Departamento de Arquitectura Instituto de Computación

Universidad de la República

Montevideo - Uruguay

Notas de Teórico

Circuitos Secuenciales

Arquitectura de Computadoras

(Versión 4.3 - 2012)

7 CIRCUITOS SECUENCIALES

7.1 Definición

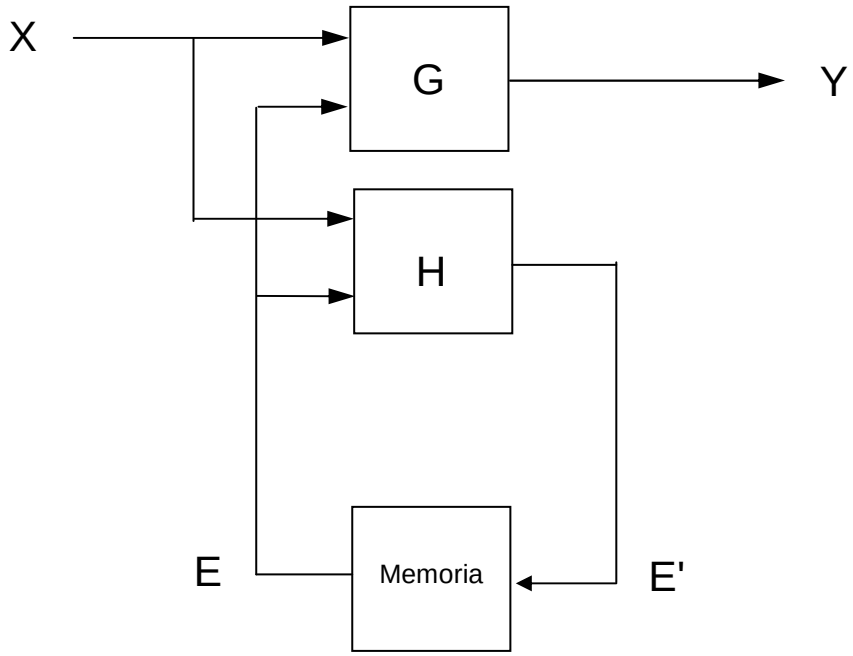
Un circuito secuencial es un sistema en el cual la salida depende de la entrada y del valor de las entradas anteriores. Un ejemplo ya visto de estos circuitos son los flip-flops.

Los sistemas secuenciales se caracterizan por tener “memoria”, la que se puede asociar a la existencia de “estados” internos. Estos estados son los que determinan, junto a la entrada actual, el comportamiento futuro del sistema. Como vimos los estados de alguna forman “memorizan” la evolución en el tiempo que tuvieron las entradas y por ende el sistema en su conjunto.

Un sistema secuencial se describe por las ecuaciones:

$$Y = G(X, E)$$
$$E' = H(X, E)$$

siendo Y la salida, X la entrada, E el estado interno y E’ el nuevo valor del estado luego de la transición provocada por la presencia de una nueva entrada.



Esto significa que la salida no depende solo de las entradas (como en los sistemas combinatorios) sino también del llamado vector de estado E.

Los estados internos se describen por un grafo denominado **Diagrama de Estados**, que no es más que un AFD con salida o máquina secuencial como las ya vistas, donde los círculos representan los estados (que a nivel circuital se almacenan en flip-flops) y los arcos representan las transiciones.

Los rótulos de los arcos indican los eventos que generan cambios de estado, que corresponden a los posibles valores del vector de entrada para un nuevo vector de entrada. Notemos que decir “nuevo vector de entrada” no implica que exista un “nuevo valor del vector de entrada” (el valor puede ser el mismo que el anterior). Por esto es necesario que el sistema tenga una forma de reconocer que debe procesar una nueva entrada (aunque el

valor no haya cambiado). Esto se logra típicamente por un reloj de sincronismo y es la forma que utilizaremos en este curso.

Notemos que las ecuaciones características de los circuitos secuenciales que presentamos al principio del capítulo:

$$Y = G(X, E)$$
$$E' = H (X, E)$$

formalmente corresponden a los circuitos que se diseñen en base a la Máquina de Mealy.

Si el diseño fuera hecho en base a la Máquina de Moore, las ecuaciones deberían lucir así:

$$Y = G(E)$$
$$E' = H (X, E)$$

ya que en el caso de Moore la salida depende solamente del estado actual de la máquina, mientras que el nuevo estado nuevo sí depende del estado actual y la entrada.

7.2 Diseño de Circuitos Secuenciales

El diseño de un circuito secuencial en base a las máquinas de estado (en particular en el curso utilizaremos la de Mealy) se realiza en base a la especificación del AFD que define el comportamiento esperado del circuito. Los estados se almacenan en flip-flops (tipo D ó J-K, sincrónicos por flanco) y las funciones G (de salida ó transferencia) y H (de transición) se implementan en lógica combinatoria.

