Reduction of the Number of Paths in Binary Decision Diagrams by Linear Transformation of Variables

Osnat Keren
Bar Ilan University, Israel
kereno@macs.biu.ac.il

Ilya Levin,
Tel-Aviv University, Israel
ilia1@post.tau.ac.il

Radomir S. Stankovic
University of Nis, Serbia
rstankovic@bankerinter.net

Abstract

The paper deals with the problem of counting and minimizing the number of paths in Binary Decision Diagrams. The suggested approach uses the linear transform of initial variables and is based on a newly introduced weighted autocorrelation function. It is shown that the total number of paths in BDD is the sum of values of a weighted autocorrelation function. The efficiency of the proposed technique is illustrated on a number of benchmarks.

1 Introduction

Binary Decision Diagrams (BDD) and Multi Terminal Binary Decision Diagrams (MTBDD) are forms of representing a Boolean function by an acyclic directed graph. This representation is convenient for several applications like synthesis and formal verification. CAD tools for synthesis and verification benefit when the size of the BDD and the number of paths are reduced. Massive research has been done to reduce the memory size by static and dynamic algorithms for reordering the BDD nodes and by replacing them by a linear combination of the input variables, see for example [4] and references therein.

In [1], Dubrova and Miller showed that "the number of nodes in a reduced ordered BDD is not a monotonically increasing function of the number of implicants in a disjoint cover, represented by the paths of this BDD". Hence the criterion for minimizing the number of nodes in a BDD doesn't necessarily obtains a smaller number of paths.

Recently Fey and Drechsler [2] suggested a dynamic number of paths minimization procedure by reordering of variables. The technique is based on nodes swapping and modified sifting with the acceptance criterion formulated as the minimal number of paths. This approach reduces the number of paths in the BDD for standard benchmarks functions. Another approach, based on evolutionary algorithm for minimizing the number of paths was introduced in [3]. The evolutionary algorithm requires a predefined set of parameters. A successful parameter setting may give improvement over the modified sifting discussed in [2].

The present paper investigates two issues concerning the number of paths: a) analytical formulation of the number of paths based on a newly introduced *weighted autocorrelation function*, b) a deterministic procedure towards minimization of the number of paths in a BDD. Several ways to count the number of paths in a BDD are described. The conventional way is an iterative bottom up counting: the leaves of the binary tree are assigned with an initial weight and the tree is folded up to its root while updating the weights. The number of paths is the weight of the root node. In this way, the calculation is done **per node** and the number of paths increases per level. In this paper, the number of paths is calculated recursively by starting from the bottom of the binary tree with the maximal possible number of paths in the corresponding decision tree and subtracting the number of redundant paths **per level** while folding the tree. The method is based on the fact that the number of redundant paths at a level i is the value of the weighted autocorrelation function of the i'th input variable.

The paper introduces a deterministic method for reducing the number of paths by replacing the original input variables by their linear combination, the technique is known as linearization. The linearized BDD of a given function f is defined by a linear transformation σ and a the corresponding linearly transformed function f_σ, where $f(x) = f_\sigma(\sigma x)$. The linear transformation σ can be described as a nonsingular matrix. When σ is a permutation matrix, the linearization becomes reordering. Linearization based on the **total** autocorrelation function, defined by Karpovsky in [5], reduces the implementation complexity of a circuit measured as the number of two-input AND/OR circuits required. In [4] the linearization was employed to reduce **on average** the number of nodes in a BDD. However, since the number of paths is a linear function of the weighted autocorrelation function, the linearization (or even reordering by the autocorrelation values) **always** reduces the number of paths.

The paper is organized as follows. The next Section 2 includes mathematical background. In Section 3 we discuss methods for counting the number of paths and the relation between the number of paths and the autocorrelation function. Section 4 formulates the linearization problem and describes the paths minimization procedure. Experimental results on standard benchmark functions are presented in Section 5. The conclusions summarizing the results are presented in Section 6.

