Tecnología de Computadores
Práctica 2
Funciones lógicas

Universidad Miguel Hernández [1]

22 de febrero de 2016

Resumen

Este documento describe la práctica número 2 de la asignatura Tecnología de Computadores de 2º del Grado en Ingeniería Informática en Tecnologías de la Información de la Universidad Miguel Hernández. Este material docente ha sido desarrollado enteramente, a menos que se indique lo contrario, por el profesor P. Pablo Garrido Abenza.

Índice general

Capítulo 1

Introducción

1.1. Objetivos

El principal objetivo de esta práctica es trabajar con funciones lógicas y realizar transformaciones sobre ellas para facilitar su implementación real.

Tras completar esta práctica el alumno será capaz de:

- Convertir cualquier función lógica a su equivalente en forma canónica, tanto en forma de suma de productos (minitérminos) como en forma de producto de sumas (maxitérminos).
- Expresar cualquier función lógica en forma de Tabla de Verdad.
- Obtener una función lógica a partir de su Tabla de Verdad.
- Diseñar circuitos combinacionales.
- Implementar un circuito esquemático con el simulador para verificar su funcionamiento.

1.2. Software necesario

El software necesario para la realización de las siguientes prácticas es *Simulín* v5.61 o posterior, el cual se encuentra ya instalado en los ordenadores del aula de informática. También puede se puede instalar en cualquier otro ordenador personal, ya que este software es de libre distribución. Está disponible para Windows, Linux y Mac OS X, y se puede descargar desde el material de la asignatura, o también, desde el siguiente enlace.

Capítulo 2

Descripción

La práctica consiste en realizar ciertas transformaciones sobre alguna función lógica a modo de ejercicio, como por ejemplo, transformar una función lógica a su forma canónica, ya sea como suma de productos o como producto de sumas, realizar transformaciones entre ambas formas canónicas, construir la Tabla de Verdad de una función lógica en forma canónica de forma manual, y realizar el proceso inverso, es decir, obtener la función lógica en forma canónica desde la Tabla de Verdad. También se implementarán los circuitos correspondientes en el simulador para demostrar la equivalencia entre todas esas formas. Por último se diseñará un circuito combinacional a partir de una descripción, implementando el circuito en el simulador.

2.1. Transformaciones en funciones lógicas

2.1.1. Transformación de una función lógica a su forma canónica

Supongamos las siguientes funciones lógicas:

$$f_1(a,b,c,d) = (a \cdot \bar{b} \cdot c) + (\bar{a} \cdot b) + (a \cdot \bar{b} \cdot c \cdot d) \tag{2.1}$$

$$f_2(a,b,c,d) = (a + \bar{b} + c) \cdot (\bar{b} + c + d) \cdot (a + b + \bar{c} + d) \tag{2.2}$$

Se pide realizar de forma manual (escribiendo en papel) la conversión de estas dos funciones lógicas a su forma canónica, la primera en forma de suma de productos, y la segunda en forma de producto de sumas. Después, implementar los dos circuitos de las funciones originales en el simulador, y los circuitos correspondientes a las funciones canónicas obtenidas. Validar los resultados comparando las tablas de verdad.

La Tabla 2.1 resume los circuitos a implementar en este apartado, y el nombre que el archivo deberá tener (obligatoriamente) en cada caso.

Cuadro 2.1: Archivos a implementar

#	Descripción	Archivo
1	f_1 original	`f1.sim`
2	f_2 original	`f2.sim`
3	f_1 canónica SoP	`f1_sop.sim`
4	f_2 canónica PoS	`f2_pos.sim`

Método para obtención de forma canónica:

- Para **suma de productos** (SoP): multiplicar cada término producto que no esté en forma canónica por un término formado por la suma de la variable que le falta y su complemento, cuyo valor lógico es 1 ($a + \overline{a} = 1$); al multiplicar por 1 no se altera el valor de un término producto. Después se aplica la propiedad distributiva ($a \cdot (b + c) = (a \cdot b) + (a \cdot c)$); el número de términos producto se duplica por cada variable que falta.
- Para **producto de sumas** (PoS): sumar a cada término suma que no esté en forma canónica un término formado por el producto de la variable que le falta y su complemento, cuyo valor lógico es 0 ($a \cdot \overline{a} = 0$); al sumar 0 no se altera el valor de un término suma. Después se aplica la propiedad distributiva ($a + (b \cdot c) = (a + b) \cdot (a + c)$); el número de términos suma se duplica por cada variable que falta.

2.1.2. Conversión a forma canónica contraria

Realizar de forma manual (escribiendo en papel) la conversión a su forma canónica contraria de las dos funciones lógicas obtenidas en el apartado anterior, es decir, la que teníamos en forma canónica de suma de productos pasarla a producto de sumas, y viceversa.

La Tabla 2.2 resume los circuitos a implementar en este apartado, y el nombre que el archivo deberá tener (obligatoriamente) en cada caso.

Cuadro 2.2: Archivos a implementar

#	Descripción	Archivo
5	f_1 canónica PoS	`f1_pos.sim`
6	f_2 canónica SoP	`f2_sop.sim`

Método para transformar una forma canónica en otra:

- Se trata de aplicar la regla involutiva o doble complementación: $a = \overline{\overline{a}}$, junto con alguno de los teoremas de DeMorgan: $\overline{(a+b)} = \overline{a} \cdot \overline{b}$ ó $\overline{(a \cdot b)} = \overline{a} + \overline{b}$, aplicados para cualquier número de variables.

