Decision Diagram Method for Calculation of Pruned Walsh Transform

Dragan Jankovic, Radomir S. Stankovic, and Rolf Drechsler, *Member*, *IEEE*

Abstract—Discrete Walsh transform is an orthogonal transform often used in spectral methods for different applications in signal processing and logic design. FFT-like algorithms make it possible to efficiently calculate the discrete Walsh spectrum. However, for their exponential complexity, these algorithms are practically unsuitable for large functions. For this reason, a Binary Decision Diagram (BDD) based recursive method for Walsh spectrum calculation has been introduced in [4]. A disadvantage of this algorithm is that the resulting Multi-Terminal Binary Decision Diagram (MTBDD) representing the Walsh spectrum for f can be large for some functions. Another disadvantage turns out if particular Walsh coefficients are to be computed separately. The algorithm always calculates the entire spectrum and, therefore, it is rather inefficient for applications where a subset of Walsh spectral coefficients, i.e., the pruned Walsh spectrum, is required. In this paper, we propose another BDD-based method for Walsh spectrum calculation adapted for application where the pruned Walsh spectrum is needed. The method takes advantage of the property that, for most switching functions, the size of a BDD for f is usually quite a bit smaller than the size of the MTBDD for the Walsh spectrum. In our method, a MTBDD representing the Walsh spectrum is not constructed. Instead, two additional fields are assigned to each node in the BDD for the processed function f. These fields are used to store the results of intermediate calculations. Pairs of spectral coefficients are calculated and stored in the fields assigned to the root node. Therefore, the calculation complexity of the proposed algorithm is proportional to the size of the BDD for f whose spectrum is calculated. Experimental results demonstrate the efficiency of the approach.

Index Terms—Logic synthesis, spectral techniques, Walsh, pruned spectrum, BDD.

1 Introduction

Spectral methods have been widely used in many applications for a long time. For example, the use of spectral methods in circuit design dates back to the early 1960s (see, e.g., [2]). The Walsh and the Reed-Muller transform are the two discrete transforms most often used in logic design.

After the highest activity in the area in the 1970s (see, e.g., [17], [18], [20]), there is apparently a renewed interest in spectral methods for logic design, started in [36], which can be easily traced in several papers in journals and presented at conferences and some rather specialized meetings [12], [24].

The main limiting factor for application of spectral methods in processing of switching functions is their calculation complexity in spite of the existence of fast FFT-like algorithms [18], [20]. For example, the time and space complexities of FFT-like algorithms are $O(n2^n)$ and $O(2^n)$, respectively, for switching functions of n variables.

However, this problem is considerably overcome by the invention of a Binary Decision Diagram (BDD)-based method [3] for efficient calculation of Walsh transform [4]. The method was extended to calculation of various other spectral transforms (see, e.g., [4], [5], [11], [23], [27], [30]).

- *D. Jankovic and R.S. Stankovic are with the University of Nis, Faculty of Electronic Engineering, 18000 Nis, Yugoslavia.*
- *R. Drechsler is with Siemens AG, Corporate Technology, 81730 Munich, Germany. E-mail: rolf.drechsler@mchp.siemens.de.*

Manuscript received 28 Feb. 2000; revised 15 Sept. 2000; accepted 19 Oct. 2000.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 112636.

As is shown in [26], these methods perform FFT-like operations over DDs, instead of over vectors, from whence originates their advantages and possibility to process large functions. These methods exploit the recursive structure of the related transform matrices to generate a recursive calculation procedure. The related algorithms start from the DD representing a given function and, as a result, produce DD representing the corresponding spectrum. Two bottlenecks of these recursive algorithms, relevant for the considerations in this paper, can be mentioned:

1. For many functions, the produced DDs representing the spectrum are large.
2. It is impossible to calculate a particular spectral coefficient or a subset of coefficients separately without calculating the complete spectrum.

This can be considered an important disadvantage since there are many spectral methods in logic design for which only the values of a few selected coefficients are needed. Examples of such applications are spectral techniques for fault detection [7], [16], [19] and logic synthesis [6], [15], [21], [22], [24], [25], [31], [34], [35]. In particular, we want to note some recent techniques for nontautology checking of the equivalence of switching functions by spectral coefficients [33]. In several such applications, the pruned Walsh spectrum is required [10], [24].

In this paper, we are attempting to overcome the mentioned disadvantages of DD-based methods for the case of calculation of the Walsh spectrum of switching functions. Present DD methods for calculation of Walsh transform, as, for example, those in [6], [13], [21], are recursive in the sense that the main calculation procedure

calls itself recursively at each node to perform calculations level by level over the DD. Intermediate results of the calculations are stored as MTBDDs of subspectra for f. The output of such procedures is the MTBDD representing the Walsh spectrum S_f for f. We propose a new bottom-up nonrecursive method for calculation of Walsh coefficients of switching functions. This method takes advantage of the property that, for the great majority of switching functions, the BDD for the function is of much smaller size than the MTBDD for its spectrum since the Walsh spectrum of a switching function is an integer-valued function with a considerable number of different values. A greater number of terminal nodes increases the number of nonterminal nodes since, for each different value of terminal nodes, some branching of edges, thus, some nonterminal nodes, are required. For this reason, we do not generate an MTBDD for the Walsh spectrum. Instead, all the calculations are performed over the BDD for f. In each node, only one addition and subtraction is performed. The intermediate results are stored into pairs of fields assigned to each nonterminal node.

In [13], the calculation of a single Walsh coefficient, or a subset of Walsh coefficients, is based on a straightforward definition of spectral coefficients as an inner product of a given function f with the basic functions in terms of which a transform is defined. In this way, for a given f and the specified w, the required Walsh coefficient $S_f(w)$ is calculated as the inner product of the Walsh function of index w and f. The corresponding Walsh function is stored as a separate MTBDD which is then multiplied by the MTBDD for f by using the method given in [6] for multiplying a matrix or a vector with a given vector, both represented by DDs. In this way, the method exploits a property that calculation of a subset of k Walsh coefficients may be interpreted as a windowing operation over the Walsh matrix with a $(k \times 2^n)$ window of size k in the multiplication of the Walsh matrix and the truth-vector for f.

As noted above, in [13], the Walsh functions used in the calculation of the required subset of coefficients are stored by separate DDs that are necessarily simpler than MTBDDs for the complete Walsh matrix. That makes the algorithm applicable to large functions where the other methods can not be applied. If this method is used to extract all coefficients, it becomes identical to the method proposed in [6].

In this paper, since we are calculating the pairs of coefficients, we can say we are using a 2×2^n window. However, unlike [13], besides the basic definition of Walsh coefficients as inner products, we additionally exploited the periodicity in Walsh functions. Due to that, we split the window into a set of (2×2) windows and distributed them over the MTBDT for the Walsh transform by simultaneously matching the recurrence in both Walsh matrices due to their Kronecker product representable structure and, in a decision tree, due to the recursive application of the decomposition rules. It should be noted that both recursive structures originate in the decomposition of the domain group of order 2^n into the direct product of cyclic subgroups of order 2. Due to the restriction to (2×2) subwindows, the related calculations match the basic Walsh transform matrix $\mathbf{W}(1)$ and we just have to take into account the periodic change of the sign in the Kronecker product of $\mathbf{W}(1)$ by itself corresponding to -1 in $\mathbf{W}(1)$.

A transformation over a transform matrix, which is then multiplied by a vector, can be expressed through a corresponding transformation over the vector by keeping the transform matrix unchanged. Due to that, we transfer the (2×2) windowing operation over the MTBDD for the Walsh matrix into the corresponding operation over the MTBDD for a given f. In this way, we are able to calculate a pair of Walsh coefficients by assigning two fields to nodes in MTBDD for f.

