Decision Diagram Optimization Using Copy Properties

Dragan Janković　Radomir S. Stanković　Rolf Drechsler

University of Niš
Faculty of Electronic Eng.
Yugoslavia

University of Bremen
Inst. of Computer Science
Germany

Abstract

In this paper, we propose an approach to the reduction of sizes of Multi-Terminal Binary Decision Diagrams (MTBDDs) [3] by using the copy properties of discrete functions. The underlying principles come from *copy theory* of discrete signals considered in [5, 6, 7]. We propose two modifications of MTBDDs, called Copy DDs (CDDs) and Half Copy DDs (HCDDs), using the corresponding copy operations from copy theory.

Functions having different types of copy properties can be efficiently represented by the proposed Copy DDs. Examples are Walsh and Reed-Muller functions as well as different binary codes.

1. Introduction

Decision Trees (DTs) for representation of discrete function can be considered as a graphical representation of an enumeration procedure expressing lexicographically all the elements in a sequence $\mathcal{Z}(f)$ defining a discrete function f. Decision Diagrams (DDs) are derived by the reduction of DTs, obtained by sharing isomorphic subtrees and deleting the redundant information from a DT. The reduction of a DT into a DD is possible iff there are constant or mutually equal subsequences $\mathcal{Z}_{(i)}$ in $\mathcal{Z}(f)$, since such subsequences result in appearance of isomorphic subtrees in the DT. Smallest isomorphic subtrees are constant nodes showing equal values. DDs with complemented edges [2] are a generalization derived by considering as isomorphic subtrees representing subsequences $\mathcal{Z}_{(i)}$ and $-\mathcal{Z}_{(i)}$. The sign minus is considered as logic negation in bit-level DDs, and as arithmetic negation in word-level DDs. Therefore, the equality up to the sign, is a single relationship that is exploited in reduction of DDs.

In this paper, we propose an extended library of relationships between sequences producing isomorphic subtrees permitting reduction of DTs and resulting in more compact DDs.

2. Background Theory

Binary Decision Diagrams (BDDs) [1] are defined by using the Shannon expansion rule $f = \bar{x}_i f_0 \oplus x_i f_1$, where f_0 and f_1 are co-factors of f for $x_i = 0$ and $x_i = 1$, respectively. In a BDD, we denote outgoing edges of a node S pointing to the co-factors f_0 and f_1 by w_0 and w_1, respectively. Therefore, a node in a BDD is represented by (S, w_0, w_1).

In BDDs with complemented edges (CEs) [2], we consider as isomorphic two subtrees representing subfunctions f and g, if $g = \overline{f}$, where bar denotes the logic complement. We denote $g = D(f)$, where D is the operator of logic complementation.

Therefore, in BDDs with CEs, the nodes are represented as $(S, Q(w_0), Q(w_1))$, with $Q \in \{I, D\}$, where I denotes the identity operator. Alternatively, the Shannon nodes with CEs can be considered as a new type of node. Thus, $(S, Q(w_0), Q(w_1)) \to (S_Q, w_0, w_1)$.

Generalization to MTBDDs [3] is straightforward. For example, if negation by logic complementation is interpreted in the integer domain as the multiplication by -1, we get MTBDDs with negated edges. In this case, D performs multiplication by -1. Extension to other DDs is also straightforward. For example, in Fourier DDs, it has been suggested to use for D also the multiplication with the complex unity $j = \sqrt{(-1)}$, complex-conjugation and properties of symmetry, sqew-symmetry, and Hermitean properties of matrices [9]. In matrix-valued Fourier DDs, D is extended to consider as isomorphic the subtrees representing the complex-conjugate transpose matrices, Hermitean matrices.

In optimization of MTBDDs with copy operations, we consider as isomorphic subtrees representing f and $g = D(f)$, where D is an operator in the set of copy operators [11] defined below.

2.1. Copy Operations

In this section, we briefly present the basic notions from copy theory [5].

As it is pointed out in [5], the copy feature is an important property of discrete signals. If it is assumed that a discrete signal is represented by a sequence of numbers, then there are two copy methods to generate a new sequence (that means a new signal) from the starting sequence.

