IMPLEMENTACIÓN EN LENGUAJE GRÁFICO DE UN ALGORITMO BASADO EN QUINE-McCLUSKEY Y PETRICK PARA MINIMIZACIÓN GLOBAL

Implementation in a graphical language of an algorithm based on Quine-McCluskey and Petrick methods for global minimization

RESUMEN

En este trabajo se presenta la implementación en un lenguaje gráfico de un algoritmo basado en los métodos de Quine-McCluskey y Petrick para la minimización global óptima de funciones simultáneas booleanas que comparten el mismo conjunto de variables de entrada. Se muestra como la minimización global, en comparación con las implementaciones clásicas, obtiene una solución de menor costo a la obtenida mediante la combinación de la minimización óptima individual de cada función. Se discute su utilidad práctica y didáctica.

PALABRAS CLAVES: Síntesis funciones booleanas, Petrick, Quine-McCluskey, Solución óptima, Programación gráfica, LabVIEW.

ABSTRACT

This paper shows the implementation in a graphic language of an algorithm based on Quine-McCluskey and Petrick methods for optimal global minimization of simultaneous switching functions sharing their input set of variables. It is shown how this global minimization, when compared to classic implementations, achieve a better solution than the one obtained combining optimal solutions for individual functions. Its practical and didactic usability is discussed.

KEYWORDS: *Switching functions minimization, Petrick, Quine-McCluskey, Optimal solution, Graphical programming, LabVIEW.*

MAURICIO HOLGUÍN L.
Ingeniero Electricista.
Profesor Catedrático
Estudiante Maestría en Ingeniería Eléctrica.
Universidad Tecnológica de Pereira
ma_hol@ohm.utp.edu.co

ANDRÉS ESCOBAR MEJÍA
Ingeniero Electricista, M. Sc.
Profesor Auxiliar
Universidad Tecnológica de Pereira
andreses1@utp.edu.co

GERMÁN A. HOLGUÍN L.
Ingeniero Electricista, M. Sc.
Profesor Asistente
Universidad Tecnológica de Pereira
gahol@utp.edu.co

Grupo de Investigación en Control e Instrumentación.

1. INTRODUCCIÓN

En la literatura relacionada con sistemas digitales se ha tratado ampliamente la minimización de funciones de conmutación como herramienta fundamental en la síntesis de circuitos lógicos [1-6].

Tradicionalmente se emplean métodos como el de Veitch-Karnaugh, el cual se basa en formas rectangulares de los diagramas de Venn para hacer representaciones de conjuntos. Los Mapas de Veitch-Karnaugh se han mostrado como la herramienta de uso más común, pero con limitación práctica a una única función y generalmente con no más de 5 variables [7-14].

En la práctica, las implementaciones de circuitos lógicos de conmutación casi siempre involucran más de dos o tres funciones y con no menos de cuatro variables; lo cual implica el uso de un mapa de Karnaugh por variable conllevando a una realización mínima por función, más no la realización mínima global, la cual por ejemplo podría aprovechar resultados intermedios de una función para evaluación de otras, sin que con ello necesariamente una función se base en una implementación mínima, pero sí produciendo una implementación general mínima para el conjunto de funciones [8, 9].

La gran mayoría de literatura siempre hace referencia a métodos de simplificación para una única función. Sin embargo en algunos textos como [8, 9, 10], se encuentra métodos alternos de simplificación tanto para aplicaciones de una sola función como para aquellas con múltiples funciones. Entre estos métodos el más destacado y recurrente es el de Quine-McCluskey que se fundamenta en representaciones tabulares de los diagramas de Venn y que siempre se referencia como método general para implementaciones programadas y con el poder de tratar más variables que los mapas de Karnaugh [8, 9, 10].

El método de Quine-McCluskey, al igual que los Mapas de Karnaugh, tiene un componente heurístico cuando se trata de completar la cubierta, o conjunto de implicantes primos necesarios para terminar de cubrir los mintérminos de una función y que no son cubiertos por los implicantes primos esenciales [8, 9]. Para implementaciones con un gran número de variables podría ser que esta parte heurística conduzca a resultados

Fecha de Recepción: (Letra Times New Roman de 8 puntos)
Fecha de Aceptación: Dejar en blanco

no mínimos, por lo que generalmente se recurre al algoritmo de Petrick con el fin de poder evaluar la realización mínima [9].

