Turk J Elec Eng & Comp Sci, Vol.18, No.6, 2010, © TÜBİTAK
doi:10.3906/elk-0904-9

Fast computation of determination of the prime implicants by a novel near minimum minimization method

Fatih BAŞÇİFTÇİ[1], **Şirzat KAHRAMANLI**[2]
[1] *Selçuk University, Department of Electronic and Computer Education, Technical Education Faculty, 42003, Selçuklu, Konya-TURKEY*
e-mail: basciftci@selcuk.edu.tr
[2] *Selçuk University, Department of Computer Engineering, Engineering and Architecture Faculty, 42003 Selçuklu, Konya-TURKEY*
e-mail: sirzat@selcuk.edu.tr

Abstract

In this study proposed is an off-set-based direct-cover near-minimum minimization method for single-output Boolean functions represented in a sum-of-products form. To obtain the complete set of prime implicants including given on-cube (on-minterm), the proposed method uses off-cubes (off-minterms) expanded by this On-cube. The amount of temporary results produced by this method does not exceed the size of the off-set. To make fast computation, we used logic operations instead of standard operations. Expansion off-cubes, commutative absorption operations and intersection operations are realized by logic operations for fast computation. The proposed minimization method is tested on several different kinds of problems and benchmarks results of which are compared with logic minimization program ESPRESSO. The results show that proposed algorithm obtains good results and faster than ESPRESSO.

Key Words: *Boolean minimization, logic operations, prime implicants, direct cover.*

1. Introduction

Two-level logic minimization is a basic problem in logic synthesis [1, 2]. The minimization of Boolean Functions (BFs) can lead to more effective computer programs and circuits. Minimizing functions can be important because, electrical circuits consist of individual components that are implemented for each term or literal for a given function. This allows designers to make use of fewer components, thus reducing the cost of a particular system [3].

A wide variety of Boolean minimization techniques have been explained in [4-10], most of which work on a two-step principles: first, identifying the prime implicants (PIs) of chosen On-minterm and second, determining a set of the essential prime implicants (EPIs) [4-6, 11]. Since the size of the PIs can be as large as $3^n/n$ for a function of n variables [11, 12], the PIs identification step can become computationally impractical as n increases [2, 11, 13, 14]. In particular, according to [15], if each PI of a BF includes exactly l ones, l zeros, and l don't care literals, then the size of the PI is equal to $|S_{PI}|=(3l)!/(l!)^3$. For example, for $l = n/3$ = 6, 7, 8, 9, 10 the values of $|S_{PI}|$ are equal to 17.1×10^6, 39.9×10^7, 94.7×10^8, 228×10^9, 555×10^{10}, respectively. According to [16], all of the procedures for minimizing the BFs into prime and irredundant form have $O(2^n)$complexity. On the other hand, it is evident that a zero-cube (minterm) can be covered by a small subset of the set of PI. This fact allows avoiding the data set complexity of the given problem by using two cube expansion concepts: tautology-based and Off-set based [17]. The heuristic tautology-based schemes are generally slower and often give somewhat inferior results [2, 14, 16-18]. The Off-set based scheme usually provides expansion quickly and in a more global way and its performance is increased with the decreasing of the size of an Off-set that usually can be reduced significantly [2, 17].

2. Definitions and notations

An incompletely-specified Boolean function F with n inputs and m outputs is a mapping $F : B^n \rightarrow Y^m$, where $B=\{0,1,x\}$ where x is a non-appearing coordinate value and $Y=\{0,1,d\}$ where d is a unspecified value. B^n is called the domain and Y^m is called the range of the function. A point in the domain of the function is called a minterm [3, 21].

F is called an *incompletely-specified single-output function* if m=1. The On-set S_{ON}, Do not care set S_{DC},and Off-set S_{OFF} of an incompletely-specified single-output function $F$are sets of minterms that are mapped by $F$to $1, x$, and 0, respectively. The function is customarily denoted by the triplet $F = (S_{ON}, S_{OFF}, S_{DC})$. Since S_{ON}, S_{OFF} andS_{DC} partition B^n, only S_{ON} and S_{DC} are usually specified. S_{OFF} is obtained by complementing the union of S_{ON} and S_{DC}. If $S_{DC} = \emptyset$, the function is called a *completely-specified single-output function* [19, 20].

