Efficient Representation and Manipulation of Switching Functions Based on Ordered Kronecker Functional Decision Diagrams *

R. Drechsler[1] A. Sarabi[2] M. Theobald[1] B. Becker[1] M. A. Perkowski[2]

[1] Computer Science Department
Johann Wolfgang Goethe University
D-60054 Frankfurt am Main, Germany
email: <name>@kea.informatik.uni-frankfurt.de

[2] Department of Electrical Engineering
Portland State University
Portland, OR 97207-0751, USA
email: <name>@ee.pdx.edu

Abstract

An efficient package for construction of and operation on ordered Kronecker Functional Decision Diagrams (OKFDD) is presented. OKFDDs are a generalization of OBDDs and OFDDs and as such provide a more compact representation of the functions than either of the two decision diagrams. In this paper basic properties of OKFDDs and their efficient representation and manipulation are presented. Based on the comparison of the three decision diagrams for several benchmark functions, a 25% improvement in size over OBDDs is observed for OKFDDs.

1 Introduction

The increasing complexity of modern VLSI circuitry is only manageable through advanced CAD systems which as one important component include logic synthesis tools. Problems encountered in synthesis can often be formulated in terms of Boolean functions. The efficiency of the manipulation algorithms on the functions largely depends on the data structures chosen for the representation of the functions. Currently, the most popular data structures are Ordered Binary Decision Diagrams (OBDDs) introduced by Bryant [5]. OBDDs have been most widely used in logic synthesis, verification, testing, modeling and simulation [3, 11, 6]. They allow efficient manipulations of switching functions and such applications as testing for satisfiability and equivalence. Nevertheless, there exist functions for which efficient OBDDs do not exist [13].

Still other decision diagrams have been studied which are especially pertinent to the design with AND/XOR circuits. Realization with AND/XOR logic often leads to a more compact realization of switching functions than AND/OR [17, 14]. The networks realized in this logic have been recently of more interest mainly due to the new technologies which make this realization more practical. The new FPGA technologies have made it possible to utilize universal logic blocks which do not distinguish between the types of logic used. This calls for more integration of AND/XOR logic into mainstream synthesis.

Decision Diagrams such as the so-called (Ordered) Functional Decision Diagrams (OFDD) [10] and (Ordered) Kronecker Functional Decision Diagrams (OKFDD) [15] utilize advantages of AND/XOR representation in their structure. While OFDDs are based solely on AND/XOR representations, OKFDDs are a generalization of OBDDs and OFDDs and provide a combination of AND/XOR and MUX-based representations. As they are not restricted to just one type of decomposition, they provide a more compact representation of the switching functions than either OBDDs or OFDDs.

It has been shown that for certain classes of Boolean functions, the size of an OFDD is exponentially smaller than the OBDD representation of the same function and vice versa [2]. Thus, it is possible to represent functions efficiently using OKFDDs for which no efficient OBDDs or OFDDs exist.

In this paper, an efficient package for construction of and operations on OKFDDs is introduced. Here, a reduction type for the minimization of OFDDs [1] is utilized which allows the usage of complemented edges analogous to that for OBDDs [3]. As the size of an OKFDD largely depends on the chosen variable ordering as well as the type of decomposition, an algorithm for minimization of OKFDDs based on dynamic variable ordering is introduced. In this approach, a modified version of the sifting algorithm introduced by Rudell [13] is utilized. In addition to the variable ordering, this sifting algorithm chooses an appropriate decomposition type for the variables. Furthermore, the implementation of new efficient synthesis algorithms for OKFDDs is given. Finally, experimental results are provided which confirm the compactness of OKFDDs and show the efficiency of the approach.

2 Decision Diagrams

In this section, essential definitions and properties of OKFDDs are presented. As OBDDs and OFDDs are special versions of OKFDDs, these two structures are also defined and will be compared with each other. Procedures for reduction of OKFDDs are also presented.

The core of the data structures is a decision diagram (DD), which is a directed acyclic graph with some additional properties.

Definition 1 A *decision diagram (DD)* over $X_n := \{x_1, x_2, \ldots, x_n\}$ is a rooted directed acyclic graph $G = (V, E)$ with vertex set V containing two types of vertices, *non-terminal* and *terminal* vertices. A non-terminal vertex v is labeled with a variable from X_n, called the *decision*

*This work was supported in part by DFG grant Be 1176/4-1 and NSF grant MIP-9110772.

