Minimizing Boolean Sum of Products Functions Using Binary Decision Diagram

Debajit Sensarma[1], Subhashis Banerjee[1], Krishnendu Basuli[1], Saptarshi Naskar[2], and Samar Sen Sarma[3]

[1] West Bengal State University, West Bengal, India
[2] Sarsuna College, West Bengal, India
[3] University Of Calcutta, West Bengal, India
{mail.sb88,debajit.sensarma2008, Krishnendu.basuli,sapgrin}@gmail.com, Sssarma2001@yahoo.com

Abstract. Two-level logic minimization is a central problem in logic synthesis, and has applications in reliability analysis and automated reasoning. This paper represents a method of minimizing Boolean sum of products function with binary decision diagram and with disjoint sum of product minimization. Due to the symbolic representation of cubes for large problem instances, the method is orders of magnitude faster than previous enumerative techniques. But the quality of the approach largely depends on the variable ordering of the underlying BDD. The application of Binary Decision Diagrams (BDDs) as an efficient approach for the minimization of Disjoint Sums-of-Products (DSOPs). DSOPs are a starting point for several applications.

The use of BDDs has the advantage of an implicit representation of terms. Due to this scheme the algorithm is faster than techniques working on explicit representations and the application to large circuits that could not be handled so far becomes possible. Theoretical studies on the influence of the BDDs to the search space are carried out. In experiments the proposed technique is compared to others. The results with respect to the size of the resulting DSOP are as good or better as those of the other techniques.

Keywords and Phrases: Binary Decision Diagram, DSOP, Unate Function, Binate Function.

1 Introduction

A DSOP is a representation of a Boolean function as a sum of disjoint cubes. DSOPs are used in several applications in the area of CAD, e.g. the calculation of spectra of Boolean functions or as a starting point for the minimization of Exclusive-Or-Sum-Of-Products (ESOPs). In some techniques for minimization of DSOPs have been introduced. They are working on explicit representations of the cubes and therefore are only applicable to small instances of the problem.

BDDs in general are an efficient data structure for presenting and manipulating the Boolean functions. They are well-known and widely used in logic synthesis and

N. Meghanathan et al. (Eds.): CCSIT 2012, Part III, LNICST 86, pp. 36–48, 2012.

formal verification of integrated circuits. Due to the canonical representation of Boolean functions they are very suitable for formal verification problems and used in a lot of tools to date [**14, 15, 16**].BDDs are well-suited for applications in the area of logic synthesis, because the cubes in the ON-set of a Boolean function are implicitly represented in this data structure. A hybrid approach for the minimization of DSOPs relying on BDDs in combination with structural methods has recently been introduced in. It has been shown that BDDs are applicable to the problem of DSOP minimization [**5**].

Given a BDD of a Boolean function, the DSOP can easily be constructed: each one-path [**6**], i.e. a path from the root to the terminal 1 vertex, corresponds to a cube in the DSOP, and moreover, different one-paths lead to disjoint cubes. For the construction of the BDD the variables of the Boolean function are considered in a fixed order. The permutation of the variables largely influences the number of one-paths in the BDD and thus the number of cubes in the corresponding DSOP. Additionally, the importance of choosing a good variable order to get a small DSOP has theoretically been supported.

1.1 Motivation

The motivation of this project is to minimize the boolean sum of product function by finding the minimal irridundant expression. Here Binary Decision Diagram(BDD) is used for finding Disjoint cubes first, because BDD is the compact representation of a boolean function, but it highly depends on variable ordering. Then this disjoint cubes are minimizes to get the minimal expression. In Quine-McCluskey method, it can be shown that for a function of n variables the upper bound on the number of prime implicantes is $3^n/n$. In this project a heuristic algorihm is used to minimize the upper bound of the prime implicant generation and it gives the near optimal solution.

1.2 Binary Decision Diagrams

A BDD is a directed acyclic graph $G_f = (V, E)$ that represents a Boolean function $f: B^n \rightarrow B^m$. The Shannon decomposition $g = x_i g_{xi} + x_i \cdot g_{xi}$, is carried out in each internal node v labeled with label (v) = x_i of the graph, therefore v has the two successors then (v) and else (v). The leaves are labeled with 0 or 1 and correspond to the constant Boolean functions. The root node root (G_f) corresponds to the function f. In the following, BDD refers to a reduced ordered BDD (as defined in **[8]**) and the size of a BDD is given by the number of nodes.

