11 INTRODUCCIÓN A LOS SISTEMAS SECUENCIALES I: CONCEPTOS

Los sistemas combinacionales se construyen mediante funciones booleanas de sus variables de entrada. Pero no todo sistema digital es combinacional: existen sistemas en que la correspondencia entre el vector de entrada y el vector de salida no es unívoca; es decir, no se pueden obtener las salidas como funciones de «sólo» las entradas.

Los sistemas secuenciales necesitan recordar su pasado, la «secuencia de vectores de entrada» a través de la cual se ha llegado a la situación presente; para ello han de tener «memoria», que se configura mediante un vector de «estado» que contiene la información que el sistema necesita sobre su pasado. La memoria se consigue mediante «realimentación lógica»: las variables de estado son, a la vez, variables de salida y de entrada en las funciones de evolución del estado.

Tres vectores: entrada, estado y salida, y dos conjuntos de funciones: las de evolución del estado (estado anterior y entradas determinan el nuevo estado) y las de activación de las salidas (que dependen, también, del estado y de las entradas).

La evolución de un sistema secuencial puede ser representada mediante un grafo de estados, a partir del cual pueden construirse las funciones de modificación del estado. En ocasiones, hay dos formas diferenciadas de configurar el grafo de estados: distinguiendo salidas distintas con estados diferentes (autómata de Moore) o reduciendo el número de estados al mínimo y admitiendo que un mismo estado pueda tener varios vectores de salida (autómata de Mealy).

El presente capítulo presenta gradualmente los conceptos básicos necesarios para comprender y describir los sistemas secuenciales, a través de una serie de ejemplos; se reserva para el capítulo siguiente la metodología de diseño secuencial.

La necesidad de memoria, como recuerdo del pasado de un sistema secuencial, se concreta en el concepto de estado. Pero existe otra perspectiva complementaria que nos lleva a la memoria como necesidad de almacenar datos y resultados, es decir, conservar información que puede ser utilizada posteriormente. El biestable como celda capaz de almacenar un bit y el registro (conjunto de biestables) como bloque capaz de almacenar una palabra binaria son los elementos básicos de la memoria.

Precisamente, el siguiente capítulo desarrollará el diseño de circuitos secuenciales, utilizando los biestables como celdas que contienen las variables de estado.

Tomás Pollán Santamaría. Tecnología Electrónica. E.U.I.T.I.Z. Universidad de Zaragoza. tpollan@posta.unizar.es

11.1. Necesidad de memoria: concepto de estado

Los sistemas combinacionales presentan una correspondencia unívoca entre el conjunto de vectores de entrada y el de vectores de salida.

Un sistema combinacional es del tipo estímulo-respuesta: al recibir un vector de entrada dado produce un vector de salida determinado, siempre el mismo; la salida es respuesta directa respecto a la entrada y, por ello, puede obtenerse por «combinación booleana» de los valores de las entradas.

En un sistema combinacional las salidas son función booleana directa de los valores presentes en las variables de entrada en dicho momento:

vector de entrada $\mathbf{X = (x_i) = (x_1, x_2, x_3, x_4, \ldots, x_m)}$

vector de salida $\mathbf{Y = (y_j) = (y_1, y_2, y_3, \ldots\ldots, y_n)}$

$\mathbf{Y = F(X)}$ $\mathbf{y_j = f_j(x_i) = f_j(x_1,x_2,x_3,x_4,\ldots,x_m)}$ para $\mathbf{j = 1, 2, \ldots, n.}$

Conocido el funcionamiento requerido para un sistema combinacional, esto es, su tabla funcional o «tabla de verdad», a partir de ella pueden obtenerse directamente las funciones booleanas de las entradas que determinan las salidas del sistema. A cada vector de entrada le corresponde uno y solo un vector de salida: la correspondencia vectores de entrada → vectores de salida es unívoca.

No todo sistema digital es combinacional. No toda correspondencia entre el conjunto de vectores de entrada y el de vectores de salida es unívoca.

Existen sistemas digitales que no encajan dentro de este tipo combinacional, sistemas que respecto a un mismo vector de entrada, en momentos diferentes, producen vectores de salida distintos. Un ejemplo sencillo y práctico lo constituye el sistema de apertura y cierre de una puerta de garaje (del tipo de persiana metálica) que actúe de la forma siguiente:

• *el sistema dispone de tres entradas: un pulsador de activación o llave de contacto* ***P*** *y dos detectores o topes de fin de carrera* $\boldsymbol{T_S}$ *y* $\boldsymbol{T_i}$ *(superior e inferior); dispone asimismo de dos salidas: movimiento hacia arriba* ***S*** *y movimiento hacia abajo* ***B*** *de la puerta*

• *al presionar el pulsador* ***P****, si la puerta está abajo (tope inferior* $\boldsymbol{T_i}$ *activado), se inicia el movimiento ascendente* ***S****, hasta que se activa el tope superior* $\boldsymbol{T_S}$*; en cambio, si es el tope superior* $\boldsymbol{T_S}$ *el que se encuentra activado en el momento de pulsar* ***P*** *(puerta arriba), la puerta efectúa el movimiento de bajada* ***B*** *hasta llegar al tope inferior* $\boldsymbol{T_i}$*.*

Es decir:	$\mathbf{P=0, T_i=1, T_S=0}$	**S=0 y B=0**
	$\mathbf{P=1, T_i=1, T_S=0}$	**S=1 y B=0**
	$\mathbf{P=0, T_i=0, T_S=1}$	**S=0 y B=0**
	$\mathbf{P=1, T_i=0, T_S=1}$	**S=0 y B=1**

pero para el vector de entrada **P=0, T_i=0, T_S=0**

son posibles dos vectores de salida distintos **S=1 y B=0** puerta subiendo

S=0 y B=1 puerta bajando.

Los sistemas secuenciales necesitan memoria. La memoria se consigue mediante realimentación.

Para un mismo vector de entrada existen dos vectores de salida posibles. En definitiva, ¿qué debe hacer la puerta? ¿subir o bajar?... Está claro que si la puerta estaba previamente subiendo seguirá hacia arriba y si se encontraba bajando, lo seguirá haciendo. Es decir, con entrada **000** la salida de control de la puerta dependerá de su estado anterior.

En cada momento, el sistema optará por uno de los vectores de salida posibles para la misma entrada y esa opción la hará en función de las situaciones por la que ha pasado el sistema; es decir, su salida dependerá, además de los valores presentes en las entradas, de la secuencia de vectores de entrada anteriores por la que ha pasado. Este tipo de sistemas se denominan «secuenciales».

El concepto de sistema secuencial está directamente relacionado con el de «memoria lógica». De alguna manera el sistema de control de la puerta ha de guardar «memoria» sobre si anteriormente había comenzado a subir o a bajar. Este tipo de memoria puede construirse mediante la «realimentación» de algunas variables (por ejemplo, en este caso, la variable **S**), que «vuelven hacia atrás» para actuar también como entradas del sistema.

