7

Symmetry Based Variable Ordering for ROBDDs

Dirk Möller, Paul Molitor*, Rolf Drechsler***
**Martin-Luther-Universität Halle*
Institut für Informatik, Fachbereich Mathematik und Informatik,
Martin-Luther-Universität Halle, 06099 Halle, Germany.
Tel: +49-345-622522. Fax: +49-345-622514.
email: {moeller,molitor}@informatik.uni-halle.de

***Johann Wolfgang Goethe-Universität Frankfurt*
Fachbereich 20 - Informatik, J. W. Goethe-Universität,
Robert-Mayer-Straße 11-15, 60054 Frankfurt am Main, Germany.
Tel: +49-69-79828472. Fax: +49-69-79828250.
email: drechsle@kea.informatik.uni-frankfurt.de

Abstract

Reduced Ordered Binary Decision Diagrams (ROBDDs) are a data structure frequently used for representation and manipulation of Boolean functions. Since the size of ROBDDs is extremely sensitive to the variable order a lot of heuristics to get a good variable order have been developed. For the class of partially symmetric Boolean functions this paper presents a new general method to improve quality of ordering heuristics based on the exchange of variables. Statistical and benchmark results are given to show the efficiency of our approach.

Keywords

variable ordering, symmetric Boolean functions

*Supported in part by DFG grant SFB 124/B6
**Supported in part by DFG grant Be 1176/4-2

G. Saucier et al. (eds.), *Logic and Architecture Synthesis*

1 INTRODUCTION

Binary Decision Diagrams (BDDs) as a data structure for representation of Boolean functions were first introduced by Lee (1959) and further popularized by Akers (1978) and Moret (1982) . In the restricted form of ROBDDs they gained widespread application, because ROBDDs are a canonical representation and allow efficient manipulations (Bryant 1986) . Some fields of application are logic design verification, test generation, fault simulation, and logic synthesis (Malik 1988, Bryant 1992). Most of the algorithms using ROBDDs have running time polynomial in the size of the ROBDDs. The sizes themselves depend on the variable order used. Thus, there is a need to find a variable order that minimizes the number of nodes in an ROBDD.

As an example of the application of ROBDDs consider the use of Field Programmable Gate Arrays (FPGA) in the construction of Combinational Logic Circuits (CLC). A BDD has a direct correspondence to a CLC when each node of the BDD is substituted by a multiplexer. Since it is straightforward to map these multiplexer circuits on a FPGA whose logic blocks are based on multiplexers, BDDs have become a good framework for logic synthesis. The usage of BDDs also simplifies the technology mapping and the routing for FPGAs (Brown 1992). Here it is already useful to save only a few nodes using a good order. Additionally, these circuits have nice testability properties (Becker 1992).

The existing methods for finding good variable orders can be classified into three categories. The first are initial heuristics starting from a CLC (Malik 1988, Fujita 1988, Fujii 1993), the second are gradual improvement heuristics based on the exchange of variables in the ROBDD (Ishiura 1991, Fujita 1991, Rudell 1993, Felt 1993), and the third are exhaustive methods to find a mininum order (Friedman 1990). The gradual improvement heuristics play an important role in dynamic variable reordering, since these methods are able to handle large circuits that cannot be represented without dynamic methods (Rudell 1993).

In this paper we consider partially symmetric functions, i.e., functions that are invariant under the permutation of some input variables. Given its ROBDD representation the symmetries of a Boolean function can be determined efficiently (Möller 1993). Knowing a Boolean function to be symmetric allows the application of special logic synthesis tools that can improve the results of the design (Hurst 1977, Kim 1991, Drechsler 1995). Furthermore, knowing the variables of a function which are symmetric often restricts the search space of a logic design problem what may yield in a remarkable decrease of running time for that problem. Such problems are, e.g. the equivalence test of Boolean functions for that the input correspondence is not known (Cheng 1993, Lai 1992, Mohnke 1993) and technology mapping (Mailhot 1990). In this paper, we especially show that symmetry properties can be efficiently used to construct good variable orders for ROBDDs using modified gradual improvement heuristics.

There are three facts that create the basis of our ordering improvements:

1. The exchange of two symmetric variables does not change the size of the ROBDD, because the function remains the same.
2. The size of the ROBDD of any totally symmetric function $f : \{0,1\}^n \rightarrow \{0,1\}$ is $O(n^2)$ (Wegener 1984, Bryant 1986).

