Programa Computacional para Síntesis de Circuitos Lógicos Combinacionales

Computer Program for Combinational Logic Circuits Synthesis

Guillermo Tejada Muñoz[1]

Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú

Resumen— Este trabajo presenta los resultados de un programa para minimizar funciones lógicas de hasta seis variables basado en el algoritmo de Quine-McCluskey, el programa ha sido codificado en lenguaje C++. Se muestran algunos de los resultados, los cuales han sido validados al compararlos con los resultados obtenidos con los mapas de Karnaugh.

Abstract— This paper presents the results of a program to minimize logic functions of up to six variables based on the algorithm of Quine-McCluskey, the program has been coded in C + +. Show some results, which have been validated by comparison with results obtained with Karnaugh maps.

Palabras clave— Método tabular, método numérico, Quine-McKluskey, programa computacional, síntesis de circuitos lógicos, minimización de circuitos lógicos.

Key words— Tabular method, numerical method, Quine-McKluskey, computer program, synthesis of logic circuits, minimization of logic circuits.

I. Introducción

En la síntesis de circuitos lógicos se emplea principalmente dos métodos de minimización, uno es llamado mapa de Karnaught, el cual es ejecutado visualmente, y el otro es llamado algoritmo Quine–McCluskey, el cual es ejecutado mediante un procedimiento basado en tablas y por eso también es llamado método tabular. Ambos métodos proporcionan resultados idénticos. Sin embargo, los mapas de Karnauhgt para funciones de más de 4 variables resultan ser gráficos muy complicados de analizar. El método tabular, en cambio, para funciones lógicas de más de 4 variables, no presenta la dificultad de los mapas de Karnaugth, aunque al resolverlo manualmente su tiempo de resolución crece exponencialmente con el aumento del número de variables. Sin embargo, el método describe un algoritmo que puede ser adaptado a un programa de computador, tal como ha sido ralizado en el presente trabajo.

A. Algoritmo Quine–McCluskey

El Algoritmo Quine–McCluskey es un método de simplificación de funciones booleanas desarrollado por Willard Van Orman Quine y Edward J. McCluskey. Es funcionalmente idéntico a la utilización del mapa de Karnaugh, pero su forma tabular lo hace más eficiente para su implementación en lenguajes computacionales, y provee un método determinístico de conseguir la mínima expresión de una función booleana. El método consta de dos pasos [1]:

- Primer paso: Encontrar todos los Implicantes Primos de la función.
- Segundo paso: Encontrar los Implicantes Primos Esenciales, los cuales son necesarios y suficientes para generar la función.

Así, por ejemplo, sea la función:

$$F(A,B,C,D) = \sum m(1, 4, 6, 7, 8, 9, 10, 11, 15)$$

1) Primer paso: Encontrar Implicantes e Implicantes Primos

Los Mintérminos se escriben en binario y se les agrupa de acuerdo al número de "unos" que contienen. En la Tabla I, al lado izquierdo se muestra el agrupamiento de los Mintérminos, así por ejemplo los minterminos 1, 4 y 8 forman una de las agrupaciones porque su número en binario contienen un solo bit. Luego se combinan los Mintérminos de los grupos adyacentes entre sí, considerando siempre que el segundo Mintérmino debe ser mayor que el primero y la diferencia entre ambos es de un solo bit, el cual en la combinación debe reemplazarse por un guión "-", lo que significa que la variable de esa posición se debe eliminar, los Mintérminos o Implicantes al combinarse se deben marcar con un aspa. En agrupaciones de Implicantes de 4 Mintérminos se encontrarán pares de Implicantes repetidos, uno de los cuales debe ser eliminado. Del mismo modo, en agrupaciones de 8

[1] Guillermo Tejada Muñoz. E-mail: gtejadam@unmsm.edu.pe

Minterminos habrán 3 repetidos, dos de los cuales deben ser eliminados y así sucesivamente [2].

TABLA I
IMPLICANTES DE LA FUNCIÓN

Mintérminos						Implicantes de 2 Mintérminos					
	A	B	C	D			A	B	C	D	
1	0	0	0	1	✓	1,9	-	0	0	1	
4	0	1	0	0	✓	4,6	0	1	-	0	
8	1	0	0	0	✓	8,9	1	0	0	-	✓
6	0	1	1	0	✓	8,10	1	0	-	0	✓
9	1	0	0	1	✓	6,7	0	1	1	-	
10	1	0	1	0	✓	9,11	1	0	-	1	✓
7	0	1	1	1	✓	10,11	1	0	1	-	✓
11	1	0	1	1	✓	7,15	-	1	1	1	
15	1	1	1	1	✓	11,15	1	-	1	1	

Implicantes de 4 Mintérminos				
	A	B	C	D
8,9,10,11	1	0	-	-
~~8,10,9,11~~	~~1~~	~~0~~	~~-~~	~~-~~

Los Implicantes sin marcar ya no pueden combinarse más y construimos con ellos la tabla de Implicantes Primos, tal como se muestra en la Tabla II.