De esta forma la situación **P=0, T_i=0, T_S=0** se divide en dos:

P=0, T_i=0, T_S=0, S=0 **S=0 y B=1** la puerta continúa bajando,

P=0, T_i=0, T_S=0, S=1 **S=1 y B=0** la puerta sigue subiendo.

La variable **S** al realimentarse y actuar como entrada informa al sistema sobre si se encontraba subiendo o bajando y elimina la duplicidad de salidas. En términos conceptuales, el valor actual de la variable **S** influye en la evolución futura del sistema, el presente afecta al futuro: «memoria». Existe, pues, una relación directa entre memoria y realimentación lógica.

Sistemas secuenciales son aquellos sistemas digitales cuya salida en algún momento depende, no solamente de las entradas del sistema en dicho momento, sino también de la evolución anterior del sistema, es decir, de la secuencia anterior de vectores de entrada a través de la cual se ha llegado al momento actual. En este sentido se dice que el circuito posee memoria.

Construyamos las funciones booleanas del ejemplo anterior:

Variables de entrada **P, T_i,T_S** Variables de salida **S, B**

Variable de estado **S**

En este caso se requiere una sola variable de estado (el sistema debe guardar memoria solamente de si la puerta se encuentra subiendo o bajando) y dicha variable de estado coincide con una de las variables de salida **S**.

Tabla de evolución del estado:

P	**T_i**	**T_S**	**S**	**S**	
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	X	entradas
0	1	1	1	X	no posibles
1	0	0	0	0	
1	0	0	1	1	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	X	no
1	1	1	1	X	posibles

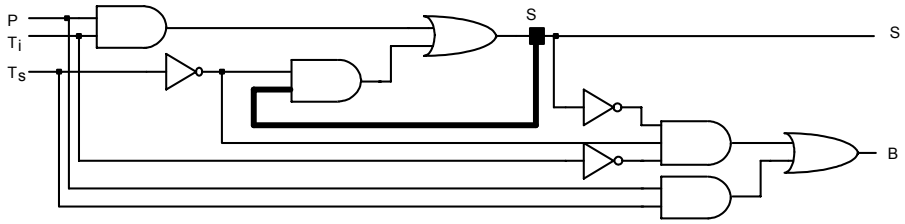
	T_S S			
P T_i	*00*	*01*	*11*	*10*
00	0	1	0	0
01	0	1	X	X
11	1	1	X	X
10	0	1	0	0

$S(t+\Delta t) = P \cdot T_i + \overline{T_s} \cdot S$: la variable de estado **S** se activa cuando se pulsa **P** estando la puerta abajo **T_i** y permanece activa hasta alcanzar el tope superior **T_S**.

Una de las funciones de salida consiste en la identificación de la variable de estado **S** con una salida **S = S**; la otra se obtiene de su correspondiente tabla booleana:

	T_S **S**			
P T_i	*00*	*01*	*11*	*10*
00	1	0	0	0
01	0	0	X	X
11	0	0	X	X
10	1	0	1	1

$B = P \cdot T_s + \overline{T_i} \cdot \overline{T_s} \cdot \overline{S}$: la variable de salida **B** se encontrará activada al pulsar **P** cuando la puerta se encuentre arriba T_S y, también, cuando no lo estén los topes inferior y superior T_i y T_S, ni la subida **S**.



En la figura anterior se ha resaltado la realimentación booleana que realiza la memoria del estado del sistema (variable **S**).

El estado (el conjunto de variables de estado) representa la información sobre la secuencia previa de vectores de entrada que el sistema secuencial necesita. De forma que un sistema secuencial presenta tres vectores propios: el de entrada, el de salida y el de estado, y dos correspondencias funcionales: la de evolución del estado y la de activación de las salidas

Un sistema secuencial se caracteriza por responder de forma distinta a un mismo vector de entrada en momentos diferentes, dependiendo de la secuencia previa de vectores de entrada. Ahora bien, para que pueda discriminar entre diversas secuencias de vectores de entrada es necesario que posea unas «variables de estado» que, de alguna forma, guarden información sobre la historia del circuito (historia = pasado = secuencia anterior).

En función del valor de las entradas en un momento dado y del valor de las variables de estado en dicho momento el circuito responde con un determinado conjunto de valores en sus salidas y, a la vez, modifica sus variables de estado para tomar en cuenta el nuevo vector de entrada.

Las variables de estado actúan, a la vez, como variables de salida y como variables de entrada en las funciones booleanas que configuran el sistema digital. Como variables de salida expresan el estado en que se encuentra el sistema como consecuencia de la secuencia de vectores de entrada anteriores. Como variables de entrada actúan en la modificación de dicho estado a la llegada de un nuevo vector de entrada y, también, determinan, junto con el vector de entrada, las salidas del sistema en cada momento.

La realimentación de las variables de estado, es decir, el hecho de que, siendo salidas intermedias del sistema, vuelven a la entrada del mismo, es lo que proporciona memoria al sistema secuencial (información sobre su historia pasada).

Algunas de las variables de estado, o en ocasiones todas ellas, pueden coincidir con variables de salida (en el ejemplo anterior, la variable de estado es una de las de salida, **S**), pero no siempre es posible tal identificación; en general, las variables de estado no coincidirán ni tendrán relación directa con las variables de salida del sistema; existen tres conjuntos de variables diferentes: entrada, salida y estado.

Incluso en aquellos casos en que alguna variable de estado coincida con una variable de salida, hay una distinción conceptual neta entre el vector de salida de un sistema y el estado del mismo: el estado representa la memoria que posee de su pasado anterior.

Un sistema secuencial dispone de tres conjuntos de variables:

vector de entrada $\mathbf{X = (x_i)}$, vector de salida $\mathbf{Y = (y_j)}$ y estado $\mathbf{Q = (q_k)}$

relacionados a través de dos conjuntos de funciones booleanas:

- función de salida $\mathbf{Y = F(X, Q)}$

- evolución del estado $\mathbf{Q(t+\Delta t) = G(X(t), Q(t))}$

La primera función representa la formación de las salidas del sistema como combinación de las entradas y del estado en cada momento; la segunda, muestra la evolución del estado del sistema en función de las entradas y del propio estado en el momento anterior.

El conjunto de variables de estado, el vector de estado, contiene la información que el sistema precisa para distinguir entre las diferentes secuencias de vectores de entrada que, teniendo un mismo vector al final de la secuencia, conducen a vectores de salida distintos.

11.2. Variables de estado y grafos de estados

Las variables de estado de un sistema secuencial representan conjuntamente el estado del sistema; individualmente cada variable de estado puede no tener significado físico alguno.

Conceptualmente, las variables de estado tienen un significado conjunto, el de numerar (y, por tanto, diferenciar) los diversos estados por los que evoluciona el sistema. El vector de estado representa la memoria que el sistema tiene de su evolución anterior.

