6 Circuitos lógicos secuenciales

Juan J. Navarro
Toni Juan

Primera versión: 10-2007
Versión actual: 09-2010

Hay dos tipos de circuitos lógicos, los combinacionales y los secuenciales. Hemos estudiado primero la síntesis de combinacionales con pocas entradas mediante un método sistemático (capítulo 3) y después el diseño modular (a bloques multinivel) de circuitos combinacionales que procesan palabras de n bits (capítulo 4 para naturales y 5 para enteros). Ahora vamos a hacer lo mismo con los circuitos secuenciales. Primero, en este capítulo estudiamos, además de otras cosas, la síntesis de circuitos secuenciales con pocas entradas y pocos estados mediante un método sistemático. Después, en el capítulo 7, trataremos el diseño modular de procesadores de propósito específico, que son circuitos secuenciales que procesan palabras de n bits.

Los circuitos secuenciales tienen capacidad de memorizar información: las salidas en un determinado momento dependen, no sólo de las entradas en ese momento sino que dependen también del estado del circuito (la información memorizada hasta ese momento). Estudiamos los circuitos secuenciales síncronos, que tienen una señal de reloj que indica los momentos concretos en los que se modifica el estado del circuito.

El procesador, que es el núcleo del computador, es un circuito secuencial grande, que procesa palabras de n bits, y que además es de propósito general (puede cambiar el tipo de procesado de la información que efectúa). En los capítulos 8, 9 y 10 caminamos hacia el procesador de propósito general, para terminar diseñando nuestro propio computador. Por último, los capítulos 11, 12 y 13 sirven para consolidar el conocimiento sobre el lenguaje máquina/ensamblador y la microarquitectura de nuestro computador uniciclo y la del multiciclo.

En este capítulo presentamos, en primer lugar, una introducción a los conceptos básicos sobre circuitos secuenciales. Después, en la sección 6.2, mostramos la estructura a bloques de los circuitos secuenciales según los modelos de Mealy y de Moore y cómo se especifica la funcionalidad del circuito mediante las tablas de transiciones y de salida. Después, en la sección 6.3, tratamos con detalle la especificación del funcionamiento lógico del circuito mediante un grafo de estados. En las secciones 6.4 y 6.5 tratamos el análisis lógico y la síntesis de secuenciales y por último, en la sección 6.6,

hacemos el análisis temporal para obtener el tiempo de ciclo mínimo del circuito.

6.1 Introducción

Aquí presentamos los conceptos básicos del tema de secuenciales (memoria y sincronización) y definimos el biestable D: el elemento de memorización más sencillo, que almacena un bit de información; y el registro: que almacena n bits. A continuación damos las reglas de interconexión de dispositivos combinacionales y biestables para formar circuitos secuenciales más complejos; y por último, observamos qué ocurre en el circuito durante un ciclo de reloj. Esto sirve para determinar cómo encontrar el tiempo de ciclo mínimo para que el circuito secuencial opere correctamente, lo que se verá en detalle al final del capítulo (sección 6.6).

6.1.1 Necesidad de memoria

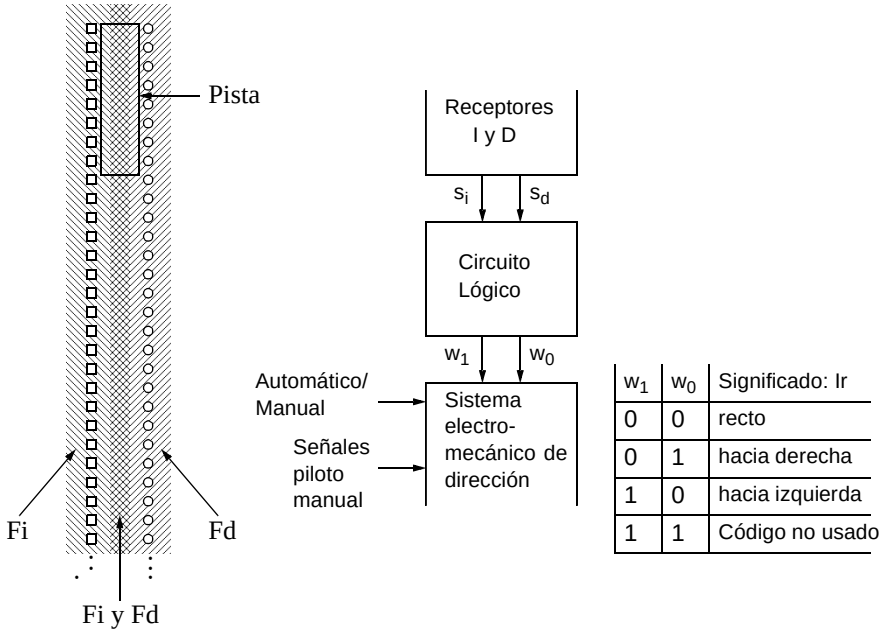
Un circuito secuencial se caracteriza porque sus salidas en un determinado momento no son sólo función de lo que valen las entradas en ese momento (como ocurre con los circuitos combinacionales) sino que son función, también, de los valores de las entradas en momentos anteriores. Esta capacidad de recordar la información que ya no está presente en las entradas permite tener en cuenta lo que ocurrió a lo largo del tiempo y con ello realizar procesados de la información mucho más complejos. Veamos un ejemplo de ello.

Ejemplo de un piloto automático para el aterrizaje de aviones

Vamos a diseñar el subsistema del piloto automático encargado de controlar la dirección de un avión durante el aterrizaje. Para que el aterrizaje sea seguro, el avión debe estar situado sobre una línea recta, prolongación de la pista de aterrizaje, unos cuantos kilómetros antes de llegar a la pista. Para que el piloto automático sepa la posición del avión respecto de esa línea recta, se dispone de dos filas de antenas, una a cada lado de la línea. Las antenas son muy direccionales y emiten hacia el cielo. Las antenas a la izquierda de la línea emiten una señal de radio con frecuencia Fi y las de la derecha con frecuencia Fd. En la figura 6.1 se muestra la vista desde el cielo de la pista de aterrizaje, las filas de antenas (pequeños cuadrados a la izquierda y círculos a la derecha), y la zona, con dos tramas distintas, en que se detectan cada una de las frecuencias. Se ve claramente la zona de intersección de las dos tramas, donde se detectan las dos frecuencias, que identifica la zona correcta de aterrizaje. Fuera de las tramas no se detecta ninguna frecuencia.

El avión dispone de dos receptores de radio, I y D, sintonizados a las frecuencias Fi y Fd respectivamente. El receptor I genera una señal binaria, s_i, que vale 1 cuando está recibiendo señal a su frecuencia de sintonía y vale 0 cuando no la recibe. Lo mismo ocurre con el receptor D que genera s_d.

El circuito lógico que debemos diseñar (que se muestra también en la figura 6.1) recibe como entradas s_i y s_d y genera como salidas dos señales binarias w_1 y w_0 que van al subsistema electromecánico que gobierna la dirección del avión. Estas señales le indican al sistema electromecánico si debe dirigir el avión recto, hacia la derecha o hacia la izquierda, según el código que indica la tabla de la figura.



w_1	w_0	Significado: Ir
0	0	recto
0	1	hacia derecha
1	0	hacia izquierda
1	1	Código no usado

Fig. 6.1 Pista de aterrizaje con las zonas de recepción de las frecuencias Fi y Fd. Subsistema del piloto automático encargado del control de dirección del avión

El piloto, al acercarse a la pista de aterrizaje, visualiza la recepción de las frecuencias de radio y cuando identifica que se reciben las dos frecuencias activa la señal Automático/Manual para indicarle al sistema electromecánico de dirección que se guíe por las señales que le envía el circuito lógico, en vez de hacer caso de las señales que le envían los mandos del piloto manual.

Durante el aterrizaje pueden venir fuertes vientos racheados y desplazar el avión en una dirección no deseada, distinta de la dirección a la que el sistema intenta llevar el avión, e incluso pueden llevarlo a la zona en la que no se detecta ninguna de las dos frecuencias.

Inicialmente vamos a especificar el sistema con un circuito combinacional, como los estudiados hasta ahora. Cuando el avión esté sobre la línea de aterrizaje, entradas del circuito $s_i = s_d = 1$, el avión debe ir recto; cuando el avión esté en la zona de la derecha, $s_i = 0$ y $s_d = 1$, el avión debe ir hacia la izquierda; y cuando se encuentre en la zona de la izquierda, $s_i = 1$ y $s_d = 0$, debe ir hacia la derecha.

s_i	s_d	a	w_1	w_0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	0	0

Pero, ¿qué debe ocurrir cuando, por un fuerte viento, el avión se salga de la zona marcada y no reciba ninguna de las dos frecuencias? En este caso, si el circuito es combinacional no puede saber en qué lado está el avión, por lo que lo más sensato es que el circuito avise al piloto para que este tome el control manual del aterrizaje y lleve el avión a una de las zonas en la que se reciba al menos una de las frecuencias, para poder activadar después el piloto automático. Para ello, disponemos de una nueva señal de salida del circuito que

denominamos a, de alarma, para que cuando se active avise al piloto de la situación. Esta señal no se ha dibujado en la figura. La tabla de verdad del circuito combinacional se muestra sobre este párrafo. Cuando no se detecta ninguna frecuencia el circuito hace sonar la alarma y le indica al sistema electromecánico que lleve el avión recto (a la espera de que el piloto tome el control manual del avión.

¿Podríamos mejorar el sistema si el circuito lógico no fuera combinacional, sino que tuviera capacidad de recordar qué valor tenían las señales de entrada antes del momento actual? La respuesta es sí. En el momento en que el circuito reciba las señales $s_i = s_d = 0$, si recuerda que antes de este momento recibió las señales, por ejemplo, $s_i = 0$ y $s_d = 1$, sabe que se encuentra en la parte de la derecha que no recibe ninguna señal y puede ordenar al sistema electromecánico que dirija el avión hacia la izquierda. Sin embargo, si cuando $s_i = s_d = 0$ el circuito recuerda que antes recibió las señales $s_i = 1$ y $s_d = 0$, sabe que el avión está sobre la zona de la izquierda donde no hay señales y puede ordenar ir hacia la derecha.

Acabamos de ver claramente que un circuito lógico con capacidad de recordar el valor de las entradas anteriores sería mucho más efectivo que un simple circuito combinacional. En la sección 6.1.5 definimos el dispositivo de memorización de un bit, el biestable D, y en la sección 6.5 diseñaremos el circuito secuencial de este piloto automático.

6.1.2 Conveniencia de sincronización

Vamos a justificar la conveniencia de la sincronización de circuitos secuenciales con dos ejemplos.

Contador secuencial de unos

Supongamos que debemos diseñar un circuito secuencial, con memoria, que tiene una entrada x de un bit y una salida w de *n* bits. Los *n* bits de salida deben codificar en binario el número de bits módulo 2^n con valor 1 presentes en la entrada desde que comenzó a funcionar el sistema.

Ante una señal x como la de la figura 6.2 no podemos saber si la secuencia de bits a lo largo del tiempo que se muestra es, por ejemplo, 0,1,0,1,0,0,1,0, o es 0,0,1,1,0,0,0,0,1,1,0,0, o es otra. Hace falta alguna otra señal u otro tipo de información que nos indique cuánto dura un bit o mejor, cuándo se debe "mirar" el valor de la entrada x porque en ese momento la señal tiene un valor válido de la secuencia de bits de entrada. Esta es la señal de reloj que definimos en la siguiente sección.

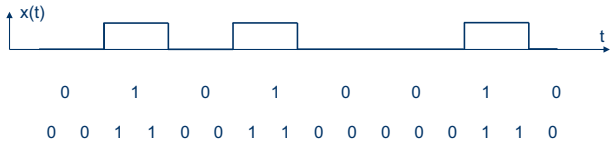


Fig. 6.2 Señal de entrada de datos al contador secuencial de unos.

Incrementador binario

Otro ejemplo distinto de necesidad de sincronización lo detectamos ya en el capítulo 2 al observar que, debido al tiempo de propagación de las señales a través de las puertas, las salidas de los circuitos

combinacionales no son correctas en todos los instantes de tiempo (no siguen el comportamiento lógico que indica la tabla de verdad del circuito). Por ello, desde que se produce un cambio en las entradas hay que esperar un tiempo a que las salidas se estabilicen al valor correcto.

Dijimos entonces que no debíamos preocuparnos por esto ya que en el computador, que es un sistema secuencial síncrono, las señales de datos, las que llevan codificada la información, no están solas. Existe una señal en el sistema, que se denomina señal de reloj, que indica cuándo la información que hay en las señales de datos es correcta. Sólo se almacena la información de los datos en la memoria del circuito secuencial en los momentos que esta es correcta, y esto lo indica la señal de reloj.

6.1.3 Señal de reloj: Clk

La señal de reloj es una señal binaria (sólo puede tomar dos valores, 0 o 1) y periódica, de periodo Tc (Tiempo de ciclo). Cada Tc unidades de tiempo, o lo que es lo mismo, cada ciclo, su comportamiento se repite. Durante la primera parte del ciclo la señal de reloj vale 1 y durante la segunda parte vale 0.

Puede ocurrir que el semiperiodo a 1 no dure el mismo tiempo que el el semiperiodo a 0, pero como esto de momento no tiene importancia, dibujaremos siempre la señal de reloj equilibrada (ver figura 6.3). Por ello, si no se indica lo contrario, la señal de reloj queda caracterizada por su tiempo de ciclo, Tc, o su frecuencia, 1/Tc.

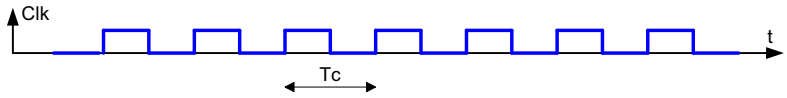


Fig. 6.3 Señal de reloj, Clk.

Para ser independientes de la tecnología medimos Tc en unidades de tiempo, u.t., aunque normalmente se hace en segundos, s, (o mejor en nanosegundos, ns, donde 1 ns = 10^{-9} s). Así, nosotros medimos la frecuencia en ciclos por unidad de tiempo; aunque normalmente se mide en hercios, Hz, donde 1 Hz es un ciclo por segundo (o mejor en gigahercios, GHz, donde 1 GHz = 10^{9} Hz).

6.1.4 Circuito secuencial síncrono

En un circuito secuencial síncrono complejo, formado por varios circuitos conectados entre sí, como es el computador que vamos a diseñar, hay una única señal de reloj que indica mediante su flanco ascendente (cuando la señal de reloj pasa de 0 a 1) que todas las señales de entrada y salida de todos los dispositivos combinacionales que forman el circuito son correctas. La señal de reloj se usa para validar las señales de datos y para indicar a los dispositivos que almacenan información (biestables D) cuándo deben almacenar el valor de una señal porque ya es correcto.

Ahora, volviendo al ejemplo del contador de unos de la sección 6.1.2, si miramos la señal x junto con la señal de reloj, Clk, ya no hay problemas de interpretación. La figura 6.4 muestra el símbolo del circuito y el cronograma de las señales Clk y x de donde se ve que la secuencia de bits es 0,1,0,1,0,0,1.

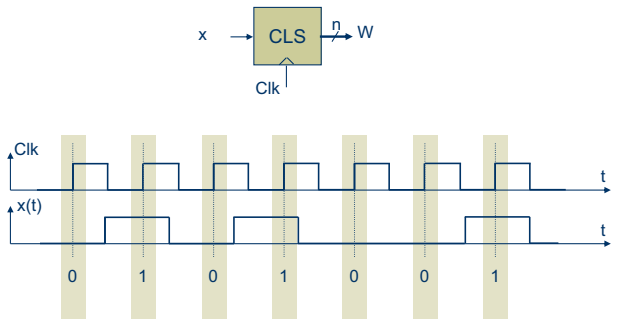


Fig. 6.4 Símbolo del circuito lógico secuencial síncrono y cronograma de la señal de datos de entrada síncrona (acompañada de la señal de reloj).

En un circuito secuencial síncrono el tiempo está partido en tramos de Tc unidades de tiempo: en ciclos. En cada ciclo, una vez que una señal se estabiliza a su valor correcto se mantiene estable al menos hasta el final del ciclo y el inicio del siguiente, cuando se produce el flanco ascendente de reloj.

Señal síncrona. En un circuito secuencial síncrono no importa el valor de las señales en cualquier momento, importa el valor correcto estabilizado: el que tienen todas las señales del circuito, con total seguridad, antes del final de cada ciclo. Para ello, claro está, el tiempo de ciclo debe ser suficientemente grande para permitir que todas las señales se estabilicen a su valor correcto. Por lo tanto, podemos referirnos a la secuencia de valores de una señal binaria síncrona, como una secuencia de bits: los bits que toma la señal al final de cada ciclo. En la parte superior de la figura 6.5 se muestra el cronograma de una señal binaria x y el reloj Clk que la hace síncrona.

Cronograma simplificado. En la parte inferior de la figura 6.5 se ha dibujado la tabla con los valores de la señal en cada ciclo (una columna por ciclo, incrementándose el número de ciclo hacia la derecha, conforme avanza el tiempo): a esta tabla la llamamos cronograma simplificado porque no muestra los detalles de los valores de la señal en cada ciclo antes de estabilizarse. Como se ve en el cronograma detallado de x(t), la señal puede fluctuar pasando de 0 a 1 y de 1 a 0 varias veces dentro de cada ciclo hasta que se estabiliza a su valor correcto antes del final de cada ciclo. Esto es debido al tiempo de propagación de los circuitos combinacionales por los que pasan las señales desde la salida de los biestables hasta el punto del circuito donde se encuentra la señal x (y a los valores concretos de las salidas de los biestables en cada ciclo). El cronograma simplificado omite esos valores intermedios, o lo que es lo mismo, es el cronograma suponiendo que los tiempos de propagación de todos los circuitos combinacionales y de los biestables es cero. Además, en el cronograma simplificado el tiempo, que es continuo, se considera discreto: dividido en ciclos.