TABLA II
IMPLICANTES PRIMOS

	A	B	C	D	Términos
1,9	-	0	0	1	$\overline{B}\ \overline{C}\ D$
4,6	0	1	-	0	$\overline{A}\ B\ \overline{D}$
6,7	0	1	1	-	$\overline{A}\ B\ C$
7,15	-	1	1	1	$B\ C\ D$
11,15	1	-	1	1	$A\ C\ D$
8,9,10,11	1	0	-	-	$A\ \overline{B}$

2) Segundo paso: Encontrar Implicantes Primos Esenciales

No todos los Implicantes Primos son parte de la función, debemos seleccionar a los Primos Esenciales, es decir aquellos Implicantes que sólo cubren a un único Mintérmino, se deben marcar con un aspa a los Primos Esenciales y encerrar con un círculo el Mintérmino cubierto y a todos los demás que puede cubrir, tal como se muestra en la Tabla III. Para los Mintérminos no cubiertos (7 y 15, en el ejemplo) se construye la Tabla IV y se listan todos los Implicantes restantes que los pueden cubrir, debemos seleccionar aquel Implicante que cubra la mayoría de los Minterminos no cubiertos (en nuestro ejemplo seleccionamos el Implicante 7, 15).

TABLA III
SELECCIÓN DE IMPLICANTES PRIMOS ESENCIALES

			Mintérminos							
		Términos	①	④	⑥	7	⑧	⑨	⑩	⑪	15
1,9	✓	$\overline{B}\ \overline{C}\ D$	X				X				
4,6	✓	$\overline{A}\ B\ \overline{D}$		X	X						
6,7		$\overline{A}\ B\ C$			X	X					
7,15		$B\ C\ D$				X					X
11,15		$A\ C\ D$								X	X
8,9,10,11	✓	$A\ \overline{B}$					X	X	X	X	
			✓	✓			✓		✓		

TABLA IV
PRIMOS ESENCIALES ALTERNOS

		Mintérminos	
	Términos	⑦	⑮
6,7	$\overline{A}\ B\ C$	X	
7,15 ✓	$B\ C\ D$	X	X
11,15	$A\ C\ D$		X

Finalmente, todos los Implicantes Esenciales que son parte de la función son los siguientes:

$$F = (1, 9) + (4, 6) + (7, 15) + (8, 9, 10, 11)$$

Es decir:

$$F = \overline{B}\ \overline{C}\ D + \overline{A}\ B\ \overline{D} + B\ C\ D + A\ \overline{B}$$

II. METODOLOGÍA

Se ha investigado si existen trabajos similares y encontrado en la web algunos; así, por ejemplo, en [3] se da acceso a un programa de gran aporte aunque bastante preliminar pues no llega a minimizar eficientemente a las funciones lógicas.

El algoritmo Quine-Mc. Klusky, el cual fue descrito en la Introducción, se ha convertido a un programa de cómputo descrito mediante los diagramas de flujo mostrados desde la Fig. 1 a la Fig. 4. En el diagrama de flujo de la Fig. 1, se detalla el programa en donde se pide al usuario que ingrese los datos de la función lógica, es decir el número de variables, el número de Mintérminos y los Mintérminos que consta la función. El programa, por divisiones sucesivas, convertirá a los Mintérminos de decimal a binario y guardará los bits binarios resultante en un arreglo de dos dimensiones, como máximo se tiene números binarios de 6 variables: ABCDEF.

El diagrama de flujo de la Fig. 2 es en realidad el compendio de cinco diagramas de flujo ligeramente diferentes que describen la combinación entre sí de Implicantes, el término "Implicantes de **n** Mintérminos", donde **n** puede tomar el valor de 1, 2, 4,

8, 16, significa que cuando **n** es igual a 1, se describe al diagrama de flujo que combina Implicantes de solo un Mintérmino. Si n=2 se describe a otro diagrama de flujo que asocia Implicantes de 2 Mintérminos y así sucesivamente. También **n** generaliza el nombre de los Arreglos, así por ejemplo AI(2n)MINT significa arreglos con los nombres de: AI2MINT, AI4MINT, AI8MINT, AI16MINT y AI32MINT. El programa al combinar dos Implicantes adyacentes elimina la variable en que difieren.

El diagrama de flujo de la Fig. 3 se encarga de la eliminación de Implicantes repetidos que se generan en los Arreglos AI4MINT, AI8MINT, AI16MINT, AI32MINT.