In this respect, the method presented in this paper is an extension of the method in [27], where the windowing procedure is replaced by multiplication with a delta function $\delta(0, x)$, $x \in \{0, \ldots, 2^i - 1\}$, where i is the number of the level in the MTBDD.

Savings in the method proposed in this paper, compared to the method in [13], are that we do not need to store an MTBDD for a Walsh function and the MTBDD for f at the same time. Further, the calculations are simplified since they are reduced to the calculations over the corresponding pairs of function values at each node in the MTBDD for f. Experimental results clearly show the advantage of our method over the method given in [13].

We calculate selected Walsh coefficients by processing different nodes in the BDD for f. The pair of calculated coefficients is stored in two fields assigned to the root node. Complexity of the algorithm to calculate a pair of Walsh coefficients is proportional to the size of the BDD for f. Therefore, the algorithm is suitable for calculation of the pruned Walsh spectrum. It is possible to calculate the set of Walsh coefficients by our method for each switching function whose MTBDD is possible to build.

2 Basic Definitions

2.1 Walsh Transform

Denote by $C(C_2^n)$ the space of functions $f : C_2^n \to C$, where $C_2 = (\{0, 1+\}, \oplus)$, $\oplus$ denotes addition modulo 2, and C is the complex field. The set of discrete Walsh functions of order n is a complete orthogonal basis in $C(C_2^n)$.

Definition 1. *The Walsh functions of order n are defined by*

$$wal(k, j) = (-1)^{\sum_{i=1}^{n} k_i j_i},$$

where $k, j \in \{0, \ldots, 2^n - 1\}$ and

$$k = \sum_{l=0}^{n-1} k_l \cdot 2^l, \qquad j = \sum_{l=0}^{n-1} j_l \cdot 2^l.$$

In applications, the Walsh functions are used in different orderings [9], [18]. However, different orderings of Walsh functions produce the same set of Walsh coefficients. Most of the calculation procedures are using the so-called natural ordering since Walsh functions thus ordered express some symmetry properties which can be efficiently used to reduce the space and time requirements for the related calculations [2], [18], [31]. If the Walsh spectra in different orderings are

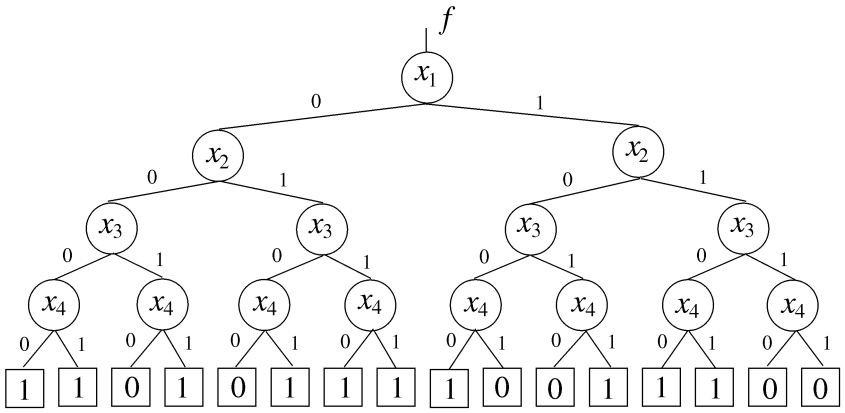

Fig. 1. $\mathrm{MTBDT}(f)$.

required, they are determined by reindexing the Walsh coefficients in the natural ordering. The natural ordering of Walsh functions is useful in matrix representations of Walsh transform and is compatible with the recursive structure of decision trees. Therefore, in this paper, we consider only the naturally ordered Walsh functions.

Definition 2. *The system of naturally ordered Walsh functions is conveniently represented by the Walsh transform matrix given by*

$$\mathbf{W}(n) = \bigotimes_{i=1}^{n} \mathbf{W}(1), \qquad W(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

where $\bigotimes$ denotes the Kronecker product.

Definition 3. *For an n variable function f, the Walsh transform and the inverse Walsh transform are defined by*

$$S_f(w) = 2^{-n} \sum_{x=0}^{2^n-1} f(x) wal(w,x),$$
$$f(x) = \sum_{w=0}^{2^n-1} S_f(w) wal(w,x).$$

In matrix notation, the Walsh transform pair is given as

$$\mathbf{S}_f = 2^{-n}\mathbf{W}(n)\mathbf{F},$$
$$\mathbf{F} = \mathbf{W}(n)\mathbf{S}_f,$$

where $\mathbf{S}_f = [S_f(0), \ldots, S_f(2^n-1)]^T$ is the vector of Walsh transform coefficients, and $\mathbf{F} = [f(0), \ldots, f(2^n-1)]^T$ is the vector representing values for f. When f is a switching function, then $\mathbf{F}$ is the truth-vector for f.

Example 1. For a four variable switching function $f(x_1, x_2, x_3, x_4)$ given by the truth-vector

$$\mathbf{F} = [1,1,0,1,0,1,1,1,1,1,0,0,1,1,1,0,0]^T,$$

the Walsh spectrum is given by the vector

$$S_f = [10,-2,2,2,0,0,0,4,2,-2,-2,-2,0,0,4,0]^T.$$

Definition 4. *The pruned Walsh spectrum is defined as a subset of 2^k, $k \leq n-1$, coefficients.*

2.2 Data Structure for the Algorithm

In the previous section, mathematical definitions are formulated in terms of transform matrices to represent operators and vectors to represent a given function and its spectrum. However, in practical implementations, both matrices and vectors are represented by decision diagrams which are used as the data structure to perform all the manipulations and calculations.

The Walsh spectrum is an integer-valued function and can be represented by a Multi-Terminal Binary Decision Diagram (MTBDD) [1], [6], derived by the reduction of the corresponding Multi-Terminal Binary Decision Tree (MTBDT). It should be noted that switching functions are a subset of integer valued functions if logic values 0 and 1 are interpreted as integers 0 and 1. With this interpretation of terminal nodes, BDDs become MTBDDs. Since, in calculations of the Walsh spectrum, we perform integer operations over DDs, in this paper, we assume that both switching functions and their integer-valued Walsh spectra are represented by MTBDDs, denoted by $\mathrm{MTBDD}(f)$ and $\mathrm{MTBDD}(S_f)$.

Unlike existing DD-based methods, the algorithm for implementation of the proposed method does not produce a $\mathrm{MTBDD}(S_f)$. Instead, we assign to each node in the $\mathrm{MTBDD}(f)$ two fields denoted as $plus$ and $minus$ fields. These fields are used to store the results of intermediate calculations. In this way, the algorithm efficiently exploits the property that, for a great majority of switching functions, $size(MTBDD(f)) << size(MTBDD(S_f))$, where the size of a MTBDD is defined as the number of nodes in the MTBDD.

In an $\mathrm{MTBDT}(f)$, a level consists of nodes to which the same variable in $f(x_1, \ldots, x_n)$ is assigned. We assume that levels in an MTBDT are denoted by indices of the variables assigned. Thus, levels are denoted by 1 to n, where n is the number of variables, with the root node at the level denoted by 1. The same convention applies to MTBDDs.

Example 2. Fig. 1 shows $\mathrm{MTBDT}(f)$ for f in Example 1 and Fig. 2 shows $\mathrm{MTBDT}(S_f)$.