Circuito secuencial síncrono. Lo definimos como un circuito secuencial (con memoria) en el que las señales de entrada y salida son síncronas y están sincronizadas con una señal de reloj, Clk, que también entra en el circuito, para indicar cuándo se deben actualizar los elementos de memorización internos.

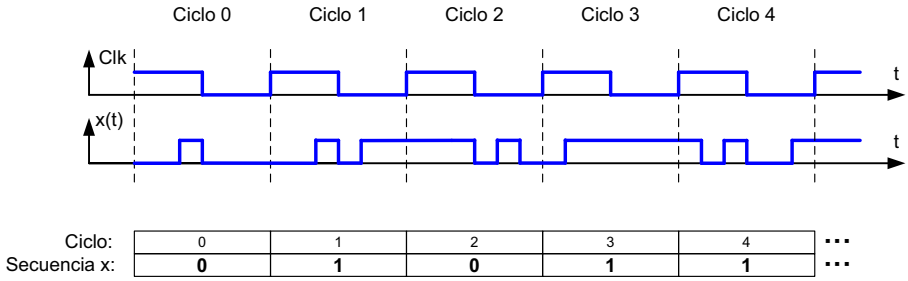


Fig. 6.5 Cronogramas de la señal x(t) *sincronizada con el reloj* Clk *y cronograma simplificado de la secuencia de bits de la señal síncrona x.*

Ya que nunca hablamos de circuitos secuenciales asíncronos, usaremos circuito secuencial como sinónimo de circuito secuencial síncrono.

6.1.5 El biestable D activado por flanco

Vamos a definir el símbolo y el funcionamiento del dispositivo secuencial más sencillo con el que construiremos otros circuitos secuenciales más complejos: el biestable D activado por flanco ascendente de reloj. Para nosotros, los biestables son a los circuitos secuenciales como las puertas a los combinacionales: los elementos más simples con los que construiremos el computador completo. Como hicimos con las puertas, no vamos a entrar en la construcción interna del biestable. Sólo decir que se puede construir con varias puertas realimentadas entre sí formando un ciclo cerrado (cosa que estaba prohibida en los combinacionales). En adelante, ya no repetiremos más lo de activado por flanco ascendente de reloj, pues siempre usamos este tipo de biestable, incluso podemos decir simplemente biestable (*Flip-Flop*), ya que sólo usamos el D.

Definición

Un biestable D es un dispositivo secuencial que consta de una entrada de datos D y una salida Q, ambas de 1 bit. Además, como todo circuito secuencial síncrono, tiene una entrada de reloj: Clk. Su función es muy simple, cuando la señal de reloj pasa de 0 a 1 (se produce un flanco ascendente de reloj), el valor que se encuentra en ese momento en la entrada D del biestable, aparece (por ahora digamos que inmediatamente) en la salida Q, y ese valor permanece estable en la salida hasta que se produzca otro flanco ascendente de reloj, aunque mientras tanto cambie el valor de la entrada D.

Símbolo. En la figura 6.6 se ve el símbolo del biestable. El nombre de la entrada y de la salida dentro del símbolo del biestable están en mayúsculas, como es usual en otros textos, en contra de nuestra norma de usar minúsculas al tratarse de señales de un bit cuando el nombre tiene una sola letra. La señal de reloj no lleva su nombre dentro del símbolo ya que es inconfundible (el indicador -|> hace referencia a que la señal de reloj actúa sobre el biestable en su flanco ascendente de reloj, ya que existen otras posibilidades que no estudiamos).

Cronograma. También se muestra en la figura 6.6 un cronograma que explica el funcionamiento del biestable para una forma de señal de entrada concreta. Como la entrada d vale cero al final del ciclo k-

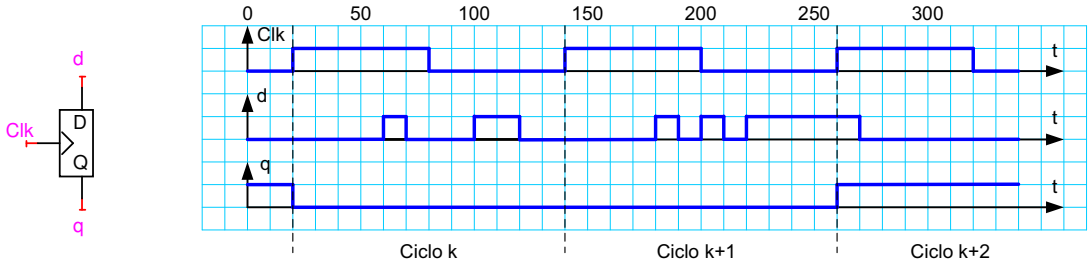


Fig. 6.6 Símbolo de un biestable D y cronograma de su comportamiento.

1, la salida q vale 0 durante todo el ciclo k aunque la entrada d varíe durante el ciclo k. Cuando llega el flanco ascendente de reloj que indica el final del ciclo k y el principio del k+1, la señal d vale 0 otra vez y por eso durante el ciclo k+1 la salida q continua valiendo 0. La señal d sigue variando, pero al final del ciclo k+1 vale 1, por lo que al ciclo siguiente, k+2, la salida q vale 1. A la vista del cronograma podemos decir que el biestable D almacena una muestra del valor de la entrada en el momento en que el reloj pasa de 0 a 1 y lo enseña en su salida de forma estable hasta que toma otra muestra.

Tiempo de propagación[1]

En realidad, al igual que ocurre con las puertas lógicas, el biestable no modifica su salida en el mismo instante de tiempo en que llega el flanco ascendente de reloj. Un biestable real tiene un tiempo de propagación distinto de cero. El **tiempo de propagación del biestable** es el tiempo desde que se produce el flanco ascendente de la señal de reloj hasta que la salida Q toma el valor que tenía la entrada D en el instante en que llegó el flanco.

Cronograma con tiempo de propagación. La figura 6.7 muestra el mismo cronograma que en la figura 6.6 pero suponiendo ahora que el tiempo de propagación del biestable es de 50 u.t. Con el origen de una flecha se indica el valor de la entrada d mientras se produce el flanco ascendente de reloj y con la punta de la flecha el cambio producido en la salida q después del tiempo de propagación del biestable. No se ha puesto flecha del ciclo *k* al *k*+1 ya que la salida no ha cambiado.

Ejemplo 1

La figura 6.8 muestra el símbolo de un biestable D (ligeramente distinto a los símbolos usados antes) y el cronograma con otra secuencia de valores de entrada. Ahora, el biestable tiene un tiempo de propagación Tp.

El biestable como retardador de secuencia de un ciclo

Al considerar las señales de entrada y salida del biestable como señales síncronas, podemos decir que la salida Q produce la misma secuencia de valores de la entrada D, pero retardada un ciclo. Esto se ve en el cronograma de la figura 6.7: la secuencia de valores de la entrada d desde el ciclo *k*-1 al *k*+1 es: 0,

1. Además hay otros parámetros temporales importantes en un biestable, como el tiempo de *start-up* y el tiempo de *hold*, que para simplificar no consideramos en este curso.

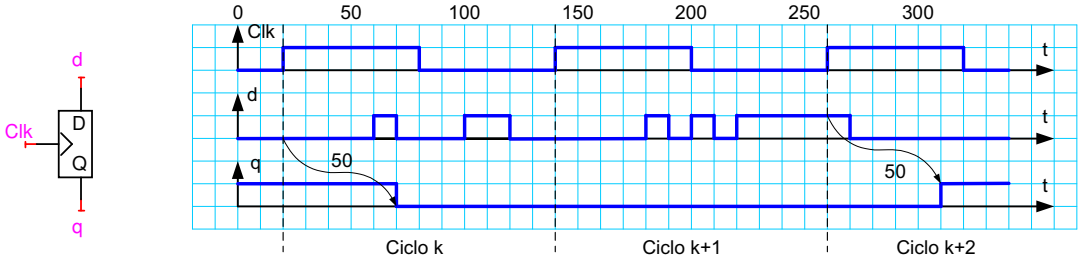


Fig. 6.7 Símbolo de un biestable D con un tiempo de propagación de 50 u.t. y cronograma de su comportamiento

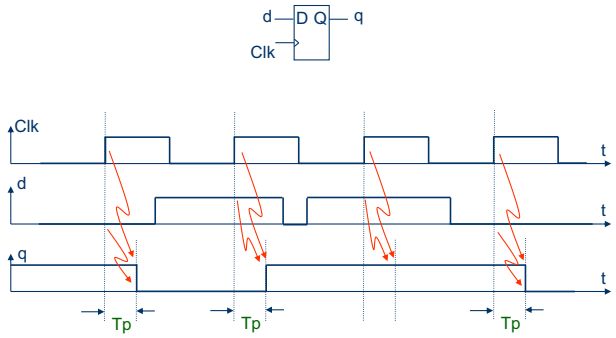


Fig. 6.8 Ejemplo de cronograma del comportamiento de un biestable D.

0, 1; y la secuencia de la salida q desde el ciclo *k* al *k*+2 es la misma 0, 0, 1. Aunque todavía se ve más cómodamente en el cronograma simplificado de la figura 6.9. Se ha sombreado el valor de la señal d en el ciclo k+2 porque en el cronograma de la figura 6.7 no se ve el valor de la señal al final de este ciclo. Por otro lado, aunque el valor de la señal q en el ciclo k-1 sí que se ve en el cronograma (es 1) se ha sombreado en el cronograma simplificado porque este valor es circunstancial, no importa para lo que se desea ver.

6.1.6 El registro

El biestable es un dispositivo de memoria de un bit, sin embargo en nuestro computador se procesan y almacenan palabras de 16 bits. Por ello es muy útil definir un dispositivo o bloque secuencial que sea

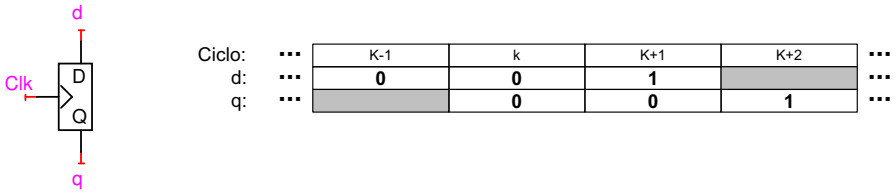


Fig. 6.9 Símbolo de un biestable D y cronograma simplificado del comportamiento como retardador de un ciclo de la secuencia: 001.

capaz de almacenar una palabra de *n* bits (para *n*>1). Este bloque se denomina registro. La figura 6.10 muestra el símbolo de un registro y el esquema interno para el caso general de *n* bits. Consta de *n* biestables D que comparten la misma señal de reloj.

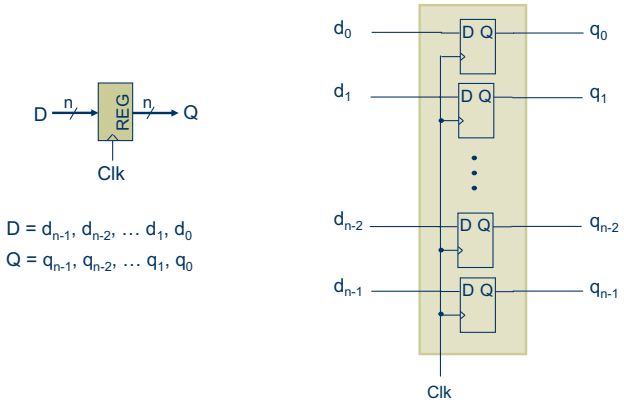


Fig. 6.10 Símbolo y esquema interno de un registro de n bits.

6.1.7 Reglas de interconexión de dispositivos combinacionales y biestables

Un circuito secuencial está formado por una red de dispositivos combinacionales y biestables (o registros) conectados entre sí siguiendo ciertas reglas, que vemos a continuación, para que el circuito sea válido. La figura 6.11 muestra un ejemplo.

Reglas de interconexión de las señales de datos. Las entradas del circuito tienen que conectarse con entradas de dispositivos combinacionales y/o entradas de biestables. Las salidas de combinacionales y biestables se tienen que conectar a entradas de combinacionales y/o biestables y/o a las salidas del circuito. Además, todas las entradas de todos los dispositivos tienen que tener un valor lógico (no pueden estar "al aire") y no se pueden conectar directamente dos salidas de dos dispositivos. No obstante, lo que acabamos de decir se cumple también para los circuitos combinacionales, como se comentó en el capítulo 2 (sin que aparezcan los biestables en el texto). ¿Cuál es la diferencia? La diferencia es que ahora, en los circuitos secuenciales: **sí que pueden producirse ciclos**, caminos cerrados que empiezan en un dispositivo y terminan en el mismo dispositivo. Pero hay una restricción:

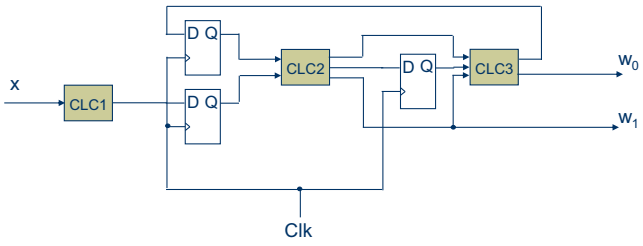


Fig. 6.11 Red de dispositivos combinacionales y biestables conectados entre sí formando un circuito secuencial.

el camino cerrado tiene que atravesar al menos un biestable o registro (de lo contrario estaríamos haciendo un ciclo en un circuito combinacional y eso está prohibido).

Reglas de interconexión de la señal de reloj. Siempre, en todos los circuitos secuenciales, incluido el computador, la señal de reloj (Clk) es una entrada al circuito que se conecta a todas las entradas de reloj de todos los biestables o registros del circuito. Tanto es así, que en ocasiones no dibujamos los cables de la señal de reloj, para no complicar el dibujo del esquema lógico, porque no aportan información ya que siempre están conectados de igual forma.

6.1.8 Tiempo de ciclo

Para terminar la introducción de este capítulo, vamos a hablar de lo que ocurre durante un ciclo en un circuito secuencial. Esto lo afinaremos más al final del capítulo, en la sección 6.6, pero es muy importante tener la idea clara cuanto antes.

Presentación del ejemplo. Imaginemos la parte de circuito secuencial que se muestra en la figura 6.12 y que consiste en dos registros conectados a través de un incrementador binario. El bus X es la entrada de esta parte del circuito y alimenta al primer registro mientras que el bus W es la salida del segundo registro y salida del circuito. Los registros tienen un tiempo de propagación (tiempo desde que llega el flanco ascendente de reloj hasta que el valor de la entrada pasa a la salida) de 40 u.t. y el incrementador combinacional tiene un tiempo de propagación de 100 u.t. (tiempo desde que la entrada está estable hasta que la salida se estabiliza al valor correcto, que es el valor de la entrada más uno). Por el bus X llega una secuencia de números que una vez incrementados aparecen por el bus W pasados dos ciclos de reloj.

El cronograma de la figura muestra lo que ocurre en el fragmento de circuito durante el ciclo k, aunque también se muestra el final del ciclo anterior y el principio del posterior. Por lo que se ve en el cronograma, en el ciclo k-2 en la entrada X se encontraba el número 57, en el ciclo k-1 el 367 y en el k el 24. Veamos en detalle qué ocurre desde el final del ciclo k-1.

Final del ciclo k-1 y principio del k. En el bus X se encuentra ya estable el valor 367 cuando llega el flanco ascendente de reloj que indica el final del ciclo k-1 y el principio del k. El registro lee el valor 367 de su entrada y pasadas las 40 u.t. del tiempo de propagación del registro, aparece el número 367 en la salida del registro, Y, que es la entrada del incrementador.

Durante el ciclo k. El incrementador empieza a realizar los cálculos. Como se comentó en el capítulo de combinacionales, debido a los tiempos de propagación distintos desde cada entrada a cada salida del incrementador combinacional, si miramos los *n* bits del bus de salida Z podemos ver como estos bits codifican distintos números conforme pasa el tiempo: 326, 367, 366, 364, 360, 362 y finalmente, pasadas las 100 u.t. del tiempo de propagación del incrementador, se estabiliza la salida Z con el valor 368, que es el correcto. Los valores intermedios incorrectos no se han escrito en el cronograma para que resulte más claro. Como durante el tiempo de inestabilidad no hay valores correctos en Z, estos se han marcado con equis, o con señales que suben y bajan, ya que los valores concretos no nos interesan. Mientras ha ocurrido esto, en el mismo ciclo *k* se ha estabilizado otro número en el bus X: el 24.

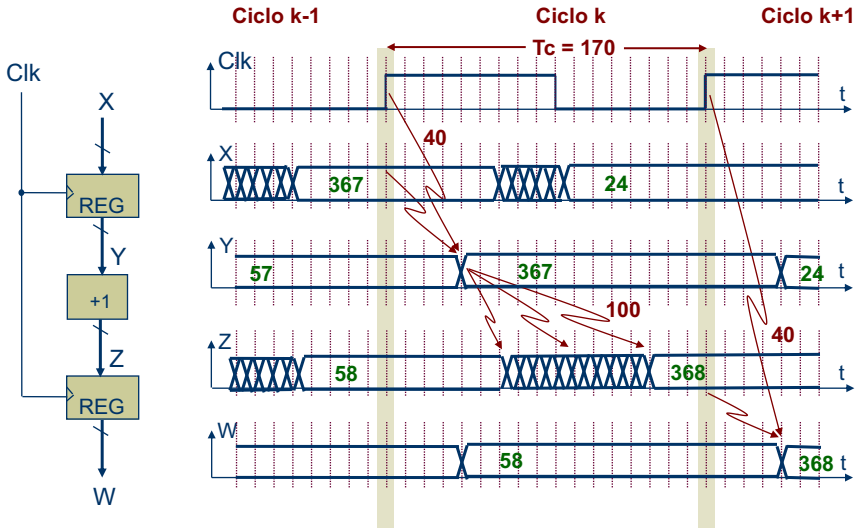


Fig. 6.12 ¿Qué ocurre durante un ciclo?