Definition 1 *Given a sequence $s = (s_1, s_2, \cdots, s_{n-1}, s_n)$. The even symmetry copy method transforms s into the sequence $s_r = (s_1, s_2, \cdots, s_{n-1}, s_n, s_n, s_{n-1}, \cdots, s_2, s_1)$ (Figure 1-a).*

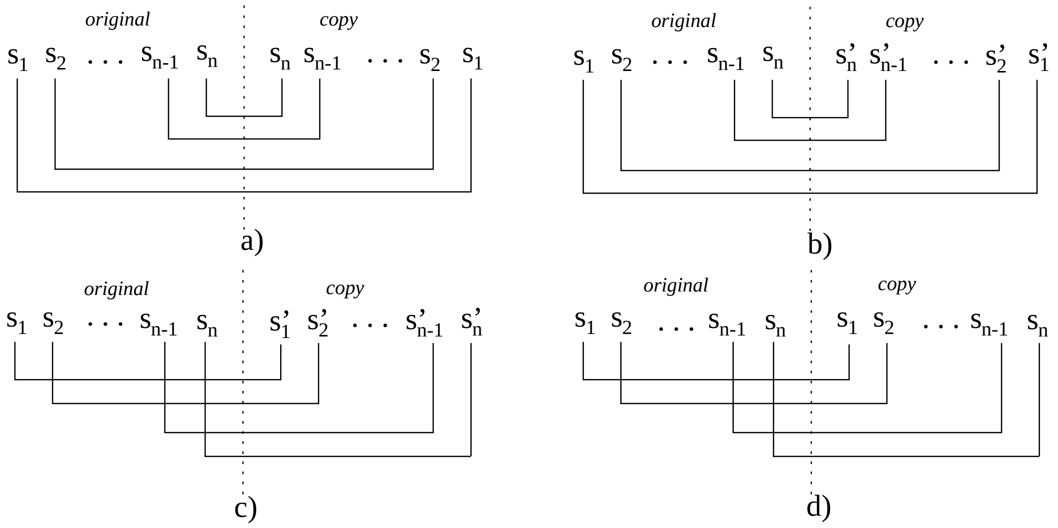

Figure 1. a) even symmetry copy b) odd symmetry copy c) minus shift copy d) plus shift copy.

Definition 2 *Given a sequence $s = (s_1, s_2, \cdots, s_{n-1}, s_n)$. The odd symmetry copy method transforms s into the sequence $s'_r = (s_1, s_2, \cdots, s_{n-1}, s_n, s'_n, s'_{n-1}, \cdots, s'_2, s'_1)$ where s'_i is the result of a suitably defined operation of negation not, i.e., $s'_i = not(s_i)$, (Figure 1-b).*

Definition 3 *Given a sequence $s = (s_1, s_2, \cdots, s_{n-1}, s_n)$. The minus shift copy method transfers s into the sequence $s_r = (s_1, s_2, \cdots, s_{n-1}, s_n, s_1, s_2, \cdots, s_{n-1}, s_n)$ (Figure 1-d).*

Definition 4 *Given a sequence $s = (s_1, s_2, \cdots, s_{n-1}, s_n)$. The plus shift copy method transforms s into the sequence $s_r = (s_1, s_2, \cdots, s_{n-1}, s_n, s'_1, s'_2, \cdots, s'_{n-1}, s'_n)$ where s'_i is result of a suitably defined operation of negation not, i.e., $s'_i = not(s_i)$, (Figure 1-c).*

3. Copy Operations and MTBDDs

Binary input integer output functions, short integer functions, are given as mappings $\{0,1\}^n \rightarrow Z$. Integer functions can be efficiently represented by Multi-Terminal Binary Decision Diagrams (MTBDDs) [3].

MTBDDs consist of nonterminal and terminal nodes. Each nonterminal node, labeled with a variable x, has two outgoing edges corresponding to the logic values 0 and 1 for x. Terminal nodes have integer values. In what follows, it will be assumed that a MTBDD is ordered and reduced, i.e., variables occur in the same order along all the paths in the DD and there are no isomorphic subgraphs. The following example illustrates MTBDDs.

Example 1 *Figure 2 shows the MTBDD for integer function f, given by the truth vector*

$$F = [0,0,0,1,0,1,2,1,1,2,1,0,1,0,0,0]^T.$$

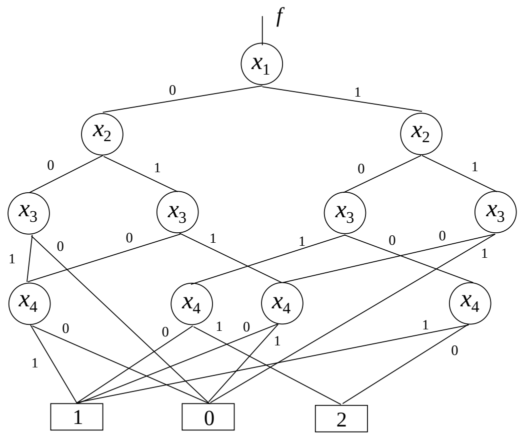

Figure 2. MTBDD of function f in Example 1.

In MTBDDs, the smallest subtree consists of a nonterminal node at the last level and two constant nodes. In matrix notation, the copy operations applied to such subtrees can be described by the following matrices

$$S0 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad S1 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \quad T1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

For application of copy operations to the subtrees rooted at the nodes at upper levels in the MTBDD, we use the second-order operators defined as follows [11], [10], [12], [13].

