FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 28, N^o 4, December 2015, pp. 597 - 609
DOI: 10.2298/FUEE1504597R

EFFICIENT CALCULATION OF THE AUTOCORRELATION OF BOOLEAN FUNCTIONS WITH A LARGE NUMBER OF VARIABLES

Miloš Radmanović[1], Radomir Stanković[1], Claudio Moraga[2]

[1]Faculty of Electrical Engineering, University of Niš, Niš, Serbia
[2]European Centre for Soft Computing, Mieres, Spain, and Department of Computer Science, Technical University of Dortmund, Dortmund, Germany

Abstract. *The autocorrelation of a Boolean function is an important mathematical concept with various applications. It is a kernel of many algorithms with essential applications whose efficiency is directly limited by the time and space complexity of methods for computing the autocorrelation. These limitations, in this paper, can be overcome by computing the autocorrelation using a Shared Multi-Terminal Binary Decision Diagram (SMTBDD) which is a data structure allowing compact representations of large Boolean functions. The computation is performed in the spectral domain by exploiting the Wiener-Khinchin theorem and the fast calculation algorithms through SMTBDDs. It is necessary to develop a specialized decision diagram package with all the standard BDD operations that supports a fast calculation algorithms through decision diagrams with dynamically resizable terminal nodes. It allows to deal with large integers that appear in computing the autocorrelation coefficients. An experimental evaluation over benchmarks favorably confirms the efficiency of the proposed data structure and related algorithm.*

Key words: *Boolean functions, autocorrelation, Wiener-Khinchin theorem, fast Walsh transform, BDD package, dynamically resizable terminal nodes.*

1. INTRODUCTION

The autocorrelation function has numerous applications in computing, telecommunications, data encoding and transmission, cryptography, etc. In particular, in computer-aided design, the autocorrelation is used in the optimization and synthesis of combinational logic [1-5], variable ordering for binary decision diagrams [6-9], and estimation the function complexity [10]. The related algorithms are deterministic and, for the classes of Boolean functions where they can be efficiently applied (depending on the properties of autocorrelation

Received November 26, 2014; received in revised form March 20, 2015
Corresponding author: Miloš Radmanović
Faculty of Electrical Engineering, University of Niš, Alaksandra Medvedeva 14, 18000 Niš, Serbia
(e-mail: milos.radmanovic@elfak.ni.ac.rs)

coefficients), the produced solutions are optimal. This can be considered an advantage compared to various heuristic approaches that have been proposed for the same applications, see, e.g., [11] and references therein, since in heuristic algorithms there is no guarantee on the obtained quality. The efficiency of algorithms based on autocorrelation is directly determined by the runtime of methods used for computing the autocorrelation as well as the complexity of the underlaid data structures used to represent the input and output data. In vector notation, the autocorrelation of a function of n variables is as a vector of length 2^n. Therefore, methods for computing the autocorrelation coefficients have an exponential complexity in the number of the variables. There are various methods for an efficient computation of individual autocorrelation coefficients of a given function depending on the data structure used to specify the Boolean function and its autocorrelation [12, 13].

The autocorrelation coefficients may also be computed from the spectral coefficients of the function by exploiting the Wiener-Khinchin theorem and the fast calculation algorithms through Multi-Terminal Binary Decision Diagrams (MTBDDs) [14]. The method may be extended to the computation of the autocorrelation for multiple-output functions. Since this method produces large integers up to the value 2^{2n}, where n is the number of variables in the function, currently available BDD based techniques are limited to functions of less than 32 variables [15].

In this paper, we present a method for the computation of the autocorrelation spectra through SMTBDDs for multiple-output Boolean functions of more than 32 variables. The computation is performed in the spectral domain by exploiting the Wiener-Khinchin theorem and the fast calculation algorithm through SMTBDDs [16]. This computation, for Boolean functions of many variables, requires calculations of large integers. For this reason, the standard BDD packages [17-22] with integer data type terminal nodes cannot be used. Experiments with a package with integer data type terminal nodes show that it is necessary to develop a package that will preserve all the standard BDD operations necessary to perform the corresponding fast calculation algorithms through decision diagrams, however, with dynamically resizable terminal nodes. Being developed by appreciating and incorporating all the standard techniques in programming decision diagrams, the specialized decision diagram package presented in this paper can be viewed as an extension of the classic BDD packages. However, it allows dealing with large integer terminal nodes. This feature was achieved by incorporating in the decision diagram package the template class "BitVector" used to define the dynamically resizable terminal nodes. The size of the node can be specified by the user as a template parameter. To estimate features of the package, we show, by experiments on benchmarks, that the proposed implementation allows us the computation of the autocorrelation of multiple-output Boolean functions with a large number of variables (over 32 variables), while the application of the decision diagram packages with integer data type terminal nodes restricts the computation of autocorrelations to Boolean functions with 32 or less variables.