Definition: A one-path in a BDD $G_f = (V, E)$ is a path
$p = (v_0, \ldots, v_{l-1}, v_l)$;
$v_i \in V$; $(v_i, v_{i+1}) \in E$
with $v_0 = root(G_f)$ and $label(v_l) = 1$. p has length l + 1.
$P_1 (G_f)$ denotes the number of all different one-paths in the BDD G_f.

1.3 BDD and DSOP

Consider a BDD Gf representing the Boolean function $f(x_1, \ldots, x_n)$. A one path $p = (v_0, \ldots, v_l)$ of length l + 1 in Gf corresponds to an (n - l)-dimensional cube that is a subset of ON(f)[1]. The cube is described by:

m_p =n l_i for i=0…l-1; where
l_i = label (v_i); if v_i+1 = else (v_i)
label (v_i); if v_i+1 = then (v_i)

Two paths p1 and p2 in a BDD are different if they differ in at least one edge. Since all paths originate from root (G_f), there is a node v where the paths separate. Let label (v) = x_i. Therefore one of the cubes includes xi, the other x_i. Hence, the cubes mp1 and mp2 are disjoint.

Now the DSOP can easily be built by summing up all cubes corresponding to the one-paths.

Remark 1 : Let Gf be a BDD of f(x_1 ,…, x_n) and M1 be the set of one-paths in G_f . Then G_f represents the DSOP

$$\sum m_p \text{ where } p \in M_1$$

where m_p is the cube given above.

From this it is clear that the number of cubes in the DSOP represented by G_f is equal to $P_1(G_f)$. Thus, as opposed to the usual goal of minimizing the number of nodes in a BDD, here the number of one-paths is minimized. Known techniques to minimize the number of nodes can be used to minimize the number of paths by changing the objective function. One such technique is sifting. A variable is chosen and moved to any position of the variable order based on exchange of adjacent variables. Then it is fixed at the best position (i.e. where the smallest BDD results), afterwards another variable is chosen. No variable is chosen twice during this process.

2 Terms Related to Sop Minimization

Unite function: A function that is monotonically increasing or decreasing in each of its variable is called unite function.

Binate function: a function that is not unate. This can also be used to mean a cover of a function that is not unate.

Canonical cover/solution: the SOP cover of a function that contains only minterms, and thus has not at all been reduced.

Cube: a one-dimensional matrix in the form of an implicant. Two cubes are said to be **Disjoint** if their intersection of the set of minterms is null.The intersection is the operation of conjunction (i.e. the Boolean AND operation).

Espresso algorithm: an algorithm that minimizes SOP functions.

Essential prime implicant: a prime implicant that the cover of a function must contain in order to cover the function

Implicant: an ANDed string of literals. It is a term in an SOP function.

Literal: an instance of a boolean variable. It may be the variable complemented, uncomplemented, or ignored (don't-care). In matrix representations or the Q-M algorithm, it may have a value of 0, 1, or 2/X, corresponding to complemented, uncomplemented, and don't-care, respectively.

Matrix representation of a function or implicant: The rows of a two-dimensional matrix representation of a function are the implicants of the function. The columns of a one-dimensional matrix representation of an implicant are the literals of the implicant.
Minterm: an implicant that contains exactly one literal for each variable. It is not at all simplified.

Monotone decreasing: A function is monotone decreasing in a variable if changing the value of the variable from 0 to 1 results in the output of the function being 0.

Monotone increasing: A function is monotone increasing in a variable if changing the value of the variable from 0 to 1 results in the output of the function being 1.

Prime implicant: an implicant that cannot be further reduced by adjacency

Quine-McCluskey (Q-M) algorithms: two algorithms that minimize a boolean function.The first algorithm finds all prime implicants, and the second algorithm eliminates nonessential prime implicants.

3 Proposed Algorithm

In this method at first from the given truth table suitable variable order is chosen based on Shannon entropy measurement, then binary decision diagram is made considering this variable order. After that disjoint cubes are calculated by following the 1-path of the BDD. Then from that the covering matrix is created where columns represent the variables and row represents the applicants or disjoints cubes. Then selecting the most binate variables and by unite simplification the ultimate minimized sop function is obtained.

3.1 Algorithm

Step 1: Generation of truth table.

Step 2:Variable reordering using Shannon entropy measure-ment **[2,8]** and create Binary Decision Diagram**[7,13,14]**.