Indagando sobre implementaciones programadas realizadas tanto en varias referencias como en programas típicos de simplificación, se ha encontrado que es prácticamente nula la implementación de los algoritmos generales de Quine-McCluskey y de Petrick y que en los casos existentes éstas implementaciones siempre se limitan a una única función.

Ya que lo común es enfrentar soluciones a problemas como más de una función y de cuatro variables, los programas existentes de simplificación no entregan la solución global óptima. Todo lo anterior ha motivado a los autores ha realizar la implementación de un programa de minimización global con base en estos algoritmos de simplificación, optando por una aproximación desde los lenguajes gráficos de programación con el fin de aprovechar al máximo las tecnologías existentes en cuanto a procesamiento multi-núcleo y sin mucha experiencia en programación de tareas multi-hilo requerida en los lenguajes textuales. Todo lo anterior con miras a futuras comparaciones, ya sea con otras implementaciones y/o algoritmos, permitiendo cuantificar de forma rápida la eficiencia y aplicabilidad práctica de cada desarrollo.

2. MARCO TEÓRICO

Inicialmente se presenta la descripción de los algoritmos de simplificación de Quine-McCluskey y Petrick, describiendo de forma rápida su naturaleza.

Todos los métodos clásicos de simplificación tienen su fundamento en los diagramas de Venn, los cuales son representaciones en contorno cerrado de conjuntos, con rectángulos para representar comúnmente el conjunto universal y círculos para los conjuntos. Un diagrama de Venn para tres variables booleanas se muestra en la Figura 1, donde es claro que los tres conjuntos describen un total de ocho regiones (cada una representa a su vez un posible mintérmino de la función) que se caracterizan por cambio de un sólo bit entre regiones adyacentes.

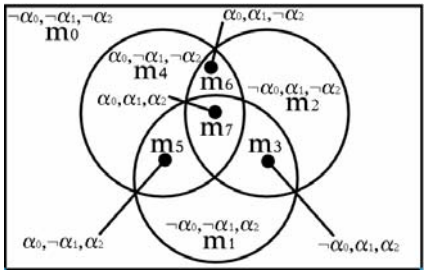


Figura 1. Diagrama de Venn para 3 variables.

Un mapa de Karnaugh no es más que una representación rectangular de un diagrama de Venn que conserva sus dos características fundamentales: cada región posee tantas regiones vecinas como número de variables y entre ellas sólo cambia un bit [7 a 14]. Un mapa de Karnaugh para tres variables se muestra en la Figura 2.

α_2 \ α_0,α_1	00	01	11	10
0	0	2	6	4
1	1	3	7	5

Figura 2. Mapa de Karnaugh para 3 Variables.

La simplificación consiste esencialmente en aprovechar el hecho que cuando regiones adyacentes representan mintérminos, éstas se pueden agrupar y eliminar las variables que cambia de valor entre ellas, por lo que las agrupaciones deben ser en potencias de 2[1,2]. Además las agrupaciones deben ser tal que cada una sea lo más grande posible y con el mínimo de agrupaciones para cubrir la totalidad de mintérminos de la función [8,9]. Éstas consideraciones que en principio parecen fáciles, son directas en la simplificación para una función, más no siempre ciertas para implementaciones con funciones simultáneas simplificadas individualmente por mapas de Karnaugh.

2.1. ALGORITMO DE QUINE-McCLUSKEY

El algoritmo de Quine-McCluskey es una implementación tabular de los mapas de Karnaugh [2, 4, 5], lo cual permite un manejo amplio de más de 5 variables y que se puede extender fácilmente a la simplificación de dos o más funciones simultáneas para garantizar la mínima cantidad de grupos para cubrir el global de la implementación y a su vez haciendo cada grupo lo más grande posible dentro del criterio global de minimización [9].