For shortness and formality of the explanations and following sections of this study the following notations are used. The number of variables of a function: n; a set of On-cubes:S_{ON}; a yet uncovered part of S_{ON}: S_{ONi}; a set of Off-cubes: S_{OFF}; a set of don't care cubes: S_{DC}; an element of the set S_{OFF}: B; an element of the set S_{ON} to be handled: A; the set of Off-cubes expanded by on cube A: $S_E(A)$; the subset of $S_E(A)$ containing only prime cubes: $S_{EP}(A)$; the complete set of prime implicants including the cube A: $S_{PI}(A)$; the essential prime implicant covering the cube A: $S_{EPI}(A)$; a set of prime implicants: S_{PI}; A set of essential prime implicants: S_{EPI}; coordinate subtraction (sharp product) operation: $\#$; commutative absorption operation: ∇; the size of the PI: $|S_{PI}|$, logic OR operation: $\vee$; logic AND operation: $\wedge$; unite operation: $\cup$; intersection operation: $\cap$; EXOR (EXCLUISIVE OR) operation: $\oplus$.

3. The basics and application of the proposed method

In order to obtain a minimum sum of product (SOP) for the given function, the typical minimization method is realized by repeating the following steps until S_{ON} is covered completely [2, 6, 8, 22].

***Minimization** // Input: S_{ON}, S_{OFF}; Output: A minimum form of the given function.*

- An On-cube to be covered is chosen form the S_{ON},
- The set of PIs covering given minterm is generated,
- The essential prime implicant (EPI) chooses from the among of the PIs,
- A covering operation is performed.

3.1. Determination of the all PIs

The determination process of $S_{PI}(A)$ may be simplified significantly via the separate handling of each $A \in S_{ON}$ by S_{OFF} . However, a special filter is necessary in order to convert S_{OFF} into a form which provides a separate computing of $S_{PI}(A)$. If $S_{OFF} = \{B_i\}_{j=1}^{J}$ is a set of Off-cubes and A is an On-cube of a BF then to obtain the set $S_{PI}(A)$ containing all of prime cubes including the cube A by the formula [13, 21, 23],

$$S_{PI}(A) = \{x\}^n \# \{B_k^{EP}\}_{k=1}^{K}, \text{ where } k = 1, 2, ..., K \text{ and } K \leq J. \tag{1}$$

It is necessary, first, to expand all cubes from the set S_{OFF} on rule [13, 21, 23],

$$\text{If } b_j^i = a_j \text{ then } C_j = x \text{ else } C_j = b_j, \text{ where } \forall_j = 1, 2, \ldots, n; \tag{2}$$

then, second, form the set $S_{EP}(A)$ to eliminate all of the non-prime cubes from the set $S_E(A)=\{C_j\}_{j=1,i}$; and third, to subtract the remaining set $S_{EP}(A)$ from the n-cube.

Example 1 Table 1 shows expansion of the elements of the set S_{OFF}={0011, 0110, 1000, 1010, 1011, 1110} on the A=0001$\in S_{ON}$, where the set of expanded Off-cubes and its prime cubes subset are denoted by $S_E(A)$ and $S_{EP}(A)$, respectively.

Table 1. The A=0001 oriented expansion of S_{OFF}.

j	$B_j \in S_{OFF}$	$A \in S_{ON}$	$S_E(A)$	Cube Status	s	$S_{EP}(A)$
1	0011	0001	$xx1x$	Prime	1	$xx1x$
2	0110		$x110$	Non-Prime		-
3	1000		$1xx0$	Prime	2	$1xx0$
4	1010		$1x10$	Non-prime		-
5	1011		$1x1x$	Non-prime		-
6	1110		1110	Non-prime		-

As seen from the Table 1 the result of the set $S_E(A)$={$xx1x, x110, 1xx0, 1x10, 1x1x$, 1110}, where the cubes ($x110, 1x10, 1x1x$, 1110)$\in S_E(A)$ are absorbed by the cubes ($xx1x$, $1xx0$)$\in S_E(A)$. Consequently, the result is $S_{EP}(A)$={$xx1x$, $1xx0$}.

At third step the result of the second step is subtracted from the n-cube and $S_{PI}(A)$ is obtained by removing the non-prime cubes.

