ISTANBUL UNIVERSITY ENGINEERING FACULTY
JOURNAL OF ELECTRICAL & ELECTRONICS

YEAR : 2002 (555-561)
VOLUME : 2
NUMBER : 2

AN EDUCATIONAL COMPUTER TOOL for SIMPLIFICATION of BOOLEAN FUNCTION's VIA PETRICK's METHOD

Necati ARSLAN[1] **Ahmet SERTBAS**[2]

[1,2] Istanbul University, Engineering Faculty, Computer Engineering Department
34850, Avcilar, Istanbul, TURKEY

[1]e-mail: necatiist@hotmail.com [2]e-mail: asertbas@istanbul.edu.tr

ABSTRACT

This paper reports on the simplification of Boolean functions problem encountered in the logic circuit design. The simplification process studied in this work is concentrated on the Petrick's Method, produce all solutions exactly rather then Quine McCluskey or Karnaugh Method. The Petrick's algorithm is described in detail. A specific example is worked through to explain the algorithm. On the example, all different possible solutions and "PLA" diagrams of these solutions are given as program outputs. Also, in this context, the comparisions of the methods used for simplification of Boolean functions are given.

Keywords: *Boolean Function, Two-level Minimization, Petrick's Method, Prime Implicant.*

1. INTRODUCTION

The logic design complexity is direct related to the complexity of the Boolean functions as the algebraic expression of the logic circuit outputs. The Boolean functions may be simplified by the alternative methods as given in the literature. The most known method called the Map Method, first proposed by Veitch[1] and slightly modified by Karnough[2], provides a simple straightforward procedure for minimizing Boolean functions. However, the map method is not convinent as long as the number of variables exceed five or six. In order to minimize the Boolean functions with many variables, the other method called as Tabulation Method which is step-by-step procedure was first formulated by Quine [3] and later improved by Mc Cluskey[4], also known as the Quine-McCluskey method. But it is quite boring and tedious work for human use.

The another two-level minimization method that is named of Petrick's method, provides the systematic way in the prime implicants selection, was improved. Instead of Quine-Mc Cluskey method, Petrick's method could give all alternative minimum solutions. Also, in Quine-Mc Cluskey method, defining the essential prime implicants is required to use the designer's intuition.

Received Date :10.2.2002
Accepted Date:12.5.2002

In this computer work, for the students take the logic design course and the logic circuit designer, an useful educational tool is improved by using Petrick method, the best minimization method explained above.

In order to develop Boolean functions and create digital circuitry, this computer tool is proposed. Even for very simple projects these functions can become quite complicated unless the students are able to simplify the expressions.

2. SIMPLIFICATION OF BOOLEAN FUNCTIONS

A boolean function is that it decribes an relation between variables and the function. If this relation is explained with the simplest way, the circuit desing would be more simple and its implementation more economic.

It is well known that, when a Boolean function is implemented with logic gates, each literal in the function designate an input to a gate and each term is implemented with a gate. The minimization of the number of literals and the number of terms (implicant) results in a circuit with less gates.

In order to begin any minimization, the goals have to be clearly defined. In our case, we make the following assumptions:

- Each implicant corresponds to a logic gate.
- The number of literals corresponds to the size of the logic gate used to implement the implicant

Therefore, we set our simplification goals as follows:

- The primary goal is to reduce the number of implicants.
- The secondary goal is to reduce the number of literals.

On the other hand, we know that reducing the number of gates is more important than reducing the size of a particular gate.

The main goal in the circuit design is to use less component, less bulk, less cost. Therefore a designer have to make the Boolean function the simplest. Some methods are described in the next chapter to simplify the functions.

3. THE SIMPLIFICATION METHODS

There are well established methods for doing these simplifications. Mainly, two methods are used in logic minimizations.

1. *Map Method*

The simplification method called Karnaugh maps are not easily used after the number of input variables exceeds four. Many circuits implemented in practice may require more than four input variables. Karnaugh Maps provide an alternative way of simplifying logic circuits. Instead of using Boolean algebra simplification techniques, you can transfer logic values from a Boolean statement or a truth table into a Karnaugh map. The arrangement of 0's and 1's within the map helps you to visualise the logic relationships between the variables and leads directly to a simplified Boolean statement.

