UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Sistemas secuenciales síncronos: síntesis desde codificación *one-hot*

Apellidos, nombre	Martí Campoy, Antonio (amarti@disca.upv.es)
Departamento	Informàtica de Sistemes i Computadors
Centro	Escola Tècnica Superior d'Enginyeria Informàtica

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resumen de las ideas clave

En este artículo vamos a realizar la **síntesis** de un Sistema Secuencial Síncrono (SSS), siguiendo el modelo de autómata de Moore. Para que te quede un poco más claro, por síntesis entendemos el diseño de los circuitos digitales necesarios para construir físicamente el SSS. Es decir, estamos en el último paso del diseño, ya que sólo queda construir físicamente los circuitos que obtendremos durante el trabajo que desarrolles en este documento.

Para ilustrar este proceso y facilitar la adquisición de conocimientos y habilidades, utilizarás como ejemplo el diseño de un control de volumen sencillo, y partiremos de un diagrama de estados, de una codificación de los estados ***one-hot***, y de una tabla de estados previamente desarrollados. Diseñar e implementar un SSS es un proceso con varios pasos y múltiples opciones y variantes en cada paso. Abordar todos los pasos y opciones en un solo documento es casi imposible y sería poco útil.

Tu trabajo en este artículo es continuar con el diseño e implementación del SSS, diseñando los circuitos combinacionales y secuenciales necesarios para hacer realidad el SSS.

Para poder adquirir los conocimientos y habilidades presentadas en este artículo, debes tener los conocimientos previos presentados en la tabla 1.

Tabla 1. Conocimientos previos

Conocimientos previos
1. Funciones lógicas Booleanas.
2. Circuitos combinacionales y simplificación de funciones.
3. Circuitos secuenciales básicos (biestables).
4. Conocer teóricamente los pasos necesarios para diseñar un SSS.

2 Objetivos

Una vez acabes de leer este artículo docente y reproduzcas los ejemplos presentados, deberás ser capaz de **diseñar** los circuitos combinacionales y secuenciales necesarios para **construir** un Sistema Secuencial Síncrono siguiendo el modelo de autómata de Moore. De este modo serás capaz de **traducir** de una forma tabular (tabla de estados) o gráfica (diagrama de estados) a un **diseño** con componentes electrónicos digitales. Esta traducción o diseño recibe el nombre de **síntesis.**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

3 Introducción

Te recuerdo que el diseño de un Sistema Secuencial Síncrono se divide, de forma general, en los siguientes pasos: obtención de la interfaz, construcción del diagrama de estados, codificación de los estados, construcción de la tabla de estados, y finalmente, síntesis.

Y este último paso, la síntesis del sistema, es el que vamos a trabajar en este documento. Y la síntesis no es más que obtener los circuitos combinacionales (aquellos en los que la salida sólo depende de las entradas actuales) y los circuitos secuenciales (aquellos en los que las salidas dependen tanto de las entradas actuales como de las pasadas) necesarios para construir físicamente nuestro SSS.

Como ya te he dicho, realizaremos la síntesis a partir de una interfaz, un diagrama de estados, una codificación de los estados *one-hot* y una tabla de estados que nos dan, que alguien ha construido a partir de la descripción, en lenguaje natural, del comportamiento del sistema.

La Figura 1 muestra la interfaz del circuito. De la interfaz podemos extraer información muy interesante e importante: el sistema tiene tres entradas y dos salidas.

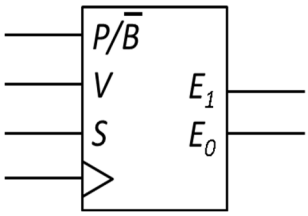

Figura 1. Interfaz del sistema

La Figura 2 muestra el diagrama de estados, una representación gráfica del comportamiento deseado para el circuito. En este diagrama de estados podemos ver que hay cuatro estados, y que las transiciones entre estados, o al mismo estado, están expresadas mediante expresiones algebraicas. Pero lo mejor de este diagrama de estados, y de cualquier diagrama de estados, es que ya no necesitas saber nada del comportamiento del circuito, ni del significado de las entradas y salidas. Al disponer del diagrama, porque tú mismo lo has construido o te lo han proporcionado como es el caso, no necesitas ninguna información en lenguaje "humano" sobre el circuito a construir.

