International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 29773-29782

http://www.ripublication.com

An Efficient Method For Computing Total Autocorrelation Over Shared Binary Decision Diagram

S. R. Malathi
Department of Electronics and Communication Engg.,
Sri Venkateswara College of Engineering
Chennai, Tamilnadu, India, malathiraj@svce.ac.in
P. Sakthivel
Department of Electronics and Communication Engg., College of Engineering, Anna University, Chennai, Tamil nadu, India, psv@annauniv.edu

Abstract

This paper describes about an efficient total autocorrelation computation method for large multiple output Boolean functions over a Shared Binary Decision Diagram (SBDD). The existing method performs the computation in a single traversal of SBDD, using an additional memory function kept in the hash table for storing the results of computation of autocorrelation between two sub-diagrams in the SBDD. During the single traversal of the SBDD, autocorrelation for each of the nodes representing the sub-diagrams is computed at least once. The proposed method computes the total autocorrelation coefficient using 'Root Node Memory Function', which stores all the cubes (minterms) representing the 'Then' sub-diagram function and 'Else' sub-diagram function for each of the functions in the multiple output Boolean function. The cubes representing the shifted function are compared with the cubes representing the original function and total autocorrelation coefficient is computed. The amount of computation required for autocorrelation coefficient is not dependent on the number of nodes in the SBDD as they do in the existing method. This makes the method an efficient and feasible one in practical applications. The effectiveness of the method is evaluated using the experimental results.

Keywords: SBDD; switching theory; autocorrelation; mutiple output functions

Introduction

In order to achieve a more global view of the switching functions, transforms such as Hadamard and Rademacher-Walsh are applied [1]. Applications of these transforms

in digital logic are well researched compared to the other transforms, such as the autocorrelation transform. The autocorrelation transform has been used in various areas including optimization and synthesis of combinational logic [2], variable ordering for Binary Decision Diagrams [3], and to compute the estimate of a function's complexity [4]. An analytic approach for the paths-reduction problem is provided in [5] based on the autocorrelation coefficient. In this, an explicit expression for the number of paths in decision diagrams as a linear function of so-called weighted autocorrelation coefficients associated with the basis vectors that span the finite Galois field GF(2^n) is framed.

Various applications in areas of communication and cryptography use Binary sequences with optimal autocorrelation property. Special sequences like *m*-sequence, generalized GMW sequences, and Legendre sequence [6], interleaved structure of sequence [7], Binary sequences of period 4N [8], all aim on optimal autocorrelation property. When a designer is given a function to work with for which no information about the function's use or structure is provided, the values of the autocorrelation coefficients for the switching function [9] may provide the designer with some information about the type of function with which they are working. In these applications, it is required to have an efficient method for computation of autocorrelation of given binary sequences and check their properties.

The methods for computing the autocorrelation coefficients are exponential in the number of inputs to the function(s). Hence in practice, the complexity of computing the autocorrelation is the main problem. New methods for computation of autocorrelation coefficient have been introduced in [10] and [11].

The problem of computing the autocorrelation coefficient for binary sequences can be overcome up to some reasonable size, if it is viewed as truth vectors of Boolean functions. Under these circumstances, Binary Decision Diagram (BDD) is used as the data structure to perform the computations.

Multi-output functions are widely used in practical applications. The total autocorrelation of the function in such cases is computed as the sum of the autocorrelations for separate outputs. That is, the algorithm has to be performed *k* times for computing the autocorrelation of a function with *k* outputs. The generalized BDD based algorithm for computing total autocorrelation to multiple output functions represented by Shared BDDs (SBDD), in a single traversal is described in [12]. This modified traversal of the sub-diagrams in SBDD makes use of an additional autocorrelation memory function kept in the separate hash table, in order to avoid repeated computation over shared sub-diagrams for different outputs.

In this paper, an improved and efficient method to compute total autocorrelation of SBDD using a 'Root Node Memory Function' is discussed. The proposed method avoids using additional autocorrelation memory functions needed in the hash table to save the results of the autocorrelation between sub-functions represented by sub-diagrams in the SBDD. Computation of autocorrelation for each function f_i is performed only at the corresponding root node of the function for the 'Then' and 'Else' sub-diagrams. Hence the complexity of this method is independent of the number of nodes in the SBDD.

