Mathematical & Computational Applications, Vol. 8, No. 3, pp. 271-278, 2003

BOOLEAN FUNCTIONS SIMPLIFICATION ALGORITHM OF O(n) COMPLEXITY

Şirzat KAHRAMANLI
Selcuk University
Department of Computer Engineering

Fatih BAŞÇİFTÇİ
Selcuk University
Department of Electronic and Computer Systems Education

Abstract – The minimization of Boolean functions allows designers to make use of fewer components, thus reducing the cost of particular system. All procedures for reducing either two-level or multilevel Boolean networks into prime and irredundant form have $O(2^n)$ complexity. Prime Implicants identification step can be computational impractical as n increases. Thus it is possible to get method in order to find the minimal set of Prime Implicants of $O(n)$ complexity instead of $O(2^n)$.
Keywords – Boolean function, simplification, minimization, Boolean expression, prime implicant, cube algebra, covering algorithm, simplification complexity, cube operations.

1. INTRODUCTION

The simplification of Boolean expression can lead to more effective computer programs and circuits. Minimizing expressions can be important because, electrical circuits consist of individual components that are implemented for each term or literal for a given expression. This allows designers to make use of fewer components, thus reducing the cost of a particular system. A wide variety of single output and multiple–outputs Boolean minimization techniques have been explain in [1,2]. Most of these techniques work on a two–step principle, the first step identifies all of the prime implicants (PI's) and the second step selects the subset of PI's that covers the function(s) being minimized [3].

The exact quantity of the end results of the identification process of all PI's can be calculated only in separate cases. In particular, if each prime implicant includes exactly l ones, l zeros and l don't care symbols, then the power of the complete set of PI is equal $M=(3\ l)!/(l!)^3$ [1, 4]. For example, for $l=1,2,3,4$ $M=6,90,1680$ and 34650, respectively. Since the number of PI's can be as large as $3^n/n$ for a function of n variables [3, 4]. Consequently the PI identification step can be computational impractical as n increases [3]. It is clear that all procedures for reducing either two-level or multilevel Boolean networks into prime and irredundant form have $O(2^n)$ complexity. [5, 6, 7, 8].In this paper, it is proposed the method of local determination of PI's that covers certain ON minterm of certain Boolean function. Since such minterm of n variables may be included in maximum n one-dimensional cubes, the power of temporary result cubes' set may not be overcome n [6]. Thus it is possible to get method in order to find the minimal set of PI's of $O(n)$ complexity instead of $O(2^n)$.

2. NOTATION

A multiple output Boolean function of n inputs and m outputs is defined as follows [9] : Input space : $B=\{0,1\}$; Output space : $Y=\{0,1,d\}$; Function $f: B^n \rightarrow Y^m$

The value *d* (don't cares) at output means that the value is unspecified, and a value of *0* or *1* will be accepted to implement this part of the function. Such a function can be represented by a list of PI's. Each PI contains an input part and an output part. Input part: *n* literals can be *{0,1,x}*; Output part: *m* literals can be *{0,1,d}*. The input part identifies the portion of the input space to which a cube (element) applies. The *x* in the input part matches all the points of the function that have either a *1* or a *0* for this variable (nonessential coordinate in cube interpretation).

In this paper it is developed a new simplification method for single-output Boolean functions, for which;

Input space: $B=\{0,1\}$; Output space: $Y=\{0,1,d\}$; Function $f: B^n \rightarrow Y$,
S_{ON} : The set of ON minterms any of that make the function equal to *1*,
S_{OFF} : The set of OFF minterms any of that make the function equal to *0*,
S_{DS} : The set of don't care minterms.

Algorithm proposed in this paper used the set S_{ON} and the set S_{OFF} exactly, and the set S_{DS} no evidently.