The result of the second step (Example 1) is $S_{EP}(A)$={*xx1x, 1xx0*}. Therefore the complete set of prime implicants covering the On-cube A=0001 is obtained by the formula 1 as

$S_{PI}(0001)=xxxx \; \# \; S_{EP}(A)=xxxx \; \# \; \{xx1x, 1xx0\}=(xxxx \; \# \; xx1x) \; \# \; 1xx0=\{0x0x, xx01\}$.

Note that the details of the subtraction operation (#) and the absorption operation (∇) used in this study exist in references [3, 13, 21, 24, 25].

3.2. The EPI identification procedure

It is evident that if the result of the $S_{PI}(A)$ consists of a single PI then it is necessary to fix this PI as $S_{EPI}(A)$. Otherwise, in order to identify the EPI the largest prime implicant is chosen as the essential one. If there is more than one largest then the one generated first is chosen. Note that, largest prime implicant is that one covering the largest number of yet uncovered On-cubes. The largest prime implicant identification procedure may be expressed as following.

Let H and L be prime implicants covering a cube A.

If $|H \cap S_{ONi}| > |L \cap S_{ONi}|$ then $H > L$,

If $|L \cap S_{ONi}| > |H \cap S_{ONi}|$ then $L > H$ (3)

If $|L \cap S_{ONi}| = |H \cap S_{ONi}|$ then either H or L may be chosen.

3.3. The On-set covering procedure

This procedure consists of the following steps:

1. Perform covering by the formula $S_{ONi} = S_{ON(i-1)} \; \# \; S_{EPI}(A)$
2. Form the new state of the EPIs set by the formula $S_{EPI} = S_{EPI} \cup S_{EPI}(A)$.

3.4. Minimization algorithm

This algorithm based on operation defined in part 3.1, 3.2 and 3.3. Figure 1 shows the *near minimum minimization algorithm*, Figure 2 shows *the determination of* $S_{PI}(A)$ and Figure 3 shows the *cover procedure* algorithms.

4. Speed up of the prime implicants determination

The cube algebra operations are used for minimizing the BFs. These cube operations are realized by bit string operations. We show that each cube operations realized by logic operations. In this section, expansion of Off-set minterms with On-set minterms in the given BF, the intersection operations and the commutative absorption operations are realized by logic operations. The proposed algorithm performed for simplifying of BFs has been done by both the cube algebra operations and the logic operations. These performed algorithms have been compared according to the period of solution.

The expressions of the coordinate values of cubes are shown below. Each coordinate value of a cube is represented by two bits as shown in the following [21,26]:

The value of logic 0 of the main coordinate: 01.

The value of logic 1 of the main coordinate: 10.

The value of the nonessential coordinate shown by the term x: 11.

```
Minimization Algorithm ($S_{ON}$, $S_{OFF}$)
Begin
    start:
        if ($S_{ON} = \varnothing$) then  goto end
        $A \leftarrow$ select any On minterm from the $S_{ON}$ set
        Call $S_{PI}(A)$ procedure
        if $|S_{PI}(A)|$=1 then goto result
        Select PI which is cover the most minterm from the $S_{ON}$
    result:
        $S_{EPI}(A) \leftarrow$ selected $S_{PI}(A)$
        set of result $\leftarrow S_{EPI}(A)$
        Call cover procedure
        goto start
    end:
        end
End
```

Figure 1. Near minimum minimization algorithm.

```
Procedure $S_{PI}(A)$
Begin
    $Q_0 \leftarrow$ expand $S_{OFF}$ set by $A$ using Formula 2
    $Q_1 \leftarrow$ Apply the Commutative Absorption Operation to $Q_0$
    $S_{PI}(A) \leftarrow (x)^n \# Q_1$
    Return ($S_{PI}(A)$)
End
```

Figure 2. Determination of $S_{PI}(A)$.

```
Cover Procedure
Begin
    $S_{ONi} \leftarrow S_{ON(i-1)} \# S_{EPI}(A)$
    $S_{EPI} \leftarrow S_{EPI} \cup S_{EPI}(A)$
    Return ($S_{ONi}$ and $S_{EPI}$)
End
```

Figure 3. Pseudo code describing cover procedure.

The coordinate values of the cube have been enabled to be represented by the bit pairs expressed here, because the symbol "x" has been used for the nonessential coordinate values shown among the coordinate values of the cube. To express this symbol in the type of 1 and 0, this kind of projection has been used. Table 2 shows the each coordinate value of the cube is expressed with two bits for the cube C=10x0x1.

Table 2. The expressions of each coordinate value of a cube with two bits.