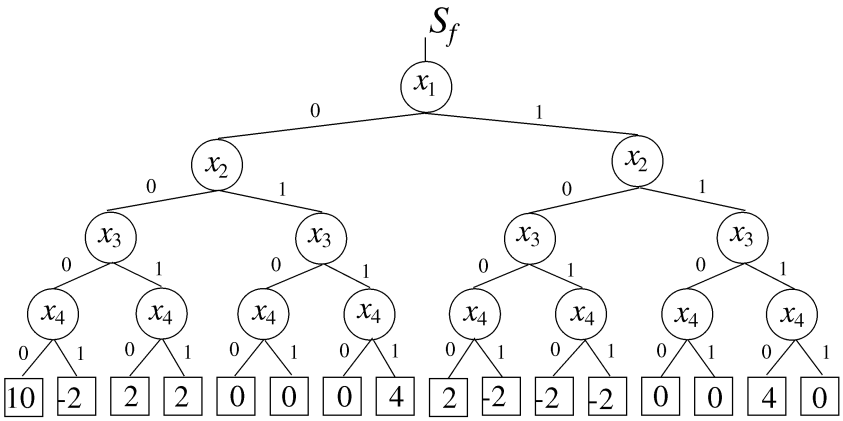

Fig. 2. $\mathrm{MTBDT}(S_f)$.

3 Recursive BDD-Based Method for Walsh Spectrum

A BDD-based method for Walsh spectrum calculation has been proposed in [4]. This method is a recursive performing of operations defined by $\mathbf{W}(1)$ over subgraphs in the BDD for f and has a structure similar to the *Apply* procedure given in [3]. The result of this procedure is an MTBDD whose terminals are the Walsh spectral coefficients. Besides the above-mentioned disadvantages of methods for calculation of spectral transforms through DDs, we want to point out that complexity of calculations for the method in [4] depends on the size of the BDD as well as on the distribution of nodes over the levels in the BDD. For two functions whose BDDs have equal size, the run times for calculation of the Walsh spectrum can be quite different. In attempting to overcome or reduce some of these disadvantages, we reorganized the calculation of the Walsh coefficients by splitting the spectrum into pairs of coefficients.

4 Calculation of Pairs of Walsh Coefficients

In this section, we present a method for calculation of pairs of Walsh coefficients. For simplicity of explanation, we first present the method for calculation through MTBDTs. However, in practice, calculations are performed through MTBDDs, which ensures efficiency. Then, we show modifications required in calculations through MTBDDs.

4.1 Calculation through MTBDTs

In the MTBDT for a given f, we assign to each node v two fields denoted by *plusf(v)* and *minusf(v)*. The values of these fields are calculated as follows:

Assume that we want to calculate a pair of Walsh coefficients W_d and $W_{d+2^{n-1}}$, $d \in \{0, \ldots, 2^{n-1}-1\}$. We determine the binary equivalent $d = (d_1, \ldots, d_n)$ through the relation $d = \sum_{i=0}^{n-2} 2^i x^i$, $d_i \in \{0, 1\}$. Then, we introduce a parameter vector P by deleting the first bit in the binary representation for d. Thus, $P = (d_2, \ldots, d_n)$. The jth element of P determines the way of calculation of fields $plusf(v)$ and $minusf(v)$ assigned to the nodes at the level j in the MTBDD(f) as follows:

1. If v is a terminal node showing the constant c, then
$$plusf(v) = minusf(v) = c. \tag{1}$$
2. If v is a nonterminal node with the successors $v.low$ and $v.high$, then
$$\begin{aligned} plusf(v) &= plusf(v.low) + plusf(v.high), \\ minusf(v) &= plusf(v.low) - plusf(v.high), \end{aligned} \tag{2}$$
for $p_j = 0$, and
$$\begin{aligned} plusf(v) &= minusf(v.low) + minusf(v.high), \\ minusf(v) &= minusf(v.low) - minusf(v.high), \end{aligned} \tag{3}$$
for $p_j = 1$.

Clearly, if $j = n$ and the outgoing edges of a node v point to the constant nodes showing the values c_q and c_{q+1}, $q \in \{0, \ldots, 2^n - 1\}$, then $plusf(v) = c_q + c_{q+1}$ and $minusf(v) = c_q - c_{q+1}$.

Fig. 3 illustrates the application of (1), (2), and (3).

A procedure $CalcWalsh$ for the calculation of the pair of Walsh coefficients $W_d, W_{d+2^{(n-1)}}$ can be formulated as follows:

Procedure(*CalcWalsh*)

1. Express the decimal index d of the required Walsh coefficient W_d by a binary sequence $d = (d_1, d_2, \ldots, d_n)$, where n is the number of variables.
2. Generate a parameter vector P as
$$P = (d_2, d_3, \ldots, d_n).$$
3. Traverse $\mathrm{MTBDT}(f)$ level by level starting from level n. Calculate the values of fields $plusf$ and $minusf$ for all nonterminal nodes as specified by (1), (2), and (3).
4. The pair of Walsh coefficients $(W_d, W_{d+2^{(n-1)}})$ in the natural ordering is determined as
$$\begin{aligned} W_d &= plusf(root), \\ W_{d+2^{(n-1)}} &= minusf(root), \end{aligned}$$
where $root$ is the root node in the $\mathrm{MTBDT}(f)$.

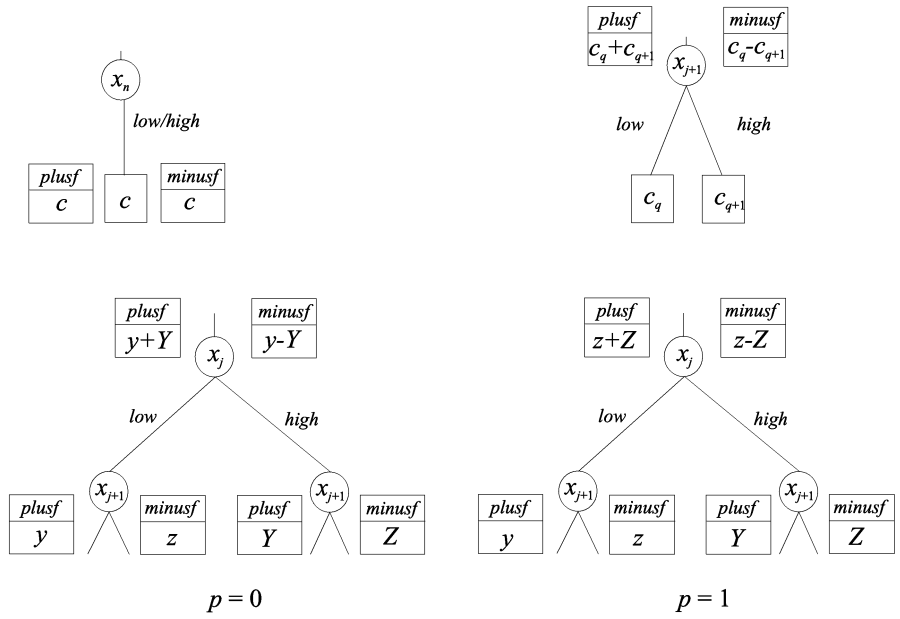

Fig. 3. $\mathrm{MTBDT}(f)$.

