UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Sistemas secuenciales síncronos: síntesis desde codificación mínima.

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 la siguiente etapa es fabricar o construir físicamente los circuitos que obtendremos al final de este documento.

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. Por eso, en este documento, para ilustrar el proceso de síntesis y facilitar la adquisición de conocimientos y habilidades, utilizarás como ejemplo un sistema secuencial síncrono sencillo (autómata de Moore), del que se nos proporciona ya realizada la **tabla de estados** con los estados codificados, siguiendo una **codificación mínima**.

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 **S**istema **S**ecuencial **S**íncrono siguiendo el modelo de autómata de Moore. De este modo serás capaz de **traducir** de una forma tabular (tabla de estados) a un **diseño** con componentes electrónicos digitales. Esta traducción o diseño recibe el nombre de **síntesis.**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

3 Introducción

Como ya te he dicho, realizaremos la síntesis a partir de una tabla de estados codificados que se nos proporciona, que alguien ha construido a partir de la descripción, en lenguaje natural, del comportamiento del sistema que queremos construir.

Aunque no es necesario para realizar la síntesis, la Figura 1 muestra la interfaz del circuito, ya que de ella podemos extraer información muy interesante, importante y que nos permitirá obtener una visión global del sistema: el sistema tiene dos entradas y tres salidas, y es activo por flanco de subida.

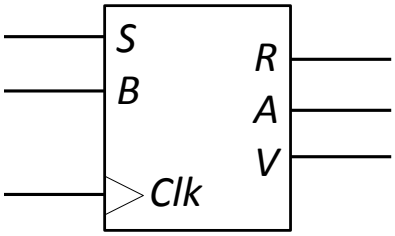

Figura 1. Interfaz del sistema

Una vez especificada la interfaz del SSS y partiendo de las especificaciones y requerimientos del sistema, el diseñador habrá construido el diagrama de estados con nombres simbólicos, habrá codificado los estados con alguna de las diferentes opciones posibles, y, cómo penúltimo paso, ha construido la tabla de estados con estados codificados y, gentilmente, nos la ha dado. La Tabla 2 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 en función del estado actual, como buen autómata de Moore que es.

Tabla 2. Tabla de estados (estados codificados).

Estado actual $Q(t)$ Q_1Q_0	Estado siguiente $Q(t+1)$ (Q_1Q_0)				Salidas
	Entradas: **S, B**				
	00	01	10	11	R A V
00	11	00	11	01	0 0 0
01	10	00	11	01	0 0 1
10	11	00	11	01	0 1 1
11	10	00	11	11	1 0 1

El siguiente y último paso es la **síntesis**, es decir, obtener un circuito digital que se comporte como se quiere y se expone en la tabla de estados. Esta síntesis es lo que vas a trabajar a continuación.

4 Síntesis

Para construir el sistema necesitas diseñar u obtener tres circuitos tal como puedes ver en la Figura 2: 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.

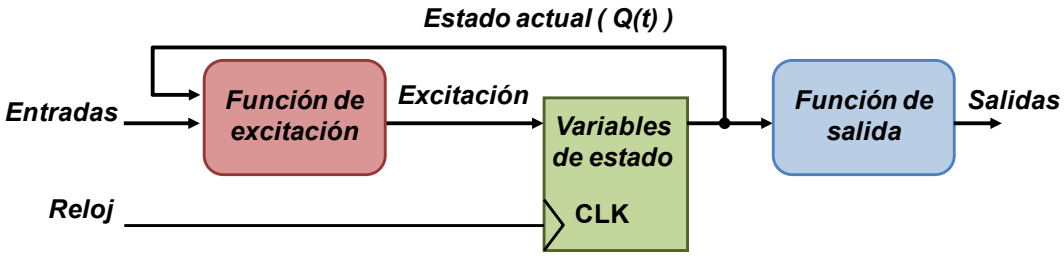

Figura 2. Diagrama genérico de un autómata de Moore.

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 momento.