3. The value of a function which is symmetric in some variables $\{x_{i_1}, \ldots, x_{i_q}\}$ does not depend on the exact assignment of these variables but only on their weight $\sum_{j=1}^{q} x_{i_j}$.

Using the first fact, the ordering heuristics can skip over the exchange of symmetric variables and so the running time decreases. However, the resulting ROBDD sizes will be the same. The second and third fact leads to a special class of variable orders. In those orders the symmetric variables are located side by side. This class will be introduced and investigated in this paper. Especially, our paper gives a negativ answer to the conjecture stated in (Panda 1994)* that for each Boolean function there is at least one order where the symmetric variables are located side by side, for which the ROBDD is of minimal size. We will present counterexamples to this conjecture.

Nevertheless, in general, it is reasonable to locate the symmetric variables side by side: To give an impressive example, consider the function $x_1x_{n+1} + \ldots + x_nx_{2n}$ (Bryant 1986). The size of the corresponding ROBDD with variable ordering $x_1, x_2, x_3, \ldots, x_{2n}$ is exponential in n whereas the size of any ROBDD with an order where the symmetric variables are side by side is linear in n.

We present statistical facts for all partially symmetric Boolean functions with up to five input variables and experimental results of functions taken from the LGSYNTH91 benchmark set proving the new class of orders to be very efficient with respect to the ROBDD size. This leads to the idea of modifying the reordering heuristics presented in literature so that they do not reorder single variables but whole symmetric blocks. The benchmark results show that the modified algorithms outperform the original ones.

The paper is structured as follows: In Section 2 ROBDDs and symmetric functions are defined. Symmetry variable orders are introduced in Section 3 and statistical results are presented in Section 4. In Section 5 benchmark results are given. We finish with a resume of the results in Section 6.

2 NOTATIONS AND DEFINITIONS

We provide a short introduction to basic notions which are important for the understanding of this paper.

2.1 Binary Decision Diagrams

We start with a brief review of the essential definitions and properties of BDDs (see Bryant (1986)).

Definition 1 A *Binary Decision Diagram (*BDD*)* 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 has as label an argument index $index(v) \in \{1, \ldots, n\}$ and two children $low(v), high(v) \in V$. A terminal vertex v is labeled with a value $value(v) \in \{0, 1\}$ and has no outgoing edge.

*Similar methods as those presented in this paper have independently be developed in that paper.

BDDs are also called *branching programs* (Wegener 1987). A BDD can be used to compute a Boolean function $f(x_1, \ldots, x_n)$ in the following way: Each input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ defines a computation path through the BDD that starts at the root. If the path reaches a non-terminal node v that is labeled by i it follows the path $low(v)$ iff $x_i = 0$ and it follows the path $high(v)$ iff $x_i = 1$. On all paths a terminal vertex is reached since a BDD is directed and acyclic. The label of the terminal vertex determines the return value of the BDD on input x.

More formally, we can define the Boolean function corresponding to a BDD, recursively.

Definition 2 A BDD having root vertex v *denotes* a Boolean function f_v defined as:

1. If v is a terminal vertex and $value(v) = 1$ ($value(v) = 0$), then $f_v = 1$ ($f_v = 0$).
2. If v is a non-terminal vertex and $index(v) = i$, then f_v is the function

$$f_v(x_1, \ldots, x_n) = \overline{x}_i \cdot f_{low(v)}(x_1, \ldots, x_n) + x_i \cdot f_{high(v)}(x_1, \ldots, x_n).$$

The variable x_i is called the *decision variable* for v.

It is well-known that for each Boolean function f there exists a BDD denoting f.

BDDs are often used as a data structure in design automation and logic synthesis. Thus there is a need of efficient manipulation of BDDs. Unfortunately, this property is not fulfilled by the general BDDs defined above (see Gergov (1992)). Therefore we need further restrictions on the structure of the BDDs.

Definition 3 A *Reduced Ordered* BDD *(*ROBDD*)* is a BDD with the following two properties:

1. The BDD is ordered, i.e. for any non-terminal vertex v, if $low(v)$ ($high(v)$) is also a non-terminal vertex, then $index(v) < index(low(v))$ ($index(v) < index(high(v))$).
2. The BDD is reduced, i.e. there exists no $v \in V$ with $low(v) = high(v)$ and there are no two vertices v and v' with identical labels such that the sub-BDDs rooted by v and v' are isomorphic.