3. CUBE ALGEBRA OPERATIONS USED IN LATERS

3.1. The Coordinate Subtraction Operation (Sharp Product)

The coordinate subtraction operation of cubes $A=a_1a_2...a_i...a_n$ and $B=b_1b_2...b_i...b_n$ is executed in two parts. In the first part, the subtraction vector $SV=A \otimes B=v_1v_2...v_i...v_n$ is formed according to the following rules:

- If $b_i=x$ or $b_i=a_i$, then $v_i=Z$
- If $a_i=x$ and $b_i \neq x$, then $v_i=\bar{b}_i$
- If $a_i=\bar{b}_i$, then $v_i=Y$

a_i \ b_i	X	1	0
x	Z	0	1
1	Z	Z	Y
0	Z	Y	Z

In the second part, according to the coordinate values of *SV*, the result of the coordinate subtraction is defined as follows:

- If $\exists i$, $v_i=y$ then the subtraction operation is not possible, i.e. $C=A\#B=A$
- If no $\exists i$, $v_i=y$ exists and $v_j,..., v_k,...,v_m \in \{0,1\}$ exist, then the result of subtraction operation is the set $\{a_1a_2...a_{j-1}v_ja_{j+1}...a_n,\ a_1a_2...a_{k-1}v_ka_{k+1}...a_n,...,a_1a_2...a_{m-1}v_ma_{m+1}...a_n\}$
- If $\forall i$, $v_i=Z$ then the result of subtraction operation is empty, i.e. $C=A\#B=\varnothing$

3.2. The Commutative Absorption Operation (Δ Operation)

The commutative absorption operation of cubes $A=a_1a_2...a_i...a_n$ and $B=b_1b_2...b_i...b_n$ is executed in two parts also. In the first part, the vector absorption $AV=A \nabla B=v_1v_2...v_i...v_n$ is formed according to the following rules:

- If $a_i=b_i$, then $v_i=Z$
- If $a_i=x$ and $b_i\neq x$, then $v_i=G$
- If $a_i=\bar{b}_i$, then $v_i=Y$
- If $a_i\neq x$ and $b_i=x$, then $v_i=L$

a_i \ b_i	x	1	0
x	Z	G	G
1	L	Z	Y
0	L	Y	Z

In the second part, according to the coordinate values of AV, the result of the absorption will be defined as follows:

- If $\exists i$, $v_i=y$ then the absorption operation is not possible, i.e. $C=A \Delta B=\{A,B\}$
- If $\forall i$, $v_i=Z$ then $A=B$, i.e $C=A \Delta B=A$
- If $\exists i$, $v_i=G$ and is not $\exists i$, $v_i=L$, then $C= A \Delta B=A$
- If $\exists i$, $v_i=L$ and not $\exists i$, $v_i=G$, then $C=A \Delta B=B$
- If $\exists i$, $v_i=G$ and $\exists i$, $v_i=L$, then the absorption operation is not possible, i.e. $C=A\Delta B=\{A,B\}$

Example:

1. A=x11x; B=x0x1 then AV=x1x0 ∇ x0x1=ZYLG; C={x1x0, x0x1}
2. A=x11x; B=x11x then AV=x11x ∇ x11x=ZZZZ; C=x11x
3. A=x1xx; B=x1x1 then AV=x1xx ∇ x1x1=ZZZG; C=x1xx
4. A=x011; B=xx1x then AV=x011 ∇ xx1x=ZLZL; C=xx1x
5. A=xx0x; B=1xxx then AV=xx0x ∇ 1xxx=GZLZ; C={xx0x, 1xxx}

4. LOCAL DEFINITION OF PRIME IMPLICANTS

The complexity of determination process of PI's may be reduced by separating the process of each of minterm from S_{ON} set by each of minterm from S_{OFF} set [10, 11]. The existing of this possibility affirms the following theorem.

Theorem : If $A=a_1a_2...a_j...a_n$ is any minterm from S_{ON} set and $B_i=b_1^ib_2^i...b_j^i...b_n^i$ is any minterm from S_{OFF} set then calculation of the PI's covered minterm a by procedure of $K_i=K_{i-1}\#B_i$, $i=1,2,...,m$, $K_0=xx...x$, it is necessary and sufficiently preserving only all such b_j^i, value of which is a logical invert to value of corresponding a_j.