Step 3: Finding Disjoint Cubes from Binary decision Diagram **[5]**.

Step 4: Disjoint cube minimization using Binate Covering with Recursion and Unate Simplification Method **[11,12]**.

3.2 Explanation

Step 1:
The truth table is generated from given Boolean expression.
Step2:
Choosing right variable order is very important for constructing Binary Decision Diagram, because if bad variable order is chosen then number of 1-paths can be increased; even number of nodes in the BDD may be increased exponentially.

The measures of a variable's importance are based on information theoretic criteria, and require computation of entropy of a variable. Entropy measures can be

quite effective in distinguishing the importance of variables. It is well known that a central problem in using OBDD is the severe memory requirements that result from extremely large OBDD size that arise in many instances.OBDD sizes are unfortunately very sensitive to the order chosen on input variables. Determining the optimal order is a co-NP complete problem **[8]**.Variable ordering heuristics can be classified as either static or dynamic approaches. A static approach , analyzes the given circuit/function and, based on its various properties, determines some variable order which has a high "Probability" of being effective. In dynamic approach to compute variable ordering, one starts with an initial order, which is analyzed and permuted at internal points in the circuit/function, such that some cost function is minimized.

Step 3:
In this step Disjoint Cubes from Binary decision Diagram are found by following the 1-path **[5]**.

Step 4:
Cover matrix is found from the resultant disjoint cubes and this is simplified using Unate Recursive Paradigm [**12**].

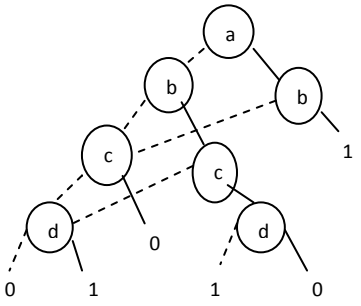
4 Illustration with an Example

f(a,b,c,d)=∑(1,5,6,9,12,13,14,15)

Step1: *Given the truth table*.

a	b	c	d	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

With this variable order the Binary Decision Diagram is:



Number of nodes=7

Step 2: *Variable ordering by calculating Entropy and choosing most ambiguous variables.*

I(a,0)=0.954
I(a,1)=0.954
E(a)=0.954
I(b,0)=0.811
I(b,1)=0.811
E(b)=0.811 Select 'b' as the first splitting variable.
I(c,0)=0.954
I(c,1)=0.954
E(c)=0.954
I(d,0)=0.954
I(d,1)=0.954
E(d)=0.954
For b=0 the truth table is:

a	c	d	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

I(a,0)=0.811
I(a,1)=0.811
E(a)=0.811
I(c,0)=1
I(c,1)=0

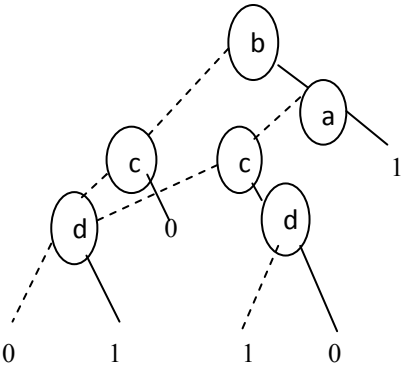
E(c)=0.5
I(d,0)=0
I(d,1)=1
E(d)=0.5

For b=1 the truth table is:

a	c	d	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

I(a,0)=1
I(a,1)=0
E(a)=0.5 Select 'a' as the next splitting variable.
I(c,0)=0.811
I(c,1)=0.811
E(c)=0.811
I(d,0)=0.811
I(d,1)=0.811
E(d)=0.811

If we proceed like this we come up with the variable order->b,a,c,d
With this variable order the Binary Decision Diagram is:



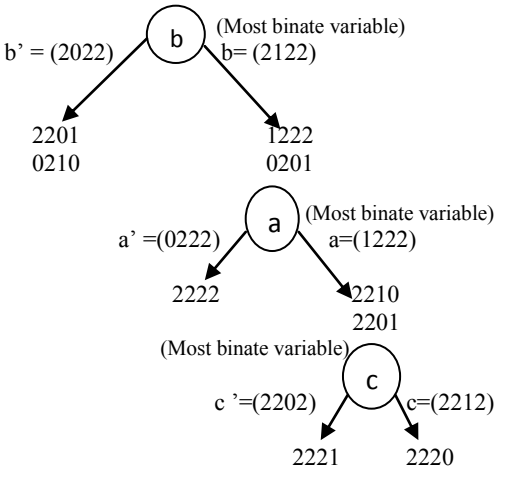
Number of nodes=6
Step 3: *Finding Disjoint Cubes from Above Binary Decision Diagram.*
The Disjoint Cubes are: **ab + a'bcd'+ b'c'd + a'bc'd**