Using K-maps to find the minimized two-level form of a function is difficult for functions of more than 4 variables and nearly impossible for functions of more than 6 variables because it is hard to visualize and spot patterns in multidimensional space[5] :

2. *Two-Level Logic Minimization Method*:

An alternative to use K-maps is Two Level Logic Minimization Methods solve this problem on two levels

1. Determination of Prime Implicants
2. Prime Implicant Chart

Quine-McCluskey and Petrick's methods can be grouped as the two-level logic minimization method. The methods classified into the group of two-level methods are similiar at determination of prime implicants, different at the approach of prime implicant chart with respect the others.

- *Quine-McCluskey Method*

Quine-McCluskey method reduces the minterm expansion of a function to obtain minimum sum of products form procedure has two steps[6] :

1. Eliminate as many literals as possible from each term by applying xy + xy' = x. The resulting terms are called prime implicants.

2. Use a prime implicant chart to select as minimum of prime implicants such that when they are all together it equals the function being simplified.

With both the K-map method and Quine-McCluskey algorithm you are trying to find a minimum number of terms that cover all of the minterms in the function[7]. For example, the two minterms AB'CD' and ABCD' are covered by the term ACD'

Although Quine-Mc Cluskey was the first methodical algorithm , it can not define essential prime implicants exactly[8]. We do that with Petrick's Method.

4. PETRICK'S METHOD

Petrick's method has two levels , as other two-level methods have [9];

1. Determination of Prime Implicants
In order to apply the Petrick's method to determine a minimum sum of products expression for a function , the function must be given as a sum of minterms. If the function is not in minterm form the minterm form must be found. In the first part of Petrick's method all of the prime implicants of a function are systematically formed by combining minterms. The minterms are represented in binary notation and combined if they differ in exactly one variable.

ABC + ABC' = AB
(111) (110) (11X)

In order to find all of the prime implicants all possible pairs of minterms should be compared and combined whenever possible. To reduce the required number of comparisons the binary minterms are sorted into groups according to the numbers of 1's in each term.

2. Prime Implicant Chart
We use prime implicant chart to select minimum number of prime implicants that covers all the minterms. We call these prime implicants essential prime implicants. There can be one or less covers that can do that. All of them are solutions.

We call choosing minimum number of prime implicants "covering problem"[10]. This problem has three replies according to simplificaton method. Quine-Mc Cluskey uses a heuristic methot to do that. Petrick's method uses a systematic method.

The prime implicant chart has minterms on its first row and prime implicants on its first column and check minterms included from prime implicants.

The petrick algorithm is shown below. Before this program is being run prime implicant are computed and put into the array "Komsular". The array "Minterm" includes minterms and don't-cares.

The Petrick's algorithm selects the minimum essential prime implicant covers and puts its essential prime implicant into the array "Arasonuç".

Algorithm:

1. Put the minterms and don't-cares into "Grup" array.

2. Compute the implicants and remove the covered minters from the "Grup" aray.

3. Put minterms except don't-cares into "Mintermler" array.

4. Select prime implicants and name them as "An" that includes a minterm from "komsular" array. "n" is prime implicant number.

   Example: A3 represents the third prime implicant in the "komºular" array.

   Repeat this for each minterm. Put the including prime implicants into the "Ptable" array on the same row for each minterm.

Table.1 Ptable of the example

1	A0	A1		
3	A2	A3		
4	A0	A1	A3	

Minterm "1" is including by PI's A0 and A1
A0 is the first PI in the Komsular array.
The solution must be like this after that step according to petrick method ;
P*=(A0+A1)(A2+A3)(A0+A1+A3)

5. Put the first row of Ptable into P array

6. Calculate P* was shown above.

7. Simplify P* with "Terimsadeleºtir" function. Remove the twin (or more) of each PI. At the end of this step we get the result of petrick method.