A continuación se hace una descripción general del algoritmo de Quine-McCluskey [3, 4, 5, 8, 9]:

1. Identificar para cada función el número de unos en la representación binaria de sus mintérminos. Si existen términos *Dont'care*, estos deben ser incluidos.
2. Poner todos los mintérminos formando una columna y organizados según el número de unos que contengan. Se debe indicar además para cada mintérmino las funciones que cubre.
3. Buscar adyacencias entre grupos vecinos, identificando con un guión la variable que cambia y las funciones que abarca. Dos términos son adyacentes si sólo cambia una variable y como mínimo comparten una función. Si un término se

agrupa abarcando todas las funciones a las que pertenece no es implicante primo.

4. Repetir búsqueda de adyacencias formando cada vez una nueva columna hasta que no se puede realizar más agrupaciones.
5. Formar una nueva tabla con implicantes primos en filas y mintérminos por función en las columnas. En esta parte no se incluyen los términos *Dont'care*. Cada mintérmino se cruza contra las agrupaciones que lo contienen. Si un mintérmino sólo es cubierto por un implicante primo, éste es un implicante primo esencial que debe ir en la cubierta y cubre de forma global todos los mintérminos adicionales que contiene.
6. Identificar el mínimo número de implicantes primos necesarios para cubrir los mintérminos aún no cubiertos de cada función.

2.2. ALGORITMO DE PETRICK

El punto 6 descrito en el algoritmo de Quine-McCluskey implica una búsqueda heurística del mínimo número de implicantes primos que terminen de cubrir los mintérminos faltantes. Este procedimiento podría conllevar a una solución no óptima (mínima) en casos donde se tengan muchas posibles soluciones, lo cual es común cuando se plantean sistemas con varias funciones.

El algoritmo de Petrick permite encontrar la solución mínima mediante un procedimiento claro y secuencial, el cual se describe a continuación [6,9]:

1. Escribir una expresión en forma de producto de sumas (POS), donde cada término suma contiene todos los posibles implicantes primos para un mintérmino no agrupado aún.
2. Llevar la forma POS obtenida a una forma de suma de productos (SOP). Para ello se emplea la ley distributiva y se simplifican literales mediante leyes de absorción o involución.
3. Cada término producto de la forma SOP resultante representa una posible combinación de cubierta, lo cual quiere decir que se debe seleccionar la combinación mínima.

3. IMPLEMENTACIÓN DEL ALGORITMO EN UN LENGUAJE GRÁFICO

Los algoritmos previamente descritos han sido implementados con ayuda del software de programación gráfica LabVIEW8.0. La programación se basa esencialmente en una estructura de secuencia, ver la Figura 3, que va progresivamente realizando los diferentes pasos descritos por los algoritmos.