The paper is organized as follows: The second section reviews basic properties of the autocorrelation of Boolean functions. The third section describes SMTBDDs and a fast calculation algorithm through SMTBDDs. The fourth section discusses computation of the autocorrelation coefficients through SMTBDDs. Section five briefly describes how the classic BDD packages can be extended into a specialized decision diagram package with dynamically resizable terminal nodes. Section six illustrates, based on benchmarks, that the proposed extension of the classical BDD packages allows the computation of the autocorrelation coefficients for Boolean functions of many variables. Furthermore, some peculiar properties for the discussed computations are pointed out and illustrated. The paper concludes with a discussion of possible directions for future research.

2. AUTOCORRELATION FUNCTION

The following notation is used throughout the paper. A binary n-tuple $x_1x_2...x_n$, $x_i \in \{0,1\}$ is denoted by x and the equivalent integer value is assigned to it as:

$$\mathrm{x} = \sum_{i=1}^{n} x_i 2^{n-i} \ . \tag{1}$$

With this notation, $f(x)$ is an n-variable Boolean function, i.e., $f(x) = f(x_1x_2...x_n)$, $x_i \in \{0,1\}$, $x \in \{0\ldots00, 0\ldots01, \ldots, 1\ldots11\}$. An m-output Boolean function is defined as the function $f(x) = f(f_0, f_1, f_0,..., f_{m-1})$: $\{0,1\}^n \rightarrow \{0,1\}^m$.

The autocorrelation function is defined as [1]:

$$B(u) = \sum_{v=0}^{2^n-1} f(v) f(v \oplus u), \qquad u \in \{0,1,...,2^n-1\} , \tag{2}$$

where $\oplus$ is the addition modulo 2, EXOR.

The autocorrelation function computes a measure of similarity between a function f and the same function under displacement. The autocorrelation function (or transform), of a Boolean function is an integer valued function. We assume that the Boolean functions are represented by BDDs, while the autocorrelation functions, being integer valued, are represented by MTBDDs. For multi-output functions the Shared BDDs (SBBDs) and SMTBDDs are used.

It should be noticed that the maximal value of an autocorrelation coefficient can be 2^n. This can be a source of difficulties when computing the autocorrelation of Boolean functions with a large number of variables and representing the autocorrelation function by decision diagrams since representing large integers in terminal nodes is required. For example, a Boolean function of 65 variables might have autocorrelation coefficients whose value could be $2^{65} \approx 3.68 \cdot 10^{19}$. This problem is addressed in the paper and a solution is proposed through decision diagrams with dynamically resizable terminal nodes by using a particular technique of object oriented programming languages.

For multiple-output functions $f = (f_0, f_1, f_0,..., f_{m-1})$ the autocorrelation functions of the individual outputs are combined into the total autocorrelation function [1]:

$$B(u) = \sum_{i=0}^{m-1} B_i(u) = \sum_{i=0}^{m-1} \sum_{v=0}^{2^n-1} f_i(v) f_i(v \oplus u) \ . \tag{3}$$

As evident from previous equations, computing the autocorrelation coefficients requires 2^n operations to compute each of the 2^n coefficients. Therefore, the run-time and computational resources are exponential in n.

For a function f defined by the truth-vector $F = [f(0), (1), \ldots, f(2^n-1)]^T$, the Walsh spectrum $S_f = [S_f(0), S_f(1), \ldots, S_f(2^n-1)]^T$ is defined as [22]:

$$S_f = W(n)F\ ,\tag{4}$$

where,

$$W(n) = \bigotimes_{i=1}^{n} W(1)\ ,\tag{5}$$

where $\otimes$ denotes the Kronecker product, and

$$W(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\tag{6}$$

is the basic Walsh matrix.

The Walsh transform is a self-inverse transform up to the constant 2^{-n} that is used as the normalization factor when defining the Walsh transform and its inverse.

In the matrix notation, if a function f and its autocorrelation function $B(u)$ are represented by vectors $F = [f(0),(1),\ldots,f(2^n-1)]^T$, and $B_f = [B(0),B(1),\ldots,B(2^n-1)]^T$, respectively, the Wiener-Khinchin theorem is defined as [1]:

$$B_f = 2^{-n}W(n)(W(n)F)^2\ .\tag{7}$$

The main advantage of this theorem comes from the existence of the Fast Walsh transform (FWT) that is an algorithm to compute the Walsh spectrum with logarithmic time complexity. Since the FWT can be performed also over decision diagrams [16], the Wiener-Khinchin theorem can be used in computing the autocorrelation function of Boolean functions with a large number of variables. It should be noticed that the maximal value of an autocorrelation coefficient, when using the Wiener-Khinchin theorem, before multiplication with 2^{-2n} could be 2^{2n}. For example, a function of 65 variables might produce the value $2^{130} \approx 1.36\cdot 10^{39}$. Again we see the problem of large integers that should be represented in terminal nodes of decision diagrams.

