Optimization of the Quine-McCluskey Method for the Minimization of the Boolean Expressions

Tarun Kumar Jain
M.N.N.I.T., Allahabad
tarun.mnnit@gmail.com

D. S. Kushwaha
M.N.N.I.T., Allahabad
dharkush@yahoo.com

A. K. Misra
M.N.N.I.T., Allahabad
arunk_misra@hotmail.com

Abstract

The basic principle in designing digital circuit hovers around reducing the required hardware thus reducing the cost too. To achieve this, we use Boolean expression that helps in obtaining minimum number of terms and does not contain any redundant pairs. The conventional methods for the minimization of the Boolean expressions are K-Map method and the Tabulation method. The minimized expressions are used to design digital circuits. Since K-Map method gets exceedingly complex when the number of the variable exceed six, hence Quine-McCluskey tabulation method scores over this and is widely used .In the following paper we present optimized Quine-McCluskey method that reduces the run time complexity of the algorithm by proposing an efficient algorithm for determination of Prime Implicants

1. Introduction

A binary variable may appear either in the normal form (x) or in its complement form (x’). If we have two binary variables x and y combined with AND operation. Since each variable can appear in either form, there are four possible combinations: x’y’, x’y, xy’ and xy. Each of these represents a minterm or a standard product. In a similar manner, ’n’ variables can be combined to from 2n minterms. Each minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the decimal equivalent of the binary number is a 0 and unprimed if 1.

Similarly, n variables form an OR term, with each variable being primed or unprimed providing 2n possible combinations, called maxterms or standard sums. Each maxterms is obtained from an OR term of the n variables, with each variable being primed if the corresponding bit of the decimal equivalent of the binary number is 1 and unprimed if a 0.

The sum of products is a Boolean expression containing AND terms, called products terms of one or more literals each. The sum denotes the ORing of these terms. A product of sums is a Boolean expression containing OR terms, called sum terms. Each term may have any number of literals. The product denotes the ANDing of these terms. Boolean functions are represented as a sum of minterms or a product of maxterms are said to be canonical form. [1]

2. Conventional methods of Minimization of Boolean Expressions

There are two methods for simplification of Boolean functions:-
1) Karnaugh Map Method. [2]
2) Quine-McCluskey Method [3]

2.1. Karnaugh Map Method

The map is a diagram made up of squares. Each square represents one minterm / maxterm. Since any Boolean expression can be represented as sum of products or product of sums, it follows that a Boolean expression is recognized graphically in the map from the area enclosed by those squares whose minterm / maxterm are included in the function. In fact, the map represents a visual diagram of all possible ways a function may be expressed in a standard form. By recognizing various patterns, the user can derive alternative algebraic expressions for the same function, from which we can select the simplest one. The simplest one is assumed to be the expression with minimum number of literals involved in the expression.

2.2. Limitations of Karnaugh Map Method

0-7695-3093-1/08 $25.00 © 2008 IEEE
DOI 10.1109/ICAS.2008.11

Karnaugh maps generally become more cluttered and hard to interpret when adding more variables. Karnaugh maps are useful for expressions having four, or fewer, variables. For more variables the map effectively becomes three dimensional and is difficult to interpret. [4]

Another disadvantage of the map method is the trial-and-error procedure which relies on the ability of the user to recognize certain patterns. For functions of more than four variables, it becomes increasingly more difficult to ensure that best selection has been made.

2.3. Quine-McCluskey method

The method was formulated by Quine and later improved by McCluskey, thus known as the Quine-McCluskey method. The Quine-McCluskey method is a tabulation method. [5] This method overcomes the limitation associated with K-Map method. The method is tedious for hand computation; the intent of the method is to provide algorithmic procedure for providing prime implicants. [6]

The tabular method makes repeated use of the law A + A'= 1. The Binary notation is used a variable in true form is denoted by 1, in inverted form by 0, and the absence of a variable by a dash (-).

Reduction of Boolean expressions by tabulation method involves two major activities

- Determination of Prime Implicants
- Selection of Essential Prime Implicants.

2.3.1. Determination of Prime Implicants: Prime Implicants are all the terms that are candidates for inclusion in the simplified function. The starting point of the tabulation method is the list of the minterms that specify the function. The first tabular operation is to find the prime-implicates by using a matching process. This process compares each minterm / maxterm with every other minterm. If the two minterms / maxterms differ in only one variable, that variable is removed and a term with one less literal is formed. This process is repeated for every minterm / maxterm until the exhaustive search is complete. The matching process cycle is repeated for the terms formed until a single pass through cycle yields no further elimination of literals. The remaining terms and all the terms that did not match during the process comprise of the prime implicants.