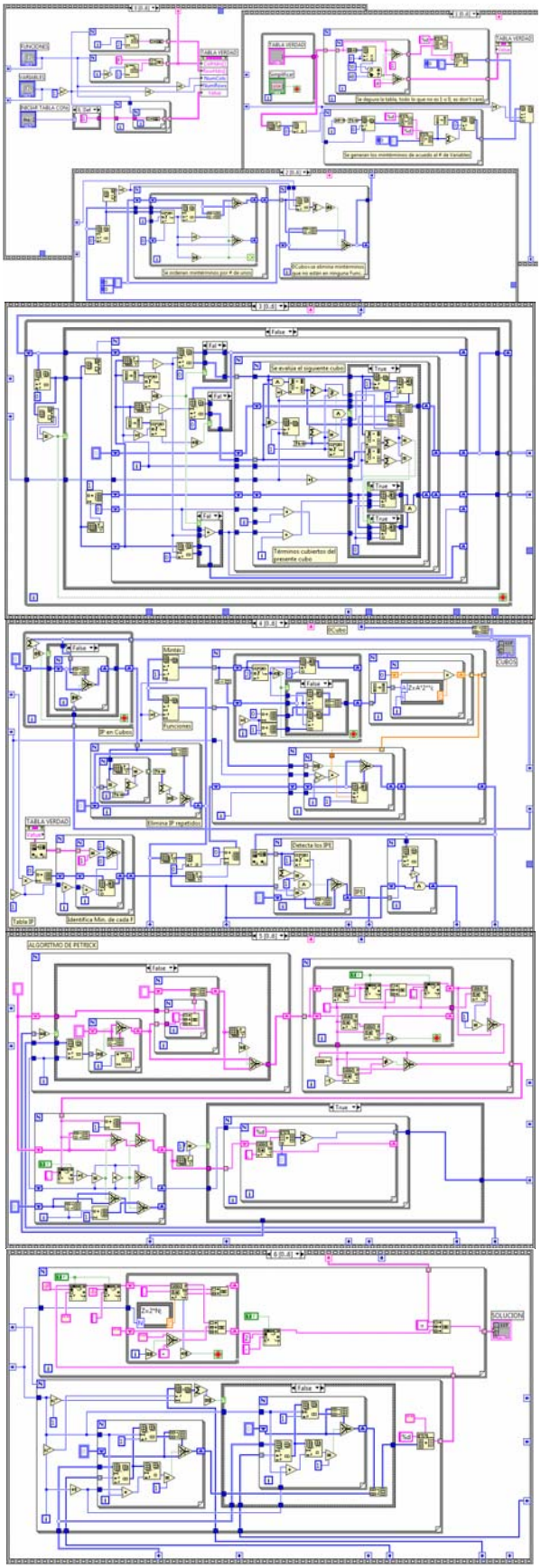


Figura 3. Marcos de estructura de secuencia para el programa.

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

Los marcos de la estructura de secuencia realizan respectivamente las siguientes labores:

Marco 0: Se encarga de inicializar la tabla de verdad en valores como: Iniciar tabla con ceros, unos o *Dont'cares*.
Marco 1: Realiza una depuración de la tabla de verdad ingresada, con lo cual todo lo que no se introduce como un 1 o un 0 lógico se trata como un *Dont'care*.
Marco 2: Realiza puntos 1 y 2 de Quine-McCluskey.
Marco 3: Realiza puntos 3 y 4 de Quine-McCluskey.
Marco 4: Realiza punto 5 de Quine-McCluskey.
Marco 5: Realiza algoritmo de Petrick.
Marco 6: Realiza ajuste para presentación de resultados.

El poder del algoritmo combinado de Quine-McCluskey con Petrick se muestra a continuación con aplicación a la simplificación de un conjunto de funciones de referencia. El objeto es mostrar la operación adecuada del algoritmo implementado, demostrar su alcance de simplificación global y además tener un sistema de funciones de referencia con base en el cual se puedan hacer futuras comparaciones. La interfaz de usuario para el programa se muestra en la siguiente sección.

4. EJEMPLO DE REFERENCIA

Para mostrar la aplicabilidad del método se ha seleccionado un ejemplo clásico encontrado en [9] páginas 220 a 222. Adicional a este ejemplo, se ha verificado la correcta operación del programa implementando todos los ejemplos que trae la misma referencia tanto para el algoritmo de Quine-McCluskey como para el de Petrick, ver páginas 211 a 244.

El sistema costa de un conjunto de tres funciones con cuatro variables, donde las variables son comunes para todas las funciones.

$$\begin{aligned} f_\alpha(A,B,C,D) &= \sum m(0,2,7,10) + d(12,15) \\ f_\beta(A,B,C,D) &= \sum m(2,4,5) + d(6,7,8,10) \\ f_\gamma(A,B,C,D) &= \sum m(2,7,8) + d(0,5,13) \end{aligned} \quad (1)$$

Inicialmente se realiza la simplificación usando mapas de Karnaugh, para lo cual cada función es llevada de forma individual a uno de estos mapas. El procedimiento de simplificación se asume conocido por parte del lector.