2 Preliminaries

Consider a multi-output logic function of n input variables and k outputs, $f : GF(2^n) \rightarrow GF(2^k)$. The input pattern x is an element of $GF(2^n)$ and can be represented as a linearly combination of the n input variables $\{x_i\}_{i=0}^{n-1}$, where each x_i is the binary vector corresponding to 2^i in base 2. The set of x_i's form a normal basis of $GF(2^n)$. For example, in $GF(2^4)$ the element $x = (1001)$ is a linear combination of $(1000) + (0001) = x_3 + x_0$. Clearly, any set of n independent elements of $GF(2^n)$ form a basis.

A MTBDD of a multi-output function is a directed acyclic graph with at most 2^k terminal nodes (leaves), each nonterminal node has an index to one base vector and has two outgoing edges. The outputs can be considered as vectors in $GF(2)^k$, in this case there are k BDDs that may share nodes (called Shared BDDs (SBDD)). An ordered BDD (or a MTBDD) is a BDD where the base vectors appear in some fixed order and once at each path. Given the order of the base vectors, it is possible to reduce the number of nodes by a) eliminating nodes whose outgoing edges point to the same subtree. b) sharing all equivalent subtrees. The Reduced Ordered BDD is called ROBDD. When it is clear from the context we omit the prefix and hence MTBDD, ROBDD, ROMTBDD, SBDD etc. are all referred as a BDD.

The size of a BDD is the number of nonterminal nodes and constant nodes. Usually the normal base is used; the root of the tree is x_{n-1} and a node associated with x_i descends from x_{i+1}. However it is possible to reduce the BDD size by reordering the base vectors [7] or by replacing them with a different set of base vectors [4]. The later operation is called linearization.

For a given order of base vectors, the node reduction is done in steps starting from the bottom of the binary tree. Each step, leaves that are rooted from the same node and carry the same value are merged and the tree is folded. Folding a tree means eliminating the lower level by increasing the alphabet size of the leaves. For example, consider the function $f_1(x_2, x_1, x_0)$ of three input variables and two outputs defined in Table 1. The folding of f_1 is described in Figure 1.

$x_2x_1x_0$	$f_1(x_2, x_1, x_0)$	
000	00	0
001	01	1
010	00	0
011	01	1
100	00	0
101	01	1
110	10	2
111	11	3

Table 1: Truth table of $f_1(x_2, x_1, x_0)$

The formal definition of folding is the following. Let v_x stand for the decimal value of a binary vector x. Denote by $f^i : GF(2)^{n-i} \rightarrow GF(2^{k2^i})$,$0 < i < n$, the folded function of level i,

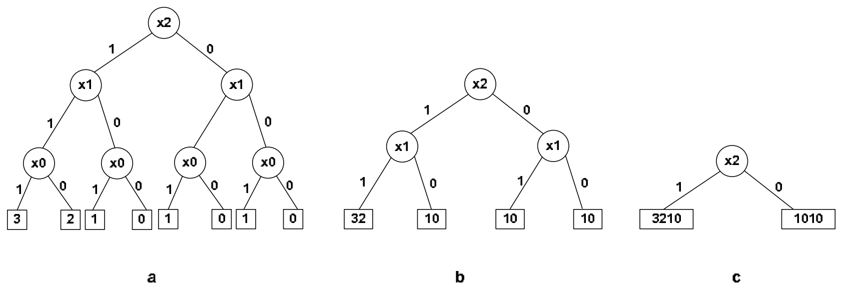

Figure 1: The folding of $f_1(x_2, x_1, x_0)$ by steps: a) Original BDD, b) BDD after folding with respect to x_0, c) After folding with respect to x_0 and x_1 , and d) BDD after folding with respect to x_0, x_1 and x_2.

and let $f^0 = f$ and $(y_{n-1-i}, \ldots y_1, y_0) = (x_{n-1}, \ldots x_{i+1}, x_i)$, then

$$\begin{aligned} f^i \quad & (y_{n-1-i}, \ldots y_1, y_0) = \\ & \sum_{x \in GF(2)^i} (2^k)^{v_x} f(y_{n-1-i}, \ldots y_1, y_0, x_{i-1}, \ldots x_1, x_0) \end{aligned}$$