3. SMTBDDs and a Fast Calculation Algorithm through SMTBDD

A BDD is a data structure convenient to represent a Boolean functions of many variables. Due to that, BDDs have become widely used for a variety of CAD applications, for example in [23-24], including symbolic simulation, verification, reliability analysis of combinational and sequential circuits.

An MTBDD is a generalization of a BDD, derived by allowing terminal nodes that show integer values [25]. A comprehensive set of arithmetic operations can be realized efficiently on MTBDDs, such as addition, subtraction, and multiplication, as well as logic operations. They are implemented by recursive algorithms and executed in time almost linear in the graph size [16]. Multiple-output integer-valued functions are represented by SMTBDDs, having a separate root node for the each output [22]. The Walsh spectrum of a Boolean function (if the scaling factor 2^{-n} is assigned to the inverse transform) is an integer-valued function and can be represented by an MTBDD.

Example 1: Fig. 1 shows the SMTBDD for the Walsh spectra of the functions $f_1(x_1,x_2,x_3) = x_1 + x_2\bar{x}_3$ and $f_2(x_1,x_2,x_3) = x_1x_2 + x_2\bar{x}_3$. The Walsh spectrum of the function f_1 is S_{f_1} = [5,1,–1,–1,–3,1, –1,–1]^T and that of the function f_2 is S_{f_2} = [3,1,–3,–1,–1,1,1,–1]^T. It should be observed that the MTBDD for S_{f_1} is compact since there are two constant subvectors of two consecutive –1 in the Walsh spectrum of f_1. It is obvious that this SMTBDD is much smaller than two MTBDDs for the functions f_1 and f_2 in the number of nodes since there are shared values of the Walsh coefficients of f_1 and f_2.

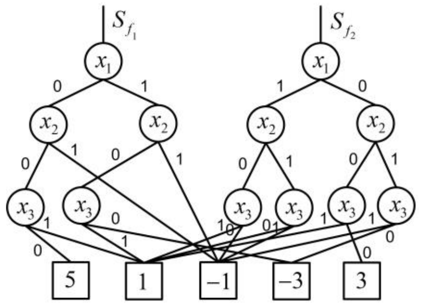
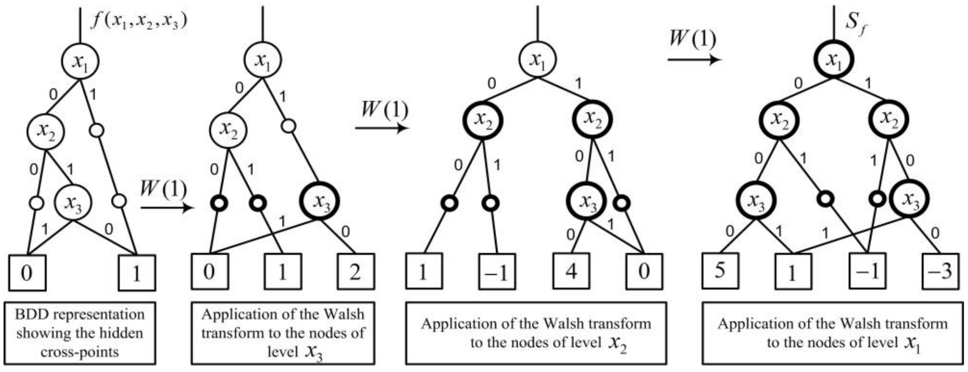


Fig. 1 SMTBDD for the Walsh spectra for the functions in Example 1

The algorithm, we refer to as the fast calculation algorithm for the Walsh transform [1] through an SMTBDD, is based on the fast algorithms for spectral transforms through BDDs. Several variants of BDD based calculation algorithms for Walsh transform as well as extensions of the BDD calculation methods to other transforms for the Boolean functions are considered in [19], [22], [25], [26], and elsewhere. The algorithm is based upon the factorization of transform matrices as used in the development of the Fast Fourier transform (FFT). Butterfly operations are implemented in terms of graph addition and subtraction operations resulting in a technique that is implemented through the use of graph manipulations only. This method takes advantage of the compactness inherent in MTBDDs and can be more effective for Boolean function transformations than traditional approaches.