The value of the cube	1	0	x	0	x	1
The expression of each coordinate with two bits	10	01	11	01	11	10

Each cube $K = k_{n-1}k_{n-2}\ \ldots\ k_i\ \ldots\ k_0$ is represented by a pair of bit series.

$$K_L = l_{n-1}l_{n-2}\ldots l_i\ldots l_0 \text{ and } K_R = r_{n-1}r_{n-2}\ldots r_i\ldots r_0. \tag{4}$$

Here, the expressed values of l_i and r_i are the values of left and right bits in each pair of bit series respectively. For instance, the expression of K=01x0x1 as the pair of bit series will be like the following.

Table 3. The expression of a cube as the pair of bit series.

K	0	1	x	0	x	1	Represented cube
K_L	0	1	1	0	1	1	The expression of the cube
K_R	1	0	1	1	1	0	as the pair of bit series

According to the obtained results, the cube K=01x0x1 has been altered to bit series K_L=011011 and K_R=101110.

The operations expressed above have been used for expansion of minterms existing in off-set, in the operations that necessitate to determine whether cubes include each other or whether a cube covers a minterm or not. These operations are explained in the next section.

4.1. Expansion of Off-set minterms with On-set minterms by logic operations

According to rule 2, while performing these operations, we obtain the cube C by comparing each Off-minterm with the On-minterm, bit by bit. This causes loss of time. The generation of $S_E(A)$ with logic operations is showed in Figure 4. We can perform these rules via logic operations as follows:

$A = a_{n-1}a_{n-2}\ \ldots\ a_i\ \ldots\ a_0$ the minterm in the On-set

$B = b_{n-1}b_{n-2}\ \ldots\ b_i\ \ldots\ b_0$ the minterm in the Off-set

```
Generate $S_E(A)$ (A, B)
Begin
    $C_L = \overline{A}_L \vee B_L$
    $C_R = \overline{A}_R \vee B_R$
    $C = C_L \cup C_R$
    Return C
End
```

Figure 4. Generate $S_E(A)$ with logic operations.

Example 2

A = *0 1 0 0 1 0* The minterm in the On-set.

B = *1 0 0 1 1 0* The minterm in the Off-set.

Each bit of the minterm of the On-set (A) and the one of the Off-set (B) is operated as the pair of bit series

A_L =*010010,* A_R =*101101,* B_L =*100110,* $B_R = 011001$

$C_L = \overline{A}_L \vee B_L = 101101 \vee 100110 = 101111$

$C_R = \overline{A}_R \vee B_R = 010010 \vee 011001 = 011011.$

The cube C is obtained by uniting the bit series pairs (C_L, C_R) as shown in Table 4.

Table 4. Determination of the cube C for Example 2.

C_L	1	0	1	1	1	1
C_R	0	1	1	0	1	1
C	***1***	***0***	***x***	***1***	***x***	***x***

4.2. Performing the intersection of two cubes by logic operations

Suppose the intersection operation is applied to cubes $H = h_1 h_2 \ldots h_i \ldots h_n$ and $L = l_1 l_2 \ldots l_i \ldots l_n$, giving cube C. The operation can be denoted as

$$C = H \cap L. \tag{5}$$

Bit series pairs C_L and C_R obtained from the intersection of H and L can be obtained via logical "AND" operation on the bit series components from cubes H and L:

$$C_L = H_L \wedge L_L \text{ and } C_R = H_R \wedge L_R. \tag{6}$$

On obtaining the bit series of C_L and C_R, the intersection value of the cubes H and L can be found. Determination of cube C follows from application of the following operations:

i) Checking the result for empty set

The following operations allow one to test if C_L and C_R give rise to a value or an empty set:

$$D = (C_L \vee C_R) \oplus \{1\}^n$$
$$\text{If } D \neq 0, \text{ then } C = \emptyset. \tag{7}$$

If the obtained cube D is not zero, then cube C is an empty set. That is, no value has been obtained by the intersection of cubes H and L. If cube D is equal to zero, a value will be obtained by the intersection of the cubes H and L.

ii) Comparing of the cubes *H* and *L* with the result cube *C*

The cube C is obtained from the intersection of cubes H and L. If cube C has any value, we can find out to which cubes H or L belongs to this value via the following operations.