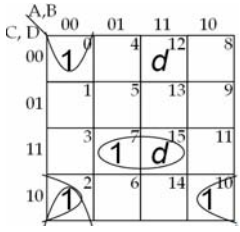


Figura 4. Mapa de Karnaugh para f_α

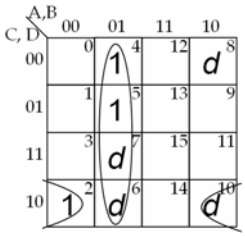


Figura 5. Mapa de Karnaugh para f_β

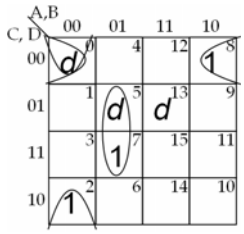


Figura 6. Mapa de Karnaugh para f_γ

De las Figuras 4, 5 y 6 se obtienen las expresiones mínimas por función, las cuales se pueden denominar como mínimos locales. Estas expresiones tienen forma SOP y son las siguientes:

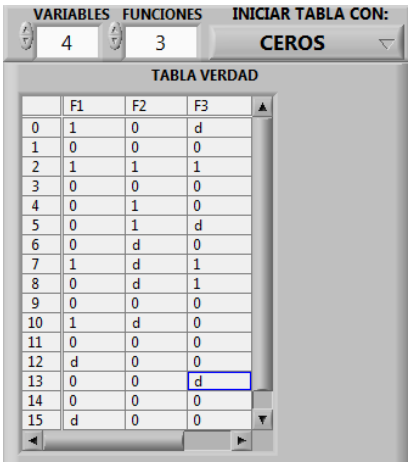
$$\begin{aligned} f_\alpha(A,B,C,D) &= BCD + \bar{A}\bar{B}\bar{D} + \bar{B}C\bar{D} \\ f_\beta(A,B,C,D) &= \bar{A}B + \bar{B}C\bar{D} \\ f_\gamma(A,B,C,D) &= \bar{A}BD + \bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{D} \end{aligned} \quad (2)$$

El costo de realización de una implementación se define como la cantidad de elementos de circuito y su tipo necesarios para llevarlo a efecto [7 a 14]. En general se debe tener en cuenta: la cantidad de términos productos diferentes; sí se desea realizar todos estos productos con un único tipo de elemento (por ejemplo, solo con compuertas AND de dos entradas); la cantidad de términos suma de acuerdo con la naturaleza del elemento que las implementa (por ejemplo, solo con compuertas OR de dos entradas) y el número total de literales usados. Para el ejemplo realizado con mapas de Karnaugh se tienen las siguientes relaciones de elementos a usar:

Total de términos producto diferentes: 6
Total de compuertas AND de 2 entradas necesarias: 9
Total de compuertas OR de 2 entradas necesarias: 5

En esta evaluación de costos se ha tenido en cuenta los términos AND de 2 variables que igualmente se pueden reutilizar en otros términos, como por ejemplo, en f_α el término $\bar{B}C\bar{D}$ puede ser implementado haciendo primero $\bar{B}\bar{D}$ para poder reutilizar esta parte en el término $\bar{B}\bar{C}\bar{D}$ de la función f_γ. Sin embargo, se debe tener en cuenta que los términos reutilizables son circunstanciales en minimizaciones locales.

Usando el algoritmo de minimización global que se ha programado, se pretende observar cómo se comportan estos mismos costos. Para ello inicialmente se introducen los datos en la interfaz de usuario, la cual se muestra en la Figura 7 con las configuraciones para las funciones ya descritas en la ecuación (1):



	F1	F2	F3
0	1	0	d
1	0	0	0
2	1	1	1
3	0	0	0
4	0	1	0
5	0	1	d
6	0	d	0
7	1	d	1
8	0	d	1
9	0	0	0
10	1	d	0
11	0	0	0
12	d	0	0
13	0	0	d
14	0	0	0
15	d	0	0

Figura 7. Interfaz de usuario del programa.

En la interfaz, se ingresan los datos directamente en forma de tabla de verdad, asignando un 1 a los mintérminos que se deben cubrir, una *d* a los términos *Dont'care* y un 0 a los demás. Por facilidad se ha programado la inicialización de la tabla con valores adecuados, por ejemplo, en el caso descrito la tabla posee más ceros que cualquier otro valor, por lo que se ha optado inicializarla en consecuencia.

Con los datos ya ingresados, se procede a solicitar la simplificación, como se observa en la Figura 8.

