) Estado inicial: E2

Tabla de transiciones	
Estado	Estado ⁺
E0	E2
E1	E0
E2	E1

Tabla de salidas	
Estado	z
E0	1
E1	0
E2	1

25. Escribid las tablas de transiciones y salidas de los circuitos que se comportan tal como describen los grafos de estados siguientes:



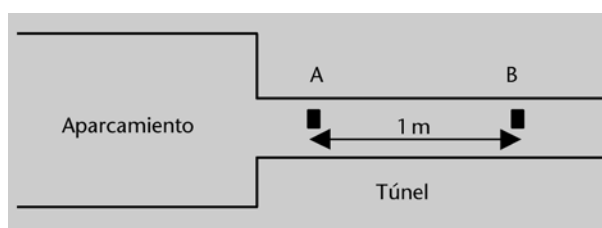
26. Dibujad el grafo de estados de un circuito que funcione como un contador reversible módulo 5. Un contador reversible cuenta adelante o atrás en función de una señal de entrada x :

- $x = 0$: cuenta adelante
- $x = 1$: cuenta atrás

Inicialmente, la salida tiene que valer 0. La salida no se codificará en sistema binario, sino como se indica en esta tabla:

Valor de la salida	Señales de salida $z_3 z_2 z_1 z_0$
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 1 1 1
4	1 1 1 1

27. En un aparcamiento se necesita saber el número de coches que hay en cada momento. Los coches entran y salen por el mismo túnel, en el que sólo cabe un coche. En el túnel hay dos sensores, A y B , separados por un metro, de forma que se puede saber si un coche entra o sale según el orden en que se activen los sensores (se supone que todos los coches miden más de un metro y que entre un coche y el siguiente habrá más de un metro). Dos coches no se encontrarán nunca de cara en el túnel. Tampoco habrá peatones.



Cuando un coche ha entrado totalmente en el aparcamiento se incrementa el número de coches aparcados, y cuando ha salido totalmente se decrementa. Mientras pasa por el túnel, un coche puede parar o hacer marcha atrás en cualquier momento.

Dibujad el grafo de estados de un circuito secuencial que a partir de las señales a y b , provenientes respectivamente de los sensores A y B (valdrán 1 si hay un coche ante un sensor y 0 si no hay ninguno), genere dos señales de salida *más* o *menos* que gobernarán el contador de coches que hay en el aparcamiento en cada momento.

4.3. Sincronización

Ya sabemos que los circuitos secuenciales están sincronizados por una señal de reloj que describe ciclos periódicos entre los valores 0 y 1.

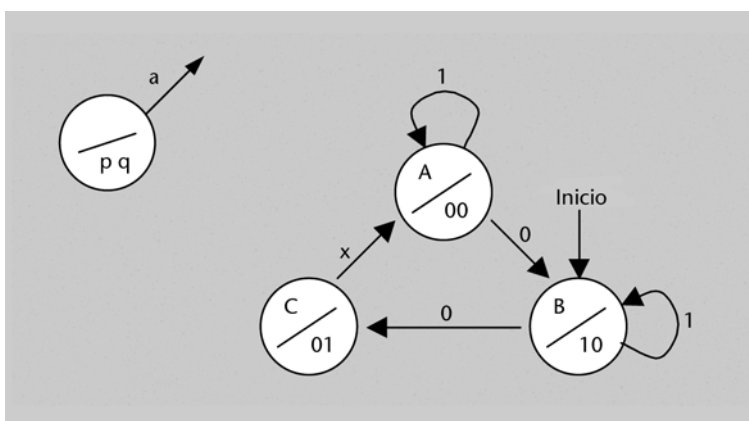
Las transiciones entre estados tienen lugar en cada flanco ascendente de reloj (porque los estados se implementan físicamente mediante biestables).

Observad la implementación de los estados mediante biestables en el subapartado 4.4. de este módulo.

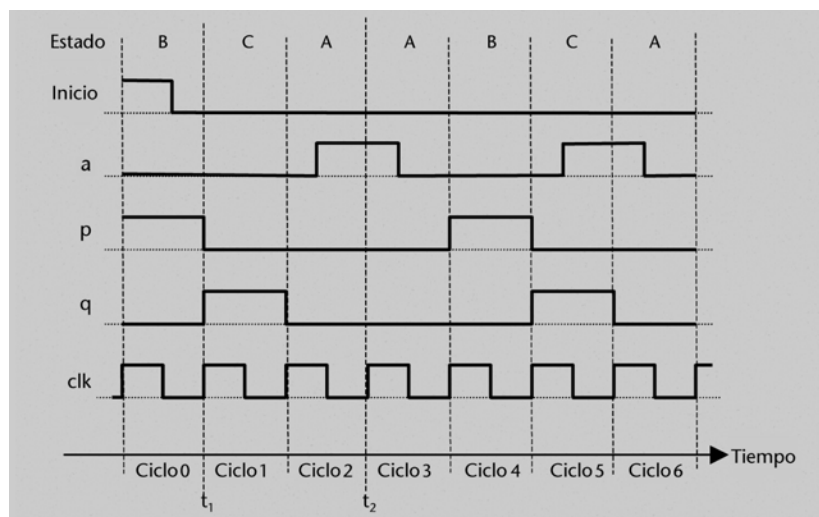
Por tanto, el circuito examina el valor de las señales de entrada en cada ciclo de reloj. En concreto, el valor que hace que se tome una transición u otra es lo que tienen las entradas al llegar al instante del flanco. Si dibujamos las entradas en un cronograma, el valor que decide qué transición se toma es el que tienen al tocar, por la izquierda, la línea vertical correspondiente a un flanco.

Ejemplo de transiciones entre estados

Sea el grafo de estados que se muestra en esta figura:



La siguiente figura muestra cómo evoluciona el circuito con el tiempo a partir de una secuencia de valores determinada en la entrada a . En el ciclo 0, la señal *Inicio* genera un pulso a 1 y hace que el circuito se ponga en el estado B . En el instante t_1 (por la izquierda), la señal de entrada a vale 0, lo que provoca que el circuito pase al estado C en ese instante. Durante el ciclo 2, el circuito se encuentra en el estado A . Dado que el instante t_2 la entrada vale 1, en este momento se produce una transición hacia el mismo estado A . El resto del cronograma se calcula de forma análoga.

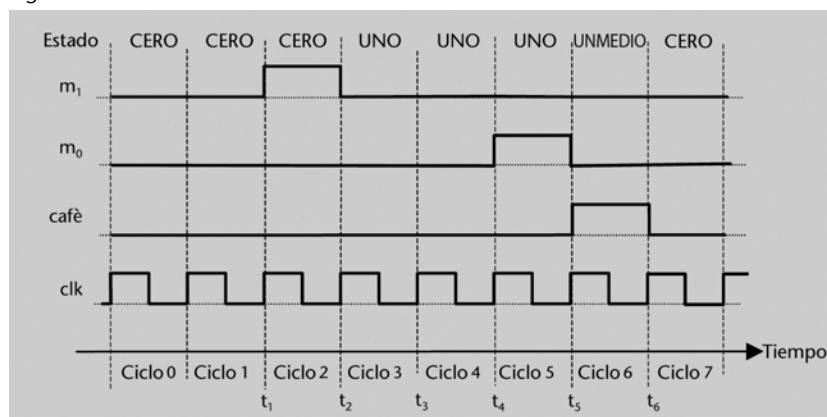


Recordemos que el valor de las señales de salida en cada momento viene determinada por el estado en que se encuentra el circuito.

Retomemos el ejemplo de la máquina de café. Cuando se introduce una moneda, pasa un intervalo de tiempo desde que se introduce en la ranura hasta que cae en la caja correspondiente. Supongamos que el sistema que codifica las señales m_1 y m_0 (recordemos la tabla 1 y la figura 14) genera pulsos de un ciclo de duración: m_0 estará a 1 durante un ciclo cuando se haya introducido una moneda de 0,5 euros, y m_1 estará a 1 durante un ciclo cuando se haya introducido una moneda de 1 euro.

La figura 18 muestra una posible evolución temporal del circuito, partiendo del estado *CERO*. Durante los dos primeros ciclos no se ha introducido ninguna moneda y, por tanto, las transiciones que se han producido han llevado siempre al estado *CERO*. En el ciclo 2, m_1 genera un pulso a 1 e indica que se ha introducido una moneda de 1 euro. Por tanto, el circuito pasa al estado *UNO* en el instante t_2 . Las próximas dos transiciones llevarán también al estado *UNO*, ya que $[m_1 m_0] = [0 0]$ en los instantes t_3 y t_4 . En el ciclo 5, $m_0 = 1$ (se ha introducido una moneda de 0,5 euros) y, por tanto, el circuito pasa al estado *UNMEDIO* en el instante t_5 . Por tanto, durante el ciclo 6, la salida *café* vale 1 (eso hará que se active el dispositivo que sirve un café). Puesto que durante este ciclo las entradas valen 0 (no se ha introducido ninguna moneda), el circuito pasa al estado *CERO* en el instante t_6 y, por tanto, la salida *café* vuelve a 0.

Figura 18



Representación de los ciclos en un cronograma

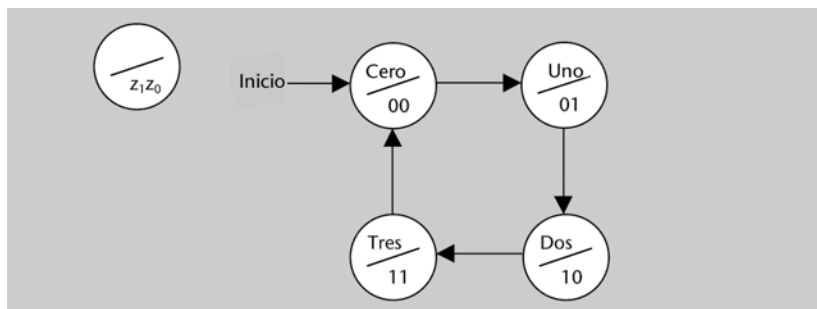
En este cronograma no hemos dibujado la señal *Inicio*. En general, los ciclos que mostramos en un cronograma no tienen por qué ser iniciales, sino que pueden corresponder a un momento cualquiera del funcionamiento del circuito.

Vemos pues que, en los circuitos secuenciales, el tiempo que el circuito esté en cada estado y, por tanto, la duración de los diferentes valores de las señales de salida, vienen determinados por la sincronización.

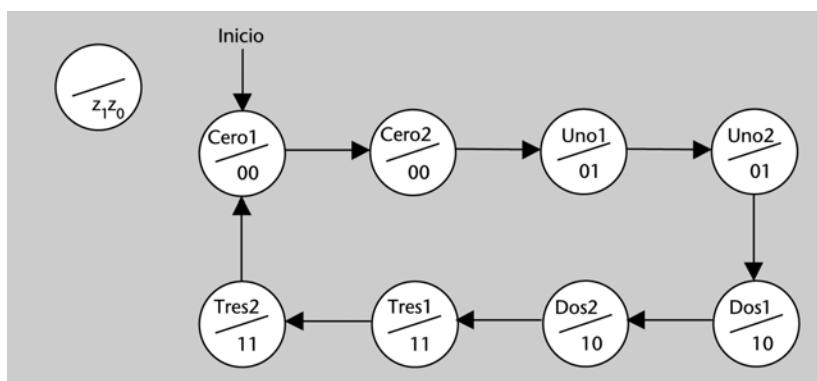
Un contador módulo 4

Queremos diseñar un contador módulo 4 como el de la siguiente figura en el que cada valor de salida dure 100 ns:

Ved el ejemplo de los contadores módulo n en el subapartado 4.2.3



Si disponemos de una señal de reloj con un periodo de 100 ns, entonces el grafo de estados del circuito es el que se muestra en la figura anterior. Sin embargo, si el periodo del reloj es de 50 ns, entonces el grafo tiene que ser el siguiente:

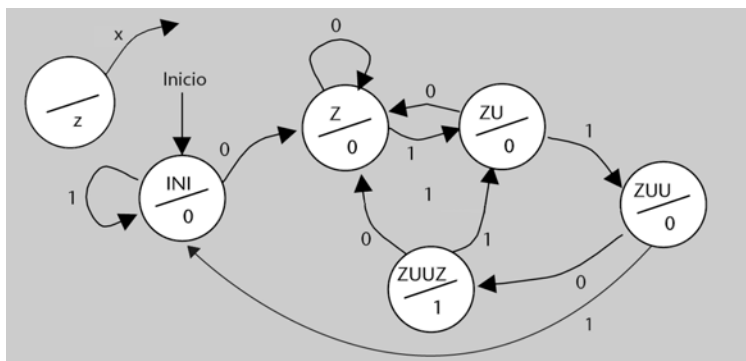


Actividades

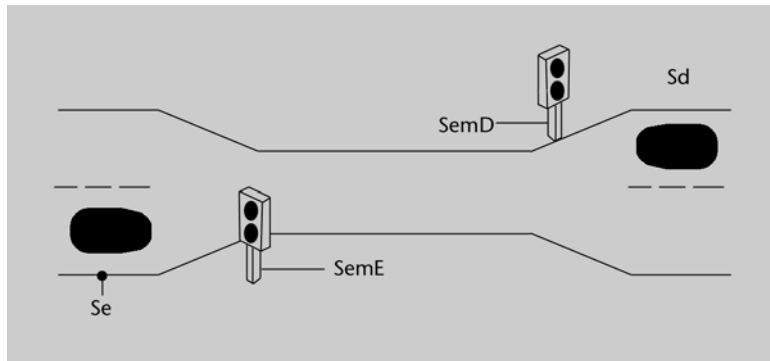
28. Dibujad el grafo de estados de un circuito que reconozca la secuencia 0110 en la entrada x (de un bit). Al reconocerla, la señal de salida z (de un bit) se tiene que poner a 1 durante un ciclo de reloj. Inicialmente la salida tiene que estar a 0.

El circuito no detecta solapamiento entre dos secuencias consecutivas. Es decir, si llegan los valores de entrada 0110110, la salida sólo se pondrá a uno después de los cuatro primeros valores.

29. El grafo de estados siguiente corresponde a un circuito que reconoce una secuencia determinada de valores en la señal de entrada x y pone la salida z a 1 durante un ciclo cuando se ha producido. ¿Cuál es esta secuencia? Describid con detalle los casos en que se produce el reconocimiento.



30. Dibujad el grafo de estados de un circuito secuencial que controle los semáforos de un puente en el que no hay suficiente espacio para que circulen simultáneamente coches en ambos sentidos.



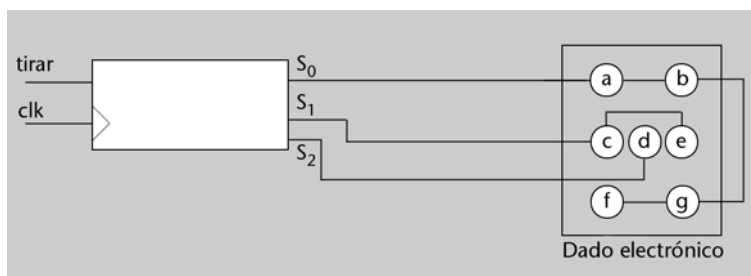
En cada extremo del puente hay:

- un sensor que pone la señal S_d (o S_e) a 1 cuando hay un coche delante.
- un semáforo que se pone en rojo cuando le llega un 0 por la señal $SemD$ (o $SemE$) y se pone en verde cuando le llega un 1.

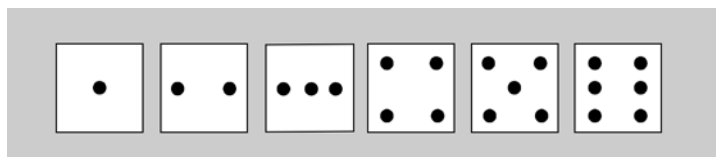
El circuito recibe como entrada las señales S_d y S_e y genera como salida las señales $SemD$ y $SemE$. Debe funcionar de la siguiente forma:

- Los semáforos se ponen en verde de forma alternada durante dos ciclos de reloj como mínimo.
- Después del segundo ciclo de reloj, el semáforo que está en verde permanece así hasta que llegue un coche por el otro extremo.
- Al ponerse el sistema en funcionamiento, el semáforo de la izquierda tiene que estar en verde durante dos ciclos.