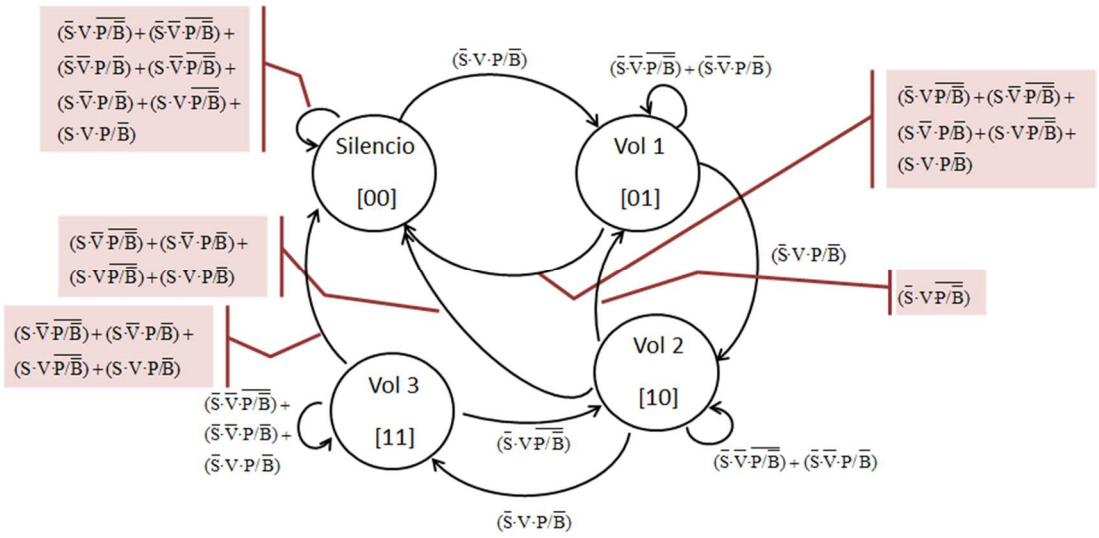

Figura 2. Diagrama de estados del control de volumen

La Tabla 2 muestra la codificación. En este caso se trata de una codificación *one-hot*. Como ya sabes, en esta codificación se usan tantos biestables como estados tiene el sistema, y para cada estado, una y sólo una de las variables de estado toma valor uno. Si bien es cierto que se necesitan más biestables que si usáramos codificación mínima, el proceso de síntesis resulta más sencillo, como verás más adelante.

Tabla 2. Codificación de los estados con codificación mínima con prioridad a la salida

Nombre simbólico	Código $Q_3Q_2Q_1Q_0$
Silencio	0001
Vol 1	0010
Vol 2	0100
Vol 3	1000

La Tabla 3 muestra la tabla de estados completa para el SSS que queremos construir. Aunque estoy seguro que lo sabes, te recuerdo que la primera columna corresponde con el estado actual, las columnas centrales corresponden con el estado siguiente en función de las entradas, y la última columna de la derecha indica el valor de las salidas para el estado actual, como buen autómata de Moore.

Tabla 3. Tabla de estados completa.

Estado actual Q(t) $Q_3Q_2Q_1Q_0$	Estado siguiente Q(t+1) Entradas: **S**, **V**, $\boldsymbol{P/\overline{B}}$								Salidas E_1E_0
	000	001	010	011	100	101	110	111	
0001	0001	0001	0001	0010	0001	0001	0001	0001	00
0010	0010	0010	0001	0100	0001	0001	0001	0001	01
0100	0100	0100	0010	1000	0001	0001	0001	0001	10
1000	1000	1000	0100	1000	0001	0001	0001	0001	11

4 Síntesis

Para construir el sistema necesitas diseñar u obtener tres circuitos tal como puedes ver en la Figura 3: el circuito que almacena el **estado actual (variables de estado)**, el circuito que implementa la **función de salida**, y el circuito que implementa la **función de excitación o transición**. Por cierto, ¿puedes decirme el color del circuito secuencial y los colores de los circuitos combinacionales? Efectivamente, el verde es secuencial y el rojo y azul son combinacionales.