¿Qué representa cada una de las variables de estado? Individualmente no tiene por qué tener un significado físico concreto; en principio, cada variable de estado es una variable «interna», que no tiene que corresponder necesariamente a ninguna variable de salida ni a ninguna situación determinada.

El significado corresponde al vector de entrada en su conjunto, sin tener cada una de sus variables una significación individualizada; si bien, en muchos casos, pueden tenerla por identificarse directa o indirectamente con alguna variable de salida o con alguna variable situacional del sistema.

El ejemplo del apartado anterior (puerta de garaje que se activa para subir y para bajar con un pulsador) es muy interesante por su sencillez y por ser un caso práctico muy habitual pero presenta aspectos que pueden inducir a error conceptual:

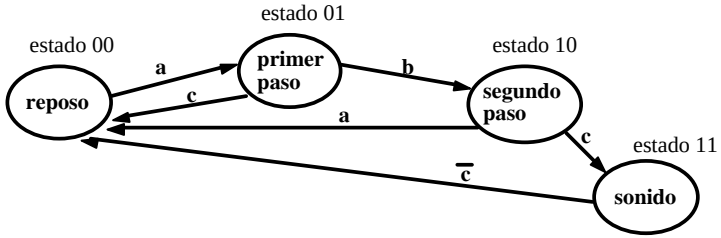
- la variable de estado coincide con una de las variables de salida, lo cual no tiene porque ser así: el estado y las salidas son, en principio, dos vectores diferentes, cuya función y significado son bien distintos (si bien es cierto que, en algunos casos, pueden coincidir algunas de sus variables);

- precisamente debido a esta identificación salida-estado, en el apartado anterior se dice que el sistema tiene que recordar «lo que estaba haciendo» (subir o bajar), afirmación que no es correcta conceptualmente: el sistema tiene que recordar su pasado en términos de vectores de entrada recibidos anteriormente, ha de recordar la secuencia de vectores de entrada a través de la cual ha llegado hasta la situación actual.

El control de la puerta de garaje, cuando la entrada es **000** (P T_s T_i), debe recordar si anteriormente provenía de la situación de entrada **101** (empezar a subir) o de la situación **110** (comenzar a bajar).

Consideremos otro ejemplo: *un simple timbre con tres pulsadores **a b c**, enlazados por un mecanismo que impide pulsar dos a la vez; el timbre debe sonar cuando se pulsan sucesivamente los tres pulsadores, primero el **a**, luego el **b** y, por último, el **c** y el sonido cesa al soltar el pulsador **c**.*

Podemos representar este tipo de actuación en la forma que sigue:



El sistema debe acordarse de 4 situaciones o estados diferentes, que pueden ser representados con dos variables de estado **q_2** y **q_1**:

$q_2 = 0$	**$q_1 = 0$**	estado de reposo
$q_2 = 0$	**$q_1 = 1$**	primer paso en la secuencia **a** b c
$q_2 = 1$	**$q_1 = 0$**	segundo paso en la secuencia a **b** c
$q_2 = 1$	**$q_1 = 1$**	se ha completado la secuencia: el timbre suena.

La secuencia de vectores de entrada (**a b c**) correcta es la siguiente:

000, **100**, **000**, **010**, **000**, **001**.

La variable de salida (sonido del timbre) se activa en el estado **11**: **$y = q_2.q_1$**

¿Qué representan las variables **q_2** y **q_1**? Individualmente cada una de ellas no tienen significado; las dos juntas expresan el estado del sistema. El estado tiene sentido físico; cada variable de estado no tiene por qué tenerlo.

Construyendo la tabla de evolución de los estados **q_2 q_1 a b c** $\rightarrow$ **q_2 q_1** (32 filas), pueden obtenerse las funciones **q_2 q_1**:

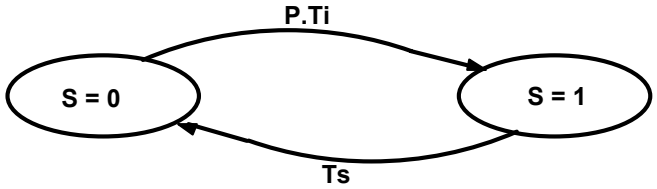
$$q_1^+ = q_2.c + \overline{q_2}.a + \overline{q_2}.q_1.\overline{c}.\overline{b} \qquad q_2^+ = q_2.c + q_2.\overline{q_1}.\overline{a} + \overline{q_2}.q_1.b\,.$$

La evolución de los estados puede expresarse eficazmente en un grafo.

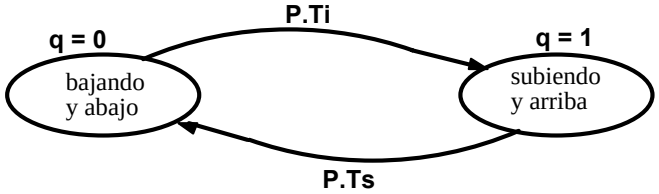
La evolución del estado de un sistema secuencial puede ser representada mediante un grafo de estados. Un grafo de estados detalla los diversos estados del sistema y las transiciones entre estados; las transiciones se representan mediante arcos orientados desde el estado de partida hasta el estado resultante de la transición, con indicación de la condición booleana que determina dicha transición.

La figura anterior (al inicio de esta misma página) corresponde al grafo de estados del circuito de control del timbre con tres pulsadores **a b c**.

El grafo de estados del control de la puerta de garaje del apartado anterior será:



También sería correcto un grafo más simétrico, en la forma que sigue:



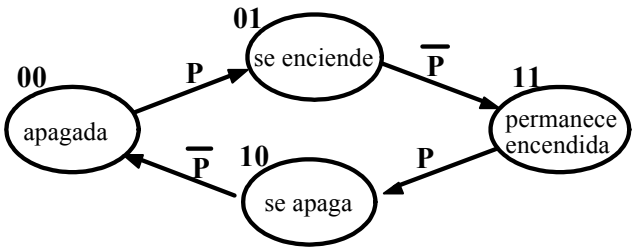
En este caso las funciones de activación de las salidas serán:

S (subir) = $q \cdot \overline{T_s}$ B (bajar) = $\overline{q} \cdot \overline{T_i}$

y la función de evolución del estado: $q^+ = P \cdot T_i + q \cdot (\overline{p \cdot T_s})$.

En este segundo grafo de control de la puerta de garaje, la variable de estado **q** no se corresponde con ninguna de las salidas del sistema **S B,** sino que distingue entre dos situaciones: la de subiendo, incluida la puerta parada arriba **q = 1** y la de bajando con inclusión de la puerta parada abajo **q=0**.

Otro ejemplo también muy habitual y sencillo: *la lámpara de luz de mesa con un pulsador, que se enciende al pulsar una vez y se apaga al pulsar una segunda vez.*



Obsérvese que para diferenciar un pulso del siguiente es necesario detectar P y $\overline{P}$; la omisión de las transiciones con $\overline{P}$ conduciría a un grafo erróneo: con P = 1 se producirían varias transiciones, que deberían corresponder a pulsos diferentes.