En base a estos principios de diseño, se pueden establecer las siguientes etapas, las que deben ser ejecutadas para obtener el diseño del circuito:

- Modelado del sistema a implementar mediante la especificación del AFD correspondiente mediante un Diagrama de Estados (este es un paso no obligatorio, que conviene realizar por razones de claridad de visualización del comportamiento que se especifica).
- Deducir la “Tabla de Estados” (tabla que establece para cada estado cual es el próximo estado y cuáles las salidas en función de las entradas) a partir del Diagrama de Estados (o especificarla directamente en base al comportamiento esperado del AFD, si es que no se hizo el paso anterior).
- Determinar cantidad de bits y sistema de codificación para la(s) entrada(s) y la(s) salida(s).
- Determinar el número de flip-flops necesarios para codificar todos los estados posibles del sistema y determinar la codificación a utilizar.
- Incorporar la codificación de los estados, entradas y salidas a la Tabla de Estados para obtener la “Tabla de Transiciones y Salidas”.
- Seleccionar los flip-flops a utilizar y en base a las ecuaciones de los mismos pasar de la Tabla de Transiciones y Salidas a las tablas de verdad de las funciones lógicas que permitirán determinar el valor de la(s) salida(s) en base a la(s) entrada(s) y las salidas de los flip-flops, y el valor a presentar en las entradas de los flip-flops para almacenar el nuevo estado en ellos.
- Minimizar las expresiones lógicas en dos niveles mediante un método sistemático (en el curso utilizamos Diagramas de Karnaugh).
- Dibujar el circuito lógico resultante en base a los flip-flops y compuertas básicas (AND, OR y NOT).

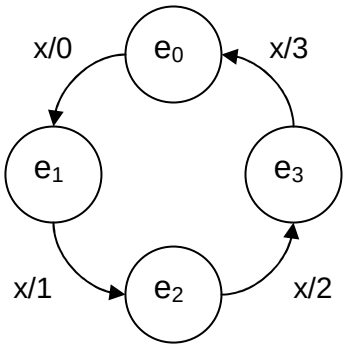
A continuación veremos ejemplos de cómo aplicar esta metodología de diseño de circuitos secuenciales.

7.3 Ejemplos

7.3.1 Contador Binario (modulo cuatro)

En este primer ejemplo vamos a diseñar un contador de módulo 4 (de 0 a 3). Este circuito es bastante particular ya que no tiene entrada, por lo que el cambio de estado es incondicional y ocurre en cada flanco ascendente del reloj. Si se quiere contar los cambios de nivel (flanco ascendente) de una señal cualquiera, basta utilizar esa señal como reloj del contador. Si, en cambio, se desea contar las transiciones de 0 a 1 de una señal en forma sincrónica respecto a un reloj general del sistema, entonces se debe utilizar el contador que veremos en el próximo punto.

Para construir el contador binario, de acuerdo al método propuesto, primero debemos especificar el problema mediante un AFD con salida (máquina de estado de Mealy), para lo que vamos a dibujar su diagrama de estados:



El siguiente paso es obtener la Tabla de Estados:

E_n	E_{n+1}	Salida
e_0	e_1	0
e_1	e_2	1
e_2	e_3	2
e_3	e_0	3

Tengamos presente que esta tabla no hace otra cosa que mostrar la información de las funciones de transición δ y de salida λ, resumidas en el comúnmente llamado “Diagrama de Estados” del AFD M = ({e0, e1, e2, e3}, {}, {0, 1, 2, 3}, δ, λ, e0) con salida dibujado.

El diagrama de estados tiene cuatro estados por lo que necesitamos dos flip-flops para poder representarlos (cuyas salidas anotaremos q_1 y q_0).

A continuación debemos asignar la codificación a cada estado. En este caso parece natural codificar en base al subíndice utilizado en la denominación de los estados expresado en binario. Lo mismo para la salida (el valor expresado en binario en 2 bits)

La Tabla de Transiciones y Salidas queda entonces así:

E_n $q_1\ q_0$	E_{n+1} $q_1\ q_0$	Salida $s_1 s_0$
00	01	00
01	10	01
10	11	10
11	00	11

Usaremos flip-flops tipo D, por lo que de la tabla anterior podemos pasar a la Tabla de Verdad, usando la ecuación del flip-flop tipo D:

$$Q_{n+1} = D_n$$

y queda:

$q_1\ q_0$	d_1	d_0	s_1	s_0
00	0	1	0	0
01	1	0	0	1
10	1	1	1	0
11	0	0	1	1

Ahora lo que resta es hacer los diagramas de Karnaugh que minimicen los circuitos combinatorios que implementarán la función de transición y la función de salida, en base a las tablas de verdad obtenidas. Dichos circuitos tienen como entrada la(s) entrada(s) del circuito secuencial, y el valor del estado anterior y como salidas deben dar el próximo estado (que oficiarán como entradas de los flip-flops tipo D) y la(s) salida(s) esperada(s) del circuito secuencial en su conjunto, respectivamente.