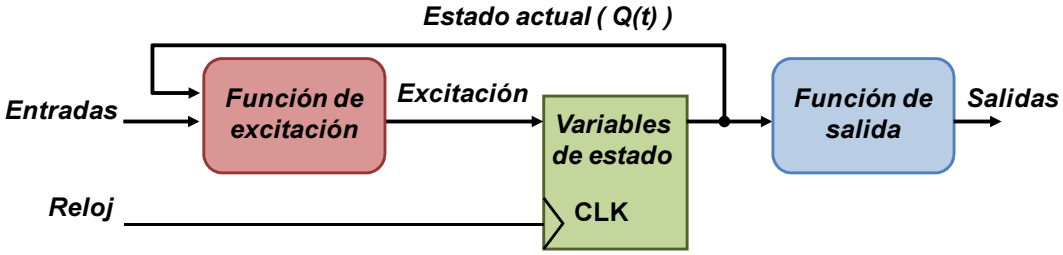

Figura 3. Diagrama genérico de un autómata de Moore.

El circuito que almacena el **estado actual** es el corazón secuencial del SSS. Es un conjunto de biestables que permiten almacenar el estado en que se encuentra el sistema en un determinado instante de tiempo.

La función de **salida** es la que produce las salidas del sistema. En un autómata de Moore las salidas dependen únicamente del estado actual.

La función de **transición o excitación** es la que indica cuál es el siguiente estado como función del estado actual y del valor actual de las entradas. Es decir, se encarga de modificar las variables de estado almacenadas en los biestables. Su complejidad es variable, ya que además de depender del número de estados y de entradas del sistema, también depende del tipo de biestables utilizados para almacenar el estado del sistema, y también se puede ver afectado por el tipo de codificación utilizada al pasar de los estados simbólicos a los estados codificados.

En los siguientes apartados se detalla el proceso para diseñar los tres circuitos enumerados anteriormente, y podrás ver que la codificación *one-hot* nos permite hacer una síntesis sin utilizar métodos de simplificación algebraica.

4.1 Almacenamiento del estado actual

Para almacenar el estado actual de un Sistema Secuencial Síncrono podemos utilizar cualquier biestable, con dos requisitos: todos los biestables son activos por el mismo flanco de reloj (subida o bajada), y a todos los biestables les llega la misma señal de reloj. ¿Y si no se cumple alguna de estas condiciones aunque sólo sea en un biestable? Entonces tendrás un Sistema Secuencial, pero no será Síncrono. Y su diseño queda fuera del alcance de este artículo.

Para diseñar esta parte de nuestro SSS hay que responder a dos preguntas. ¿Cuántos biestables necesitamos y de qué tipo?

La primera ya la han respondido por nosotros. Cuando se eligió el tipo de codificación también se decidió cuantas variables de estado hacían falta para codificar todos los estados del sistema. Y cada variable de estado necesita un biestable. Por tanto, mirando la tabla de estados vemos que se han utilizado cuatro

variables, Q_3, Q_2, Q_1 y Q_0 para almacenar los estados, lo que nos indica que necesitamos cuatro biestables.

La segunda pregunta tiene varias respuestas. Podríamos utilizar biestables tipo D, J-K o T. La utilización de biestables J-K o T puede proporcionar circuitos más sencillos, es decir, con menor número de puertas lógicas que si utilizamos biestables D. Pero el diseño de los circuitos puede hacerse más complicado. En este artículo no podemos cubrir todas las opciones de diseño, por lo que utilizaremos biestables tipo D. La Figura 4 muestra el circuito que almacena el estado actual Q(t) del SSS.