Definition 5 *Let A be a sequence of complex numbers. The operators E, $\overline{E}$, and $-\overline{E}$ are defined by*

$$EA = AE = A,$$

$$\begin{aligned}\overline{E}A &= A\overline{E} = \overline{A},\\ (-\overline{E})A &= A(-\overline{E}) = r\overline{A},\end{aligned}$$

where $\overline{A}$ stands for the reversed sequence A, and $r\overline{A}$ stands for the reversed sequence A with sign of elements reversed.

Definition 6 *The copy operator G_2 is a second-order matrix operator defined as*

$$G_2 = \begin{bmatrix} E & \overline{E} \\ E & -\overline{E} \end{bmatrix}.$$

Definition 7 *The shift operator S_2 is a second-order operator defined as*

$$S_2 = \begin{bmatrix} E & 0 \\ 0 & E \end{bmatrix}.$$

4. Copy Decision Diagrams

By using the copy operations, we define three new nodes that can be used in MTBDDs besides the Shannon nodes.

Definition 8 *Given a Shannon node Sh representing a function g. The even symmetry node $S0$ derived from Sh is a node that represents a function $f_{S0} = [g, g_r]^T$, where g is the function corresponding to the successor of the node $S0$ and g_r is the function derived from g by assigning the function values for g in the reverse order, i.e., $g_r(00\cdots0) = g(11\cdots1)$, etc.*

Example 2 *Figure 3 shows an even symmetry node S0 derived from a Shannon node Sh. If a function g represented by Sh is given by the vector $G = [v_1, v_2, v_3, v_4]^T$, then the function f_{S0} represented by $S0$ is given by the vector $F_{S0} = [v_1, v_2, v_3, v_4, v_4, v_3, v_2, v_1]^T$.*

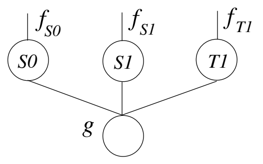

Figure 3. S0 node, S1 node, and T1 node.

Definition 9 *The function corresponding to an* odd symmetry *node $S1$ is given by $f_{S1} = [g, g_r']^T$, where g is the function corresponding to the successor of the node S1, and g_r' is the function derived from g by* "complemented" *values (not(g)) in reverse order.*

Example 3 *Figure 3 shows a node $S1$ derived from the Shannon node Sh. If a Boolean function g represented by Sh is given by the vector $G = [0, 0, 1, 0]^T$, then the function f_{S1} represented by $S1$ is given by the vector $F_{S1} = [0, 0, 1, 0, 1, 0, 1, 1]^T$ if the operation not is defined as a Boolean operation NOT or as $not(x) = 1 - x$ if the logic values for the Boolean function are interpreted as integers.*

It should be noted that the complementation can be defined in a different way, permitting consideration of different generalizations of Shannon nodes.

Definition 10 *The function represented by the* plus shift copy *node $T1$ is given as $f_{T1} = [g, g']^T$, where g is the function corresponding to the successor of the node $S0$ while g' is the function derived from g by the complementation of the values for g, i.e., g'=not(g).*

Example 4 *Let a node T1 is shown in Figure 3 . If Boolean function g is given by the truth vector $G = [0, 0, 1, 0]^T$, the function f_{T1} is given by the truth vector $F_{T1} = [0, 0, 1, 0, 1, 1, 0, 1]^T$ if the operation not is the Boolean operation NOT.*

Definition 11 *A Copy Decision Diagram (CDD) is a DD consisting of terminal nodes and nonterminal nodes from the set $\{Sh, S0, S1, T1\}$, where Sh is the Shannon node, $S0$ is the even symmetry copy node, $S1$ is the odd symmetry copy node and $T1$ is the plus shift copy node.*

Note, that minus shift copy node is not introduced because it corresponds to the Shannon node having both edges pointing to the same node. Such Shannon node is deleted by the reduction rules for MTBDDs and it is denoted as "cross point" like other deleted nodes in MTBDDs.

Example 5 *Figure 4 shows a CDD representing the function f given in Example 1 with the operation not defined as $not(x) = (x + 1) \quad mod \quad 3$. It is shown the function represented by each node in the CDD.*

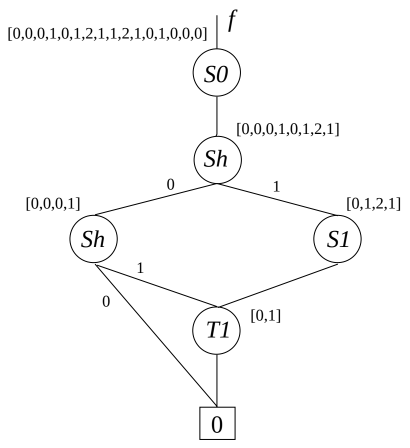

Figure 4. CDD for function f in Example 4.