Final del ciclo k y principio del k+1. Según se ve en el cronograma, pasadas 30 u.t. desde que quedó estable la salida del incrementador, llega el flanco ascendente de reloj que marca el final del ciclo *k* y el principio del *k*+1. En este momento, el valor 368, que está en la entrada del segundo registro, es leido por este para aparecer en su salida, W, al cabo de las 40 unidades del tiempo de propagación del biestable. En este momento también aparece el número 24 en la salida del primer biestable, ya que es el valor que había en X cuando se produjo el flanco. En este ciclo *k*+1 se calculará en el incrementador el número 25 y así sigue el funcionamiento...

Conclusión del ejemplo. Es importante comprender que si el tiempo de ciclo de este circuito fuera menor que 140 u.t. por W no saldría la secuencia de números que entra por X incrementandose cada número en una unidad. Saldrían otros números, porque el segundo registro leería, cogería, valores incorrectos de la salida del incrementador. Sin embargo, el tiempo de ciclo del sistema es 170, que como vemos es un poco superior a 140, con lo que es seguro que cuando llegue el flanco de fin de ciclo, la salida del incrementador estará estable al valor correcto.

Conclusión general. Esto, que acabamos de ver en detalle para una parte de un circuito secuencial, está ocurriendo a la vez en todos los caminos que hay desde la salida de un biestable hasta la entrada de otro biestable del circuito completo, pasando por los circuitos combinacionales que hay entre los dos biestables. Por ello, *el tiempo de ciclo del circuito secuencial tiene que ser mayor que el tiempo que las señales tardan en atravesar el camino más largo (en tiempo) que va desde la salida de un biestable hasta la entrada de otro biestable, pasando por circuitos combinacionales (no atravesando ningún biestable, ya que cuando llega a uno termina el camino). Este camino se llama camino crítico del circuito secuencial. El tiempo de ciclo tiene que ser mayor que el tiempo del camino crítico del circuito*. Para encontrar el camino crítico de un circuito secuencial (o los caminos críticos, pues puede haber varios con el mismo tiempo máximo), si no nos dicen lo contrario, supondremos que cada

entrada del circuito viene de la salida de un biestable y cada salida del circuito va a parar a la entrada de otro biestable. En la sección 6.6 profundizamos en este importante tema.

6.2 Estructura general de un circuito secuencial

6.2.1 Modelo de Mealy

Esquema lógico con un sólo bloque combinacional. En la sección 6.1.7 hemos dicho que un circuito secuencial consistía en la interconexión de dispositivos combinacionales y biestables siguiendo unas reglas. La figura 6.13 muestra el esquema general de un circuito secuencial que se obtiene agrupando todos los dispositivos combinacionales en un solo bloque (CLC) y todos los biestables en otro que se denomina registro de estado del circuito (REG). La salida Q de este registro se denomina estado actual (información que contiene la memoria del circuito en el ciclo actual), o simplemente estado, mientras que la entrada D de REG se denomina estado siguiente, Q^+, porque será el estado al ciclo siguiente. Como se ve, tanto la salida (W) como el estado siguiente (Q^+) son funciones lógicas ($\mathcal{H}$ y $\mathcal{G}$ respectivamente) de la entrada (X) y del estado actual (Q).

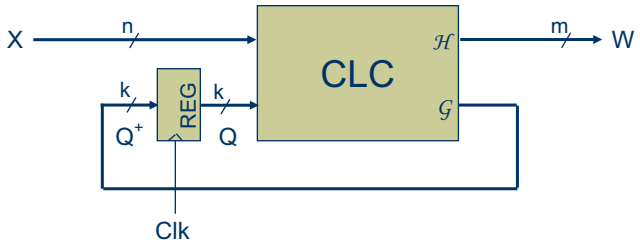


Fig. 6.13 Estructura general (modelo de Mealy) de circuito secuencial con un solo bloque combinacional.

Esquema con dos bloques combinacionales. También se puede descomponer el único bloque con todos los dispositivos combinacionales en dos bloques, uno que calcula las salidas del circuito (funciones $\mathcal{H}$) y otro que calcula el estado siguiente (funciones $\mathcal{G}$). En principio, las salidas de los dos bloques son función de las entradas al circuito y del estado actual, como se ve en la figura 6.14. Esta estructura (y la de la figura 6.13) se denomina modelo de Mealy, para diferenciarlo del modelo de Moore que es un caso particular de este en el que las salidas son solamente función del estado, y que estudiamos en la sección 6.2.2.

Tabla de transiciones y tabla de salida

Especificación lógica de un secuencial. Al observar la estructura a bloques de un circuito secuencial general (de Mealy) podemos decir que queda perfectamente especificado si nos indican lo siguiente:

- Cuántas variables de entrada (n) tiene el circuito y su nombre (X), cuántas variables tiene de salida (m) y su nombre (W), cuántos bits de estado tiene el circuito (k) (el nombre no hace falta ya que por

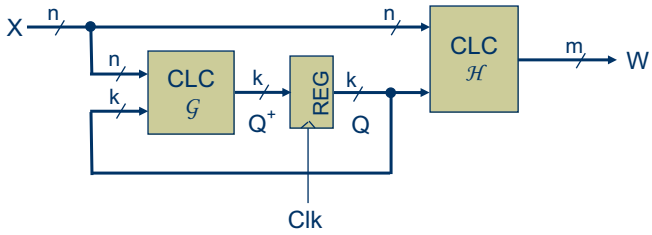


Fig. 6.14 Estructura general (modelo de Mealy) de circuito secuencial con dos bloques combinacionales, uno para las salidas y otro para el estado siguiente.

convenio siempre denominamos al estado actual Q, que es la salida de los biestables o del registro de estado, y al estado siguiente Q^+, que es la entrada de datos de los biestables).

- Las dos tablas de verdad de cada uno de los dos bloques combinacionales: la **tabla del estado siguiente**, o **tabla de transiciones**, que especifica Q^+ en función del estado actual Q y las entradas X y la **tabla de salida** que especifica la salida W en función del estado actual y la entrada.
- El **estado inicial** del circuito: el estado que tienen los biestables cuando se pone en marcha el circuito, el estado durante el primer ciclo de funcionamiento del circuito (veremos la importancia de esto más adelante).

Ejemplo de especificación. La figuras 6.15 y 6.16 muestran un ejemplo de especificación de un circuito secuencial mediante las dos tablas de verdad. Los nombres de las entradas y salidas del circuito y el número de bits se ven en las figuras. El estado inicial (que no se indica en las figuras) es: q_1=0 y q_0=0. En la primera figura se muestra la tabla de transiciones y, de forma resaltada, la parte del circuito encargada de implementarla; y en la segunda se muestra la tabla de salida y la parte del circuito correspondiente. Nos resulta más claro y más cómodo poner siempre las variables de estado como variables de más peso (más a la izquierda) en las tablas de verdad y las variables de entrada como las de menos peso (a la derecha). Esto sería correcto de cualquier otra forma, incluso mezclando las variables de entrada y las de salida, pero podría dificultar el trabajo y aumentar la probabilidad de cometer algún error.

Estados inalcanzables. Aunque el ejemplo que nos ocupa tiene dos biestables y por lo tanto puede tener 4 estados diferentes, el estado Q=10 nunca se va a dar. Si observamos la tabla del estado siguiente de la figura 6.15 y sabiendo que el estado inicial del circuito es el q_1=0 y q_0=0, Q=00, sea cual sea la secuencia de entrada el circuito nunca llegará a estar en el estado q_1=1 y q_0=0, Q=10: del estado 00 puede ir al 00, al 01 o al 11, dependiendo de las entradas, pero nunca al 10, lo mismo pasa cuando el circuito está en los estados 01 o 11. Por eso, en la tabla de transiciones no se ha especificado cuál es el estado siguiente cuando el circuito está en el estado 10, porque nunca estará en ese estado. Esto se indica con valores x en las salidas de la tabla q_1^+ y q_0^+, que dicen que no importa el estado siguiente.

$Q^+ = \mathcal{G}(Q, X)$:

$q_1 q_0 x_1 x_0$	$q_1^+ q_0^+$
0 0 0 0	1 1
0 0 0 1	0 1
0 0 1 0	0 0
0 0 1 1	0 1
0 1 0 0	1 1
0 1 0 1	0 1
0 1 1 0	0 0
0 1 1 1	0 1
1 0 0 0	x x
1 0 0 1	x x
1 0 1 0	x x
1 0 1 1	x x
1 1 0 0	0 0
1 1 0 1	0 0
1 1 1 0	1 1
1 1 1 1	0 1

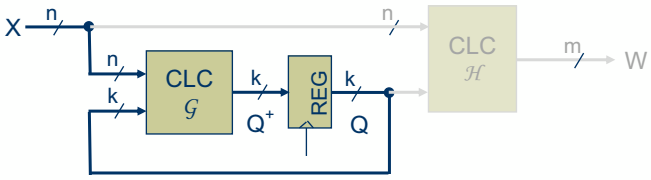


Fig. 6.15 Ejemplo de especificación de un circuito secuencial con dos bits de entrada (x_1, x_0), dos de estado (q_1, q_0) y dos de salida (w_1, w_0) mediante las dos tablas de verdad. En esta figura tabla de transiciones y circuito que la implementa.

$W = \mathcal{H}(Q, X)$:

$q_1 q_0 x_1 x_0$	w_1 w_0
0 0 0 0	1 0
0 0 0 1	1 1
0 0 1 0	1 1
0 0 1 1	1 0
0 1 0 0	0 1
0 1 0 1	1 0
0 1 1 0	1 1
0 1 1 1	0 0
1 0 0 0	x x
1 0 0 1	x x
1 0 1 0	x x
1 0 1 1	x x
1 1 0 0	1 0
1 1 0 1	1 0
1 1 1 0	0 1
1 1 1 1	0 1

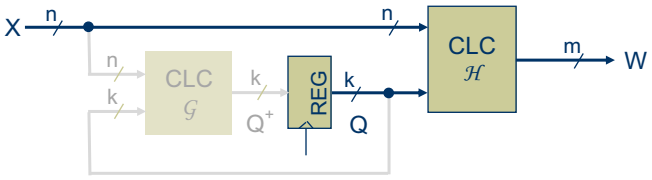


Fig. 6.16 Ejemplo de especificación de un circuito secuencial con dos bits de entrada (x_1, x_0), dos de estado (q_1, q_0) y dos de salida (w_1, w_0) mediante las dos tablas de verdad. En esta figura tabla del estado siguiente y circuito que la implementa.

6.2.2 Modelo de Moore

Estructura a bloques. Un circuito secuencial de Moore es un caso particular del caso general de Mealy. En un circuito de Moore las salidas en cada ciclo son solamente función del estado actual del circuito (no son función del valor de las entradas en ese ciclo). La figura 6.17 muestra el esquema general con los dos bloques combinacionales: el del estado siguiente (esto es igual que el caso de

Mealy) y el combinacional que calcula las salidas del circuito, W, que ahora es sólo función del estado actual, Q (la salida de los biestables de estado o registro de estado).

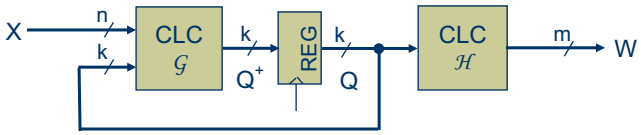


Fig. 6.17 Estructura a bloques de un circuito secuencial de Moore.

Especificación mediante tablas. Para especificar el funcionamiento de un circuito de Moore hay que dar la misma información que para uno de Mealy; la única diferencia es que la tabla de salida es más pequeña, tiene menos filas: las salidas son sólo función del estado. La figura 6.18 muestra un ejemplo de especificación de un circuito (que no hace la misma función que el de Mealy de la sección anterior). El estado inicial no se indica en la figura, pero es el $q_1q_0 = 00$. En este caso el circuito no se encontrará nunca en el estado 11, por lo que la tabla de transiciones y la de salidas indican que en caso de estar en el estado 11 el estado siguiente y las salidas pueden tomar cualquier valor.

$Q^+ = \mathcal{G}(Q, X)$:

q_1	q_0	x_1	x_0	q_1^+	q_0^+
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	x	x
1	1	0	1	x	x
1	1	1	0	x	x
1	1	1	1	x	x

$W = \mathcal{H}(Q)$:

q_1	q_0	w_1	w_0
0	0	0	0
0	1	0	1
1	0	1	1
1	1	x	x

Fig. 6.18 Ejemplo de especificación de un circuito secuencial de Moore con dos bits de entrada (x_1, x_0), dos de estado (q_1, q_0) y dos de salida (w_1, w_0) mediante las dos tablas de verdad.

6.3 Grafo de estados

Veamos la forma de especificar el comportamiento lógico de un circuito secuencial mediante un grafo, que es una representación mucho más amigable que las tablas de transiciones y de salida. Nos centramos en el caso de Moore usando como ejemplo el circuito especificado mediante las tablas de la figura 6.18. Sólo al final, en la sección 6.3.4, hacemos algún comentario sobre los grafos de estado para el caso de Mealy, ya que no los usaremos en este curso.

6.3.1 La tabla de transiciones en el grafo

El grafo de estados representa la información de la tabla de transiciones: cuál es el estado siguiente para cada uno de los posibles estados y para cada una de las posibles combinaciones de los valores de las entradas.

Los nodos

El número de nodos del grafo es igual al número de estados distintos en los que puede encontrarse el circuito. Cada nodo representa un estado distinto. Un circuito con k biestables puede codificar hasta 2^k estados diferentes, por lo que el grafo puede tener 2^k nodos. Puede tener menos de 2^k nodos si, debido a las particularidades del circuito, hay combinaciones de estos k bits que no se dan nunca. Así que el grafo del ejemplo que nos ocupa, especificado mediante la tabla de transición de la figura 6.18, tiene tres nodos, para los estados 00, 01 y 10, ya que nunca se encontrará en el estado 11.

Cada nodo se representa mediante un círculo, en cuyo interior (en la mitad superior) se indica la codificación del estado que representa el nodo (en nuestro ejemplo los valores concretos de q_1 y q_0). El nodo del estado inicial se representa con un doble círculo. En la figura 6.19 se muestran los 3 nodos del grafo de nuestro ejemplo (con una flecha que explicamos a continuación, pero faltan muchas más).

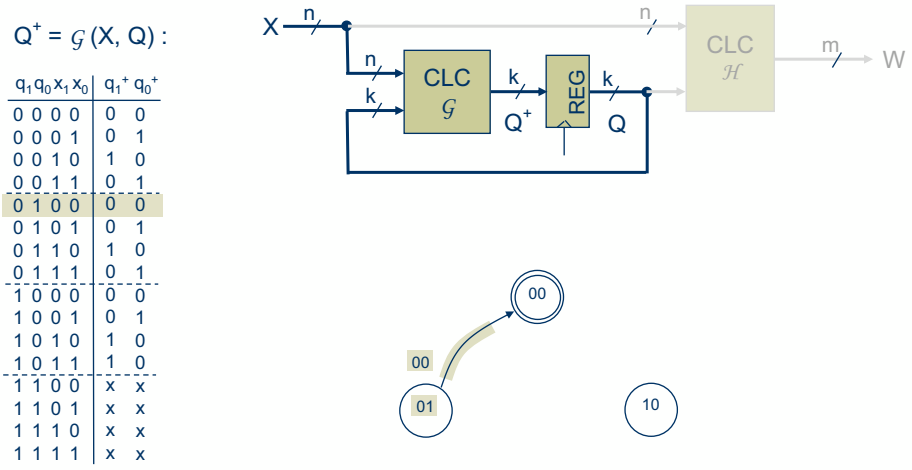


Fig. 6.19 Tabla de transiciones, esquema a bloques encargado de implementarla y grafo incompleto que representará, cuando esté completo, la misma información que la tabla de transiciones. Se ha sombreado una entrada de la tabla y el arco del grafo que la representa.

Los arcos

En un grafo de estados, de cada nodo salen tantos arcos dirigidos, flechas, como el número de posibles combinaciones de valores de las entradas. Así, como el circuito del ejemplo tiene 2 bits de entrada, de cada nodo deben salir 4 flechas, una para cada combinación de los bits de entrada: 00, 01, ... Cada arco se etiqueta con la combinación de entradas a que hace referencia. La punta de la flecha va a parar al

nodo del estado que tendrá el circuito al siguiente ciclo (estado siguiente), siempre y cuando en el ciclo actual las entradas tomen la combinación de bits que indica la etiqueta del arco.

En la figura 6.19 se muestran los 3 nodos del grafo de nuestro ejemplo y el arco relativo a la entrada 4 de la tabla de transiciones. Esta entrada de la tabla, que se indica con un fondo gris en la figura, dice: "si estamos en un ciclo (ciclo actual) en el que el circuito está en el estado 01 (q_1=0 y q_0=1) y las entradas en este ciclo valen 00 (x_1=0 y x_0=0), el circuito combinacional que calcula el estado siguiente genera en sus salidas los valores q_1^+=0 y q_0^+=0, por lo que al ciclo siguiente el circuito secuencial pasará al estado 00". Esta transición se muestra en la figura con la flecha etiquetada con 00.

De la tabla al grafo y del grafo a la tabla

De la misma forma que hemos puesto un arco en el grafo se procede con todas las filas de la tabla de transiciones. En la figura 6.20 a) se ve el grafo anterior al que se le ha añadido además el arco relativo a la entrada 5 de la tabla de verdad: si estamos en el estado 01 y la entrada vale 01, el estado siguiente será el 01. Por último en la parte b) de la figura se muestra el grafo completo. Podemos observar que en los casos que de un nodo salen varios arcos hacia el mismo nodo se ha dibujado sólo uno que representa a todos ellos, pero se ha etiquetado con todas las combinaciones de valores de entrada de las transiciones que representa. Este es el caso de las dos transiciones del estado 01: tanto si la entrada vale 01 como si vale 11 el estado siguiente es el 01.