Los diagramas de Flujo de la Fig. 4 y Fig. 5 encuentran los Implicantes primos Esenciales. El diagrama de la Fig. 4, describe la selección de aquellos Implicantes Primos que sólo cubren a un Mintérmino respectivamente y son marcados como Implicantes Primos Esenciales, los Implicantes Primos son obtenidos del Arreglo AIP. El diagrama de la Fig. 5 describe la ejecución del programa que solo se realiza siempre y cuando haya Mintérminos que no hayan sido marcados en el paso previo. Si fuera el caso, se agrega como Implicante Primo Esencial aquel (aquellos) Implicantes Primos que no hayan sido marcados y que cubre(n) a los Mintérminos faltantes, los Implicantes Primos Esenciales se guardan en el arreglo AIPE.

Finalmente, el diagrama de flujo de la Fig. 6 presenta el resultado de la minimización, para ello cada posición binaria (1 o 0) del Arreglo AIPE se corresponde con una variable, en los casos en que para determinada posición no se ha colocado un número binario la variable de esa posición no debe ser representada (variable eliminada en el proceso de asociación de implicantes adyacentes).

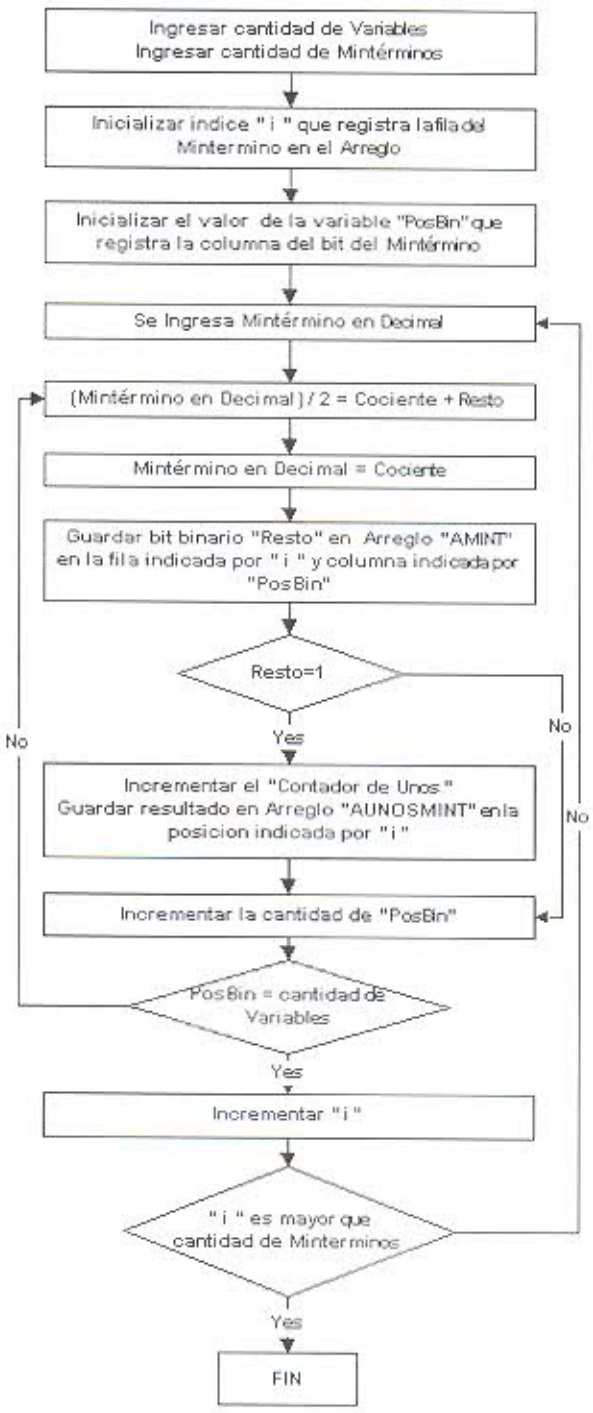


Fig.1. Ingreso de datos y conversión de Mintérminos de decimal a binario.