4.1. Half CDD

CDDs are a simple example of application of copy theory to DDs. A further step in this direction can be done if we modify the definition of CDD nodes in the following way:

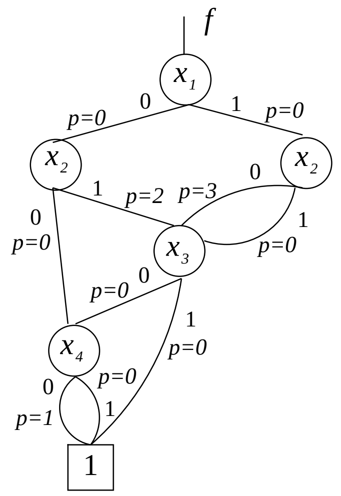

Figure 5. HCDD for function f in Example 6.

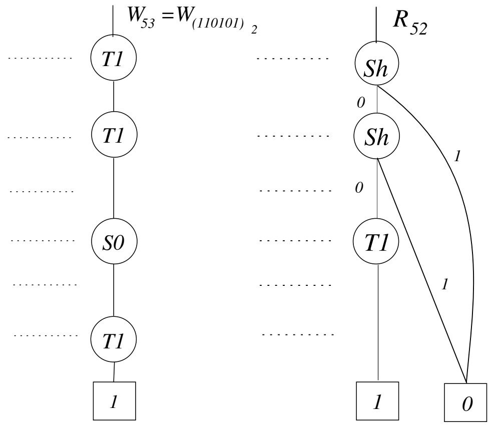

Figure 6. CDD for the Walsh function W_{53} and CDD for the Reed-Muller function R_{52} of order 6.

Denote by a and b the low and the high successors of a Shannon node v. A *Half-Copy Decision Diagram* (HCDD) is a DD consisting of nodes representing the function $f_v = [t(f_a)t(f_b)]$ where

$$t(x) = \begin{cases} x, & p = 0, \\ not(x), & p = 1, \\ x_r, & p = 2, \\ not(x_r), & p = 3, \end{cases}$$

where not and $_r$ denote the operation of negation and order reversing. In this definition p is a parameter assigned to the node, and it can be represented by a field assigned to the node or as a label at the outgoing edge.

Example 6 *Let*

$$F = [0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1]^T$$

be the truth vector of the 4-variable Boolean function f. HCDD representing f is shown in Figure 5. Parameter p is given for each edge.

It should be noted that HCDDs for Boolean functions are similar to BDDs with dual edges or dual markers as considered in [4].

5. Examples

In this section, we give some examples of functions that have copy properties. Such functions can be efficiently represented by CDDs and HCDDs. We define the rules for the generation of these DDs for the considered functions.

5.1. Representation of Walsh Functions

The Walsh matrix of order n, $\mathbf{W}(n)$, whose columns are the Walsh functions in natural or Hadamard ordering is given as

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

where $\bigotimes$ denotes the Kronecker product.

Example 7 *The Walsh matrix of order 3 is given by*

$$\mathbf{W}(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

The CDD representing the i-th Walsh function W_i can be generated by using the following rule.

Let $(i_1, i_2, \cdots, i_n)$ be the binary representation for i.

$$if \begin{cases} i_j = 0, & \text{at level } j \text{ no node,} \\ i_j = 1, & \text{if preceding node was T1 put node S0 at level } j, \\ & \text{if preceding node was S0 put node T1 at level } j, \\ & \text{if preceding node is terminal node put node T1} \\ & \quad \text{at level } j. \end{cases}$$

Example 8 *The CDD for the Walsh function $W_{53} = W_{(110101)_2}$, is shown in Figure 6.*

Note that in this case the operation not is defined as the multiplication by -1.

Example 9 *CDD and HCDD representing the set of Walsh functions of order 3 are shown in Figure 7 a) and b), respectively.*

5.2. Representation of Reed-Muller Functions

The Reed-Muller matrix of order n, $\mathbf{R}(n)$, whose columns are the Reed-Muller functions is given as

$$\mathbf{R}(n) = \bigotimes_{i=1}^{n} \mathbf{R}(1) = \bigotimes_{i=1}^{n} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