31. Se quiere diseñar un circuito secuencial para tirar un dado. La entrada de la señal *tirar*, que está conectada a un pulsador que manipula el jugador (pulsado: *tirar* = 1; no pulsado: *tirar* = 0). Las salidas del circuito estarán conectadas a los puntos de un “dado electrónico” de la siguiente forma:



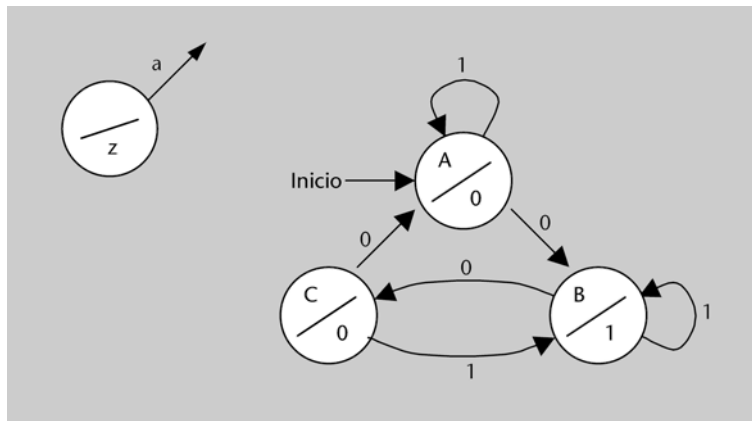
Los puntos se iluminan cuando les llega un 1. Las combinaciones posibles del dado son las siguientes:



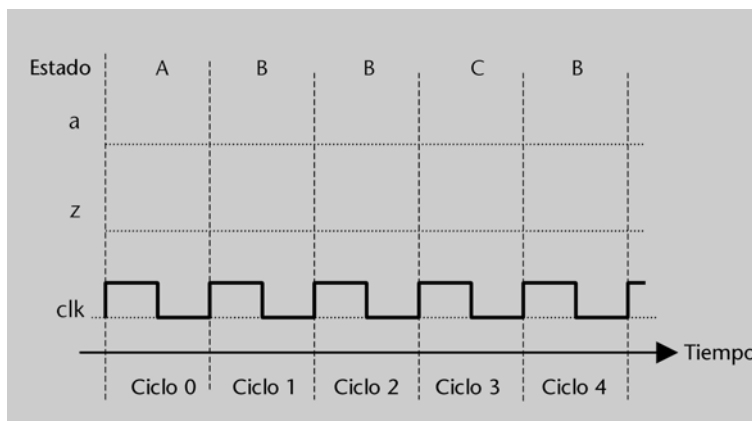
Cuando el dado empiece a funcionar, tiene que mostrar un “1”. A partir del momento en que el jugador pulse el pulsador y mientras lo mantenga pulsado, el circuito generará cíclicamente y en orden las seis combinaciones del dado. Cuando el jugador libere el pulsador, no se producirá ninguna transición, y verá la combinación que ha salido en el dado (se supone que la frecuencia del reloj es muy alta y no se pueden llegar a distinguir las combinaciones intermedias).

Dibujad el grafo de estados del circuito.

32. Sea el siguiente grado de estados:



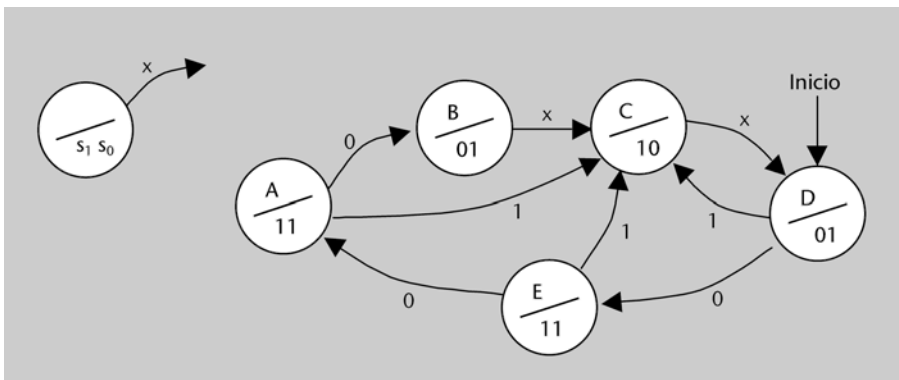
Completad el siguiente cronograma. Suponed que la señal de entrada a sólo cambia de valor en los instantes de los flancos.



4.4. Implementación

Veamos cómo se puede implementar físicamente un circuito secuencial descrito de acuerdo con el modelo de Moore. Tomemos como ejemplo el circuito descrito por el grafo de la figura siguiente:

Figura 19



A continuación se muestran las tablas de transiciones y de salidas correspondientes a este grafo:

Tabla de transiciones		
Estado	x	Estado ⁺
A	0	B
A	1	C
B	0	C
B	1	C
C	0	D
C	1	D
D	0	E
D	1	C
E	0	A
E	1	C

Tabla de salidas		
Estado	s ₁	s ₀
A	1	1
B	0	1
C	1	0
D	0	1
E	1	1

Estas dos tablas se pueden convertir fácilmente en tablas de la verdad de funciones lógicas si codificamos los estados mediante variables lógicas. En concreto, si hay n estados, serán necesarios $\lceil \log_2 n \rceil$ variables para codificarlos. En nuestro ejemplo, ésta es una posible codificación de estados:

Estado	q ₂	q ₁	q ₀
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

Una vez codificados los estados, las transiciones y salidas del circuito son funciones lógicas que se describen con las siguientes tablas de la verdad:

Tabla de transiciones							
Estado				Estado ⁺			
q ₂	q ₁	q ₀	x	q ₂ ⁺	q ₁ ⁺	q ₀ ⁺	
0	0	0	0	0	0	1	
0	0	0	1	0	1	0	
0	0	1	0	0	1	0	
0	0	1	1	0	1	0	
0	1	0	0	0	1	1	
0	1	0	1	0	1	1	
0	1	1	0	1	0	0	
0	1	1	1	0	1	0	
1	0	0	0	0	0	0	
1	0	0	1	0	1	0	
1	0	1	0	x	x	x	
1	0	1	1	x	x	x	
1	1	0	0	x	x	x	
1	1	0	1	x	x	x	
1	1	1	0	x	x	x	
1	1	1	1	x	x	x	

Tabla de salidas					
Estado					
q ₂	q ₁	q ₀	s ₁	s ₀	
0	0	0	1	1	
0	0	1	0	1	
0	1	0	1	0	
0	1	1	0	1	
1	0	0	1	1	
1	0	1	x	x	
1	1	0	x	x	
1	1	1	x	x	

Las variables que codifican los estados, q_i , se guardan en biestables. De esta forma, el circuito guarda en todo momento la memoria del estado en que se encuentra. En nuestro ejemplo, serán necesarios tres biestables; cuando éstos valgan, por ejemplo, $[q_2 q_1 q_0] = [0 1 0]$, sabremos que el circuito se encuentra en el estado C.

Las señales de salida s_1 y s_0 se pueden implementar como funciones lógicas de q_2 , q_1 y q_0 , a partir de la tabla de verdad anterior, de cualquiera de las formas que conocéis.

Podéis ver varias formas de implementar funciones lógicas en el módulo "Los circuitos lógicos combinacionales" de esta asignatura.



Por lo que se refiere a las transiciones, las columnas q_2^+ , q_1^+ y q_0^+ nos indican los valores que deben tomar los biestables en el siguiente flanco de reloj. Dado que un biestable toma el valor que hay en su entrada D , sabemos que en las entradas de los biestables tenemos que poner, para cada una de las combinaciones posibles de estados y entradas, lo que muestra la siguiente tabla (en la que d_i corresponde a la entrada D de cada uno de los biestables):

q_2	q_1	q_0	x	d_2	d_1	d_0
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	0
0	1	1	1	0	1	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0
1	0	1	0	x	x	x
1	0	1	1	x	x	x
1	1	0	0	x	x	x
1	1	0	1	x	x	x
1	1	1	0	x	x	x
1	1	1	1	x	x	x

Esta tabla se llama **tabla de excitaciones**, ya que nos indica cómo hay que "excitar" los biestables para que tengan lugar las transiciones adecuadas. Fijémonos en que las columnas d_i de la tabla de excitaciones coinciden con las columna q_i^+ de la tabla de transiciones.

Las señales d_2 , d_1 y d_0 , llamadas **funciones de excitación**, son funciones lógicas de q_2 , q_1 y q_0 , y de la entrada x . Podemos implementarlas, pues, por cualquiera de los métodos que conocemos.

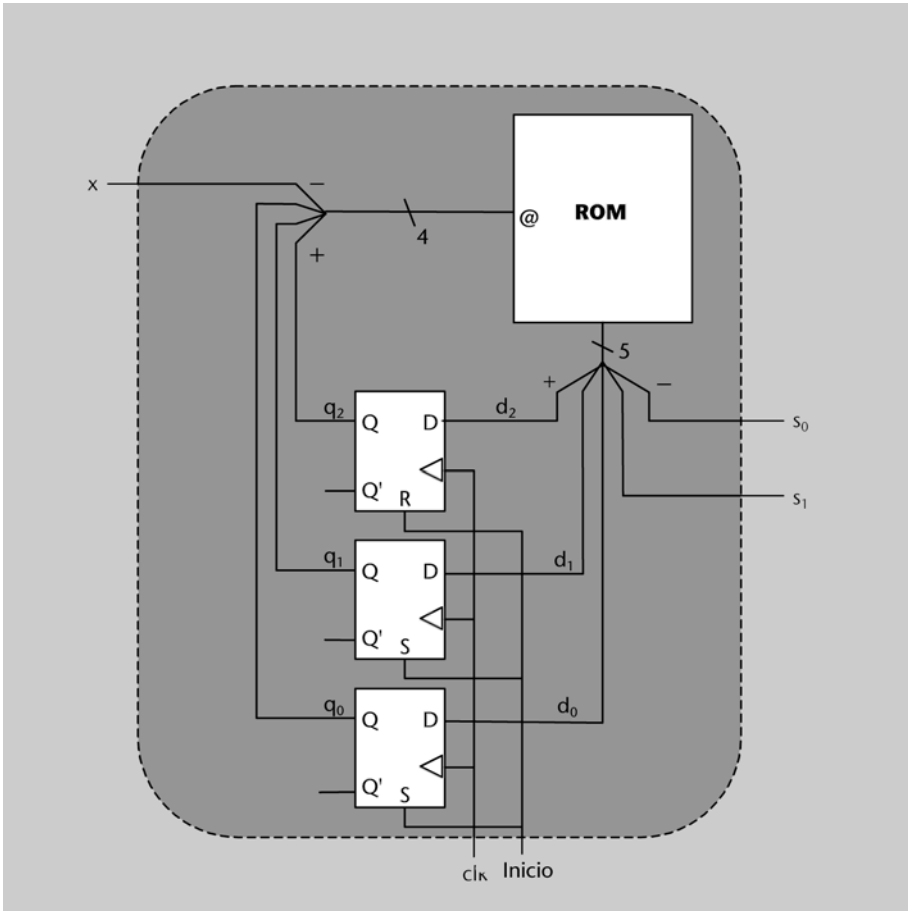
Podemos escribir una sola tabla de verdad que incluya las funciones de excitación y las de salida, tal como se muestra en la tabla siguiente. Fijémonos en que, por el hecho de que las funciones de salida dependen sólo del estado (va-

riables q_i), su valor es el mismo en las filas correspondientes a una misma combinación de q_i , y a diferentes valores de la entrada x .

q_2	q_1	q_0	x	d_2	d_1	d_0	s_1	s_0
0	0	0	0	0	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	1	0	0	1	0	0	1
0	0	1	1	0	1	0	0	1
0	1	0	0	0	1	1	1	0
0	1	0	1	0	1	1	1	0
0	1	1	0	1	0	0	0	1
0	1	1	1	0	1	0	0	1
1	0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	1	1
1	0	1	0	x	x	x	x	x
1	0	1	1	x	x	x	x	x
1	1	0	0	x	x	x	x	x
1	1	0	1	x	x	x	x	x
1	1	1	0	x	x	x	x	x
1	1	1	1	x	x	x	x	x

Una vez se ha expresado el comportamiento del circuito mediante esta única tabla de verdad, se puede implementar de forma muy sencilla utilizando una memoria ROM, tal como se muestra en la figura que presentamos a continuación:

Figura 20



Nota

En este circuito los biestables están dibujados con las entradas a la derecha y las salidas a la izquierda.

La memoria ROM de la figura 20 se podría sustituir por cualquier otra forma de implementación de funciones lógicas.

El contenido de la memoria ROM, que corresponde a la tabla anterior, es el siguiente:

Dirección	d_2	d_1	d_0	s_1	s_0
0	0	0	1	1	1
1	0	1	0	1	1
2	0	1	0	0	1
3	0	1	0	0	1
4	0	1	1	1	0
5	0	1	1	1	0
6	1	0	0	0	1
7	0	1	0	0	1
8	0	0	0	1	1
9	0	1	0	1	1
10	x	x	x	x	x
11	x	x	x	x	x
12	x	x	x	x	x
13	x	x	x	x	x
14	x	x	x	x	x
15	x	x	x	x	x

En la figura 20 también puede verse cómo se lleva a cabo la inicialización del circuito, conectando la señal *Inicio* a las entradas asíncronas de los biestables (recordemos que la señal *Inicio* genera un pulso a 1 para indicar al circuito que se ponga en funcionamiento). Dado que en nuestro ejemplo el estado inicial es el *D* (podéis ver la figura 19), al empezar a funcionar el circuito, los biestables toman los valores $[q_2, q_1, q_0] = [0\ 1\ 1]$. El circuito volverá a este estado siempre que *Inicio* haga un pulso.

Resumen

En este módulo se han estudiado los circuitos lógicos secuenciales. Se ha visto que lo que los caracteriza es su capacidad de memoria y, por tanto, son capaces de determinar el valor de las señales de salida de acuerdo no sólo con el valor actual de las señales de entrada, sino también con el valor que han tenido estas señales de entrada en anteriores momentos.

Se ha visto la necesidad de un mecanismo de sincronización para controlar la evolución temporal de las distintas señales, y se ha presentado la señal de reloj.

Se ha conocido el dispositivo más elemental de memoria, el biestable D, que es capaz de guardar el valor de un bit. Se ha visto que su valor se puede modificar de forma síncrona y asíncrona, gracias a las entradas *R* y *S*. También se ha visto que se puede “congelar” el valor mediante una señal de carga.

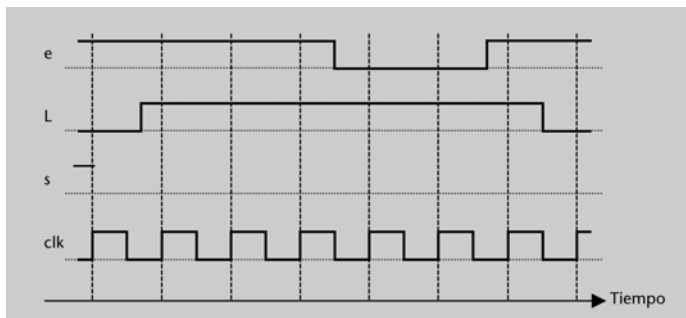
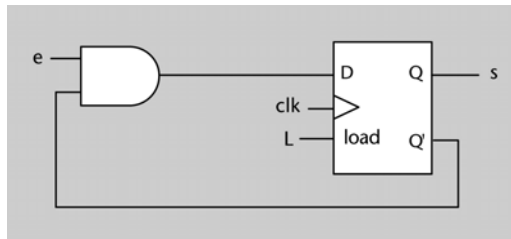
Después se han presentado los diferentes bloques secuenciales que permiten guardar el valor de una palabra (el registro), de un número pequeño de palabras (el banco de registros) o de un gran volumen de palabras (la memoria RAM).