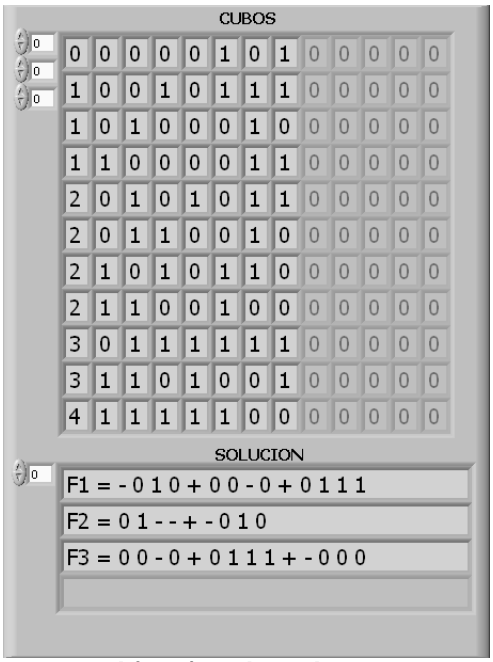


Figura 8. Simplificación mediante algoritmo programado.

En la parte superior a la solución, Figura 8, se pueden consultar los valores intermedios producto de los puntos 3 y 4 del algoritmo de Quine-McCluskey y que de forma generalizada se conocen como los *n*-Cubos.

En la solución obtenida, para cada función la respuesta está igualmente en forma SOP, asignando un cero a la posición de una variable complementada, un uno a la posición de una variable sin complementar y un guión para la posición de una variable que se simplifica. Por tanto, llevando a una notación igual a la mostrada para la ecuación (2), los resultados son:

$$\begin{aligned} f_\alpha(A,B,C,D) &= \bar{B}C\bar{D} + \bar{A}\bar{B}\bar{D} + \bar{A}BCD \\ f_\beta(A,B,C,D) &= \bar{A}B + \bar{B}C\bar{D} \\ f_\gamma(A,B,C,D) &= \bar{A}\bar{B}\bar{D} + \bar{A}BCD + \bar{B}\bar{C}\bar{D} \end{aligned} \quad (3)$$

Los costos de implementación de esta realización se muestran a continuación:

Total de términos producto diferentes: 5
Total de compuertas AND de 2 entradas necesarias: 8
Total de compuertas OR de 2 entradas necesarias: 4

En el anterior totalizado de elementos, se ha tenido en cuenta que ahora es posible reutilizar adicionalmente un resultado parcial de una operación OR entre dos términos comunes a las funciones f_α y f_γ.

Una observación rápida de los resultados muestra que la realización obtenida por el programa no implica una solución mínima para cada una de las funciones a implementar, pero evidentemente la solución requiere ahora de menos elementos de circuito y por ende, se logra una solución mínima global en comparación a la obtenida con los mapas de Karnaugh (2). Lo anterior se logra gracias a que el método hace una búsqueda exhaustiva de todos los posibles implicantes primos y le asigna prioridad a aquellos que son compartidos entre varias funciones lo que permite a la postre una reutilización mayor de resultados intermedios en la implementación global.

5. TRABAJOS FUTUROS

El propósito de los autores es seguir ahondando en métodos de simplificación para funciones de conmutación, donde se pueda superar las limitaciones prácticas de los métodos tradicionales.

Se pretende continuar trabajando sobre métodos que permitan la simplificación de varias funciones simultáneas y garanticen a la vez redes libres de riesgos por tiempo.

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

Es igualmente de interés métodos que superen los retos computacionales que plantean algoritmos como el presentado en este artículo, y que se ven afectados en el tiempo de ejecución a medida que el número de variables crece debido a la inherente búsqueda exhaustiva de implicantes primos. Aunque en la práctica, al igual que en el ejemplo planteado, es común ver funciones con menos mintérminos que posibles combinaciones, solamente una función de cinco variables con todos sus mintérminos conlleva a problemas de cómputo, así la solución sea tan trivial como 1 lógico.