The efficiency of the proposed method is estimated by extending the standard BDD package [13] framework, with a procedure for calculating the autocorrelation over SBDD. Experimental results using the benchmark functions for switching theory and logic design, confirm the efficiency of the proposed method.

Background

A. Total Autocorrelation Function

Application of the autocorrelation transform to a switching function results in a comparison of the function to itself, shifted by a specified amount. The autocorrelation transform is a special case of the correlation transform, which is defined as [14]:

$$B^{fg}(\tau) = \sum_{x=0}^{2^n-1} f(x).g(x \oplus \tau) \tag{1}$$

If f and g are the same function, then this becomes the autocorrelation transform also called the cross-correlation, or convolution function. To refer to the autocorrelation transform, the superscript is generally omitted. Here autocorrelation coefficient B(τ) is evaluated with f in the Boolean domain of {0,1}.

The total autocorrelation function B(τ) for the multiple output function $f = (f_1 f_2 ... f_k)$ is defined as the sum of autocorrelation functions for the outputs $B_i(\tau)$ [10], [11], [12].

$$B(\tau) = \sum_{i=1}^{k} B_i(\tau) = \sum_{i=1}^{k} \sum_{x=} \tag{2}$$

The functions f_i and their autocorrelation coefficients B_i are represented by vectors in matrix notation, $F_i = [f_i(0), f_i(1),. . . . , f_i(2^n\text{-}1)]^T$ and $B_i = [B_i(0), B_i(1),. . . . , B_i(2^n\text{-}1)]^T$, respectively. The autocorrelation coefficients are computed by using the autocorrelation coefficient matrix $R_i(n)$ as [11];

$$B_i = R_i(n)\, F_i \tag{3}$$

Where the matrix $R_i(n)$ consists of:

$$R_i(n) = \begin{matrix} f_i(0 \oplus 0) & \dots & f_i(0 \oplus 2^n - 1) \\ f_i(1 \oplus 0) & \dots & f_i(1 \oplus 2^n - 1) \\ \vdots & & \vdots \\ f_i(2^n - 1 \oplus 0) & \dots & f_i(2^n - 1 \oplus 2^n - 1) \end{matrix} \tag{4}$$

The first row of $R_i(n)$ shows the function values of $f_i(x)$. The other rows show the shifted values of the function $f_i(x \oplus \tau)$, for τ = (1, 2, ….. 2^n-1). It is shown in [11] that the shifting of the argument corresponds to the permutation of labels at the edges of certain nodes in the BDD. This feature is exploited in computing the autocorrelation over SBDD.

B. Shared Binary Decision Diagram

Shared Binary Decision Diagram (SBDD) is an effective way of representing multiple output switching functions. In an SBDD, the output of a particular switching function

is derived by sharing isomorphic sub-diagrams in the BDD [15].The main feature of SBDDs is illustrated by the following example.

Example-1: For a two output, four variable function $f = (f_1, f_2)$, where f_1 and f_2 are defined by the truth vectors F_1 = [0000, 0011, 0100, 0111, 1000]^T, F_2 = [0011, 0100, 1000, 1100]^T , the corresponding SBDD is shown in Fig. 1.

The main feature of SBDD is that it can compactly represent multiple output functions with large number of identical sub-functions, which is illustrated in this example. It is this feature that is highly exploited in computing the autocorrelation over SBDDs.