Por último, se ha conocido la forma de especificar el comportamiento de circuitos secuenciales llamada *modelo de Moore*, que se fundamenta sobre los conceptos de *estado* y *transición*. Se ha visto que el comportamiento del circuito se puede expresar mediante las tablas de salidas y transiciones, o bien gráficamente mediante grafos de estados. Se ha aprendido a dibujar la evolución temporal de un circuito sobre un cronograma.

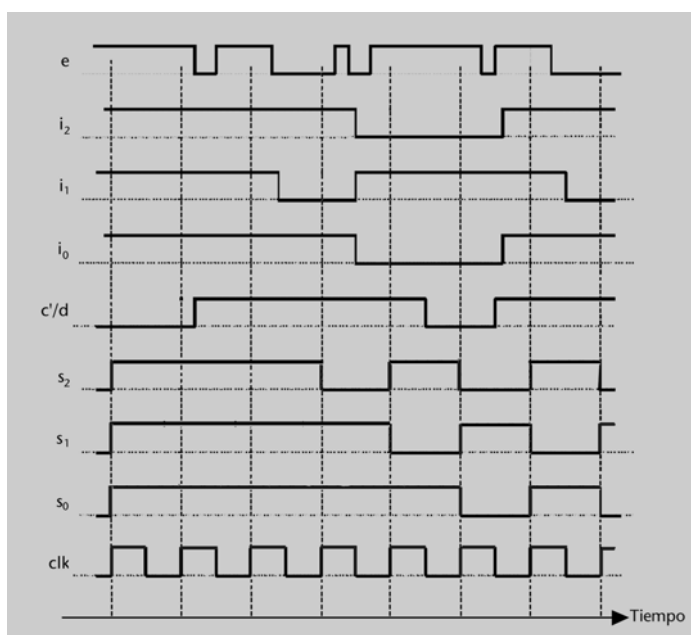
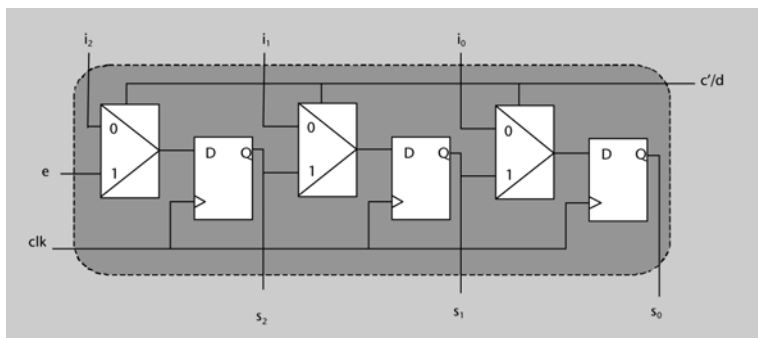
Los bloques secuenciales y combinacionales que se han estudiado en este curso constituyen un conjunto de dispositivos suficiente para diseñar un computador sencillo.

Ejercicios de autoevaluación

1. Completad el cronograma que corresponde al circuito de la figura, suponiendo que inicialmente la salida Q vale 1. ¿Cuál es el papel de la señal e en el circuito?



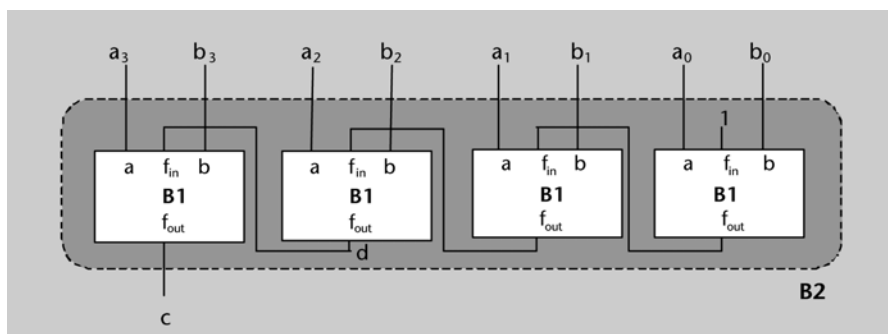
2. Completad el cronograma que corresponde al circuito de la figura, suponiendo que inicialmente las salidas Q de todos los biestables valen 0. Describid en pocas palabras qué hace el circuito según la señal c'/d .



3. El circuito combinacional B2 es un comparador de dos números naturales A y B representados en binario. Su funcionamiento es el siguiente:

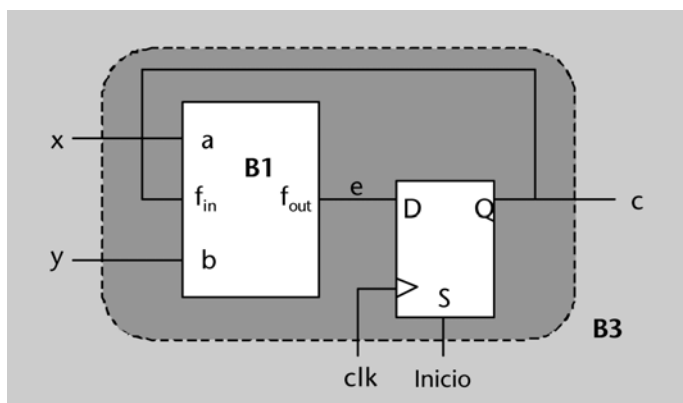
$c = 0$ si A es menor que B .

$c = 1$ si A es mayor o igual que B .

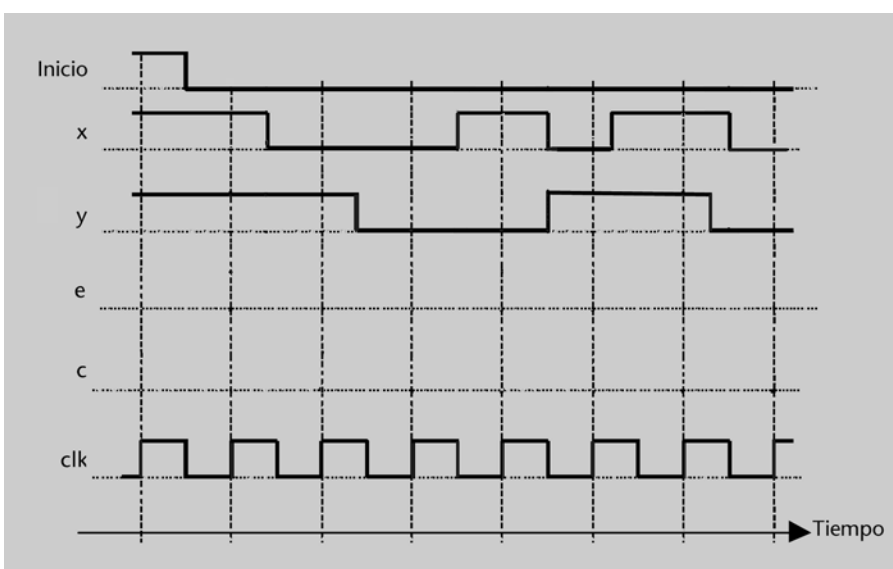


a) Viendo cómo se ha construido el circuito B2 y su funcionalidad, deducid la tabla de verdad del bloque B1.

b) Utilizando el mismo bloque B1 se ha construido este otro circuito, B3. Describid qué función hace este circuito, y comparadlo con el circuito B2.



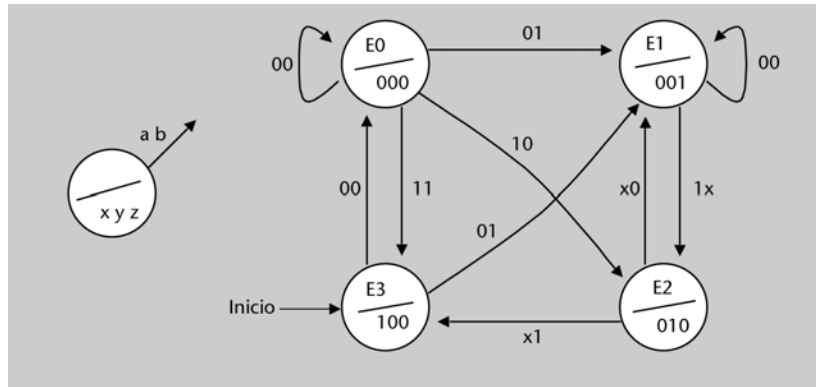
c) Completad el siguiente cronograma, que corresponde al circuito B3. Si interpretamos las entradas x e y en cada ciclo de reloj como los distintos bits de un par de números A y B , ¿qué números son A y B ?



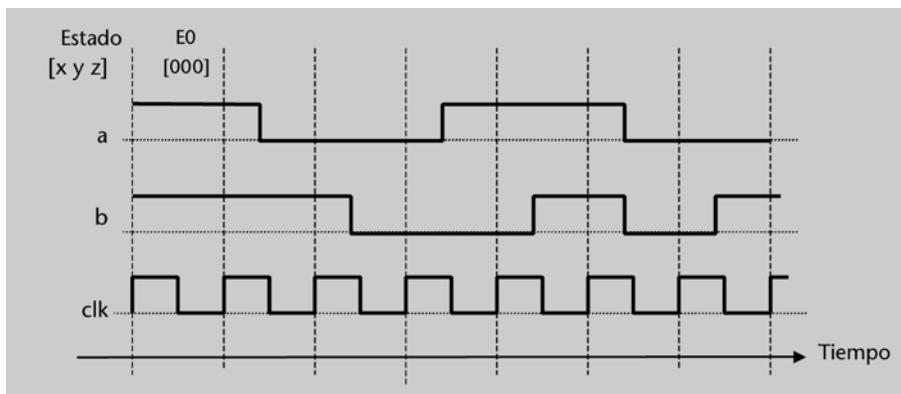
4. Se quiere diseñar un circuito secuencial que controle un semáforo. De día (de 8:00 a 20:00), el semáforo tiene que estar en verde durante tres ciclos, en amarillo durante un ciclo y en rojo durante dos ciclos. Por la noche, tiene que estar en verde durante dos ciclos, en amarillo durante un ciclo y en rojo durante tres.

- a) ¿Qué entradas y salidas debe tener el circuito?
 b) Expresad el comportamiento del circuito mediante un grafo de estados.

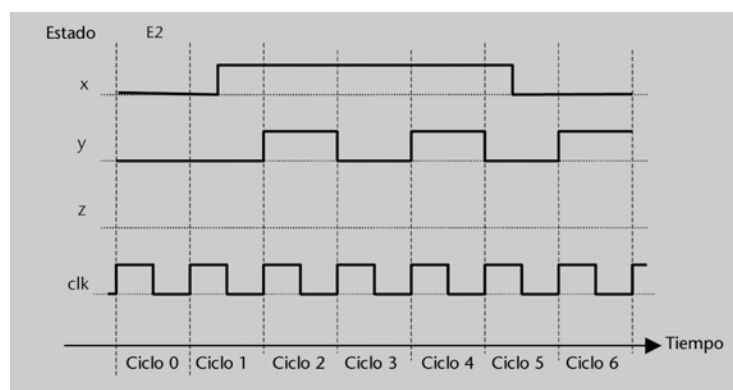
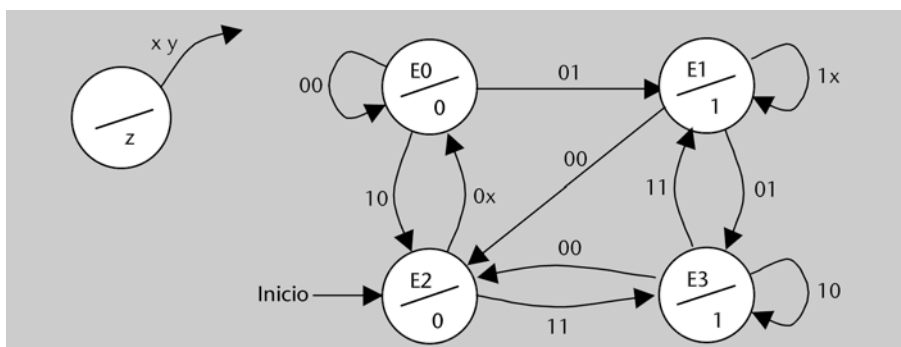
5. Dado el siguiente grafo de estados:



- a) Escribid la tabla de salidas y la tabla de transiciones del circuito.
 b) Completad las líneas correspondientes al estado y a las salidas del siguiente cronograma:



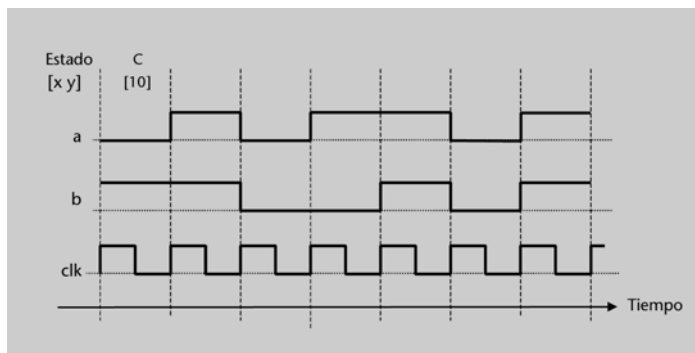
6. Completad el siguiente cronograma, que corresponde a un circuito con este grafo de estados:



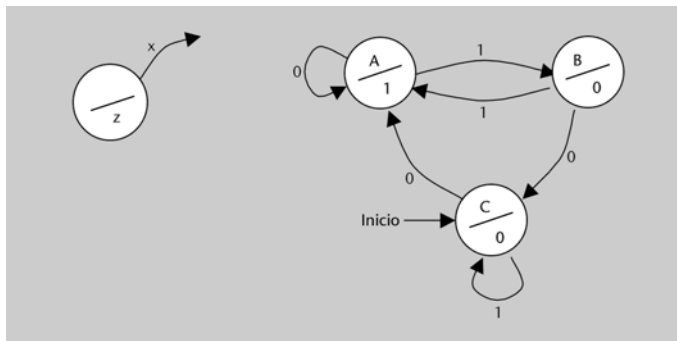
7. A continuación se muestran las tablas de salidas y las de transiciones de un circuito secuencial.

Tabla de salidas		Estado	a	b	Estado ⁺
Estado	xy				
A	0 0	A	0	0	B
B	0 1	A	0	1	A
C	1 1	A	1	0	B
		A	1	1	A
		B	0	0	x
		B	0	1	x
		B	1	0	C
		B	1	1	C
		C	0	0	C
		C	0	1	B
		C	1	0	A
		C	1	1	A

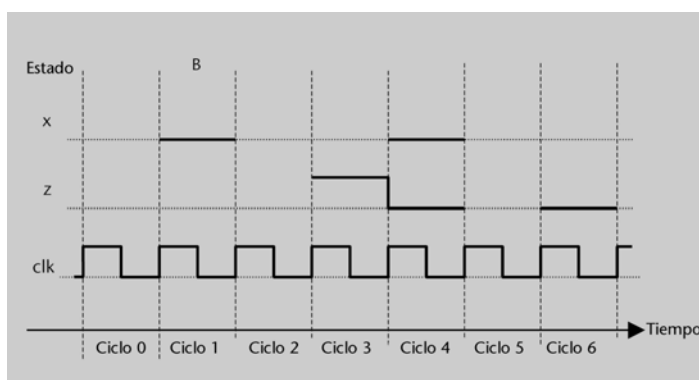
- a) Dibujad el grafo de estados del circuito, suponiendo que el estado inicial es el A.
 b) Completad las líneas correspondientes al estado y a las salidas del siguiente cronograma:



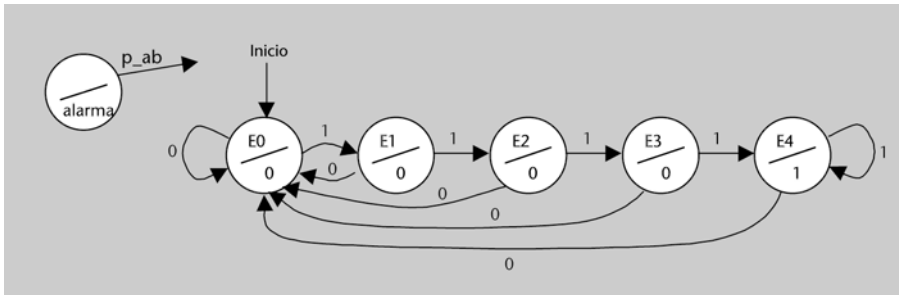
8. Sea el siguiente grafo de estados:



Completad el siguiente cronograma. Suponed que la señal de entrada x sólo cambia de valor en los instantes de los flancos.