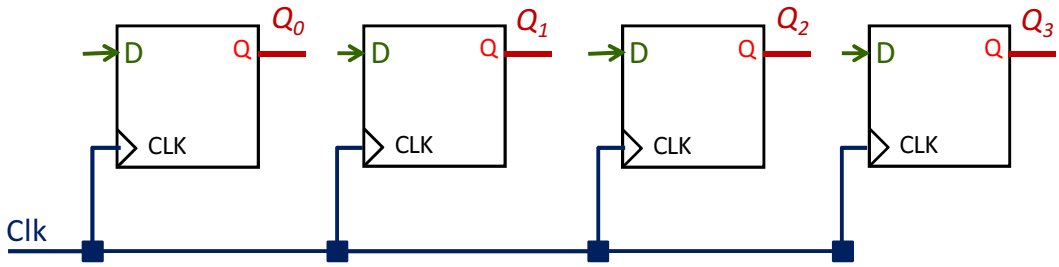

Figura 4. Circuito para almacenar el estado actual (variables de estado).

4.2 Función de salida

Como te he dicho antes, la función de salida es una función combinacional que genera las salidas del sistema en función del estado actual. Cuando hablamos de funciones lógicas (o circuitos combinacionales), la primera idea que nos debe venir a la cabeza es "tabla de verdad". Y es cierto, una función lógica se representa por su tabla de verdad. Por tanto, lo primero sería construir la tabla de verdad de la función de salida. En esta fabla las entradas son las variables de estado (Q_3, Q_2, Q_1 y Q_0) y las salidas son E_1 y E_0, que controlan el volumen del reproductor. Pero antes de construir la tabla, quiero hacerte unas preguntas. ¿Cuál es la aridad de la función de salida? Efectivamente, su aridad es 4, el número de entradas que tiene la función. Entonces, ¿Cuántas filas tendrá esta tabla de verdad? Correcto. El número de filas o valoraciones será de $2^4 = 16$. Sin embargo, de esas 16 combinaciones sólo 4 son significativas, ya que sólo existen 4 estados en el sistema.

Al utilizar codificación *one-hot* existen muchos más códigos binarios que estados tiene el sistema (exactamente hay 2^n códigos binarios y sólo n estados). Por ello es habitual no construir la tabla de verdad completa, sino una tabla de verdad reducida con sólo aquellas valoraciones que representan a estados del sistema. Puedes ver esta tabla reducida en la Tabla 4.

Tabla 4. Tabla de verdad reducida de la función de salida

Estado actual $Q_3Q_2Q_1Q_0$	Salidas E_1E_0
0001	00
0010	01
0100	10
1000	11

Si recuerdas el diseño de circuitos combinacionales, hay múltiples caminos para obtener el circuito a partir de la tabla de verdad: formas canónicas, simplificación por mapas de Karnaugh, y otros métodos que nos proporcionan una expresión algebraica fácilmente convertible en un circuito con puertas lógicas. Todos estos métodos necesitan la tabla de verdad completa (en este caso con sus 16 filas y un montón de X para indicar las entradas indiferentes[1]).

Sin embargo, y aquí radica la ventaja de la codificación *one-hot*, puedes ver que la salida E_0 se activa cuando estamos en el estado 0010 o cuando estamos en el estado 1000. Es decir, se activa si Q_1= 1 o Q_3= 1. Y esto se puede representar directa y sencillamente con la siguiente expresión algebraica: $E_0 = Q_1 + Q_3$

Lo mismos sucede con la salida E_1, pero en este caso puedes ver que se activa si el estado actual es 0100 o si es 1000, es decir, si Q_2= 1 o Q_3= 1. Esto se puede expresar algebraicamente por $E_0 = Q_2 + Q_3$

Una vez obtenidas las expresiones algebraicas de las salidas es fácil obtener el circuito digital que las implementa, que puedes ver en la Figura 5.

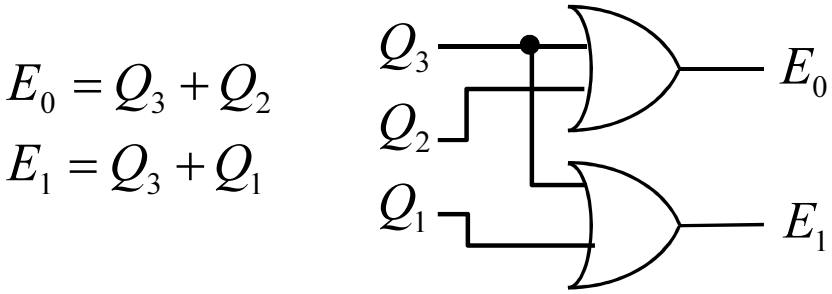

Figura 5. Expresiones algebraicas y circuito de la función salida.