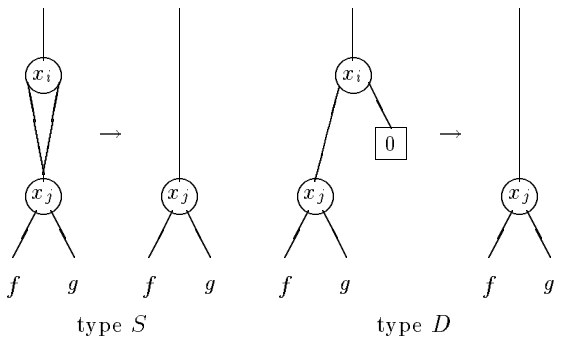

Figure 1: Reduction types

variable for v, and has exactly two successors denoted by $low(v), high(v) \in V$. A terminal vertex v is labeled with a 0 or 1 and has no successors.

The size of a DD, denoted by $|DD|$, is given by its number of internal nodes. If DDs are to be used as a data structure in design automation, it turns out that further restrictions on their structure will be necessary. Two such restrictions are defined below:

Definition 2 A DD is *free* if each variable is encountered at most once on each path in the DD from the root to a terminal vertex. A DD is *ordered* if it is free and the variables are encountered in the same order on each path in the DD from the root to a terminal vertex.

In the following, letter O will be used to denote ordered DDs.

It is possible to define certain reductions on the decision diagrams in order to reduce their size. Three reduction types will be used.

Type I: Delete a node v' whose successors are identical to the successors of another node v and redirect the edges pointing to v' to point to v.

Type S: Delete a node v whose two outgoing edges point to the same node and connect the incoming edges of the deleted node to the corresponding successor.

Type D: Delete all nodes v whose successor $high(v)$ points to the terminal 0 and connect the incoming edges of the deleted node to the corresponding successor.

In Figure 1, graphical representations of reductions of type S and D are shown. While each node in a DD is a candidate for the application of reduction type I, candidates for the application of the remaining reduction types have to be defined in advance: The non-terminal nodes of a DD are partitioned in two classes, Shannon nodes and Davio nodes. Reduction of type S (D) is only allowed at Shannon (Davio) nodes. In the following it is assumed that this partition is given.

Definition 3 A DD is *reduced* if no reductions can be applied to the DD.

Two DDs, G_1 and G_2, are called *equivalent* iff G_2 results from G_1 by repeated applications of reductions and inverse reductions. A DD, G_2, is called the *reduction* of a DD, G_1, if G_1 and G_2 are equivalent and G_2 itself is reduced.

A careful analysis of the proofs in [5, 8] shows that the following lemma is valid for DDs:

Lemma 1 The reduction of a free DD, G, is uniquely determined and can be computed in linear time in the size of G.

Until now it has not been defined how DDs can be related to Boolean functions. To do this, the following notions are helpful. All nodes labeled with the same variable are denoted as a *level* in the following. Let $f : \mathbf{B}^n \rightarrow \mathbf{B}$ be a Boolean function over the variable set X_n. Then f_i^0 denotes the *cofactor* of f with respect to $x_i = 0$, defined by $f_i^0(x) := f(x_1, .., x_{i-1}, 0, x_{i+1}, .., x_n)$ for $x = (x_1, x_2, \ldots, x_n) \in \mathbf{B}^n$. Similarly, f_i^1 denotes the cofactor for $x_i = 1$. Finally, f_i^2 is defined as $f_i^2 := f_i^0 \oplus f_i^1$, where $\oplus$ stands for Exclusive OR operation. (Notice that the three functions f_i^0, f_i^1, f_i^2 can naturally be interpreted as Boolean functions from $\mathbf{B}^{n-1}$ to $\mathbf{B}$ defined over the variables $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$.) Using the above definitions, the following decompositions can be proven for an arbitrary Boolean function f:

$$f = \bar{x_i} f_i^0 + x_i f_i^1 \quad \textit{Shannon} \tag{1}$$

$$f = f_i^0 \oplus x_i f_i^2 \quad \textit{positive Davio} \tag{2}$$

$$f = f_i^1 \oplus \bar{x_i} f_i^2 \quad \textit{negative Davio} \tag{3}$$

Furthermore, these are the only possible single-variable decompositions which can lead to the unique representation of the functions, up to negation [16]. Single-variable decompositions refer to all the decompositions to subfunctions f^i which totally remove a single variable from both subfunctions. It has to be mentioned that the uniqueness of the representation is only under the condition that all negations are transformed as described later on by complemented edges.