9. Un circuito secuencial tiene una entrada p_{ab} y una salida *alarma*, ambas de un bit. La entrada p_{ab} proviene de la caja fuerte de un banco, y vale 1 cuando está abierta y 0 cuando está cerrada. La salida *alarma* está conectada a una señal de alarma, que se activa cuando es $alarma = 1$. El funcionamiento del circuito viene descrito por este grafo de estados:



Si sabemos que el reloj del circuito tiene un periodo de 30 segundos, describid el comportamiento del sistema de alarma de la caja fuerte. Suponed que la puerta de la caja sólo se puede abrir o cerrar cada 30 segundos, coincidiendo con los flancos ascendentes del reloj.

Solucionario

Actividades

1. Se trata de reconocer si los cuatro bits que llegan por la entrada del sistema valen 1010 o no. Para saberlo es suficiente examinar el valor de la palabra de entrada en el momento actual. Por tanto, el sistema es de tipo combinacional.

En cambio, si la entrada de un circuito fuese de un bit, el circuito debería ser de tipo secuencial, porque en cada momento debería recordar los tres últimos valores que han llegado por la entrada, aparte del actual.

2. Un dígito decimal (rango 0 al 9) requiere cuatro bits para ser codificado. Por tanto, la entrada del sistema que se tiene que diseñar puede leer un dígito en cada momento.

Dado que se tiene que detectar una secuencia de cuatro dígitos, éstos tienen que entrar uno tras otro por la entrada del sistema, y el circuito tiene que recordar los dígitos que han entrado con anterioridad para reconocer la secuencia.

Se trata, pues, claramente, de un sistema secuencial.

3. El periodo T es el inverso de la frecuencia $T = 1/F$.

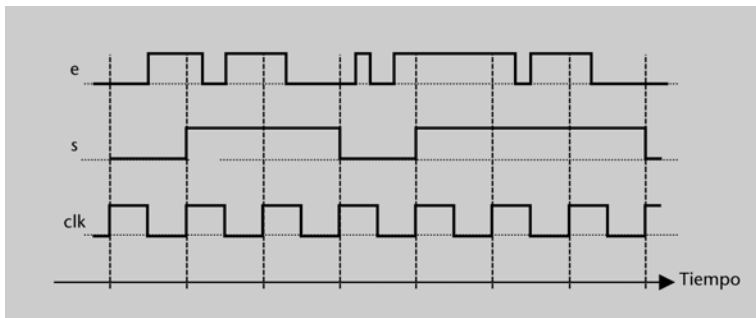
Puesto que la frecuencia es de $1,6 \cdot 10^9$ Hz, el periodo es el siguiente:

$$T = 1/(1,6 \cdot 10^9) \text{ s} = 0,625 \cdot 10^{-9} \text{ s} = 0,625 \text{ ns (nanosegundos)}$$

4. Como se puede apreciar en el cronograma, en la salida s del biestable está el valor leído de la entrada e en cada flanco ascendente del reloj.

Fijémonos en que la salida del biestable sólo cambia en los flancos ascendentes del reloj (es decir, cambia de forma síncrona), mientras que la entrada puede cambiar en cualquier momento (es decir, cambia de forma asíncrona).

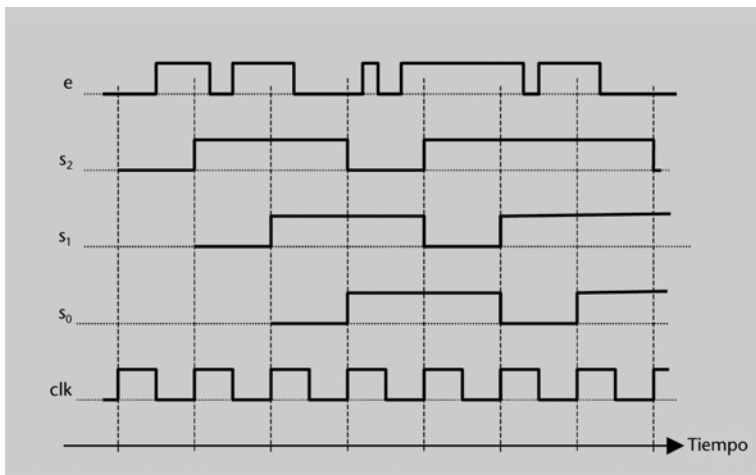
Es importante notar que el valor de s no se conoce antes del primer flanco de reloj, porque no se conoce el valor que había en la entrada D del biestable en el instante del flanco anterior. Por esta razón, s no tiene valor.



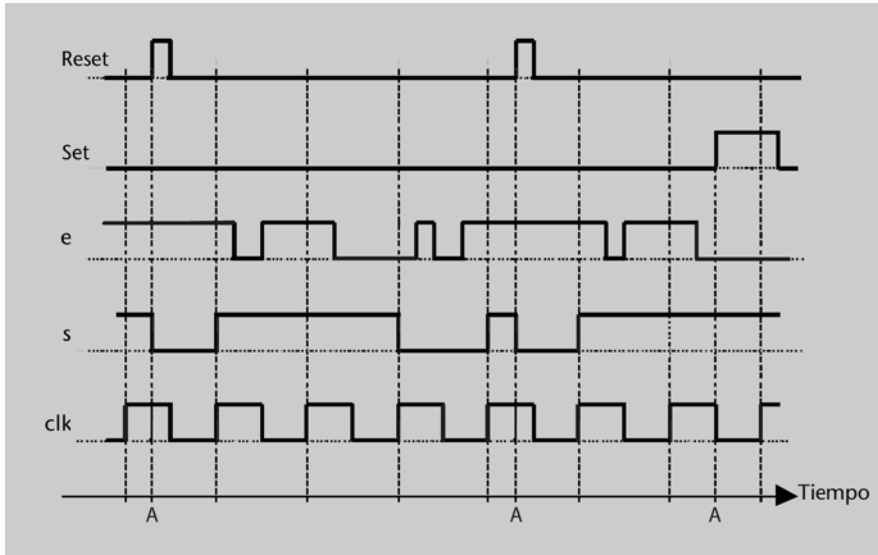
5. En este caso se puede ver que la entrada e se “desplaza” de forma sucesiva por las salidas s_2 , s_1 y s_0 de los biestables.

La mejor forma de hacer este tipo de cronogramas es dibujar primero la línea correspondiente a la señal s_2 , que sólo depende de e , entera. Una vez ésta se ha dibujado, se dibuja la señal s_1 , que depende sólo de la señal s_2 . Por último, se dibuja la línea correspondiente a la señal s_0 , que depende sólo de la señal s_1 .

Notamos que el valor de s_1 no se puede determinar hasta el segundo flanco del reloj, y el de s_0 hasta el tercero.



6. Para resolver este ejercicio debemos tener en cuenta que las entradas asíncronas tienen prioridad sobre las síncronas, es decir, primero se evalúan los valores de las entradas R y S de los biestables, y sólo cuando ambas están a 0 se evalúa el valor presente en la entrada D en cada flanco de reloj (dado que el biestable no tiene señal de carga, asumimos que está a 1). Por esta razón, en el cronograma aparecen marcados con líneas verticales con la letra A los momentos en que cambian las entradas asíncronas, además de los flancos ascendentes del reloj.



7. A la entrada D del biestable llega o bien la salida del biestable (cuando L vale 0), o bien la señal e (cuando L vale 1).

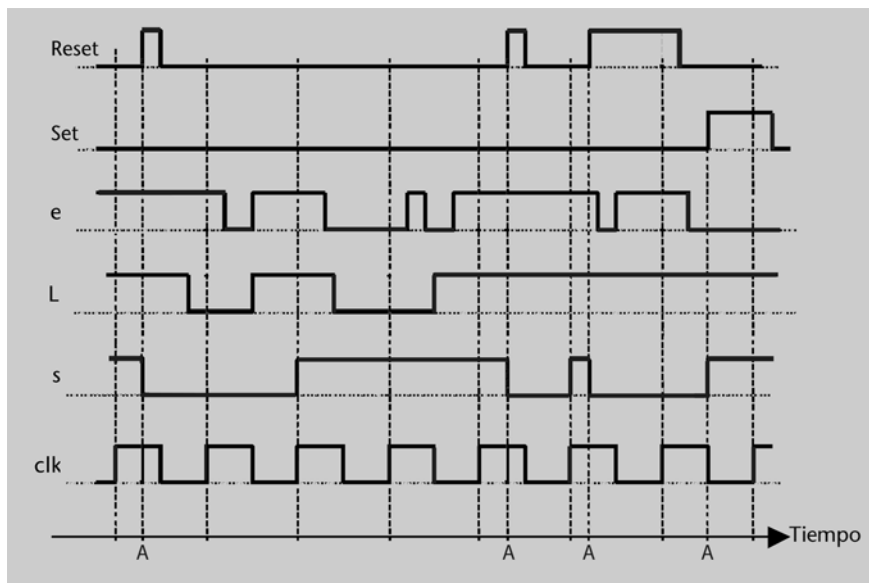
Dado que el biestable se carga en cada flanco de reloj, tenemos que cuando $L = 0$ se carga lo mismo que había. El efecto es que su salida Q no cambia.

Por el contrario, cuando $L = 1$, se almacena en el biestable el valor de la señal e .

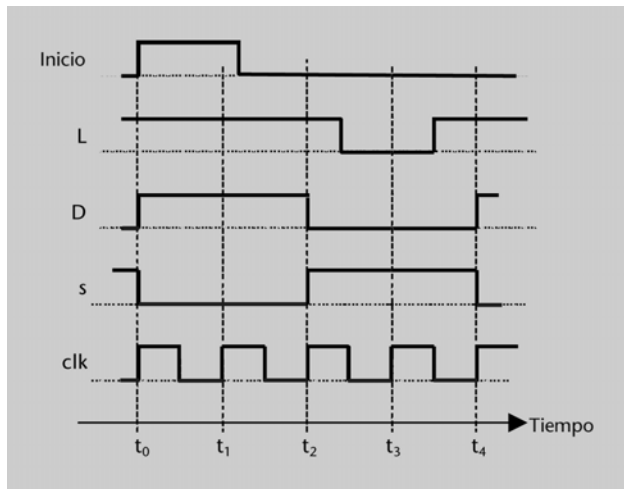
Por tanto, podemos decir que este circuito se comporta igual que un biestable con señal de carga, en el que la señal L hace el papel de la entrada *load*.

Fijémonos en que después del último flanco el biestable se mantiene a 1, aunque $L = 1$ y $e = 0$. Esto se debe al hecho de que *Set* está a 1.

Como en la actividad 6, en el cronograma aparecen marcados los momentos en que cambian las entradas asíncronas con la letra A además de los flancos de subida del reloj.

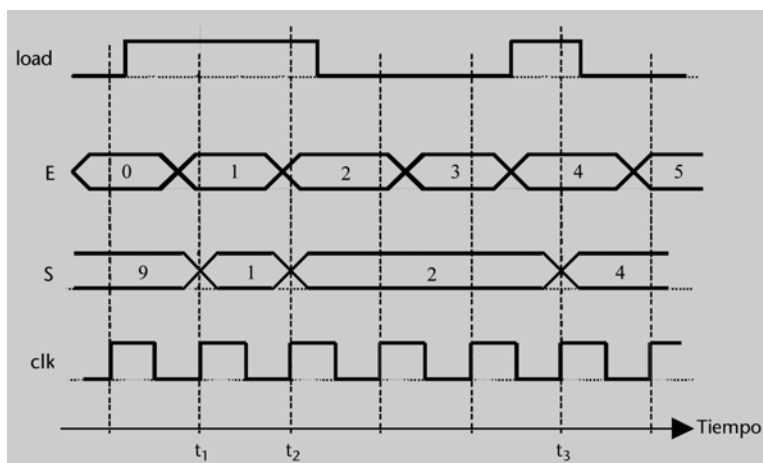


8. Inicialmente, $D = 0$, porque $s = 1$. En el instante t_0 , *Inicio* se pone a 1, por tanto, el biestable se pone a 0. Se quedará al menos hasta t_2 , porque *Inicio* continúa valiendo 1. En t_2 se carga con lo que hay en la entrada D , es decir, un 1. En t_3 , el biestable no se carga, porque $L = 0$. En t_4 se carga con lo que hay en la entrada D , es decir, un 0.



9. Podemos observar en el cronograma que el registro sólo se carga en los flancos en los que $load = 1$: t_1 , t_2 y t_3 . En estos instantes, se carga con el valor de la entrada E . La secuencia de valores que toma la salida del registro, tal como se muestra en el cronograma, es la siguiente:

9 1 2 4.



10.

a) En este circuito, la salida del sumador se conecta directamente a una de sus entradas. Esto implica que S nunca será estable, continuamente estará variando.

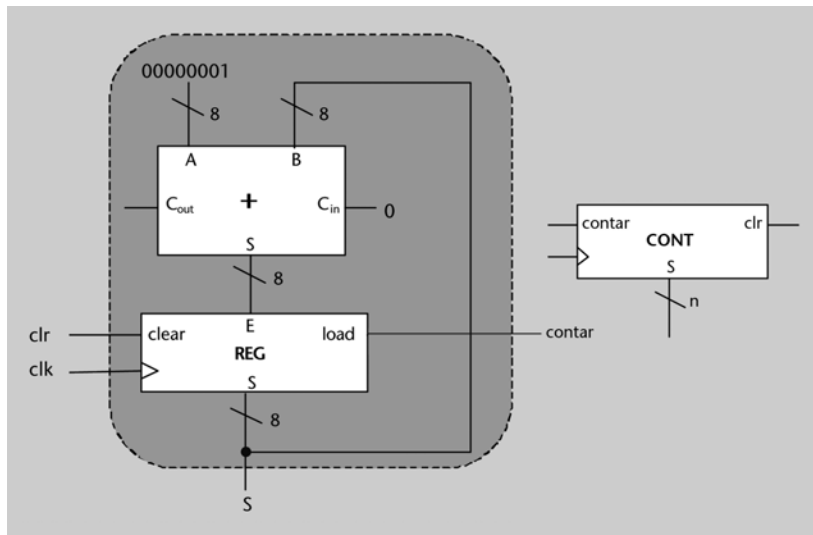
b) En este caso, S expresa correctamente la suma acumulada de los valores que ha tomado X (en los instantes anteriores a cada flanco, tal y como indica la convención que se ha establecido en estos apuntes) desde la inicialización del circuito. Sin embargo, en el momento en el que esta suma acumulada sea mayor que 255 el valor de S dejará de ser correcto, porque no basta con 8 bits para representarlo. Podríamos pensar en hacer que el registro sea de más de 8 bits, pero sea cual sea el número de bits que tenga, siempre llegará un momento en el que se producirá desbordamiento (a no ser que *Inicio* haga un nuevo pulso a 1 antes de producirse el desbordamiento, pero que esto pase o no es incontrolable).