Functions denoted by ROBDDs can be manipulated efficiently (Bryant 1986). For our practical experiments we use a ROBDD package with complemented edges as described in (Brace 1990). If not other mentioned, all results are given for this kind of ROBDDs.

We associate with each ROBDD an array π such that $\pi[i]$ denotes the variable that corresponds to label i. The array π is called the *variable order* of the ROBDD. Using this notation, the function f_v at the non-terminal node v with label i is

$$f_v(x_1, \ldots, x_n) = \overline{\pi[i]} \cdot f_{low(v)}(x_1, \ldots, x_n) + \pi[i] \cdot f_{high(v)}(x_1, \ldots, x_n).$$

2.2 Symmetric functions

Let $f : \{0,1\}^n \to \{0,1\}$ be a completely specified Boolean function and $\mathcal{V}_n = \{x_1, \ldots, x_n\}$ be the corresponding set of variables. The function f is said to be *symmetric* with respect

to a set $\lambda \subseteq \mathcal{V}_n$ if f remains invariant under all permutations of the variables in λ. For completely specified functions the symmetry is an equivalence relation which partitions the set $\mathcal{V}_n$ into disjoint classes $\lambda_1, \ldots, \lambda_k$ that will be named the *symmetry sets*. A function f is called *partially symmetric* if it has at least one symmetry set λ_i with $|\lambda_i| > 1$. If a function f has only one symmetry set $\lambda = \mathcal{V}_n$, then f is called *totally symmetric*. For totally symmetric functions all variable orders generate ROBDDs of the same size. Thus, no reordering is necessary and we will focus on partially symmetric functions only.

Sometimes, it may be occur that $\{x_i, x_j\}$ is not a symmetric pair but $\{x_i, \overline{x}_j\}$ is one. This kind of symmetry was introduced by Hurst (1977) and named *equivalence symmetry*. In such a case, we change the phase of variable x_j in the ROBDD and get symmetry in $\{x_i, x_j\}$. Note that the negation of a variable does not change the size of an ROBDD, because it exchanges the successors of the nodes with label x_j only.

3 SYMMETRY VARIABLE ORDERS

In this section, we introduce the new class of symmetry variable orders that we will use to improve the existing reordering heuristics.

Definition 4 Let f be a partially symmetric function with the set of symmetry sets $S = \{\lambda_1, \ldots, \lambda_k\}$. A variable order π is called a *symmetry variable order* if for each symmetry set $\lambda_i \in S$ there exists j so that $\{\pi[j], \pi[j+1], \ldots, \pi[j+|\lambda_i|-1]\} = \lambda_i$.

By this definition, the class of symmetry variable orders consists of all variable orders where the variables of each symmetry set are located side by side. The ROBDDs that correspond to symmetry orders are called *symmetry ordered* ROBDDs. In the remainder of this section the efficiency of symmetry orders will be motivated.

It is well known that the ROBDD size of any totally symmetric function f is $O(n^2)$. In a symmetry ordered ROBDD there exists a lot of sub-ROBDDs where all variables in the upper part form a symmetry set. Thus, the upper part of these sub-ROBDD forms a totally symmetric function and the size of this part is $O(k^2)$ if k is the size of the symmetry set. Thus, symmetry ordered ROBDDs contain a number of small $O(k^2)$-sized sub-ROBDDs.

Furthermore, the value of a function that is symmetric in some variables $\{x_{i_1}, \ldots, x_{i_q}\}$ does not depend on the exact assignment of these variables but only on their weight $\sum_{j=1}^{q} x_{i_j}$. If one uses symmetry ordered ROBDDs, this weight is computed in neighbouring levels and no information about partial weights has to be kept over several non-symmetric levels – and keeping information may cause large ROBDD sizes. Symmetry variable orders avoid this drawback.