4.3 Función de excitación o transición

Esta es una función combinacional que, partiendo del estado actual y de las entradas actuales, genera el siguiente estado. Como toda buena función combinacional la podemos representar por una tabla de verdad cuyas entradas serán las variables de estado y las entradas del sistema, y sus salidas serán las entradas de los biestables para poder establecer el nuevo estado. Ya hemos comentado que vamos a utilizar biestables D, por lo que las salidas de esta función serán las entradas D de dichos biestables, es decir, D_x. Y aunque estoy seguro que ya lo sabes, te recuerdo que si queremos que un biestable D almacene el estado 0, debemos poner un 0 en su entrada D, y si queremos que almacene el estado 1, debemos poner un 1 en su entrada D. Por tanto, las entradas de los biestables coincidirán con el estado siguiente que queremos alcanzar.

Vamos pues, con la tabla de verdad de la función de excitación. Permíteme, nuevamente, que te haga unas sencillas preguntas. ¿Cuál es la aridad de la función de excitación? Y por tanto, ¿Cuántas valoraciones o filas tendrá su tabla de verdad? Seguro que lo sabes. La aridad es la suma del número de variables de

[1] Te recuerdo que, aunque se llaman **entradas** indiferentes, se pone X en las **salidas** de una función lógica para aquellas combinaciones de las entradas que son imposibles o que no nos importa su valor de salida.

entrada (4 en este caso) y del número de entradas (3 en este caso), es decir, la aridad es 7 y por tanto hay 2^7 = 128 valoraciones, y la tabla de verdad tendrá la nada despreciable cantidad de 128 filas. Manejar tablas de verdad de este tamaño y obtener un circuito no es sencillo.

Y aquí vuelve a aparecer la ventaja de la codificación *one-hot*. Es posible obtener directamente desde la tabla de estados (o incluso desde el diagrama de estados si se cambian los nombres simbólicos por su codificación) de forma similar a como hemos obtenido las expresiones algebraicas de la función de salida.

Para obtener las expresiones algebraicas de la función de excitación vamos a buscar en la tabla de estado las combinaciones de estado actual y valor de las entradas que hacen que cada variable de estado tome valor 1. Siguiendo esta metodología es probable que no obtengamos el circuito más sencillo posible, pero obtendremos un circuito no muy complejo de forma rápida y sencilla. En la Tabla 5 tienes replicada la tabla de estados con diferentes colores para que puedas seguir el razonamiento siguiente mejor.

Tabla 5. Tabla de estados completa.

Estado actual Q(t) $Q_3Q_2Q_1Q_0$	Estado siguiente Q(t+1) Entradas: **S**, **V**, $\boldsymbol{P/\overline{B}}$								Salidas E_1E_0
	000	001	010	011	100	101	110	111	
0001	0001	0001	0001	0010	0001	0001	0001	0001	00
0010	0010	0010	0001	0100	0001	0001	0001	0001	01
0100	0100	0100	0010	1000	0001	0001	0001	0001	10
1000	1000	1000	0100	1000	0001	0001	0001	0001	11

Vamos a obtener la expresión algebraica para D_0. Para ello miramos en las columnas de estado siguiente cuál es el estado actual y el valor de las entradas que hacen que el estado siguiente sea 0001, es decir, que D_0 sea 1. Los he marcado en rojo para que lo veas mejor. D_0 es 1 si: el estado actual es 0001 y las entradas son 000, o son 001, o son 010, o son 100, o son 101, o son 110, o son 111, que es lo mismo que decir que el estado actual es 0001 y las entradas **NO** son 011; D_0 también es 1 si el estado actual es 0010 y las entradas son 010; y también o si la entrada S es 1 y da igual en qué estado este el sistema. Esto se puede expresar de forma algebraica por la siguiente expresión: $D_0 = (\boldsymbol{Q_0} \cdot (\overline{\boldsymbol{S} \cdot \boldsymbol{V} \cdot \boldsymbol{P/\overline{B}}})) + (\boldsymbol{Q_1} \cdot (\boldsymbol{\overline{S}} \cdot \boldsymbol{P} \cdot \boldsymbol{\overline{P/\overline{B}}})) + \boldsymbol{S}$