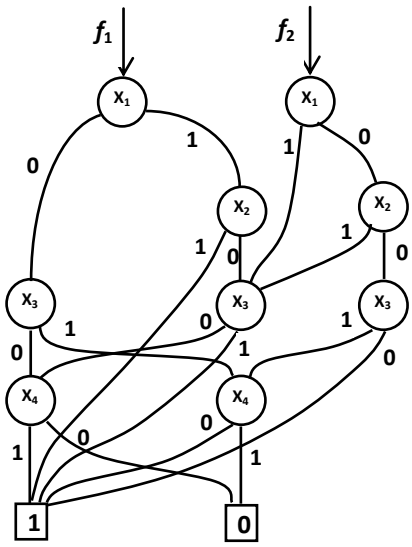

Figure 1: SBDD of $f = (f_1, f_2)$

C. Autocorrelation Coefficient Computation using BDD Method

It requires an exponential number of calculations in the number of variables to compute the autocorrelation coefficients for a given function using the Equation (2). But, the computation complexity of BDD based method is proportional to the size of the diagram (number of nodes). The corresponding BDD based methods to compute the autocorrelation coefficient are discussed as the recursive BDD method in [10] and the in-place BDD method in [11]. Where, using the single BDD for $f(x)$, the shifted function $f(x \oplus \tau)$ is determined with permuted labels at the edges of all the nodes for the variables x_i whose indices i correspond to the 1-bits in the binary expansion of $\tau = \tau_1 \tau_2 \ldots \tau_n$.

Example-2: Fig. 2 shows the BDD of the function $f_1(x) = f_1(x_1 x_2 x_3 x_4)$ in Example 1, defined by the truth vector F_1 = [0000, 0011, 0100, 0111, 1000]^T. Fig. 3 shows the BDD of the shifted function $f_1(x \oplus 4)$. For $\tau = 4 = (0100)_2$ over the BDD of $f_1(x)$ the values of $f_1(x \oplus 4)$ are determined by exchanging the traversal over the outgoing edges for the nodes x_3, since the nonzero bit in $\tau = (0100)_2$corresponds to the variable x_2. The traversal over the BDD from the root node to the constant node for $f_1(x)$ and $f_1(x \oplus 4)$ are labelled as (a) and (b) in Fig. 2 and Fig. 3 respectively.

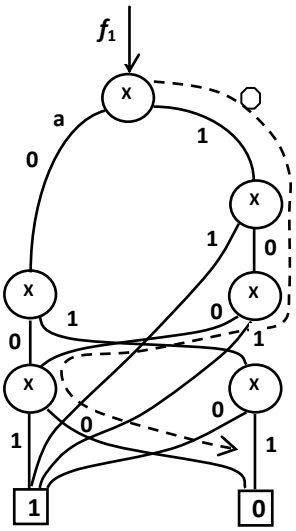

Figure 2: BDD of the function f_1(x)

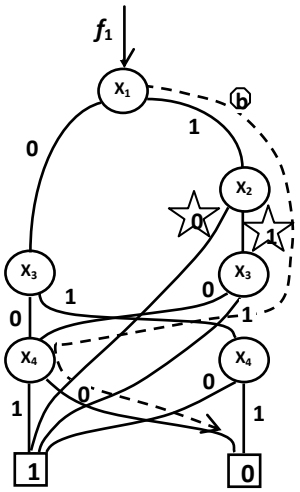

Figure 3: BDD of the function $f_1(x \oplus 4)$

D. Autocorrelation Coefficient Computation using SBDD Traversal Method

Additional autocorrelation memory functions are used in the hash table [12] to speed up the recursive traversal of the SBDD. These memory functions are used to save the results of the computation of the autocorrelation between sub-functions represented by sub-diagrams in the SBDD. Each unique autocorrelation memory function has an entry in the hash table. Before a new autocorrelation value is calculated, the hash table is looked up to determine if the result of the autocorrelation already exists. If so, the existing result is used. In [12] the autocorrelation memory function is represented by the triple (original, shifted, result), where result denotes the integer result of the autocorrelation performed on the two input arguments original (sub-diagram of SBDD($f(x)$)) and shifted (sub-diagram of SBDD($f(x \oplus 4)$)). The autocorrelation

coefficients for sub-functions represented by all the nodes in the SBDD are determined. The required value is the autocorrelation memory function B(τ) for the root node. Therefore the complexity of this method is proportional to the number of nodes in the SBDD for a multi output function. Fig. 4 shows the computation of the autocorrelation coefficient B(4) using the BDD for the function $f_1(x)$ in Example 2 with the autocorrelation memory functions (ACMF)assigned to corresponding pairs of sub-diagrams. The fields that store the original and shifted integer values are represented by dots and the result B(τ) that should be computed for the sub-diagram rooted in the considered node is shown.