La función de **salida** es la que produce las salidas del sistema. En un autómata de Moore, las salidas dependen únicamente del estado actual, por lo que esta función, y por tanto su circuito, tiene como entradas sólo el estado actual.

La función de **transición o excitación** es la que indica cuál es el siguiente estado en función del estado actual y del valor actual de las entradas. 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.

4.1 Almacenamiento del estado actual

Para almacenar el estado actual de un Sistema Secuencial Síncrono podemos utilizar cualquier biestable, cumpliendo sólo 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 (mínima en este caso) 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 dos variables, Q_1 y Q_0, para almacenar los estados, lo que nos indica que necesitamos dos biestables.

La segunda pregunta tiene varias respuestas. Podríamos utilizar biestables tipo D, JK o T. La utilización de biestables JK o T puede proporcionar circuitos más sencillos, es decir, con menor número de puertas que si utilizamos biestables D. Pero la tabla de verdad de la función de excitación puede complicarse en algunos casos al utilizar biestables JK o T. En este artículo no podemos cubrir todas las opciones de diseño, por lo que utilizaremos biestables tipo D. La Figura 3 muestra el circuito utilizado para almacenar el estado actual Q(t) del SSS.

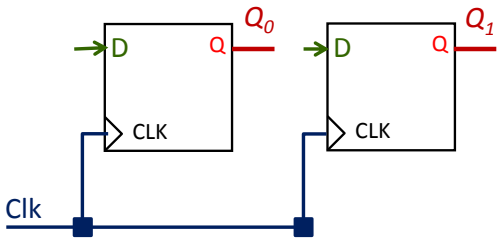

Figura 3. Circuito para almacenar el estado actual (variables de estado).

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. Y como debes saber, una función lógica se representa por su tabla de verdad. Desde la tabla de estados codificados podemos construir la tabla de verdad de esta función de salida, cuyas entradas son las variables de estado Q_1 y Q_0 y sus salidas R, A y V son las salidas del sistema que estamos diseñando. Para construir esta tabla de verdad sólo hemos tenido que copiar la primera y la última columna de la tabla de estados, pero con mucho cuidado de no saltarnos ninguna fila. Puedes ver esta tabla de verdad en la Tabla 3.

Tabla 3. Tabla de verdad de la función de salida

Estado actual Q_1Q_0	Salidas R A V
00	0 0 0
01	0 0 1
10	0 1 1
11	1 0 1

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 también otros métodos que nos proporcionan una expresión algebraica fácilmente convertible en un circuito con puertas lógicas. Utilizando alguno de los métodos existentes, se obtienen las expresiones algebraicas, y de ellas, fácilmente un circuito con puertas. La Figura 4 muestra las expresiones algebraicas mínimas y el circuito que implementa la función de salida.

$$R = Q_1 \cdot Q_0$$
$$A = Q_1 \cdot \overline{Q_0}$$
$$V = Q_1 + Q_0$$

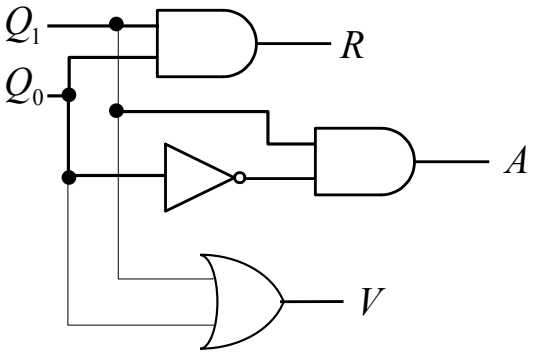

Figura 4, Expresiones algebraicas y circuito de la función de salida.

4.3 Función de excitación o transición

Esta es una función combinacional que, partiendo del estado actual y de las entradas actuales, genera el estado siguiente. Como toda buena función combinacional, la podemos representar por una tabla de verdad donde las entradas serán las variables de estado y las entradas del sistema, y las 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.

Tabla 4. Tabla de verdad de la función de excitación.