Step 4: *Binate Covering with Recursion.*

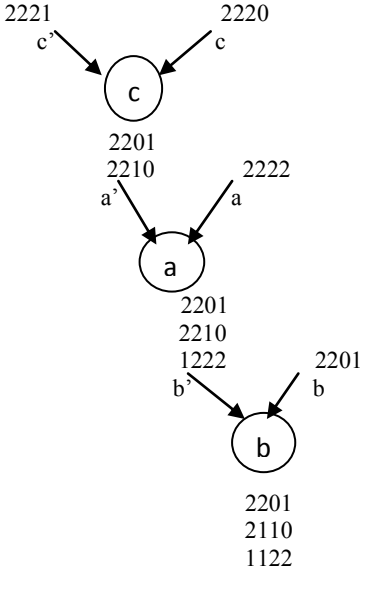
The Covering Matrix is:

	a	b	c	d
ab	1	1	2	2
a'bcd'	0	1	1	0
b'c'd	2	0	0	1
abc'd	0	1	0	1

Case 1: *Binate Select.*



Case 2: *Merge.*



After simplification the expression is:
ab + c'd + bcd'.

Karnaugh Map Representation:

cd / ab	00	01	11	10
00		1		
01	1	1	1	1
11		1		1
10		1		

5 Result

This program is done on Intel Pentium 4 CPU, 2.80 GHz and 256 MB of RAM and with Visual c++ 6.0 standard edition. The result is presented here with the following figures.

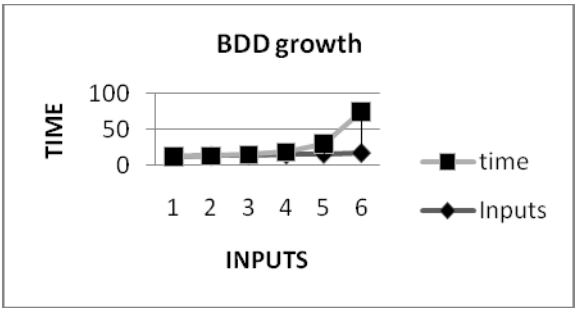
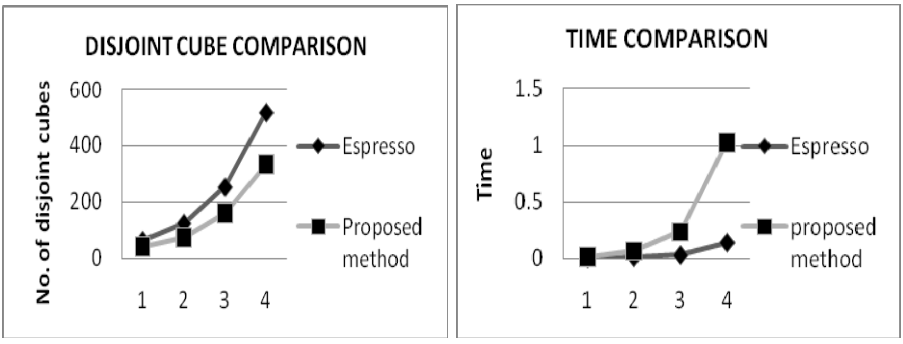


Fig. 1. growth of Binary Decision Diagram. Here X-axis represents the number of inputs and Y-axis represents the required time.

Here growth of the creation of Binary decision diagram with proposed method is shown with respect to 6 variables.

