CIENCIA BÁSICA Y CULTURA

Boletín de Ciencias Básicas

Año 2024 Número 12 15 de septiembre

Metodología visual de enseñanza de los sistemas de numeración

Parte 2: Sistemas numéricos ortogonal, hipercubo

Dra. Elizabeth Fonseca Chávez,

(Coordinadora de ingeniería en telecomunicaciones DIE)

Sistema de numeración ortogonal basado en el código gray.

El código Gray o código reflejado está construido especialmente para que de un estado a otro se ejecute un cambio de un sólo bit exclusivamente. Se muestra a continuación la tabla 1 que comúnmente colocan los autores de libros de diseño lógico. El formato de trabajo está fundamentado en la base de los sistemas de numeración, los cuales se explicaron en el boletín anterior.

Tabla 1. Tabla de código Gray para cuatro bits.

Decimal	Código Reflejado 4 bits
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Algunos autores mencionan determinadas aplicaciones del código Gray en el proceso de convertir un sistema físico a uno discreto Mano[1]. El código Gray es muy utilizado en mapas de Karnaugh para minimizar funciones booleanas y para crear visualmente hipercubos, que también sirven para minimizar funciones boolenas.

La transición de un sólo bit, entre un estado y otro, nos permite corregir errores digitales (estados no esperados) por ejemplo del número 3 al 4 en decimal; en binario tenemos 011 y 100, para pasar un circuito del 3 al 4 en binario se va cambiando bit por: 011 010 000 100; estos estados no eran los pedidos y pueden afectar nuestro sistema, mandando algún control no pedido a otro sistema conectado.

Ahora, si no los vemos como índices de estados sino como valores, el cambio de un bit solamente nos permite trabajar con bases ortogonales.

Comenzamos por explicar cómo se construye este código reflejado. Se parte de un bit 0,1, después, para construir dos bits se aumenta un cero a la izquierda en este primer par 0 0, 0 1 y se asigna un 1 en el siguiente par, con el "reflejo de los bits anteriores, es decir se coloca el último bit 1 y luego el 0, quedando los pares de esta manera: 1 1, 1 0. Obsérvese la tabla siguiente para el armado de los primeros dos bits. Tenemos la figura 1, con una representación de espejo que intenta mostrar que el bit menos significativo se intercambia y se repite abajo del espejo, mientras que el otro bit sólo cambia de cero a uno. Para tres bits, en la figura 2 tenemos el código Gray con tres bits y su representación de espejo con líneas horizontales.

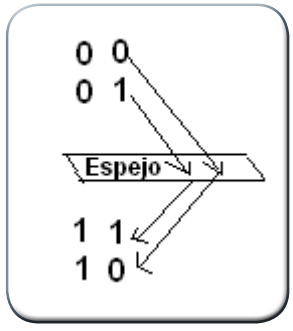

Figura 1. Representación Visual de la formación del código Gray mediante la utilización de un espejo virtual

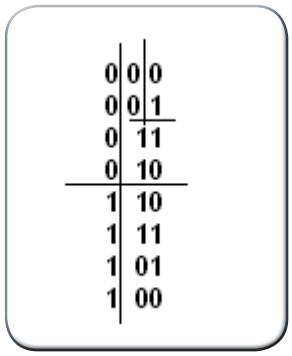

Figura 2. Representación del código Gray para tres bits con líneas horizontales que representan un espejo virtual

De esta manera se continúa el código a cuatro bits, donde nuevamente se refleja una parte y la otra se sigue expandiendo.

Sistema de numeración binario ortogonal (GRAY)

Si es un código, puede limitarse a los bits establecidos, en general trabajan hasta 4 bits, dependiendo del código. En el caso de construirlo como sistema de numeración se vuelve casi ilimitado.

Este singular sistema que estamos creando está fundamentado en la base 2, pero la base completa de este sistema de numeración ortogonal comienza con DOS bits. Por lo tanto, su construcción será exclusivamente cada 2, 4, 8, 16, etc. pares de bits. Sin embargo, existen todos los números (véase figura 3).