Dado que la codificación elegida para los estados coincide exactamente con la salida, resulta que el circuito combinatorio que implementa la función de salida es la identidad sobre salidas q de los FF. Por esto sólo nos resta minimizar el circuito combinatorio para la función de transición.

d_1

$q_1 \setminus q_0$	0	1
0	0	1
1	1	0

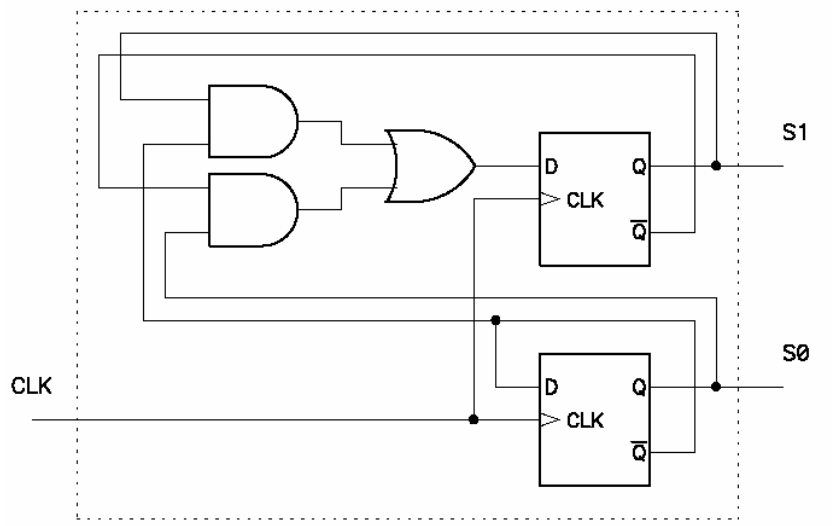
$$d_1 = q_1'.\ q_0 + q_1.\ q_0'$$

d_0

$q_1 \setminus q_0$	0	1
0	1	0
1	1	0

$$d_0 = q_0'$$

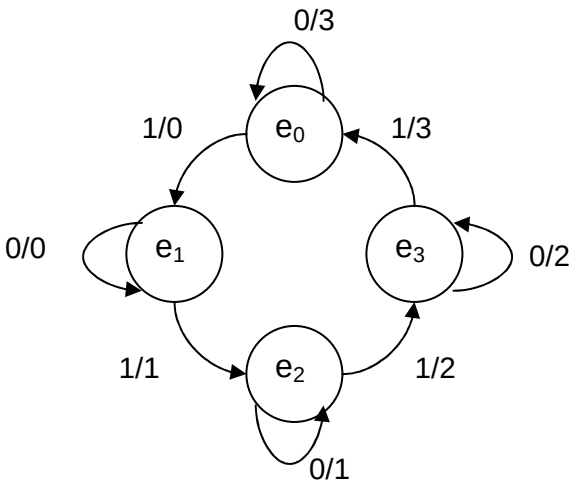
Quedando entonces el siguiente circuito secuencial:



7.3.2 Contador de Transiciones Binario (modulo cuatro)

En este ejemplo veremos el caso ya adelantado en el cual el interés es contar transiciones de una señal (de 0 a 1, es decir flancos ascendentes) en forma sincrónica con un reloj. Al igual que en el caso anterior vamos a diseñar un contador de módulo 4 (de 0 a 3). A diferencia con el contador del ejemplo anterior este circuito sí tiene entrada, y los cambios de estado se producirán cuando la entrada sea 1 al momento del flanco del reloj de sincronismo. Notar que en este caso solamente se tendrá en cuenta los valores de la señal de entrada al momento del flanco del reloj (si cambia durante el período del reloj el contador construido de esta forma no lo detectará).

Siguiendo nuevamente los pasos de diseño del método propuesto primero especificaremos el problema mediante un AFD con salida (máquina de estado de Mealy), que en este caso tendrá el siguiente diagrama de estados:



El siguiente paso es obtener la Tabla de Estados, en la que en este caso sí se debe tener en cuenta la entrada, por lo que la cantidad de filas se duplica, aunque para el caso de la entrada = 1 es igual a la anterior:

E_n	Entrada	E_{n+1}	Salida
e_0	0	e_0	3
e_0	1	e_1	0
e_1	0	e_1	0
e_1	1	e_2	1
e_2	0	e_2	1
e_2	1	e_3	2
e_3	0	e_3	2
e_3	1	e_0	3

Al igual que en el caso anterior tenemos cuatro estados por lo que necesitamos dos flip-flops para poder representarlos. Haremos la misma codificación de los estados en base al subíndice en binario. La Tabla de Transiciones y Salidas queda de la siguiente forma:

E_n q_1 q_0	Entrada e	E_{n+1} q_1 q_0	Salida s_1 s_0
00	0	00	11
00	1	01	00
01	0	01	00
01	1	10	01
10	0	10	01
10	1	11	10
11	0	11	10
11	1	00	11

En este caso también usaremos flip-flops tipo D, con lo que pasamos a las siguientes tablas de verdad:

q_1 q_0 e	d_1	d_0	s_1	s_0
000	0	0	1	1
001	0	1	0	0
010	0	1	0	0
011	1	0	0	1
100	1	0	0	1
101	1	1	1	0
110	1	1	1	0
111	0	0	1	1