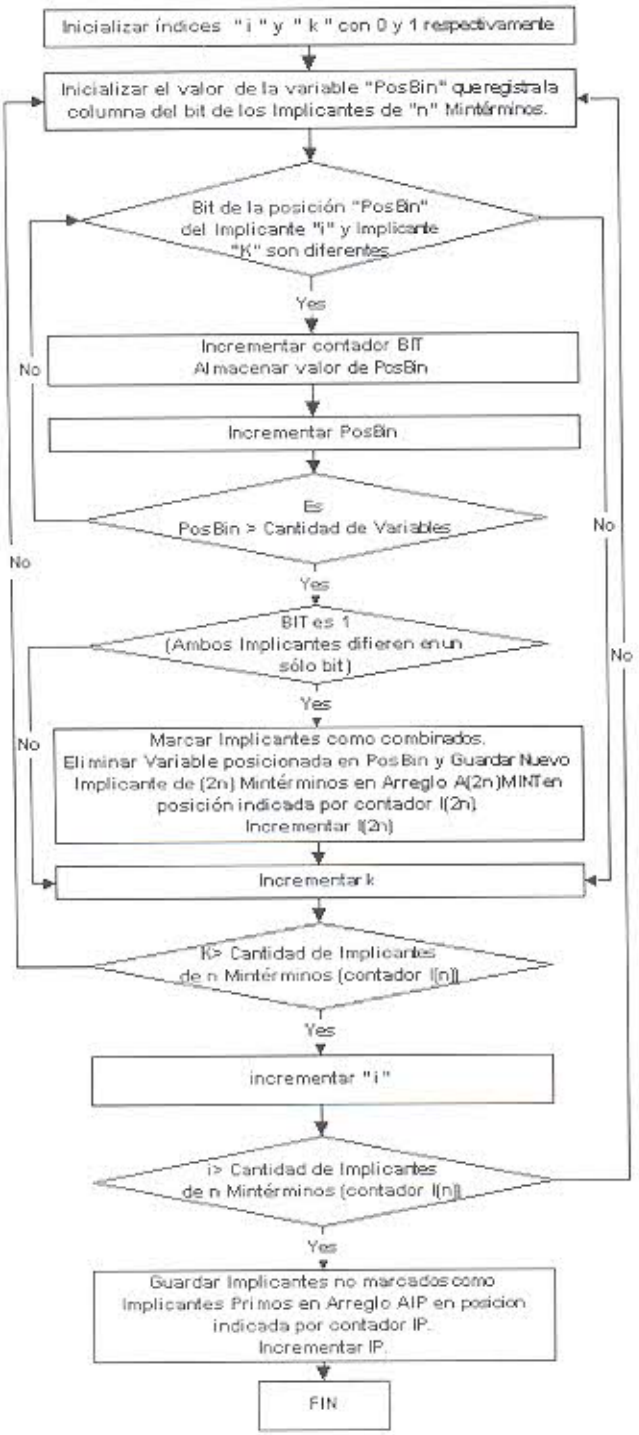


Fig. 2. Combinación o asociación de Implicantes.

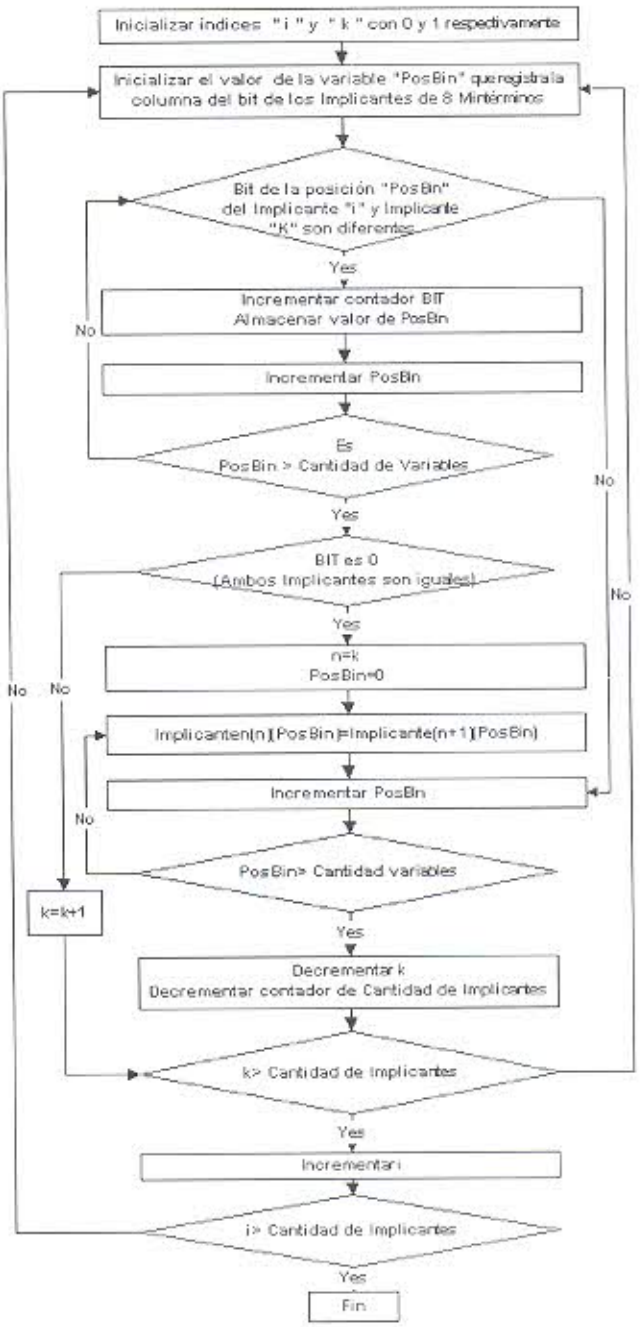


Fig. 3. Eliminación de Implicantes repetidos.