El sistema bidimensional es como un mapa de Karnaugh (métodos de minimización para funciones booleanas).

Base binaria ortogonal: 00 01 11 10.

Con intercambios reflejados.

Para 2 bits:	Para 4 bits:	Para 8 bits:
00 01 11 10	0000 0001 0011 0010 0100 0101 0111 0110 1100 1101 1111 1110 1000 1001 1011 1010	0000 00001 0000 00011 0000 00010 0000 00110 0000 00111 0000 00101 0000 00100 0000 01100

Figura 3. Sistemas de numeración binaria Gray para 2,4 y 8 bits

Gráficamente, tenemos la figura 4 con la visualización con mapas de Karnaugh (tabla de ordenamiento de mintérminos de una tabla de verdad) para dos bits y con flechas de orientación donde se continúa la numeración. Un mapa de este estilo debe tener valores internos para combinar, por lo tanto, deben ser ortogonales, es decir, sistemas independientes como lo permite el código Gray, con el objetivo de reducir una función en sus dimensiones y ahorrar hardware.

Tabla de verdad	
Minterminos	Función
00	
01	
10	
11	

Mapa de Karnaugh

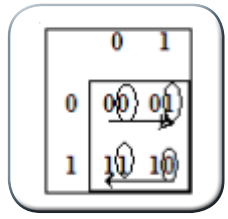

Figura 4. Muestra el trabajo de formación del código Gray en una representación de mapas de Karnaugh para dos bits y su respectiva tabla de verdad sin datos en la función

En la figura 5, tenemos el mapa de Karnaugh para 4 bits adentro de cada casillero y el conteo de la numeración.

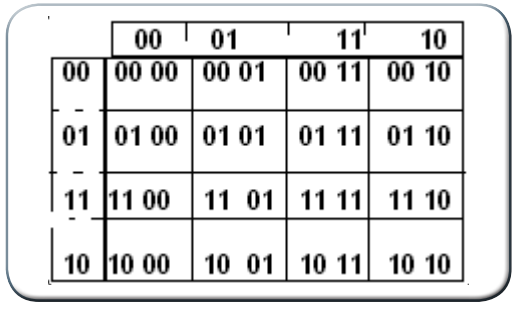

Figura 5. Muestra del código Gray en un mapa de Karnaugh

De este mapa de Karnaugh se debe notar cómo va avanzando la numeración, en ZIGZAG. A continuación, se muestra en la figura 6 marcado con flechas sobre cómo va avanzando la numeración en este formato.

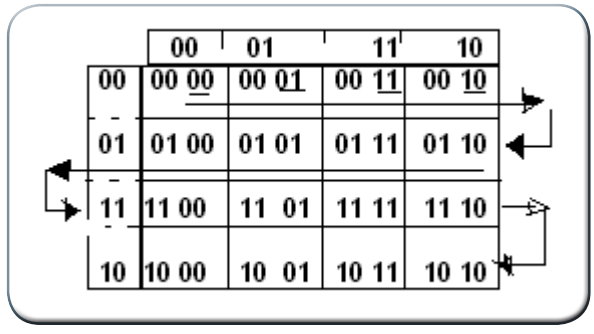

Figura 6. Rutas de cómo se va dirigiendo el sistema de numeración dentro del mapa de Karnaugh

Una de las aplicaciones de un mapa de Karnaugh es reducir variables mediante la agrupación de grupos de 2n , es decir de 2, 4, 8, 16,... de una tabla de verdad con resultados de "1", lo que evita utilizar componentes de compuertas AND, OR o NOT en forma excesiva y que el costo del sistema disminuya.

Finalmente, para n bits, tenemos la figura 7, que nos muestra sólo una idea de cómo se seguiría formando el código Gray.