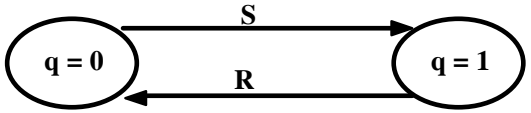
Son necesarios cuatro estados y dos variables de estado **q_2 q_1**; numerando los estados según el código Gray, la secuencia de funcionamiento es **00 → 01 → 11 → 10** y la variable de salida (lámpara encendida) coincide con **q_1**: **y = q_1**.

La tabla de evolución de estados permite obtener las funciones **q_2 q_1**:

q_2	q_1	P	q_2^+	q_1^+	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	1	1	
0	1	1	0	1	
1	0	0	0	0	$q_2^+ = q_2.P + q_1.\overline{P}$
1	0	1	1	0	
1	1	0	1	1	$q_1^+ = \overline{q_2}.P + q_1.\overline{P}$
1	1	1	1	0	

En este caso, la variable **q_1** coincide con la salida (encendido de la lámpara) y, en cambio, la variable **q_2** no se refleja exteriormente: es una variable de estado necesaria para diferenciar un pulsado del siguiente.

El ejemplo de sistema secuencial más simple posible es aquel que solamente tiene dos estados **q = 0** y **q = 1**; supongamos que posee dos entradas, una **S** (*set*) para marcar (poner a **1**) a la variable de estado **q** y la otra **R** (*reset*) para borrarla (llevar a **0**). Por ejemplo, *una lámpara con dos pulsadores diferentes **S** para encenderla y **R** para apagarla (y suponemos que nunca se pulsan los dos a la vez, pues resulta contradictorio desear encender y apagar la lámpara al mismo tiempo).*



La variable de salida coincide con la de estado y su tabla de evolución será la siguiente:

q	S	R	q^+	
0	0	0	0	
0	0	1	0	
0	1	0	1	
0	1	1	X	
1	0	0	1	
1	0	1	0	
1	1	0	1	$q^+ = S + q.\overline{R}$
1	1	1	X	

Precisamente este tipo de sistema secuencial mínimo será considerado en el próximo apartado y recibe el nombre de biestable por tener solamente dos estados.

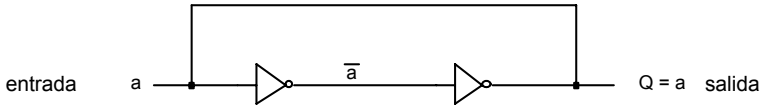
11.3. La memoria como almacén de información: biestables y registros

En los dos apartados anteriores consideramos la memoria referida al estado de un sistema secuencial: la necesidad de recordar la secuencia de vectores de entrada anteriores y el estado como memoria efectiva de dicha secuencia. En este apartado consideraremos la memoria desde otra perspectiva: como almacén de información, es decir, conservación de los valores booleanos de algunas variables.

Los sistemas combinacionales permiten «procesar» la información, tanto numéricamente (operaciones de cálculo) como lógicamente (combinación de proposiciones) y, también, distribuirla adecuadamente (multiplexado) y efectuar cambios entre diferentes codificaciones de la misma información (codificadores).

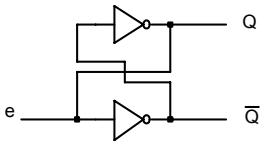
Ahora bien, tales sistemas no bastan para manejar eficientemente números y palabras binarios; se requiere, además, disponer de registros o memorias donde almacenar la información, son necesarios unos «módulos» que almacenen y conserven las palabras digitales (conjuntos de ceros y unos) que contienen la información.

La unidad elemental de memoria será un dispositivo capaz de almacenar un **0** o un **1** y de conservarlo en ausencia de entrada; tal sistema recibe el nombre de *biestable* (dos estados estables). La forma más simple de configurar un biestable (y, a la vez, su forma «conceptual» básica) consiste en conectar dos inversores en lazo cerrado.



Al establecer durante un momento un valor booleano sobre la entrada de este par de inversores, el primero de ellos invierte dicho valor y el segundo vuelve a invertirlo, coincidiendo con el valor de la entrada: no es necesario mantener exteriormente el valor booleano en la entrada pues, aun dejando libre dicha entrada, el acoplo cerrado de los dos inversores conserva el estado adquirido.

El sistema se encontrará en uno de sus dos estados estables (**Q=0** ; **Q=1**) dependiendo de que el último valor booleano aportado a su entrada sea **0** ó **1**. La existencia de dos estados estables da lugar a la denominación de *biestable*; si bien, con mucha frecuencia se utiliza el nombre de *Flip-Flop* **FF**, cuya derivación es meramente onomatopéyica.



2 estados estables

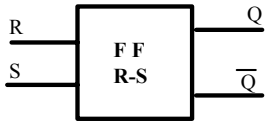
Q= 0 , $\overline{Q}$= 1

Q= 1 , $\overline{Q}$= 0

En un sistema digital no tiene sentido «dejar libre la entrada», pues cada entrada estará conectada a la salida de otra puerta o a una entrada exterior o a un sistema anterior que le proporcionará un **0** o un **1** y no tiene significado afirmar «la ausencia de entrada». En tal sentido, se definen los biestables **RS** y **D** en la forma que a continuación se detalla.

11.3.1. Biestable RS

El biestable **RS** presenta dos entradas **R** y **S,** la primera de ellas **R** (*reset*) de puesta a **0** o borrado y la otra **S** (*set*) de puesta a **1** o marcado, no siendo admisible activar ambas entradas a la vez. De esta forma el marcado (memorización de un **1**) y el borrado (almacenamiento de un **0**) se realizan por dos entradas distintas.



R S	Q
0 0	conserva su valor anterior
0 1	1
1 0	0
1 1	valores de entrada no admisibles

Se puede configurar un biestable **RS** directamente con puertas "**o-negada**" (*Nor*) o con puertas "**y-negada**" (*Nand*):

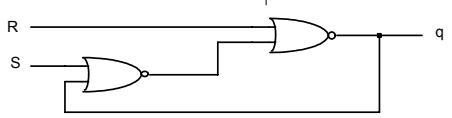
R	S	q	q^+
0	0	0	0
0	0	1	1
0	1	X	1
1	0	X	0
1	1	X	¿?