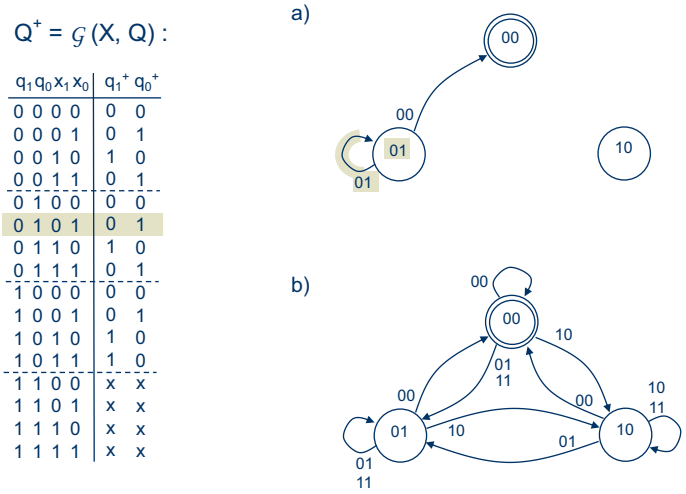


Fig. 6.20 Tabla de transiciones. a) Grafo incompleto al que se le ha añadido, a partir del grado de la figura 6.19, el arco relativo a la fila 5 de la tabla de transiciones (marcado con sombra gris). b) Grafo completo.

Al poner los arcos en el grafo es razonable empezar por la fila cero de la tabla e ir avanzando hasta completar el grafo. Haremos algún ejercicio de esto en la sección 6.6, como parte del **análisis** de un circuito secuencial: a partir del esquema lógico obtener las tablas de transiciones y de salida y de estas el grafo de estados. También haremos lo contrario como uno de los pasos de la **síntesis** de circuitos

secuenciales: a partir del grafo obtener la tabla de transiciones. En definitiva, es importante entender la relación entre las dos formas de especificar la parte combinacional que calcula el estado siguiente.

6.3.2 La tabla de salida en el grafo

Dado que para el caso de Moore la tabla de salida es muy sencilla (las salidas sólo dependen del estado), la información de la tabla puede integrarse fácilmente en el grafo. Solamente se tiene que asociar a cada estado (cada nodo del grafo) la salida que produce. Para ello se diferencian dos zonas en el interior de cada nodo: arriba se escribe la codificación del estado a que corresponde el nodo, como ya hemos visto, y abajo se indican los valores de las salidas cuando el circuito está en ese estado.

El sombreado de la figura 6.21 muestra cómo se pone en el grafo la información de la fila 2 de la tabla de salida: en el ciclo en el que el estado es 10 (q_1=1, q_0=0) las salidas toman el valor 11 (x_1=1, x_0=1).

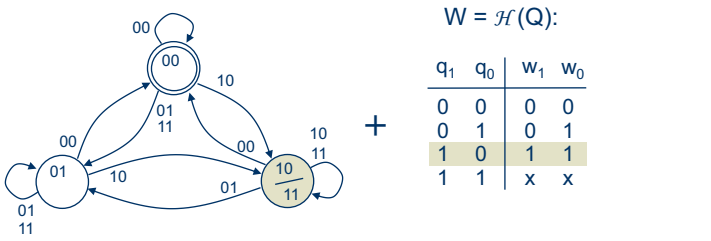


Fig. 6.21 Especificación de la fila 2 de la tabla de salidas (marcada con sombra gris) en el nodo correspondiente.

La figura 6.22 muestra el grafo de estados completo, que contiene la información de las tablas de transición y de salida del ejemplo de la figura 6.18.

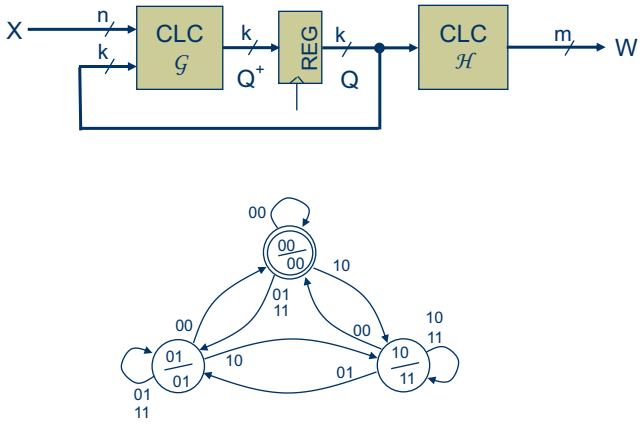
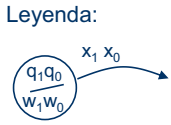


Fig. 6.22 Especificación completa del circuito de Moore del ejemplo de la figura 6.18 mediante un grafo de estados.

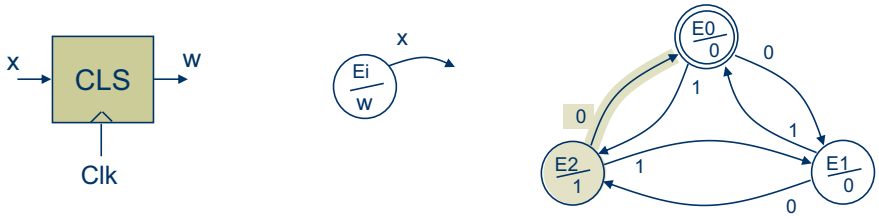
La leyenda del grafo. Dado que al eliminar las tablas de verdad perdemos los nombres de las variables y su orden en las tablas, dar sólo el grafo de estados es incompleto. Por ello, para especificar correctamente un circuito secuencial de Moore (lo mismo ocurre para Mealy) hay que indicar en qué orden codificamos los bits de las entradas y salidas en el grafo (también indicamos el nombre de los bits de estado, aunque esto no es estrictamente necesario). Esta información se indica en un nodo del que sale una flecha, pero dispuesto aparte del grafo de estados. A este nodo y esta flecha se les etiqueta con el nombre de las variables en vez de con valores de bits concretos, que es como se hace en el grafo de estados. Así, la leyenda del grafo de la figura 6.22 se muestra en el margen derecho de este párrafo.



6.3.3 Seguimiento de grafos de estados

Un tipo de ejercicios que ayuda a entender la información que hay en un grafo consiste en obtener un cronograma simplificado que indique las señales del estado y las de salida en cada ciclo a partir del grafo de estados completo del circuito, del estado inicial, y del valor de las entradas en cada ciclo.

Veamos un ejemplo. La figura 6.23 muestra el símbolo de un circuito secuencial de una entrada x y una salida w junto con el grafo de estados que define su comportamiento. El nombre de los estados del grafo, E0, E1 y E2, no se ha codificado, por lo que se usan estos mismos nombres en el cronograma. Vamos a completar el cronograma simplificado de la figura que muestra el comportamiento del circuito para una secuencia concreta de valores de la entrada, una vez estabilizados.



Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
Estado	E0	E2	E0	E1	E2	E0	E1	E0	E2	E1					
Entrada x	1	0	0	0	0	0	1	1	1	0	0	1	0	1	0
Salida w	0	1	0	0	1	0	0	0	1	0					

Fig. 6.23 Símbolo de un bloque secuencial, leyenda del grafo y grafo de estados de Moore del circuito. Cronograma simplificado para una secuencia de entradas concreta .

Para mostrar cómo se completa este cronograma fijémonos en el ciclo 4, por ejemplo. La información de lo que ocurre en este ciclo se ha marcado en gris tanto en la tabla del cronograma como en el grafo. En el ciclo 4 el circuito está en el estado E2 y en la entrada x hay un 0. Estos son los datos que nos dan. El grafo nos dice que estando en el estado E2 la salida que produce el circuito es 1, por lo que podemos completar este dato en el cronograma. El grafo también nos dice que si estando en el estado E2 la entrada vale 0, al ciclo siguiente se pasa al estado E0 (durante el ciclo actual se calcula el estado

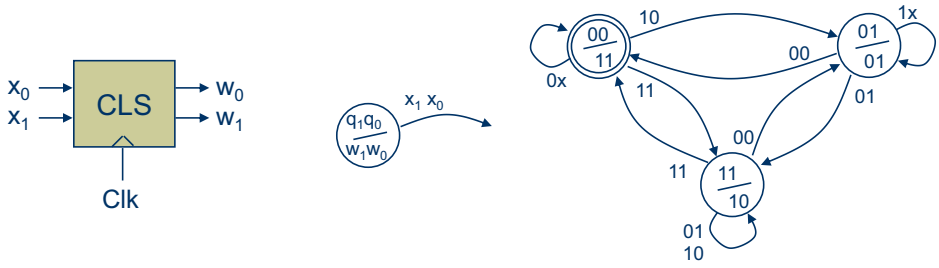
siguiente E0). Este estado se puede completar en la tabla y ponerlo en el ciclo 5 (esta información también se ha sombreado). Siguiendo este proceso se completa toda la tabla de izquierda a derecha. El cronograma de la figura se ha dejado incompleto para que completéis vosotros el estado y la salida para los ciclos del 10 a 14. La solución se muestra en el cronograma de la figura 6.24.

Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
Estado	E0	E2	E0	E1	E2	E0	E1	E0	E2	E1	E2	E0	E2	E0	E2
Entrada x	1	0	0	0	0	0	1	1	1	0	0	1	0	1	0
Salida w	0	1	0	0	1	0	0	0	1	0	1	0	1	0	1

Fig. 6.24 Cronograma simplificado que completa al de la figura 6.23.

Ejemplo 2

Es como el ejemplo anterior, con la diferencia que el circuito tiene 2 entradas (x_1, x_0) y dos salidas (w_1, w_0) y en este grafo se han codificado los estados con dos bits (q_1,q_0). Hay que completar la tabla del cronograma de la figura 6.25: estado y salidas para los ciclos del 5 al 14. La solución se muestra en la figura 6.26.



Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
Estado q_1q_0	00	00	00	11	11										
Entrada x_1x_0	00	01	11	10	01	00	00	11	10	01	11	01	00	10	10
Salida w_1w_0	11	11	11	10	10										

Fig. 6.25 Símbolo de un bloque secuencial, leyenda del grafo y grafo de estados de Moore del circuito. Cronograma simplificado incompleto.

6.3.4 Grafos de estados para circuitos de Mealy

Completamos el tema de los grafos de estado con el caso de Mealy, aunque en este curso no vamos a trabajar con estos circuitos. Lo hacemos porque es importante conocer la estructura y la especificación del caso general de un circuito secuencial. El grafo sirve para especificar la tabla de transiciones de un secuencial tanto si es de Moore como si es de Mealy, ya que en ambos casos el estado siguiente es

Núm. Ciclo	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Estado q_1q_0	00	00	00	11	11	11	01	00	11	11	11	00	00	00	01
Entrada x_1x_0	00	01	11	10	01	00	00	11	10	01	11	01	00	10	10
Salida w_1w_0	11	11	11	10	10	10	01	11	10	10	10	11	11	11	01

Fig. 6.26 Cronograma simplificado solución del ejercicio 2.

función de las entradas y del estado actual. No obstante, no ocurre lo mismo con la tabla de salida, que tiene diferente estructura, es más compleja, en el caso general de Mealy que en el particular de Moore. Para Mealy las salidas son función no solamente del estado sino también de las entradas. Esto hace que no podamos asociar cada nodo con un valor concreto de las salidas.

En el caso de Mealy, el valor de las salidas se asocian a los arcos ya que cada arco es función del estado actual y del valor de las entradas en ese ciclo[1]. Aunque este tipo de representación se usa normalmente en otros textos, no la usaremos aquí. Siempre podemos usar el grafo de estados tal como lo hemos visto en las secciones anteriores para representar solamente la tabla de transiciones y al lado del grafo indicar la tabla de salidas Esto se hace en la figura 6.27 para el ejemplo de circuito de Mealy que se especificó con las dos tablas en la sección 6.2.1 (figuras 6.15 y 6.16). A partir de este momento, si no se dice lo contrario, siempre que hablemos de circuitos secuenciales y de grafos de estados que los especifican nos referimos al caso de Moore.

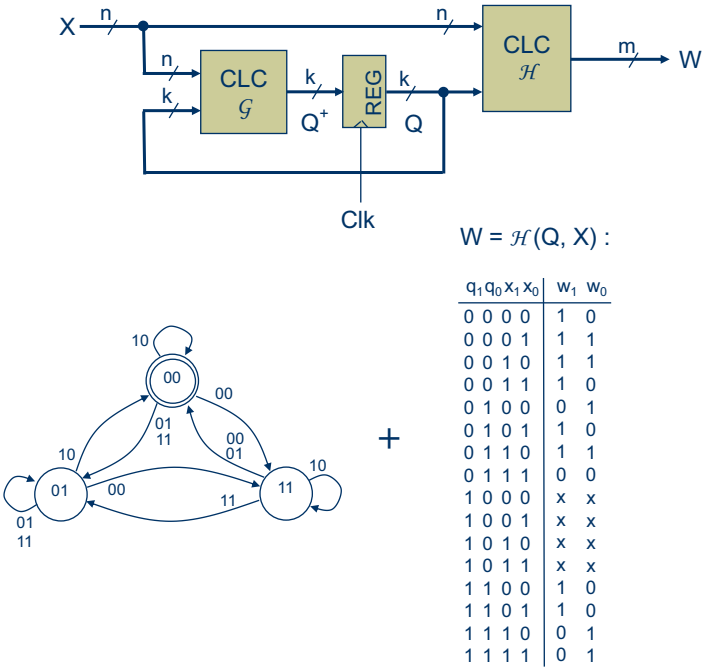
6.4 Análisis lógico

Diferenciamos dos partes en el análisis de un circuito secuencial, al igual que hicimos con el análisis de un combinacional: análisis lógico y análisis temporal. Los dos tipos de análisis parten del esquema lógico del circuito. El análisis lógico consiste en encontrar el grafo de estados que especifica la funcionalidad del circuito y el temporal consiste en encontrar el tiempo de ciclo mínimo que hace que el circuito realice correctamente la funcionalidad descrita por el grafo. Vamos a usar el ejemplo de la figura 6.28 para realizar los dos tipos de análisis. No obstante, el análisis temporal lo dejamos para el final del capítulo (sección 6.6)

Nombre de las señales

El primer paso consiste en identificar los biestables de estado del circuito y las entradas y las salidas del circuito, para saber cuántos hay y sus nombres. Los nombres de las entradas y salidas del circuito normalmente nos las darán junto con el esquema (supongamos que son x_{n-1}, ..., x_1, x_0 y w_{m-1}, ..., w_1, w_0 respectivamente), pero los nombres de los bits de estado no tienen porqué estar especificados en el circuito. Si no lo están, debemos darles nombre, por ejemplo: q_{k-1}, ..., q_1, q_0). La asignación del

1. Dado que en un circuito de Mealy, la salida es función del estado actual (nodo) y del valor de las entradas durante ese ciclo (arco), podemos asociar a cada arco que sale de un nodo otra etiqueta con los valores de las salidas que corresponden al nodo del que sale la flecha y a las entradas que representa esa flecha. Por ejemplo, en vez de etiquetar un arco con x_1x_0 lo etiquetaríamos con x_1x_0/w_1w_0. Ahora solo se podrían fundir varias flechas si coinciden las etiquetas de las entradas y salidas (además del nodo origen y destino). Esto hace que en general aparezcan más flechas y complica bastante el grafo.



q_1 q_0 x_1 x_0	w_1	w_0
0 0 0 0	1	0
0 0 0 1	1	1
0 0 1 0	1	1
0 0 1 1	1	0
0 1 0 0	0	1
0 1 0 1	1	0
0 1 1 0	1	1
0 1 1 1	0	0
1 0 0 0	x	x
1 0 0 1	x	x
1 0 1 0	x	x
1 0 1 1	x	x
1 1 0 0	1	0
1 1 0 1	1	0
1 1 1 0	0	1
1 1 1 1	0	1

Fig. 6.27 Esquema a bloques de un circuito de Mealy con el grafo de estados y la tabla de salidas que lo especifican completamente.

nombre a la salida de cada biestable es arbitraria. También hay que dar nombre a las entradas de los biestables de estado, que definen el estado siguiente: la entrada D del biestable cuya salida está etiquetada con q_i se denomina q_i^+. El ejemplo que nos ocupa tiene una señal de entrada x, una de salida w y dos bits de estado (que ya están etiquetados) q_1 y q_0.

¿Circuito de Mealy o de Moore?

Una vez hecho esto debemos identificar si el esquema lógico sigue una estructura de Mealy (figura 6.13 o figura 6.14) o una estructura de Moore (figura 6.17). El ejemplo es de Moore, porque la salida w es solamente función del estado, pero no de la entrada. Aquí solo tratamos casos de Moore, pero es conveniente comprobarlo antes de continuar.

Obtención de la tabla de transiciones y la de salida

Ahora debemos hacer el análisis lógico de los dos circuitos combinacionales que definen el circuito secuencial, tal como se hizo en el capítulo 2. La tabla de transiciones define el funcionamiento de la parte del circuito encargada de generar el estado siguiente, q_{k-1}^+, ..., q_1^+, q_0^+, en función del estado actual, q_{k-1}, ..., q_1, q_0 y de las entradas x_{n-1}, ..., x_1, x_0. La tabla de salidas define la parte del circuito que genera las salidas, w_{n-1}, ..., w_1, w_0 en función del estado q_{k-1}, ..., q_1, q_0.