8. Put each result into the "Arasonuc" array. There can be more the one result.

9. Write the results to the result list of the program.

5. ALGORITHM EXAMPLE

To make the algorithm clearer, use it to simplify the following function:

$$F = SUM(0, 1, 2, 5, 6) \qquad (1)$$

To simplify this function with the program we use karnaugh map in order to input the function. Only click on the minterm to make it "1" ,"X" don't-care or "0".

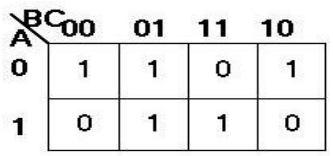


Figure 1. Input interface

Figure 2. Minterms and Don't-Cares window

Thus the function given in (1) is represented by the following list of minterms.

Table 2. The minterm list of the function

G	**PI**	A	**B**	**C**	
$\mathbf{G_0}$	0	0	0	0	✓
$\mathbf{G_1}$	1	0	0	1	✓
	2	0	1	0	✓
$\mathbf{G_2}$	5	1	0	1	✓
	6	1	1	0	✓

In the above list the terms in group 0 has zero "1", the terms in group 1 has one "1", the terms in group 2 has two "1".

Step1: (Creating 'Grup' Array)
Put this table into "Grup" array. We process the first step of Petrick's method on this array. Its dimentions are 500 row , "variable number+1" column and "variable number+1" table. For variable number "3" for this example Grup array's dimentions are 500 row , 4 column and 4 table. This table is defined as string.

Every row represents a minterm or implicant.

Every column represents a variable (A,B,C...).

Last column is used to represent the "checked" minterms.

Every table represents a step of determination prime implicants. First table must be like this;

Table 3. 'Grup' Array

A	B	C	Check
0	0	0	0
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	0

Check = 0 not checked
Check = 1 checked

Step2: (Combining Minterms)

Two terms can be combined if they differ in exactly one variable. Comparison of terms in non-adjacent groups is unnecessary since such terms will always differ in at least two variables and can not be combined using XY+XY'=X. Similarly Comparison of terms within a group is unnecessary since two terms with the same number of 1's must differ in at least two variables.

First we will compare the term in group 0 with all of group 1.

terms "000" and "001" can be combined to eleminate the third variable C which yields "00X".
terms "000" and "010" can be combined as "0X0"

Since comparison of groups 0 with groups 2 is unnecassary we proceed to compare terms in groups 1 and 2.

terms "001" and "101" can be combined as "X01"
terms "010" and "110" can be combined as "X10"

The program does these steps on "Grup" array. Compares each row (minterm) with others and uses "komsu" function to determine whether or not these minterms can combineable. If two minterms are combined two of them are checked off with "1" and combined prime put into grup array to the next table. Last a new table formed including primes.

Table 4. Komsular Aray

PI	**A**	**B**	**C**
PI1 0,1	0	0	X
PI2 0,2	0	X	0
PI3 1,5	X	0	1
PI4 2,6	X	1	0

Table 5. 'Grup' Array

A	B	C	Check
0	0	X	0
0	X	0	0
X	0	1	0
X	1	0	0

Even though two terms have already been combined with another terms they still must be compared and combined if possible. This is necessary since the resultant term may be needed to form the minimum sum solution.

But we can not combine any prime implicant in our example. Therefore the first, determination of prime implicants , step is finished.

In order to use prime implicants easly , put them Rom Grup array to "Komsular" array. It is like that;

Table 6. 'Komsular' array

A	B	C
0	0	X
0	X	0
X	0	1
X	1	0

We name prime implicants as PIn in order to use it easyly. n is row number.