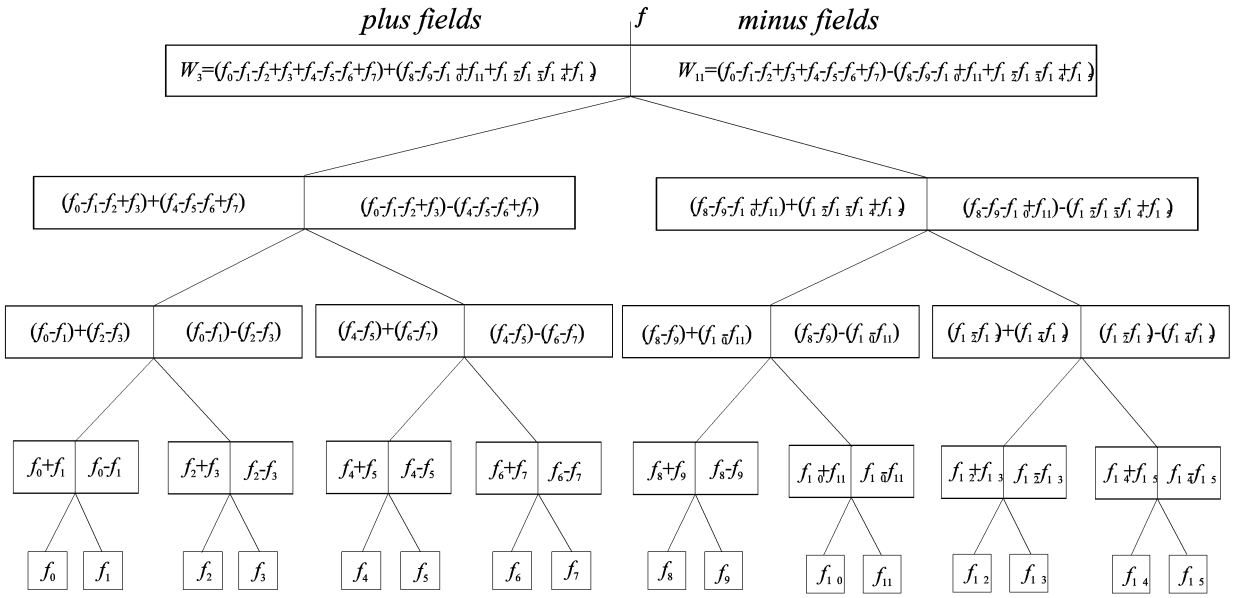

Fig. 4. Calculation of coefficients W_3 and W_{11}.

Example 3. Fig. 4 shows the calculation of the pair of Walsh coefficients W_3 and W_{11} for an arbitrary four variable function f. Since $d = (0011)$ for W_3 and $d = (1011)$ for W_{11}, for both coefficients $P = (011)$, which shows that they can be calculated simultaneously. In Fig. 4, for each node v, the values of fields $plusf(v)$ and $minusf(v)$ are shown on the left and the right side of the node, respectively. If the procedure explained in Fig. 4 is applied to f in Example 1, then we get $W_3 = 2$ and $W_{11} = -2$.

4.2 Calculation through MTBDDs

In practice, the calculations are always performed over MTBDDs and the advantages are taken from the compactness of the MTBDDs compared to MTBDTs.

In MTBDDs, there are edges connecting nodes at nonadjacent levels. A crossing of an edge to a level is denoted as the cross point [29]. The cross points should be considered in each calculation procedure through MTBDDs to take into account, in a proper manner, the impact of the deleted nodes [26], [29]. For this reason, calculation rules (1) and (2) should be changed. The required modifications can be determined from the following considerations:

Fig. 5a shows a node v at the ith level, with the low successor node u at the level $i + k$. Assume that the values

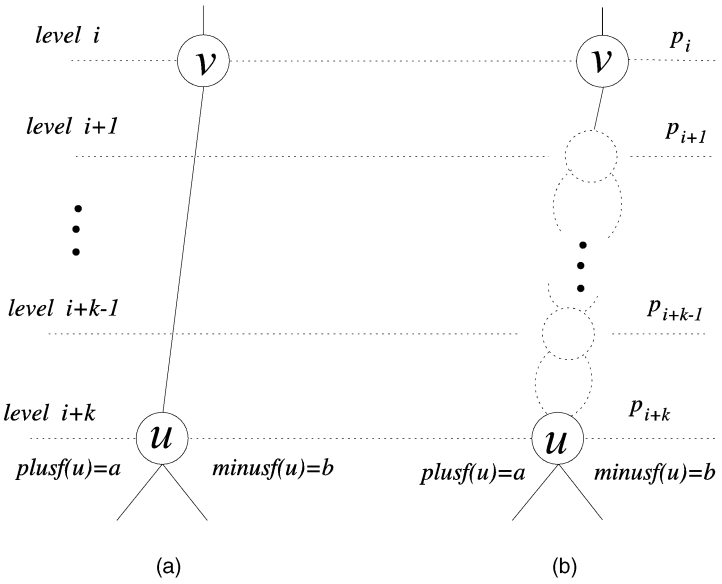

Fig. 5. Impact cross points.

of fields assigned to the node u are $plusf(u) = a$ and $minusf(u) = b$.

We assign a cross node to each point where a level crosses the edge between v and u. In Fig. 5b, cross points are shown by dotted circles. Each cross point corresponds to a deleted node whose both edges point to the same node.

From (1) and (2), for a given value of k and a given parameter vector P, we first calculate $plusf'(v.low)$ and $minusf'(v.low)$ in the following way:

If $k = 2$,

$$\begin{array}{ll} plusf'(v.low) = 2a, & \\ minusf'(v.low) = 0, & k = 2, p_{i+1} = 0, \\ plusf'(v.low) = 2b, & \\ minusf'(v.low) = 0, & k = 2, p_{i+1} = 1, \end{array}$$

Similarly, if $k = 3$,

$$\begin{array}{ll} plusf'(v.low) = 4a, & \\ minusf'(v.low) = 0, & k = 3, p_{i+1} = 0, p_{i+2} = 0, \\ plusf'(v.low) = 4b, & \\ minusf'(v.low) = 0, & k = 3, p_{i+1} = 0, p_{i+2} = 1, \\ plusf'(v.low) = 0, & \\ minusf'(v.low) = 0, & k = 3, p_{i+1} = 1. \end{array}$$

In the general case, for a node v at the level $level(v)$ in MTBDD(f) with $level(root) = 1$, the values for the $plusf$ and the $minusf$ fields assigned to v are calculated as

$$\begin{array}{lll} plusf(v) & = plusf'(v.low) & \\ & +plusf'(v.high), & p_{level(v)} = 0, \\ minusf(v) & = plusf'(v.low) & \\ & -plusf'(v.high), & p_{level(v)} = 0, \\ plusf(v) & = minusf'(v.low) & \\ & +minusf'(v.high), & p_{level(v)} = 1, \\ minusf(v) & = minusf'(v.low) & \\ & -minusf'(v.high), & p_{level(v)} = 1, \end{array} \tag{4}$$

where

$$plusf'(v.low) =$$
$$\begin{cases} 2^{level(v.low)-level(v)-1} plusf(v.low), & p_l{=}0, level(v){<}l{\le}level(v.low); \\ 2^{level(v.low)-level(v)-1} minusf(v.low), & p_{level(v.low)}{=}1, \\ & p_l{=}0, level(v){<}l{<}level(v.low); \\ 0, & otherwise; \end{cases}$$
$$minusf'(v.low) =$$
$$\begin{cases} minusf(v.low), & level(v.low) - level(v) = 1; \\ 0, & otherwise; \end{cases}$$
$$plusf'(v.high) =$$
$$\begin{cases} 2^{level(v.high)-level(v)-1} plusf(v.high), & p_l{=}0, level(v){<}l{\le}level(v.high); \\ 2^{level(v.high)-level(v)-1} minusf(v.high), & p_{level(v.high)}{=}1, \\ & p_l{=}0, level(v){<}l{<}level(v.high); \\ 0, & otherwise; \end{cases}$$
$$minusf'(v.high) =$$
$$\begin{cases} minusf(v.high), & level(v.high) - level(v) = 1; \\ 0, & otherwise. \end{cases}$$

Fig. 6 and Fig. 7 illustrate determination of values for fields $plusf'$ and $minusf'$ for $k = 2$ and $k = 3$.