En azul he marcado los casos en que D_1 debe ser 1: si estamos en el estado 0001 y la entrada es 011; o si estamos en el estado 0010 y las entradas son 000 o 001; o si el estado actual es 0100 y las entradas son 010. Esto se puede expresar algebraicamente de la siguiente manera: $D_1 = (\boldsymbol{Q_0} \cdot (\boldsymbol{\overline{S}} \cdot \boldsymbol{V} \cdot \boldsymbol{P/\overline{B}})) + (\boldsymbol{Q_1} \cdot ((\boldsymbol{\overline{S}} \cdot \boldsymbol{\overline{V}} \cdot \overline{\boldsymbol{P/\overline{B}}})) + (\boldsymbol{\overline{S}} \cdot \boldsymbol{\overline{V}} \cdot \boldsymbol{P/\overline{B}}))) + (\boldsymbol{Q_2} \cdot (\boldsymbol{\overline{S}} \cdot \boldsymbol{V} \cdot \overline{\boldsymbol{P/\overline{B}}}))$

En verde puedes ver marcados los casos en que D_2 debe ser 1: si estamos en el estado 0010 y las entradas son 011; o si estamos en el estado 0100 y las entradas son 000 o 001; o si estamos en el estado 1000 y las entradas son 010. Esto se puede expresar de forma algebraica por la siguiente expresión: $D_2 = (\boldsymbol{Q_1} \cdot (\boldsymbol{\overline{S}} \cdot \boldsymbol{V} \cdot \boldsymbol{P/\overline{B}})) + (\boldsymbol{Q_2} \cdot ((\boldsymbol{\overline{S}} \cdot \boldsymbol{\overline{V}} \cdot \overline{\boldsymbol{P/\overline{B}}})) + (\boldsymbol{\overline{S}} \cdot \boldsymbol{\overline{V}} \cdot \boldsymbol{P/\overline{B}}))) + (\boldsymbol{Q_3} \cdot (\boldsymbol{\overline{S}} \cdot \boldsymbol{V} \cdot \overline{\boldsymbol{P/\overline{B}}}))$

Finalmente, en amarillo, tienes los casos en que D_3 debe ser 1: si estamos en el estado 0100 y las entradas son 011; o si estamos en el estado 1000 y las entradas son 000, o 001 o 011. Esto se puede expresar algebraicamente de la siguiente manera:
$D_3 = (\boldsymbol{Q_2} \cdot (\bar{\boldsymbol{S}} \cdot \boldsymbol{V} \cdot \boldsymbol{P}/\bar{\boldsymbol{B}})) + (\boldsymbol{Q_3} \cdot ((\bar{\boldsymbol{S}} \cdot \bar{\boldsymbol{V}} \cdot \overline{\boldsymbol{P}/\bar{\boldsymbol{B}}})) + (\bar{\boldsymbol{S}} \cdot \bar{\boldsymbol{V}} \cdot \boldsymbol{P}/\bar{\boldsymbol{B}}) + (\bar{\boldsymbol{S}} \cdot \boldsymbol{V} \cdot \boldsymbol{P}/\bar{\boldsymbol{B}})))$

Una vez tenemos las expresiones algebraicas sólo queda implementar los circuitos utilizando puertas lógicas. La Figura 6 muestra los cuatro circuitos, uno para cada salida D_x. Aunque en realidad se podría hacer en un solo circuito con siete entradas y cuatro salidas, y reutilizando puertas lógicas, he preferido mostrártelos separados para que puedas ver mejor el paso de las expresiones algebraicas a las puertas.