2.3.2. Selection of Essential Prime Implicants.

I. Construct the prime implicant table.

II. Reduce the prime implicant table by:

(a) Removing the essential prime implicants,

(b) Removing rows that dominate other rows,

(c) Removing columns dominated by other columns,

(d) Repeating Steps 2 (a)–(c) until no further reduction is possible.

III. If necessary, solve the remaining prime implicant table. [7]

2.4 Example: Quine-McCluskey method.

Minterms = Sum(0,1,2,3,9,11,12,13,14,15)

X1 X2 X3 X4	X1 X2 X3 X4	X1 X2 X3 X4
(0) 0 0 0 0 *	(0, 1) 0 0 0 – *	(0, 1, 2, 3) 0 0 – –
	(0, 2) 0 0 – 0 *	
(1) 0 0 0 1 *		(1, 3, 9, 11) – 0 – 1
(2) 0 0 1 0 *	(1, 3) 0 0 – 1 *	
	(1, 9) – 0 0 1 *	(9, 11, 13, 15) 1 – – 1
(3) 0 0 1 1 *	(2, 3) 0 0 1 – *	(12, 13, 14, 15) 1 1 – –
(9) 1 0 0 1 *		
(12) 1 1 0 0 *	(3, 11) – 0 1 1 *	
	(9, 11) 1 0 – 1 *	
(11) 1 0 1 1 *	(9, 13) 1 – 0 1 *	
(13) 1 1 0 1 *	(12, 13) 1 1 0 – *	
(14) 1 1 1 0 *	(12, 14) 1 1 – 0 *	
(15) 1 1 1 1 *	(11, 15) 1 – 1 1 *	
	(13, 15) 1 1 – 1 *	
	(14, 15) 1 1 1 – *	

* Is used to mark the elements that have taken part in combining with some other element.

Since each of the rows of the two other tables could be combined with at least one other row, only the rows of this final table correspond to the prime implicants. Hence, the disjunctive normal form consisting of the prime implicants is given by

L = A!B!+B!D+AD+AB

In the next step, the set of prime implicants is reduced by following the earlier stated procedure:

L = A!B!+B!D+AB

2.5. Limitations

The Quine-McCluskey algorithm has its practical limits too. Both the K-map method and the Quine-McCluskey algorithm find the guaranteed two-level minimized form of a function. The Quine-McCluskey algorithm has its practical limits also because the algorithm is NP-Complete. In other words, the runtime of the Quine-McCluskey algorithm grows exponentially with the input size.

3. Optimized Quine-McCluskey method

In our new approach, we introduce the concept of Reduced Mask (R.M.). We use the pair of Reduced Mask and the Term Value. Initially all the Reduced Masks are initialized to zero. The Reduced Mask has bits set corresponding to the literal reduced. The two terms can combine only when the have the same Reduced Mask. If the Reduced Mask is the same, then if the X-OR (Exclusive-OR) of the Term Values of the two terms is an integral power of two, then the two terms from a pair.

3.1. How complexity is reduced

The Quine-McCluskey method uses three states 0, 1 and – respectively. Since computers have only two states 0 and 1. So we require at least two bits to represent these three states. The computers work on integers (set of bits) rather on a single bit. The only possible meaning representation for the three states can be made only by using a separate integer for each variable or by using bit fields in integers.

The first method requires N integer comparisons one for each literal. Second method can be implemented programmatically using masking techniques but there to we have increased number for the comparisons because of looping (N comparisons) and for comparing the result of masking operation (N comparisons one for each variable).

While, in our method we require only 2 comparisons each of this is an integer comparison. One comparison is required for comparing Reduced Mask and one for testing the Term Values.

Since two terms can combine only when they have the same literal reduced in the terms combining and as the Reduced Mask has bits set for the reduced literals only, those terms that have the same value of the Reduced Mask can combine. This is the necessary but not the sufficient condition.

A’C’D + A’CD= A’D’ (Both will have the reduced mask value 4 as B is reduced in both the terms). Instead of N comparisons (N= Number of literals) we require only one comparison.

If the terms differ in only one bit, they combine. For finding out the number of bits that are different, we take the X-OR (Exclusive-OR) of the two terms (having the same Reduced Mask) and if it’s a power of two, it implies that the terms differ in only one bit. Since the X-OR is implemented in the hardware of the computer, it’s much faster and efficient than comparing literals of the terms, to find out if they differ by only one term. For finding out whether a number is a power of 2, we can AND (Logical AND) the (Number) AND (Number-1). If this is equal to zero, then the number is integral power of two.

Hence Run Time complexity for determining the prime implicates is reduced.