4 STATISTICAL RESULTS

Due to the remarks above, it seems that symmetry orders generate smaller ROBDDs than general orders. To check this assumption, we investigated all partially symmetric functions with three, four and five inputs. For each function we determined the number of general orders and and the number of symmetry orders that create an ROBDD with $x\%$ more

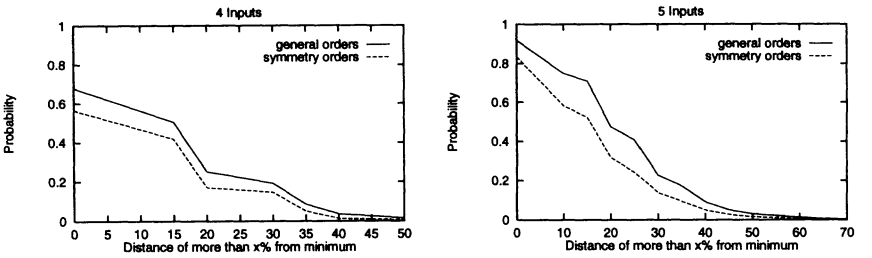

Figure 1 Distribution of general orders and symmetry orders

nodes than the minimum ROBDD. Using these data, we computed the probability to get an ROBDD with more than $x\%$ more nodes than the minimum for an arbitrary function and an arbitrary order. Figre 1 shows the result obtained for the four and five input functions. The dashed line shows the probability that the ROBDD for an arbitrary partially symmetric function with an arbitrary *symmetry* order has more than $x\%$ additional nodes with respect to the minimum. The solid line shows the same for *general* orders. It turns out that the probability to get a $x\%$ oversized ROBDD with a symmetry order is always smaller than it is for general orders. This shows from a statistical point of view that the symmetry orders constitute an efficient subclass of variable orders. For completeness, we computed

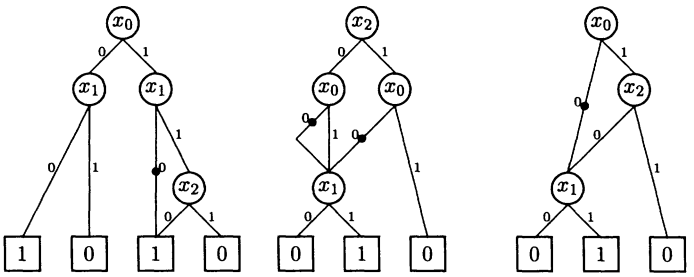

Figure 2 ROBDDs of $f = \overline{x}_0\overline{x}_1 + x_o x_1 \overline{x}_2$

the number of partially symmetric functions for that each symmetry order results in a non-minimal ROBDD. For the 120 partially symmetric functions with three inputs there are 24 (20%) such functions. For example, the $\{x_0, x_1\}$ symmetric function shown in Figure 2 has symmetry ordered ROBDDs with 4 internal nodes while the minimum ROBDD has 3 internal nodes. For the 20.548 partially symmetric functions with four inputs there are 960 (4.7%) and for the 162.535.140 partially symmetric functions with five inputs there are only 972.280 (0.6%) such functions. The distance of the best symmetry order to the minimum was at most two nodes. We confirmed our results by performing experiments with some functions with more than five inputs. Thus, considering only symmetry variable orders seems to result in ROBDDs with sizes not larger than the minimum in almost all cases.

Table 1 Initial ordering with symmetry orders

Heuristic	ROBDD size			nodes		Heuristic	ROBDD size			nodes	
	<	=	>				<	=	>		
initial				58688		initial_so				49199	
first	9	5	3	58432	1.3%	first_so	312	236	101	47275	1,2%
median	14	3	0	58020	1.5%	median_so	398	174	77	46353	1.5%
last	14	3	0	58007	1.8%	last_so	409	161	79	46362	1.6%
best	15	2	0	57888	2.5%	best_so	534	112	3	45252	2.4%

In (Panda 1994) it was conjectured that for ROBDDs *without* complemented edges for each functions one of its symmetry orders results in an ROBDD of minimal size. A negative answer to this conjecture was given by our experiments with these kind of ROBDDs. The four input function $f = \overline{x}_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + \overline{x}_1\overline{x}_2\overline{x}_3x_4$ which is symmetric in $\{x_1, x_2, x_3\}$ has best symmetry order of ROBDD size 9 and the minimum size is 8. For ROBDDs without complemented edges there are 80 partially symmetric functions with four inputs and 1.262.800 functions with five inputs without a minimum symmetry order.