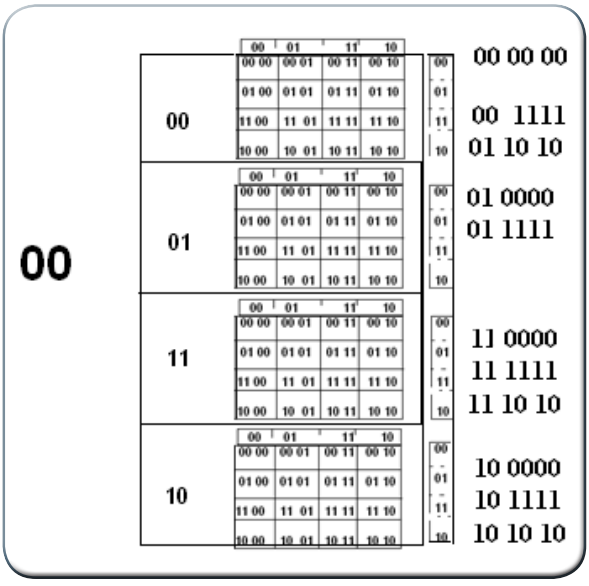

Figura 7. Ejemplo ilustrativo de cómo se formaría visualmente el sistema de numeración de base ortogonal Gray para n bits

HIPERCUBO

El hipercubo se representa con un sistema de numeración ortogonal. Específicamente utiliza el código Gray, ya que se pide que cada vértice debe estar unido al siguiente vértice por solamente UN CAMBIO de bit. La utilización del hipercubo es la misma que los mapas de Karnaugh, para reducir dimensiones de una función y ahorrar hardware.

En la figura 9 se visualiza la construcción del cubo base con un bit, sólo se construye una línea de dos vértices (dos posibilidades) en el inciso a; si se trabaja con dos bits sólo podemos construir una cara del cubo conteniendo 4 vértices, por lo que la trayectoria de construcción de generación de numeración se escogió en sentido antihorario, como se observa en la figura 9 inciso b.

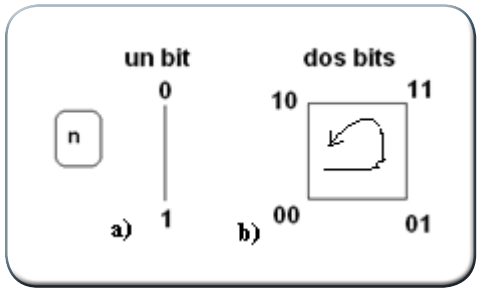

Figura 9. La Representación del hipercubo para un bit se muestra en el inciso a y para dos bits en el inciso b

En la figura 10, inciso a, tenemos 3 bits, 8 posibilidades y representa un cubo. En el inciso b, tenemos bits 16 posibilidades.

En la figura 10 tenemos la representación del hipercubo con 5 bits y 32 posibilidades; 2N. (Las mismas combinaciones que el sistema binario pero ordenado de otra manera).

Para un hipercubo de 6 bits tendríamos 64 posibilidades, etc.

BASE: CUBO

Se repite el cubo tantas veces requiramos aumentar la dimensión o bit.

Un bit: Una línea. Figura 7 inciso a.
Dos bits: Un cuadrado. Figura 7 inciso b.
Tres bits: Un CUBO. Figura 8 inciso a.
Cuatro bits: Un CUBO base dentro de un CUBO más grande.

Nótese que los vértices del cubo pequeño deben conectarse al cubo grande. Figura 10 inciso b.

Cinco bits: CUBO base dentro de cubo grande, y este dentro de cubo más grande. Todos los vértices se conectan en cascada.

Obsérvese en la figura 11.