3.2. Example

Minterms = Sum(0,1,2,3,9,11,12,13,14,15)

Term Value	R. M.	Term Value	R. M.	Term Value	R M.
(0) 0 *	0	(0,1) 0 *	1	(0,1,2,3) 0	3
		(0,2) 0 *	2		
(1) 1 *	0			(1,3,9,11) 1	10
(2) 2 *	0	(1,3) 1 *	2		
		(1,9) 1 *	8	(9,11,13,15) 9	6
(3) 3 *	0	(2,3) 2 *	1	(12,13,14,15) 12	3
(9) 9 *	0				
(12) 12 *	0	(3,11) 3 *	8		
		(9,11) 9 *	2		
(11) 11 *	0	(9,13) 9 *	4		
(13) 13 *	0	(12,13) 12 *	1		
(14) 14 *	0	(12,14) 12 *	2		
(15) 15 *	0	(11,15) 11 *	4		
		(13,15) 13 *	2		
		(14,15) 14 *	1		

* Is used to mark the elements that have taken part in combining with some other element.

Don't care conditions are introduced in the matching process like the minterms / maxterms. While determining the essential prime implicants only minterms / maxterms are considered. The don't care terms are not essential, hence not included in the table for determining the essential prime implicants.

For checking if the two terms can combine, Quine-McCluskey method requires a minimum of N comparison whereas the Optimized Quine-McCluskey method requires only two comparisons. From this observation we can easily conclude that with the increase in the number of literals involved in the expression, the time required for finding out the set of prime implicants is reduced by same factor.

4. Result

Table 1. Run time for the Quine-McCluskey method and the Optimized Quine-McCluskey method with the Number of Literals

Number of Literals	Run Time Quine-McCluskey (ms)	Run Time Optimized Quine-McCluskey (ms)
7	109	31
8	703	93
9	5921	546
10	54140	4162
11	491250	34796

Table 1 shows the results obtained when optimization of Boolean expression with different number of literals was carried out by Quine-McCluskey method and our proposed optimized Quine-McCluskey method. The above algorithms were run on machine with 3.0 GHz processor and 512 DDR RAM and the operating system was Microsoft Windows XP Professional Edition.

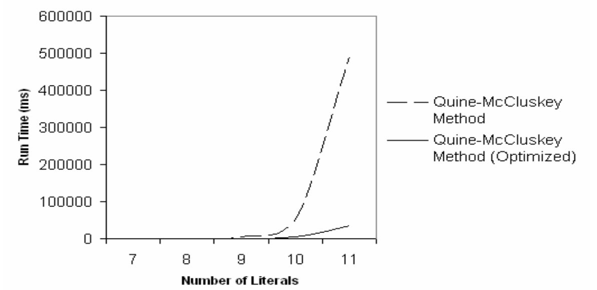

Figure 1. Run time for the Quine-McCluskey method and the Optimized Quine-McCluskey method with the Number of Literals

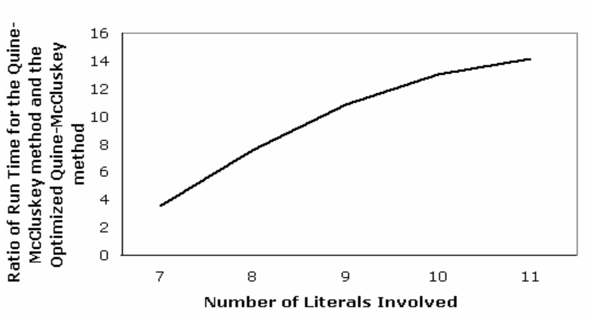

Figure 2. Ratio of Run time for the Quine-McCluskey method to the Optimized Quine-McCluskey method with the Number of Literals

Graph 1 clearly shows that there is manifold reduction in the run time for the Quine-McCluskey method. From Graph 2 it is clear that the ratio of Run time for the Quine-McCluskey method to the Optimized Quine-McCluskey method increase with the increase in the number of literals involved.

5. Conclusion

An effort has been made to present an optimized Quine-McCluskey method for determination of prime implicants that provides an optimal solution for reducing Boolean expressions. It is also able to establish that the proposed optimized Quine-McCluskey method reduces the run time complexity of the algorithm.

6. References

[1] M Morris Mano, Digital Logic and Computer Design, Prentice Hall of India 2004 Edition, pp 72-112.

[2] M Karnaugh, "The map method for synthesis of combinational logic circuits," Trans AIEE, Commiin & Electron, vol 72, no 1, pp 593-598, 1953.

[3] E J McCluskey, "Minimization of Boolean functions," Bell Syst Tech, J , vol 35, no 5, pp 1417-1444, 1956.

[4] Graham Wilson, "Embedded Systems and Computer Architecture", Newnes 2001, pp 25.

[5] Sivarama P. Dandamudi, "Fundamentals of Computer Organization and Design", Springer 2003 Edition, pp 67.

[6] Donald D. Givone, "Digital Principles and Design", McGraw-Hill Professional 2003 Edition, pp 173.

[7] Holger Schwender, "Minimization of Boolean Expressions Using Matrix Algebra" Collaborative Research Center SFB 475 University of Dortmund.