11. Usaremos un registro que contendrá en todo momento la salida S del circuito. Siempre que *contar* valga 1, su contenido se debe incrementar en una unidad en cada flanco del reloj, por lo cual conectaremos la salida del registro a la entrada de un sumador. A la otra entrada del sumador, conectaremos un 1. La salida del sumador está conectada a la entrada del registro para que se guarde en el próximo flanco de reloj si *contar* vale 1. Por lo tanto, conectaremos *contar* a la entrada *load* del registro.

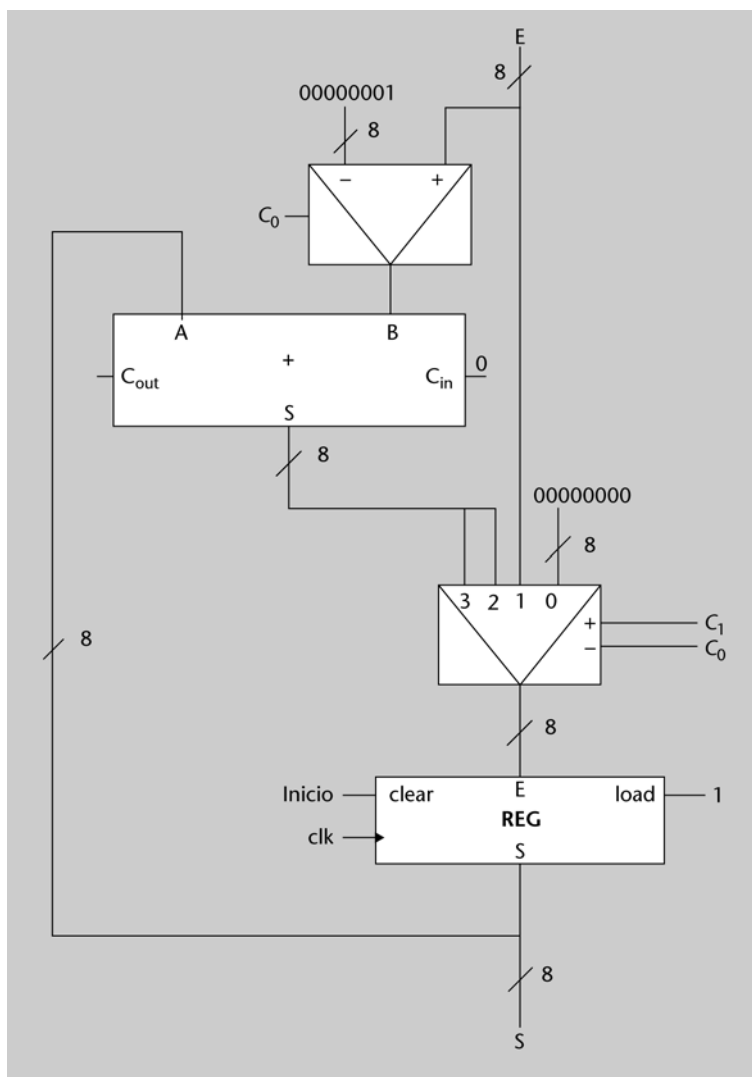
Otra posibilidad es poner un 0 a la otra entrada del sumador y sumar el 1 por medio de la entrada de transporte del sumador, que en el diseño mostrado tiene un 0.

Por otro lado, la señal *clr* estará conectada a la entrada *clear* del registro para ponerlo a 0 de manera asíncrona.

La figura muestra el circuito que conforma el bloque, y el dibujo del bloque con sus entradas y salidas.



12. El valor de S se debe actualizar en cada ciclo, y por lo tanto tiene que estar guardado en un registro, a cuya entrada *clear* conectamos la señal *Inicio* y a cuya entrada *load* conectamos un 1. Para determinar qué valor se cargará a cada flanco ascendente, usaremos un multiplexor 4-1 gobernado por las señales de control c_1 y c_0 . Cuando $c_1 = 1$, el valor que se debe cargar al registro es su contenido actual sumado con 1 (si $c_0 = 0$) o E (si $c_0 = 1$). Esto lo conseguiremos con un sumador, a una de cuyas entradas conectamos S y a la otra conectamos la salida de un multiplexor 2-1 gobernado por c_0 .



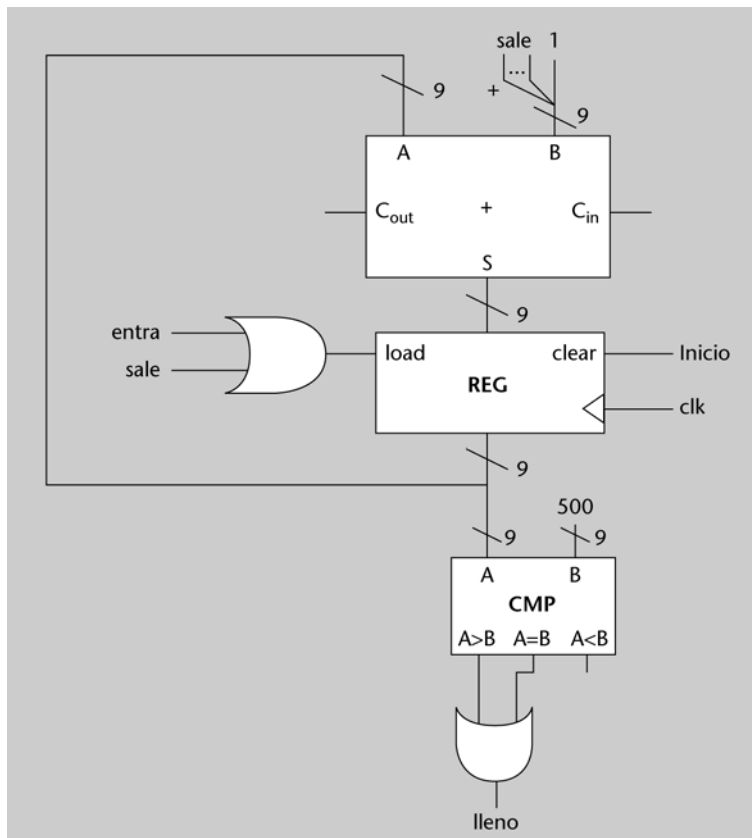
13.

a) El circuito debe saber cuándo entra o sale un coche, y por lo tanto *entra* y *sale* deben ser señales de entrada, así como también la señal *Inicio*. Y su misión es dar en todo momento el valor correcto a la señal *lleno*, que será su señal de salida. Todas las señales son de un bit.

b) El circuito debe ser secuencial porque debe saber en todo momento cuántos coches hay en el aparcamiento, para poder determinar si ya está lleno o no. Por lo tanto, debe tener memoria de cuántos han entrado y salido desde la inicialización y hasta el momento actual.

c) El circuito debe tener un registro que guarde el número de vehículos que hay en el aparcamiento. Puesto que puede haber 500 como mucho, el registro debe tener 9 bits. Conectaremos *Inicio* a su entrada *clear*.

Este registro se debe incrementar o decrementar en una unidad siempre que se produzca un pulso en la señal *entra* o *sale*, respectivamente. Por lo tanto, la entrada *load* debe estar a 1 cuando alguna de estas dos señales esté a 1. Habrá que disponer de un sumador que sume el contenido del registro más 1 o -1 dependiendo del caso. La suma en binario y en complemento a 2 se hacen de la misma manera (es decir, el resultado de una suma será correcto tanto si interpretamos las entradas y salida del sumador en binario como en complemento a 2). Por lo tanto, en caso de que queramos sumar 1 podemos poner a la entrada del sumador 000000001, y en caso de que queramos sumar -1 podemos poner 111111111 (que es -1 codificado en complemento a 2). Teniendo en cuenta que siempre que *sale* valga 1 *entra* valdrá 0, podemos conseguir el valor deseado en la entrada *B* del sumador tal y como se muestra en el circuito: el bit de menos peso vale siempre 1 (recordemos que el registro sólo se cargará si *entra* o *sale* valen 1), y los otros 8 bits se forman replicando 8 veces la señal *sale*.



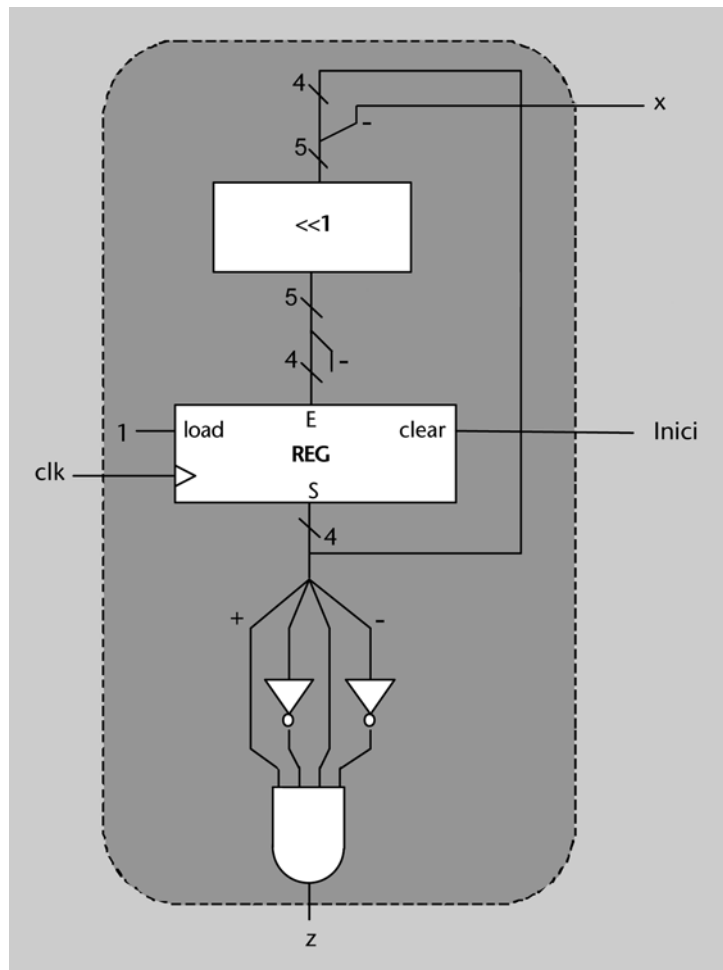
14. Queremos reconocer si en la entrada x de un bit se produce la secuencia de valores 1010, y disponemos de un registro de cuatro bits. Por tanto, tenemos que hacer que los valores que lleguen por la entrada x se desplacen por los biestables del registro, por ejemplo, de derecha a izquierda. El biestable que contiene el bit de menos peso se cargará en cada flanco con el valor que contenía su “vecino” de la derecha.

Para conseguirlo, haremos que el registro se cargue en cada flanco con el valor que contenía hasta ahora desplazado un bit a la izquierda, excepto por el bit situado a la derecha de todos, que se cargará con el nuevo valor de x .

De esta forma obtenemos que la salida z debe valer 1 cuando el contenido del registro sea 1010. Si llamamos $[s_3 s_2 s_1 s_0]$ a los bits de salida del registro, tenemos lo siguiente:

$$z = s_3 \cdot s_2' \cdot s_1 \cdot s_0'$$

Inicialmente, el registro tiene que contener un 0, ya que de otra forma se podría reconocer la secuencia sin que se hubiese producido. Por ejemplo, si el contenido inicial del registro fuese 0101 y el primer valor de x fuese 0, el circuito reconocería la secuencia 1010 después del primer flanco de reloj, de forma errónea. Por eso conectamos la señal *Inicio* en la entrada *clear* del registro.



15. Analizaremos punto por punto el circuito:

- La entrada X se carga en el registro $REG1$ en cada flanco de reloj (dado que no se han dibujado las entradas *load* y *clear*, se asume que valen 1 y 0, respectivamente).
- El bit de más peso del registro $REG1$, R_{n-1} , controla el multiplexor.
- Dado que los números están representados en complemento a 2, el bit R_{n-1} es el bit de signo del último número que se ha almacenado en el registro $REG1$. Este bit es 1 si el número es negativo y 0 si es positivo.
Por tanto, el multiplexor deja pasar el número que hay en $REG1$ si éste es positivo, y deja pasar un 0 si es negativo.
- El sumador suma este número con el contenido de registro de $REG2$, que inicialmente está a 0, y el resultado se guarda otra vez en $REG2$.
- La salida S del circuito está conectada a la salida de $REG2$.

Podemos concluir, pues, que la salida S del circuito es la suma de todos los números positivos que entran por X .

16. Vemos que la señal *Inicio* está conectada a todas las entradas *clear* de los registros. Por tanto, todos se ponen a 0 cuando empieza a funcionar el circuito.

Las entradas *load* de cada registro están conectadas a las salidas de un decodificador. Por tanto, sólo una de éstas está a 1 en cada momento, si $e = 1$, (la señal e está conectada a la entrada de validación del decodificador). Si $e = 0$, entonces no se carga ningún registro.

En las entradas del decodificador están las señales c_1 y c_0 . Por tanto, deducimos que estas dos señales controlan qué registro se carga en cada momento con el valor de la entrada X , que está conectada a la entrada de datos de todos los registros.

La salida S del circuito está conectada a un multiplexor de buses controlado por c_3 y c_2 . Cada entrada de datos del multiplexor está conectada a la salida de datos de uno de los cuatro registros. Por tanto, deducimos que c_3 y c_2 controlan qué contenidos de los cuatro registros sale en cada momento por la salida.

17.

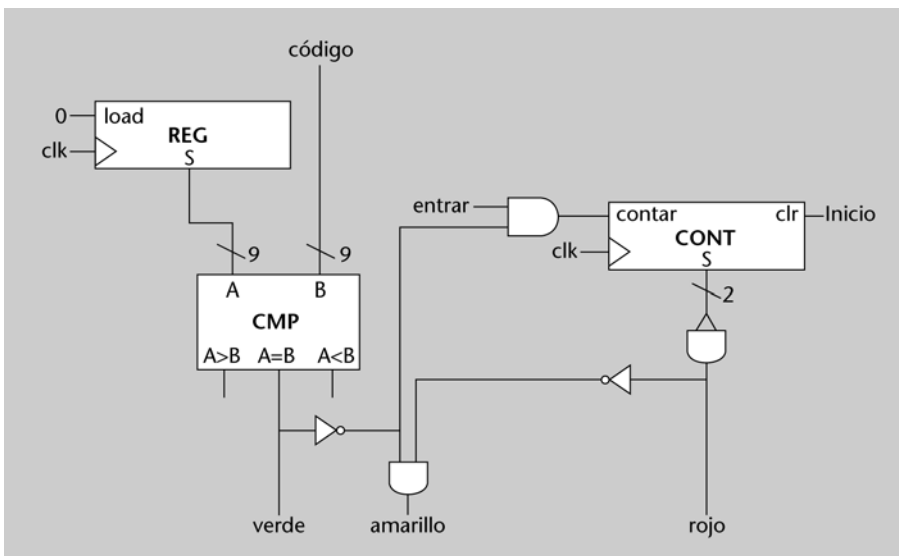
a) Las entradas son:

- *entrar*, de un bit.
- *código*, de 9 bits, porque el máximo valor que puede tener es 444.
- *Inicio*, de un bit.

Y las salidas son *verde*, *amarillo* y *rojo*, las tres de un bit.

b) El circuito debe tener un comparador para comparar el código que se ha tecleado con la contraseña correcta. Cuando la salida $A=B$ valga 1, se debe activar la señal *verde*, mientras que si vale 0 se deben activar o bien *amarillo* o bien *rojo*, dependiendo de cuántas veces se haya introducido un código incorrecto.

Para saber cuántas veces se ha tecleado un código incorrecto, usaremos un bloque contador como el que se diseña en la actividad 11; puesto que sólo debe contar hasta 3, puede tener sólo 2 bits. Se tendrá que incrementar en 1 siempre que la salida $A=B$ del comparador valga 0 y se haya introducido un nuevo código; esto lo conseguimos conectando a la entrada *contar* del contador la salida de una puerta AND, a cuyas entradas conectamos *entrar* y la negación de la salida $A=B$ del comparador. Observemos que, puesto que *entrar* vale 1 sólo durante un ciclo de reloj, el contador se incrementará sólo una vez para cada nuevo código que se teclee. La señal *rojo* se deberá activar cuando la salida del contador sea 11, y cuando esto pase la señal *amarillo* debe estar a 0. La luz amarilla se debe encender cuando el código tecleado no coincida con la contraseña correcta pero el contador aún no haya llegado a 3. La figura muestra el circuito completo. El registro de la izquierda es el que contiene la contraseña correcta.