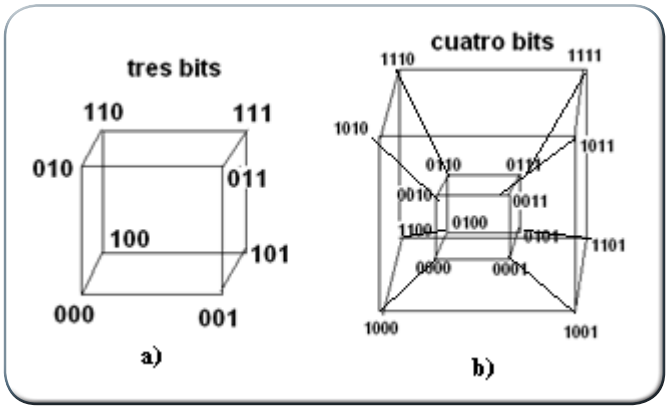

Figura 10. La Representación del hipercubo para tres bits se muestra en el inciso a y para cuatro bits en el inciso b

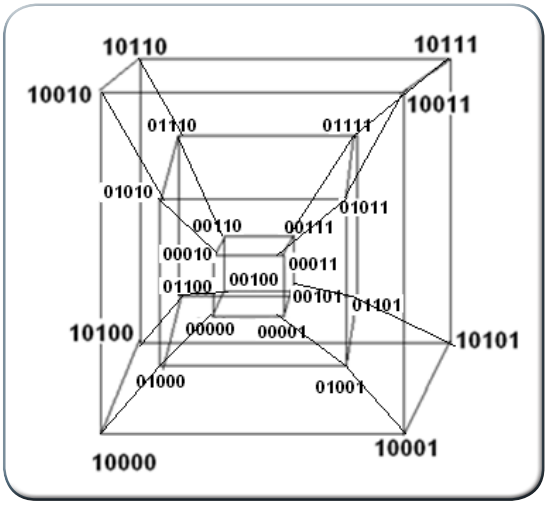

Figura 11. La Representación del hipercubo para cinco bits

Es posible comenzar a realizar sistemas de numeración de un sistema ortogonal con aplicaciones a mapas de Karnaugh o trabajar con hipercubos, para reducir variables en los sistemas digitales en las materias de diseño digital de compuertas lógicas, y hasta llegar a comprender los Qbits en otro boletín.

Referencias

[1] Morris Mano. Diseño digitales. Tercera edición. (También en todas sus ediciones) pág. 1-35.

[2] Brown Stephen –Vranesic Zvonko. Cap. 5. Representación de números. Fundamentos de Logica digital con VHDL, segunda edición. Mc Graw Hill Pag. 245-249

[3] Lloris, -Prieto-Parrilla. Sistemas Digitales.2003, ed. Mc Graw Hill pag.173-186

[4] Deitel, Apéndice E, Como programar en c/c++, segunda edición, Prentice Hall, pag.892-906.

[5] Ronald J. Tocci Cap.2 Sistemas Numéricos y códigos. SISTEMAS DIGITALES principios y aplicaciones Sistemas, pag. 20-44

[6] Santiago Casado, Los sistemas de Numeración a la largo de la historia, santiago@airastur.es http://thales.cica.es/rd/Recursos/rd97/Otros/SISTNUM.html

[7] Luis de la Peña, capitulo 3.Ecuacion estacionaria de Schrodinger, introducción a la mecánica cuántica, UNAM/fondo de cultura económica. Pag. 62.

[8] Ivan S. Oliveira, NMR Quantum Informatión Processing, Ed.ELSEVIER, 2007; pag. 114.

[9] David McMahon, Quantum Computing Explained, Wiley-Interscience. 2007.

[10] Zbigniew Oziewicz,Jose Hugo Max, oarrafo 3.1 Dirac braket, pag.11.

[11] Zbigniew Oziewicz, BIRKHOFF'S THEOREMS VIA TREE OPERADS, parrafo2: Algebras in Graphic, pag. 2 Bulletin of the Section of Logic Volume 28/3 (1999), pp. 1–14.

Diseño y formación editorial: L.D.G. Luis Enrique Vite Rangel