Let $u \in GF(2^k)$ an output value of a multi-output function, the characteristic function of u denoted by f_u is a Boolean function from $GF(2)^n$ to $GF(2)$,

$$f_u(x) = \begin{cases} 1 & f(x) = u \\ 0 & otherwise \end{cases}$$

The autocorrelation function of f_u is

$$R_u(\tau) = \sum_{x \in GF(2^m)} f_u(x) f_u(x \oplus \tau).$$

Denote by δ_i the binary vector of the length n that represents the decimal value 2^i in base two. $R_u(\delta_0)$ equals to the number of leaves rooted from the same node and carry the value u.

The w-weighted autocorrelation function, $R^w(\tau)$, is defined as

$$R^w(\tau) = \sum_{u \in GF(2^k)} w_u R_u(\tau) \tag{1}$$

where $\{w_u\}_{u \in GF(2^k)}$ are the set of weights. The total autocorrelation function defined in [5] is obtained from the weighted autocorrelation by assigning the value 1 to all the weights.

3 Relation between the number of paths and the autocorrelation function

A planar BDD or a Binary Decision Tree (BDT) is derived from a complete binary decision tree by eliminating nodes whose edges are pointing to the same subtrees, but without sharing equivalent subtrees that are not rooted from the same node. It is well known that for a given order of variables, the number of paths in a BDD is equal to the number of paths of the equivalent planar BDD. Therefore the number of paths of a given BDD is the number of edges connected to the terminal nodes of the equivalent planar BDD.

The number of paths in a planar BDD can be calculated recursively starting from the bottom of the binary tree, by assigning weights to the leaves of the folded functions. Each output value $u \in GF(2^k)$ of the original function f, is assigned with a weight $c_u^{(0)} = 1$. The leaves of the BDD of the folded function f^i represent the output values $u \in GF(2^k)^{2^i}$ where the symbol u is a pair of two symbols (u_m, u_l) over $GF(2^k)^{2^{i-1}}$. The leaves are assigned with weights $c_u^{(i)}$ where

$$c_u^{(i)} = \begin{cases} c_{u_m}^{(i-1)} + c_{u_l}^{(i-1)} & u_m \neq u_l \\ c_{u_m}^{(i-1)} & u_m = u_l \end{cases} \tag{2}$$

$c_u^{(i)}$ equals to the number of paths connected to the nonterminal nodes of the subtree that the leaf u represents. Note that the number of distinct values of u's of the folded function f at the level i is at most 2^{n-i}. At the upper level there is a single u, $u = (f(2^n-1), \dots f(2), f(1), f(0))$ and therefore the number of paths of a BDD is $c^{(n)}$.

For example, the path count of f_1 described in Table 1 is the following,

level	$c - weights$
0	$c_0 = c_1 = c_2 = c_3 = 1$
1	$c_{10} = c_1 + c_0 = 2$ $c_{32} = c_3 + c_2 = 2$
2	$c_{1010} = c_{10} = 2,$ $c_{3210} = c_{32} + c_{10} = 4$
3	$c_{32101010} = c_{3210} + c_{1010} = 6$

Thus, the BDD of f_1 has six paths.

Let us discuss a different way to count the number of paths by using a weighted autocorrelation function. The computation is done per level rather than per node. The proposed method to reduce the number of paths by a linear transform of the input variables is based on representing of the number of paths as a linear function of the weighted autocorrelation function.

