UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Generación de funciones lógicas mediante decodificadores binarios con salidas activas a nivel bajo

Apellidos, nombre	Martí Campoy, Antonio (amarti@disca.upv.es)
Departamento	Informática de Sistemas y Computadores
Centro	Universidad Politécnica de Valencia

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resumen de las ideas clave

En este artículo se va a presentar la utilización de decodificadores binarios con salidas activas a nivel bajo para la generación de funciones lógicas. Son muchas las formas de diseñar una función lógica, y una de las más sencillas es la utilización del bloque combinacional conocido como decodificador binario. Para poder adquirir los conocimientos y habilidades presentadas en este artículo debes contar con unos conocimientos previos, listados en la tabla 1. Pero tranquilo, durante el texto se incluirán breves descripciones de estas ideas previas.

Tabla 1 Conocimientos previos

Conocimientos previos
1. Qué es una función lógica y su aridad
2. Tipos y tablas de verdad de puertas lógicas
3. Formas de representar una función lógica: tabla de verdad, formas canónicas y expresiones algebraicas
4. Funcionamiento de un decodificador binario
5. Circuito interno de un decodificador binario

2 Objetivos

Una vez hayas leído este artículo docente y reproducido los ejemplos presentados, deberás ser capaz de **implementar** una función lógica mediante el uso de decodificadores binarios con salidas activas a nivel **bajo**.

Además, la implementación de la función lógica podrá tomar como punto de partida diferentes representaciones de la misma, como la tabla de verdad o una forma canónica, por lo que serás capaz de **traducir** desde una representación a otra.

Por último, y atendiendo a criterios de simplificación de circuitos, serás capaz de **escoger** el tipo de puerta lógica más adecuada.

3 Introducción

En primer lugar, una breve descripción de los conceptos previos más importantes para poder alcanzar los objetivos propuestos en este artículo. Estas descripciones pueden ampliarse consultando la bibliografía propuesta al final del documento.

- Función lógica: expresión formal del comportamiento de un circuito digital. La aridad de una función lógica es el número de variables de entrada.

- Tabla de verdad: representación única en forma de tabla de una función lógica.
- Formas canónicas: representación única como suma de productos o como producto de sumas de una función lógica.
- Expresión algebraica: combinación de variables y operadores lógicos para expresar una función lógica.
- Puerta lógica: circuito digital que implementa una función lógica básica.
- Circuito o función combinacional: circuito en el que las salidas en un instante de tiempo dependen exclusivamente de las entradas en ese mismo instante de tiempo.
- Decodificador binario: circuito combinacional, con m entradas binarias y $n=2^m$ salidas binarias. Las salidas se activan de forma exclusiva, es decir sólo se activa una de ellas en un instante dado.
- La función realizada por un decodificador binario consiste en activar la salida de orden i que corresponde con la codificación binaria de sus entradas. La Figura 1 presenta el símbolo lógico de un decodificador binario de m a $n=2^m$ con salidas activas a nivel **bajo**.
- Una salida activa a nivel bajo quiere decir que tomará valor cero cuando esté activada, y valor uno cuando no esté activada.

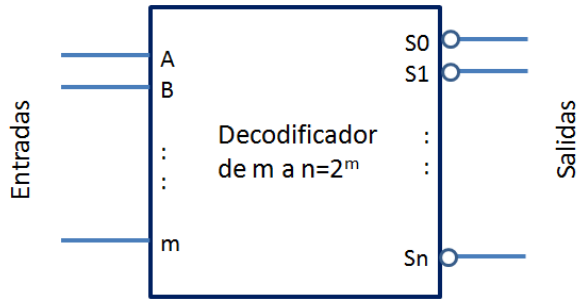

Figura 1 Símbolo lógico de un decodificador de m a n con salidas activas a nivel bajo.