Example 4. Consider an integer function $f(x_1, x_2, x_3, x_4)$ given by the MTBDD shown in Fig. 8a. For the Walsh coefficients W_6 and W_{14}, the binary representations of decimal indices are $d = (0110)$ and $d = (1110)$,

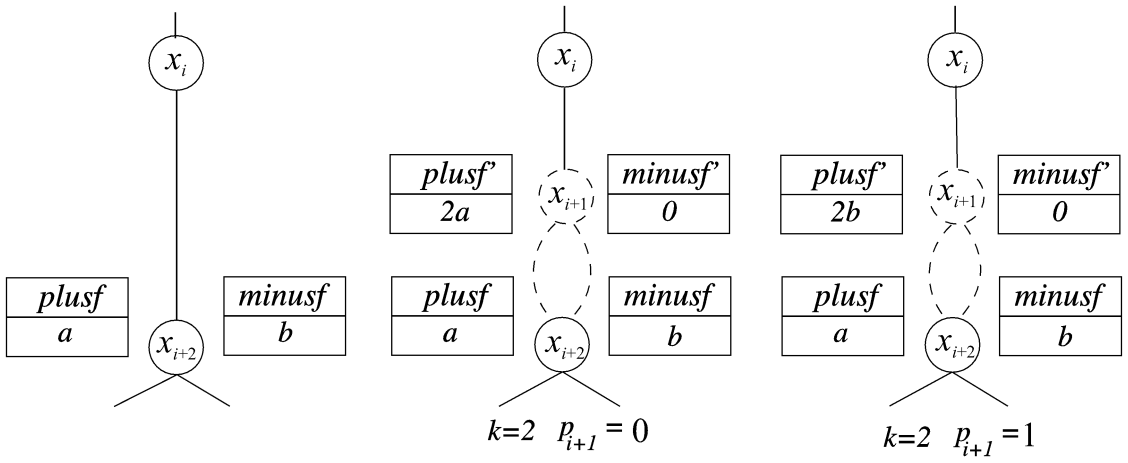

Fig. 6. Calculations in cross points for $k = 2$.

respectively. Thus, the parameter vector P is $P = (110)$. Fig. 8b shows the values of fields $plusf$ and $minusf$ calculated by using the procedure $CalcWalsh$. The recursive method [4] applied to the function f as the result gives an $\mathrm{MTBDD}(S_f)$ representing the complete Walsh spectrum (Fig. 8c). It is obvious from Fig. 8 that $\mathrm{MTBDD}(S_f)$ is much larger than $\mathrm{MTBDD}(f)$.

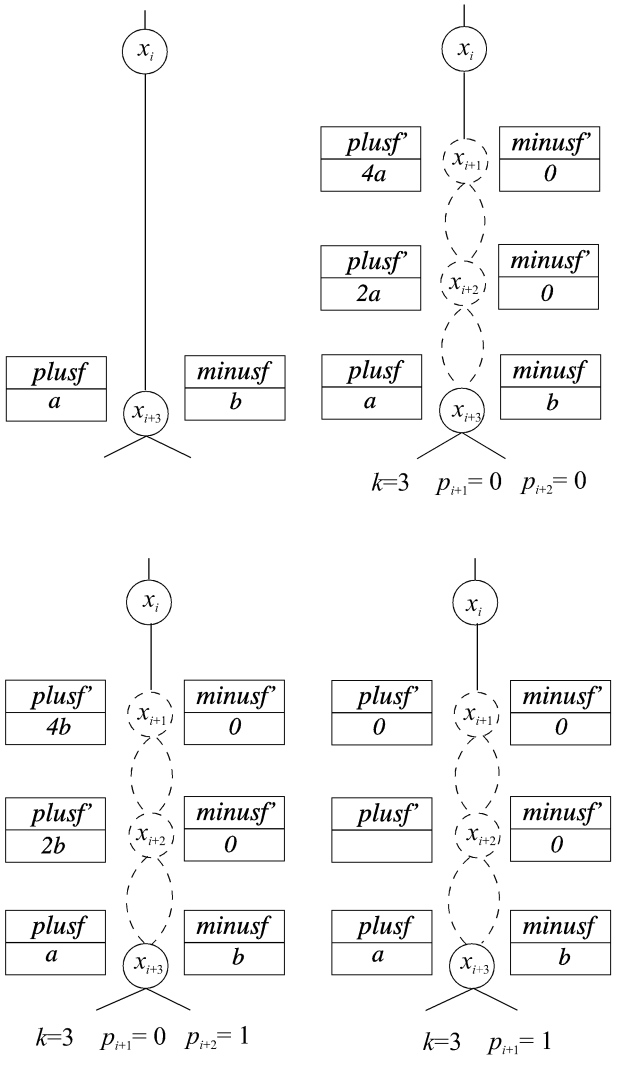

Fig. 7. Calculations in cross points for $k = 3$.

5 Calculation of the Pruned Walsh Spectrum

We calculate the pruned Walsh spectrum with r coefficients by running the *CalcWalsh* procedure a few times for different values of elements of the parameter vector P. The choice between the fields $plusf$ or $minusf$ of the successors which will be used in calculation of a Walsh coefficient depends on the index of the spectral coefficients which is going to be calculated. Thus, in some situations, for calculation of new coefficients, processing of all the nodes in the MTBDD(f) is not needed. Instead, only nodes on some levels are processed. That choice also depends on the indices of the calculated coefficients. The following example explains this method.

Example 5. Let f be an n variable integer function represented by an MTBDD. We want to calculate the following Walsh coefficients $W_{(0,0,i_3,\ldots,i_n)}$, $W_{(0,1,i_3,\ldots,i_n)}$, $W_{(1,0,i_3,\ldots,i_n)}$, $W_{(1,1,i_3,\ldots,i_n)}$. In the first step, all the nodes in the MTBDD are processed, resulting in coefficients $W_{(0,0,i_3,\ldots,i_n)}$ and $W_{(1,0,i_3,\ldots,i_n)}$. In the second step, only the root node is processed ($minus$ fields of successors are used), resulting in two other required coefficients.

5.1 Complexity of Calculation

The previous example shows that the order of calculation is important in the cases when a set of spectral coefficients is calculated. The best order of calculation is the lexicographic bit-reverse order. The calculation starts from the coefficient with the lowest index and the coefficients are calculated in the increasing order of indices.

The number of operations used in the proposed algorithm is obviously proportional to the size of the $\mathrm{MTBDD}(f)$. If only one or two coefficients are calculated, in each node only one addition and one subtraction is performed.