Los diagramas de Karnaugh tienen ahora tres variables y resultan así:

d_1

$q_1\ q_0$ \ e	0	1
00		
01		1
11	1	
10	1	1

$$d_1 = q_1.e' + q_1.q_0' + q_1'.q_0.e$$

d_0

$q_1\ q_0$ \ e	0	1
00		1
01	1	
11	1	
10		1

$$d_0 = q_0.e' + q_0'.e$$

s_1

$q_1\ q_0$ \ e	0	1
00	1	
01		
11	1	1
10		1

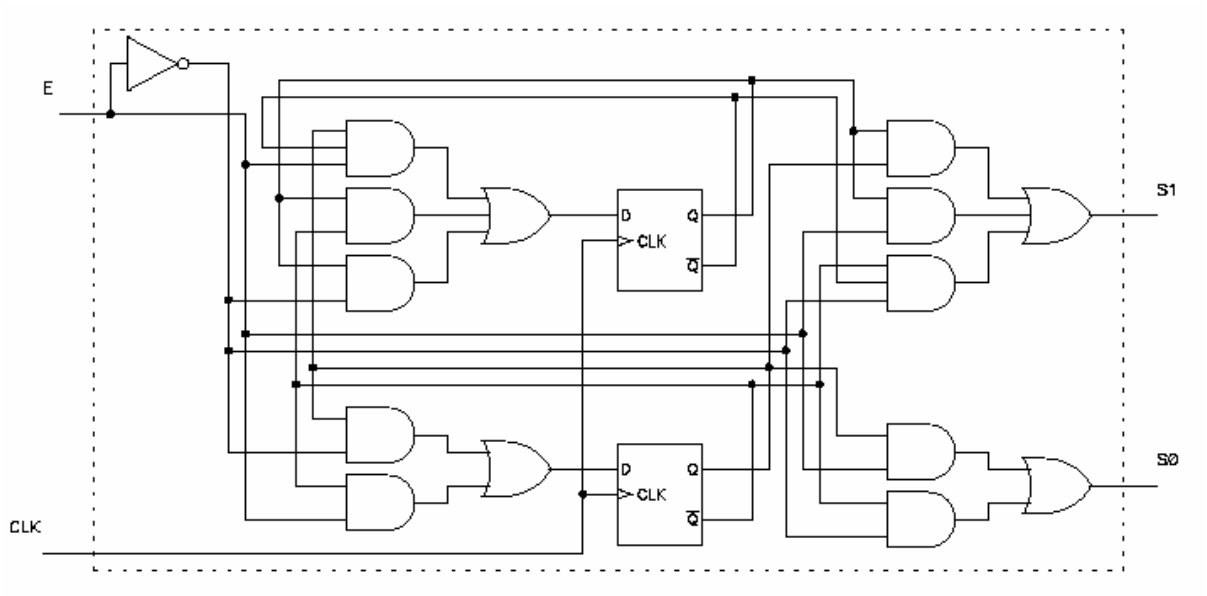
$$s_1 = q_1.q_0 + q_1.e + q_1'.q_0'.e'$$

s_0

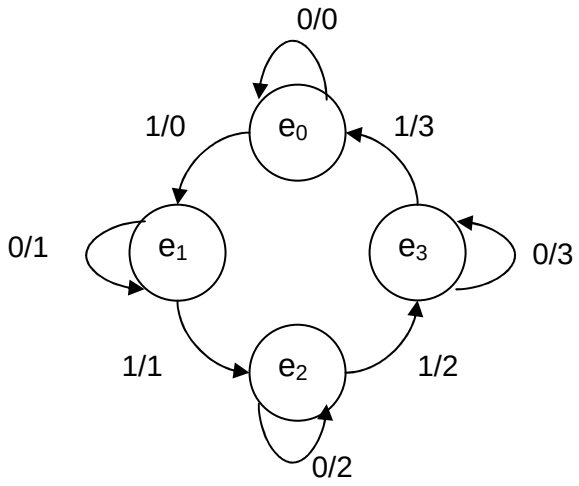
$q_1\ q_0$ \ e	0	1
00	1	
01		1
11		1
10	1	

$$s_0 = q_0'.e' + q_0.e$$

El circuito queda entonces:



Las funciones de salida están directamente vinculadas al diagrama de estados que especificamos para el contador, y dependen también de la codificación elegida para los estados. Notemos que en el diagrama que hicimos la salida quedó vinculada al identificador del estado hacia el que vamos. Si en cambio especificáramos que la salida coincida con el estado que abandonamos quedaría el diagrama de estados siguiente:



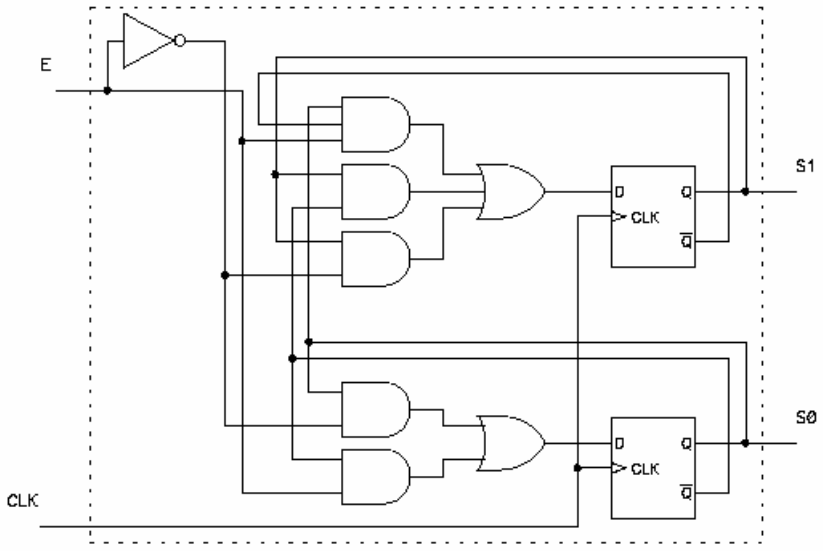
Notemos que el efecto que tiene este cambio sobre el comportamiento del circuito es que provoca que los cambios en el valor de las salidas del contador ocurren en el flanco descendente de la entrada (cuando cambia de 1 a 0), en vez del flanco ascendente (cambio de 0 a 1) como ocurría en el diseño original. Si mantenemos la codificación de los estados este diagrama se traduce en la siguiente Tabla de Transición y Salida:

E_n q_1 q_0	Entrada e	E_{n+1} q_1 q_0	Salida s_1 s_0
00	0	00	00
00	1	01	00
01	0	01	01
01	1	10	01
10	0	10	10
10	1	11	10
11	0	11	11
11	1	00	11

Usando la ecuación del flip-flop D pasamos a las tablas de verdad:

q_1 q_0 e	d_1	d_0	s_1	s_0
000	0	0	0	0
001	0	1	0	0
010	0	1	0	1
011	1	0	0	1
100	1	0	1	0
101	1	1	1	0
110	1	1	1	1
111	0	0	1	1

Como la tabla de transición (cambio de estado) no cambió, tampoco cambiarán las expresiones lógicas minimizadas de las entradas de los flip-flops (d_1 y d_0). Pero sí hay un cambio significativo en las funciones de salida. Es fácil ver que ahora $s_1 = q_1$ y $s_0 = q_0$. Por lo que el circuito se simplifica significativamente y queda:



7.3.3 Reconocedor de Secuencia

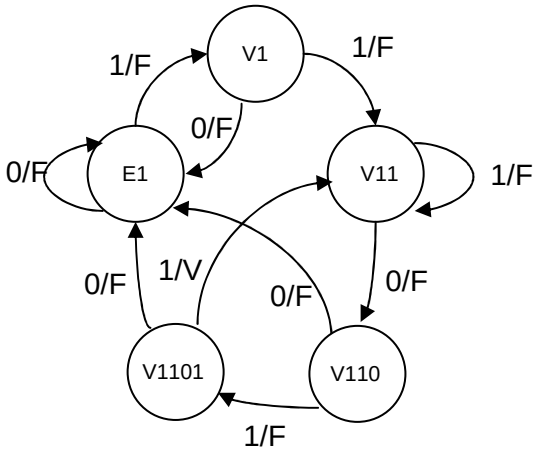
En este ejemplo construiremos un circuito que reconozca la secuencia 11011 en su único bit de entrada. La salida será Falsa (F) mientras la secuencia no sea reconocida. Será Verdadera (V) al momento de reconocerla y volverá a Falsa (F) para esperar una

nueva secuencia correcta.

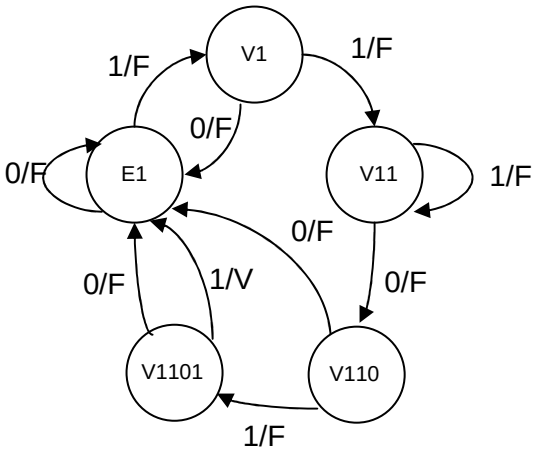
Comencemos por plantear el diagrama de estados de la máquina. Para ello consideramos un estado inicial E1 (espero 1) en el cuál la máquina estará hasta que llegue el primer "1" de la secuencia a reconocer. Cuando llega el primer "1" pasamos al estado V1 (vino 1). Estando en V1, si llega otro "1" pasamos al estado V11 (vino 1 y luego 1), pero si llega 0 volvemos al anterior (la secuencia debe comenzar de nuevo, ya que "10" no forma de la secuencia válida). En el estado V11 en caso que llegue un "0" entonces pasamos a un nuevo estado V110 (vino 1, luego 1 y luego 0). Si en cambio llega un "1" entonces debemos quedarnos en el mismo estado, porque significa que llegaron hasta ahora "111", lo que significa que también vino un "1" y luego otro "1". El próximo paso es analizar si estando en V110 llega un "1": hay que pasar al estado V1101 (ya que por ahora la secuencia puede ser válida), en cambio si llega un "0" debemos volver al inicio (E1) ya que la posible secuencia válida se frustró. Finalmente si estando en el estado V1101 viene un "0" entonces la salida será Falsa y pasaremos al estado E1 a esperar el primer "1" de una nueva posible secuencia válida. Si en cambio llega un "1" entonces la salida será Verdadera. Queda por definir a qué estado vamos. De hecho existen dos alternativas: si consideramos el "1" que llegó como fin de una secuencia válida y como posible comienzo de una nueva secuencia válida, entonces debemos pasar al estado V11 (ya que habrían llegado dos "1" seguidos de una posible nueva secuencia válida). En caso contrario debemos ir al estado E1.