- If $C_L = H_L$ and $C_R = H_R$ then $C = H$;
- If $C = H$ then remove cube H, else save the cube H;
- If $C = L$ then remove cube L, else save the cube L.

The inclusion condition is also determined by the same results.

- Only if $C \neq H$ then the cube H includes the cube of L $(H \supset L)$;
- Only if $C \neq L$ then the cube L includes the cube of H $(L \supset H)$;
- If both $C \neq H$ and $C \neq L$ then neither cubes include each other.

Example 3 Suppose that $H = 10x0x1$ and $L = 10x001$ are two cubes. If we examine the intersection between these cubes and the inclusion of each other by logic operations, we form the bit series $H_L - H_R$ and $L_L - L_R$:

$H_L = 101011$, $H_R = 011110$, $L_L = 101001$, $L_R = 011110$.

The bit series C_L and C_R, and consequently the cube C is obtained by formula (6):

$C_L = H_L \wedge L_L = 101011 \wedge 101001 = 101001$

$C_R = H_R \wedge L_R = 011110 \wedge 011110 = 011110$

$C\ (C_L, C_R) = H \cap L =$ 10x001

Now we apply tests (i) and (ii) from above.

i) Checking the result for empty set:

$D = (C_L \vee C_R) \oplus \{1\}^n = (101001 \vee 011110) \oplus 111111 = 0$

Thus, as D=0, $C \neq \emptyset$. That is, as a result of these operations a value has been obtained by the intersection of the cubes H and L.

ii) Comparing of the cubes H and L with the result cube C:

$C_L = L_L = 101001$ and $C_R = L_R = 011110$ C $= L$.

As $C = L$, the result of the intersection operation is the cube L:

$C = H \cap L = L$.

Looking at the inclusion condition between the cubes, we see cube H includes cube L as $C \neq H$ $(H \supset L)$.

5. The experimental results

The algorithm realizing the near minimum method (NMM) was tested against 30 benchmarks to evaluate the runtime and quality of the results. The tests were conducted on a PC with Intel® Core 2 Duo T7200 2.0 GHz and 1024 MB RAM. Quality of the results was measured by numbers of EPIs forming the minimized functions. Benchmarks were solved by ESPRESSO-EXACT, near minimum method realizing with cube algebra operations (NMM-CAO), and near minimum method realizing with logic operations (NMM-LO). The number of EPIs are the same for the NMM-CAO and NMM-LO, but the runtimes are different.

The results of solutions of 30 benchmarks are shown in Table 5. As seen from Table 5, for 7 benchmarks the results generated by NMM are significantly better than those obtained by ESPRESSO. But there are 3 benchmarks, e, *exps* and *spla*, for which NMM generated a little worse result than ESPRESSO. For the remaining 20 benchmarks both methods obtained the same results. In general, our method generated better, equivalent and worse result for 23.3%, 66.7% and 10% of the benchmarks, respectively. Our method (NMM-CAO) has proved faster than ESPRESSO for all of the benchmarks by a factor of 1.7. And also NMM-LO has proved faster than ESPRESSO for all of the benchmarks by a factor of 2.1. Near minimum method realizing with cube algebra operations (NMM-CAO) is slower than near minimum method realizing with logic operations (NMM-LO) by a factor of 1.3.

Table 5. Number of cubes and runtimes for each benchmark.