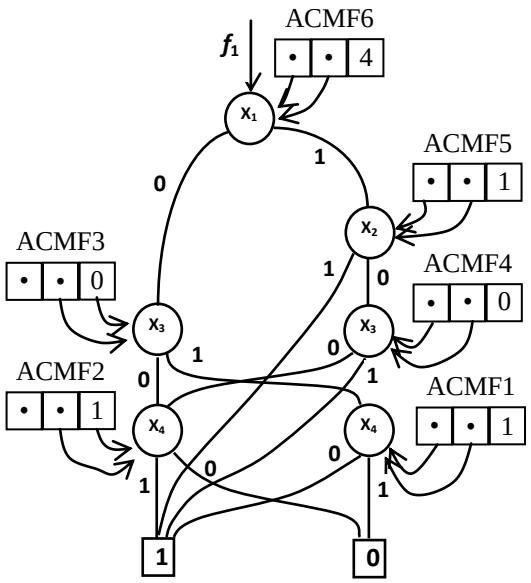

Figure 4: BDD of the function $f_1(x)$

Autocorrelation Coefficient Computation Using Root Node Memory Function (RNMF)

Given an SBDD for the function $f(x_1\ x_2 \ldots x_n) = f_1(x_1\ x_2 \ldots x_n), \ldots\ldots, f_k(x_1\ x_2 \ldots x_n)$, the autocorrelation coefficient can be computed for each function and added to get the total autocorrelation coefficient. The procedure to compute the autocorrelation coefficient for SBDD using the Root Node Memory Function is discussed below.

Procedure: Root Node Memory Function (RNMF) method

1. Traverse from a root node of the given SBDD and determine all the cubes (minterms) of the function at the corresponding root node. A Root Node Memory Function Table is used to save the corresponding cubes of each function.
2. Divide the set of cubes representing the function $f_i(x_1\ x_2 \ldots x_n)$, into 'Then' sub-diagram cubes $f_{Ti}(x_1\ x_2 \ldots x_n)$ and 'Else' sub-diagram cubes $f_{Ei}(x_1\ x_2 \ldots$

x_n). [In a BDD all the cubes of a function $f_i(x_1\ x_2\ .\ .\ .\ x_n)$ starting with '1', represents the 'Then' sub-diagram and all those starting with '0' represents the 'Else' sub-diagram]

3. Mark all the nodes in the SBDD for the variables x_i whose indices correspond to the 1-bits in the binary expansion of autocorrelation coefficient $\tau = \tau_1\tau_2. . . \tau_n$ and permute the labels at the outgoing edges of the nodes x_i .
4. *Shifted*'Then' sub-diagram cubes $f_{Ti}((x_1\ x_2\ .\ .\ .\ x_n)\oplus(\tau_1\tau_2. . . \tau_n))$ and 'Else' sub-diagram cubes $f_{Ei}((x_1\ x_2\ .\ .\ .\ x_n)\oplus(\tau_1\tau_2. . . \tau_n))$ are computed by complementing the variables x_i whose indices correspond to the 1-bits in the binary expansion of autocorrelation coefficient $\tau = \tau_1\tau_2. . . \tau_n$.
5. Compare *original*'Then' sub-diagram cubes $f_{Ti}(x_1\ x_2\ .\ .\ .\ x_n)$ with the *Shifted*'Then' sub-diagram cubes $f_{Ti}((x_1\ x_2\ .\ .\ .\ x_n)\oplus(\tau_1\tau_2. . . \tau_n))$. Similarly compare *original* and *shifted*'Else' sub-diagram cubes.
6. Autocorrelation coefficient of a function f_i is equal to the sum of total number of similar (matching) cubes in the 'Then' and 'Else' sub-functions.
7. Repeat step-2 onwards for all the 'k' Root Nodes.
8. Total autocorrelation $B(\tau)$ of SBDD is $B(\tau) = B_1(\tau) + \ldots\ldots + B_k(\tau)$.

In this procedure autocorrelation coefficients for sub-functions represented by each of the node in the SBDD need not be determined. Hence additional autocorrelation memory functions needed in the hash table [12] to save the results of the autocorrelation between sub-functions represented by sub-diagrams in the SBDD are avoided. Computation of the autocorrelation for each function $f_i(x)$ is performed only at the corresponding Root Node of the function. Therefore, ignoring the negligible amount of time needed for comparing the cubes, the complexity of this procedure is independent of the number of nodes.