Notar que en cualquier máquina secuencial para diseñar correctamente el diagrama de estados es necesario analizar qué sucede a partir de cada estado con todos y cada uno de los posibles valores de la entrada (en este caso se reduce a "0" ó "1", en otro caso podrían haber más posibilidades).

Alternativa 1:



Alternativa 2:



Vamos a continuar con la implementación de la alternativa 1. La tabla de salidas y transiciones unificada para este caso queda:

E_n	Entrada	E_{n+1}	Salida
E1	0	E1	F
E1	1	V1	F
V1	0	E1	F
V1	1	V11	F
V11	0	V110	F
V11	1	V11	F
V110	0	E1	F
V110	1	V1101	V
V1101	0	E1	F
V1101	1	V11	V

En este caso tenemos 5 estados, por lo que requerimos 3 flip-flops para almacenar la codificación de los mismos.

Debemos ahora codificar los estados, la entrada y la salida. La entrada ya está en booleano (0 ó 1) y la salida parece natural codificarla F $\rightarrow$ 0 y V $\rightarrow$ 1. Los estados los codificamos de 000 a 100 en el orden presentado en la tabla. Con estas codificaciones la Tabla de Transiciones y Salidas queda:

E_n $q_2\ q_1\ q_0$	Entrada e	E_{n+1} $q_2\ q_1\ q_0$	Salida s
000	0	000	0
000	1	001	0
001	0	000	0
001	1	010	0
010	0	011	0
010	1	010	0
011	0	000	0
011	1	100	0
100	0	000	0
100	1	010	1

Usando flip-flops tipo D pasamos de la Tabla de Transiciones y Salidas a las tablas de verdad de las funciones de transición y salida. Notemos que en este caso hay más códigos binarios de estados que estados reales, con lo que las tablas de verdad incluirán “X” (don’t care) en los lugares que correspondan a las codificaciones que no representan ningún estado de la máquina. Esto es así porque estas codificaciones pueden considerarse como estados "inalcanzables", ya que ninguna evolución de la máquina secuencial pasará por ellos. Por esta razón las funciones de transición y de salida para esos estados inalcanzables no importa qué valor toman (ya que nunca sucederán), por lo que puede usarse como valores "comodín" (X) a los efectos de lograr una mayor minimización de los circuitos.

$q_2\ q_1\ q_0$ e	q_2	q_1	q_0	s
0000	0	0	0	0
0001	0	0	1	0
0010	0	0	0	0
0011	0	1	0	0
0100	0	1	1	0
0101	0	1	0	0
0110	0	0	0	0
0111	1	0	0	0
1000	0	0	0	0
1001	0	1	0	1
1010	X	X	X	X
1011	X	X	X	X
1100	X	X	X	X
1101	X	X	X	X
1110	X	X	X	X
1111	X	X	X	X

A continuación realizaremos la minimización de las funciones, utilizando los diagramas de Karnaugh:

d_2

q_0e \ q_2q_1	00	01	11	10
00			X	
01			X	
11		1	X	X
10			X	X

$$d_2 = q_1.q_0.e$$

d_1

q_0e \ q_2q_1	00	01	11	10
00		1	X	
01		1	X	1
11	1		X	X
10			X	X

$$d_1 = q'_1.q_0.e + q_2.e + q_1.q'_0$$

d_0

$q_0e \setminus q_2q_1$	00	01	11	10
00		1	X	
01	1		X	
11			X	X
10			X	X

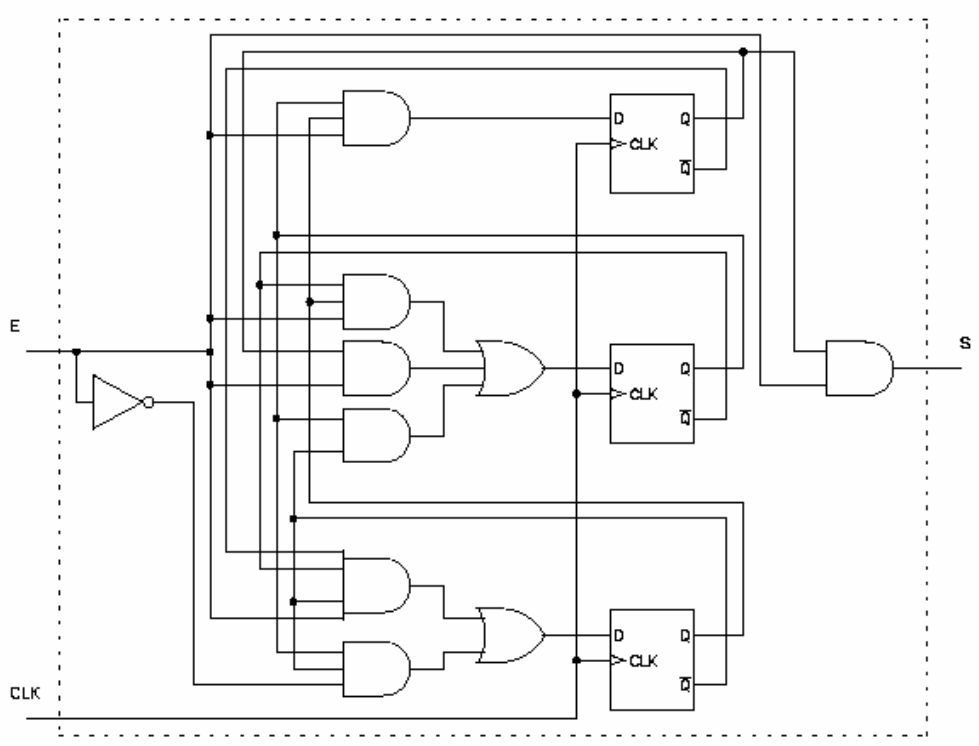
$$d_0 = q'_2.q'_1.q'_0.e + q_1.q'_0.e'$$

s

$q_0e \setminus q_2q_1$	00	01	11	10
00			X	
01			X	1
11			X	X
10			X	X

$$s = q_2.e$$

El circuito queda entonces de la siguiente forma:

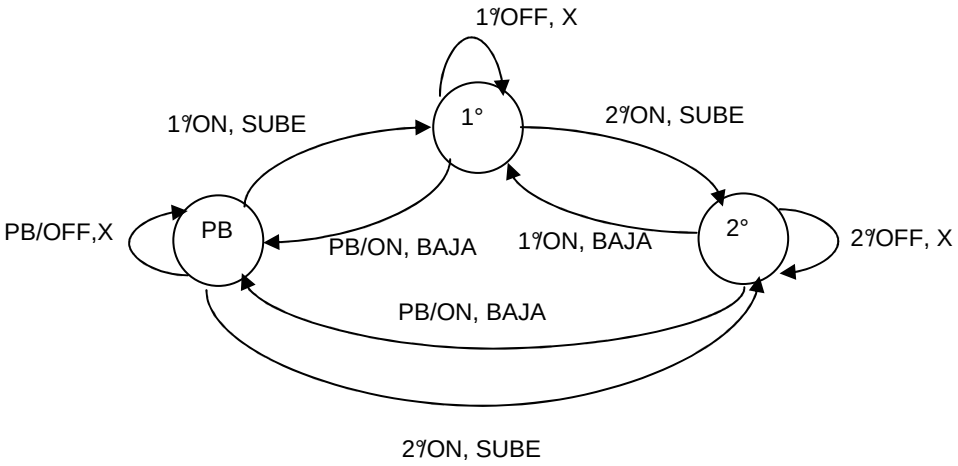


7.3.4 Control de Ascensor

Como forma de ejemplificar que los circuitos secuenciales no solamente son apropiados para generar o reconocer secuencias, como lo podría estar indicando su denominación y los ejemplos vistos hasta ahora, vamos a encarar el diseño de un circuito para controlar un modelo simplificado de un ascensor.

Este ascensor atenderá tres pisos: PB, 1° y 2°. Con sideraremos solamente la botonera interna en la caja como entrada (botones PB, 1° y 2°) y como salida deberemos controlar el encendido del motor (ON/OFF) y el sentido de marcha (SUBE/BAJA). No nos preocuparemos de cómo es que se detiene el motor al llegar al piso (obviamente en un control de ascensor real esto debería ser tenido en cuenta).

Comencemos con el diagrama de estados. Los estados "naturales" de este sistema parecen ser los pisos en los que está el ascensor (dado que hemos simplificado el modelo y no tenemos en cuenta el "viaje" de un piso a otro, no hay estados que no sean estar en un piso). En función del piso actual (estado) y de qué botón pulse el usuario (entrada) quedará determinado si el ascensor debe moverse (encendido del motor) y con qué sentido. No es difícil arribar al siguiente diagrama de estados, siendo la única consideración especial el hecho de utilizar la salida "X" (indiferente) en "sentido" en el caso que el motor esté detenido (ya que en ese caso no importa si sube o baja ya que no se mueve):



El próximo paso es escribir, a partir del diagrama de estados, la Tabla de Estados:

E_n	Entrada	E_{n+1}	Salidas	
			Motor	Sentido
PB	PB	PB	OFF	X
PB	1°	1°	ON	SUBE
PB	2°	2°	ON	SUBE
1°	PB	PB	ON	BAJA
1°	1°	1°	OFF	X
1°	2°	2°	ON	SUBE
2°	PB	PB	ON	BAJA
2°	1°	1°	ON	BAJA
2°	2°	2°	OFF	X

En este caso tenemos 3 estados por lo que precisamos 2 bits para representarlos (PB $\rightarrow$ 00, 1° $\rightarrow$ 01, 2° $\rightarrow$ 10). Por su lado vamos a codificar la entrada también en 2 bits (usaremos la misma codificación que para los estados), mientras que para las salidas tomaremos 1 bit para cada una (OFF $\rightarrow$ 0, ON $\rightarrow$ 1, BAJA $\rightarrow$ 0, SUBE $\rightarrow$ 1).