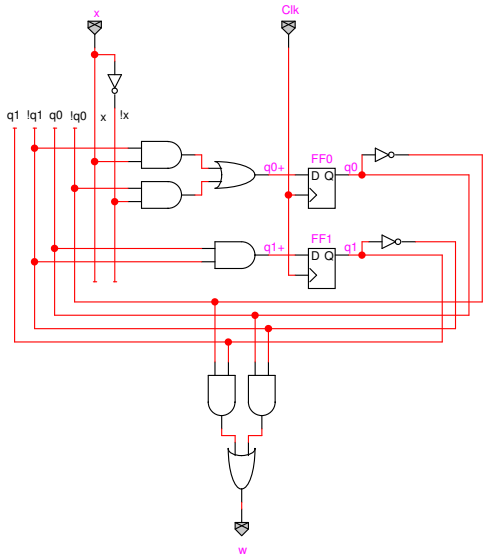


Fig. 6.28 Esquema lógico del circuito secuencial a analizar.

Las funciones lógicas que definen el estado siguiente y la salida en nuestro ejemplo se muestran en la figura 6.29 (expresiones lógicas y tablas de transiciones y de salida).

a) Tabla de transiciones:

q_1	q_0	x	q_1^+	q_0^+
0	0	0	0	1
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

$q_1^+ = !q_1 \cdot q_0$

$q_0^+ = !q_1 \cdot x + !q_0 \cdot !x$

b) Tabla de salidas:

q_1	q_0	w
0	0	0
0	1	1
1	0	1
1	1	0

$w = q_1 \cdot !q_0 + !q_1 \cdot q_0$

Fig. 6.29 Tabla de transiciones y de salidas resultante del análisis lógico del circuito de la figura 6.51.

Obtención del grafo de estados

Para terminar el análisis lógico debemos dibujar el grafo de estados, que contiene la misma información que las tablas de transiciones y de salida y que define el funcionamiento del circuito de una forma más clara para nosotros que las dos tablas de verdad. Con esto termina el análisis ya que no podemos ponerle un texto que describa el funcionamiento del circuito: puede haber infinidad de textos que describen el funcionamiento de un circuito secuencial con un único grafo.

La relación entre las tablas y el grafo de estados lo vimos al describir los grafos en la sección 6.3.1 (para la tabla de transiciones) y en la sección 6.3.2 (para la tabla de salidas), por lo que presentamos directamente el resultado para nuestro ejemplo en la figura 6.30. El estado inicial, que se indica en el grafo con un doble círculo, debe estar indicado en el esquema lógico. En nuestros ejemplos se indica con la notación de LogicWorks, pero sólo cuando el estado inicial de un biestable es 1. Cuando es 0 no lo indicamos, ya que este es el valor de estado inicial que tienen por defecto los biestables de nuestra biblioteca. En el esquema lógico del ejemplo no se indica nada, por lo que suponemos que el estado inicial es el 00.

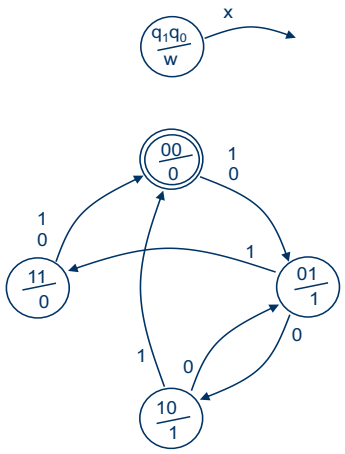


Fig. 6.30 Grafo de estados resultante del análisis lógico del circuito de la figura 6.51.

6.5 Síntesis de circuitos secuenciales

Para nosotros el proceso de síntesis de circuitos secuenciales parte de la descripción funcional del circuito y termina con el esquema lógico del circuito que lo implementa. Descomponemos el proceso en tres pasos:

1. De la descripción funcional al grafo de estados.
2. Del grafo de estados a las tablas de transiciones y de salidas.
3. De las tablas de transiciones y de salidas al esquema lógico.

De estos tres pasos el primero es posiblemente el más complejo; el segundo tiene que ver con lo que hemos hecho en la sección 6.3.1 y 6.3.2 al pasar de las tablas de transiciones y de salida al grafo de estados, pero ahora en sentido contrario; y el tercero es exactamente igual a loque hicimos en el capítulo 2 al sintetizar circuitos combinacionales: implementar las tablas de verdad (de transiciones y de salida) en suma de minterms, con un decodificador y puertas Or, o con una ROM (aunque ahora veremos una nueva estructura que implementa las dos tablas con una sola ROM y un multiplexor de buses). Por ello nos centramos en el primer paso, mientras que de los dos últimos solamente veremos un ejemplo.

6.5.1 De la descripción funcional al grafo de estados

La descripción o especificación del funcionamiento de un circuito secuencial se suele hacer mediante un texto escrito de manera informal o también, más formalmente, mediante un pseudocódigo en un lenguaje de alto nivel, usando fórmulas matemáticas, etc. Además, se puede completar la descripción indicando el funcionamiento para una secuencia concreta de valores de entrada mediante cronogramas simplificados (que indican el valor de las señales en cada ciclo una vez estabilizadas). En algunas ocasiones se pide completar un cronograma para asegurarnos que se entiende la funcionalidad descrita.

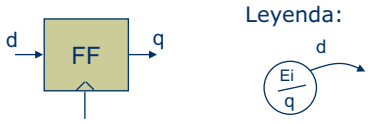
Pasar de una de estas formas de especificación funcional a expresar el funcionamiento mediante un grafo de estados no consiste en seguir un procedimiento sistemático. Es un tema que requiere una comprensión profunda del tema y cierta experiencia. Es más, incluso la solución no es única: dada una descripción funcional puede haber varios grafos correctos (aunque alguno será mas eficiente que los otros, por ejemplo, en cuanto al hardware necesario para su implementación; aunque no vamos a pretender optimizar nuestros diseños). Para aprender a hacer esto vamos a resolver algunos ejemplos.

Ejemplo 3 El biestable D

En la sección 6.1.5 hemos definido la funcionalidad del biestable D y hemos hecho algunos cronogramas detallados de su funcionamiento (suponiendo que no tiene retardo, primero y considerando el retardo después). Ahora, partiendo de la descripción funcional, vamos a sintetizar un circuito que tenga el comportamiento del biestable D. Este es un ejercicio muy sencillo, pues de hecho ya sabemos que al menos una implementación válida consiste en un único biestable.

En la figura 6.31 se ha dispuesto el símbolo del circuito (FF) mostrando el nombre de la entrada d y la salida q y la leyenda del grafo que vamos a crear con el nombre de las entradas y salidas. La codificación de las variables de estado no se hace ahora, se deja para el paso 2 del diseño (del grafo a las tablas de transiciones y de salidas) pues ahora, antes de crear el grafo ni siquiera sabemos cuántos estados tiene). Se ha dibujado un cronograma simplificado que visualiza con claridad el comportamiento del biestable como retardador de un ciclo de la secuencia de entrada: la salida en el ciclo siguiente (k+1) es igual a la entrada del ciclo actual (k): q(k+1) = d(k), denotando x(k) al valor de la variable lógica x durante el ciclo k.

Dado este funcionamiento, vemos que el biestable D es un circuito secuencial muy sencillo. El estado siguiente, que denominamos Q^+, es igual a la entrada d y la salida q es igual al estado Q. Así que, como hay dos valores posibles del bit de salida y la salida es igual al estado, el biestable tiene 2 estados: E0 cuya salida es, por ejemplo, 0 y E1 cuya salida es 1. Estando en cualquiera de los dos estados, si la



Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
Entrada d	1	1	0	1	0	0	0	1	0	1	0	0	0	1	0
Salida q	0	1	1	0	1	0	0	0	1	0	1	0	0	0	1

Fig. 6.31 Símbolo y cronograma simplificado de ejemplo de funcionamiento del circuito secuencial más sencillo, un biestable D.

entrada es 0 el estado siguiente tiene que ser el E0 y si la entrada es 1 se pasa al estado E1 al ciclo siguiente. La figura 6.32 muestra el grafo resultante.

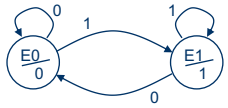
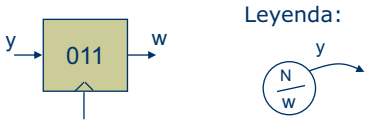


Fig. 6.32 Grafo de estados del biestable D.

Ejemplo 4 Reconocedor de la secuencia 011

Este ejemplo es más complejo que el anterior. Debemos dibujar el grafo del circuito secuencial reconocedor de la secuencia 011 que se muestra en la figura 6.33.



Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
Entrada y	1	1	0	1	1	1	1	0	1	1	0	1	1	0	0
Salida w	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0

Fig. 6.33 Símbolo y cronograma simplificado del reconocedor de la secuencia 011.

Como estamos sintetizando circuitos de Moore, la salida en el ciclo *k* no puede ser función de la entrada en ese ciclo, por lo que si en el ciclo *k*-2 la entrada vale cero, en el *k*-1 la entrada vale 1 y en el *k* también vale 1, (esto es, ha llegado la secuencia objetivo) la salida no puede valer 1 en el ciclo *k*. Lo que sí se calcula en el ciclo *k*, en función de la entrada y del estado, es el estado siguiente, y el estado

siguiente del ciclo *k* codifica la información de que "ha llegado la secuencia objetivo". Al pasar del ciclo *k* al *k*+1 el estado siguiente del ciclo *k* pasa a ser el estado actual del ciclo *k*+1, por lo que la salida, pasa a ser 1 en el ciclo *k*+1. Así que, cuando la salida vale 1 indica que ha llegado la secuencia objetivo (el último bit de la secuencia llegó en el ciclo anterior). En la figura 6.33 se muestra un cronograma para una secuencia de entradas concreta que clarifica el funcionamiento.

Un grafo con 8 estados

Hay muchas estrategias posibles para crear un grafo de estados que plasme la funcionalidad descrita. Podemos preguntarnos, ¿qué debe recordar el circuito para poder hacer lo que nos piden? Una posible respuesta es que si la memoria del circuito (el estado) guarda los tres últimos bits que llegaron por la entrada, el circuito podrá hacer lo que nos piden. Con esta idea el circuito puede estar en 8 estados diferentes.

Podemos denotar o denominar la información del estado actual como tdu, que son las primeras letras de tres, dos y uno. Con esta notación, el estado actual denotado como 110, por ejemplo, indica que hace tres ciclos llego un 1, hace dos un 1 y en el ciclo anterior llegó un 0. Así si el estado actual es el Q = tdu, y en el ciclo actual en la entrada hay un 0, el estado siguiente será el Q^+ = du0, mientras que si llega un 1 será el Q^+ = du1. Con estos nombres para los estados es sencillo dibujar el grafo con los 8 nodos y las dos flechas que salen de cada nodo. La salida del circuito valdrá 1 cuando el circuito esté en el estado Q = 011. En cualquier otro estado la salida valdrá 0.

Siempre tenemos que plantearnos cuál debe ser el estado inicial. De momento supongamos, ya que no nos han dicho nada en el enunciado, que el estado inicial es el 000, esto es suponer que en el ciclo cero es como si hubieran llegado ya tres ceros. Con esta suposición creamos el grafo que se muestra en la figura 6.34. ¡Pero este no es el único grafo posible!

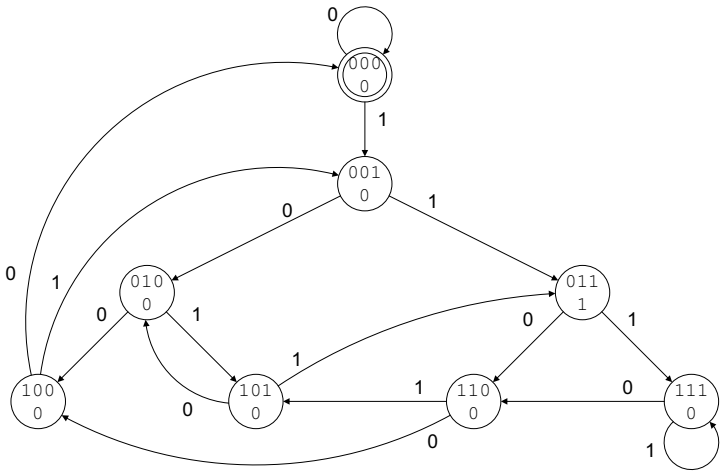


Fig. 6.34 Un posible grafo de estados del reconocedor de la secuencia 011 suponiendo que el estado inicial es el mismo que si hubieran llegado tres ceros seguidos.

Un grafo con 15 estados

Con la decisión anterior de que el estado inicial sea el 000, resulta que si en el primer ciclo de funcionamiento del circuito la entrada vale 1 y al ciclo siguiente vale otra vez 1, el circuito detecta la secuencia 011 y un ciclo más tarde saca un 1 por la salida; cuando realmente por la entrada no ha llegado la secuencia deseada.

Para corregir esto vamos a dibujar un grafo en el que el estado inicial tenga la información de que no ha llegado ningún bit. Pasado un ciclo el estado debe guardar la información de que ha llegado un 0 o un 1 según el caso y así hasta que pasados tres ciclos después del inicial el estado sea uno de los 8 que hemos dibujado en el grafo de la figura 6.34.

Para ello podemos añadir, a los 8 nodos de la figura 6.34, nuevos nodos para considerar los tres primeros ciclos de funcionamiento del circuito. Este subgrafo (la parte del grafo con los nuevos nodos) se ve en la figura 6.35. El estado inicial ahora se ha denotado con tres puntos ($\cdots$) porque no ha llegado nada en los tres ciclos anteriores, el estado al que se pasa después del primer ciclo se ha denotado con ($\cdot\cdot 0$) si llego un 0 estando en el estado inicial, o($\cdot\cdot 1$) si llegó un 1. En el tercer ciclo de funcionamiento el circuito está en uno de los 4 estados que hemos denotado ($\cdot 00$), ($\cdot 01$), ($\cdot 10$), o ($\cdot 11$). En el cuarto ciclo el circuito ya estará en uno de los 8 nodos del grafo de la figura 6.34, que en el subgrafo de la figura 6.35 hemos dibujado con líneas discontinuas, pero que son los mismos nodos que los de la figura 6.34, aunque dibujados en otra posición y sin los arcos que los unen. El grafo completo debería dibujarse fusionando los dos subgrafos y dejando como estado inicial el ($\cdots$).

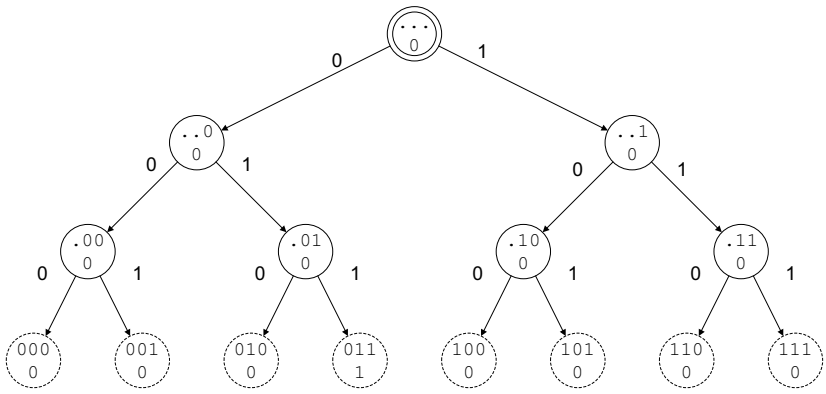


Fig. 6.35 Fragmento del grafo encargado de los tres primeros ciclos de funcionamiento del circuito secuencial reconocedor de la secuencia 011.

Un grafo con 8 estados equivalente al de 15

El diseño anterior tiene 15 nodos ¿podemos hacer otro grafo que manteniendo la misma funcionalidad tenga menos nodos? La respuesta es sí. Por ejemplo, si cambiamos el estado inicial del grafo de la figura 6.34 al estado (111) tenemos un grafo con 8 estados que tiene la misma funcionalidad que el de 15 nodos. No obstante, vamos a crear un grafo con todavía menos estados y la misma funcionalidad.

El grafo óptimo con 4 estados

El grafo con el número mínimo de estados debe memorizar la mínima información posible. No hace falta memorizar todas las posibles combinaciones de los tres bits de la entrada en los tres ciclos anteriores, sino sólo la información relevante.

Fusionando estados. Por ejemplo, en el grafo de 15 estados, ¿de qué nos sirve saber si estamos en el estado $(\cdots)$, en el $(\cdot\cdot1)$, o en el (111)? No nos sirve de nada, porque en todos esos estados no ha llegado nada relevante para detectar la secuencia que buscamos: 011. La secuencia objetivo comienza por 0 y como en todos los estados que acabamos de nombrar no llegó ningún cero en el ciclo anterior, ni un 0 hace dos ciclos y un 1 en el anterior, ni tampoco llego la secuencia en los últimos tres ciclos, los tres estados informan de lo mismo: no ha llegado nada relevante para detectar la secuencia. Por ello, estos tres estados se pueden fusionar en uno. Los estados que indican que ha llegado un cero en el último ciclo, sea lo que sea que llegó en los anteriores (los estados: $(\cdot\cdot0)$, $(\cdot00)$, $(\cdot10)$, (000), (010), (100) y (110)) se pueden juntar en un estado que indique: en el último ciclo llego un 0, por lo que hay indicios que está comenzando la secuencia objetivo.

Se puede seguir fusionando estados a partir del grafo de 15 para obtener uno con menos estrados, pero vamos a ir por otro camino, como si no hubiéramos creado los grafos anteriores. La idea es guardar sólo la información relevante.

Empezando de nuevo. Para crear el nuevo grafo, nos situamos en el estado inicial, que denominamos también $(\cdots)$ y que representa la información "no ha llegado nada interesante". Si durante este ciclo inicial llega un 1 por la entrada, nos quedamos en el mismo estado, $(\cdots)$, porque estamos esperando un 0 que es el principio de la secuencia objetivo. De momento el grafo que estamos creando se ve en la figura 6.36 a).