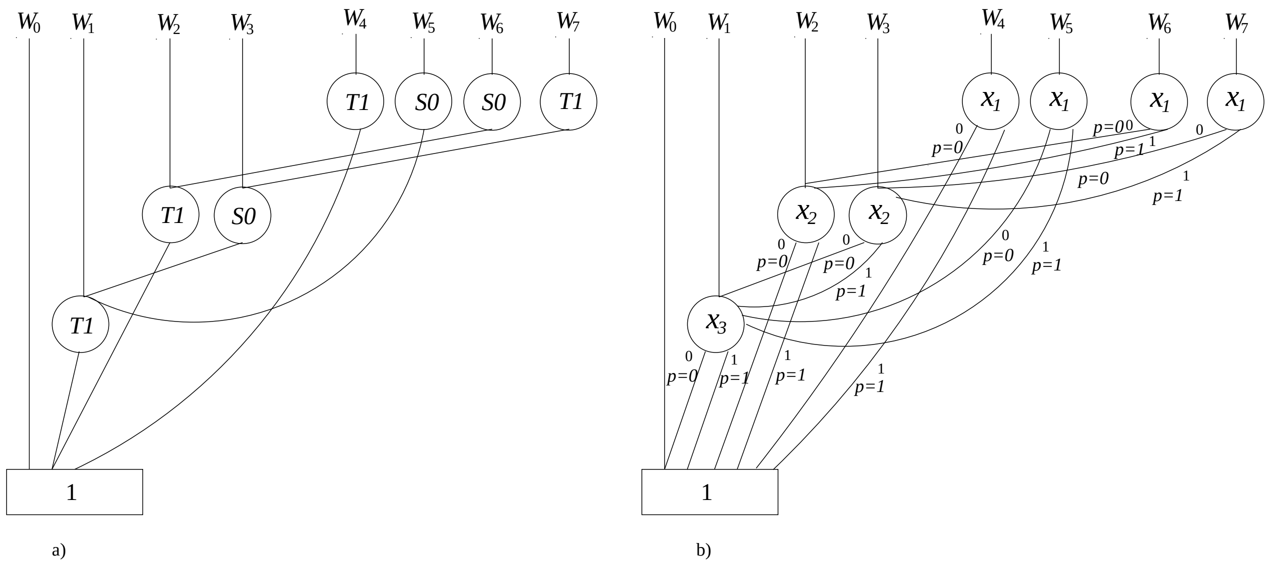

Figure 7. CDD(a) and HCDD(b) representing set of Walsh functions for $n = 3$.

Let R_i be the i-th Reed-Muller function in the set of Reed-Muller functions of order n. The CDD representing R_i can be generated by using the following rule:

Let $(i_1, i_2, \cdots, i_n)$ be the binary representation for i.

$$if \begin{cases} i_j = 1, & \text{at level } j \text{ no node,} \\ i_j = 0, & \text{put node T1 at level } j \text{ if } i_j \text{ is '0' with the smallest index in binary representation of } i, \\ otherwise & \text{put the node Sh at level } j. \end{cases}$$

Example 10 *The CDD for the Reed-Muller function $R_{52} = R_{(110100)_2}$ of order 6 is shown in Figure 6.*

The number of nodes in the CDD representing an arbitrary Reed-Muller function of order n, is less or equal than n, while the number of nodes in the CDD representing the set of Reed-Muller functions of order n, is equal to $2^n - 1$.

5.3. Representation of Binary Codes

Some binary codes have copy properties and due to that can be efficiently represented by CDDs. Consider the realization of the natural code, Gray code, and K-codes given in Table 1. Figure 8 shows the CDD representing these three codes.

It can be shown that the number of nodes in the CDD representing these three n-bit codes is $n(n + 1)/2$.

The HCDDs for the natural code, Gray code, and K-code are similar to the corresponding CDDs shown in Figure 8. Each $T1$ node is replaced with a node whose both edges point to same node and the high edge performs the operation not, i.e., in this case, $p = 1$.

Table 1. Natural, Gray, and K-code

	natural			Gray			K-code		
dec.in.	n_0	n_1	n_2	g_0	g_1	g_2	k_0	k_1	k_2
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	1
2	0	1	0	0	1	1	0	1	1
3	0	1	1	0	1	0	0	1	0
4	1	0	0	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0
6	1	1	0	1	0	1	1	0	0
7	1	1	1	1	0	0	1	0	1

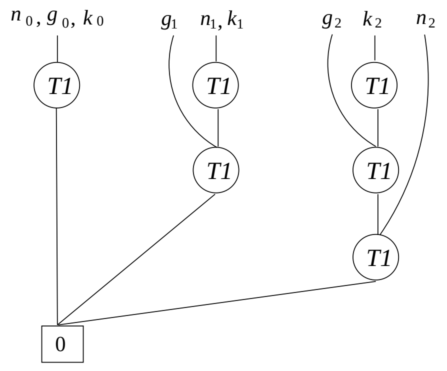

Figure 8. CDD representing 3-bit natural code, Gray code, and K-code.

6. Procedures for the generation of CDDs from MTBDDs

Starting from MTBDDs representing a given function f, CDDs can be generated by using the following steps. Procedure **Replace** is called by the root node in the MTBDD as an argument.