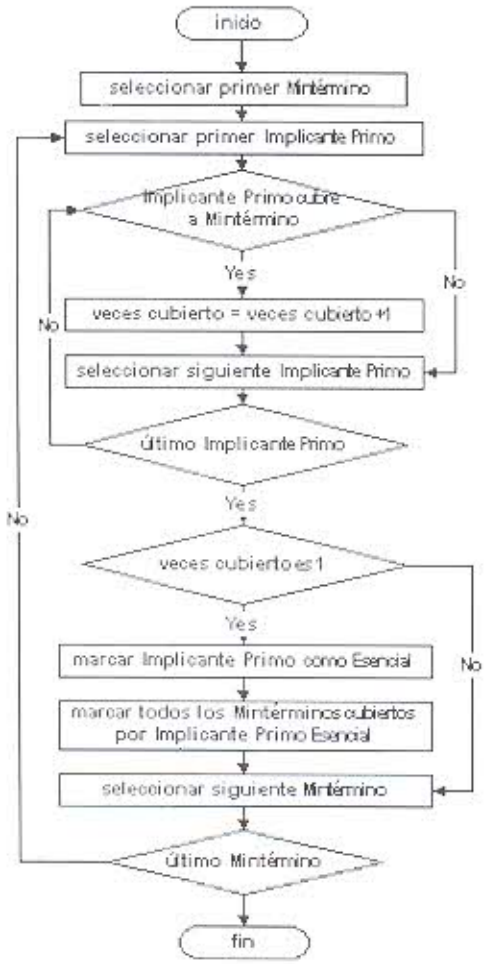


Fig. 4. Selección de Implicantes Primos Esenciales.

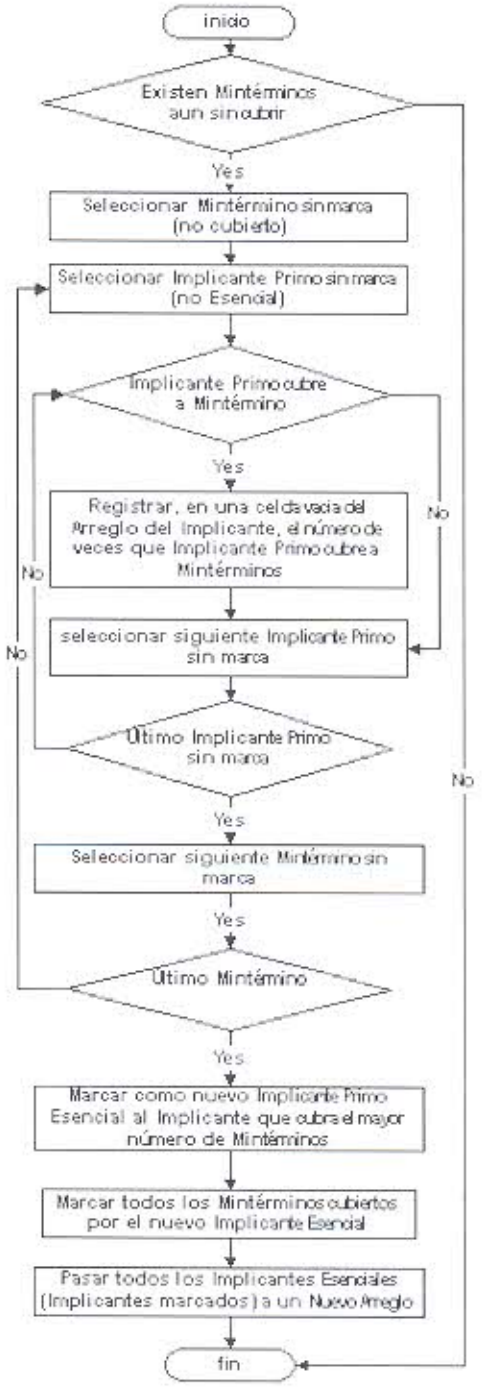


Fig. 5. Selección de Implicantes Primos Esenciales complementarios.