Entradas	Salidas
Q_1 Q_0 **S B**	D_1D_0
0000	11
0001	00
0010	11
0011	01
0100	10
0101	00
0110	11
0111	01
1000	11
1001	00
1010	11
1011	01
1100	10
1101	00
1110	11
1111	11

La tabla de verdad de la función de excitación no se construye de forma tan sencilla como la de la función de salida. En esta caso hay que combinar todos los posibles estados con todas las posibles valoraciones (posibles combinaciones) de las entradas. Esto nos da un total de 16 combinaciones o valoraciones[1]. La Tabla 4 muestra esta tabla de verdad, donde las entradas son el estado actual ($Q(t) = Q_1Q_0$) y las entradas del sistema (**S, B**). Las salidas de la función de excitación son las entradas a los biestables (D_1D_0) que almacenan el estado actual, para conseguir el estado siguiente deseado.

En este caso podemos utilizar los mapas de Karnaugh para obtener las expresiones algebraicas mínimas, y el circuito resultante, que se muestran en la Figura 5.

[1] Fíjate que la aridad de la función de excitación es 4, ya que tenemos como entradas las dos variables de estado y las dos entradas S y B. Y si la aridad es 4, el número de filas es $2^4 = 16$.

Pero estarás de acuerdo conmigo que en un sistema más complejo, con mayor número de estados o de entradas, o ambas cosas, esta tabla se volvería inmensa e imposible de simplificar a mano. En los sistemas reales se utilizan aplicaciones informáticas que, a partir de la tabla de verdad, simplifican e incluso generan el circuito final, como por ejemplo [3].

$$D_0 = \left(\overline{Q_0} \cdot \overline{B}\right) + S$$
$$D_1 = \left(Q_1 \cdot Q_0 \cdot S\right) + \overline{B}$$

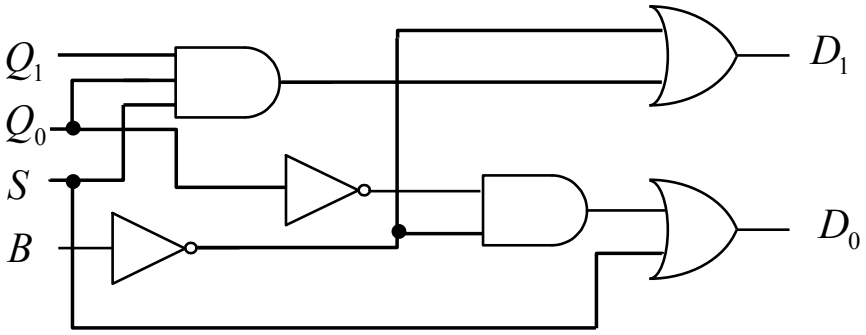

Figura 5. Expresiones algebraicas y circuito de la función de excitación.

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 6.