Para ver si el funcionamiento del decodificador está claro, vamos a hacernos un par de preguntas. Suponemos un decodificador binario de 2 a 4 con las salidas activas a nivel bajo. Las entradas se llaman B y A, siendo A la de menor peso, y las salidas reciben el nombre de S0, S1, S2 y S3. Si los valores de las entradas son B=1 y A=0, el valor de las salidas S0, S1, S2 y S3 será:

Por favor, piensa la respuesta antes de ver la solución[1]

Hagamos otro intento. Si los valores de las entradas son B=0 y A=1, el valor de las salidas S0, S1, S2 y S3 será:

Por favor, piensa la respuesta antes de ver la solución[2]

[1] El valor de las salidas será S0=1, S1=1, S2=0, S3=1, ya que BA=10 se corresponde con el valor binario 2, por lo que se activa la salida S2 con valor 0 (bajo) y el resto de las salidas tomarán valor 1.

4 Generación de funciones

En este apartado veremos, en primer lugar, el significado algebraico de las salidas del decodificador binario con salidas activas a nivel bajo. En segundo lugar recordaremos brevemente que una función lógica puede crearse a partir de la forma canónica conjuntiva conocida como el producto sus maxitérminos. Por último, uniendo las dos ideas previas, usaremos puertas lógicas para generar una función usando decodificadores binarios con salidas activas a nivel bajo.

4.1 Significado de las salidas del decodificador binario

La implementación interna de un decodificador binario con salidas activas a nivel bajo es muy sencilla. Para cada una de las salidas se realiza un circuito que activa dicha salida (poniendo un cero) si las entradas toman el valor correspondiente. Por ejemplo, para un decodificador de 2 a 4, las salidas se corresponden con las expresiones algebraicas mostradas en la Ecuación 1, que coinciden además, con las expresiones de los maxitérminos[3].

Ahora que conocemos que las salidas de un decodificador binario corresponden con la implementación de cada uno de los maxitérminos, podemos incluir esta información en el símbolo lógico, que se muestra en la Figura 2. Esto es importante para, posteriormente, comprender como generar una función usando decodificadores binarios con salidas activas a nivel bajo.

Ecuación 1 Expresiones algebraicas para las salidas de un decodificador binario de 2 a 4 con salidas activas a nivel bajo

$$S0 = B + A = \prod_{B,A}(0)$$

S0 tomará valor 0 si B = A = 0

$$S1 = B + \bar{A} = \prod_{B,A}(1)$$

S1 tomará valor 0 si B = 0 y A = 1

$$S2 = \bar{B} + A = \prod_{B,A}(2)$$

S2 tomará valor 0 si B = 1 y A = 0

$$S3 = \bar{B} + \bar{A} = \prod_{B,A}(3)$$

S3 tomará valor 0 si B = 1 y A = 1

4.2 Forma Canónica Conjuntiva

Una forma sencilla de construir una función lógica es a partir de la Forma Canónica Conjuntiva, también conocida como producto de sumas o producto de los maxitérminos de la función.

---

[2] El valor de las salidas será S0=1, S1=0, S2=1, S3=1, ya que BA=01 se corresponde con el valor binario 1, por lo que se activa la salida S1 con valor 0 (bajo)

[3] Un maxitérmino es la suma de todas las variables de entrada, que aparecen en forma directa si su valor es 0 y en forma negada si su valor es 1

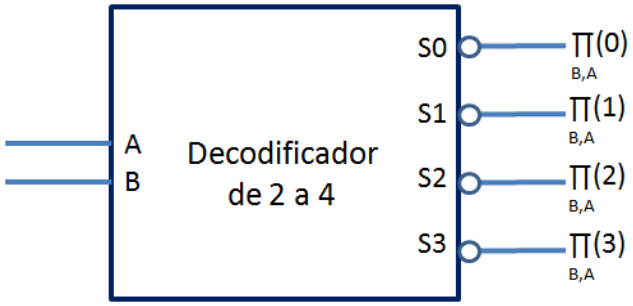

Figura 2 Decodificador 2 a 4 con identificación de los minitérminos

¿Cuáles son los maxitérminos de una función? Son aquellos para los que la función toma valor 0. Y la forma Canónica Conjuntiva dice que una función es el producto de sus maxitérminos. Pero mejor veamos un ejemplo. La Tabla 2 muestra una función de nombre H y aridad 3, y la Ecuación 2 muestra la Forma Canónica Conjuntiva de esta función.