Proof: Assume that $a_j=b_j^i$ for any coordinate j. Then $v_j=x\#b_j^i=\bar{a}_j$. Consequently, for $a_j=b_j^i$ the difference cube formed on pair (a_i, b_j^i) contains the coordinate $\bar{a}_j$ which shows that this cube does not contain the minterm A. In this case, to prevent the producing of cubes which does not cover minterm A, it is necessary to replace the b_j^i by the symbol x. For $a_j\neq x$ and $b_j^i=x$ the difference cube on coordinate j is not formed. Therefore the value of b_j^i is not changed. The version $a_j=x$ and $b_j^i\neq x$ is not possible since A is a minterm. If $a_i=\bar{b}_j^i$, then $v_i=x\#b_j^i=\bar{b}_j^i=a_j$. Therefore, in the case where $a_j=\bar{b}_j^i$, the difference cube formed on coordinate j certainly covered the minterm A.

Using this theorem the minterm $B_i=b_1^ib_2^i...b_j^i...b_n^i \in S_{OFF}$ can be transformed to cube $Q_i=q_1^iq_2^i...q_j^i...q_n^i$ by the following rules:

If $b_j^i=x$ then $q_j^i=x$; If $a_j=b_j^i$ then $q_j^i=x$; If $b_j^i=\bar{a}_j$ then $q_j^i=b_j^i$

5. NEAR-MINIMAL COVER ALGORITHM

This algorithm based on operation defined in part 3.1 and 3.2 and on rules obtained in part 4. The result of this algorithm will be one of the possible minimal form of simplified Boolean function. It may be most simplificant or not most simplificant. But this result may be sufficient in most practical applications where one or a few extra product terms (AND gates) are not important. Generally, the nearlest of obtained result to most minimal form depends on ordering of minterm in S_{ON} set. However obtaining an appropriate order is more difficult than obtaining an irregular PI set. The near-minimal cover algorithm for any given Boolean function with any ordering minterms as follows:

1. Put $I=0$,
2. Select the first minterm from S_{ON} set, mark it by λ and put $I=I+1$,
3. Transform one by one all of elements of S_{OFF} set according to rule denoted in section 4. Mark the result by $Q0$,
4. Apply the absorption operation to $Q0$. Mark the result by $Q1$,
5. Coordinate Subtract the set $Q1$ from the n dimensional full cube $xx...xx$. Where n the number of variables of simplifying Boolean function. Mark the result by SI,
6. Apply the Great or Less operation to the elements of SI set. Note that element α is greater than element β if the set of $S_{ON} \# \alpha$ is powerless than the set of $S_{ON} \# \beta$,
7. Remove all the powerless elements from SI. If the result is single element then mark it by EI. Otherwise select one of them and mark it by EI,
8. Put $S_{ON}=S_{ON}\#EI$, $SPI=SPI \cup EI$,
9. If $S_{ON} \neq \varnothing$ then Go to 2,
10. END.

Example:

Let $S_{ON1}=\{0000, 0001, 0100, 0101, 0110, 1000, 1010, 1110\}$, consequently, $\lambda 1=0000$
$S_{OFF}=\{0010, 0011, 0111, 1001, 1011, 1100, 1101, 1111\}$

1.1. Definition of sets $Q0.1$ and $Q1.1$

$\lambda 1=0000$

S_{OFF}	Q0.1	Cube Status	Q1.1
0010	xx1x	Prime	xx1x
0011	xx11	Absorbed by xx1x	
0111	x111	Absorbed by xx1x	
1001	1xx1	Prime	1xx1
1011	1x11	Absorbed by xx1x	
1100	11xx	Prime	11xx
1101	11x1	Absorbed by 11xx	
1111	1111	Absorbed by xx1x	

As can be seen from this table
$Q1.1=\{xx1x, 1xx1, 11xx\}$

1.2. Definition of cube set covered of the minterm 0000

$S1=xxxx\#Q1=((xxxx\#xx1x)\#1xx1)\#$
$11xx=(xx0x\#1xx1)\#11xx=\{0x0x, xx00\}\#$
$11xx=\{0x0x, 0x00, x000\}=\{0x0x, x000\}$