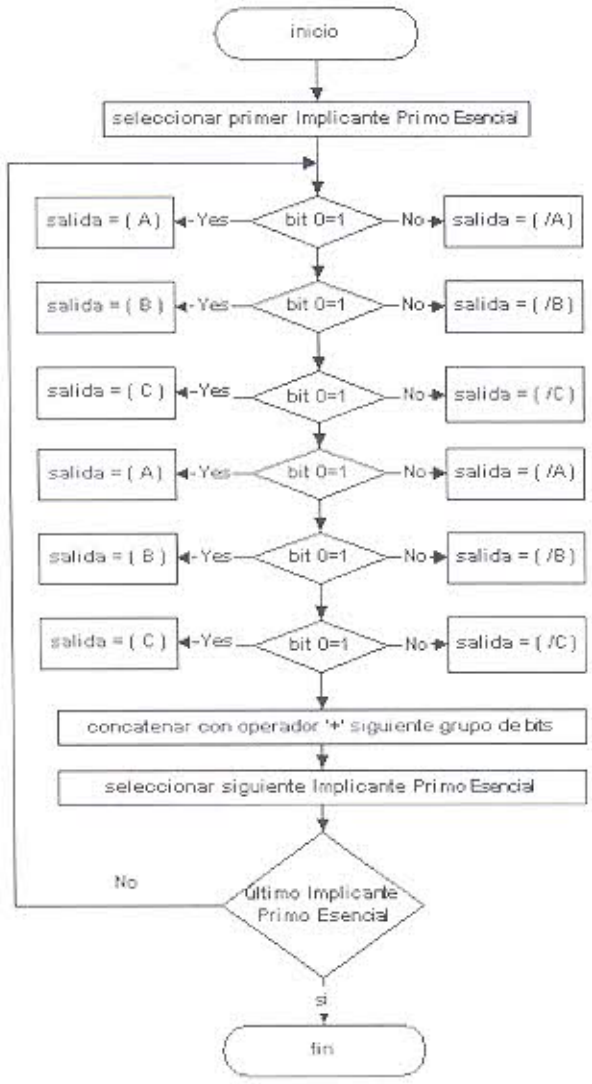
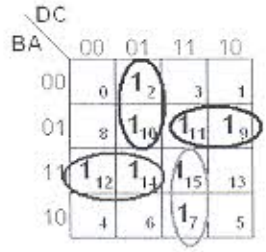


Fig. 6. Presentación de resultado.

El trabajó ha sido codificado en lenguaje C++ utilizando como herramienta de compilación al Borland C++ versión 5.01, se ha ejecutado en una computadora con CPU Pentium IV de 2.8 GHz y 1.25 Gb de memoria RAM. El programa ha sido probado con funciones lógicas de hasta 6 variables y validado comparando los resultados con los obtenidos de mapas de Karnaugh.

III. Resultados

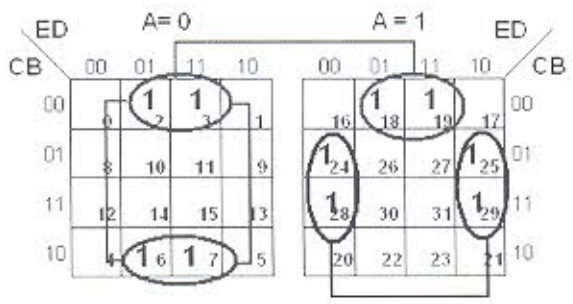
En las Figuras 7, 8 y 9 se muestran los resultados de minimizar funciones lógicas de cuatro, cinco y seis variables, respectivamente, utilizando Mapas de Karnaugh. Para estas mismas funciones en las figuras 10, 11, y 12 se muestran los resultados utilizando el programa computacional, se puede verificar que ambos resultados son idénticos. Se han realizado otras pruebas más complejas y en todos los casos se han obtenido los resultados esperados.



$$F = (m_2 + m_{10}) + (m_7 + m_{15}) + (m_9 + m_{11}) + (m_{12} + m_{14})$$

$$F = \bar{B}\,C\,\bar{D} + B\,C\,D + A\,\bar{B}\,D + A\,B\,\bar{D}$$

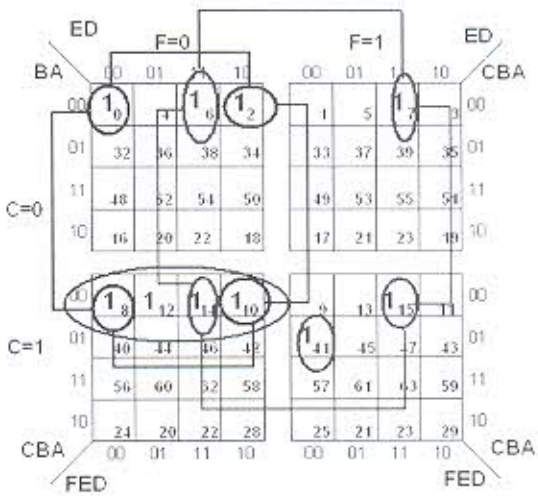
Fig. 7. Mapa de Karnaugh para una función de 4 variables