The effectiveness of the RNMF method is highlighted using an illustration. For the SBDD shown in Fig. 1, the RNMF procedure will result in the list of cubes for f_1, f_2 as shown in Table-1. The comparison between *original* and *shifted* sub-functions for 'Then' and 'Else' sub-diagrams results in the total autocorrelation coefficient $B(4)$ = 6.

Table 1: Computing The Total Autocorrelation Coefficient Using RNMF Method

Original and Shifted Sub-functions	**Cubes stored due to RNMF procedure**	**Number of similar cubes**	**Autocorrelation Coefficient**
$f_{T1}(x)$	1000	0	B1(4) = 4
$f_{T1}(x\oplus 4)$	1100		
$f_{E1}(x)$	0000, 0100, 0011, 0111	4	
$f_{E1}(x\oplus 4)$	0100, 0000, 0111, 0011		
$f_{T2}(x)$	1000, 1100	2	B2(4) = 2
$f_{T2}(x\oplus 4)$	1100, 1000		
$f_{E2}(x)$	0011, 0100	0	
$f_{E2}(x\oplus 4)$	0111, 0000		
Total Autocorrelation Coefficient			B(4) = 6

Experimental Results

The proposed method is evaluated by comparing the time to compute the autocorrelation coefficient with the existing in-place BDD method and the SBDD traversal method [12]. The standard BDD package [13] framework was extended, with a procedure for calculating the total autocorrelation over SBDD using C++. Standard benchmarks were used to test the proposed RNMF procedure on a Pentium IV PC, at 2.66 GHz speed and 4GB RAM. The time in seconds to compute the total autocorrelation coefficient includes the time to construct SBDD. Table-2 gives a comparison of the time to compute the autocorrelation coefficient with the existing in-place BDD method, the SBDD traversal method and the proposed RNMF method. The corresponding graphical representation is shown in Fig. 5.

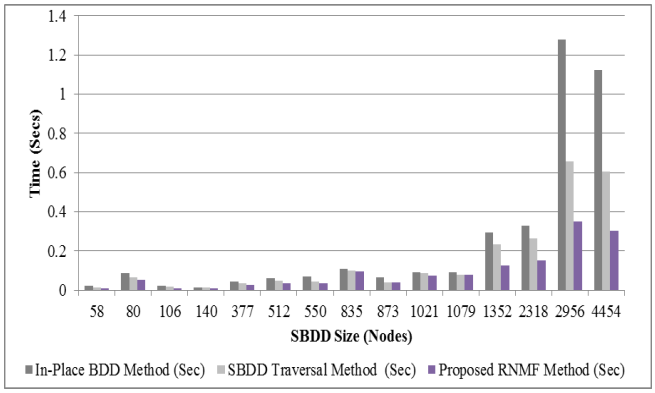

Figure 5: Comparison of the time to compute the autocorrelation coefficient

In Table-2, the data are in the increasing order of the number of SBDD nodes. Depending on the SBDD size the improvement in the speed of computing the autocorrelation coefficient varies from 7 to 50 percentages compared to the SBDD Traversal method. It can be observed that the speedup is very large for functions with nodes greater than 1000. Clearly the time to compute the autocorrelation coefficient is slightly greater than the time to construct the SBDD.

Table 2: Comparison of the total autocorrelation computation time between the existing In-place BDD method [11], SBDD Traversal method [12] and the proposed RNMF method

Bench-mark	**In**	**Out**	**SBDD Size (Nodes)**	**In-Place BDD Method (Sec)**	**SBDD Traversal Method (Sec)**	**Proposed RNMF Method (Sec)**
misj	35	14	58	0.021	0.015	0.011
cordic	23	2	80	0.089	0.067	0.055
b7	8	31	106	0.025	0.017	0.011
misex2	25	18	140	0.016	0.013	0.010
in3	35	29	377	0.046	0.036	0.028

b4	33	23	512	0.062	0.047	0.035
jbp	36	57	550	0.072	0.046	0.036
ibm	48	17	835	0.109	0.102	0.095
table5	17	15	873	0.064	0.041	0.038
apex4	9	19	1021	0.094	0.089	0.073
ex1010	10	10	1079	0.093	0.078	0.077
alu4	14	8	1352	0.296	0.234	0.128
cps	24	109	2318	0.327	0.265	0.154
signet	39	8	2956	1.276	0.658	0.349
b2	16	17	4454	1.123	0.605	0.304