Now, the ordered Kronecker Functional Decision Diagrams can formally be defined as follows:

Definition 4 An *OKFDD over* X_n is given by an ordered DD over X_n together with a uniquely determined decomposition type, $d_i \in \{$Shannon, positive Davio, negative Davio$\}$, assigned to each variable x_i ($i \in \{1, .., n\}$). The function $f_G : \mathbf{B}^n \rightarrow \mathbf{B}$ represented by an OKFDD, G, over X_n is defined as:

1. If G consists of a single node labeled with 0 (1), then G is an OKFDD for $f = 0$ ($f = 1$).
2. If G has a root v with label x_i, then G is an OKFDD for

$$\begin{cases} \overline{x}_i f_{low(v)} + x_i f_{high(v)} & : \quad d_i \text{ is Shannon} \\ f_{low(v)} \oplus x_i f_{high(v)} & : \quad d_i \text{ is positive Davio} \\ f_{low(v)} \oplus \overline{x}_i f_{high(v)} & : \quad d_i \text{ is negative Davio} \end{cases}$$

where $f_{low(v)}$ ($f_{high(v)}$) are the functions represented by the OKFDD rooted at $low(v)$ ($high(v)$).

If at every node in above definition only Shannon decomposition is applied, the OKFDD will be an OBDD. If only Davio decompositions are applied, the OKFDD will be an OFDD. As it is evident, the OKFDD is the more general decision diagram than both OBDD and OFDD.

Definition 5 A node in an OKFDD is called a *Shannon-node* if it is expanded by Shannon decomposition - Equation (1). It is called a *Davio-node* if it is expanded by Davio decompositions - Equations (2) or (3); the latter being a *negative Davio-node* and the former a *positive Davio-node.*

Utilizing reductions, it is possible to define canonical representations of functions. The combination of reduction types I and S is well-known for OBDDs. The function f_G represented by a DD G is well defined for OBDDs in the sense that $f_{G_1} = f_{G_2}$ iff G_1 and G_2 are equivalent OBDDs. The same is also true for OFDDs with positive Davio nodes if reduction types I and D are used [1]. Using Lemma 1 canonicity can also be proven for OKFDDs:

Theorem 1 Reduced OKFDDs are canonical representations for Boolean functions if the decomposition types are fixed for every variable.

OBDDs and OFDDs are special cases of OKFDDs for which Shannon decomposition and Davio decompositions are used for all splitting variables, respectively. In the case of OKFDDs in general, each variable can be split by any of the three decompositions given in equations (1), (2), and (3). The advantage of using OKFDDs over just OBDDs or OFDDs is for example that there are classes of functions for which OBDDs are exponential in size while OFDDs with only positive Davio-nodes are polynomial and vice versa [2]. Using the OKFDDs, it is possible to achieve a reduced size DD which is not restricted by the type of decomposition.

The OKFDDs can further be reduced in size by using complemented edges. The constraints of complemented edges to maintain a canonical form for BDDs were given in [3]. Similarly, complemented edges can be used for the representation of a function and its complement by the same node in the case of Davio-nodes [1].

3 Implementation

In this section, implementational details of the OKFDD package are described and OKFDD manipulation algorithms are introduced.

3.1 Technical Details

First, programming techniques and methods of implementation used to speed-up the package are described. The methods are similar to other packages used for representation and manipulation of OBDDs and OFDDs [3, 12, 1]. Hence, these techniques are only briefly reviewed.

For the fast availability of the functions, a hash-based unique table is used to store the nodes. A computed table is implemented for the optimization of the synthesis algorithms. Furthermore, level lists are used for the management of the nodes of each stage.

In this way, fast access to the nodes is possible by the algorithms and efficient local transformations can be performed. The memory management is done by garbage collection. The nodes are only deleted if the storage place is needed for other nodes. Thus, it would not be needed to recompute the results each time if they were used earlier on. By the unique table, different OKFDDs can share the same sub-OKFDDs. Therefore, several functions can efficiently be represented at the same time.

3.2 Operations on and Construction of OKFDDs

In the following, the algorithms for OKFDDs with fixed variable ordering and a list of decomposition rules are given.