A0=A'B'
A1=A'C'
A2=B'C
A3=BC'

Step3 : (Prime Implicant Chart)

Table 7. Prime Implicant Chart

PI	0	1	2	5	6
A0	X	X			
A1	X		X		
A2		X		X	
A3			X		X

We represent this table on cumputer with "Ptable" array. It has $2^{\text{variable number}}$-1 rows and 50 columns. Every row represents a minterm without don't- cares , every column represents a implicant that includes the rows minterm. Ptable must be like this;

Table 8. "Ptable" array

A0	A1	0
A0	A2	1
A1	A3	2
A2		5
A3		6

Form a P function;

1) for each minterm write a sum of prime implicants that includes this minterm.

For "0" (A0+ A1)
For "1" (A0+ A2)
For "2" (A2+ A3)
For "5" (A2)
For "6" (A3)

2) product these sums. We call this product of sums **pos**

P= (A0+ A2)(A0+ A3)(A2+ A4)(A3)(A4)

Step4: (Expanding P into SOP)

Apply (X+Y)(X+Z)=X+YZ and / or X+XY=X to reduce a minimum sop expression of P. In order to that step we use a auxiliary "P" array with 500 rows and 1000 columns. First we put the first row of Ptable to that array then multiply each row of Ptable Of this , first , row and write the results to a new row. At the last row , we have the result. But this result must be simplified with "terimsadeleºtir" function. Last we have the exact result after we find the minimum term(s). Each term represents a result.

P = A0A3A4 + A2A3A4

Result: each term of P is a solution. But we take the term / terms with minimum number of variables as solution. Here there are two solutions.

Solution1: A0 A2 A3 means the solution is;
A'B'+B'C+BC'

Solution2: A1 A2 A3 means the solution is;
A'C'+B'C+BC'

Step5: (Show the Result)

In order to show the result on the program , we use "Arasonuc" array. Ýt includes each essential prime implicant (at the columns) of each solution (at the rows).

The result is shown on 4 companent;

I. Solutions list (a list of each solution by its EPIs)
II. EPS list that shows the selectted solution
III. Karnaugh Map shows the minterm with red forecolor that EPI included which is selected from the EPIs list
IV. PAL Diagram that shows the selected solution.