Example 2: The fast calculation algorithm for the Walsh spectrum through an MTBDD of the function $f(x_1,x_2,x_3)=x_1+x_2\bar{x}_3$ with Walsh spectrum $S_f = [5,1,–1,–1,–3,1, –1,–1]^T$ is shown in Fig. 2. An MTBDT (Multi-Terminal Binary Decision Tree) can be reduced into an MTBDD. The subtrees may be shared and the redundant information (nodes) deleted from the MTBDT. The impact of the deleted nodes can be represented by the cross-points defined as points where an edge crosses a level in the MTBDD [26]. In this introductory example, all cross-points or "hidden" nodes must be considered as explicitly present to apply the local Walsh transform. The Walsh transform algorithm is bottom up. The transform of the level corresponding to the variable x_3 has as effect, that for each node and cross-point, it replaces the subtree connected to a node with a 0-labeled edge, by the sum of the values of both subtrees of that node (which in this case are leaves) and the subtree connected to the same node with a 1-labeled edge, by the difference. The same procedure is applied step by step to nodes and cross-points in higher levels of the MTBDD. Computation of Walsh transform through an MTBDD by applying the traversal in a bottom-up manner is denoted as bolded nodes and cross-points.



$f(x_1,x_2,x_3)$
$W(1)$
$W(1)$
$W(1)$
S_f
BDD representation showing the hidden cross-points
Application of the Walsh transform to the nodes of level x_3
Application of the Walsh transform to the nodes of level x_2
Application of the Walsh transform to the nodes of level x_1

Fig. 2 Fast calculation algorithm for the Walsh spectrum of the function *f* in Example 2 through an MTBDD

This technique can be used for any transformation that has a Kronecker product based transformation matrix and can be extended to SMTBDDs [26].

4. Autocorrelation through an SMTBDD by Using the Wiener-Kinchin Theorem

The computation of the Walsh spectrum can be performed through the flow-graph describing the Fast Walsh transform (FWT).

Example 3: Computing the autocorrelation coefficients using the Wiener-Khinchin theorem as specified in Eq. (7) through the Fast Walsh transform for the function $f(x_1,x_2,x_3)=x_1+x_2\bar{x}_3$ is shown in Fig. 3. Step 1 is dedicated to calculate the Walsh spectrum $S_f = [5,1,-1,-1,-3,1,-1,-1]^T$ from the truth-vector of the function *f*. In the final step, the autocorrelation spectrum is multiplied by the normalization factor 1/8. Notice that normalizing after the computation of the Walsh transform allows us the use of the integer arithmetic in the whole process, even though at the price of accepting 2^{2n} as the upper boundary. Moving ahead the normalizing factor, would allow an upper bound of 2^n, but would require working with rational numbers and a complex floating point implementation. This is why the integer arithmetic version was adopted for the present work.

The Wiener-Khinchin theorem and the Walsh spectrum computation through an MTBDD [13] leads to the following algorithm for the computation of the autocorrelation over SMTBDDs.

$$\begin{array}{c} 0\\0\\1\\0\\1\\1\\1\\1 \end{array} \xrightarrow{FWT} \begin{array}{c} 5\\1\\-1\\-1\\-3\\1\\-1\\-1 \end{array} \xrightarrow{(\ )^2} \begin{array}{c} 25\\1\\1\\1\\9\\1\\1\\1 \end{array} \xrightarrow{FWT} \begin{array}{c} 40\\32\\32\\32\\16\\16\\16\\16 \end{array} \xrightarrow{1/8} \begin{array}{c} 5\\4\\4\\4\\2\\2\\2\\2 \end{array}$$

Fig. 3 Computing the autocorrelation for the function in Example 3

Algorithm 1. Autocorrelation spectrum through an SMTBDD

Let $SMTBDD(f)$ be the representation of multi-output function *f*. Suppose that function can be efficiently represented by SMTBDD.

1) Update the size of SMTBDD terminal node according with the maximal value of an autocorrelation coefficient.
2) Conversion of the $SMTBDD(f)$ into an $SMTBDD(S_f)$, where S_f denotes the Walsh spectra of the function *f*.
3) Multiplication of the $SMTBDD(S_f)$ by itself using the standard procedure for multiplication of functions represented by BDDs (see, e.g., [2]).
4) Conversion of the resulting $SMTBDD(S_f^2)$ into a new $SMTBDD(2^n B_f)$.
5) Normalization with 2^{-n}, since the Walsh matrix is self-inverse up to the constant 2^n, where n is the number of variables in the function *f*.

Example 4: Computing the autocorrelation coefficients using the Wiener-Khinchin theorem and the FWT through an SMTBDD for the functions $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ and $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$ is shown in Fig. 4. It is fairly obvious that the number of "butterflies" and multiplication operations in this SMTBDD is smaller than the number of operations in two MTBDDs for the functions f_1 and f_2, since there are shared subtrees in the MTBDD for the functions f_1 and f_2.

5. The BDD Package with Dynamically Resizable Terminal Nodes

5.1. Motivation

Decision diagrams are a standard part of many CAD-CAM systems since they permit compact representations of large Boolean functions and efficient manipulations and calculations with them.