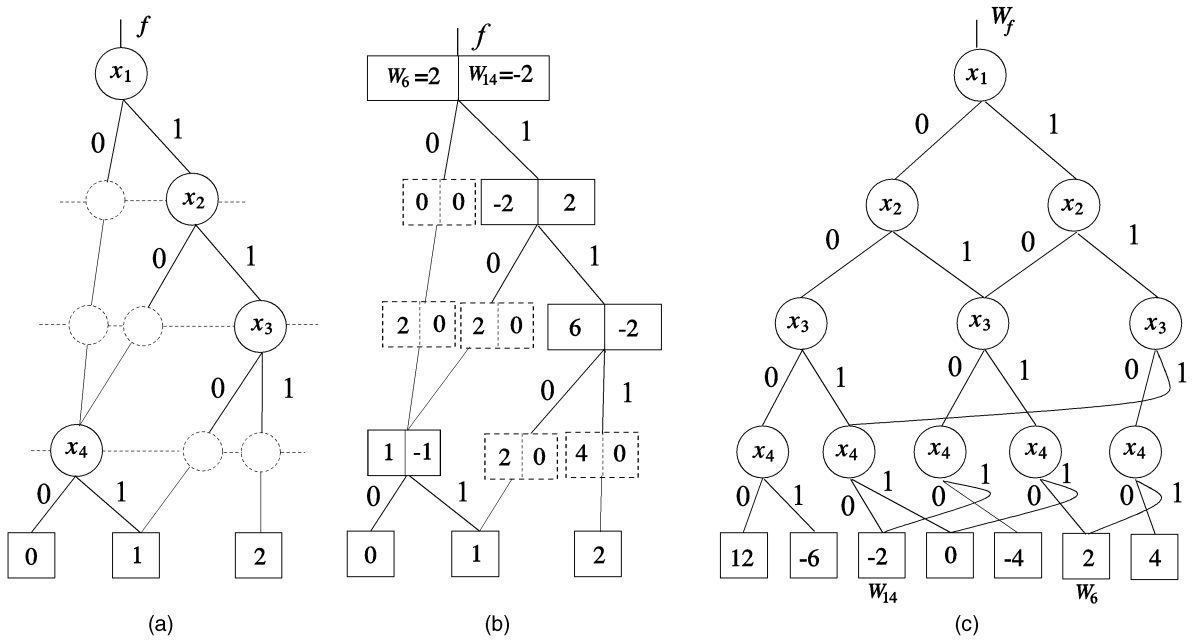

Fig. 8. (a) MTBDD(f), (b) Calculation of W_6 and W_{14}, (c) $\mathrm{MTBDD}(S_f)$.

The number of operations for calculation of the complete spectrum can be determined as follows:

Let f be represented by an MTBDT and the coefficients are calculated in the lexicographic bit-reverse order as explained above. The nodes at the last nonterminal level (level n for an n variable function) are processed once, but the nodes at the level $n-1$ are processed twice. First, the calculations are performed over the values $plusf$ of successors and then the $minusf$ values. Similarly, the nodes at the level $n-2$ are processed four times, etc. Thus, the total number of additions and subtractions performed, O_{MTBDT}, to calculate the Walsh spectrum of an n variable function represented by the MTBDT, is given by

$$O_{MTBDT} = 2 \cdot (1 \cdot 2^{n-1} + 2 \cdot 2^{n-2} + \ldots + 2^{n-1} \cdot 1) = n \cdot 2^n.$$

Obviously, in calculations through MTBDTs, the number of performed operations is equal to the number of operations in a FFT-like algorithm for the Walsh transform.

If f is given by a MTBDD, the number of performed operations O_{MTBDD} is given by

$$O_{MTBDD} = \sum_{i=1}^{n} 2^{n-i} \cdot NoNodesAtLevel(i), \qquad (5)$$

where $NoNodesAtLevel(i)$ is the number of nodes at the level i. Since the size of a MTBDD is always smaller than the size of the corresponding MTBDT, the efficiency of our method is obvious. The number of operations to calculate a set of Walsh coefficients depends on the function (i.e., on the size of its MTBDD) and on the indices of the required coefficients. For a given $\mathrm{MTBDD}(f)$, that number for each pair of coefficients can be calculated by using (5).

6 Experimental Results

The proposed method is suitable for the calculation of a subset of Walsh coefficients. The following experimental results show the advantage of our method over the recursive method if only a subset of the Walsh coefficients is calculated. We have developed a programming package in C++ which performs both the recursive algorithm in [4] and the algorithm proposed in this paper. This integration of both algorithms in the same environment allows direct comparisons of these algorithms in calculation of a set of Walsh coefficients and the entire spectrum. The experiments were carried out on a *SUN Sparc 4* with 128 MByte of main memory and all calculation times are given in CPU seconds.

For all experiments, we assume that the BDD for the benchmark function is already constructed. For the variable ordering of the BDDs, we used the initial variable ordering, i.e., as it occurs in the benchmark description. Using more advanced BDD minimization techniques, like dynamic variable ordering, will allow us to apply the method to larger functions. But, since this is not the objective of this paper, only benchmarks are considered for which the BDD construction directly worked.

6.1 Space Complexity

The output of the recursive algorithm in [4] is the $\mathrm{MTBDD}(S_f)$. The algorithm proposed in this paper stores the results of the calculation at fields assigned to the nodes in the $\mathrm{MTBDD}(f)$. Therefore, an overall estimate of the memory needed by both approaches can be obtained by comparing the sizes of $\mathrm{MTBDD}(S_f)$ and $\mathrm{MTBDD}(f)$.

Table 1 shows the sizes of $\mathrm{MTBDD}(f)$ and $\mathrm{MTBDD}(S_f)$ as well as their ratios for some benchmark functions. As can be seen in the last column, for all benchmarks considered, the $\mathrm{MTBDD}(f)$ is much smaller than the $\mathrm{MTBDD}(S_f)$. The ratio between the sizes illustrates a saving of memory achieved by the proposed algorithm. In some cases, the sizes differ by a factor of 30 (see, e.g., *in5*). For larger functions, $\mathrm{MTBDD}(S_f)$ often cannot be constructed at all, while the MTBDD for f can be easily built. This clearly

TABLE 1
Experimental Results: Memory

name	*in*	*out*	MTBDD(f)	MTBDD(S_f)	*ratio*
apex4	9	19	927	4917	18.85
ex1010	10	10	1106	6386	17.37
alu4	14	8	1196	6340	18.86
misex3	14	14	1300	18738	6.94
add6	12	7	183	591	30.96
duke2	22	29	972	6854	14.18
vg2	25	8	1043	3906	26.70
misex2	25	18	135	1029	13.12
e64	65	65	128	949	13.49
in5	24	14	491	14830	3.31
planet	14	25	474	8816	5.38

demonstrates the efficiency of our approach with respect to the memory consumption.

6.2 Time Complexity

In Table 2, we give the execution times to calculate the Walsh spectrum by the recursive method (column *recur.*) and to calculate the first 64, 256, and 1,024 coefficients and the entire spectrum by our method (columns denoted by 64, 256, 1,024, and *entire*, respectively). As can be seen easily, our method is more efficient if only a part of the Walsh spectrum is calculated. If the entire spectrum is calculated, the recursive method is faster. For large functions, like *e64*, the calculation of the entire spectrum is impossible within a reasonable time by using our method. It is interesting that, for functions with a smaller number of variables (less than 20) as well as smaller number of outputs (less than 25), our method is faster than the recursive method even for the calculation of the entire spectrum (apex4, ex1010, alu4, misex3, add6, planet). But, our technique also allows us to compute (at least) some coefficients in the cases where the recursive approach fails.

Table 3 compares the time requirements for the method in [13] (column *ISCAS'94-method*) and the method we are proposing (column *Pruned-Walsh-method*) for computing the first-order coefficients for the set of benchmark functions used in [13]. The machine used in [13] is a SUN Sparc 2, while we performed the experiments on a PC Pentium/133. CPU times are given in seconds. It may be seen that our method is much faster than the method given in [13]. That is not surprising since the method given in [13] is based on the matrix by vector multiplication method proposed in [6]. The difference from [6] is that no whole Walsh matrix is used. The authors of [13] exploit the property that each Walsh coefficient for a given f is defined as the inner product of the corresponding Walsh function and f. Thus, instead of calculating with the complete Walsh matrix as in [6], they organized the calculation of a subset of Walsh coefficients by using the corresponding subset of Walsh functions in terms of which the required coefficients are defined.

In [13], for each Walsh function, the corresponding BDD is generated and multiplied by the BDD for f. Obviously, that is a quite a bit more time consuming procedure than the method proposed in this paper.