Tabla 2 Tabla de verdad de la función H

Maxitérmino	C B A	H
0	0 0 0	0
1	0 0 1	1
2	0 1 0	0
3	0 1 1	0
4	1 0 0	1
5	1 0 1	0
6	1 1 0	1
7	1 1 1	0

Ecuación 2 Forma Canónica Conjuntiva para la función H

$$H = \prod_{C,B,A}(0,2,3,5,7)$$

Para construir el circuito que implementa la función H se puede utilizar puertas lógicas, implementado los maxitérminos de la función con puertas OR y usando una puerta AND para realizar el producto lógico de los maxitérminos. En total, contando las puertas NOT necesarias para construir los maxitérminos, necesitamos _____[4] puertas.

4.3 Generación de funciones con decodificadores

Tal como hemos visto antes, podemos crear una función siguiendo los pasos mostrados en la Figura 3.

El último paso corresponde a la implementación de los maxitérminos de la función mediante puertas NOT y puertas OR, y al producto usando una puerta AND. Pero, como hemos visto anteriormente, un decodificador binario con salidas activas a

[4] Se necesitan una puerta AND de 5 entradas, 5 puertas OR de 3 entradas, y 3 puertas NOT. En total, 9 puertas.

nivel bajo implementa, en cada una de sus salidas, un maxitérmino, por lo que la primera parte de la construcción del circuito puede ser sustituida por un decodificador binario, con tantas entradas como variables de entrada tenga la función lógica que se desea implementar.

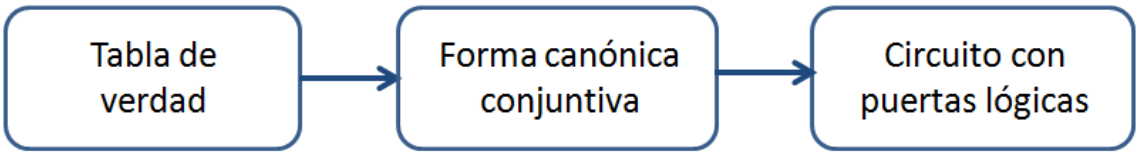

Figura 3 Pasos para la implementación de una función lógica

4.3.1 Implementación con una puerta AND

Podemos aplicar la propiedad asociativa para el producto lógico, mostrada en la Ecuación 3, al producto de maxitérminos de la función de ejemplo H, mostrada en la Ecuación 4.

Ecuación 3 Propiedad asociativa para la suma

$$(a \cdot b \cdot \cdots \cdot n) = a \cdot b \cdot \cdots \cdot n$$

Ecuación 4 Aplicación de la propiedad asociativa a la forma canónica disyuntiva de la función G del ejemplo de la Tabla 1

$$H = \prod_{C,B,A}(0,2,3,5,7) = \prod_{C,B,A}(0) \cdot \prod_{C,B,A}(2) \cdot \prod_{C,B,A}(3) \cdot \prod_{C,B,A}(5) \cdot \prod_{C,B,A}(7)$$

Bien, lo que nos dice la propiedad asociativa en este caso es que podemos coger los maxitérminos de una función, es decir, las salidas correspondientes del decodificador, y realizar el producto lógico mediante una puerta AND. Y la salida de esta puerta AND corresponde con la implementación de la función. La Figura 4 muestra la implementación de la función H utilizando un decodificador de 3 a 8 con salidas activas a nivel bajo y una puerta AND.

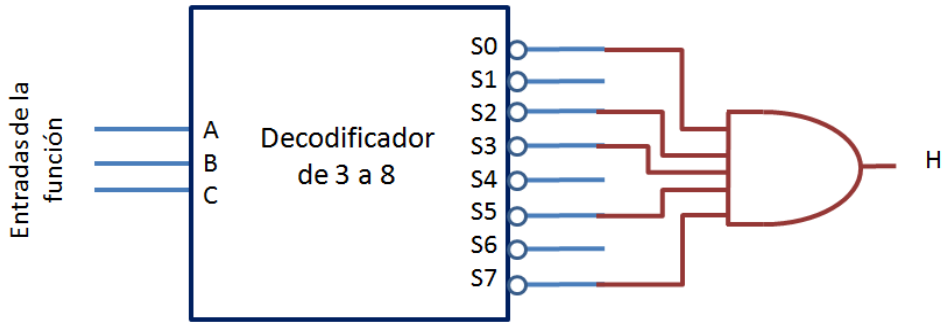

Figura 4 Implementación de la función G mediante un decodificador binario y una puerta AND

De este modo, para implementar una función lógica tan sólo necesito un decodificador binario con tantas entradas como tenga la función y una puerta AND.