It is interesting to discuss the order of examining the meaning of a node v. As is shown in Figure 9, we assume the following hierarchy of nodes. We first check if v is a $T1$ node, than $S0$ and at the last, $S1$. This fixed order ensures the canonicity of CDD. In some cases, the node v can be considered as the $T1$ or the $S0$ node. For example, a subfunction 11000011 can be represented by both $T1$ and $S0$ nodes, but if the order of interpretation of nodes is fixed, we will use the node $T1$.

```
function S0($l$, $h$:node):boolean;
begin
    if ( $l$ is terminal && $h$ is terminal ) return $l == h$;
    return S0($l.low$, $h.high$) && S0($l.high$, $h.low$);
end;

function S1($l$, $h$:node):boolean;
begin
    if ( $l$ is terminal && $h$ is terminal ) return $l == not(h)$;
    return S1($l.low$, $h.high$) && S1($l.high$, $h.low$);
end;

function T1($l$, $h$:node):boolean;
begin
    if ( $l$ is terminal && $h$ is terminal ) return $l == not(h)$;
    return T1($l.low$, $h.low$) && T1($l.high$, $h.high$);
end;

procedure Replace($v$:node)
begin
    if(T1($v.low$, $v.high$)) then
        replace $v$ by T1-node; Replace($v.low$); exit;
    if(S0($v.low$, $v.high$)) then
        replace $v$ by S0-node; Replace($v.low$); exit;
    if(S1($v.low$, $v.high$)) then
        replace $v$ by S1-node; Replace($v.low$); exit;
    Replace($v.low$);
    Replace($v.high$);
end;
```

Figure 9. Sketch for procedure Replace

7. Operations over CDDs and HCDDs

In this section we consider the algorithms to perform different operations over CDDs and HCDDs.

7.1. Level Exchange

Reduction of the size of DDs is possible by changing the order of variables. The same applies to CDDs and HCDDs. A suitable order of variables that reduce the size of a DD can be determined by a level exchange (LE). Therefore, it is interesting to consider the LE operation in CDDs.

It is obvious that performing LE operation is similar to MTBDDs. However, it is not a local operation as in MTBDDs, since for some nodes we have to perform the complementation or/and inversion. These operations are simple and do not increase the number of nodes in CDD. The rules for reverse operation are given in Figure 10. LE operation over HCDDs is much more complex than over CDDs.

7.2. Binary operations over CDDs

Figure 11 shows the rules for performing a binary operations over two CDDs.

It is obvious that binary operations over CDDs are recursive, however, in some cases, it is required to perform the operations of reversing and negation of function values over subgraphs.

8. Experimental Results

For the experimental estimation of properties of CDDs, we developed a program in the CUDD environment [8], for transforming MTBDDs into CDDs and HCDDs. The size of CDDs and the corresponding MTBDDs from which they are derived for some benchmark functions are given in Table 2. The used benchmark functions are Boolean functions, therefore their MTBDDs are actually BDDs. We compare the sizes of BDDs without CEs, BDDs with CEs, and CDDs without sifting and with sifting. For CDDs, the number of $T1$, $S0$, and $S1$ nodes is given.

Since the experiments are performed for Boolean functions, the operation not is defined as the logic NOT.

Reduction capability of CDDs for Boolean function is not so large but, in general for MTBDDs and integer-valued functions, we expect the better results, since in this case CEs are not usually used for many possible different values for constant nodes.

Experimental results show that for large benchmark functions the number of nodes $T1$, $S0$, and $S1$ is reasonably large.

Experimental results for HCDDs are given in Table 3. BDDs without CEs and HCDDs, without sifting and with sifting are compared. The HCDD size is on the average 36.57% smaller than that of BDDs without sifting and 43.99% for BDDs with sifting. There are examples where the HCDD size is more than 80% smaller than the size of the corresponding BDD without CEs.

As we noted above, for Boolean function BDDs with CEs and dual markers are a subset of HCDDs. Therefore, results given in [4] can be used to estimate the complexity of HCDDs. Since dual markers which increase the size of BDD for some functions (for example for multipliers [4]), the size of HCDDs is equal or smaller than the size of corresponding BDDs with dual markers.

9. Closing Remarks

This paper introduced the usage of copy properties of functions in reduction of size of DDs. Initial definitions of copy properties are taken from [5]. Based on those copy properties two modifications of MTBDDs, called CDDs and HCDDs, have been proposed.