TABLE 2
Expermental Results: Runtime

function			recur.	pruned			
name	*in*	*out*	*entire*	64	256	1024	*entire*
apex4	9	19	0.76	0.01	0.02	-	0.04
ex1010	10	10	1.11	0.01	0.02	0.06	0.06
alu4	14	8	3.51	0.01	0.01	0.04	0.49
misex3	14	14	9.35	0.01	0.02	0.07	0.97
add6	12	7	0.38	0.01	0.01	0.02	0.06
duke2	22	29	6.30	< 0.01	0.01	0.02	72.52
vg2	25	8	65.71	0.01	0.01	0.02	429.06
misex2	25	18	0.39	< 0.01	< 0.01	0.02	630.95
e64	65	65	10.62	0.01	0.02	0.06	no fin.
in5	24	14	32.38	< 0.01	< 0.01	0.02	411.07
planet	14	25	1.62	< 0.01	0.01	0.07	1.11

TABLE 3
Experimental Results for Large Functions: Runtime

circuit name	*output*	*No. inputs*	*ISCAS'94-method*	*Pruned-Walsh-method*
alu4	r	14	0.5	0.01
C1908	last output	33	5.5	0.05
C1355	1326gat	41	15.0	0.11
C432	195	36	1.7	0.02
C3540	all of 22 outputs	50	not available	0.55

7 Closing Remarks

In many applications, the complete spectrum of a discrete transform is not required. Instead, only a small set of spectral coefficients is sufficient. This paper proposes a method for efficient calculation of pairs of Walsh spectral coefficients through MTBDDs. To calculate 2^k Walsh coefficients for a given f, we process the $\mathrm{MTBDD}(f)$ 2^{k-1} times. However, we do not always process all the nonterminal nodes. Depending on the index of the Walsh coefficients calculated, the processing of some nonterminal nodes is avoided in some steps of the algorithm.

The method does not generate the MTBDD for the Walsh spectrum. Instead, the results of intermediate calculations, as well as the pair of calculated coefficients, are stored in two fields assigned to each nonterminal node. Therefore, the complexity of the related algorithm is proportional to the size of the MTBDD for the processed function f.

7.1 Extensions

7.1.1 Memory Reduction

If we want to further reduce the storage requirements, instead of two additional fields $plusf$ and $minusf$, one additional file can be used. In this case, a minor modification of the procedure *CalcWalsh* leads to a procedure for calculation of a single Walsh coefficient. However, in this case, the time complexity of the algorithm increases.

7.1.2 Calculation of Other Transforms

By simple modifications, our method can be used to calculate spectral coefficients for other discrete transforms whose transform matrices have the Kronecker product structure [5], [8]. Modifications are made depending on the basic transform matrix and the used operations. The following example shows the modifications of the proposed method for calculation of the pruned Reed-Muller spectrum.

Example 6. The Reed-Muller transform is defined by the Reed-Muller transform matrix

$$\mathbf{R}(n) = \bigotimes_{i=1}^{n} \mathbf{R}(1),$$

where the basic Reed-Muller transform matrix is given as

$$\mathbf{R}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

the entries of which are the logic values 0 and 1.

In calculation of the Walsh transform, (4) describes performing operations determined by $\mathbf{W}(1)$ at each node and the cross point of $\mathrm{MTBDD}(f)$. In calculation of the Reed-Muller spectral coefficients, we should perform operations determined by $\mathbf{R}(1)$. In this case, all the calculations are performed modulo 2. Therefore, in this case, relations corresponding to (4) have a form

$$\begin{array}{ll} plusf(v) = & \\ plusf'(v.low), & p_{level(v)} = 0, \\ minusf(v) = & \\ plusf'(v.low) \oplus plusf'(v.high), & p_{level(v)} = 0, \\ & \\ plusf(v) = & \\ minusf'(v.low), & p_{level(v)} = 1, \\ minusf(v) = & \\ minusf'(v.low) \oplus minusf'(v.high), & p_{level(v)} = 1, \end{array} \tag{6}$$

where

$$plusf'(v.low) = \begin{cases} plusf(v.low), & p_l = 0, level(v) < l \le level(v.low); \\ minusf(v.low), & p_{level(v.low)} = 1, \\ & p_l = 0, level(v) < l < level(v.low); \\ 0, & otherwise; \end{cases}$$

$$minusf'(v.low) = \begin{cases} minusf(v.low), & level(v.low) - level(v) = 1; \\ 0, & otherwise; \end{cases}$$

$$plusf'(v.high), = \begin{cases} plusf(v.high), & p_l = 0, level(v) < l \le level(v.high); \\ minusf(v.high), & p_{level(v.high)} = 1, \\ & p_l = 0, level(v) < l < level(v.high); \\ 0, & otherwise; \end{cases}$$

$$minusf'(v.high) = \begin{cases} minusf(v.high), & level(v.high) - level(v) = 1; \\ 0, & otherwise; \end{cases}$$

and $\oplus$ denotes $sum\ mod\ 2$ operation.

A further extension of the proposed method can be made to calculate spectral transforms for multiple-valued functions, as, for example, Galois field transform [14] or the Reed-Muller-Fourier transform [28]. The method can be also used to calculate the discrete transform spectra optimized by choosing different polarities for variables. In this case, the Multi-valued DDs (MDDs) [32] are assumed as the underlying data structure to represent the processed functions and their spectra.

The approach proposed in this paper for the pruned Walsh spectrum can be further extended to the efficient calculation of pruned spectra of discrete transforms on different finite groups which can be represented as a direct product of some subgroups of smaller orders and whose transform matrices have the Kronecker product structure.

Acknowledgments

This work was carried out while Dragan Jankovic was at the Albert-Ludwigs-University in Freiburg, Germany, based on grant No. A/99/14064-324 from *Deutscher Akademischer Austauschdienst* (DAAD).

References

[1] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.P. Hatchel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic Decision Diagrams and Their Applications," *Proc. Int'l Conf. Computer-Aided Design*, pp. 180-191, Nov. 1993.