6. CONCLUSIONES Y RECOMENDACIONES

Métodos tradicionales de simplificación de funciones de conmutación, tales como los mapas de Karnaugh o simplificaciones mediante aplicación iterativa de teoremas del álgebra de Boole, son herramientas clásicas que siempre conducen a soluciones mínimas para una única función, además éstos métodos son difícilmente aplicables a funciones con más de cinco variables; en el caso de los mapas de Karnaugh su representación gráfica presenta dificultadas de abstracción y los métodos de simplificación por álgebra podrían no asegurar una solución mínima. Sin embargo, en el campo práctico las implementaciones comunes conllevan siempre varias funciones con varias variables, lo cual debe ser afrontado desde un punto de vista diferente.

El algoritmo de Quine-McCluskey es uno de los más reconocidos gracias a su aplicabilidad a la simplificación de varias funciones y además permite encontrar una solución mínima global, aunque presenta situaciones heurísticas en su punto 6 (ver Sección 2).

La solución al problema heurístico de Quine-McCluskey es resuelta por Petrick mediante el algoritmo que lleva su nombre y el cual realiza una búsqueda completa de la solución mínima entre los implicantes primos necesarios para terminar de cubrir los mintérminos no cubiertos por los implicantes primos esenciales.

Los dos párrafos anteriores justifican la elección por parte de los autores de estos métodos como algoritmos de base en la implementación programada realizada. Además la selección de implementación en un lenguaje gráfico entrega ventajas didácticas que no sería fácilmente explorables en lenguajes textuales tradicionales, ya que se facilita el seguimiento al flujo de la información, comprensión de las diferentes etapas de los algoritmos, especialmente por parte de alumnos de pregrado, así como la mantenibilidad y optimización del mismo algoritmo implementado.

Los resultados de la implementación muestran el poder de simplificación global del programa montado, donde el número total de elementos de circuito es menor a una realización con métodos que sólo garantizan mínimos por función. Aunque ciertas situaciones en términos que se pueden reutilizar pueden ser circunstanciales de un conjunto de funciones dadas, lo que sí se puede garantizar es que el número total de términos producto a implementar gracias a los algoritmos programados es menor, y en el peor de los casos igual, a los obtenidos por métodos tradicionales; lo cual conlleva necesariamente a una implementación mínima global.

Se debe abordar temas adicionales como requerimientos de redes libres de riesgo por tiempo o esfuerzos computacionales de los métodos programados, los cuales son fuente de interés y motivación para los autores en futuros trabajos.

7. BIBLIOGRAFÍA

[1] M. Karnaugh, The map method for synthesis of combinational logic circuits. AIEE Comm electronics, 1953.
[2] E. W. Veitch, A chart method for simplifying truth functions. Proc Computing Machinery Conf, 1952.
[3] W. V. Quine, “The problem of simplifying truth functions,” American Mathematical Monthly, vol. 59, no. 8, pp. 521–531, 1952.
[4] E. J. McCluskey, “Minimization of boolean functions,” Ph.D. dissertation, M.I.T., 1956
[5] E. J. McCluskey, Introduction to the theory of switching circuits. New York, McGraw-Hill Books, 1965.
[6] S. R. Petrick, Ed., On the minimization of boolean functions. Symposium on Switching theory, 1959.
[7] Floyd Thomas L. Fundamentos de sistemas digitales, 7ª Edición. Prentice Hall, 2000.
[8] John Wakerly. Diseño Digital: Principios y prácticas, 3ra Edición. Pearson Educación, 2001.
[9] Victor P. Nelson, Nagle H. Troy, Carroll Bill, Irwin J. David. Análisis y diseño de circuitos lógicos digitales. Prentice Hall, 1996.
[10] Tinder Richard. Engineering digital design, Second Edition Revised. Academic Press, 2000.
[11] M. Morris Mano. Diseño Digital 3ra ed., Pearson, Prentice Hall, 1995.
[12] R. L Tokheim. Principios digitales, 3ra Edición. McGraw-Hill, 1995.
[13] E. Mandado. Sistemas electrónicos digitales, 7ma Edición. Marcombo Boixereu Editores, 2004.
[14] Ronald J. Tocci. Sistemas Digitales: Principios y aplicaciones. Prentice Hall, 2007.