No obstante, si estando en el estado $(\cdots)$ llega un0 tenemos que pasar a un estado en el que recordemos que llegó un 0 ya que un cero es el primer bit de la secuencia esperada. Así que pasamos al estado que denotamos $(\cdot\cdot0)$, que indica "ha llegado un 0", es un estado en el que empieza a verse la posibilidad de que llegue la secuencia deseada. Si estando en $(\cdot\cdot0)$ llega un 1 tenemos que pasar a otro estado, $(\cdot01)$, que informa que ha llegado un 0 y luego un 1. En este estado estamos más cerca del objetivo. Si cuando estamos en $(\cdot01)$ llega un 1 tenemos que pasar un nuevo estado, (011), que nos indica que ha llegado la secuencia que buscamos, hemos alcanzado el objetivo. En este estado la salida debe ser 1, mientras que en los otros la salida debe ser 0, pues todavía no se ha dado la secuencia deseada cuando estamos en uno de ellos. La secuencia de estados y transiciones por la que hemos pasado en lo comentado en este párrafo se puede ver en la figura 6.36 b).

Hemos creado los nodos y las transiciones para el caso en que llegan seguidos los bits de la secuencia a reconocer. Ahora tenemos que situarnos en cada nodo y ver cuál es el estado siguiente si no llega el valor de la entrada que ya hemos puesto en el grafo sino que llega el otro bit. Por ejemplo, si estamos en el estado $(\cdot\cdot0)$, que indica que llegó un 0 en el ciclo anterior, y la entrada vale 0 en ese ciclo, el estado siguiente debe ser el mismo $(\cdot\cdot0)$, ya que seguimos teniendo lo mismo que teníamos de la secuencia objetivo, sólo tenemos el primer 0. La figura 6.36 c) muestra el grafo que estamos creando con esta nueva transición.

Hay que repasar que salgan dos arcos de cada nodo ya que la entrada del circuito es de un bit, un arco para el valor 0 de la entrada y otro para el 1. Por ejemplo, del nodo (·01) (llegaron los dos primeros bits de la secuencia) sale la flecha etiquetada con 1 hacia el nodo en el que se detecta la secuencia, pero falta la flecha que sale con un 0. ¿A dónde va? Después de este ciclo habrá llegado un 0, luego un 1 y luego un 0, para detectar la secuencia sólo nos vale el último 0, que es el bit con que comienza, por lo que de (·01) el arco etiquetado con 0 va a parar al nodo (··0), que indica que llegó un 0. Esta transición junto con todas las anteriores se muestra en la figura 6.36 d).

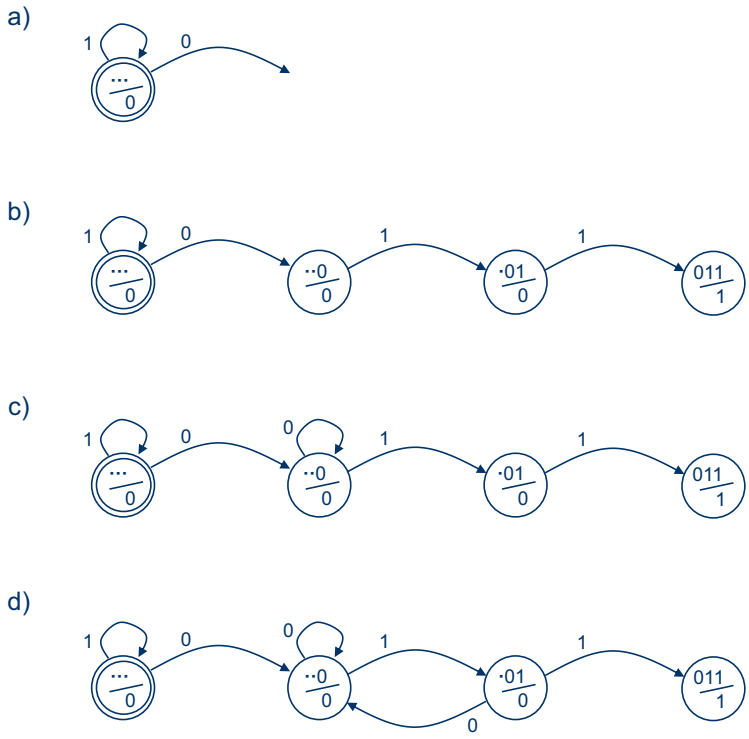


Fig. 6.36 Fases de creación del grafo de estados del reconocedor de la secuencia 011.

Por último sólo nos queda el nodo (011), que indica que hemos detectado la secuencia, del que no sale de momento ninguna flecha. Si estando en (011) llega un 0 pasaremos al (··0) ya que tenemos el primer bit de la secuencia objetivo, pero si llega un 1 no tenemos nada, por lo que pasamos al (···). El grafo completo se muestra en la figura 6.37.

Por último aconsejamos hacer algún recorrido por el grafo para una secuencia de valores de entrada concretos y comprobar con ello que la secuencia de salida es la deseada. Por ejemplo, en la figura 6.38 se muestra el cronograma con las salidas para la secuencia de entradas: 10010111 y sobre el grafo el recorrido que se hace (los nodos por los que se pasa) durante los ciclos del 0 a 7. Se observa que la secuencia de salidas que indica el grafo al seguir el recorrido marcado es la misma que indica el

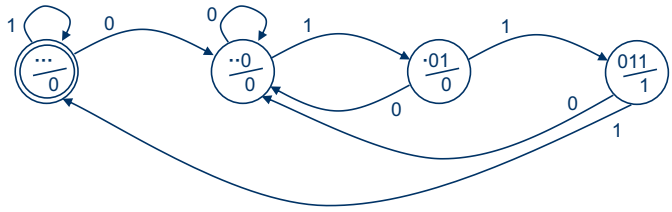


Fig. 6.37 Grafo de estados del reconocedor se la secuencia 011.

cronograma. Esto no es una prueba exhaustiva del correcto funcionamiento del grafo, pero puede ayudar a detectar errores en caso de que los haya.

Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**
Entrada y	1	0	0	1	0	1	1	1
Salida w	0	0	0	0	0	0	0	1

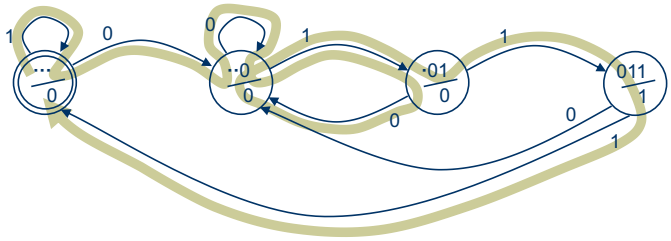
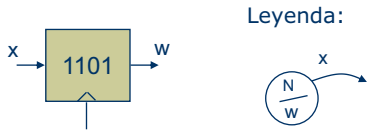


Fig. 6.38 Cronograma simplificado y recorrido por el grafo del reconocedor de la secuencia 011 siguiendo la secuencia de entradas del cronograma.

Es interesante ver que este grafo tiene la misma funcionalidad que el de 15 nodos de la figura 6.35, pero con sólo 4 nodos.

Ejemplo 5 Detector de la secuencia 1101 sin y con solapamiento

La figura 6.39 muestra el símbolo y la leyenda del grafo del circuito detector de la secuencia 1101 (N representa el nombre del nodo que no se codifica en binario todavía). En general, cuando una secuencia tiene los *k* bits iniciales iguales a los *k* bits finales, el enunciado nos tiene que indicar si se considera como secuencia válida una secuencia que usa los últimos bits de la secuencia anterior para formar sus primeros bits. Si esto está permitido se habla de detector de secuencias con solapamiento y si no lo está es el caso de un detector sin solapamiento. En el ejemplo anterior no nos planteamos esto ya que la secuencia empezaba con 0 y terminaba con 1. Pero esta secuencia empieza y termina por 1, con lo que hay un bit de posible solape. Un ejemplo de secuencia que puede solapar hasta 3 bits es: 1010011101. Vamos a crear dos grafos, uno para el caso que no está permitido el solapamiento y otro para el que sí lo está. En la figura 6.39 se muestran dos cronogramas simplificados para la misma secuencia de entrada, indicando la salida para el caso a) sin solapamiento y el b) con solapamiento.



a) **Sin solapamiento**

Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
Entrada y	0	1	0	1	1	0	1	1	0	1	1	1	0	1	1
Salida w	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

b) Con solapamiento

Núm. Ciclo	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
Entrada y	0	1	0	1	1	0	1	1	0	1	1	1	0	1	1
Salida w	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1

Fig. 6.39 Símbolo, leyenda del grafo y cronograma simplificado del reconocedor de la secuencia 1101, a) sin solapamiento y b) con solapamiento.

Comenzamos por dibujar el nodo del estado inicial, en el que no ha llegado nada relevante, nodo al que denominamos (····), siguiendo con la notación usadaen el anterior reconocedor de secuencia, ya que la secuencia a reconocer tiene 4 bits. Como la secuencia objetivo comienza por 1, si estando en este estado llega un 0 pasaremos al estado siguiente a este mismo estado, ya que sigue sin haber llegado nada relevante. Supongamos ahora que llega la secuencia objetivo. El grafo de los nodos hasta reconocer la secuencia se muestra en la figura 6.40 a). En todos los nodos la salida es 0 excepto en el (1101) en el que se reconoce que llegó la secuencia deseada.

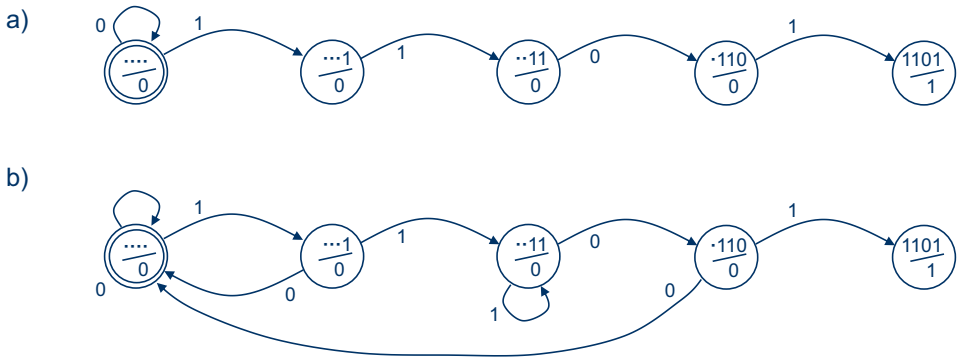


Fig. 6.40 Fases de creación del grafo de estados del reconocedor de la secuencia 1101.

Ya tenemos todos los estados en los que hay algo de información válida y no redundante. Veamos ahora cuál ha de ser el estado siguiente si estando en cada uno de estos nodos llega el valor no deseado, el que

frustra el avance hacia el nodo (1101). Si estamos en (···1) y llega un 0 tendríamos que ir a un estado que podríamos llamar (··10), pero en este estado no llegó nada relevante para la detección de la secuencia, así que no ponemos un nuevo nodo sino que vamos al nodo (····) que tiene esa información: no ha llegado nada relevante. Si estando en (··11) llega un 1, cuando lo deseado sería un 0, se nos frustra el avance. Tendríamos que pasar al que podríamos llamar estado (·111), pero lo relevante de este posible estado es lo mismo que nos informa el estado (··11) del que partimos. Por ello, si estamos en (··11) y llega un 1 volvemos al (··11) y no creamosningún nuevo nodo. En la parte b) de la figura se muestran estas transiciones añadidas al grafo de la parte a), junto con el arco de cuando llega 0 estando en (·110).

La figura 6.41 a) muestra el grafo completo, después de añadir los dos arcos que salen del estado en el que se detecta que ha llegado la secuencia. Si estando en el estado (1101) llega un 0, los cuatro últimos bits que han llegado son 1010, por lo que pasamos al estado que indica que no ha llegado nada relevante (····). Pero si estando en el (1101) llega un 1, los cuatro últimos bits son 1011, lo que nos dice que el último 1 que llegó puede ser el primer 1 de una nueva secuencia, por lo que se pasará al estado (···1). Esta última transición cambia si el reconocedor admite secuencias solapadas. En este caso, de los cuatro últimos bits 1011 se pueden aprovechar los dos últimos para considerarlos como los dos primeros de la nueva secuencia. Esto conlleva que estamos considerando el último 1 de la secuencia anterior como el primero de la nueva. La parte b) de la figura muestra el grafo con solapamiento.

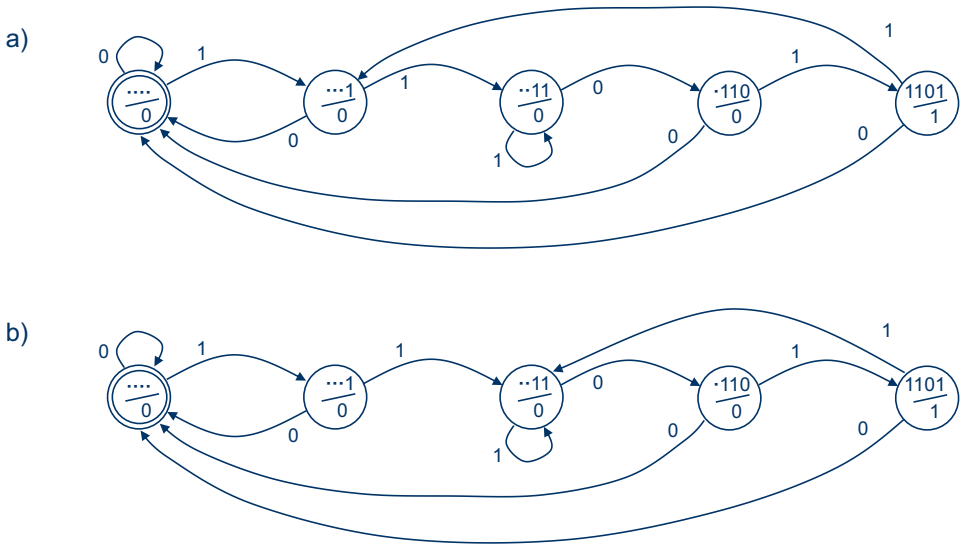


Fig. 6.41 Grafo de estados completo del reconocedor de la secuencia 1101, a) sin solapamiento y b) con solapamiento.

Ejemplo 6 El piloto automático

Vamos a dibujar el grafo de estados de un circuito secuencial encargado del piloto automático de aterrizaje de un avión, que en la introducción (sección 6.1.1) diseñamos con un circuito combinacional, pero intuimos que con un secuencial el circuito podría saber, cuando no recibe señal de ninguna

frecuencia, si está en la parte sin cobertura de la derecha o en la de la izquierda y podría dirigir el avión hacia el centro sin necesitar la ayuda del piloto. El circuito secuencial es el encargado de generar las señales w_1 y w_0 que controlan el sistema electromecánico de dirección de un avión en función de las señales de entrada s_i y s_d que indica que se recibe la frecuencia de la izquierda y/o derecha respectivamente.

Como hemos hecho en el ejemplo anterior, pensemos primero qué estados debe tener el circuito para su correcto funcionamiento. En función del estado se generan las salidas en cada ciclo. Es inevitable tener al menos un estado diferente para cada una de las posibles salidas del circuito (ya que es un circuito de Moore). Las salidas posibles son w_1w_0 igual a 00, para ir recto, 01 para ir hacia la derecha y 10 para ir hacia la izquierda. Al menos tenemos que tener estos tres estados. El estado en cada ciclo depende de las entradas del ciclo anterior y del estado del ciclo anterior. Por ello, el estado en el ciclo actual nos indica la posición del avión en el ciclo anterior. Veamos una secuencia de posibles valores de las entradas s_is_d. Supongamos que el avión viaja por la línea central, recibiendo las dos frecuencias. Denominemos C (de central) al estado que indica que en el ciclo anterior el avión estaba en el centro. En este estado (independientemente del valor de las entradas, ya que es un circuito de Moore) lo más sensato es dirigir el avión recto, por lo que en el estado C las salidas son w_1w_0 =00. El estado C además es el estado inicial, ya que el enunciado de la introducción indica que cuando el avión viaja en la zona central de aterrizaje es cuando el piloto transfiere el control al piloto automático y suponemos que en ese momento se inicializa el circuito secuencial que estamos diseñando.

Si en el ciclo en que estamos en C las entradas valen s_is_d = 11, nos informan de que estamos en la zona central y por lo tanto el estado siguiente debe ser el mismo, C. Pero si las entradas son s_is_d = 01 quiere decir que estábamos en la zona central en el ciclo anterior pero ahora estamos en la derecha de la pista, recibiendo sólo la frecuencia de la derecha. El estado siguiente debe ser distinto, para indicar la nueva situación, por lo que pasaremos a un estado que podemos denominar CD (centro-derecha). En el estado CD se debe dirigir el avión hacia el centro y como estamos un poco a la derecha del centro, hay que dirigir el avión a la izquierda con w_1w_0 =10. Si estando en CD llega s_is_d = 01 el estado siguiente debe ser el mismo CD y si llega 11 volvemos al estado C. De la misma forma procedemos si el avión se va hacia la izquierda, definiendo el nuevo estado CI (centro-izquierda). El grafo con el nodo C, el CD y el CI y las transiciones comentadas se muestra en la figura 6.42 a).