[2] K.G. Beauchamp, *Applications of Walsh and Related Functions.* New York: Academic Press, 1984.
[3] R.E. Bryant, "Graph-Based Algorithms for Boolean Functions Manipulation," *IEEE Trans. Computers,* vol. 35, no. 8, pp. 667-691, Aug. 1986.
[4] E.M. Clarke, X. Zhao, M. Fujita, Y. Matsunaga, and P. McGeer, "Fast Walsh Transform Computation with Binary Decision Diagram," *Proc. IFIP WG 10.5 Workshop Applications of the Reed-Muller Expansion in Circuit Design,* pp. 82-85, 1993.
[5] E.M. Clarke, M. Fujita, and X. Zhao, "Multi-Terminal Decision Diagrams and Hybrid Decision Diagrams," *Representation of Discrete Functions,* T. Sasao and M. Fujita, eds., pp. 93-108, Kluwer Academic, 1996.
[6] E.M. Clarke, K.L. McMillan, X. Zhao, M. Fujita, and J. Yang, "Spectral Transforms for Large Boolean Functions with Application to Technology Mapping," *Proc. Design Automation Conf.,* pp. 54-60, 1993.
[7] T. Damarla and M.G. Karpovsky, "Reed-Muller Spectral Techniques for Fault Detection," *IEEE Trans. Computers,* vol. 38, pp. 788-797, 1989.
[8] D.F. Elliot and K.R. Rao, *Fast Transforms: Algorithms, Analysis, Applications.* London: Academic Press, 1982.
[9] B.J. Falkowski, "Recursive Relationships, Fast Transforms, Generalizations and VLSI Iterative Architecture for Gray Code Ordered Walsh Functions," *IEE Proc. Computerized Digital Technology,* vol. 142, no. 5, pp. 325-331, 1995.
[10] B.J. Falkowski, I. Schäfer, and M.A. Perkowski, "Effective Computer Methods for the Calculation of Rademacher-Walsh Spectrum for Completely and Incompletely Specified Boolean Functions," *IEEE Trans. Computer-Aided Design,* vol. 11, no. 10, pp. 1207-1226, 1992.
[11] B.J. Falkowski and C.H. Chang, "Efficient Algorithm for the Calculation of Arithmetic Spectrum from OBDD and Synthesis of OBDD Form Arithmetic Spectrum," *Proc. 27th IEEE Int'l Symp. Circuit and Systems,* pp. 197-200, May 1994.
[12] B.J. Falkowski and C.H. Chang, "Hadamard-Walsh Spectral Characterization of Reed-Muller Expressions," *Computers and Electrical Eng.,* vol. 25, no. 2, pp. 111-134, 1999.
[13] M. Fujita, J.C.-Y. Yang, M. Clarke, X. Zhao, and P. McGeer, "Fast Spectrum Computation for Logic Functions Using Binary Decision Diagrams," *Proc. Int'l Symp. Computer-Aided Surgery (ISCAS '94),* pp. 275-278, 1994.
[14] D.H. Green and I.S. Taylor, "Modular Representation of Multiple-Valued-Logic Systems," *Proc. IEE,* vol. 121, pp. 409-418, 1974.
[15] J.P. Hansen and M. Sekine, "Synthesis by Spectral Translation Using Boolean Decision Diagrams," *Proc. 33rd Design Automation Conf.,* pp. 248-253, 1996.
[16] T. Hsiao and S.C. Seth, "An Analysis of the Use of Rademacher-Walsh Spectrum in Compact Testing," *IEEE Trans. Computers,* vol. 33, pp. 934-938, 1984.
[17] S.L. Hurst, *The Logical Processing of Digital Signals.* Crane Russak, and Edward Arnold, 1978.
[18] S.L Hurst, D.M. Miller, and J.C. Muzio, *Spectral Techniques in Digital Logic.* Academic Press, 1985.
[19] S.L. Hurst, "Use of Linearization and Spectral Techniques in Input and Output Compaction Testing of Digital Networks," *IEE Proc. Computerized Digital Technology,* vol. 136, pp. 48-56, 1989.
[20] M.G. Karpovsky, *Finite Orthogonal Series in the Design of Digital Devices.* New York and Jerusalem: Wiley and JUP, 1976.
[21] Y.T. Lai, M. Pedram, and S.B.K. Vrudhula, "EVBDD-Based Algorithms for Integer Linear Programming, Spectral Transformation, and Functional Decomposition," *IEEE Trans. Computer-Aided Design,* vol. 13, no. 8, pp. 959-975, 1994.
[22] D.M. Miller, "A Spectral Method for Boolean Function Matching," *Proc. European Design and Test Conf.,* pp. p. 602, 1996.
[23] D.M. Miller, "Spectral Transform of Multiple-Valued Decision Diagrams," *Proc. 24th Int'l Symp. Multiple-Valued Logic,* pp. 89-96, 1994.
[24] C. Moraga and R. Heider, "Tutorial Review on Applications of the Walsh Transform in Switching Theory," *Proc. First Int'l Workshop Transforms and Filter Banks,* pp. 494-512, June 1998.
[25] J.C. Muzio and S.L. Hurst, "The Computation of Complete and Reduced Sets of Orthogonal Spectral Coefficients for Logic Design and Pattern Recognition Purposes," *Computers & Electrical Eng.,* vol. 5, pp. 231-249, 1978.
[26] R.S. Stankovic and B.J. Falkowski, "FFT and Decision Diagrams Based Methods for Calculation of Spectral Transforms," *Proc. IEEE Int'l Conf. Information, Comm., and Signal Processing,* vol. 1, pp. 241-245, 1997.
[27] M. Stankovic, D. Jankovic, and R.S. Stankovic, "Efficient Algorithm for Haar Spectrum Calculation," *Scientific Review,* nos. 21-22, pp. 171-182, 1996.
[28] R.S. Stankovic and C. Moraga, "Reed-Muller-Fourier Representations of Multiple-Valued Functions over Galois Fields of Prime Cardinality," *Proc. IFIP WG 10.5 Workshop Applications of the Reed-Muller Expansion in Circuit Design,* pp. 115-124, 1993.
[29] R.S. Stankovic, T. Sasao, and C. Moraga, "Spectral Transform Decision Diagrams" *Representations of Discrete Functions,* T. Sasao and M. Fujita, eds., pp. 55-92, Kluwer Academic, 1996.
[30] R.S. Stankovic, M. Stankovic, C. Moraga, and T. Sasao, "Calculation of Reed-Muller-Fourier Coefficients of Multiple-Valued Functions through Multiple-Place Decision Diagrams," *Proc. 24th Int'l Symp. Multiple-Valued Logic,* pp. 82-88, 1994.
[31] M.R. Stojic, M.S. Stankovic, and R.S. Stankovic, *Discrete Transforms in Applications,* second ed. Belgrade: Nauka, 1993 (in Serbian).
[32] A. Thayse, M. Davio, and J.-P. Deschamps, "Optimization of Multiple-Valued Decision Algorithms," *Proc. Eighth Int'l Symp. Multiple-Valued Logic,* pp. 171-177, 1978.
[33] M. Thornton, R. Drechsler, and W. Günther, "Probabilistic Equivalence Checking Using Partial Haar Spectral Diagrams," *Proc. IFIP WG 10.5 Workshop Applications of the Reed-Muller Expansion in Circuit Design,* pp. 123-132, 1999.
[34] M. Thornton and V. Nair, "Efficient Calculation of Spectral Coefficients and Their Application," *IEEE Trans. Computer-Aided Design,* vol. 14, no. 11, pp. 1328-1341, 1995.
[35] M. Thornton and V. Nair, "BDD Based Spectral Approach for Reed-Muller Circuit Realization," *IEE Proc. Computerized Digital Technology,* vol. 193, no. 2, pp. 145-150, 1996.
[36] J. Yang and G. De Micheli, "Spectral Transforms for Technical Mapping," Technical Report CSL-TR-91-498, Stanford Univ., Dec. 1991.

Dragan Jankovic received the BSc and MSc degrees in computer science from the Faculty of Electronic Engineering, University of Nis, in 1991 and 1995, respectively. He is currently working toward the PhD degree and as an assistant for programming and logic design at the same university. His research interests include decision diagrams, multiple-valued logic, spectral techniques in logic design, and programming.

Radomir S. Stankovic received the BE degree in electronic engineering from the Faculty of Electronics, University of Nis, in 1976, and the MSc and PhD degrees in applied mathematics from the Faculty of Electrical Engineering, University of Belgrade, Serbia, Yugoslavia, in 1984 and 1986, respectively. Currently, he is a professor in the Department of Computer Science, Faculty of Electronics, University of Nis, Serbia, Yugoslavia. His research interests include spectral techniques, switching theory and multiple-valued logic, and signal processing.

Rolf Drechsler received his diploma and Dr. phil. nat. degree in computer science from the J.W. Goethe-University in Frankfurt am Main, Germany, in 1992 and 1995, respectively. He was with the Institute of Computer Science at the Albert-Ludwigs-University of Freiburg im Breisgau, Germany from 1995 to 2000. He recently joined the Corporate Technology Department of Siemens AG, Munich. His research interests include verification, logic synthesis, and evolutionary algorithms. He is a member of the IEEE.