$$F = (m_2 + m_3 + m_6 + m_7) + (m_2 + m_3 + m_{18} + m_{19}) + (m_{24} + m_{25} + m_{28} + m_{29})$$

$$F = \bar{A}\,\bar{B}\,D + \bar{B}\,\bar{C}\,D + A\,B\,\bar{D}$$

Fig. 8 - Mapa de Karnaugh para una función de 5 variables.



$$F = (m_{41}) + (m_0 + m_2 + m_8 + m_{10}) + (m_6 + m_7 + m_{14} + m_{15}) + (m_8 + m_{10} + m_{12} + m_{14})$$

$$F = A\,\bar{B}\,C\,\bar{D}\,\bar{E}\,F + A\,\bar{B}\,\bar{D}\,F + \bar{A}\,\bar{B}\,D\,E + \bar{A}\,\bar{B}\,C\,\bar{F}$$

Fig. 9. Mapa de Karnaugh para una función de 6 variables.

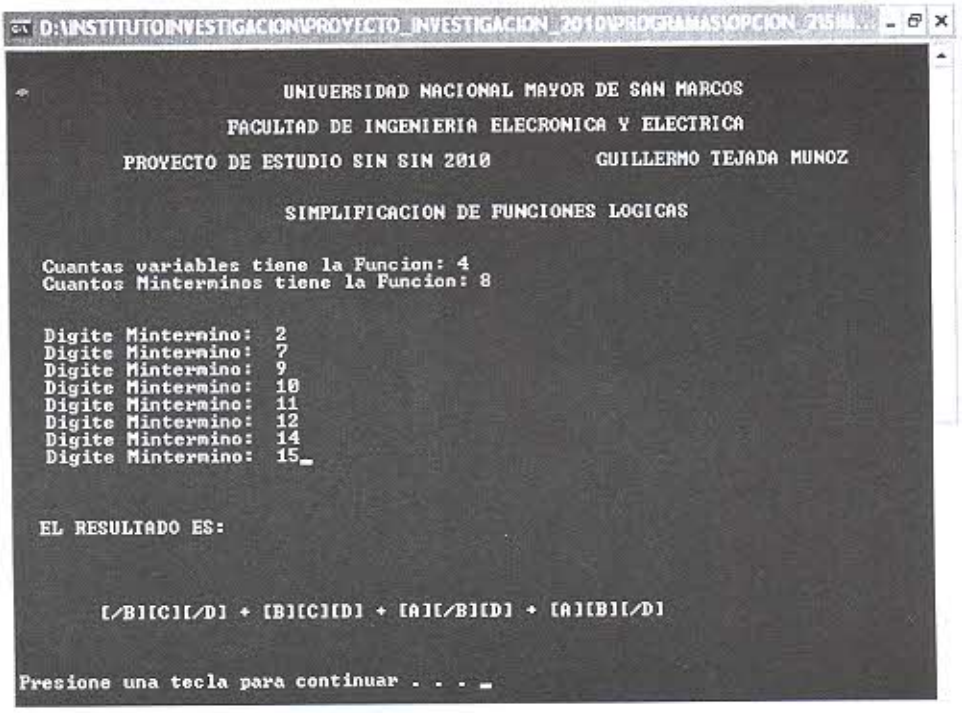


Fig. 10. Resultados del programa para la misma función de la Fig. 7.