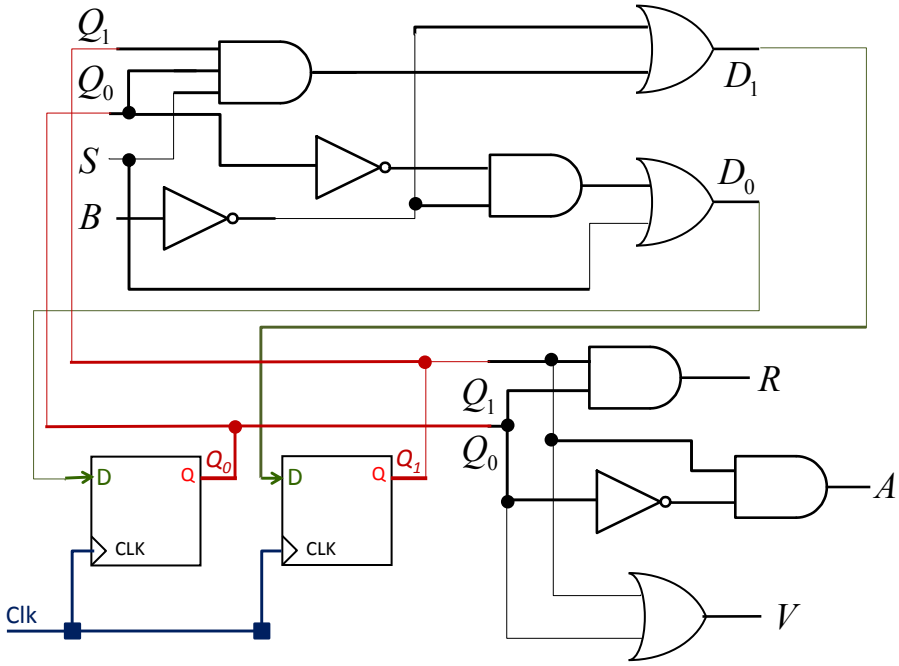

Figura 6. Implementación completa del Sistema Secuencial Síncrono.

4.5 Estados no definidos y transiciones imposibles

En algunos casos la codificación permite que hayan más códigos de estado que estados reales tiene el sistema. En otros casos hay transiciones que no existen, bien porque representan situaciones imposibles (por ejemplo, un pulsador no puede estar al mismo tiempo pulsado y sin pulsar) o simplemente porque nos da igual el comportamiento del circuito para una determinada combinación de estado y entradas. En este caso, y con el objetivo de obtener los circuitos de salida y de excitación más sencillos posibles, en la tabla de verdad de la funciones de salida y de excitación se indican estas situaciones como entradas indiferentes, es decir, poniendo en la salida X.

En la Tabla 5 puedes ver un ejemplo de tabla de estados de un sistema con tres estados, codificación mínima, una salida, y una transición imposible o indiferente.

En la Tabla 6 y en la Tabla 7 puedes ver la tabla de verdad de la función de transición y de la función de salida con las X generadas por el estado inexistente y la transición indiferente. Como ya sabes, estas X hacen que el circuito resultante sea más sencillo.

Tabla 5. Tabla de estados codificados con estados y transiciones no definidas.

Estado actual Q(t) Q_1Q_0	Estado siguiente Q(t+1) Entrada: **P**		Salida S
	0	1	
00	01	10	0
01	--	10	0
10	10	00	1

Esta transición es imposible o indiferente, por lo que no tiene estado siguiente.

No existe el estado 11 en el sistema (no aparece en la tabla de estados) por lo que en las tablas de verdad aparecerá como X

Tabla 6. Tabla de verdad de la función de excitación

Entradas Q_1Q_0 P	Salidas D_1D_0
000	01
001	10
010	**XX**
011	10
100	10
101	00
110	**XX**
111	**XX**

Tabla 7. Tabla de verdad de la función de salida

Entradas Q_1Q_0	Salida S
00	0
00	0
01	1
11	**X**

5 Conclusiones

Una vez tenemos la interfaz, el diagrama de estados de un SSS, la codificación de estados, y la tabla de estados codificados, bien porque lo hemos diseñado nosotros mismos o porque nos lo proporcionan, 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. Son necesarios tantos biestables como variables de estado hayan. Puede utilizarse cualquier tipo de biestable, e incluso combinarlos. En función del biestable utilizado es posible que el sistema sea más sencillo de diseñar o que el circuito final sea más simple. En cualquier caso, y dado que se trata de un sistema secuencial **síncrono** todos los biestables deben ser activos por el mismo flanco (bajada o subida) y conectados a la misma señal de reloj.

El segundo circuito es 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.

El tercer circuito es para implementar la función de excitación. La función de excitación es la función lógica que calcula el estado siguiente a partir del estado actual y las entradas del sistema (esto aplica tanto a los autómatas de Moore y de Mealy). La tabla de verdad de esta función tiene como aridad la suma del número de variables de estado (es decir, cuantos biestables se utilizan para almacenar el estado actual) y el número de entradas del sistema. 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 uno o varios circuitos combinacionales que calculan el estado siguiente y las salidas del sistema.

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

6.2 Aplicaciones:

[3] Logic Friday. http://www.sontrak.com/. Accedida 21 de marzo de 2016