There are several code packages and development environments using BDDs and their various generalizations and extensions, as the main data structure. These decision diagram packages are built in various programming languages, especially in C, C++, and Java. Basic principles in programming decision diagrams are set in [17] and then further elaborated by many authors, e.g., [11, 18, 21, 22, 27, 28], and references therein. Most packages appreciate these fundamental principles and share common features, however, a specification and suitable modification of the basic decision diagram packages is usually required to meet demands in particular concrete applications. The same is true when decision diagrams are used to compute the autocorrelation of Boolean functions. In this case, a particular problem is the requirement to deal with large integers.

It should be noticed that the maximal value of a terminal node, when performing the Wiener-Khinchin theorem and the FWT through an SMTBDD, before multiplication with 2^{-n} could be 2^{2n}, where n is the number of variables in the function. Therefore, if the computation of the autocorrelation coefficients is performed through the SMTBDD, the usability of classical BDD packages which have 32-bits or 64-bits integer terminal nodes

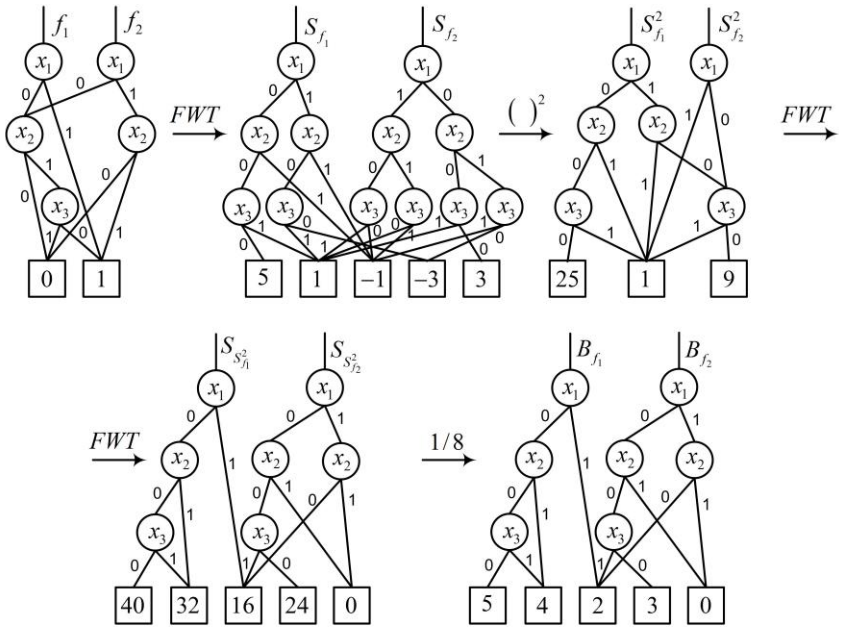


Fig. 4 Computing the autocorrelation for the multiple-output function in Example 4

is necessary limited to relatively moderate size Boolean functions (with 16 or 32 variables respectively). With this motivation, in this paper, we propose an extension of the classical BDD packages, with dynamically resizable terminal nodes, that allows computation of the autocorrelation for large Boolean functions. The idea comes from the consideration in [29] where decision diagrams with terminal nodes replaced by vectors are discussed. These vectors can be viewed as binary representations of large integers. In computing the autocorrelation, it is convenient to have flexibility in determining the size of the binary vectors corresponding to the binary representations of integers. This consideration leads to decision diagrams with dynamically resizable terminal nodes. We however appreciate and preserved all other basic recommendations in programming decision diagrams, as for instance implementation of the unique table, compute table, garbage collection, etc., and implemented them as in many other decision diagram packages. Therefore, a description of these features is omitted. Instead we refer to basic principles in programming decision diagrams, and focus on the modifications that we did in implementation of terminal nodes. Some other implementation details are briefly presented in Sect. 5.3 discussing computation of the autocorrelation functions.

5.2. Implementation of dynamically resizable terminal nodes

The class diagram for the static structure of nodes implementation for the BDD package with dynamically resizable nodes is shown in Fig. 5.

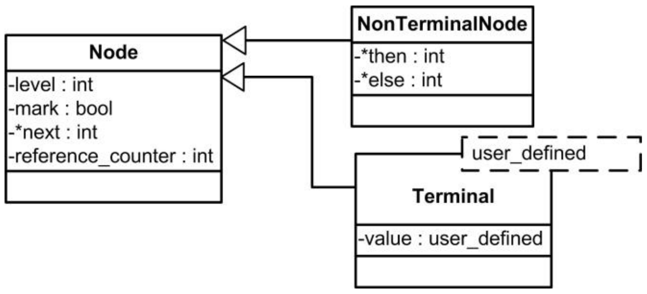


Fig. 5 BDD node implementation for the BDD package with dynamically resizable nodes

The class "Node" is used as basic class to represent a non-terminal or a terminal node. It is an object-oriented class that contains the attributes: "level", "next", and "reference counter". The "level" denotes a variable that labels the node and it uses the integer data type. The "next" pointer links nodes together that belong to the same level. The visited flag for a BDD traversal can be stored as the least significant bit of the "next" pointer. The "reference counter" is implemented for garbage collection of nodes and it uses the integer data type [17]. A non-terminal node is a class, derived from the class "Node", and contains the attributes: "then" and "else" children pointers. A terminal node is also a class, derived from the class "Node", and contains the attribute "value", which stores the constant value of the terminal node.