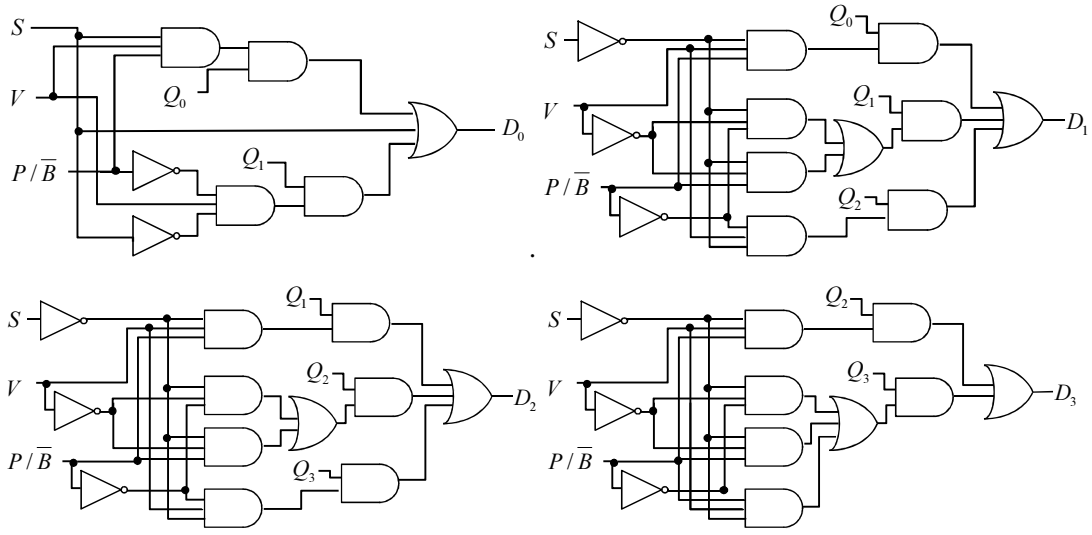

Figura 6. Circuito de la función de excitación.

4.4 Circuito final

Una vez tenemos los tres circuitos: almacenamiento del estado actual, función de salida, y función de excitación, sólo queda juntarlos. Esta interconexión es muy sencilla, y puedes verla en la Figura 7. Implementación completa del Sistema Secuencial Síncrono. (el circuito de excitación se presenta como un bloque para simplificar la figura).

5 Conclusiones

Una vez tenemos la interfaz, el diagrama de estados de un SSS (sistema secuencial síncrono), la codificación de estados y la tabla de estados codificados, el siguiente y último paso hacia la implementación del sistema es la síntesis o diseño de los circuitos digitales necesarios para construirlos.

Tres son los circuitos que necesitamos:

Los biestables para almacenar el estado actual. Al utilizar codificación *one-hot* son necesarios tantos biestables como variables de estado hayan. Puedes utilizar cualquier tipo de biestable, e incluso combinarlos, pero todos deben ser activos por el mismo flanco (bajada o subida) y conectados a la misma señal de reloj.

El segundo circuito es el circuito para implementar la función de salida. La función de salida es la función lógica que calcula el valor de las salidas del sistema para

cada uno de los estados (en los autómatas de Moore las salidas dependen sólo del estado actual, no como en los autómatas de Mealy donde las salidas dependen tanto del estado actual como de las entradas en cada momento). Normalmente, y para los autómatas de Moore, esta función y el circuito resultante suele ser más sencilla que la función de transición y su correspondiente circuito.

El tercer circuito implementa la función de transición o de excitación. La función de transición es la función lógica que calcula el estado siguiente a partir del estado actual y de las entradas del sistema. Cuando se utiliza codificación *one-hot* la aridad de esta función (suma del número de variables de estado y del número de entradas del sistema) suele ser muy elevada, por lo que no es habitual construir su tabla de verdad, sino que se hace un diseño que podríamos llamar "a mano alzada" en lugar de utilizar procedimientos sistemáticos de simplificación, tal como has visto en este artículo. Las salidas son las entradas de los biestables que almacenan el estado actual, por lo que tendrá una o dos salidas por cada biestable, dependiendo de si se utilizan biestables D, J-K o T.