Benchmarks	n / $\|S_{ON}\|$ / $\|S_{OFF}\|$	Number of Cubes		Runtime (msec.)			
		NMM	ESPRESSO	NMM-CAO (T$_C$)	NMM-LO (T$_L$)	ESPRESSO (T$_E$)	$\frac{T_C}{T_L}/\frac{T_E}{T_C}/\frac{T_E}{T_L}$
check	04/4/9	1	1	44.77	35.82	58.16	1.2/1.3/1.6
check3	04/8/5	2	2	44.66	36.46	57.30	1.2/1.3/1.6
p82	05/11/13	4	5	45.54	35.63	70.50	1.3/1.5/2
m	06/27/5	4	4	46.54	36.30	71.85	1.3/1.5/2
poperom	06/56/8	7	7	44.55	37.04	70.49	1.2/1.6/1.9
sqr	06/18/46	2	2	44.69	35.37	70.95	1.3/1.6/2
inc	07/12/22	6	7	45.31	35.29	71.32	1.3/1.6/2
linrom	07/65/63	24	24	47.77	36.44	72.84	1.3/1.5/2
max128	07/29/99	8	8	46.73	36.06	71.39	1.3/1.5/2
max3	07/12/116	7	7	45.74	40.62	77.31	1.1/1.7/1.9
sqn	07/48/48	8	12	45.27	37.25	72.18	1.2/1.6/1.9
z5xp1	07/25/103	3	3	45.62	36.22	70.33	1.3/1.5/1.9
dist	08/53/203	12	12	45.85	35.86	70.48	1.3/1.5/2
e	08/65/128	21	20	45.98	35.58	70.79	1.3/1.5/2
ex5	08/33/223	2	2	44.60	38.07	72.05	1.2/1.6/1.9
exp	08/18/52	3	4	45.29	35.81	71.48	1.3/1.6/2
exps	08/65/131	21	20	45.14	36.54	73.42	1.2/1.6/2
f51m	08/128/128	23	23	46.49	36.13	73.99	1.3/1.6/2
mlp4	08/32/224	9	9	47.02	35.98	75.39	1.3/1.6/2.1
root	08/15/241	4	4	46.64	36.19	76.52	1.3/1.6/2.1
rd84	08/120/136	84	84	45.74	37.55	75.80	1.2/1.7/2
apex4	09/4/434	4	4	44.53	35.94	71.96	1.2/1.6/2
max512	09/258/254	10	10	45.24	38.32	73.27	1.2/1.6/1.9
prom2	09/142/145	7	8	48.12	36.03	74.18	1.3/1.5/2.1
max1024	10/516/508	4	4	46.47	36.96	74.50	1.3/1.6/2
br1	12/29/5	5	8	44.79	38.57	58.91	1.2/1.3/1.5
t3	12/27/121	6	6	44.84	36.32	73.18	1.2/1.6/2
spla	16/67/2036	31	29	46.90	37.14	72.56	1.3/1.5/2
bcb	26/10/289	2	5	45.67	38.29	74.68	1.2/1.6/2
bcc	26/3/242	2	2	45.24	36.39	71.43	1.2/1.6/2

6. Conclusion

In this study is proposed an Off-set based direct-cover near-minimum minimization method for single-output Boolean functions represented in a sum-of-products form. To favor fast computation, we used logic operations instead of standard cube algebra operations. The experiments on single-output benchmarks show that this approach can speed up the minimization process. The proposed simplification method (NMM) is faster and the results generated by NMM are significantly better than ESPRESSO. This method may be relatively slower only for functions with the Off-set significantly greater than the On-set. Our method can be useful especially in fields such as image processing, logic synthesis, artificial intelligence and many others field related to processing large Boolean data sets.

Acknowledgement

This work is supported by the Coordinatorship of Selcuk University's Scientific Research Projects.

References

[1] T. Sasao, "Worst and Best Irredundant Sum-of–Product Expressions", IEEE Transactions on Computers, Vol. 50(9), pp. 935-947, 2001.

[2] A. Mishchenco, T. Sasao, "Large-Scale SOP Minimization Using Decomposition and Functional Properties", IEEE CNF, Design Automation Conference, Proceedings, pages:149-154, 2-6 June 2003.

[3] S. Kahramanli, F. Başçiftçi, "Boolean Functions Simplification Algorithm of O(N) Complexity", Journal of Mathematical And Computational Applications Vol. 8, No 4, pp. 271–278, 2002.

[4] O. Coudert, "Two-Level Logic Minimization: an Overview", The VLSI Journal, 17-2, pp. 97-140, October 1994.

[5] D.L. Dietmeyer, "Logical Design of Digital Systems", 2^{nd} ed. Boston, 1978.

[6] R. K. Brayton, G.D. Hachtel, C.T. McMullen, A. Singiovanni-Vincentelli, "Logic Minimization Algorithms for VLSI Synthesis", Boston, Kluwer Academic, 1984.

[7] E. Hong, S. Muroga, "Absolute Minimization of Completely Specified Switching Functions", IEEE Trans. Computers, vol. 40, no 11, pp 53-65, January, 1991.

[8] P.R. Tirumalai, J.T. Butler, "Minimization Algorithms for Multiple-Valued Programmable Logic Arrays", IEEE Trans. Comp, vol.40, no:2, pp:167-178, 1991.

[9] T. Sasao, "A Simplification Algorithm for Exclusive OR-Sum-of Products Expressions for Multiple–Valued- Input Two- Valued – Output Functions", IEEE Trans. Computer Aided Design, vol. 12, no 5, pp.621-632, May, 1993.