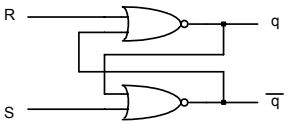
¿? = 0 borrado prioritario

$q^+ = \overline{R} . (S + q) = R \Delta (S \Delta q)$

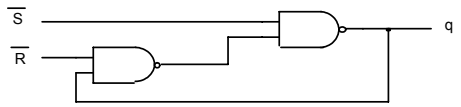
¿? = 1 marcado prioritario

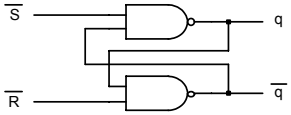
$q^+ = S + \overline{R} . q = \overline{S} * (\overline{R} * q)$





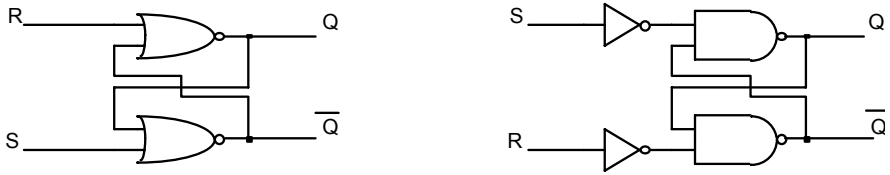
Biestable RS configurado con puertas *Nor*





Biestable RS configurado con puertas *Nand*

El vector de entrada **11** no es admisible, pues si va seguido del vector **00**, la situación posterior del biestable no es predecible, ya que dependerá de los transitorios de conmutación de las dos entradas **RS** (en principio, dependerá de cual de ellas se retrase más en pasar a **0**).



En ambos casos se utilizan dos puertas en lazo cerrado; los dos esquemas son estructuralmente análogos (difieren en su respuesta para el vector de entrada **R=S=1**, situación que no debe producirse) pero presentan dos diferencias a tener en cuenta:

a) en el circuito con puertas "**o-negada**" **R** actúa sobre el terminal superior y **S** sobre el inferior mientras que en el que utiliza puertas "**y-negada**" es al revés

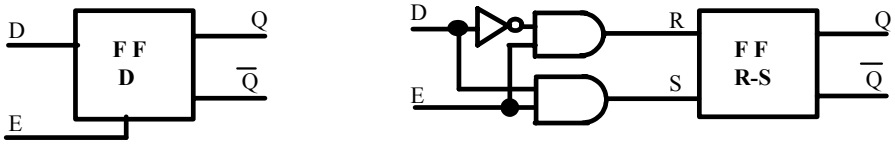
b) los valores activos sobre dichas entradas son el **1** en las puertas "**o-negada**" y el **0** ($\overline{R}$, $\overline{S}$) en la configuración con puertas "**y-negada**" (o bien, la inclusión de inversores en dicha configuración).

11.3.2. Biestable D

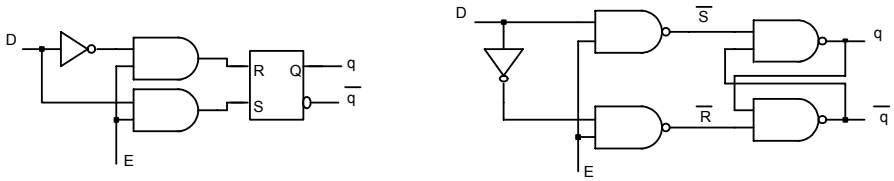
El biestable **D** posee una entrada de almacenamiento **D** y otra de habilitación **E,** de forma que cuando el biestable es habilitado (**E=1**) almacena el valor booleano presente en la entrada **D** y lo conserva hasta una nueva habilitación. Si durante el tiempo de habilitación dicho valor de entrada se modifica, el biestable va aceptando los diversos valores presentes en **D**, reteniendo el último de ellos cuando la habilitación desaparece.

Para configurar un biestable **D** a partir de un biestable **RS**, basta activar sus entradas **R** y **S** con las siguientes condiciones booleanas:

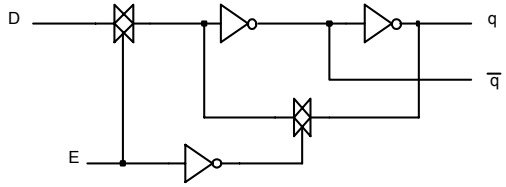
$$S = D \cdot E \qquad R = \overline{D} \cdot E$$



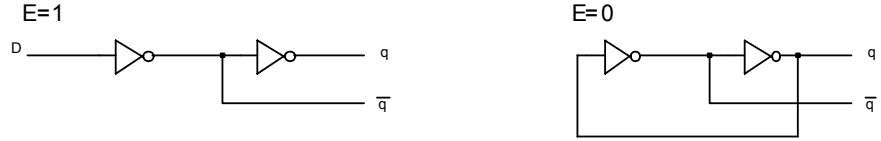
Tal esquema, realizado con puertas "**y-negada**", queda en la forma:



Este biestable puede construirse también con puertas de transmisión:

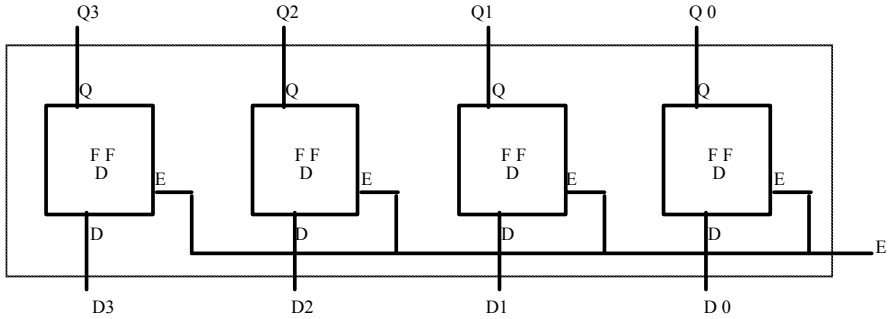


Cuando **E = 1** la primera puerta conduce y el valor presente en **D** se comunica a la salida **q**; al hacerse **E = 0**, la entrada **D** queda aislada y el valor de la salida se conserva gracias a la realimentación que realiza la segunda puerta de transmisión.



[En tecnologías MOS la opción de puertas de transmisión presenta ventajas respecto a la de puertas *Nand*: menor número de transistores, del orden de la mitad, que repercuten en menor área de integración y menores tiempos de propagación].

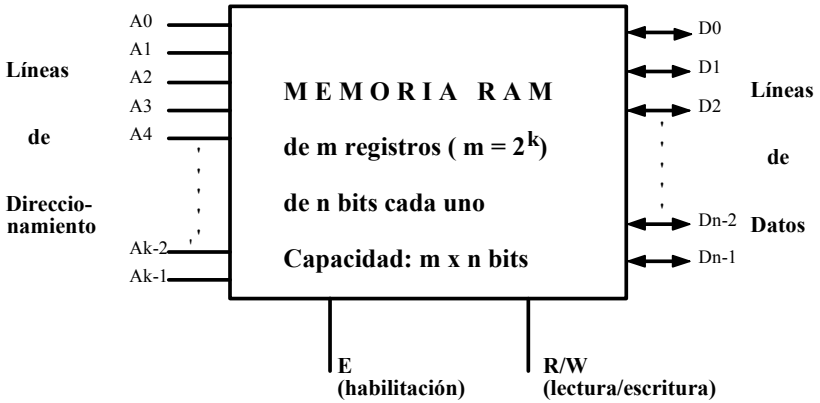
El biestable **D** constituye la célula básica de memoria capaz de almacenar y conserva un valor booleano, bajo control de la entrada de habilitación. Agrupando **n** de estos biestables en paralelo, con su entrada de habilitación común, se configura un registro de longitud **n**, capaz de almacenar una palabra de **n** dígitos o bits. Tal registro recibe el nombre de *memoria de retención* o «memoria cerrojo» (*latch-memory*), porque retiene la información almacenada cuando **E=1** durante todo el tiempo en que **E=0**, es decir, la conserva entre dos habilitaciones sucesivas.