5 BENCHMARK RESULTS

In this section we show the efficiency of the symmetry variable orders in practical application. For our experiments we used the CMU-BDD package contained in SIS-1.2 (Sentovich 1992) and processed 109 combinational two-level and multi-level circuits from the LGSYNTH91 benchmark set. Since the single primary outputs of a multiple output function sometimes have more symmetry we also processed each primary output of each circuit separately, to get more results about symmetric functions. Symmetry detection was executed on the ROBDDs using the algorithm proposed in (Möller 1993). We modified this algorithm to detect equivalence symmetry as well. This results in about 10 % more symmetry.

If an ROBDD is to be created from a circuit description, a heuristic, e.g. (Malik 1988), generates good initial order which is not necessarily a symmetry order. As discussed above, the size of the ROBDD may be reduced, if the initial non-symmetry order is transformed into a symmetry one. We have applied three algorithms to get a symmetry order. They differ only in the way they select the new position for a symmetry set. Heuristic first selects as position for a symmetry set the position of the first variable of the symmetry set, median selects the position of the middle variable and last selects the position of the last symmetric variable. Heuristic best calls all three methods and then selects the best order. The suffix _so denotes the methods that handle each primary output seperately. The results obtained by initial reordering are shown in Table 1. The first column gives the name of the reordering heuristic. The second, third and fourth column shows the total number of benchmark functions where the size of the symmetry ordered ROBDD is smaller, equalsized, or greater than the initial one when it was reordered with the corresponding

Table 2 Reordering with symmetry orders

Heuristic	ROBDD size			nodes		time (sec)
	<	=	>			
win3				66350		14
Swin3	25	29	2	64200	5.7%	16
sift				33878		92
Ssift	26	26	4	33149	7.1%	93
win3_so				67961		36
Swin3_so	693	1443	42	63668	3.4%	41
sift_so				58177		116
Ssift_so	452	1695	4	54970	2.6%	99

heuristic. The last column shows the total number of nodes of all ROBDDs and the average improvement over all benchmarks.

For the 109 multiple output functions we detected 56 to be partially symmetric. The initial ordering heuristic already generates a symmetry order for 39 of these functions. For more than the half of the remaining non-symmetry ordered ROBDDs the order has been improved by each of the three symmetry reordering methods and the overall number of nodes decreases. The best heuristic seems to be last and we select it for our next experiments. However, row best shows that the heuristics work well on different functions. There are only three of the single output functions for which all three heuristics generate a symmetry ordered ROBDD that is greater than the initial one. This shows that symmetry orders are also good in practice.

To reduce the size of an ROBDD several reordering heuristics have been developed. Two of them, win3 and sift (Rudell 1993) are implemented in the CMU-BDD package. To work with symmetry orders we make use of the variable blocking feature of the CMU-BDD package. Before starting reordering, we block the symmetric variables which were made adjacent by last. The modified heuristics are called Swin3 and Ssift, respectively. For all symmetric functions from the benchmark set the original heuristics win3 and sift and the modified heuristics Swin3 and Ssift were applied to the initial ROBDDs. Results are presented in Table 2. The first column denotes the reordering heuristic. The second, third and fourth column shows the total number of benchmark functions for that the modified heuristics generate a smaller, equalsized, or greater ROBDD than the original heuristic. Column nodes shows the number of nodes of all the optimized ROBDDs and the average improvement over all benchmarks. Column time shows the running time† of the heuristics. The additional overall running time for symmetry detection for multiple-output and single-output functions is about 88 seconds.

†All running times are seconds at Sparcstation 10/64 Mb

Table 3 Benchmark results of reordering with symmetry orders

circuit	symsets			init	Sinit	win3	Swin3	sift	Ssift
apex2		1(3)	3(2)	2947	2846	910	634	700	654
cps			1(4)	1455	1445	1301	1294	1035	991
ex4			14(2)	895	822	692	691	537	539
seq			2(2)	5638	5532	3737	2586		
t481			8(2)	63	33	33	21	33	31
vg2			2(2)	390	385			132	146
comp			16(2)			146	128	146	107
count			1(2)			232	201	201	82
dalu			1(2)	4575	4346			1322	1323
frg2			1(2)					2297	2171
i2	2(64)	3(16)	3(4)	1586	1583	795	298		
i4		16(3)	50(2)	349	333	333	245	308	233
lal			5(2)	122	110	97	95	75	72
my_adder		1(3)	15(2)			457	452	457	411
pcler8_cl			1(2)			138	122	130	86
rot		2(3)	2(2)	10224	10222	8212	8204	4574	4568
too_large		1(3)	3(2)			667	676	500	439
x1			1(2)	1211	1190	784	799	544	518
z4ml		1(3)	2(2)	37	30	21	17	24	17