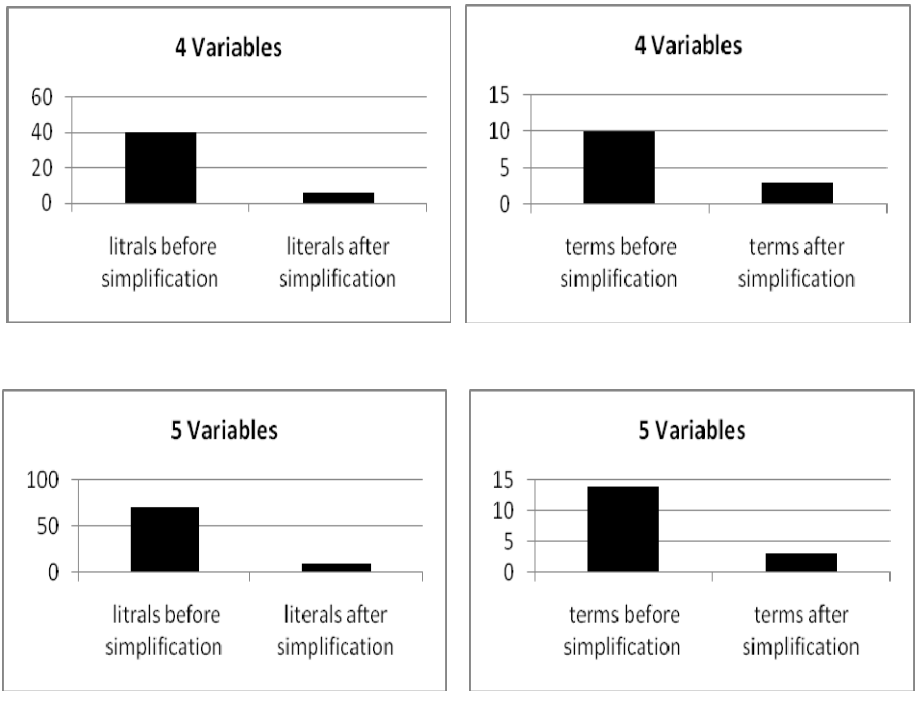
Here comparison of creation of number of disjoint cubes and time taken to create the disjoint cube of Espresso Logic Minimizer and the proposed method is done with respect to 4 variables.

Here number of literals and terms before and after minimizing the given Boolean sum-of-product function is given with respect to 4,5 and 6 variables with the Proposed method.



(a) (b)

Fig. 2. (a) Comparison of number of disjoint cubes generated by ESPRESSO heuristic logic minimizer and the Proposed method. X-axis represents the number of variables and Y-axis represents the number of disjoint cubes. **(b)** Comparison of time taken to generate disjoint cubes by ESPRESSO heuristic logic minimizer and the Proposed method. X-axis represents the number of variables and Y-axis represents the time.



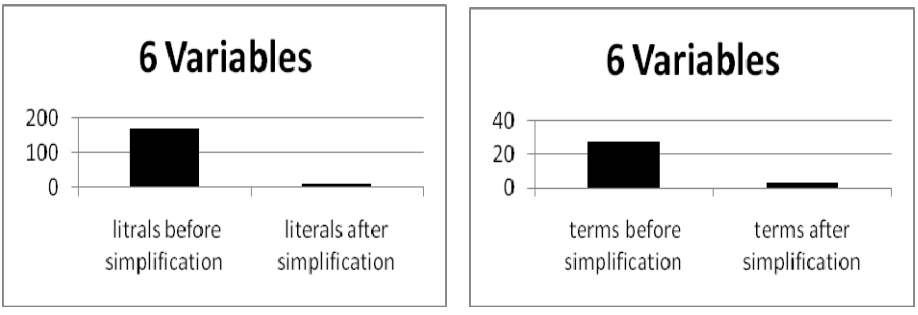


Fig. 3. Comparison of the number of literals and terms before simplification and generated by above program for 4,5 and 6 variables