Estos registros pueden ampliarse a cualquier número de bits, sin más que utilizar varios de ellos, uniendo en una sola las diversas entradas de habilitación **E**.

La agrupación de **m** registros de **n** bits, controlados a través de **k** entradas de direccionamiento (**$m=2^k$**), de forma que cada registro queda seleccionado por su número en binario, da lugar a un bloque de memoria **RAM**, *memoria de acceso directo* o aleatorio (*random access memory*).

Los terminales de entrada y de salida de los registros son comunes para todos ellos y, en cada momento, el vector presente en las entradas de control o direccionamiento indicará sobre cuál de los registros se actúa.



Además, por simplicidad de acceso, no hay distinción entre terminales de entrada y terminales de salida de los registros (es decir, la misma línea actúa como entrada y como salida), de forma que el tipo de acceso al registro seleccionado (lectura del registro o escritura del mismo) ha de ser indicado por una línea adicional **R/W** (*lectura/escritura*).

Un bloque **RAM** tendrá **k** líneas de direccionamiento **A_i**, que actúan como entradas, **n** líneas de datos **D_i**, que actúan bidireccionalmente, una entrada de selección de la operación a realizar **R/W**, que distingue entre lectura y escritura y una entrada de habilitación **E**. Su esquema conceptual está conformado por un sub-bloque central que contiene los **m** registros de **n** bits, un decodificador de **k** líneas de entrada que selecciona los registros (**$m = 2^k$**) y un circuito adaptador de entradas/salidas que controla la actuación de las **n** líneas de datos a tenor de las entradas de habilitación **E** y lectura/escritura **R/W**.

De esta forma, se dispone de unidades de memoria (biestables **D**) capaces de almacenar y conservar un bit, de registros (conjuntos de biestables **D**) capaces de almacenar una palabra binaria y de bloques de memoria (conjuntos de registros numerados) capaces de almacenar múltiples datos ordenados.

11.4. Estado, biestables y variables de salida; autómatas de Moore y de Mealy

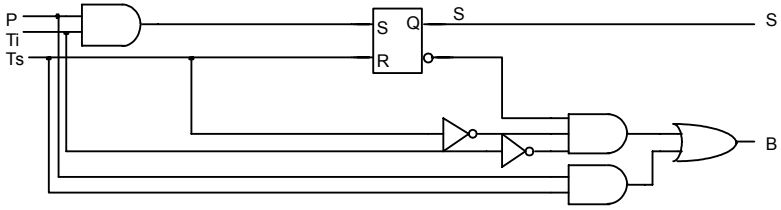
Las variables de estado de un sistema secuencial pueden configurarse sobre biestables RS.

Los biestables **RS** pueden ser utilizados para la configuración de sistemas secuenciales como unidades de memoria capaces de almacenar las variables de estado; será preciso emplear un biestable para cada una de las variables de estado y, en este caso, el resto del circuito será exclusivamente combinacional.

En el sistema de control de la puerta de garaje utilizando la variable de salida **S** como variable de estado, un biestable tipo **RS** servirá para «memorizar» dicha variable, cuyas condiciones de marcado y de borrado serán, respectivamente:

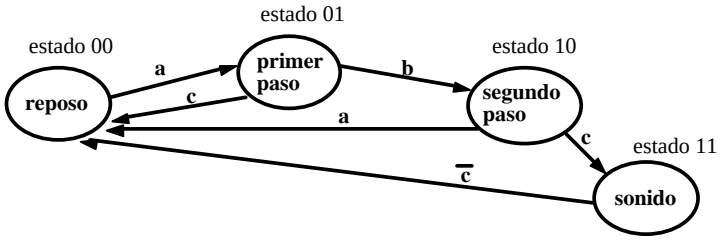
- la variable **S** (subir) se marcará cuando la puerta se encuentre abajo (**$T_i = 1$**) y se active el pulsador **P** y se borrará al llegar arriba (**$T_s = 1$**).

S (marcado) **$= P.T_i$** **R** (borrado) **$= T_s$**



En este esquema no queda reflejada la realimentación booleana, ya que la memoria del estado está configurada por el biestable.

En el caso del timbre con tres pulsadores **a b c** (con un mecanismo que impide pulsar dos a la vez y que debe sonar cuando se pulsan sucesivamente los tres pulsadores: **a**, **b** y **c**), del grafo de estado pueden extraerse las condiciones de marcado y borrado de cada una de las variables de estado:



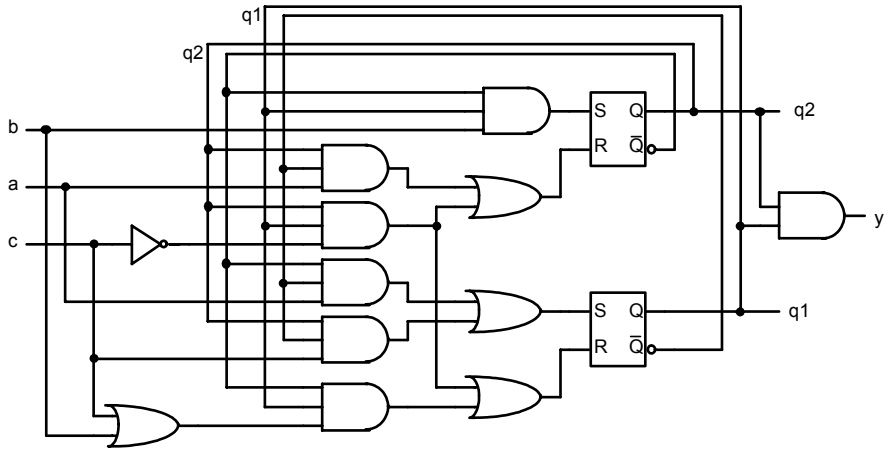
- dos variables de estado **q_2 q_1** sirven para «dar nombre» a los cuatro estados
- la variable de estado **q_1** se marca (pasa de **0** a **1**) desde el estado **00** con **a** y desde el estado **10** con **c** y se borra (pasa de **1** a **0**) desde el estado **01** con **b** y con **c** y desde el estado **11** con **c** negado
- la variable de estado **q_2** se marca (pasa de **0** a **1**) desde el estado **01** con **b** y se borra (pasa de **1** a **0**) desde el estado **10** con **a** y desde el estado **11** con **c** negado**.**

q1: $S = \overline{q_2}.\overline{q_1}.a + q_2.\overline{q_1}.c$ $\qquad R = \overline{q_2}.q_1.(b+c) + q_2.q_1.\bar{c}$

q2: $S = \overline{q_2}.q_1.b$ $\qquad R = q_2.\overline{q_1}.a + q_2.q_1.\bar{c}$

- variable de salida: **timbre = y = q_2 . q_1**

La siguiente figura representa el circuito correspondiente a estas funciones:



De esta forma, todo sistema secuencial puede dividirse en tres subsistemas: dos de ellos combinacionales y el tercero constituido por los biestables que almacenan las variables de estado.