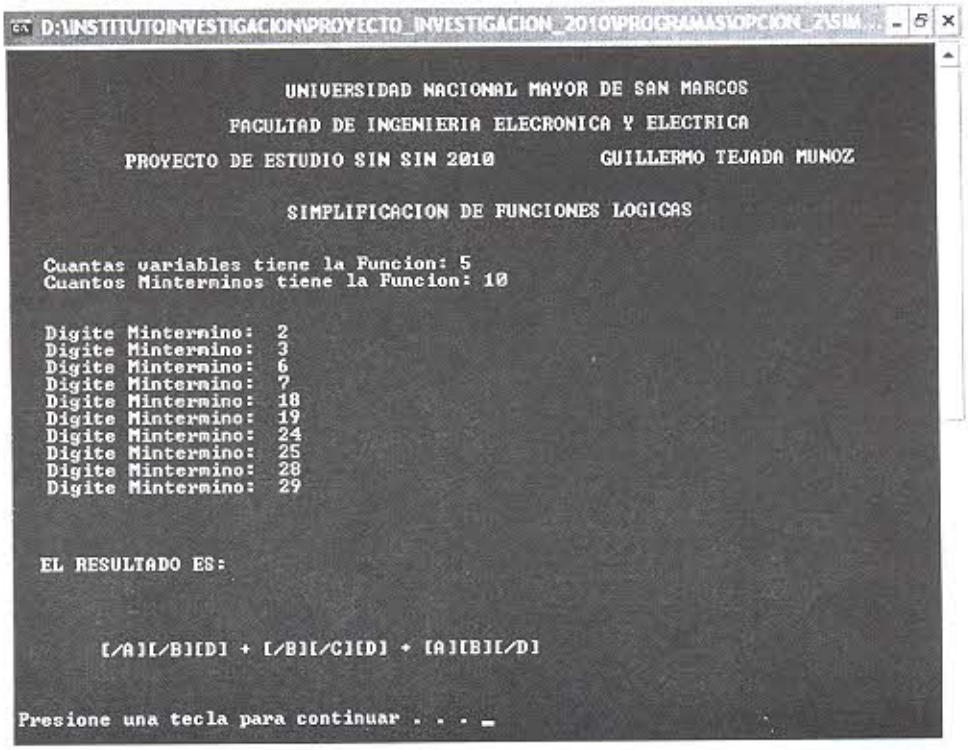


Fig. 11. Resultados del programa para la misma función de la Fig. 8.

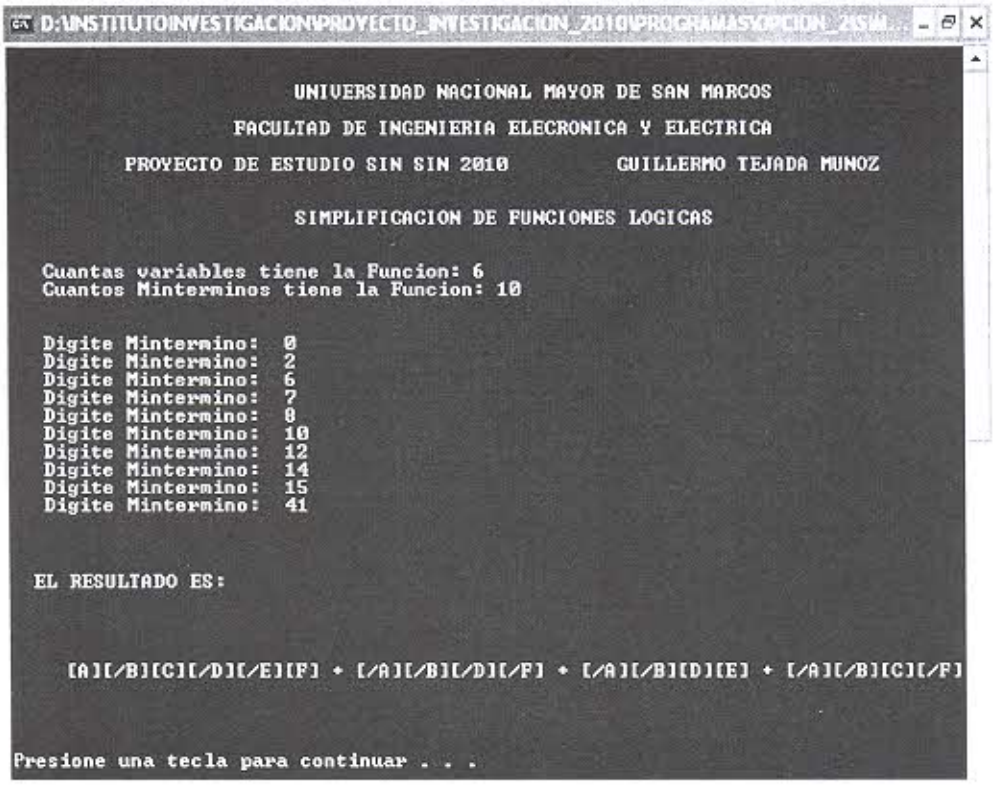


Fig. 12. Resultados del programa para la misma función de la Fig. 9.

IV. Conclusiones

Se ha elaborado un programa para minimización de funciones lógicas de hasta seis variables empleando el algoritmo Quine-McCluskey. El programa es flexible de modificaciones para minimizar funciones lógicas de mayor número de variables.

Referencias

[1] Wakerly, John F. *Digital Design Principles and Practices*. Pearson, 2001.

[2] Morris Mano. *Digital Design*. 3.ra edición. Prentice Hall, 2001.

[3] http://www.lawebdelprogramador.com/codigo/512/Mtodo_de_Quine-Mcclusky.html. Fecha de acceso: Abril 2010.