Por último, me gustaría que prestaras atención a que un sistema secuencial síncrono es una combinación de un circuito secuencial (los biestables que almacenan el estado actual) y varios circuitos combinacionales que calculan el estado siguiente y las salidas del sistema.

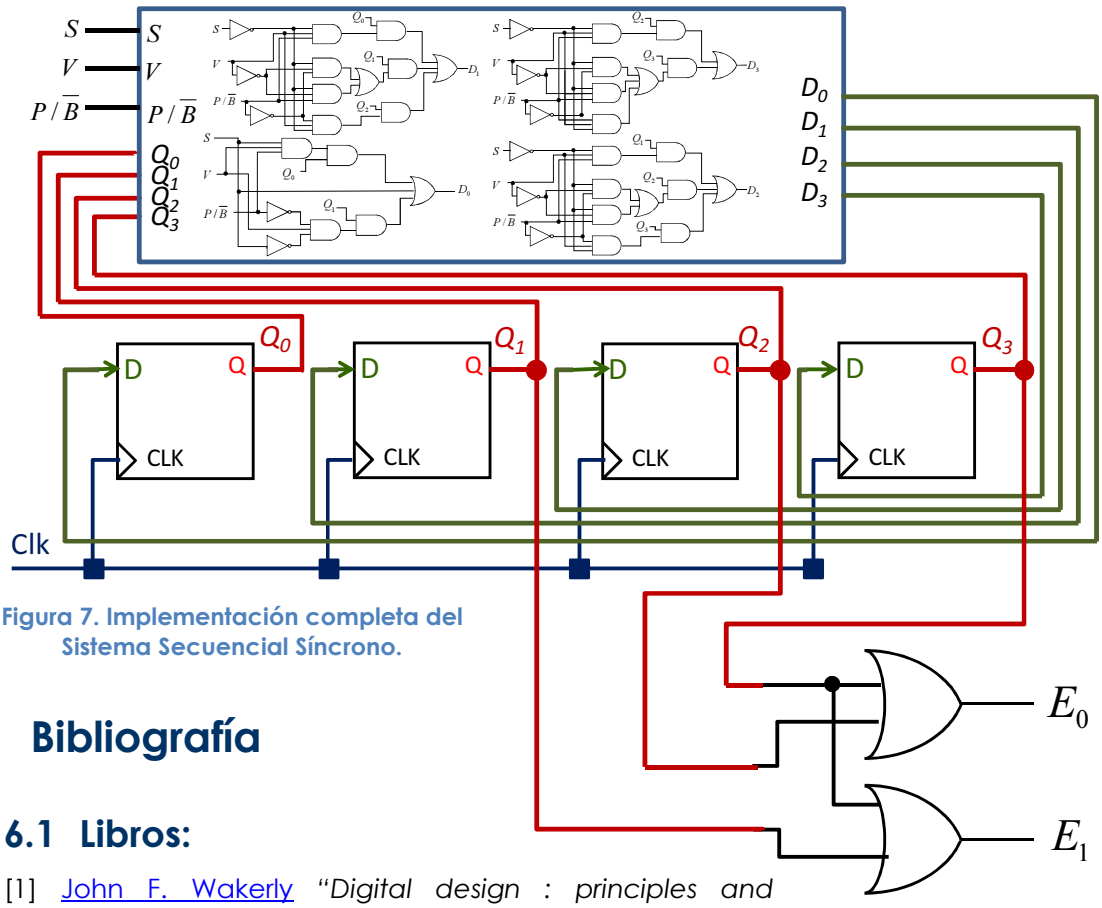

Figura 7. Implementación completa del Sistema Secuencial Síncrono.

6 Bibliografía

6.1 Libros:

[1] John F. Wakerly "*Digital design : principles and practices*", Prentice Hall. 2006

[2] Antonio Lloris Ruiz; Alberto Prieto Espinosa; Luis Parrilla Roure "*Sistemas digitales*", Aravaca, Madrid : McGraw-Hill/Interamericana de España. 2003