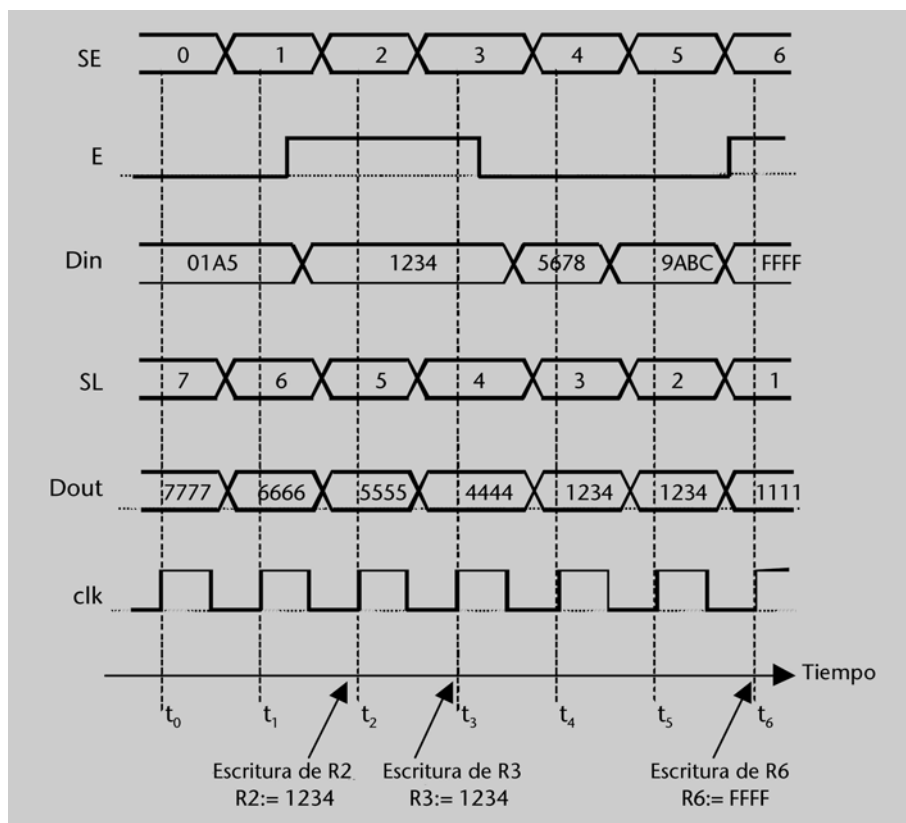
c) Sería necesario definir una señal *cambio_contraseña* que se activaría cuando se quisiera hacer el cambio y que se debería conectar a la entrada *load* del registro que guarda la contraseña correcta. También sería preciso establecer alguna manera de teclear la nueva contraseña, y hacer llegar su valor codificado en binario a la entrada de datos del registro.

18.

a) Las escrituras en el banco de registros tienen lugar en los flancos ascendentes si $E = 1$. Por tanto, los instantes en que se escribirá algún registro son t_2 , t_3 y t_6 . Para saber qué registro se escribe analizamos el valor de SE en estos instantes; para saber qué valor se carga, analizaremos Din . Obtenemos lo siguiente:

En el instante t_2 , $R2 := 1234$
 En el instante t_3 , $R3 := 1234$
 En el instante t_6 , $R6 := FFFF$

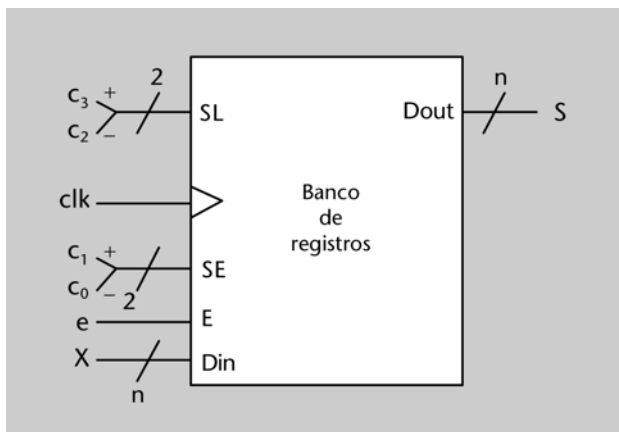
b) En el cronograma se puede ver la secuencia de valores en *Dout*. Para deducirlo, hay que observar en cada momento qué registro se lee (*SL*) y cuál es su contenido (recordemos que las lecturas se hacen de forma asíncrona).



c) El valor final de los registros es el siguiente:

$R0 = 0000$,
 $R1 = 1111$,
 $R2 = 1234$,
 $R3 = 1234$,
 $R4 = 4444$,
 $R5 = 5555$,
 $R6 = FFFF$,
 $R7 = 7777$.

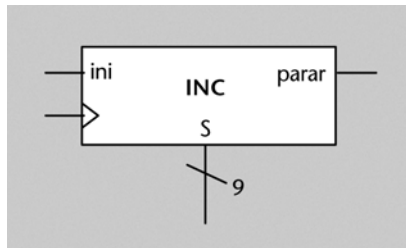
19. La funcionalidad del circuito de la actividad 16 (sin la entrada *Inicio*) se consigue con un banco de registros que conecta las señales a las entradas y salidas tal como se muestra en la figura.



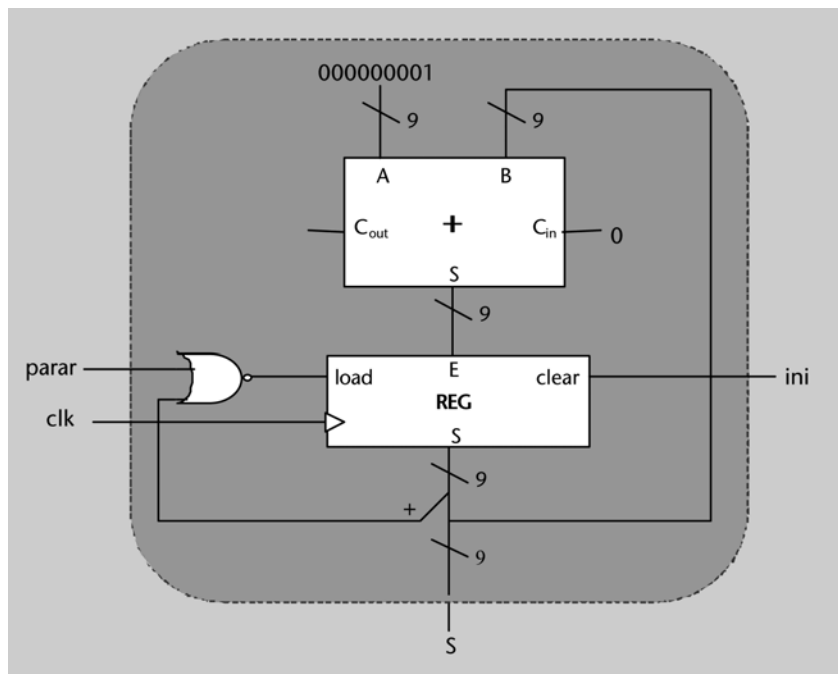
Deducimos, pues, que el circuito de la actividad 16 corresponde a una posible implementación de un banco de cuatro registros, al que se le añade la funcionalidad de inicializar todos los registros a 0.

20.

a) El circuito que se tiene que diseñar es el siguiente:



- Para hacer este bloque utilizaremos, básicamente, un registro y un sumador. El registro almacenará la salida S y el sumador permitirá incrementarla. Para hacer esto, a una entrada del sumador llegará el valor de S y a la otra, un 1.
- Implementaremos la señal *ini* con el *clear* del registro.
- Implementaremos la señal *parar* con la señal de *load* del registro. Cuando *parar* sea 1 o el bit de más peso de S sea 1, pondremos un 0 en *load*. En cualquier caso, la señal de *load* será 1. Por tanto, $load = (parar + S_8)'$.



b) El circuito tiene tres entradas y tres salidas.

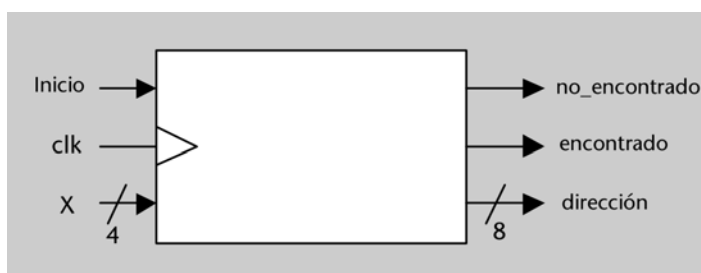
Entradas:

- Una entrada (*Inicio*) de un bit, que inicializa el circuito.
- Una entrada de reloj: *clk*.
- Una entrada de datos de cuatro bits: *X*.

Salidas:

- Una salida de ocho bits: *dirección*.
- Dos salidas de un bit: *encontrado*, *no_encontrado*.

A continuación se muestra una figura con esta descripción.



c) El bloque CMP es combinacional, y el resto son secuenciales.

d) El tamaño de la RAM es de $2^8 \cdot 4$ bits.

e) Para ver qué hace el circuito, lo analizaremos por partes:

- Cuando el circuito empieza a funcionar, la entrada X se almacena en el registro (porque *Inicio* está conectado en la entrada *load*).
- El bloque INC se incrementa en cada flanco de reloj. Empieza desde 0 gracias a la señal *Inicio*.
- La salida del bloque INC está conectada a la entrada de direcciones de la memoria. Puesto que $L/E = 0$, en cada ciclo de reloj se lee una palabra de la memoria, desde la dirección 0 hasta la 255. Si el bloque INC llega a 256 (100000000), parará de incrementarse, y se habrá recorrido toda la memoria. En este momento, el bit de más peso del bloque INC (no-encontrado) valdrá 1, y en $M@$ habrá un 0 (los ocho bits de menor peso de INC).
- El bloque INC también se puede parar antes si la entrada *parar* se pone a 1. Esto pasa cuando la señal *encontrado* vale 1.
- La salida de datos de la RAM se compara en cada instante con el contenido del registro (la entrada X). Si son iguales, la señal *encontrado* se pone a 1.
- Dado que *encontrado* detiene el incrementador, éste se para cuando se ha encontrado una palabra en la memoria que es igual a la palabra de entrada X . En este momento, la salida *dirección* contiene la posición de memoria donde se encuentra la palabra X .

Podemos concluir que lo que hace este circuito es buscar en qué posición de memoria se encuentra una palabra X . Si esta palabra está en la memoria, la señal *encontrado* se pone a 1 y por la salida *dirección* sale la posición de memoria donde se ha encontrado. Si la palabra no está en la memoria, la señal *no_encontrado* se pone a 1 y el contenido de la salida *dirección* no tiene ningún significado.

21.

a) El coche puede estar en cuatro situaciones: parado, girando hacia la derecha, girando hacia la izquierda y moviéndose adelante. El circuito tendrá, pues, cuatro estados, que llamaremos respectivamente *PARADO*, *DERECHA*, *IZQUIERDA* y *ADELANTE*.

b) Lo más habitual es que, al pulsar el botón “on” del coche para empezar a jugar, éste permanezca parado. Por tanto, podemos decir que el estado inicial es *PARADO*.

c) A partir de la tabla que describe la acción de las señales z_1 y z_0 sobre el coche, obtenemos que las señales de salida deben tener los siguientes valores:

Estado	z_1	z_0
PARADO	1	0
DERECHA	0	0
IZQUIERDA	0	1
ADELANTE	1	1

El enunciado nos dice que las transacciones que se producirán serán las siguientes:

Estado	e	d	Estado ⁺
PARADO	0	0	PARADO
PARADO	0	1	DERECHA
PARADO	1	0	IZQUIERDA
PARADO	1	1	ADELANTE
DERECHA	0	0	DERECHA
DERECHA	0	1	DERECHA
DERECHA	1	0	ADELANTE
DERECHA	1	1	PARADO
IZQUIERDA	0	0	IZQUIERDA
IZQUIERDA	0	1	ADELANTE
IZQUIERDA	1	0	IZQUIERDA
IZQUIERDA	1	1	PARADO
ADELANTE	0	0	ADELANTE
ADELANTE	0	1	DERECHA
ADELANTE	1	0	IZQUIERDA
ADELANTE	1	1	PARADO

22.

a) El circuito necesita saber si los valores de x e y han sido iguales en al menos tres ocasiones.

- Para esto tiene que ser capaz de recordar estas situaciones:
- Los valores de x e y no han sido iguales en ninguna ocasión.
- Lo han sido en una ocasión.

- Lo han sido en dos ocasiones.
- Lo han sido en tres ocasiones o más.

Éstos serán los estados del circuito, que llamaremos respectivamente, *NINGUNA*, *UNA*, *DOS* y *TRES_O_MÁS*.

b) Cuando el circuito se ponga en funcionamiento, puesto que aún no habrá llegado a ningún valor por las entradas, tendrá que encontrarse en el estado *NINGUNA*.

c) La salida sólo tiene que valer 1 cuando *x* e *y* hayan sido iguales al menos tres veces y, por tanto, la tabla de salidas es la siguiente:

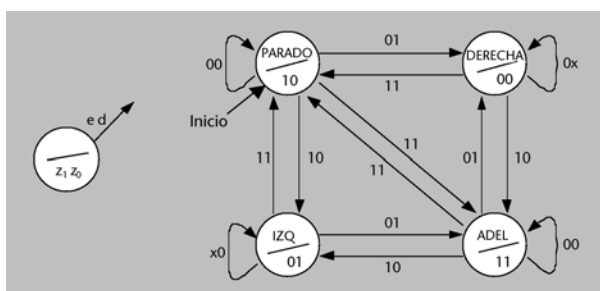
Estado	z
NINGUNA	0
UNA	0
DOS	0
TRES_O_MÁS	1

Siempre que *x* e *y* sean diferentes, el circuito permanecerá en el estado en que se encuentre. Cuando sean iguales, pasará a recordar que han sido iguales en una ocasión más. Una vez haya llegado al estado *TRES_O_MÁS*, ya no saldrá de allí. Por tanto, la tabla de transiciones es ésta:

Estado	x	y	Estado ⁺
NINGUNA	0	0	UNA
NINGUNA	0	1	NINGUNA
NINGUNA	1	0	NINGUNA
NINGUNA	1	1	UNA
UNA	0	0	DOS
UNA	0	1	UNA
UNA	1	0	UNA
UNA	1	1	DOS
DOS	0	0	TRES_O_MÁS
DOS	0	1	DOS
DOS	1	0	DOS
DOS	1	1	TRES_O_MÁS
TRES_O_MÁS	X	X	TRES_O_MÁS

23.

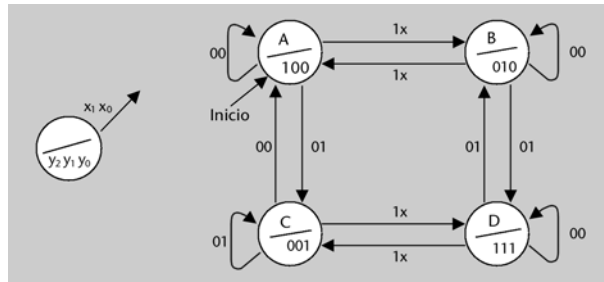
Llamaremos “IZQ”, al estado *IZQUIERDA* y “ADEL”, al estado *ADELANTE*. Recordemos que el estado inicial es *PARADO*; entonces traducimos directamente de las tablas de transiciones y salidas que hemos obtenido en la actividad 21 y podemos dibujar el siguiente grafo:



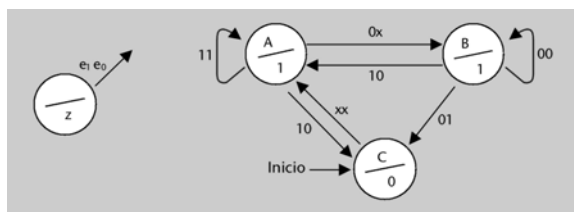
24.