Todo sistema secuencial puede ser construido mediante un conjunto de biestables que contengan las variables de estado y el resto del sistema será meramente combinacional:

a) las funciones que activan (marcado) y desactivan (borrado) a los biestables

b) y las funciones que producen las salidas del sistema.

Dichas funciones de marcado y borrado de los biestables que contienen a las variables de estado pueden obtenerse directamente de los arcos que representan las transiciones en el grafo de estados.

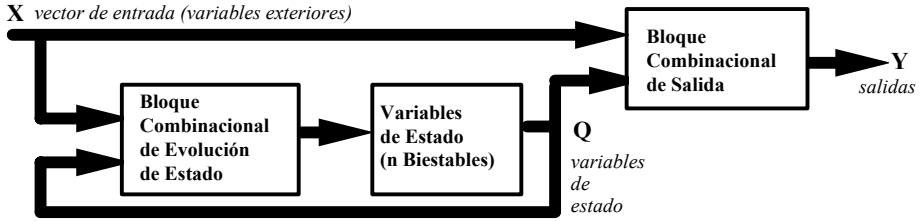
Conviene insistir en que el diseño de un circuito secuencial no se reduce a sus variables de estado, sino que requiere configurar, también, las variables de salida. De hecho, las variables de salida son «lo que se ve» desde fuera, el resultado efectivo de la actuación del circuito; las variables de estado son componentes internas instrumentales que el circuito necesita como memoria de su evolución anterior.

Autómata es el nombre que los matemáticos dan a los sistemas secuenciales, es la denominación de la estructura matemática que describe a dichos sistemas: una quíntupla [**X**, **Q**, **Y**, **f**, **g**] conformada por tres vectores: el vector de entrada, el vector de estado y el vector de salida, y dos funciones: la correspondencia que determina la evolución del estado y la correspondencia que determina la activación de las salidas.

Un sistema secuencial o autómata engloba tres conjuntos de variables o vectores booleanos, entrada **X**, salida **Y** y estado **Q**, y dos conjuntos de relaciones funcionales:

- evolución del estado $\quad \mathbf{X.Q} \longrightarrow \mathbf{Q}$
- activación de las salidas $\quad \mathbf{X.Q} \longrightarrow \mathbf{Y}$

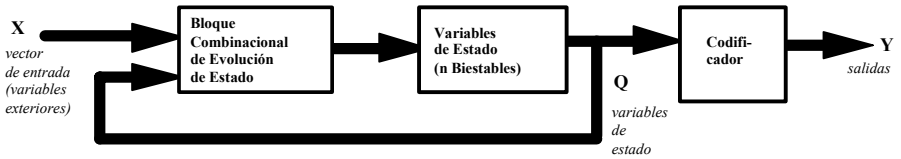
Este tipo de estructura, con separación funcional entre estado y salidas, de forma que a un mismo estado le pueden corresponder diversos vectores de salida (para vectores de entrada diferentes), recibe el nombre de *autómata de Mealy*.



Ahora bien, si nos fijamos en los grafos de estado desarrollados en este capítulo, resulta que en todos (menos en los de la puerta de garaje) el vector de salida es función únicamente del estado (no depende del vector de entrada).

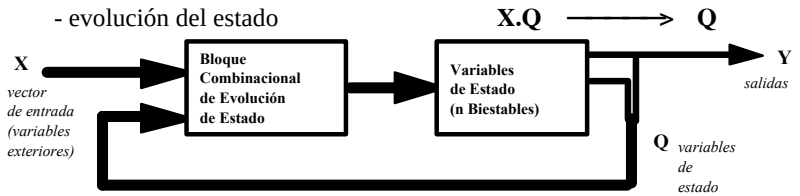
Ello da lugar a un tipo de estructura alternativa denominada *autómata de Moore*, en la cual la correspondencia entre estado y vector de salida es unívoca, de forma que a cada estado le corresponda un solo vector de salida. En este caso el estado ha de contener, además de la información sobre la evolución anterior del sistema, la relativa al vector de salida que debe producir en el momento actual; de forma que las relaciones funcionales en el autómata de Moore son las siguientes:

- evolución del estado $\quad \mathbf{X.Q} \longrightarrow \mathbf{Q}$
- activación de las salidas $\quad \mathbf{Q} \longrightarrow \mathbf{Y}$



En el *autómata de Moore* el vector de salida se encuentra contenido en el estado, es una simple recodificación del vector de estado. De manera que un estado del autómata de Mealy que produzca diversos vectores de salida tiene que desdoblarse en varios estados del autómata de Moore, uno para cada vector de salida (distinguibles estos estados entre sí por los vectores de entrada que, en el caso de Mealy, diferenciaban entre los vectores de salida posibles).

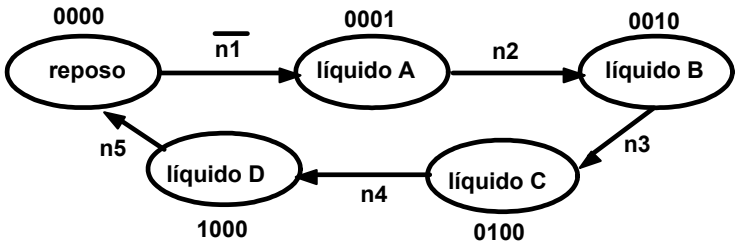
Entre estado y salida hay una simple codificación booleana; más aún, en muchos casos las variables de salida son una parte de las variables de estado: el vector de estado queda configurado por el vector de salida junto con algunas otras variables adicionales. Ello evita el codificador estado-salidas, reduciendo las relaciones funcionales a un solo conjunto:



Ambas estructuras de autómata (Moore y Mealy) son funcionalmente equivalentes, si bien el autómata de Moore puede contener un número de estados considerablemente mayor. Por ello, el autómata de Mealy suele resultar mucho más simple en cuanto a descripción de la evolución del estado y a síntesis de las funciones correspondientes a las variables de estado; en cambio, en el autómata de Moore resultan más sencillas las funciones de activación de las salidas.