It is shown that the heuristics that use symmetry orders generate better or same results in most cases. Swin3 saves 5.7% nodes and Ssift3 saves 7.1% nodes in average. The running time for symmetric reordering remains nearly the same. Unfortunately, there is the extra running time for symmetry detection. This increases the running time of sift in general by factor 2 and of win3 up to factor 7. One can overcome this difficulty if the symmetry detection is integrated in the reordering method following the *Idea 3* of neighbouring symmetry in (Möller 1993) as presented in (Panda 1994). Table 3 shows the effect of symmetry based reordering for some individual benchmarks. In column symsets the symmetry is given. 2(3) means that there are wo symmetry sets of three input variables. The following columns show the ROBDD size achieved by the mentioned heuristics. The leading S denotes the symmetric version. If the symmetric reordering results in the same size as the original the results are omitted.

It is shown that the symmetry modified algorithms in general outperform the original ones. Furthermore, even a small number of symmetry sets and variables can cause a great improvement. For example, for seq with only two symmetry sets of size two Swin3 saves about 30% of all nodes and for count with only one symmetry pair Ssift saves about 60%. Thus, symmetry based ordering is not only suitable for high symmetric functions but also for low symmetric functions.

6 CONCLUSIONS

We presented a new general method to improve the quality of ROBDD reordering heuristics for the class of partially symmetric functions by using symmetry variable orders. In this special type of orders the symmetric variables are located side by side.

Our statistical experiments as well as our benchmark experiments have shown that the efficient symmetry orders form a good subclass of all efficient variable orders. So it is useful to work only with symmetry orders when representing partially symmetric functions as ROBDDs.

It is also possible to collapse nodes of neighbouring levels corresponding to symmetric variables to *symmetry nodes*, i.e. nodes with more than two outgoing edges. This simplifies the procedure of exchanging symmetry sets. First steps in this direction can be found in (Becker 1994).

It is focus of current work to generalize the present appraoch to more powerful graph representations, like Ordered Kronecker Functional Decision Diagrams (OKFDDs) (Drechsler 1994).

REFERENCES

S.B. Akers (1978). Binary Decision Diagrams. *IEEE Trans. on CAD*, 27(6):509–516, 1978.

B. Becker (1992). *Synthesis for Testability: Binary Decision Diagrams*, volume 577 of *LNCS*. Symp. on Theoretical Aspects of Comp. Science, pages 501–512, 1992.

B. Becker and R. Drechsler (1994). Efficient Graph Based Representation of Multi-Valued Functions with an Application to Genetic Algorithms. In *IEEE 24th International Symposium on Multi-Valued Logic*, pages 65–72, 1994.

K.S. Brace, R.L. Rudell, and R.E. Bryant (1990). Efficient Implementation of a BDD Package. In *Design Automation Conf.*, pages 40–45, 1990.

R.J. Brown, S.D.and Francis, J. Rose, and Z.G. Vranesic (1992). *Field-Programmable Gate Arrays*. Kluwer Academic Publisher, 1992.

R.E. Bryant (1986). Graph–Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on CAD*, 35(8):677–691, 1986.

R.E. Bryant (1992). Symbolic Boolean Manipulation with Ordered Binary Decision diagrams. *ACM, Comp. Surveys*, 24:293–318, 1992.

D. I. Cheng and M. Marek Sadowska (1993). Verifying Equivalence of Functions with Unknown Input Correspondence. In *European Conf. on Design Automation*, pages

81–85, 1993.

R. Drechsler and B. Becker (1995). Sympathy: Fast Exact Minimization of Fixed Polarity Reed-Muller Expressions for Symmetric Functions. In *European Conf. on Design Automation*, pages 91–97, 1995.

R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski (1995). Efficient Representation and Manipulation of Switching Functions Based on Ordered Kronecker Functional Decision Diagrams. In *Design Automation Conf.*, pages 415–419, 1994.

E. Felt, G. York, Brayton R., and A. Sangiovanni-Vincentelli (1993). Dynamic Variable Reordering for BDD Minimization. In *European Conf. on Design Automation*, pages 130–135, 1993.