[10] P. McGeer, J. Sanghavi, R.K. Brayton, A. Sangiovanni-Vincentelli, "ESPRESSO-SIGNATURE: A New Exact Minimizer for Logic Functions", IEEE Transactions on VLSI, Vol. 1, No. 4, pp. 432-440, 1993.

[11] R. S. Perkins, T. Rhyne, "An Algorithm for Identifying and Selecting The PI's of a Multiple-Output Boolean Function", IEEE Trans. Computer Aided Design, vol. 7, no 11, pp.1215-1218, November 1988.

[12] B. Gurunath, N. N. Biswas, "An Algorithm for Multiple Output Minimization", IEEE Trans. Comp.Aided Design, Vol. 8, no 9, September 1989.

[13] Ş. Kahramanlı,S. Güneş, S. Şahan, F. Başçiftçi, "A New Method Based on Cube Algebra for The Simplification of Logic Functions", The Arabian Journal For Science and Engineering Volume 32, Number 1B, pages:1-14, April 2007.

[14] P. Fisher, J. Hlavicka, H. Kubatova, "FC-Min: "A Fast Multi-Output Minimizer", IEEE CNF, Digital System Design, Proceedings, Euromicro Symposium on, 1-6, pages:451-454, Sept. 2003.

[15] R. E. Miller, "Switching Theory", Vol. 1. New York: Wiley, 1965.

[16] K.A. Bartlett, R.K. Brayton, G. D. Hachtel, R. M. Jacoby, C. R. Morrison, R. L. Rudell, A. Sangiovanni-Vincentelli, Albert R. Wang, "Multilevel Logic Minimization Using Implicit Don't Cares", IEEE Trans. Computer Aided Design, vol. 7, no 6, pp. 723-740, June 1988.

[17] A.A. Malik, R.K. Brayton, A.R. Newton, A.Singiovanni-Vincentelli, "Two-Level Minimization of Multi valued Functions with Large Offsets", IEEE Trans. Comput., vol. 42, no 11, pp. 1325-1342, November 1993.

[18] T. Sasao, J.T. Butler, "Worst and Best Irredundant Sum-of-Product Expressions", IEEE Trans. Comput., vol. 50, no 9, pp. 935-947, September 2001.

[19] A.A. Malik, R.K. Brayton, A.R. Newton, A. Singiovanni-Vincentelli, "Reduced Offsets for Minimization of Binary-Valued Functions", IEEE Trans. Comput., vol. 10, no 4, pp. 413-426, April 1991.

[20] C. Umans, T.Villa and A. Singiovanni-Vincentelli, "Complexity of Two-Level Logic Minimization", IEEE Trans. Comput. Aided Design of Integrated Circuits and Systems, vol. 25, no 7, pp. 1230-1246, July 2006.

[21] F. Başçiftçi, "Local Simplification Algorithms for Switching Functions", PhD Thesis, Graduate School of Natural and Applied Sciences, Selcuk University, 2006.

[22] De Micheli Giovanni, "Synthesis and Optimization of Digital Circuits", New York, Mccraw-Hill, 1994.

[23] F. Başçiftçi, Ş. Kahramanlı, "An Off-Cubes Expanding Approach to the Problem of Separate Determination of the Essential Prime Implicants of the Single-Output Boolean Functions" EUROCON 2007, Warsaw-Poland, ISBN:1-4244-0813-X, pages: 432-438, 09-12 September 2007.

[24] F. Başçiftçi, "Simplification of Single Output Boolean Functions by Exact Direct Cover Algorithm Based on Cube Algebra" EUROCON 2007, Warsaw-Poland, ISBN:1-4244-0813-X, pages: 427-431, 09-12 September 2007.

[25] E. Nadjafov and. Ş. Kahramanlı, "On The Synthesis of Multiple Output Switching Scheme", Scientific Notes of Azerbaijan Institute of Petroleum and Chemistry, 9(3), pp. 65-69, 1973.

[26] F. Başçiftçi, Ş. Kahramanlı, "The Modelling of Cube Algebra Operations by the Basic Computer Operations", Proceeding of The International Conference on Modeling and Simulation, The Association for Modeling and Simulation in Enterprises (AMSE), Selçuk Uni., Vol.II, pages:595-599, 2006.