$S1=\{S1.1, S1.2\}=\{0x0x, x000\}$

1.3. Definition the greatest cube
$P1.1=S_{ON1}\#S1.1=S_{ON1}\#0x0x=\{0000,0001,0100,0101,0110,1000,1010,1110\}\#0x0x$
$=\{0110,1000,1010,1110\}$
$P1.2=S_{ON1}\#S1.2=S_{ON1}\#x000=\{0000, 0001, 0100, 0101, 0110, 1000, 1010, 1110\}\#x000$
$=\{0001, 0100, 0101, 0110, 1010, 1110\}$

As seen the set P1.1 is powerless (4 element) than the set P1.2 (6 element). Consequently cube *0x0x* is greater than cube *x000*. Thus;

$E1=0x0x;\ SPI=\{0x0x\}$

2.1. Definition of so far non covered part of S_{ON} set
$S_{ON2}=P1.1=\{0110,1000,1010,1110\}$. From here, $\lambda 2=0110$

$\lambda 2=0110$

S_{OFF}	Q0.2	Cube Status	Q1.2
0010	x0xx	Prime	x0xx
0011	x0x1	Absorbed by x0xx	
0111	xxx1	Prime	Xxx1
1001	1001	Absorbed by x0xx	
1011	10x1	Absorbed by x0xx	
1100	1x0x	Prime	1x0x
1101	1x01	Absorbed by xxx1	
1111	1xx1	Absorbed by xxx1	

As can be seen from this table
$Q1.2=\{x0xx,\ xxx1,1x0x\}$

2.2. Definition of cube set covered of the minterm *0110*

$S2=xxxx\#Q1=((xxxx\#x0xx)\#xxx1)\#$
$1x0x=(x1xx\#xxx1)\#1x0x$
$=(x1x0\#1x0x)=\{01x0,x110\}$

$S2=\{S2.1,S2.2\}=\{01x0,x110\}$

2.3. Definition the greatest cube
$P2.1=S_{ON2}\#S2.1=S_{ON2}\#01x0=\{0110,\ 1000,\ 1010,\ 1110\}\#01x0=\{1000,\ 1010,\ 1110\}$
$P2.2=S_{ON2}\#S2.2=S_{ON2}\#x110=\{0110,\ 1000,\ 1010,\ 1110\}\#x110=\{1000,1010\}$
So *P2.2* is powerless than *P2.1*, i.e. cube *x110* greater than cube *01x0*. Thus;

$E2=x110;\ SPI=\{0x0x,x110\}$

3.1. Definition of so far non covered part of S_{ON} set
$S_{ON3}=P2.2=\{1000,1010\}$, from here, $\lambda 3=1000$

$\lambda 3=1000$

S_{OFF}	Q0.3	Cube Status	Q1.3
0010	0x1x	Prime	0x1x
0011	0x11	Absorbed by 0x1x	
0111	0111	Absorbed by 0x01	
1001	xxx1	Prime	xxx1
1011	xx11	Absorbed by xxx1	
1100	x1xx	Prime	x1xx
1101	x1x1	Absorbed by xxx1	
1111	x111	Absorbed by xxx1	

As can be seen from this table
$Q1.3=\{0x1x,xxx1,x1xx\}$

3.2. Definition of cube set covered of the *1000* minterm.

$SI3=xxxx\#Q1=((xxxx\#0x1x)\#xxx1)\#$
$x1xx=(\{1xxx,xx0x\}\#xxx1)\#x1xx$
$=\{1xx0,xx00\}\#\ x1xx=\{10x0,x000\}$

$SI3=\{10x0,x000\}$

3.3. Definition the greatest cube
$P3.1=S_{ON3}\#10x0=\{1000,1010\}\#10x0=\varnothing$
$P3.2=S_{ON3}\#x000=\{1000,1010\}\#x000=\{1010\}$
So the $P3.1$ is powerless than $P3.2$, i.e. cube $10x0$ is greater than cube $x000$. Thus;

$E3=10x0;\ SPI=\{0xOx,\ x110,10x0\}$

4.1 Definition of so far non covered part of S_{ON} set
$S_{ON4}=P3.1=\varnothing$. Consequently, the simplification process is completed.