Object oriented languages allow us the definition of a template class to represent a class member of any possible datatype (including a user-defined datatype). In order to implement dynamically resizable terminal nodes, the "TerminalNode" class is implemented as a template class. The attribute "value" uses a template data type and can be of any possible datatype. The main program declares the used type of the attribute "value". This implementation for the attribute "value" allows a user-defined implementation of a class that supports work with large integers, where the length of large integers can be preset with a parameter.

5.3. Computation of the autocorrelation through SMTBDDs with dynamically resizable terminal nodes

To compute the autocorrelation coefficients of a large Boolean function in the case of restricted memory resources, we developed a BDD package with dynamically resizable nodes. The computation procedure is implemented in C++. The unique table and the operation table are implemented as a hash tables with collision chains [17]. The hash key is composed of the memory position of the node and its successors. Since this implementation uses the Walsh transformation over the BDD package, currently available techniques require the usage of the field "level" in the node implementation. For this reason, the unique table is not divided into subtables as proposed in [20]. Since this implementation uses BDD operations that require operations for user-defined types, the BDD operations are implemented as templates of overloaded operator functions.

For a user-defined implementation of the class that supports work with large integers, we developed the template class "BitVector<size>", where the size of the binary vector (large integer) can be preset with the template parameter “size”. Since this implementation uses BDD operations that require operations for the binary vectors, the class "BitVector" must support overloaded operator functions for assignments, relational, and arithmetic operators.

The implementation of the class "BitVector" uses standard programming technique for data structures.

Example 5: The implementation of the update of the size of SMTBDD terminal node where the maximal value of an terminal node can be 2^{128} (expressed in C++) uses the following two lines of code:
bdd_manager<BitVector<128>> manager;
smtbdd<BitVector<128>> bdd(manager);

In terms of BDD package implementation, it is common to use class “bdd_manager” for initialization of the BDD package. After this initialization a “bdd” object must be defined that handles the SMTBDD.

6. EXPERIMENTAL RESULTS

The autocorrelation computation was tested on a set of large benchmarks [30] on a PC Pentium IV running at 2,66 GHz with 4 GB of RAM. The size of the unique table and the operation table was limited to 262139 entries. The garbage collection was activated when available memory runs low. Computation run-time statistics of the BDD-based method includes the creation of the SMTBDDs. All benchmarks were used in the Espresso-mv or pla format [31].

Table 1 gives the experimental results of the computation run-time and the terminal node size of the autocorrelation computation through an SMTBDD with dynamically resizable terminal. All times are given in seconds. The fifth column shows the number of bits required to represent values of terminal nodes. It can be seen as an experimental justification of the necessity for the extension of the BDD package that is presented in this paper.

Table 1 Statistics of the computation run-time and the terminal node size of the autocorrelation computation through an SMTBDD with dynamically resizable terminal nodes

Benchmark	Inputs	Outputs	Cubes	Terminal size [bits]	Computation time [s]
b4	33	23	54	96	1.28
in3	35	29	75	96	1718.38
jbp	36	57	166	96	0.88
signet	39	8	124	96	--
apex2	39	3	1035	96	--
seq	41	35	336	96	--
apex1	45	45	206	96	--
ti	47	72	271	96	33.18
ibm	48	17	173	128	113.27
apex3	54	50	280	128	--
misg	56	23	75	128	0.28
e64	65	65	65	160	0.49
x7dn	66	15	622	160	16035.81
x2dn	82	56	112	192	0.59
soar	83	94	529	192	7.52
mish	94	43	91	192	0.65
apex5	117	88	1227	256	32.96
ex4p	128	28	620	288	360.26
o64	130	1	65	288	--

In the case of computing the autocorrelation coefficients for benchmarks with 30 or more variables using the Wiener-Khinchin theorem as specified in Eq. (7) through the Fast Walsh transform the computation failed, due to the memory limitations of 4GB for storing Walsh and the autocorrelation spectrum. Moreover, currently available BDD based techniques are limited to benchmarks of less than 32 variables. Therefore, experimental results are not compared with results using other approaches.

Table 2 gives the experimental results of space statistics. All results are given in number of nodes. Table entries with dashes indicate that the method failed to complete for that particular benchmark because of running out of memory.