Si traducimos directamente de las tablas de transiciones y de salidas, podemos dibujar los siguientes grafos:

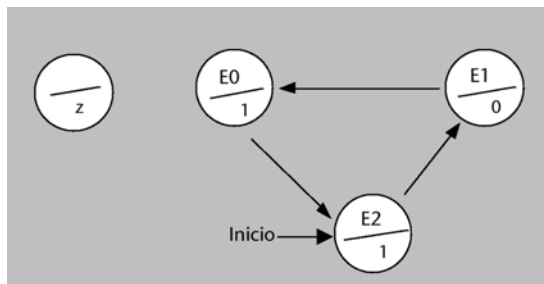
a)



b) En este caso, estando los estados en *B* o *C*, no se producirá nunca la combinación de entrada [1 1]. Por otro lado, del estado *C* pasamos siempre al estado *A*.



c) Fijémonos en que este circuito no tiene ninguna señal de entrada.



25.

a) Vemos que se trata de un circuito sin ninguna señal de entrada. Fijémonos en que hemos escrito en la tabla de salidas las señales en un orden diferente del que aparece en la leyenda del grafo (los podemos escribir en cualquier orden, siempre que mantengamos la coherencia).

Tabla de transiciones	
Estado	Estado ⁺
A	B
B	C
C	A

Tabla de salidas			
Estado	x	y	z
A	1	0	0
B	0	1	0
C	0	0	1

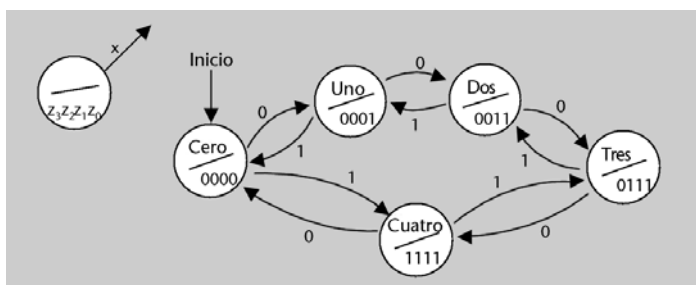
b) Si observamos el grafo, vemos que estando en el estado *B*, la entrada no vale nunca 1. Pese a todo, incluimos en la tabla de transiciones la fila correspondiente a esta combinación.

Tabla de transiciones		
Estado	a	Estado ⁺
A	0	B
A	1	A
B	0	C
B	1	x
C	0	A
C	1	A

Tabla de salidas		
Estado	p	q
A	0	0
B	1	0
C	0	1

26.

El grafo de estados es el siguiente:



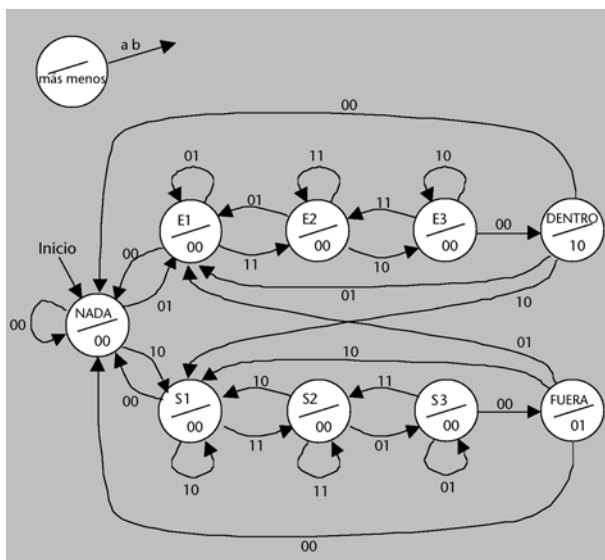
27.

Para entrar en el aparcamiento, un coche tiene que pasar cuatro etapas: primero pasará por delante del sensor *B*, después estará ante ambos sensores, después, sólo ante el sensor *A* y, por último, entrará en el aparcamiento. Para salir también tendrá que pasar por cuatro etapas análogas, pero en sentido contrario. En consecuencia, para poder determinar el valor de las señales *más* o *menos*, el circuito tiene que ser capaz de distinguir las situaciones que se muestran en la tabla siguiente. Se incluye también la tabla de salidas del circuito; vemos que la señal *más* sólo se tiene que poner a 1 cuando un coche ya ha entrado (estado *DENTRO*), y la señal *menos* sólo se tiene que poner a 1 cuando un coche ya ha salido (estado *FUERA*). El estado inicial es *NADA*.

Estado	Nombre	más	menos
No hay ningún movimiento	NADA	0	0
Entra un coche, etapa 1	E1	0	0
Entra un coche, etapa 2	E2	0	0
Entra un coche, etapa 3	E3	0	0
Ha entrado un coche completamente	DENTRO	1	0
Sale un coche, etapa 1	S1	0	0
Sale un coche, etapa 2	S2	0	0
Sale un coche, etapa 3	S3	0	0
Ha salido un coche completamente	FUERA	0	1

Las transiciones entre estados se muestran en el siguiente grafo. Vemos que hay estados en los que no se darán algunas combinaciones de entrada. Por ejemplo, en el estado *E1* (hay un coche que está ante el sensor *B*) no se dará nunca la combinación $[a \ b] = [1 \ 0]$ (porque el enunciado dice que nunca se encontrarán dos coches de cara).

Recordemos que un coche puede parar o hacer marcha atrás en cualquier momento. Por eso, por ejemplo, si en el estado *E2* (el coche está ante dos sensores) se produce la combinación $[a \ b] = [0 \ 1]$, vamos al estado *E1* (el coche ha hecho marcha atrás hasta situarse ante el sensor *B*). Si se produce la combinación $[a \ b] = [1 \ 1]$, nos quedaremos en el estado *E2* (el coche continúa ante los dos sensores).

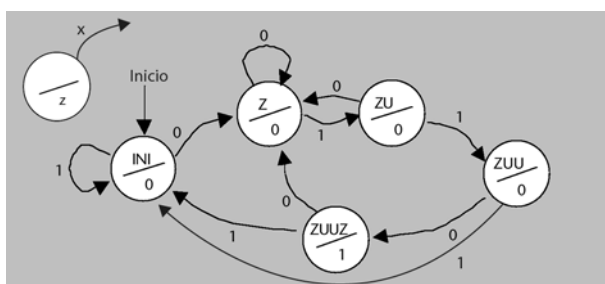


28.

a) El grafo de estados se muestra a continuación. Utilizamos los siguientes mnemotécnicos para los nombres de los estados:

Estado	nombre
Estado inicial	INI
Ha llegado un 0	Z
Ha llegado la subsecuencia 01	ZU
Ha llegado la subsecuencia 011	ZUU
Ha llegado la secuencia 0110	ZUUZ

Fijémonos en que en el estado *ZUUZ* se pasa siempre a un estado en que la salida vale 0 y, por tanto, cuando se reconoce la secuencia, la salida está a 1 durante un único ciclo de reloj.



29.

Este grafo de estados es muy parecido al que se ha obtenido en la actividad 28. La diferencia es que si llega un 1 estando en el estado *ZUUZ*, vamos al estado *ZU*, es decir, el circuito reconoce que ha llegado la subsecuencia 01. Por tanto, descubrimos que el 0 que nos ha llevado hasta el estado *ZUUZ* se considera como el primero de una secuencia nueva. Así pues, el circuito reconoce la secuencia 0110, pero permite el solapamiento entre secuencias consecutivas.

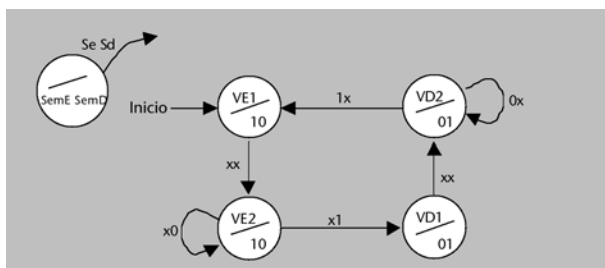
30.

Las entradas del circuito son *Sd* y *Se*, y las salidas, *SemD* y *SemE*. En cada momento puede haber sólo un semáforo en verde y, por tanto, siempre se cumplirá que $SemD = SemE'$.

Cuando un semáforo se pone en verde debe permanecer en este estado por lo menos dos ciclos de reloj; por tanto, habrá dos estados (*VD1* y *VD2*) en los que $SemD = 1$ y dos estados (*VE1* y *VE2*) en los que $SemE = 1$.

Cuando un semáforo ya ha estado en verde dos ciclos, no se volverá a poner en rojo hasta que lleguen coches por el otro lado.

Dado que inicialmente el semáforo izquierdo se tiene que mantener en verde durante dos ciclos, el estado inicial es *VE1*. El grafo, pues, es el siguiente:

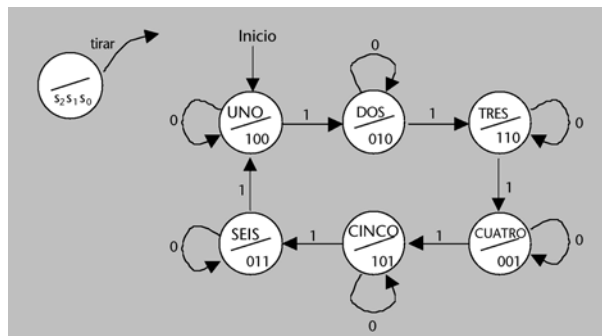


31.

Veamos cómo hay que iluminar los puntos del dado para formar las diferentes combinaciones:

Combinación	Puntos iluminados	Salidas		
		s_2	s_1	s_0
1	d	1	0	0
2	c, e	0	1	0
3	c, d, e	1	1	0
4	a, b, f, g	0	0	1
5	a, b, d, f, g	1	0	1
6	a, b, c, e, f, g,	0	1	1

El grafo de estados es el siguiente (el estado inicial es U):



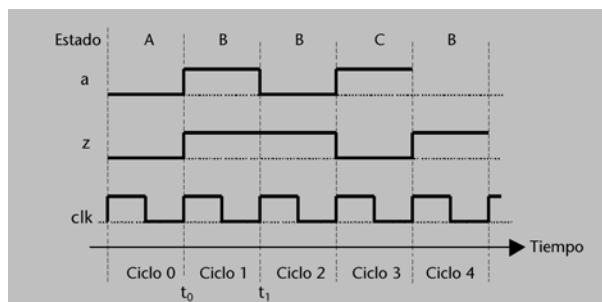
32.

En el ciclo 0, el circuito está en el estado A. En el ciclo 1, pasa al estado B. Por tanto, en el instante t_0 la entrada tenía que valer 0. Dado que suponemos que la entrada sólo cambia de valor en los flancos, obtenemos que $a = 0$ durante todo el ciclo 0.

Para que el circuito continúe en el estado B en el ciclo 2, es preciso que en el instante t_1 (y, por tanto, durante todo el ciclo 1), la entrada valga 1. Durante el ciclo 2 la entrada tiene que valer 0, para que el circuito pase al estado C en el ciclo 3. No disponemos de ninguna información que nos permita determinar cuánto vale la entrada durante el ciclo 4.

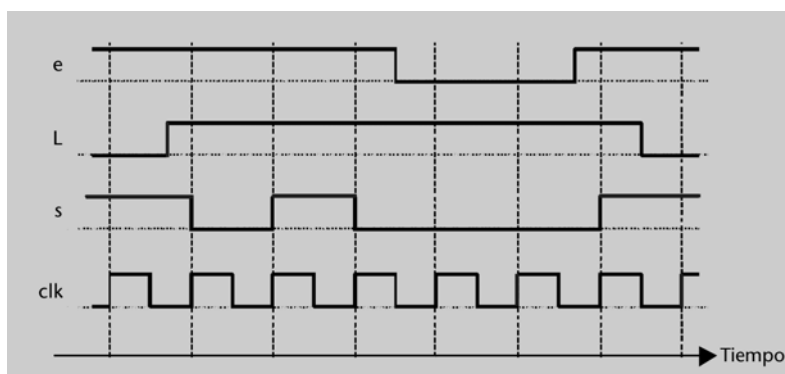
El valor de la señal de salida viene determinada por el estado en que se encuentra el circuito: 0 mientras está en el estado A o en el C, y 1 mientras está en el estado B.

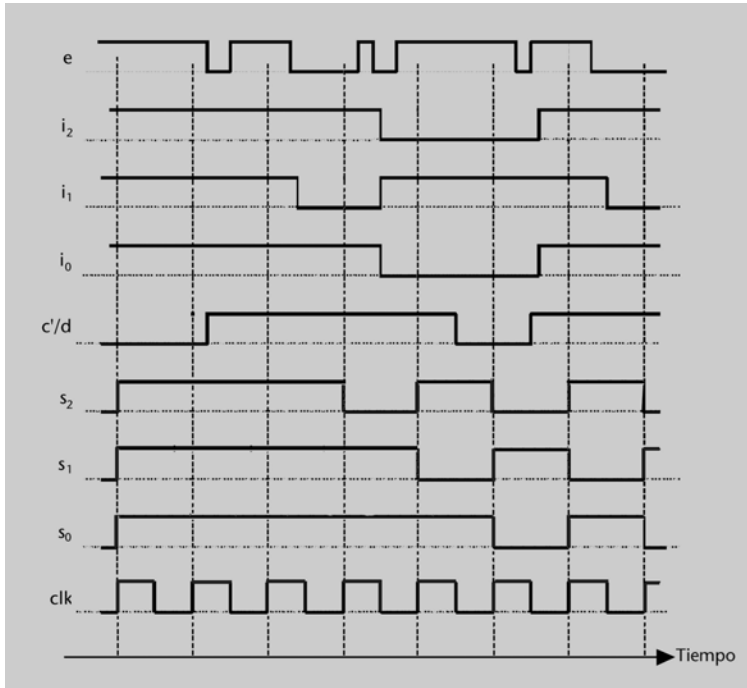
El cronograma completo se muestra a continuación:



Ejercicios de autoevaluación

1. La señal e está conectada a una puerta AND con la salida conectada a la entrada D del biestable. Por tanto, siempre que e valga 0 llegará un 0 a esta entrada. Podríamos decir que cumple el papel de una señal de *reset* síncrono. Mientras e vale 1, el biestable invierte su valor en cada ciclo. La señal L , conectada a la entrada *load*, inhibe la carga del biestable cuando se pone a 0.





3.

a) El bit c vale 0 si $A < B$ y 1 si $A \geq B$.

Este bit c está conectado a la salida f_{out} del bloque B1 situado a la izquierda de todos. Veamos cuánto tiene que valer esta salida f_{out} :

- Si $a_3 = 0$ y $b_3 = 1$, entonces $A < B$, de forma que c (y por tanto, f_{out}) tiene que valer 0.
- Si $a_3 = 1$ y $b_3 = 0$, entonces $A > B$, con lo que c (y por tanto, f_{out}) tiene que valer 1.
- Si $a_3 = b_3$, entonces hay que saber cuánto valen los bits de menos peso de A y B para decidir el valor que tendrá que tomar c .