Con estas codificaciones la tabla anterior se transforma en la siguiente Tabla de Transiciones y Salidas:

E^n	Entrada	E^{n+1}	Salidas	
$q_1\ q_0$	$e_1\ e_0$	$q_1\ q_0$	motor	sentido
00	00	00	0	X
00	01	01	1	1
00	10	10	1	1
01	00	00	1	0
01	01	01	0	X
01	10	10	1	1
10	00	00	1	0
10	01	01	1	0
10	10	10	0	X

Si usamos flip-flops tipo D y aplicamos su ecuación, agregando los códigos de estados inalcanzables (como en el ejemplo anterior) y aprovechando también que hay entradas "inexistentes" (codificaciones que no corresponden a entradas reales), podemos pasar a las tablas de verdad de las funciones de transición y salida:

$q_1\ q_0\ e_1\ e_0$	q_1	q_0	m	s
0000	0	0	0	X
0001	0	1	1	1
0010	1	0	1	1
0011	X	X	X	X
0100	0	0	1	0
0101	0	1	0	X
0110	1	0	1	1
0111	X	X	X	X
1000	0	0	1	0
1001	0	1	1	0
1010	1	0	0	X
1011	X	X	X	X
1100	X	X	X	X
1101	X	X	X	X
1110	X	X	X	X
1111	X	X	X	X

El próximo paso es la minimización de las tablas de verdad de las entradas de los flip-flops y las salidas:

d_1

$e_1e_0 \setminus q_1q_0$	00	01	11	10
00			X	
01			X	
11	X	X	X	X
10	1	1	X	1

$$d_1 = e_1$$

d_0

$e_1e_0 \setminus q_1q_0$	00	01	11	10
00			X	
01	1	1	X	1
11	X	X	X	X
10			X	

$$d_0 = e_0$$

m

$e_1e_0 \setminus q_1q_0$	00	01	11	10
00		1	X	1
01	1		X	1
11	X	X	X	X
10	1	1	X	

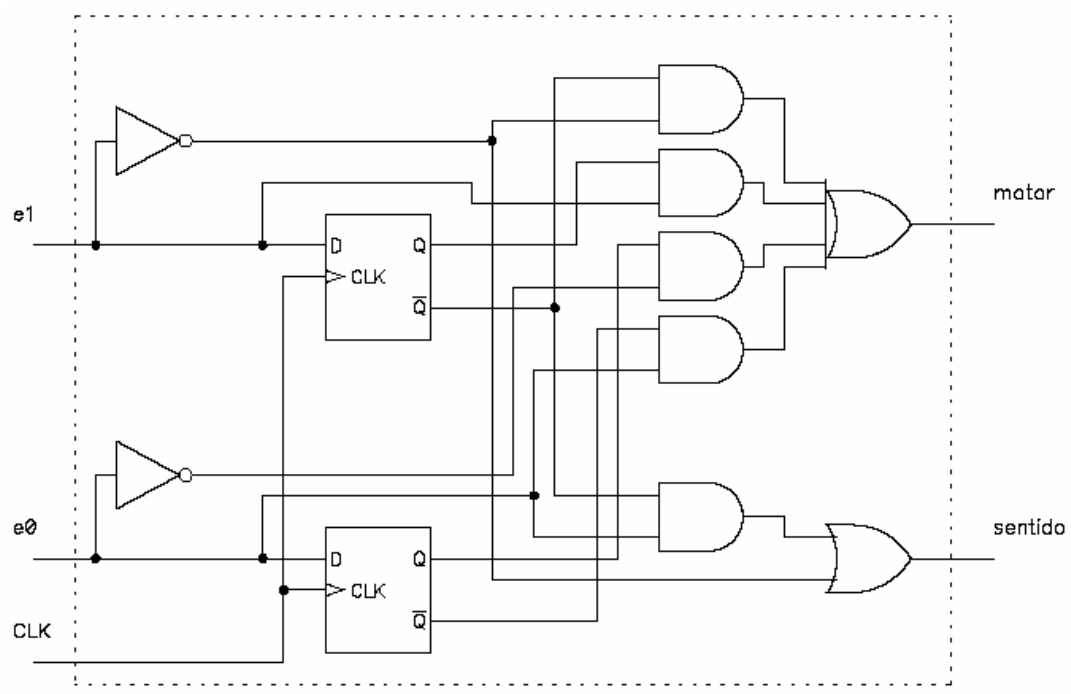
$$m = q'_1.e_1 + q_1.e'_1 + q_0.e'_0 + q'_0.e_0$$

s

$E_1e_0 \setminus q_1q_0$	00	01	11	10
00	X		X	
01	1	X	X	
11	X	X	X	X
10	1	1	X	X

$$s = e_1 + q'_1.e_0$$

El circuito queda como se ve en la figura siguiente:


e1
e0
CLK
D
Q
CLK
Q
D
Q
CLK
Q
matar
sentido