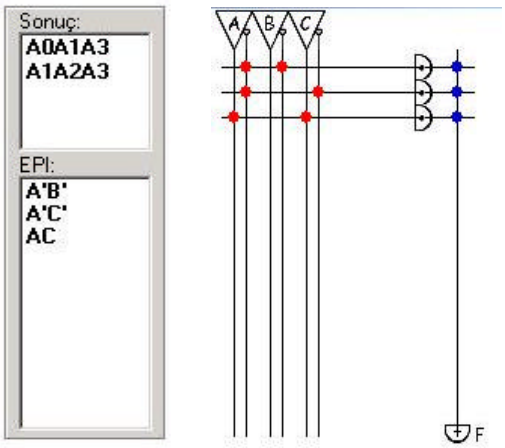


Figure 3. Solution, EPI lists and PAL diagram

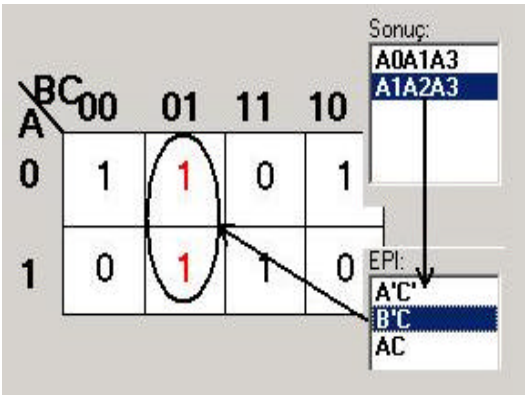


Figure 4. Showing the groups

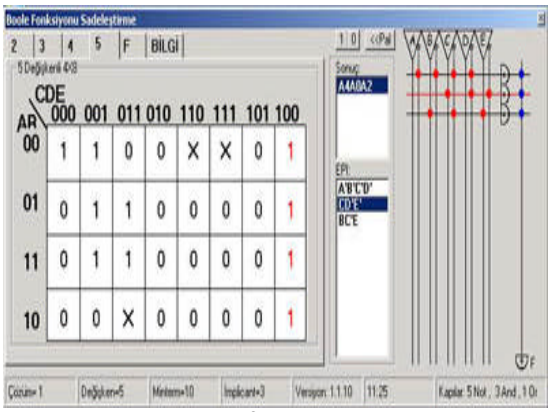
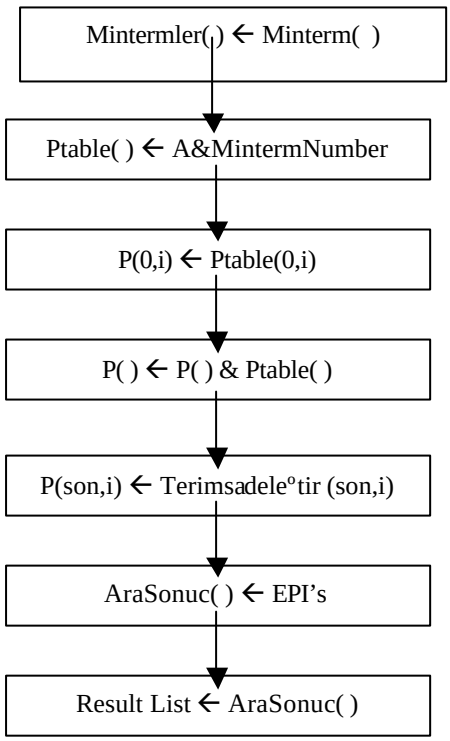


Figure 5. Final view of the program output

PETRICK ALGORITHM FLOW CHART



6. CONCLUSIONS

Petrick's method is the one that can define all of minimum solutions. This is especially important , because in Quine-Mc Cluskey method , it is often left to user to define essential prime implicants. Therefore the user sometimes, especially in complex problems with lots of prime implicants, can not define minimum solution or other alternative solutions. The error risk is very high.

Unlike Quine-Mc Cluskey method, Petrick's method can define all exact solutions correctly. It gives us all alternative minimum solutions. In Quine-Mc Cluskey method you use your intuition to define essential prime implicants. Therefore sometimes you can't define them. But in Petrick's method you do what the method says. Because this method is systematic but rather tedious.

Finally, in this work, a very useful computer design tool is developed for a graduate student takes the logic design course and a designer to have to realize the logic circuit quickly. Also, the tool can give the PAL implementation of the designed circuit.

REFERENCES

1) E.W. Veith, 'A Chart Method for Simplifiying Truth Functions.' , Proc. Of the ACM, pages 127-133, May 1952.
2) M. Karnaugh, 'A Map Method for Synthesis of Combinational Logic Circuits', Trans. AIEE, Comm. And Electronics, Vol.72, Part 1, pages 593-99, November 1953.
3) W.V. Quine,' The Problem of Simplifiying Truth Functions', Am. Math. Montly, Vol.59, No.8, 521-31, October 1952.
4) McCluskey, E.J. Jr., 'Minimization of Boolean Functions', Bell System Tech.J. , Vol.35, No.6, 1417-44, November 1956
5) http://www.dei.isep.ipp.pt/~acc/bfunc/
6) http://www.capcol.edu/faculty/andresho/ee304/quine.htm
7) http://users.ece.gatech.edu/~mooney/Courses/ECE3060/pdfs/quine-mccluskey.pdf
8) http://www.cs.columbia.edu/~cs4861/handouts/quine-mccluskey-handout/quine-mccluskey-handout.html
9) http://poppy.snu.ac.kr/~kchoi/class/lc_intro/two_level.pdf
10) http://users.ece.gatech.edu/~mooney/Courses/ECE3060/pdfs/exact2lev.pdf
11) http://www.necatiist.cjb.net
(for all documents and program)

Necati Arslan: He was born in 1980 in Çan Çanakkale. After the primary school he studied at Çan Ýbrahim Bodur Anatolian High School.. He was graduated from there in 1998 with 2nd degree. Then he studied at Istanbul University Computer Engineering department He was graduated there this year (2002).

Ahmet Sertbaº: See Vol.2, Number 1, page 415