Fijémonos en el valor del punto d del circuito. Ésta sería la salida del circuito B2 en el caso de que los números fuesen de tres bits en lugar de cuatro. Por tanto, $d = 0$ si $[a_2 a_1 a_0] < [b_2 b_1 b_0]$ y 1 en el caso contrario. Ésta es la información que falta para decidir el valor de c en el caso de que $a_3 = b_3$. Por tanto, debido a que d está conectada a la entrada f_{in} del último bloque B1, obtenemos que la tabla de verdad del bloque B1 situado más a la izquierda es la siguiente (en general, la tabla de verdad de un bloque B1 cualquiera será la misma, cambiando sólo a_3 y b_3 por a_i y b_i):

a_3	b_3	f_{in}	f_{out}	
0	0	0	0	$A < B$, ya que $a_3 = b_3$ i $[a_2 a_1 a_0] < [b_2 b_1 b_0]$
0	0	1	1	$A \geq B$, ya que $a_3 = b_3$ i $[a_2 a_1 a_0] \geq [b_2 b_1 b_0]$
0	1	0	0	$A < B$, ya que $a_3 < b_3$
0	1	1	0	"
1	0	0	1	$A \geq B$, ya que $a_3 > b_3$
1	0	1	1	"
1	1	0	0	$A < B$, ya que $a_3 = b_3$ i $[a_2 a_1 a_0] < [b_2 b_1 b_0]$
1	1	1	1	$A \geq B$, ya que $a_3 = b_3$ i $[a_2 a_1 a_0] \geq [b_2 b_1 b_0]$

Los bits de peso -1 no existen, pero el bloque B1 de la derecha se tiene que comportar "como si fuesen iguales", es decir, como si $a_{-1} = b_{-1}$. Esto se consigue conectando un 1 a su entrada f_{in} .

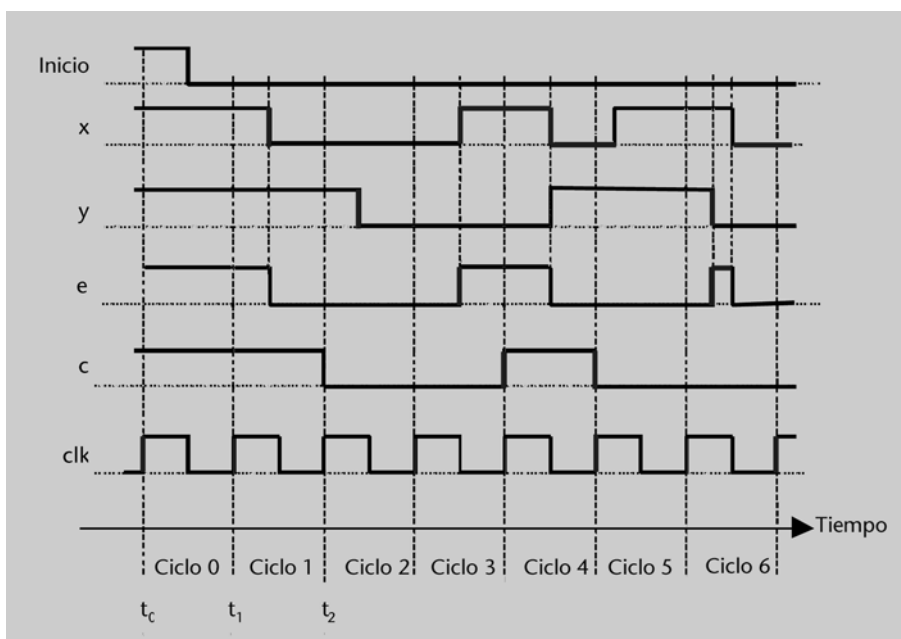
b) En el apartado a) hemos visto que cada bloque B1 hace la comparación de dos bits A y B . Esta comparación consulta, cuando no puede decidir sólo con los bits a_i y b_i , el resultado de la comparación de los bits de menos peso.

El circuito B3 compara dos bits de entrada, x e y , y almacena el resultado de la comparación en un biestable. El bloque B1 consultará este resultado cuando compare los bits x e y en el siguiente flanco de reloj, ya que la salida del biestable está conectada a la entrada f_{in} .

Si comparamos este circuito con el del apartado a), podemos ver que pueden cumplir la misma función, si en el primer ciclo de reloj se conectan $[a_0 b_0]$ a los puntos $[x y]$, en el segundo ciclo se conectan $[a_1 b_1]$, y así sucesivamente. La señal *Inicio* se puede usar para cargar un 1 en el biestable cuando empieza a funcionar el circuito, es decir, puede hacer el papel del 1 que se conecta a la entrada f_{in} del bloque B1 que está a la derecha del todo en el circuito B2.

La diferencia entre los circuitos B2 y B3 es que el primero es combinacional, mientras que el segundo es secuencial. En el circuito B2, todos los bits de los números A y B se comparan a la vez, y el resultado de la comparación estará disponible al instante (para ser más precisos, habrá que esperar sólo el tiempo de retraso de las puertas que forman los bloques B1). En cambio, en el circuito B3, los bits de A y B se comparan pareja por pareja, de forma secuencial en el tiempo. El circuito debe tener un biestable para recordar en cada momento el resultado de la comparación del par anterior. El resultado final de la comparación estará disponible cuatro ciclos después de que el circuito empiece a funcionar.

c) El cronograma se muestra a continuación. Observamos que el punto e cambia de valor de forma asíncrona, de acuerdo con las variaciones en x e y .



Durante el ciclo 0, $x = y = 1$. Por tanto, $a_0 = b_0 = 1$. Durante el ciclo 1, x cambia de valor. El valor que se guarde en el biestable depende del valor de x , y e e al final del ciclo 1 (en el instante t_2) y, por tanto, diremos que $a_1 = 0$. Si seguimos el mismo razonamiento en todos los casos, obtenemos que los números que se han comparado son los siguientes:

$$A = 0101001, B = 0110011.$$

Se cumple que $A < B$. Podemos comprobar que, efectivamente, la salida c del circuito vale 0.

4.

a) El circuito tiene que saber si es de día o de noche para regular adecuadamente la duración de cada luz. Por tanto, necesita una señal de entrada, que llamaremos d/n , y supondremos que vale 0 cuando es de día y 1 cuando es de noche.

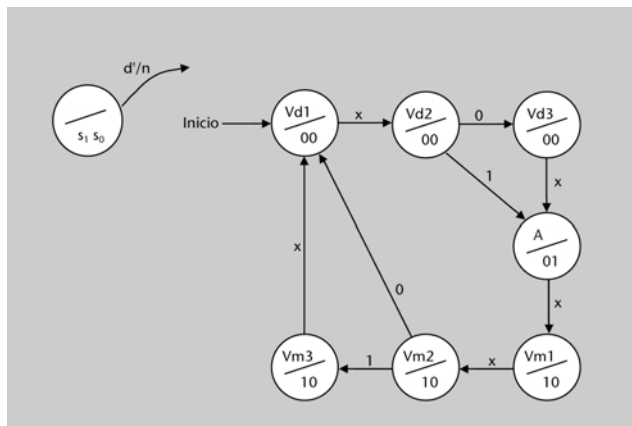
Las salidas tienen que indicar qué luz está encendida en cada momento. Por ejemplo, el circuito podría generar 3 salidas, una asociada a cada luz. Otra posibilidad es que genere sólo dos señales de salida, que controlen el semáforo de la siguiente forma:

s_1	s_0	Semáforo
0	0	Verde
0	1	Amarillo
1	0	Rojo

Tomaremos esta segunda posibilidad.

b) El grafo de estados se muestra a continuación. Hemos identificado para *Vdi* los estados durante los cuales el semáforo está en verde, para *A* el estado durante el cual el semáforo está en amarillo, y para *Vmi* los estados durante los cuales el semáforo está en rojo. Hemos supuesto que el estado inicial es *Vd1*, pero el enunciado no nos indicaba nada al respecto, por lo que podíamos haber tomado cualquier estado como inicial.

Observamos que, durante el día, sólo se recorren dos de los estados en que el semáforo está en rojo, y por la noche sólo se recorren dos de los estados en que el semáforo está en verde.

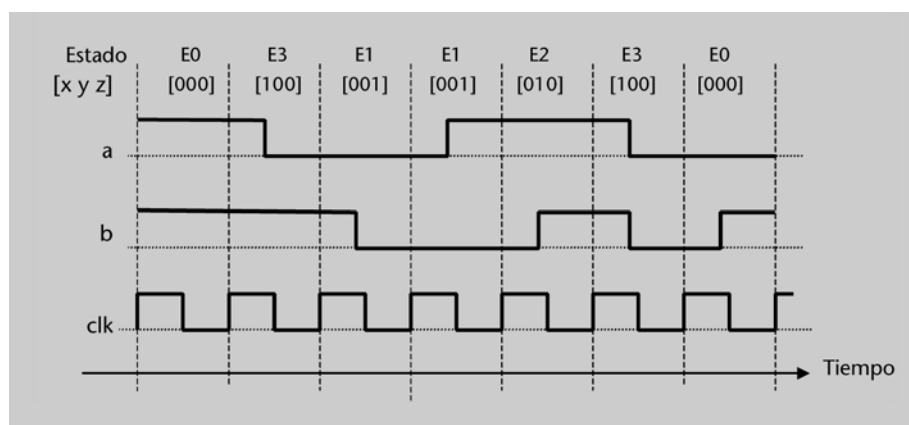


5.

a) Las tablas se muestran a continuación:

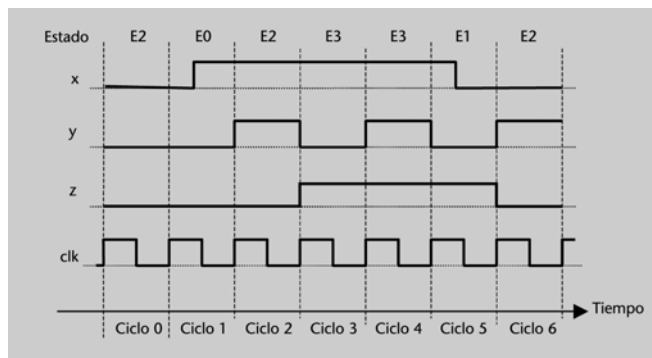
Tabla de salidas		Tabla de transiciones			
Estado	x y z	Estado	a	b	Estado ⁺
E0	0 0 0	E0	0	0	E0
E1	0 0 1	E0	0	1	E1
E2	0 1 0	E0	1	0	E2
E3	1 0 0	E0	1	1	E3
		E1	0	0	E1
		E1	0	1	x
		E1	1	0	E2
		E1	1	1	E2
		E2	0	0	E1
		E2	0	1	E3
		E2	1	0	E1
		E2	1	1	E3
		E3	0	0	E0
		E3	0	1	E1
		E3	1	0	x
		E3	1	1	x

b) El cronograma se muestra a continuación:



6.

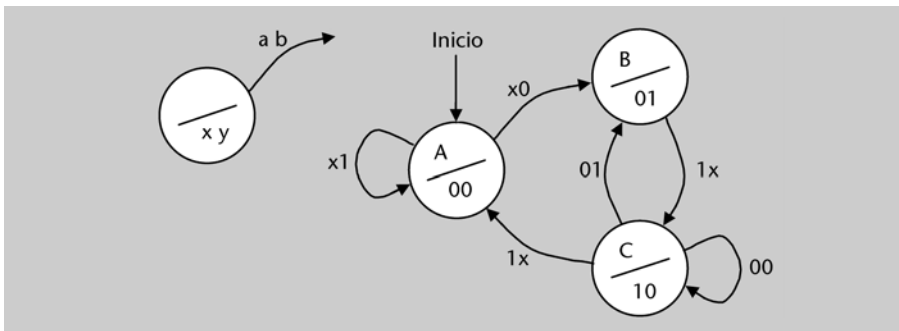
El cronograma completo se muestra a continuación:



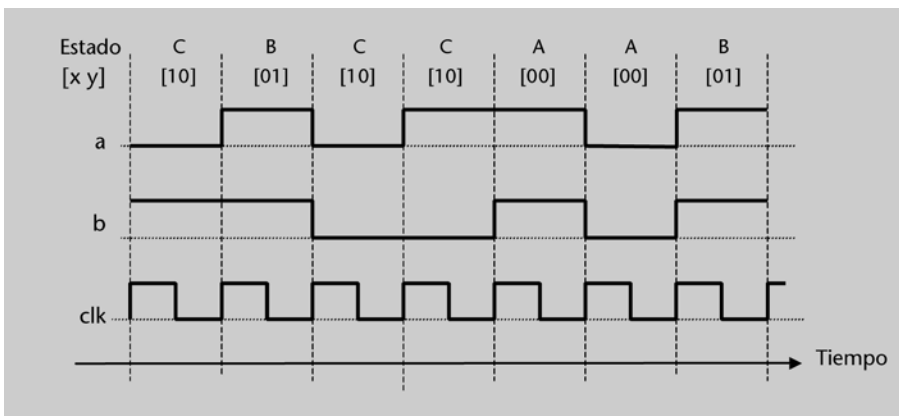
Recordad que en los circuitos secuenciales en el momento del flanco ascendente se debe coger el valor que tienen las entradas a la izquierda del flanco. Por ejemplo, al inicio del ciclo 2, x vale 1 y y vale 0.

7.

El grafo se muestra a continuación.



b) El cronograma se muestra a continuación:



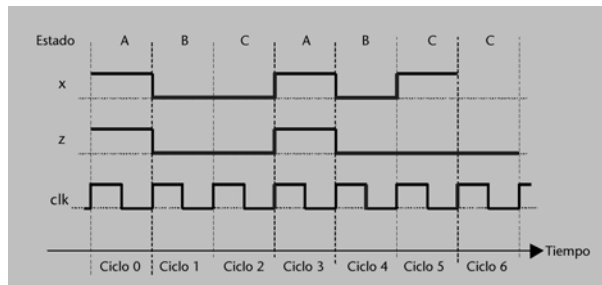
8.

Completaremos el cronograma por pasos:

- Durante el ciclo 1 el circuito está en el estado B y, por tanto, la salida vale 0. Sólo se puede llegar al estado B desde el A con entrada 1. Por tanto, deducimos que durante el ciclo 0 el circuito estaba en el estado A (por tanto, la salida valía 1) y que la entrada valía 1.
- Durante el ciclo 1, estamos en el estado B y la entrada es 0; por tanto, en el siguiente ciclo vamos al estado C con salida 0.
- Durante el ciclo 3, la salida vale 1. El único estado en el que la salida vale 1 es el A . Para llegar al estado A a partir del C , se tiene que cumplir que durante el ciclo 2, la entrada valga 0.
- Durante el ciclo 4, la salida vale 0. Si tenemos en cuenta que venimos del estado A , deducimos que en un ciclo 4 estamos en el estado B (ya que se puede ir al B desde el A); por tanto, durante el ciclo 3 la entrada valía 1.
- Durante el ciclo 4 estamos en el estado B y la entrada vale 0; por tanto, en el ciclo 5 estaremos en el estado C , y la salida valdrá 0.

- En el ciclo 6 vamos a un estado en el que la salida vale 0. Si tenemos en cuenta que durante el ciclo 5 estábamos en el estado C, deducimos que nos hemos quedado en el mismo estado; por tanto, la entrada durante el ciclo 5 tenía que valer 1.
- No tenemos ninguna información que nos permita saber cuánto vale la entrada durante el ciclo 6.

El cronograma completo se muestra a continuación:



9.

Inicialmente, el circuito está en el estado $E0$, y se queda mientras $p_{ab} = 0$. Si p_{ab} vale 1, el circuito pasa al estado $E1$. Deducimos, pues, que el estado $E1$ indica que la puerta de la caja fuerte ha estado abierta durante 30 segundos (un ciclo de reloj). Podemos seguir el mismo razonamiento para ver que el estado $E2$ señala que la caja ha estado abierta durante un minuto, y así sucesivamente hasta el estado $E4$, que indica que la caja ha estado abierta durante dos minutos seguidos. La salida *alarma* vale 1 sólo en el estado $E4$.

A partir de las transiciones, concluimos que la señal de alarma se activa cuando la caja permanece abierta durante dos minutos o más, y se desactiva un ciclo después de que se cierre la puerta de la caja fuerte.

Bibliografía

Gajsky, D.D. (1997). *Principios de diseño Digital*. Prentice-Hall.

Hermida, R.; Corral, A. del; Pastor, E.; Sánchez, F. (1998). *Fundamentos de Computadores*. Madrid: Síntesis.