6. THE ASYMPTOTICAL ESTIMATION OF PRESENTED ALGORITHM

As mentioned above, a certain minterm of function of n variables may be included in maximum n one-dimensional cubes. Therefore, the power of temporary (intermediate) result cube set which is formed by near minimal cover algorithm may not be overcome number n. The truthful of this may be easily seen from the example mentioned above.

To estimate, the proposed algorithm let's compare it with world wide Quine-McCluskey method that consist of followings [1, 2]:

- Elements (minterms) of S_{ON} set grouped according to the number of *1*'s contained. This is done by grouping the minterms into $n+1$ subsets. The first zeros subset contains elements with no *1*'s, the first subset contains those elements that have only one *1*, the second subset contains elements with two *1*'s, the *n*th subset contains elements with n *1*'s in, accordingly. That is to say, the *i*th subset contains elements with i *1*'s in it. Therefore, the power of *i*th subset is defined as,
$$P_i = C_n^i$$
- All the minterms in one subset are compared with all the minterms of the next subset. For example, the minterms of second subset must be compared only with the minterms of third subset. Therefore the asymptotical quantity of comparison of *i*th set with *(i+1)*th set and the total asymptotical quantity of all these comparisons is defined as,
$$W_i = C_n^i \times C_n^{i+1} \qquad \text{and} \qquad WT = \sum_{i=0}^{n-1} C_n^i \times C_n^{i+1}, \quad \text{respectively.}$$
- A certain minterm from *i*th subset has *(n-i)* neighbors in *(i+1)*th subset. Therefore, the asymptotical number of non empty results of comparison of *i*th subset with *(i+1)*th subset and the total asymptotical quantity of this results is defined as,
$$R_i = (n-i) \times C_n^i \qquad \text{and} \qquad RT = \sum_{i=0}^{n-1} (n-i) \times C_n^i, \qquad \text{respectively.}$$

As can be seen from mentioned above, the asymptotical number of all comparisons and all of non empty results of the first stage of the Quine-McCluskey method is defined as,

$$WT = \sum_{i=0}^{n-1} C_n^i \times C_n^{i+1} \qquad \text{and} \qquad RT = \sum_{i=0}^{n-1} (n-i) \times C_n^i, \text{ respectively.}$$

In accordance with this formulation, the comparison of the near-minimal algorithm and of the Quine-McCluskey method for *1-20* variables is shown in the following table.

Table 1. Comparation of Complexity Near-Minimal Cover Algorithm with Quine-McCluskey Method

Number of Variables	Quine-McCluskey Method				Near-Minimal Cover Algorithm
	Total Temporary Results		Non Empty Temporary Results		Number of Temporary Results (O(n) Complexity)
	Asymptotical Number	$O(2^n)$ Complexity	Asymptotical Number	$O(2^n)$ Complexity	
1	1	$1*2^1$	1	$0{,}50*2^1$	1
2	4	$2*2^2$	4	$1{,}00*2^2$	2
3	15	$3*2^3$	12	$1{,}50*2^3$	3
4	56	$4*2^4$	32	$2{,}00*2^4$	4
5	210	$5*2^5$	80	$2{,}50*2^5$	5
6	792	$6*2^6$	187	$2{,}92*2^6$	6
7	3.003	$7*2^7$	414	$3{,}23*2^7$	7
8	11.440	$8*2^8$	893	$3{,}49*2^8$	8
9	43.758	$9*2^9$	1.930	$3{,}77*2^9$	9
10	167.960	$10*2^{10}$	4.246	$4{,}15*2^{10}$	10
11	646.646	$11*2^{11}$	9.516	$4{,}65*2^{11}$	11
12	2.496.144	$12*2^{12}$	21.542	$5{,}26*2^{12}$	12
13	9.657.700	$13*2^{13}$	48.764	$5{,}95*2^{13}$	13
14	37.442.160	$14*2^{14}$	109.581	$6{,}69*2^{14}$	14
15	145.422.675	$15*2^{15}$	243.554	$7{,}43*2^{15}$	15
16	565.722.720	$16*2^{16}$	534.891	$8{,}16*2^{16}$	16
17	2.203.961.430	$17*2^{17}$	1.161.526	$8{,}86*2^{17}$	17
18	8.597.496.600	$18*2^{18}$	2.497.440	$9{,}53*2^{18}$	18
19	33.578.000.610	$19*2^{19}$	5.325.568	$10{,}16*2^{19}$	19
20	131.282.408.400	$20*2^{20}$	11.280.076	$10{,}76*2^{20}$	20