4.3.2 Implementación con una puerta NAND

Pero, ¿qué pasa si no puedo conseguir la puerta AND que necesito?

Hagamos otra pregunta. ¿Qué sucede si en lugar de coger los maxitérminos que **SÍ** son de la función, cogemos los que **NO** son de la función?

Esto sería lo mismo que cambiar los ceros por unos y los unos por ceros en la salida de la función. Por ejemplo, con la función H, tendríamos la tabla de verdad mostrada en la Tabla 3, con la función $\overline{H}$, la negación de la función H. La Ecuación 5 muestra la forma canónica disyuntiva para $\overline{H}$,

Tabla 3 Tabla de verdad de la función H y $\overline{H}$

Minitérmino	C B A	H	$\overline{H}$
0	0 0 0	0	1
1	0 0 1	1	0
2	0 1 0	0	1
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	0	1
6	1 1 0	1	0
7	1 1 1	0	1

Ecuación 2 Forma Canónica Conjuntiva para la función H

$$H = \prod_{C,B,A}(0,2,3,5,7)$$

Ecuación 5 Forma Canónica Disyuntiva para la función $\overline{H}$

$$\overline{H} = \prod_{C,B,A}(1,4,6)$$

Pero nosotros no queremos implementar la función $\overline{H}$, sino que queremos implementar la función H. Y esto lo podemos conseguir cogiendo los maxitérminos que **NO** son de la función y utilizar, en lugar de una puerta **AND**, una puerta **NAND**. De esta forma, aprovechamos la propiedad llamada involución[5] y al negar $\overline{H}$, obtenemos H, que es lo que queríamos.

La Figura 5 muestra la implementación de la función H mediante el uso de un decodificador de 3 a 8 con salidas activas a nivel alto y una puerta NAND.

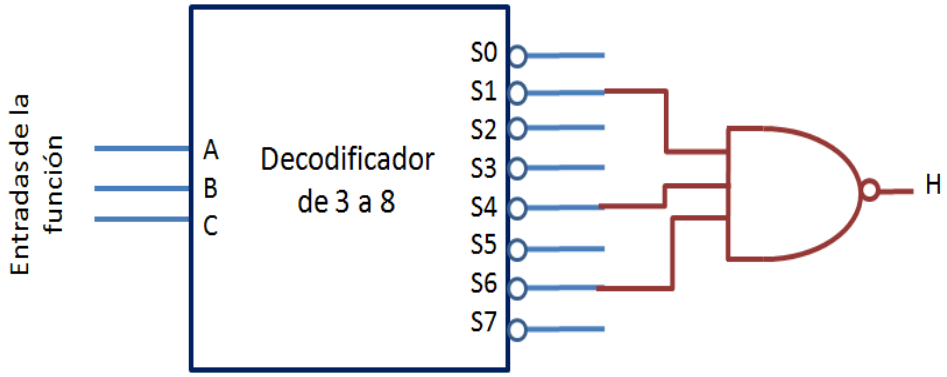

Figura 5 Implementación de la función G mediante un decodificador binario y una puerta NOR

[5] La propiedad llamada involución dice que $\bar{\bar{a}} = a$, es decir, que si negamos una función dos veces, obtenemos la función original.

4.3.3 Resumen

El resultado de utilizar una puerta AND usando los maxitérminos que SÍ son de la función es el mismo que el de utilizar una puerta NAND usando los maxitérminos que NO son de la función. Así pues, a nivel funcional las dos opciones son equivalentes. Sin embargo, algunas funciones tienes un número mayor de ceros que de unos en sus salidas, o viceversa. En estos casos, la utilización de una puerta AND o de una puerta NAND puede tener importancia en la complejidad y coste del circuito final, ya que es posible escoger la puerta con menor número de entradas.