Table 2 Space statistics of the autocorrelation computation through an SMTBDD with dynamically resizable terminal nodes

Benchmark	SMTBDD (f) size	SMTBDD (S_f) size	SMTBDD (S_{f2}) size	SMTBDD (B_f) size
	[Non-terminal nodes / Terminal nodes]			
b4	512 / 2	5923 / 277	3831 / 156	1842 / 158
in3	377 / 2	13874 / 538	9563 / 335	7496 / 1618
jbp	550 / 2	6813 / 260	4290 / 157	1501 / 211
signet	2956 / 2	--	--	--
apex2	7102 / 2	--	--	--
seq	142321 / 2	44743 / 3993	24093 / 2013	--
apex1	28414 / 2	77535 / 3191	55024 / 2193	--
ti	6187 / 2	16437 / 950	8782 / 493	49947 / 2091
ibm	835 / 2	40264 / 517	23680 / 311	5085 / 1078
apex3	--	--	--	--
misg	107 / 2	2994 / 120	1748 / 72	377 / 73
e64	1446 / 2	3039 / 99	1610 / 50	1686 / 39
x7dn	863 / 2	73602 / 1046	53813 / 761	32217 / 6280
x2dn	223 / 2	5541 / 116	3283 / 68	657 / 107
soar	995 / 2	35346 / 465	16116 / 254	2584 / 453
mish	131 / 2	5595 / 99	3832 / 64	142 / 65
apex5	2705 / 2	73230 / 238	29499 / 122	4645 / 158
ex4p	1301 / 2	133953 / 1095	56278 / 621	4621 / 1149
o64	--	--	--	--

7. Conclusions and Future Work

The complexity of methods for computing the autocorrelation is exponential in the number of variables of the function. The method presented in this paper is based on SMTBDD representations of the functions. Besides allowing the processing of multi-output Boolean functions of a large number of variables with a restricted memory, the SMTBDD offers a considerable flexibility in calculations of user defined subsets of particular autocorrelation coefficients. The computation is performed in the spectral domain by exploiting the Wiener-Khinchin theorem and the fast calculation algorithm through an SMTBDD. This computation, for large Boolean functions, requires calculations with large integers. For this reason, the usability of classical BDD packages is necessarily limited to relatively moderate size Boolean functions. With this motivation, we propose an extension of the classical BDD packages with dynamically resizable terminal nodes that allows us the computation of the autocorrelation for large Boolean functions. An experimental verification confirms that the proposed implementation allows us the computation of the autocorrelation of large Boolean functions. In a few cases the computation failed, due to the memory limitations caused by the size of the SMTBDD to represent either the function, its Walsh spectrum, or the autocorrelation spectrum.

Thus, the main concept presented in the paper is achieved. However, additional work towards further optimization of the implementation in terms of memory and time requirements is advisable. The implementation can be easily modified for the computation of the convolution, the correlation, and related mathematical operators. This concept can be successfully applied to the computation of other spectral transformations for large Boolean functions. The proposed BDD package can be used in other applications where dealing with functions having large integer values is required.

Acknowledgement: *The authors are very grateful to the reviewers for their constructive comments which significantly improved the contents and the presentation of the paper.*

REFERENCES

[1] M. G. Karpovsky, *Finite Orthogonal Series in the Design of Digital Devices*, New York: Wiley, 1976.

[2] M. G. Karpovsky, R. S. Stanković, and J. T. Astola, "Spectral Techniques for Design and Testing of Computer Hardware", In Proceedings of the 1st Int. Workshop on Spectral Techniques and Logical Design for Future Digital Systems, pp. 9-43, 2000.

[3] J. E. Rice, and J. C. Muzio, "On the Use of Autocorrelation Coefficients in the Identification of Three-level Decompositions", In Proceedings of the IEEE/ACM Int. Workshop on Logic Synthesis, pp. 187-191, 2003.

[4] O. Keren, I. Levin, and R. S. Stanković, "Linearization of Logical Functions Defined by a Set of Orthogonal Terms - Theoretical Aspects", *Automation and Remote Control*, vol. 72, no. 3, pp. 615-625, 2011.

[5] O. Keren, and I. Levin, "Linearization of Multi-Output Logic Functions by Ordering of the Autocorrelation Values", *Facta Universitatis Series: Electronics and Energetics*, vol. 20, no. 3, pp. 479-498, Dec. 2007.

[6] J. E. Rice, M. Serra, and J. C. Muzio, "The Use of Autocorrelation Coefficient for Variable Ordering for ROBDDs", In Proceedings of the Int. Workshop on Applications of Reed-Muller Expansion in Circuit Design, pp.185-196, 1999.

[7] M. G. Karpovsky, R. S. Stanković, and J. T. Astola, "Reduction of Sizes of Decision Diagrams by Autocorrelation Functions", *IEEE Trans. on Computers*, vol. 52, no. 5, pp. 592-606, 2003.