7. CONCLUSION

The new algorithm proposed in this paper produces the most minimal amount of temporary results which may be kept in the fastest memory levels and can cause the greatest performance in practical applications. By using the "Great or Less" operation this algorithm select one by one the essential PI. Because each of implicants that processed includes the given minterm, no redundant implicant appears and the special covering operation becomes not necessary. The proposed near-minimal simplification algorithm may be used independently and as subprocedure of the exact-minimal simplification algorithm in that only extremely or the greatest single PI is selected. In

this case, the algorithm is repeated until emptying of S_{ON} set. Note that as the function that is processed becomes highly simplificable, simplification process becomes more and more faster. From this point of view, the proposed algorithm may be seen as opposite to the Quine-McCluskey method with respect to some aspect. The related program of proposed algorithm is implemented and shows reliable, robust and satisfactory results. It can be said that the proposed algorithm that depends on transformation method and presented in this paper is shown time effective, reliable, robust and optimal characteristics in the light of implemented program's output.

REFERENCE

1. R. E. Miller, *Switching Theory,* Vol. 1 Combination Circuits, New York; John Wiley and sons, 1965.

2. M. Morris Mano, *Digital Design,* Prentice-Hall International Editions, 1984.

3. Sharon R. Perkins, Tom Rhyne, *An Algorithm for Identifying and Selecting The Prime Implicants of a Multiple-Output Boolean Function,* IEEE Transactions On Computer Aided Design, Vol. 7, No:11, November 1988.

4. B. Gurunath, Nripendra N. Biswas, *An Algorithm for Multiple Output Minimization,* IEEE Transactions On Computer Aided Design, Vol. 8, No:9, September 1989.

5. Karen A. Bartlett, Robert K. Brayton, Gary D. Hachtel, Reily M. Jacoby, Christopher R. Morrison, Richard L. Rudell, Alberto Sangiovanni-Vincentelli, Albert R. Wang, *Multilevel Logic Minimization Using Implicit Don't Cares,* IEEE Transactions On Computer Aided Design, Vol. 7, No:6, June 1988

6. Novruz M. Allahverdi, Şirzad Ş. Kahramanlı, Kayhan Erciyeş, *A Fault Tolerant Rou5ting Algorithm Based On Cube Algebra For Hypercube Systems,* Journal of Systems Architecture 46, 2000, pages 201-205.

7. Anna Bernasconi, Valentina Ciriani, Fabrizio Luccio, Linda Pagli, *Fast Three-Level Logic Minimization Based on Autoymmetry,* 2001, http://citeseer.nj.nec.com/cachedpage /462188/1

8. Bernhard Beckert, Reiner Iıahnle, Gonzalo Escalada-Imaz, *Simplification of Many-Valued Logic Formulas Using Anti-Links,* 1997, http://citeseer.nj.nec.com/cachedpage /221052/1

9. Michel R. Dagenais, Vinod K. Agarwal, Nicholas C. Rumin, *McBOOLE: A New Procedure for Exact Logic Minimization,* IEEE Transactions On Computer Aided Design, Vol. CAD-5, No:1, Jaunary 1986.

10. E.M. Nadjafov, S.S. Kahramanlı, *On the Synthesis of Multiple Output Switching Scheme,* Scientific notes of Azerbaijan Institute of Petroleum and Chemistry, Baku, Azerbaijan, Vol. IX, No 3 (1973) 458-473.

11. S.S. Kahramanlı, N.M. Allahverdi, *Compact Method of Minimization of Boolean Functions with Multiple Variables,* Proc. Intern. Symp. Application of Computers, Selcuk University, Konya, Turkey, (June 1993) 433-440.