4.3.4 Ejercicios

A continuación unos sencillos ejercicios. Para la función lógica H expresada por su forma canónica conjuntiva, responde a las siguientes preguntas:

$$H = \prod_{D,C,B,A} (0,2,3,6,8,12,13,15)$$

a. ¿Qué tamaño de decodificador binario necesitamos para implementarla?
b. Si utilizamos una puerta AND, ¿De cuantas entradas será la puerta?
c. Si utilizamos una puerta AND, ¿Cuáles serán las salidas del decodificador que conectaremos a la puerta AND?
d. Si utilizamos una puerta NAND, ¿De cuantas entradas será la puerta?
e. Si utilizamos una puerta NAND, ¿Cuáles serán las salidas del decodificador que conectaremos a la puerta NAND?
f. ¿Es mejor utilizar una puerta AND o una puerta NAND?
g. Intenta responder las preguntas antes de ver las soluciones, por favor [6].

5 Conclusiones

En este artículo has podido conocer una forma rápida y sencilla de implementar una función lógica. La Figura 6 muestra los pasos a seguir para, a partir de la tabla de verdad de una función lógica, llegar a la implementación de la misma utilizando un decodificador con salidas activas a nivel bajo y una puerta AND o una puerta NAND.

Algunas ideas importantes que debes recordar:

- El decodificador debe tener tantas entradas como entradas tenga la función. Esto es, deben tener la misma aridad.
- La entrada de menor peso de la función debe conectarse a la entrada de menor peso del decodificador. Y así sucesivamente con las siguientes entradas hasta la de mayor peso.
- Las salidas del decodificador que no se utilizan se dejan al aire.
- No es mejor utilizar una puerta AND o una puerta NAND, pero dependiendo de la función, es posible que resulte más sencillo una opción frente a la otra. Pero funcionalmente las dos opciones son idénticas.

[6] a: De 4 entradas a 16 salidas; b: De 8 entradas; c: S0, S2, S3, S6, S8, S12, S13, S15; d: De 16-8=8 entradas; e: S1, S4, S5, S7, S9, S10, S11, S14 f: La puerta NAND necesita una entradas menos que la puerta AND, por lo que parece mejor utilizar la NAND.

- Si el decodificador tiene entrada de habilitación, esta debe estar siempre activada, ya que de lo contrario no se genera la función deseada.

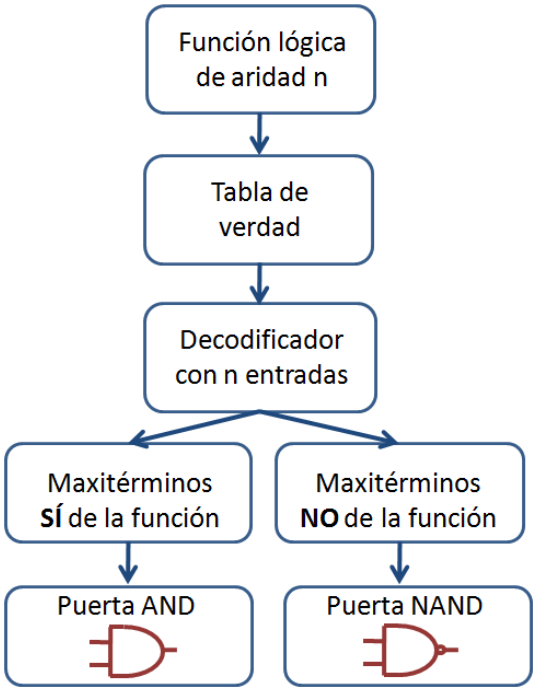

Figura 6 Pasos para la implementación de una función lógica utilizando decodificadores binarios con salidas activas a nivel bajo

6 Bibliografía

6.1 Libros:

[1] John F. Wakerly "Digital design : principles and practices", Upper Saddle River : Pearson 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 Referencias de fuentes electrónicas:

[3] Martí Campoy, Antonio "Circuitos combinacionales: decodificadores" Universitat Politècnica de València. Escola Tècnica Superior d'Enginyeria Informàtica. 2011. http://politube.upv.es/play.php?vid=46040