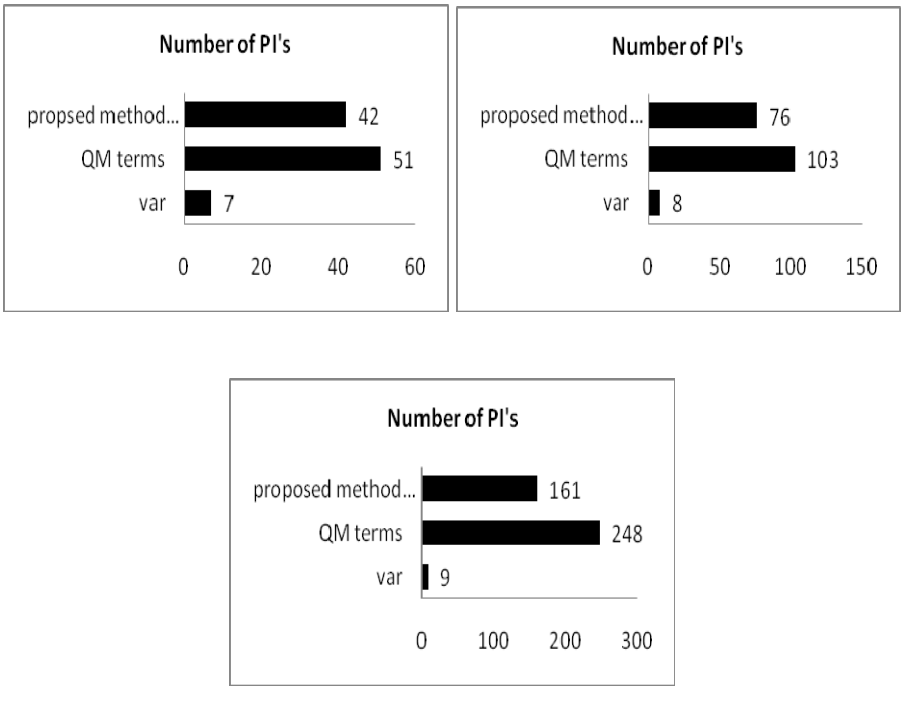


Fig. 4. Comparison of number of Prime implicantes generated by Quine-McCluskey algorithm and the Proposed method for 7, 8 and 9 variables. Here QM stands for Quine-McCluskey

Prime implicants generated by the Quine-McCluskey procedure and the Proposed method is compared here with respect to 7,8 and 9 variables.

5 Conclusions

An approach based on Binary Decision Diagram and Binate covering algorithm to minimize the Boolean SOP function with DSOP representation of a Boolean function was presented. It is completely based on heuristics and gives the near optimum solution. But this procedure will only work for single output and completely specified functions and gives the near optimal solution. Whether it will work for incompletely specified function and multiple-output function is not tested yet.

6 Future Works

1.Generation of all possible minimal covers or
minimal expressions.
2.Compare with ESPRESSO logic minimizer.
3.Test whether it will work for incompletely specified function and multiple-output functions.

References

[1] Yang, C., Ciesielski, M.: BDD-Based Logic Optimization System. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 21(7), 866–876 (2002)
[2] Popel, D.V.: Towards Efficient Calculation of Informationmeasures for Reordering of Binary Decision Diagrams. Computing Research Repository - CORR, cs.AR/0207 (2002)
[3] Knuth, D.E.: The Art of Computer Programming, vol. 4
[4] Swamy, G.M.: An Exact Logic Minimizer Using Implicit Binary Decision Diagram Based Methods. In: ICCAD 1994 Proceedings of the 1994 IEEE/ACM International Conference on Computer-Aided Design (1994)
[5] Fey, G., Drechsler, R.: Utilizing BDDs for Disjoint SOP Minimization. In: 45th IEEE International Midwest Symposium on Circuits and Systems (2002)
[6] Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.): EvoWorkshops 2004. LNCS, vol. 3005. Springer, Heidelberg (2004)
[7] Andersen, H.R.: An Introduction to Binary Decision Diagrams. Lecture Notes, Technical University of Denmark (October 1997)
[8] Jain, J., Bitner, J., Moundanos, D., Abraham, J.A., Fussell, D.S.: A new scheme to compute variable orders for binary decision diagrams. In: Proceedings of the Fourth Great Lakes Symposium on, Design Automation of High Performance VLSI Systems, GLSV 1994, March 4-5, pp. 105–108 (1994)
[9] Hilgemeier, M., Drechsler, N., Drechsler, R.: Minimizing the number of one-paths in BDDs by an evolutionary algorithm. In: Congress on Evolutionary Computation (2003)
[10] Nosrati, M., Hariri, M.: An Algorithm for Minimizing of Boolean Functions Based on Graph DS. World Applied Programming 1(3), 209–214 (2011)
[11] Coudert, O.: On solving Covering Problems. In: Proc. of 33rd DAC, Las Vegas (June 1996)

[12] Brayton, R.K.: Logic minimization algorithms for VLSI synthesis
[13] Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers C-35-8, 677–691 (1986)
[14] Bryant, R.E.: Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. ACM Computing Surveys (1992)
[15] Bryant, R.E.: Binary decision diagrams and beyond: Enabeling techniques for formal verification. In: Int'l Conf. on CAD, pp. 236–243 (1995)
[16] Drechsler, R., Becker, B.: Binary Decision Diagrams – Theory and Implementation. Kluwer Academic Publishers (1998)
[17] Kohavi, Z.: Switching and Finite Automata Theory