Veamos un interesante ejemplo de sistema secuencial, expresando su grafo en las dos formas de Mealy y Moore: *Un depósito se llena con una mezcla de cuatro líquidos diferentes, para lo cual dispone de cuatro electroválvulas* ***A, B, C, D*** *que controlan la salida de dichos líquidos y de cinco detectores de nivel* n_1, n_2, n_3, n_4, n_5 *siendo* n_1 *el inferior y* n_5 *el de llenado máximo. Solamente cuando el nivel del depósito desciende por debajo del mínimo* n_1 *se produce un ciclo de llenado: primero con el líquido A hasta el nivel* n_2*, luego el líquido B hasta el nivel* n_3*, el líquido C hasta* n_4 *y, finalmente, D hasta completar el depósito* n_5*.*

El grafo de estado correspondiente al autómata de Moore será el siguiente:



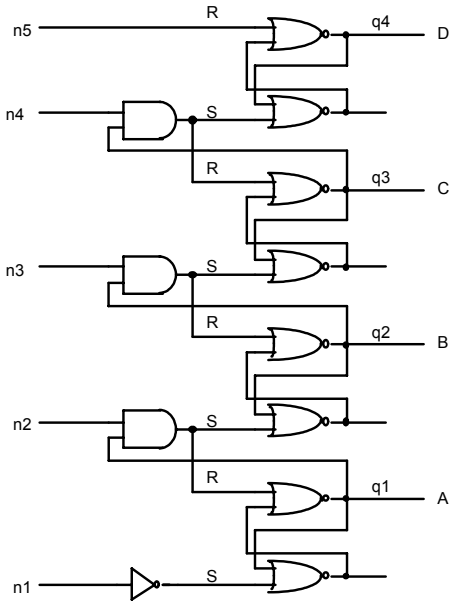
En el grafo anterior se utiliza para los estados el código de «un solo uno» con cuatro variables de estado $q_4\ q_3\ q_2\ q_1$; de esta forma cada variable de estado coincide con una de las variables de salida (q_1 = llenado con líquido **A**; q_2 = líquido **B**; q_3 = líquido **C**; q_4 = líquido **D**). El marcado y borrado de cada variable de estado coincide, en este grafo circular y codificado con «un solo uno», con las transiciones de entrada y de salida al estado en que la correspondiente variable vale **1**:

q_1:	$S = \overline{n_1}$	$R = q_1.n_2$
q_2:	$S = q_1.n_2$	$R = q_2.n_3$
q_3:	$S = q_2.n_3$	$R = q_3.n_4$
q_4:	$S = q_3.n_4$	$R = n_5$

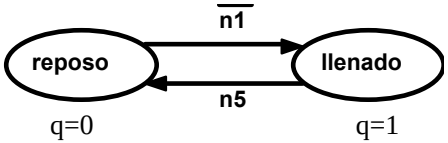
Obsérvese que para identificar cada estado solamente es necesaria una variable (ya que el código es de *un solo uno*).

S_1 no contiene el estado desde el que se marca $\overline{q_4}.\overline{q_3}.\overline{q_2}.\overline{q_1}$ ya que la transición con $n_1 = 0$ solamente puede darse desde dicho estado; lo mismo sucede con R_4, pues $n_5 = 1$ sólo puede suceder desde el estado correspondiente a q_4.

La siguiente figura representa el circuito resultante:



El mismo sistema secuencial puede configurarse con un número más reducido de estados, según el siguiente grafo que corresponde a un autómata de Mealy:



En este caso se necesita solamente una variable de estado **q**, pero las funciones de activación de las salidas resultan más complejas, pues dependen de las entradas, de la información que aportan los detectores de nivel:

electroválvula A = $q \cdot \overline{n_2}$ *líquido A hasta que se alcanza el nivel n_2*

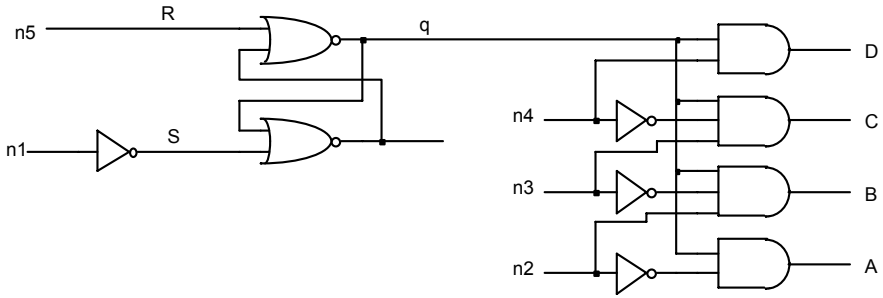
electroválvula B = $q \cdot n_2 \cdot \overline{n_3}$ *líquido B desde el nivel n_2 hasta el nivel n_3*

electroválvula C = $q \cdot n_3 \cdot \overline{n_4}$ *líquido C desde el nivel n_3 hasta el nivel n_4*

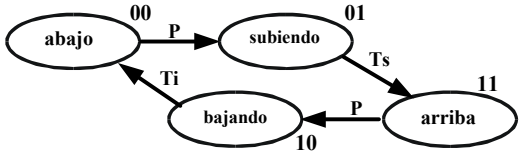
electroválvula D = $q \cdot n_4$ *líquido D por encima del nivel n_4*

y, según el sencillo grafo de estados anterior, la variable q pasa a valor **1** cuando $n_1 = 0$ y a valor **0** cuando $n_5 = 1$:

q: $S = \overline{n_1}$ $R = n_5$.



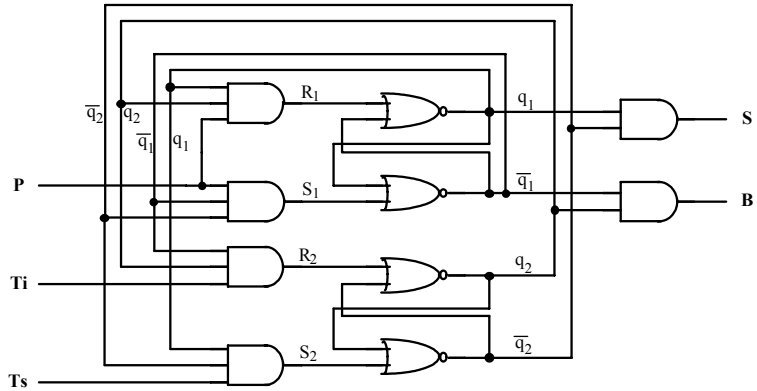
De igual forma, el primer ejemplo considerado en este capítulo, el control de la puerta de garaje, puede expresarse como autómata de Moore:



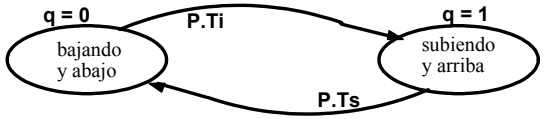
q_1: $S = P.\overline{q_2}.\overline{q_1}$ $R = P.q_2.q_1$

q_2: $S = Ts.\overline{q_2}.q_1$ $R = T_i.q_2.\overline{q_1}$

S (subir) = $\overline{q_2}.q_1$ B (bajar) = $q_2.\overline{q_1}$



y como autómata de Mealy:



q_1: $S = P.T_i$ $R = P.T_s$

S (subir) = $q \cdot \overline{T_s}$ B (bajar) = $\overline{q} \cdot \overline{T_i}$