[8] O. Keren, "Reduction of Average Path Length in Binary Decision Diagrams by Spectral Methods", *IEEE Trans. on Computers,* vol. 57, no. 4, pp. 520-531, 2008.

[9] O. Keren, I. Levin, and R. S. Stanković, "Determining the Number of Paths in Decision Diagrams by Using Autocorrelation Coefficients", *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 30, no. 1, pp. 31-44, 2011.

[10] M. G. Karpovsky, and E. S. Moskalev, "Utilization of Autocorrelation Functions for Realization of Systems of Logical Functions", *Automation and Remote Control*, vol. 31, no. 2, pp. 243-250, 1970.

[11] R. Ebendt, G. Fey, and R. Drechsler, *Advanced BDD Optimization*, Netherlands: Springer, 2005.

[12] J. E. Rice, and J. C. Muzio, "Methods for Calculating Autocorrelation Coefficients", In Proceedings of the 4th Workshop on Boolean Problems, pp. 69-76, 2000.

[13] M. Radmanović, R. Stanković, and C. Moraga, "Analysis of Decision Diagram based Methods for the Calculation of the Dyadic Autocorrelation", *Int. Journal of Systemics, Cybernetics and Informatics*, pp.11-19, July 2007.

[14] R. S. Stanković, M. Bhattacharaya, and J. T. Astola, "Calculation of Dyadic Autocorrelation Through Decision Diagrams", In Proceedings of the European Conf. Circuit Theory and Design (ECCTD'01), pp. 28-31, 2001.

[15] G. Janssen, "A Consumer Report on BDD Packages", In Proceedings of the 16th Symposium on Integrated Circuits and Systems Design, pp. 217-223, 2003.

[16] E. M. Clarke, K. L. McMillan, X. Zhao, and M. Fujita, "Spectral Transforms for Extremely Large Boolean Functions", In Proceedings of the IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expression in Circuit Design, pp. 86-90, 1993.

[17] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient Implementation of a BDD Package", In Proceedings of the 27th Design Automation Conf., pp. 40-45, 1990.

[18] G. Janssen, "Design of a Pointerless BDD Package", In Proceedings of the 10th Int. Workshop on Logic and Synthesis, pp. 310-315, 2001.

[19] M. Thornton, and R. Drechsler, "Spectral Decision Diagrams Using Graph Transformations", In Proceedings of the Design, Automation and Test in Europe Conf. and Exhibition, pp. 713-717, 2001.

[20] F. Somenzi, "Efficient Manipulation of Decision Diagram", *Int. Journal on Software Tools for Technology Transfer*, vol. 3, no. 2, pp. 171-181, 2001.

[21] S. N. Yanushkevich, D. M. Miller, V. P. Shmerko, and R. S. Stanković, *Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook*, CRC Press, 2006.

[22] T. Sasao, and M. Fujita, *Representations of Discrete Functions*, Boston: Kluwer Academic Publishers, 1996.

[23] P. Dziurzanskii, V. P. Shmerko, and S. N. Yanushkevich, "Representation of Logical Circuits by Linear Decision Diagrams with Extension to Nanostructures", *Automation and Remote Control*, vol. 65, no. 6, pp. 920-937, 2004.

[24] D. Grobe and R. Drechsler, "BDD-based Verification of Scalable Designs", *Facta Universitatis Series: Electronics and Energetics*, vol. 20, no. 3, pp. 367-379, Dec. 2007.

[25] E. M. Clarke, M. Fujita, P. C. McGeer, K. McMillan, J. C. Yang, and X. Zhao, "Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure For Matrix Representation", In Proceedings of the Int. Workshop on Logic Synthesis, vol. 6a, pp. 1-15, 1993.

[26] R. S. Stanković, and B. Falkowski, "Spectral Transform Calculation through Decision Diagrams", *VLSI Design*, vol. C-14, no.1, pp. 5-12, 2002.

[27] G. D. Hachtel, and F. Somenzi, *Logic Synthesis and Verification Algorithms*, Norwell: Kluwer Academic Publishers, 1996.

[28] J. V. Sanghavi, R. K. Ranjan, R. K., Brayton, and A. Sangiovanni-Vincentelli, "High Performance BDD Package By Exploiting Memory Hierarchy", In Proceedings of the 33rd IEEE/ACM Design Automation Conference (DAC'96), pp. 635-640, 1996.

[29] H. Hasan Babu, and T. Sasao, "Shared Multi-Terminal Binary Decision Diagrams for Multiple-Output Functions", *IEICE Trans. on Fundamentals*, vol. E81-A, no. 12, pp. 2545-2553, 1998.

[30] F. Brglez, "The benchmark archives at CBL - ACM/SIGDA benchmarks", Nort Carolina State University, 2011, http://www.cbl.ncsu.edu/benchmarks.

[31] R. Rudell, *Espresso Misc. Reference Manual Pages*, Berkeley University of California, 1993.