Conclusion

In this paper, the problem of computing the autocorrelation coefficient for binary sequences of reasonable size is overcome up to some extent, by viewing the sequences as the truth vectors of Boolean functions. Shared Binary Decision Diagram (SBDD) is used as the data structure to perform the computations. The autocorrelation of multiple output function $f = (f_1, \ldots, f_k)$ is computed by introducing only 'k' Root Node Memory Function (RNMF). Whereas in 'SBDD Traversal Method [12],' ACMF for each node has to be computed once before the required final root node autocorrelation is computed. By the RNMF procedure, maintaining a hash table as in to store the autocorrelation of sub-functions represented by each node is avoided. The proposed RNMF procedure was evaluated by the experiments conducted on benchmark functions and the efficiency confirmed. Experiments show that the time to compute the autocorrelation coefficient is slightly greater than the time to construct the SBDD it self, from which it is evident that no significant additional time is required to compute the autocorrelation compared to other methods.

References

[1]. S. L. Hurst, D. M. Miller, and J. C. Muzio, Spectral Techniques in Digital Logic.Orlando, Florida: Academic Press, Inc., 1985.

[2]. R. Tomczuk, "Autocorrelation and Decomposition Methods in Combinational Logic Design," Ph.D. dissertation, University of Victoria, 1996.

[3]. J. E. Rice, J. C. Muzio, and M. Serra, "The Use of Autocorrelation Coefficients for Variable Ordering for ROBDDs," in Proceedings of the 4th International Workshop on Applications of th e Reed-M¨uller Expansion in Circuit Design, 1999.

[4]. M. Karpovsky, Finite Orthogonal Series in the Design of Digital Devices. John Wiley & Sons, 1976.

[5]. O. Keren, I. Levin, and R. S. Stankovi´c, "Determining the number of paths in decision diagrams by using autocorrelation coefficients," IEEE

Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 30, no. 1, pp. 31–44, Jan. 2011.

[6]. S. W. Golomb and G. Gong, Signal Design for Good Correlation for Wireless Communication , Cryptography, and Radar, Cambridge University Press, Cambridge, 2005.

[7]. G. Gong, "Theory and applications of q-ary interleaved sequences," IEEE Trans. Inf. Theory, vol.41, no.2, pp.400-411, March 1995.

[8]. X. Ma, Q. Wen, J. Zhang, and X. Zhang, "New constructions of binary sequences with good autocorrelation based on interleaving technique," IEICE Trans. Fundamentals, vol. E94-A, no.12, pp.2874-2878, Dec. 2011.

[9]. J. E. Rice, "Autocorrelation Coefficients in the Representation and Classification of Switching Functions," Ph.D. dissertation, University of Victoria, 2003.

[10]. J. E. Rice and J. C. Muzio, "Methods for calculating autocorrelation coefficients," in Proc. 4th Int. Workshop on Boolean Problems, pp. 69–76, Freiberg, Germany, Sept. 2000.

[11]. R. S. Stanković and M. G. Karpovsky, "Remarks on calculation of autocorrelation on finite dyadic groups by local transformations of decision diagrams," Lect. Notes Comput. Sci., vol.3643, pp.301-310, Springer Berlin, 2005.

[12]. M. Radmanović, R. S. Stanković, and C. Moraga, "Computation of the Total Autocorrelation over Shared Binary Decision Diagrams," IEICE Trans. Fundamentals, vol. E97-A, No.5, pp.1140-1143, May. 2014.

[13]. Fabio Somenzi. "CUDD: Colorado UniversityDecision Diagram package"University of Colorado at Boulder, http://vlsi.colorado.edu/ fabio/CUDD/.

[14]. M. Karpovsky, Finite Orthogonal Series in the Design of Digital Devices. John Wiley & Sons, 1976

[15]. T. Sasao and M. Fujita, Representations of Discrete Function, Kluwer Academic Publishers, Boston, 1996.