2.1.3. Construcción de la Tabla de Verdad

Construir de forma manual las cuatro Tablas de Verdad correspondientes a las cuatro funciones lógicas en forma canónica obtenidas en los apartados anteriores: circuitos #3, #4, #5, y #6. Comprobarlas con las obtenidas utilizando el simulador.

Método para construir la Tabla de Verdad desde una función lógica canónica:

- Desde suma de productos (SoP): colocamos un 1 en las combinaciones que hacen que los productos de la expresión sean 1 (cierto). Colocaremos un 0 en el resto.
- Desde producto de sumas (PoS): colocamos un 0 en las combinaciones que hacen que los términos suma sean 0 (falso). Colocaremos un 1 en el resto.

2.1.4. Obtención de la función lógica

Una vez obtenida la tabla de verdad del apartado anterior, haremos el proceso inverso, para conocer la técnica de obtención de una función lógica en forma canónica desde la tabla de verdad, al tiempo que validamos los resultados. Se trata de hacer la tarea de forma manual, mentalmente o escrito en papel.

Método para obtener la función lógica canónica desde la Tabla de Verdad:

- Para suma de productos: consideramos las combinaciones que hacen cierta la función (1). Cada combinación así será un término producto (minitérmino) de forma directa, y luego se sumarán todos.
- Para producto de sumas: consideramos las combinaciones que hacen falsa la función (0). Cada combinación así será un término suma (maxitérmino) de forma inversa, y luego se multiplicarán todos.

2.2. Diseño de un circuito combinacional de un caso real

En este apartado vamos a diseñar un circuito lógico para resolver un caso real: la división de un número BCD entre 3.

El **procedimiento** que vamos a seguir es el siguiente (notar que el paso 4 se hará en la próxima práctica):

1. Comprender el sistema a diseñar, identificando las **entradas** y **salidas** del sistema.

2. Construir la **Tabla de Verdad**, asignando los valores 0 y 1 a las salidas según el valor de las entradas.

3. Obtener las **funciones lógicas** (una por cada salida), tanto en forma de Suma de Productos como en forma de Producto de Sumas.

4. **Simplificar** mediante Mapas de Karnaugh las funciones, utilizando o no términos indiferentes, cosa que haremos en próximas prácticas.

5. Implementar el **circuito**.

Como ya se sabrá, la división de un número entre otro tiene dos partes: cociente y resto, ambos enteros. El circuito a diseñar se encargará de realizar la división de un número codificado en BCD (Binary-Coded Decimal) entre 3 y proporcionar el cociente.

Se recuerda que el código BCD asigna un valor binario de 4-bit a los valores decimales 0..9, por lo que en nuestro circuito tendremos 4 **entradas**, a las que llamaremos: `a`, `b`, `c`, `d`, siendo `a` al bit más significativo (MSB) y `d` el bit menos significativo (LSB).

Primero calcularemos la tabla de la izquierda, con el cociente de todas las divisiones entre 3, desde 0 hasta 9. Como se puede apreciar, necesitaremos 2 bits para almacenar el cociente, por lo que el circuito tendra 2 **salidas**, a las que llamaremos C1 y C0, siendo `C1` al bit más significativo (MSB) y `C0` el bit menos significativo (LSB).

Número BCD	Cociente / 3
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

a	b	c	d	C1	C0
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Una vez calculados todos los resultados, procederemos a rellenar la **Tabla de Verdad**, que contendrá lo mismo que la tabla de la izquierda pero en binario; a las combinaciones que no se deben dar en BCD (10 a 15) les asignaremos 0 (de momento). A continuación obtendremos las dos **funciones lógicas** para **C1**, y **C0**, tanto en forma de Suma de Productos como en forma de Producto de Sumas:

- Suma de Productos (archivo: `divisionbcd_sop.sim`):
  C1 = _____
  C0 = _____

- Producto de Sumas (archivo: `divisionbcd_pos.sim`):
  C1 = _____
  _____
  C0 = _____
  _____

Finalmente se implementará y validará con el simulador el **circuito** correspondiente a las funciones expresadas tanto en forma de Suma de Productos como en forma de Producto de Sumas.

Capítulo 3

Entrega de la práctica

La forma de entrega de esta práctica se realizará de la siguiente forma:

1. Comprimir en un único archivo todos los circuitos *.sim implementados. Para ello se puede utilizar programas como WinZip o 7z, admitiéndose los formatos: .zip, .7z, o .rar.
2. Subir el archivo comprimido a través del apartado *Tareas* de la página de la UMH, en su práctica correspondiente, antes del plazo fijado.

Los archivos a entregar para esta práctica se resumen en la tabla 3.

#	Descripción	Archivo
1	f_1 original	`f1.sim`
2	f_2 original	`f2.sim`
3	f_1 canónica SoP	`f1_sop.sim`
4	f_2 canónica PoS	`f2_pos.sim`
5	f_1 canónica PoS	`f1_pos.sim`
6	f_2 canónica SoP	`f2_sop.sim`
7	División BCD SoP	`divisionbcd_sop.sim`
8	División BCD PoS	`divisionbcd_pos.sim`

Cuadro 3.1: Archivos para los circuitos a implementar

RECUERDA: las prácticas serán corregidas por un programa informático, por lo que se insiste en que los archivos de los circuitos tengan el nombre que se especifica en las tablas, respetando incluso mayúsculas/minúsculas; el nombre del archivo comprimido que los contiene no importa.