First, the XOR-operation is presented as it provides the basis for construction of certain other operations. Notice that for two functions, f and g one has:

$$\begin{aligned} f \oplus g &= (f_0 \oplus g_0) \oplus x_i (f_2 \oplus g_2) \\ f \oplus g &= (f_1 \oplus g_1) \oplus \overline{x}_i (f_2 \oplus g_2) \\ f \oplus g &= \overline{x}_i (f_0 \oplus g_0) + x_i (f_1 \oplus g_1) \end{aligned}$$

These equations make it possible to recursively split up the Davio- and Shannon-nodes into their left and right subgraphs and perform XOR-operations on the subgraphs. The resulting algorithm for XOR-operation on two OKFDDs, *kfdd_xor_kfdd ()*, is a generalization of the XOR-operation on two OFDDs, presented in [1].

The efficient XOR-operation allows the construction of OKFDDs from OBDDs. Here, one starts with a recursive computation in the OBDD. At each Shannon-node, v (labeled x_i), which is to be transformed into a positive Davio-node, the Davio-node, v', corresponding to the function represented by v is constructed. The successor $low(v)$ can be directly used for $low(v')$ since it represents the cofactor with respect to $x_i = 0$. For the case of a negative Davio-node, $high(v)$ needs to be used. For the successor $high(v')$, the XOR-operation has to be performed on the successors of v. Although the operations for each node can be performed efficiently, the algorithm has exponential worst case behavior if it is iterated for all nodes in an OBDD [2]. On the other hand, it can be shown that the complexity remains polynomial, as long as only a constant number of levels are transformed.

An algorithm to transform OKFDDs to OKFDDs with an other choice of decomposition rules follows directly from the algorithm described above with some slight modifications. Here, the recursive call of XOR-operation on OBDDs, performed by *if-then-else* operation [3], has to be substituted with the procedure for the general case *kfdd_xor_kfdd ()*. Additionally, different cases for the availability of the successors have to be distinguished. For instance, if a negative Davio-node must be transfered to a Shannon-node, the function f^0 must first be computed.

The realization of the AND-operation turns out to be more complicated for Davio-nodes in comparison to the XOR-operation. The following recursive equation holds for positive Davio-nodes:

$$\begin{aligned} f \cdot g &= (f_0 \oplus x_i f_2) \cdot (g_0 \oplus x_i g_2) \\ &= (f_0 \cdot g_0) \oplus x_i ((f_2 \cdot g_2) \oplus (f_0 \cdot g_2) \oplus (g_0 \cdot f_2)) \end{aligned}$$

This equation again defines a recursive algorithm which has exponential worst case running time [2]. The same results hold for negative Davio-nodes. However, for OKFDDs with a constant number of levels, where the Davio expansion is performed, it can be shown that the operation is polynomial, since in these cases efficient synthesis operations on Shannon nodes can be carried out in the rest of the graph.

The negation of a function, f, can be computed by observing that $\overline{f} = 1 \oplus f$. Thus, the operation requires an

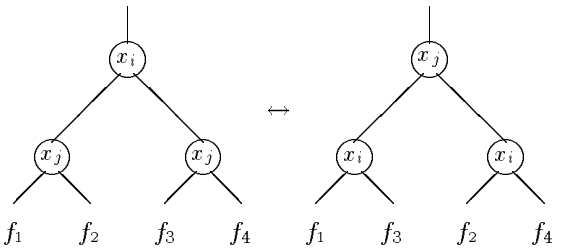

Figure 2: Exchange of i-th and adjacent variable

XOR-operation with the constant 1. Since the package uses complemented edges, negation can be performed even more efficiently, simply by setting a complement edge.

Now, using the algorithms for the XOR-, AND-, and NOT-operations, any binary operation can be realized.

For an OKFDD, G, the restriction $G|_{x_i=c}$ for variable x_i and constant c can be computed by traversing the graph and performing the corresponding substitutions. The case for Shannon-nodes is given by [3]. For the case of positive Davio-nodes, if $x_i = 0$, edges from nodes v with label x_i to $high(v)$ have to be deleted. If nodes with indegree 0 result, these and their outgoing edges are also deleted. Clearly, all this can be done in linear time. If $x_i = 1$, then at each node, v, with label x_i and subfunctions g_0 and g_1, the following has to be done. As before, the *high*-edge has to be deleted, at the *low*-edge an OKFDD for $g_0 \oplus g_1$ must be rooted, i.e., an XOR-operation has to be executed. For negative Davio-nodes, a similar procedure is required.