Denote by N_u^i the number of ones of the characteristic function f_u^i of the folded function f^i at the level i and define C^i as the accumulated paths at the level i,

$$C^i = \sum_{u \in GF(2^{k2^i})} N_u^i c_u^i.$$

Clearly

$$C^0 = \sum_{u \in GF(2^k)} N_u^0 c_u^0 = \sum_{u \in GF(2^k)} N_u^0 = 2^n.$$

The following Theorem 1 states that the number of accumulated paths at the level i depends on the number of accumulated paths at the level i and the weighted autocorrelation of the folded functions;

Theorem 1 *Let $R^{c,i}$ stands for the weighted autocorrelation of the folded function f^i at the level i with weights $\{c_u^i\}$ as defined in (2), then*

$$C^i = C^{i-1} - 0.5R^{c,i-1}(1) \tag{3}$$

The accumulated number of paths at the level i is decreasing as i is increasing. From the definition of the C^i's it is clear that $N^n = 1$ and therefore,

Lemma 1 *The number of paths of the BDD equals C^n.*

Moreover,

Theorem 2 *Let $R^{c,i}$ stands for the weighted autocorrelation of the folded function f^i at the level i with weights $\{c_u^i\}$ as defined in (2). Then the number of paths in the corresponding BDD equals to*

$$C^n = 2^n - 0.5 \sum_{i=0}^{n-1} R^{c,i}(1). \tag{4}$$

In other words, the number of paths is directly related to the autocorrelation function, and the higher the weighted autocorrelations values at the corresponding δ_0 are, the smaller the number of paths.

4 Minimization of the number of paths

A linear transform enables implementation of f as a superposition of a linear transform function σ followed by a non-linear part, f_σ, such that $f(x) = f_\sigma(\sigma(x))$. The linearization problem is to find a non-singular matrix σ such that the number of paths is reduced. From Theorem 2, the accumulated number of paths in the BDD corresponding to f_σ is $C^n = 2^n - 0.5\sum_{i=0}^{n-1} R^{c,i}_{f_\sigma}(\delta_0)$, therefore, the optimal σ has the property that the weighted sum of the autocorrelation functions $R^{c,i}_{f_\sigma}(\delta_0)$ of the folded linearized function f^i_σ is maximal.

Let $T = (t_{n-1}, \dots t_1, t_0) = \sigma^{-1}$, the weighted autocorrelation of the linearized function is

$$R^c_{f_\sigma}(x) = R^c_f(\sigma^{-1}x) = R^c_f(Tx).$$

Since

$$R^c_{f_\sigma}(1) = R^c_f(T\delta_0) = R^c_f(t_0),$$

the optimization problem can be formulated as follows:

Construct a set of vectors $\tau_i \in GF(2^{n-i})$, $i = 0, 1, \dots n-1$, for which $\sum_{i=0}^{n-1} R^{c,i}_f(\tau_i)$ is maximal.

The computational complexity of obtaining the optimal σ is NP-complete, therefore a greedy algorithm is to be used. The suggested algorithm is similar to the K-procedure presented at [4] for reducing the number of nodes in a BDD. The suggested algorithm starts from the bottom of the binary tree, and constructs a set of n base vectors. At each level, a vector τ that carries the maximal weighted autocorrelation value is chosen, and the corresponding $\hat{\tau}$ is added to the set instead of one of the original vectors. In other words, at the level i, a vector $\hat{\tau_i} = (\tau_i, 0, 0 \dots 0)$, $\tau_i \in GF(2^{n-i})$, may replace a base vector δ_k , $k \geq i$ if its weighted autocorrelation satisfies

$$R^{w,i}(\tau_i) \geq R^{w,i}(\tau)$$

for all $\tau \in GF(2^{n-i})$.

The linearization procedure for reduction of the number of paths:

Set $c^0_u = 1$ for all $u \in GF(2^k)$
Set $i = 0$

a. For all $\tau \in GF(2^{n-i})$ calculate the weighted autocorrelation function $R^{c,i}_f(\tau)$.

b. Determine τ that maximizes the weighted autocorrelation function. In the case that there are more than one τ, choose one randomly.

c. Replace the node variable at level i by τ and fold the BDD.

d. Update the set of weights according to Eq.(2).

e. Set $i = i + 1$ and repeat the procedure until $i = n - 1$ or $R_f(\tau) = 0$ for all calculated τ's.