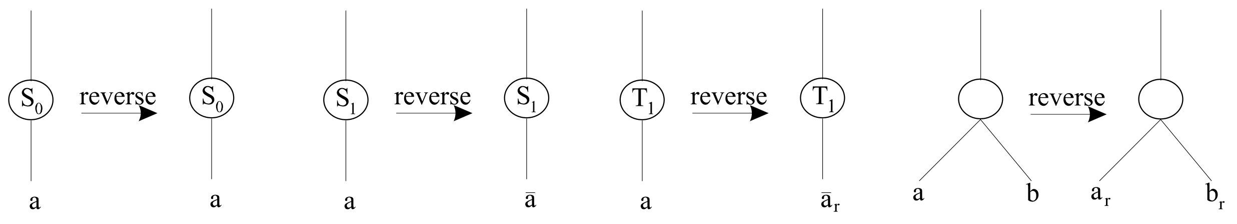

Figure 10. Reverse operation rules.

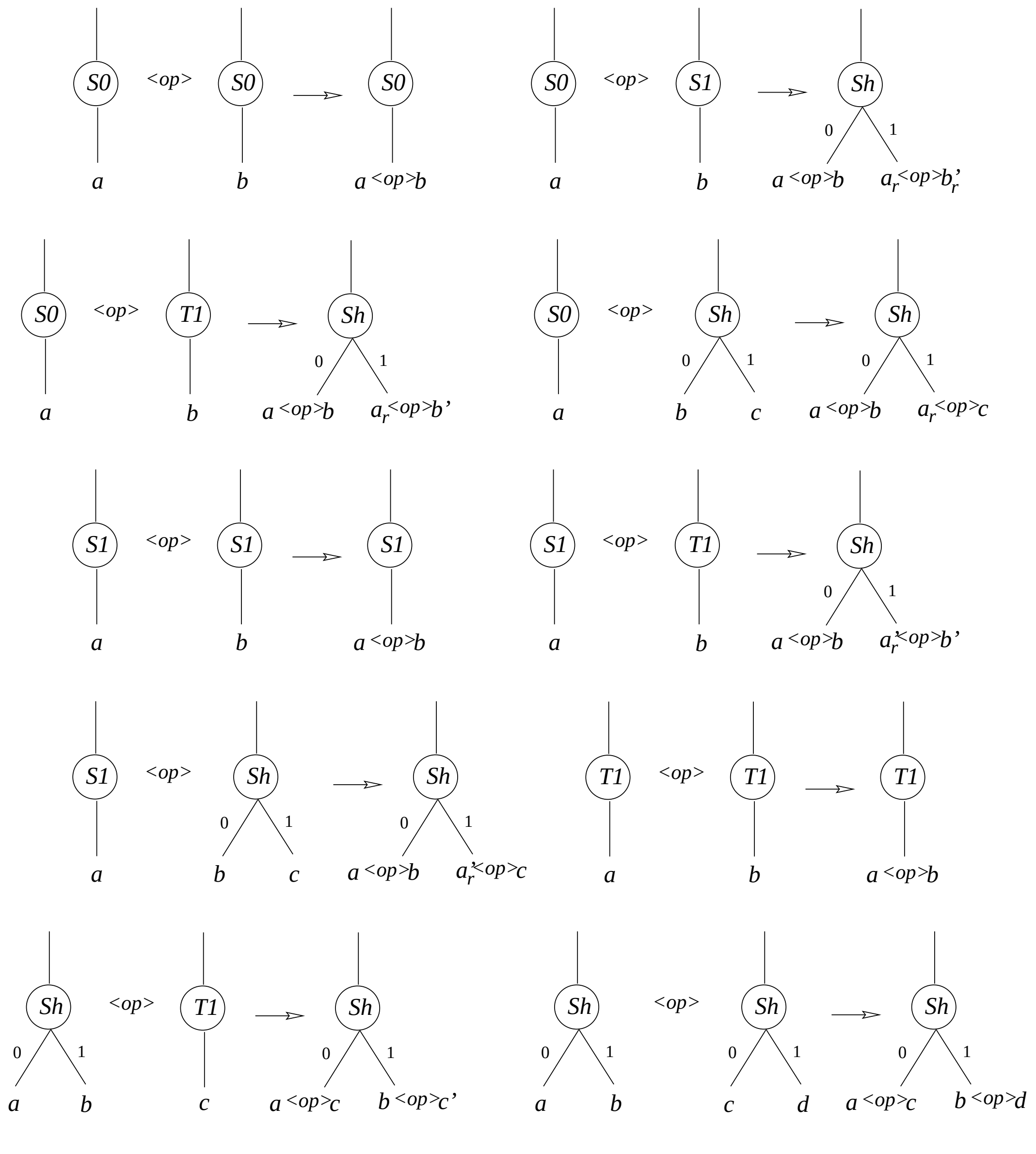

Figure 11. CDD binary operation rules.