4 Optimization of OKFDD-Size

While the variable ordering plays a dominant role in the identification of the minimal OBDD representation of the functions, in OKFDDs both the ordering and the decomposition type are important. Depending on the order of the variables and the particular decomposition among the possible three, the size of the OKFDD can vary from linear to exponential [2]. It is well-known that in the case of OFDDs and OBDDs, the size of the decision diagram can be minimized by exchange of adjacent variables [7, 1]. It can be proven that this idea can be extended to OKFDDs. Therefore, it is also possible to use all techniques based on exchanging of adjacent variables for OKFDDs. Especially the sifting algorithm, window permutation, and exact minimization algorithms [13, 9] can be used. The general case for the exchange of variable, i, and an adjacent variable, j, is shown in Figure 2. Notice that the exchange pattern is independent of the decomposition type of the nodes. The exchange is performed very quickly since only edges must be redirected. In this approach complemented edges are also used.

By the sifting algorithm, the variables are sorted into decreasing order based on the number of nodes at each level and then each variable is traversed through the DAG in order to locate its locally optimal position while all other variables remain fixed.

The dynamic variable ordering based on the sifting algorithm can be utilized in the minimization of OKFDD sizes as well. In this scheme, the sifting algorithm is modified so that at each position all three types of decompositions are tested and the local optimum is chosen based on both the position and the type of the decomposition of the variable. Thus, each time a variable changes the decomposition rule, the new locally optimal position is determined by exchange of adjacent variables.

The minimization scheme can be summarized as follows: A heuristic or random order OBDD is constructed initially. If the size of the OBDD exceeds a chosen number of nodes, the reordering is applied. Here, each variable traverses through the levels, exchanging its level with its adjacent variable. With each exchange, an OKFDD is constructed with the chosen variable expanded with each of the Davio decompositions respectively and the optimal position of the variable is found by sifting. This is repeated for all levels and the locally optimal position and type of decomposition is designated. The procedure is repeated for a next variable, and so on. Thus, only one variable is changed in each step and from this point of view this heuristic is very simple.

Name	in	out	OBDD	OFDD	OKFDD
b1	3	4	6	5	5
c17	5	2	6	8	6
cm82a	5	3	11	9	9
majority	5	1	7	7	7
rd53	5	3	16	13	13
rd73	7	3	30	21	21
wim	4	7	19	22	17
Z5xp1	7	10	41	45	28

Table 1: Comparison of the number of nodes for optimal *OKFDD* vs optimal *OBDD* and *OFDD* with positive Davio nodes

5 Experimental Results

The experimental results confirm the advantage of OKFDDs over OBDDs and OFDDs and show the efficiency of the approach presented.

Herein the size of OKFDDs is compared with the size of OBDDs and OFDDs. First, the minimal size for arithmetical benchmark circuits [4] is considered. The benchmarks are minimized by an algorithm similar to the one presented in [9]. The results are given in Table 1. The optimum size can only be determined for small benchmarks due to the exponential running time of the minimization algorithm. For small functions there is only a minor improvement. But with increasing size of the functions the gains improve, see e.g. Z5xp1.

In a next series of experiments, larger benchmarks are considered for which the optimal ordering can not be determined. For this purpose, some benchmarks from [4] and MCNC91 are used.

For all types of decision diagrams (OBDDs, OFDDs and OKFDDs), dynamic variable ordering [13], starting from the original variable ordering was used. The results are presented in Table 2. In the last row the *total* sum of nodes for all the considered benchmarks is given. As it can be observed, for these functions, the average gain is 25% when the simple ordering heuristic from Section 4 is used.

Currently, incorporation of more sophisticated schemes such as changing several decompositions in parallel are being investigated. From the available results it can be inferred that the use of OKFDDs can potentially have drastic

Name	in	out	OBDD	OFDD	OKFDD
add6	12	7	68	45	44
apex7	48	37	296	360	266
bc0	26	11	535	727	431
chkn	29	7	322	459	279
cps	24	109	1040	1293	766
f51m	8	8	38	35	25
intb	15	7	656	624	480
mlp4	8	8	134	107	106
radd	8	5	33	20	19
s1423	91	79	2691	3821	1791
tial	14	8	617	600	479
ts10	22	16	193	155	155
total			6623	8246	4841

Table 2: Comparison of the number of nodes for *OKFDD* vs *OBDD* and *OFDD*

influence on realizations for which efficient OBDDs do not exist.