Si estando en el estado CD las entradas valen s_is_d = 00 quiere decir que en el ciclo anterior estábamos en la zona de cobertura de la derecha y hemos pasado a la zona en la que no se recibe ninguna señal. En este caso, con un circuito combinacional estábamos perdidos pues no sabíamos si estábamos en la zona exterior de la derecha o de la izquierda ya que no había forma de saber donde estábamos momentos antes. Ahora esta información está en el estado del circuito y podemos asegurar que, en este caso, estamos en la zona externa de la derecha. Esto es así porque suponemos que la frecuencia del reloj del circuito es mucho mayor que la velocidad del avión en atravesar cualquiera de las zonas en las que se recibe de una o las dos frecuencias. Así, estando en CD para pasar a la zona exterior de la izquierda primero tendríamos que haber estado varios ciclos en C y varios en CI. Podemos, de momento, dedicar un nuevo nodo, D, para la zona exterior derecha. En este nodo lo razonable es dirigir el avión hacia la izquierda, por lo que w_1w_0 =10. Estando en esta zona si llega s_is_d = 00 seguimos en la misma zona, si llega 01 pasamos a CD y no puede llegar ninguna otra señal de entrada por la suposición que hemos

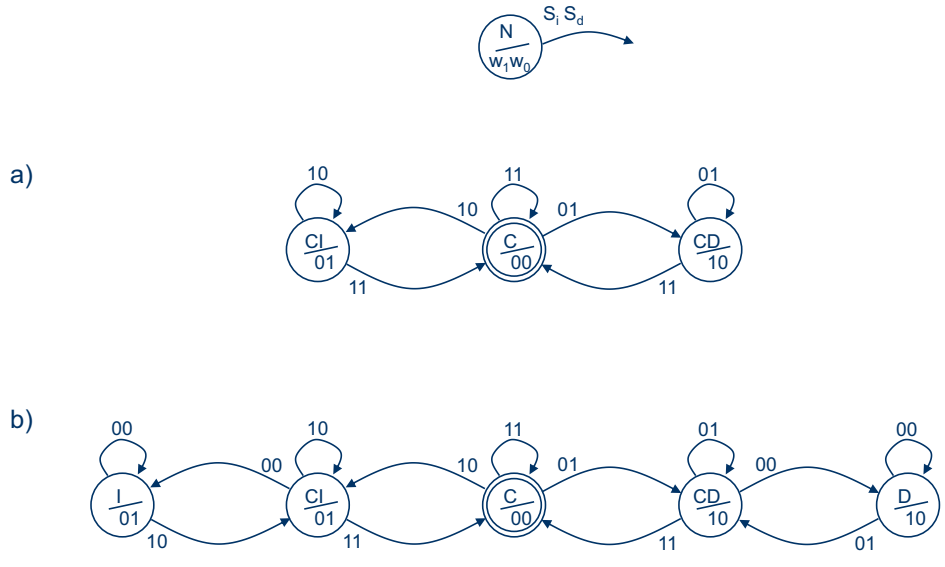


Fig. 6.42 Fases de creación del grafo de estados del piloto automático.

hecho de la frecuencia de reloj elevada. Lo equivalente ocurre con la zona exterior de la izquierda, I. El grafo parcialmente construido con estos dos nuevos nodos se muestra en la figura 6.42 b).

Por último dibujamos los arcos que no se pueden dar por la "física" del problema y la suposición de que la frecuencia es tan grande que mientras el avión atraviesa una de las zonas pasan varios ciclos. Estos arcos que no se pueden dar se han dibujado solamente indicando la salida del nodo (para que de cada nodo salgan 4 arcos ya que hay 2 entradas) pero no su destino, ya que no importa a dónde vayan a parar. Otra posibilidad que usamos en algunas ocasiones es no dibujar estas flechas ya que no añaden más información al grafo, sabemos que si no están es porque no se pueden dar.

Por otro lado, cuando dos arcos tienen el mismo origen y el mismo destino sólo se ha pintado una flecha con las dos etiquetas. El resultado final se ve en la figura 6.43.

Aunque este grafo es correcto, puede observarse que como las acciones, las salidas, en el nodo CD y D son las mismas, ir hacia la derecha, y no hay más razón para diferenciar entre esos dos estados, los dos se pueden fundir en uno solo (que denominamos D, por ejemplo), dibujando adecuadamente las transiciones. Lo mismo ocurre con los nodos I y CI. El grafo resultante con sólo 3 nodos se muestra en la figura 6.44.

6.5.2 Del grafo de estados a las tablas transiciones y salidas

Hemos visto al explicar el grafo de estados en la sección 6.2 cómo pasar de la tabla de transiciones al grafo de estados y luego en la sección 6.2.2 cómo incorporar en el grafo de estados la tabla de las

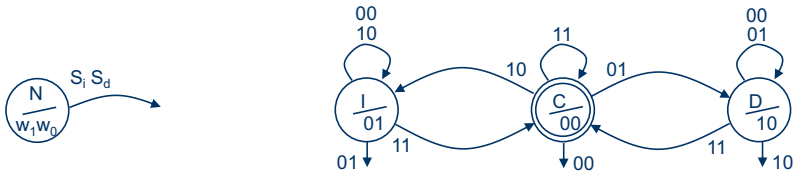


Fig. 6.44 Grafo de estados del piloto automático con 3 estados.

salidas. Con esto comprendimos qué significa cada elemento del grafo (nodos, flechas y las etiquetas en nodos y flechas) y su relación con la tabla de transiciones y con la de salidas de un circuito secuencial. Entendido esto, pasar del grafo a las tablas es trivial. Veamos dos ejemplos

Biestable D

El biestable D es el caso más simple de circuito secuencial. El símbolo del circuito, con el nombre de la entrada y salida y la leyenda del grafo se mostró en la figura 6.31 y el grafo en la figura 6.32. Lo primero que debemos hacer para obtener la tabla de transiciones y de salidas es codificar con el número mínimo de bits los estados del grafo. Si el grafo tiene m estados, hacen falta logaritmo en base 2 de m (parte entera por arriba) bits para codificar los m estados. En nuestro caso, como hay dos estados, sólo hace falta un bit para codificarlos. Ahora hay que asignar un código para cada estado. Cualquier asignación es correcta, pero según sea esta asignación cambiará el circuito resultante, que será más o menos complejo. Hay algoritmos para elegir una buena asignación, pero nosotros para simplificar no los usaremos. Haremos una asignación cualquiera, excepto en casos triviales como el que nos ocupa. Si asignamos el código 0 al estado E0 y el 1 al E1 la tabla de transiciones (que en general muestra el estado siguiente, Q+, en función del estado actual y de la entrada) es:

Q	**d**	**Q^+**
0	0	0
0	1	1

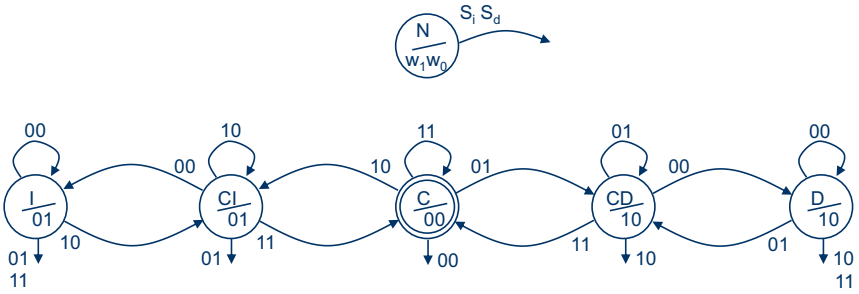


Fig. 6.43 Grafo de estados del piloto automático con 5 estados.

Q	d	Q^+
1	0	0
1	1	1

Es fácil ver que el estado siguiente, en este circuito tan simple, no es función del estado actual, sino que es función solamente de la entrada, es más, es directamente igual a la entrada. Por ello, en el caso de un circuito formado solamente por un biestable D, el estado siguiente es directamente la entrada, quedando la tabla de transiciones así:

d	Q^+
0	0
1	1

La salida es función del estado. En este caso la función es la más sencilla posible, la salida es igual al estado:

Q	q
0	0
1	1

Un ejemplo más complejo

La figura 6.45 muestra un grafo con su leyenda, del que vamos a obtener las tablas de transiciones y de salidas. Como tiene 3 estados hacen falta 2 bits para codificarlos. En la figura 6.46 se muestra en una tabla el código asignado a cada estado (elegido sin ningún criterio) y el grafo con los nodos codificados.

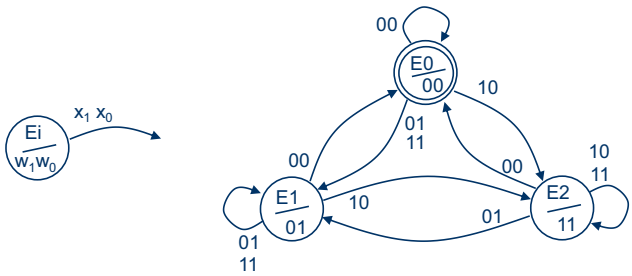


Fig. 6.45 Grafo de estados.

La figura 6.47 muestra las tablas de transiciones y de salidas que resultan de extraer la información del grafo.

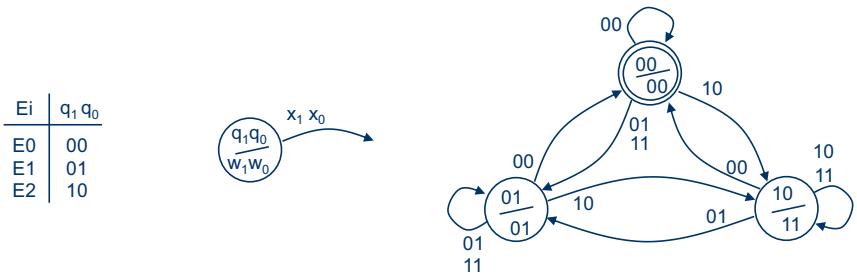


Fig. 6.46 Tabla de codificación y grafo con los estados codificados.

Tabla de transición

q_1	q_0	x_1	x_0	q_1^+	q_0^+
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	x	x
1	1	0	1	x	x
1	1	1	0	x	x
1	1	1	1	x	x

Tabla de salida

q_1	q_0	w_1	w_0
0	0	0	0
0	1	0	1
1	0	1	1
1	1	x	x

Fig. 6.47 Tabla de transiciones y de salida del grafo de la figura 6.46.

6.5.3 De las tablas al esquema lógico

Este tercer y último paso en la síntesis de circuitos secuenciales consiste en disponer los biestables que requiere el circuito y dibujar el esquema lógico de los dos circuitos combinacionales que calculan el estado siguiente y las salidas usando para cada uno de ellos alguna de las técnicas que vimos en el capítulo 2:

- En suma de minterms.
- Con un decodificador y puertas Or.
- Con una ROM.

La estructura del circuito secuencial de Moore es la de la figura 6.17. La funcionalidad del circuito combinacional encargado de generar el estado siguiente, Q+, a partir del estado actual, Q, y de las entradas, la especifica la tabla de transiciones y la funcionalidad del circuito que genera las salidas la especifica la tabla de salida. Ambas tablas las acabamos de crear en el paso anterior, así que este tercer

paso consiste en sintetizar dos circuitos combinacionales y esto ya lo vimos en el capítulo 2, por lo que no requiere nuevas explicaciones. Veamos el resultado de los dos ejemplos de la sección anterior.

Biestable D

De la tabla de transición que hemos obtenido se ve que el estado siguiente es igual a la entrada del circuito, que hemos denominado d. En general (ver figura 6.17) el estado siguiente se conecta con la entrada D de los biestables que guardan el estado del circuito. En nuestro caso la entrada d del circuito se conecta directamente con la entrada D del biestable. Según la tabla de salidas que hemos obtenido en el apartado anterior, la salida q del circuito es igual al estado actual, que es la salida Q del biestable de estado. El circuito del biestable es el que se muestra en la figura 6.48 junto con el símbolo del circuito, con el nombre de la entrada y salida, y el grafo de estados.

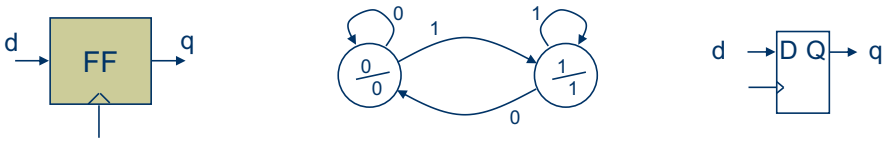


Fig. 6.48 Símbolo del circuito secuencial más sencillo, un biestable D, grafo de estados y esquema lógico de su implementación.

El resultado no podía ser otro ya que cuando hemos definido su funcionalidad para sintetizarlo hemos usado la del biestable D. Aunque ahora es un buen momento para comentar que si en el paso anterior hubiéramos codificado los dos estados del grafo de la figura 6.32 con un 1 para el estado E0 y un 0 para el E1, en este último paso el circuito resultante hubiera tenido una puerta Not entre la entrada d del circuito y la D del biestable y otra Not entre la salida Q del biestable y la salida q del circuito.

Un ejemplo más complejo

Vamos a dibujar un esquema lógico que implemente el grafo de la figura 6.45 a partir de las tablas de transiciones y de salidas obtenidas en la sección anterior en la figura 6.47. Vamos a sintetizar con una memoria ROM el circuito combinacional del estado siguiente y con un decodificador y puertas Or el de las salidas, que junto con los dos biestables de estado forman el circuito secuencial que nos ocupa. La figura 6.49 muestra el resultado.

En general comenzamos disponiendo los *k* (en este ejemplo *k*=2) biestables D que forman el estado del computador. Es muy importante etiquetar cada una de las salidas Q de los biestables con su nombre, que habremos usado al crear las tablas en el paso anterior y que normalmente denominamos q_{k-1}, ..., q_1, q_0 y cada una de sus entradas D que codifican los bits del estado siguiente q_{k-1}^+, ..., q_1^+, q_0^+. Supongamos que el circuito secuencial tiene *n* entradas y *m* salidas (en este ejemplo *n*=*m*=2) denominadas x_{n-1}, ..., x_1, x_0 y w_{m-1}, ..., w_1, w_0. Con los biestables dispuestos y las entradas y salidas del circuito dibujadas, ya sólo tenemos que dibujar:

- el circuito del estado siguiente usando la tabla de transiciones que define q_{k-1}^+, ..., q_1^+, q_0^+ en función del estado q_{k-1}, ..., q_1, q_0 y las entradas x_{n-1}, ..., x_1, x_0 y

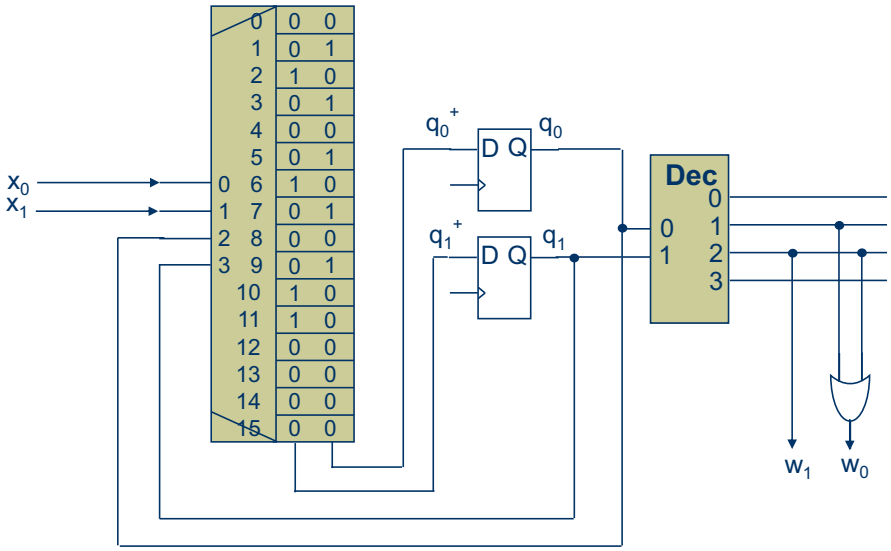


Fig. 6.49 Esquema lógico resultante de sintetizar el grafo de la figura 6.45 donde se han implementado las tablas de transiciones y de salidas de la figura 6.47 con una ROM la primera y con un decodificador y puertas Or la segunda.

- el circuito de las salidas usando la tabla de salidas que expresa w_{m-1}, ..., w_1, w_0 en función del estado q_{k-1}, ..., q_1, q_0.

Con una ROM y un multiplexor de buses. Es posible sintetizar a la vez el circuito del estado siguiente junto con el de las salidas usando una ROM y un multiplexor de buses. El ejemplo anterior se muestra en la figura 6.50. En general, si tenemos un circuito secuencial con *n* bits de entrada, *m* de salida y *k* de estado, la ROM tiene 2^k palabras de kx$2^n + m$ bits por palabra y el multiplexor de buses tiene *n* bits de selección para elegir entre 2^n buses de *k* bits cada bus. Los bits de dirección de la ROM son los bits de estado del circuito y los bits de selección del multiplexor son los bits de entrada del circuito. Hay una palabra de la ROM para cada estado del circuito y cada una tiene la salida del circuito para ese estado y además el estado siguiente para cada uno de los posibles arcos que salen de ese estado (para cada combinación de valores de las entradas). Es un diseño sencillo, pero es fácil olvidarse alguna de las etiquetas de la ROM o del multiplexor o cometer un error al especificar la tabla que representa el contenido de la ROM, lo que haría que el circuito no quedara correctamente especificado.

6.6 Análisis temporal

El análisis temporal de un circuito secuencial consiste en obtener el tiempo de ciclo mínimo para que el circuito realice correctamente la funcionalidad descrita por el grafo que se ha obtenido en el análisis lógico (que vimos en la sección 6.4). Vamos a hacer el análisis temporal del mismo circuito que usamos para ejemplificar el análisis lógico En la figura 6.51 reproducimos el esquema lógico del circuito al que le hemos añadido junto a cada dispositivo su tiempo de propagación.

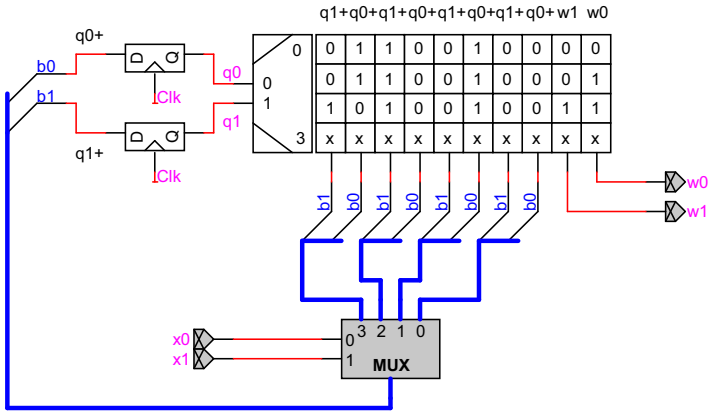


Fig. 6.50 Esquema lógico resultante de sintetizar el grafo de la figura 6.45 donde se han implementado las tablas de transiciones y de salidas de la figura 6.47 con una ROM y un multiplexor de buses.