Table 2. Experimental results

			without sifting						*with sifting*					
circuit	*in*	*out*	*BDD*	*BDD-CE*	*CDD*	T1	S0	S1	*BDD*	*BDD-CE*	*CDD*	T1	S0	S1
add2	4	3	15	11	11	5	0	0	11	8	8	4	0	0
add3	6	4	42	29	29	10	0	0	20	13	13	6	0	0
alu2	10	8	257	230	227	6	2	1	248	161	160	12	1	0
alu4	14	8	1219	1181	1166	6	4	3	903	602	598	39	10	5
des	256	245	15281	11461	11216	293	163	135	13997	3053	2736	301	386	63
rd53	5	3	23	16	15	5	2	1	23	16	15	5	2	1
rd73	7	3	43	30	24	7	2	5	43	30	24	7	2	5
rd84	8	4	59	41	41	8	4	5	59	41	41	8	5	4
9sym	9	1	33	24	18	1	3	2	33	24	18	1	3	2
c880	60	26	346688	346659	346654	79	0	89	7084	7062	7042	36	0	6
5xp1	7	10	88	73	73	15	1	0	68	41	41	9	0	0
clip	9	5	254	225	225	8	2	0	108	86	84	8	0	6

Table 3. Experimental results for HCDD

			without sifting		*with sifting*	
circuit	*in*	*out*	*BDD*	*HCDD*	*BDD*	*HCDD*
add2	4	3	15	10	11	8
add3	6	4	42	24	20	13
alu2	10	8	257	225	248	160
alu4	14	8	1219	1157	903	579
des	256	245	15281	6150	13997	2638
rd53	5	3	23	14	23	14
rd73	7	3	43	22	43	22
rd84	8	4	59	33	59	33
9sym	9	1	33	15	33	15
5xp1	7	10	88	61	68	27
clip	9	5	254	174	108	84

Experimental results show that reduction in size by using HCDDs is better than for CDDs. In general, the size of CDDs and HCDDs is smaller or equal to the size of the corresponding BDD or MTBDD.

Disadvantages of CDDs and HCDDs is the non local character of the level exchange operation. This means that sifting cannot be used for minimization of CDDS and HCDDs as efficient as for BDDs.

Besides copy operations that have been already defined in copy theory [5], other copy operations can be defined by exploiting structural properties of DDs. Therefore, further reductions based on copy properties can be defined. This is focus of current work.

References

[1] Bryant, R.E., ”Graph-based algorithms for Boolean functions manipulation”, *IEEE Trans. Comput.*, Vol.C-35, No.8, 1986, pp.667-691.

[2] Brace, K.S., Rudell, R.L., Bryant, R.E., ”Efficient implementation of a BDD package, In *Design Automation Conf.*, 1990, 40-45.

[3] Clarke, E.M., Fujita, M., McGeer, P., McMillan, K.L., Yang, J., Zhao, X., “Multi-terminal binary decision diagrams: An efficient data structure for matrix representation”, In *Int’l workshop on Logic Synth.*, 1993, 6a:1-15.

[4] Miller, D.M., Drechsler, R., “Dual edge operations in reduced ordered binary decision diagrams”, *International Symposium on Circuits and Systems (ISCAS’98)*, pp. VI:159-VI:162, 1998

[5] Minglu J., Qishan, Z., “Copy Theory of Signals and Its Applications”, in *Recent Developments in Abstract Harmonic Analysis with Applications in Signal Processing*, Edited by Stanković, R.S., Stojić, M.R., Stanković, M.S., Nauka, Belgrade, 1996, 313-328.

[6] Minglu, J., Qishan, Z., “Shift copy analysis of signals”, *Proc. of ICT’94*, England, Jan. 1994.

[7] Qishan, Z., “A summary of Bridge functions”, in *Recent Developments in Abstract Harmonic Analysis with Applications in Signal Processing*, Edited by Stanković, R.S., Stojić, M.R., Stanković, M.S., Nauka, Belgrade, 1996, 305-312.

[8] Somenzi, F., *CUDD: CU Decision Diagram Package Release 2.2.0.*, University of Colorado at Boulder, 1998.

[9] Stanković, R. S., ”Fourier decision diagrams on finite non-Abelian groups with preprocessing”, *Proc. 27th Int. Symp. on Multiple-Valued Logic*, Antigonish, Nova scotia, Canada, May 1997, 281-286.

[10] Stanković, R.S., Zhang, Q., “Complex Bridge functions”, *Proc. 4th Int. Workshop on Spectral Techniques*, March 15-17, 1994, Beijing, China, .

[11] Zhang, Q., Zhijing, M., “A group of three-valued functions”, *Proc. Int. Conf. on Signal Processing Beijing ’90*, October 22.-26. 1990, 1183-1186.

[12] Zhihua, L., Zhang, Q., “Introduction to bridge functions”, *IEEE Trans.*, Vol.EMC-25, No.4, 1983, 459-464.

[13] Zhihua, L., Zhang, Q., “Ordering of Walsh functions”, *IEEE Trans.*, Vol.EMC-25, No.2, 1983, 115-119.