5 Simulation results

In this section, we illustrate the theoretical results on a number of small MCNC benchmark functions. The corresponding software was developed by using standard packages in MatLab, however, existing programming packages for handling large switching functions can be equally applied without restrictions. The performance of various algorithms are compared in terms of the number of paths in the corresponding BDDs.
The columns of Table 2 are as follows; *orig* is the number of path of the original BDD. *ordered* is the number of path when ordering the input variables by increasing autocorrelation values. This corresponds to the T with column vectors of the Hamming weight restricted to 1. *Kproc* stands for the number of paths in the BDD linearized using the K-procedure, and *weights* is the number of paths of the BDD linearized using the suggested algorithm with weighted autocorrelation.

In general, reordering of variables in a BDD almost always reduces the number of paths compared to the original BDD, and the linearization almost always reduces the number of paths compared to the reordered BDD. The linearization algorithms, i.e. the K-procedure and the suggested algorithm with the weighted autocorrelation, are greedy, hence, for small number of inputs, the accumulated weights are not dominant and therefore the differences between *Kproc* and *weights* are negligible.

Table 2: Number of paths in BDDs of benchmark functions

Benchmark	n	k	*orig*	*ordered*	*Kproc*	*weights*
clip	9	5	454	468	204	204
9sym	9	1	220	220	58	58
dk27	9	9	86	47	47	47
sao2	10	4	237	95	88	89
alu2	10	8	581	407	407	407
alu3	10	8	707	478	478	487
dk17	10	11	377	106	107	107
apla	10	12	264	161	168	166
add6	12	7	4096	4096	729	729
alu1	12	8	1754	1468	1468	1387
misex3c	14	14	15288	8924	8945	8882

6 Conclusions

This paper deals with a technique for minimizing the number of paths in a BDD. We showed that the number of paths of a BDD is a linear function of a weighted autocorrelation function. Therefore, the number of paths can be reduced by the linearization with respect to the values of the autocorrelation function. The minimization objective is to construct a new set of base vectors having maximal weighted autocorrelation values. These vectors define the linear transform matrix σ and a linearized BDD having minimal number of paths.

A greedy deterministic procedure was suggested for deriving the linear transform matrix σ and the linearized function f_σ. When σ is a permutation matrix, the linearization is equivalent to reordering. The proposed method increases the number of possible ways to reorder elements in the truth vector of f, compared to the reorderings induced by reordering of variables, and due to that, may further reduce the number of paths in BDDs. In addition, the method is effective also in the case of symmetric functions, where reordering of variables cannot be used. The suggested approach shows significant reduction in the number of paths of the linearized BDD in comparison to the number of paths in the original BDDs for standard benchmark functions.

References

[1] E. V. Dubrova D.M. Miller, On Disjoint Covers and ROBDD Size, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 1999, pp. 162-164.

[2] G. Fey and R. Drechsler, Minimizing the number of paths in BDDs, In Symposium on Integrated Circuits and System Design 2002, pp. 359-364, 2002.

[3] M. Hilgemeier N. Drechsler and R. Drechsler, Minimizing the Number of One-Paths in BDDs by an Evolutionary Algorithm, The Congress on Evolutionary Computation 2003,Vol.3, 2003, pp. 1724- 1731.

[4] M.G. Karpovsky, R.S. Stankovic, J.T. Astola, Reduction of sizes of decision diagrams by autocorrelation functions, *IEEE Trans. on Computers*, Vol. 52, No. 5, 2003, pp.592-606.

[5] M.G. Karpovsky, *Finite orthogonal series in the design of digital devices*, New York, Wiley, 1976.

[6] O. Keren and I. Levin, Linearization of the Logic Functions Defined in SOP Form, UROMICRO SEAA / DSD 2005 Porto (Portugal) , 2005 .

[7] D. M. Miller, R. Drechsler and M. A. Thornton, *Spectral Techniques in VLSI CAD*, Kluwer Academic Pub, 2001.

[8] S. Reda, A. Orailoglu and R. Drechsler, On the Relation between SAT and BDDs for Equivalence Checking, International Symposium on Quality Electronic Design, March 2002