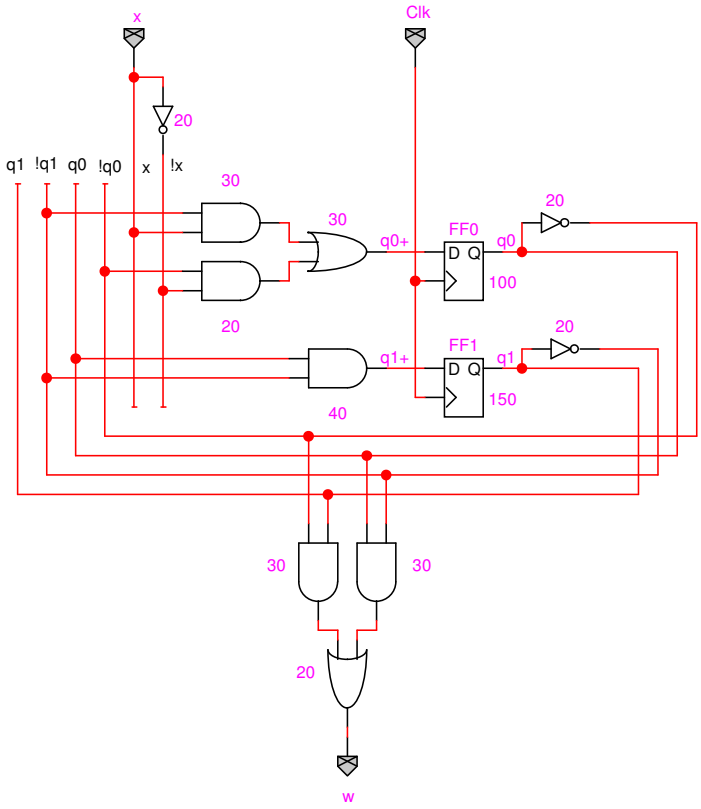


Fig. 6.51 Esquema lógico del circuito secuencial a analizar. Al lado de los dispositivos se indica su tiempo de propagación.

Este análisis temporal se hace a partir del esquema lógico del circuito, del tiempo de propagación de todos los dispositivos combinacionales, del tiempo de propagación de los biestables y por último de las especificaciones temporales de las entradas y salidas del circuito: tiempo desde que se produce el flanco ascendente de reloj que marca el inicio del ciclo hasta que las señales a la entrada del circuito están estables (110 u.t. en el ejemplo) y tiempo que se requiere que las señales de salida estén estables a su valor correcto antes de que llegue el flanco ascendente de reloj que marca el final del ciclo (20 u.t. en el ejemplo).

Camino crítico de un circuito secuencial y tiempo de ciclo

En un circuito secuencial el tiempo de ciclo (periodo) de la señal de reloj tiene que ser suficientemente grande como para que todas las señales que llegan a las entradas D de los biestables estén estables en su valor correcto cuando llegue el flanco ascendente de reloj que indica el final del ciclo y el inicio del siguiente. Si esto no ocurre, los biestables almacenarán valores incorrectos y el sistema no funcionará como debe. Para ello, el periodo de la señal de reloj, el tiempo entre dos flancos ascendentes de reloj, debe ser mayor que el tiempo del camino crítico del circuito. En un principio definimos **camino crítico** de un circuito secuencial al camino con mayor tiempo de propagación de entre todos los posibles caminos que van desde la salida Q de un biestable hasta la entrada D de otro biestable, atravesando circuitos combinacionales (no atravesando ningún biestable). En la figura 6.52 se marca un camino que va de un biestable de estado a otro biestable (camino b-b) sobre la estructura a bloques de un circuito secuencial de Moore. No obstante, al ser un esquema a bloques no se detalla el biestable donde comienza el camino, ni las puertas por las que pasa dentro del circuito combinacional que calcula el estado siguiente, ni se detalla el biestable donde termina el camino.

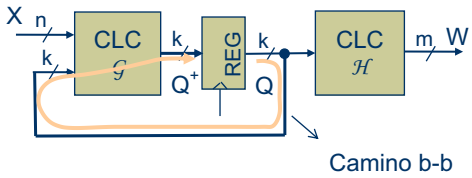


Fig. 6.52 Esquema lógico a bloques de un circuito secuencial de Moore sobre el que se ha marcado un camino que va de la salida de un biestable de estado a la entrada de otro (Camino b-b).

El camino crítico del circuito no tiene porqué ser un camino como el marcado en la figura 6.52. Hay otros tipos de camino. Un sistema síncrono, como es la CPU (central processing unit) de un computador, está formado por varios circuitos secuenciales conectados entre sí, operando todos con la misma señal de reloj. En este entorno, y siguiendo el esquema a bloques de la figura 6.53, cada entrada del circuito secuencial que estamos analizando (que llamamos, por ejemplo, circuito B) está conectada, se alimenta, de la salida de otro circuito secuencial (al que llamamos A). A su vez, cada salida del circuito B, esta conectada a otro circuito al que alimenta (que llamamos C).

El valor de la señal a la entrada del circuito B es la señal de salida del circuito A y por ello, en general, desde que se produce el flanco ascendente de reloj ha de pasar un cierto tiempo hasta que la entrada del circuito B, o la salida del A, se estabilice. Este es el tiempo de propagación de los biestables de estado del circuito A más el tiempo de propagación del circuito de salida de A. Por ello, el camino crítico del

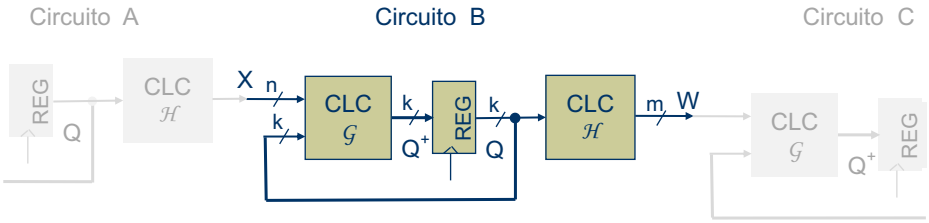


Fig. 6.53 Esquema lógico a bloques de un circuito secuencial de Moore, circuito B, que es alimentado por el circuito A y que alimenta al circuito C. El circuito B es el que estamos analizando y el A y C se muestran parcialmente.

sistema puede ser uno que parte de un biestable de estado del circuito A y termina en un biestable del circuito B (Un camino de este tipo se encuentra marcado y etiquetado con el nombre “Camino e-b” en la figura 6.54). Ocurre que normalmente no nos dan el tiempo de propagación de los biestables ni del circuito de salida del secuencial A, sino que nos dan la suma de ambos: para cada entrada del circuito nos dan el tiempo desde que se produce el flanco ascendente de reloj hasta que la entrada está estable a su valor correcto.

Por otro lado, el camino crítico del circuito podría ir desde la salida de uno de los biestables de estado del circuito B a la entrada de uno de los del circuito C. (Un camino de este tipo se encuentra marcado y etiquetado con el nombre “Camino b-s” en la figura 6.54). En este caso, desde que la salida del circuito B se estabiliza la señal debe atravesar todavía el combinacional del estado siguiente del circuito C para llegar al biestable de estado de este circuito antes de que se produzca el flanco de final de ciclo de la señal de reloj. Al especificar el circuito B, normalmente, nos indicarán el tiempo en que cada salida debe estar estable antes de que llegue el flanco ascendente de reloj (que es el tiempo de propagación del combinacional del estado siguiente del circuito C), para que podamos calcular el tiempo del camino que va de la salida de un biestable del circuito a analizar a una salida del circuito (y que realmente llega a la entrada de un biestable del circuito alimentado por el que estamos analizando.

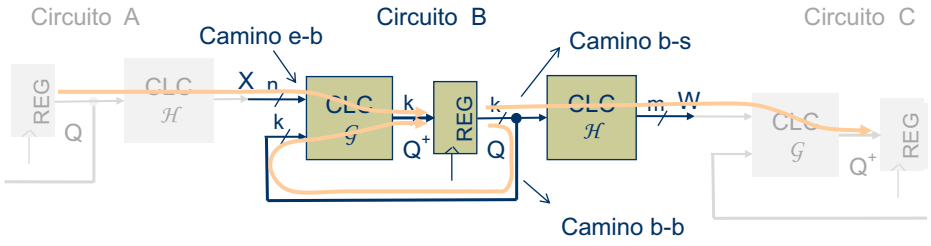


Fig. 6.54 Esquema lógico a bloques de un circuito secuencial de Moore, circuito B, que es alimentado por el circuito A y que alimenta al circuito C. El circuito B es el que estamos analizando y el A y C se muestran parcialmente. Se muestran los tres tipos de camino en este circuito: camino de biestable a biestable (b-b), camino de entrada a biestable (e-b) y camino de biestable a salida (b-s).

Incorporando las entradas y salidas en la definición de camino crítico esta queda así. El **camino crítico** de un circuito secuencial es el camino con mayor tiempo de propagación de entre todos los posibles caminos que van (detallamos el caso de Moore):

- **de biestable a biestable (camino del tipo b-b)**, desde la salida Q de un biestable del circuito hasta la entrada D de otro, o del mismo, biestable del circuito, atravesando dispositivos combinacionales (no atravesando ningún biestable),
- **de entrada a biestable (camino del tipo e-b)**, desde una de las entradas del circuito a la entrada D de un biestable del circuito, atravesando dispositivos combinacionales (no atravesando ningún biestable),
- **de biestable a salida (camino tipo b-s)**, desde la salida Q de un biestable del circuito hasta una salida del circuito, atravesando dispositivos combinacionales (no atravesando ningún biestable).

Para calcular el tiempo de propagación de un camino hay que tener en cuenta lo siguiente.

- **De biestable a biestable**. En el tiempo del camino que sale de un biestable y llega a otro biestable hay que contabilizar el tiempo de propagación del biestable donde empieza el camino y acumular sobre este valor la suma de los tiempos de propagación de entrada-salida de cada uno de los dispositivos combinacionales por donde pasa el camino hasta llegar a la entrada de uno de los biestables de estado del circuito.
- **De entrada a biestable**. En el tiempo de un camino que comienza en una entrada del circuito y llega a uno de sus biestables de estado hay que contabilizar el tiempo que tarda la entrada en estabilizarse desde que se produce el flanco ascendente de reloj (este dato nos lo darán junto con el esquema del circuito) y acumular sobre este valor la suma de los tiempos de propagación de entrada-salida de cada uno de los dispositivos combinacionales por los que pasa el camino hasta llegar a la entrada de uno de los biestables de estado del circuito.
- **De biestable a salida**. En el tiempo de un camino que sale de un biestable y termina en una salida del circuito, después de acumular sobre el tiempo de propagación del biestable donde empieza el camino la suma de los tiempos de propagación de entrada-salida de cada uno de los dispositivos combinacionales por los que pasa el camino hasta llegar a la salida hay que sumar el tiempo que tiene que estar estable la salida a su valor correcto antes de que llegue el flanco ascendente de reloj.

Encontrando el camino crítico del ejemplo

En el ejemplo que nos ocupa, vamos a encontrar el camino más largo de cada uno de los tipos de caminos. Los tiempos de propagación de los dispositivos que forman el circuito se indican junto al dispositivo en el esquema lógico de la figura 6.51. Para que dé más juego el ejemplo hemos puesto distintos tiempos de propagación a dispositivos que normalmente tienen el mismo tiempo. Por ejemplo hay una puerta And-2 de 20 u.t. cuando el resto son de 30 u.t., hay dos biestables uno de 100 u.t. y otro de 150. Las especificaciones temporales de las entradas y salidas, que no se indican el esquema, son las siguientes. El tiempo desde que se produce el flanco ascendente de reloj hasta que las señales a la entrada del circuito están estables es de 110 u.t. y tiempo que se requiere que las señales de salida estén estables a su valor correcto antes de que llegue el flanco ascendente de reloj es de 20 u.t. Con estos datos ya podemos encontrar el camino crítico.

Comenzamos por el camino más largo de tipo e-b, de la entrada del circuito a un biestable de estado. Este camino se muestra en la figura 6.55. El tiempo de este camino es de 180 u.t. y se obtiene de sumar los siguientes tiempos:

- 110 u.t. del tiempo en que tarda en estabilizarse la entrada (esto nos lo dan en el enunciado junto al esquema lógico),
- 20 u.t. de la puerta Not,
- 20 u.t. de la And-2 y
- 30 de la Or-2.

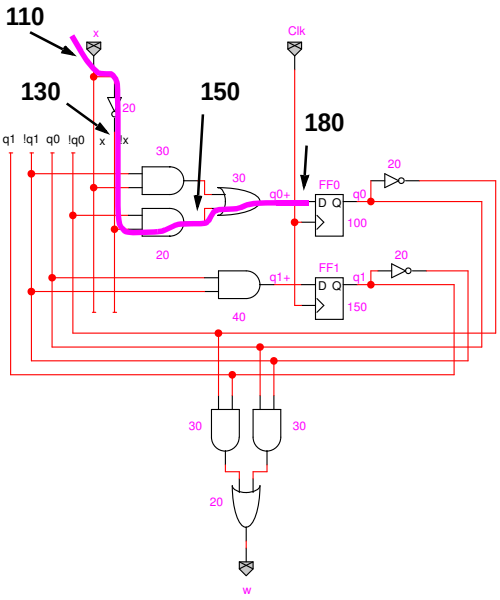


Fig. 6.55 Esquema lógico del circuito secuencial a analizar sobre el que se ha dibujado el camino más largo de tipo e-b.

El camino más largo en tiempo del tipo b-b se ha marcado sobre el esquema de la figura 6.56. El tiempo de este camino es de 230 u.t. como se detalla en la figura. Por último, el camino más largo del tipo b-s es de 240 u.t. y se muestra en la figura 6.57. Este es el camino crítico del circuito, porque es el de mayor tiempo de todos los caminos.

Tiempo de ciclo. Por lo que respecta al circuito que hemos analizado, el tiempo de ciclo tiene que ser superior a 240 u.t. para que el funcionamiento sea correcto. Claro está que si este circuito está inmerso en otro más grande, el tiempo de ciclo del circuito grande puede que tenga que ser mayor que el calculado, si el camino crítico del circuito grande no pasa por el circuito que hemos analizado.

Por último comentar que, en algunos ejercicios, cuando se nos pide calcular el tiempo de ciclo mínimo para que un circuito secuencial determinado funcione correctamente, se nos dice que cada entrada al circuito llega directamente de un biestable de un hipotético sistema secuencial que alimenta al nuestro,

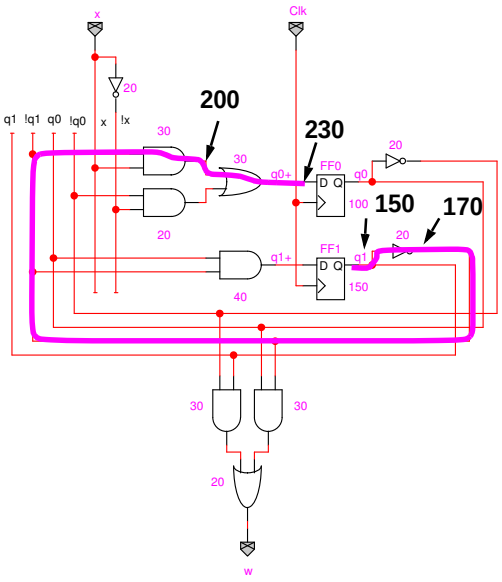


Fig. 6.56 Esquema lógico del circuito secuencial a analizar sobre el que se ha dibujado el camino más largo de tipo b-b.

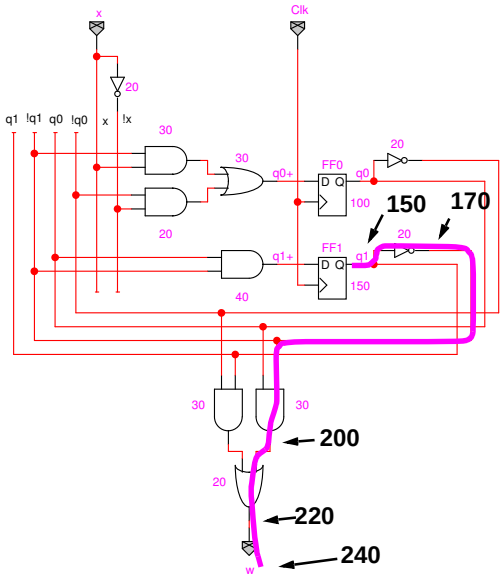


Fig. 6.57 Esquema lógico del circuito secuencial a analizar sobre el que se ha dibujado el camino más largo de tipo b-s.

sin atravesar ningún circuito combinacional. En estos casos, y si no se indica lo contrario, suponemos que este biestable que alimenta nuestra entrada tiene un tiempo de propagación igual a los biestables de nuestro circuito. Por ello, en el tiempo del camino desde una entrada del circuito hasta la entrada D de un biestable, debe sumarse el tiempo de propagación del hipotético biestable que alimenta la entrada de nuestro circuito. Además, en algunos ejercicios también se indica que cada salida de nuestro circuito se conecta directamente (sin pasar por ningún circuito combinacional) a la entrada D de un biestable. Por ello, en el cálculo del tiempo de un camino que va de la salida Q de un biestable a una salida de nuestro circuito, no hace falta sumar el tiempo de ningún circuito combinacional externo.