6 Conclusions

In this paper, the efficiency of ordered Kronecker Functional Decision Diagrams as a more compact decision diagram than OBDDs or OFDDs was shown. OKFDDs as a generalization of OBDDs and OFDDs will be always more compact than the two and this was confirmed through experimental results. For small functions, this was shown by comparison of optimal OBDDs, OFDDs, and OKFDDs. For larger benchmarks, an average 25% improvement in size over OBDDs was achieved.

Furthermore, a package for efficient representation and manipulation of Boolean functions by this data structure was introduced. Efficient algorithms were presented which make fast construction and manipulation of OKFDDs possible. The construction algorithm uses the direct correspondence between OFDDs and OBDDs.

The canonicity of the OKFDDs and efficient construction and manipulation techniques presented here make OKFDDs a prime candidate for utilization in applications where OBDDs have been the main construct. Applications in synthesis and verification as well as technology mapping to various FPGA architectures are among those that OKFDDs can be utilized.

The presented results provide the incentive to further investigate the efficient construction and application of OKFDDs. It is the focus of current research to develop more sophisticated ordering and decomposition heuristics. With the help of these heuristics, even more compact representations - also for functions with prohibitively large OBDD and OFDD sizes - should be constructible in reasonable time.

References

[1] B. Becker, R. Drechsler, and M. Theobald, On the Implementation of a Package for Efficient Representation and Manipulation of Functional Decision Diagrams, *IFIP WG 10.5 Workshop on Appl. of Reed-Muller Expans. in Circ. Design*, 1993.

[2] B. Becker, R. Drechsler, and R. Werchner, On the Relation Between BDDs and FDDs, Technical report, University of Frankfurt, 12/93, 1993.

[3] K. S. Brace, R. L. Rudell, and R. E. Bryant, Efficient Implementation of a BDD Package, *Proc. 27th Design Automation Conf.*, pages 40–45, 1990.

[4] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, 1984.

[5] R. E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. on Comp.*, C35, pages 677–691, 1986.

[6] R. E. Bryant, Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams, *ACM Comp. Surveys*, Vol. 24, pages 293–318, 1992.

[7] M. Fujita, Y. Matsunga, and T. Kakuda, On Variable Ordering of Binary Decision Diagrams for the Application of Multi-Level Synthesis, *European Conf. on Design Automation*, pages 50–54, 1991.

[8] J. Gergov and C. Meinel, Efficient Analysis and Manipulation of OBDDs Can be Extended to Read-Once-Only Branching Programs, to be published in *IEEE Trans. on Comp.*, 1994.

[9] N. Ishiura, H. Sawada, and S. Yajima, Minimization of Binary Decision Diagrams Based on Exchange of Variables, *Proc. Int. Conf. on Computer-Aided Design*, pages 472–475, 1991.

[10] U. Kebschull, E. Schubert, and W. Rosenstiel, Multilevel Logic Based on Functional Decision Diagrams, *Proc. European Design Automation Conf.*, pages 43–47, 1992.

[11] S. Malik, A. R. Wang, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment, *Proc. Int. Conf. on Computer-Aided Design*, pages 6–9, 1988.

[12] S. Minato, N. Ishiura, and S. Yajima, Shared Binary Decision Diagrams with Attributed Edges for Efficient Boolean Function Manipulation, *Proc. 27th Design Automation Conf.*, pages 52–57, 1990.

[13] R. L. Rudell, Dynamic Variable Ordering for Ordered Binary Decision Diagrams, *Proc. Int. Conf. on Computer-Aided Design*, pages 42–47, 1993.

[14] A. Sarabi and M. A. Perkowski, Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/XOR Canonical Networks, *Proc. 29th Design Automation Conf.*, pages 30–35, 1992.

[15] A. Sarabi, P. F. Ho, K. Iravani, W. R. Daasch, and M. A. Perkowski, Minimal Multi-Level Realization of Switching Functions Based on Kronecker Functional Decision Diagrams, *Int. Workshop on Logic Synth.*, pages P3a:1–6, 1993.

[16] T. Sasao. *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993.

[17] T. Sasao and Ph. Besslich, On the Complexity of Mod-2 Sum PLAs, *IEEE Trans. on Comp.*, Vol. 39, No. 2, pages 262–266, 1990.