St.J. Friedman and K.J. Supowit (1990). Finding the Optimal Variable Ordering for Binary Decision Diagrams. *IEEE Trans. on CAD*, 39(5):710–713, 1990.

H. Fujii, G. Otomo, and C. Hori (1993). Interleaving Based Variable Ordering Methods for Ordered Binary Decision Diagrams. In *IEEE Int'l Conf. on CAD*, pages 38–41, 1993.

M. Fujita, H. Fujisawa, and N.Kawato (1988). Evaluation and Improvements of Boolean Comparison Methods Based on Binary Decision Diagrams. In *IEEE Int'l Conf. on CAD*, pages 2–5, 1988.

M. Fujita, Y. Matsunaga, and T. Kakuda (1991). On Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis. In *European Conf. on Design Automation*, pages 50–54, 1991.

J. Gergov and C. Meinel (1992). Analysis and Manipulation of Boolean Functions in Terms of Decision Graphs. In *WG'92, LNCS*, pages 310–320, 1992.

S.L. Hurst (1977). Detection of Symmetries in Combinatorial Functions by Spectral Means. *IEE Electronic Circuits and Systems*, 1(5):173–180, 1977.

N. Ishiura, H. Sawada, and S. Yajima (1991). Minimization of Binary Decision Diagrams Based on Exchanges of Variables. In *IEEE Int'l Conf. on CAD*, pages 472–475, 1991.

B.-G. Kim and D.L. Dietmeyer (1991). Multilevel Logic Synthesis of Symmetric Switching Functions. *IEEE Trans. on CAD*, 10(4):436–446, 1991.

Y.-T. Lai, S. Sastry, and M. Pedram (1992). Boolean Matching using Binary Decision Diagrams with Applications to Logic Synthesis and Verification. In *Int'l Conf. on Comp. Design*, pages 452–458, 1992.

C.Y. Lee (1959). Representation of Switching Circuits by Binary Decision Diagrams. *Bell System Technical Jour.*, 38:985–999, 1959.

F. Mailhot and G.D. Micheli (1990). Technology Mapping using Boolean Matching and Don't Care Sets. In *European Conf. on Design Automation*, pages 212–216, 1990.

S. Malik, A.R. Wang, R.K. Brayton, and A. Sangiovanni-Vincentelli (1988). Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment. In *IEEE Int'l Conf. on CAD*, pages 6–9, 1988.

J. Mohnke and S. Malik (1993). Permutation and Phase Independent Boolean Comparison. In *European Conf. on Design Automation*, pages 86–92, 1993.

D. Möller, J. Mohnke, and M. Weber (1993). Detection of Symmetry of Boolean Functions Represented by ROBDDs. In *IEEE Int'l Conf. on CAD*, pages 680–684, 1993.

B.M.E. Moret (1982). Decision Trees and Diagrams. *ACM, Comp. Surveys*, 14(4):593–623, 1982.

S. Panda, F. Somenzi, and B.F. Plessier (1994). Symmetry Detection and Dynamic Variable Ordering of Decision Diagrams. In *IEEE Int'l Conf. on CAD*, pages 628–631, 1994.

R. Rudell (1993). Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *IEEE Int'l Conf. on CAD*, pages 42–47, November 1993.

E. Sentovich, K. Singh, L. Lavagno, Ch. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli (1992). SIS: a system for sequential circuit synthesis, 1992. Department of EE and CS, UC Berkeley.

I. Wegener (1984). *Optimal Decision Trees and One-Time-Only Branching Programs for Symmetric Boolean Functions. Information and Control*, 62:129–143, 1984.

I. Wegener (1987). *The Complexity of Boolean Functions.* John Wiley & Sons Ltd., and B.G. Teubner, Stuttgart (Wiley–Teubner Series in Computer Science), 1987.

Dirk Möller received the diploma degree in computer science from the Humboldt-University Berlin, Germany, in 1992.
Between 1992 to 1994, he was with the Sonderforschungsbereich "VLSI Design Methods and Parallelism" at the Humboldt-University.
Since 1995 he is with the Computer Science Department at Martin-Luther-University Halle. His research interests are logic and physical synthesis.

Rolf Drechsler received the diploma degree in computer science from the J.W.Goethe-University Frankfurt, Germany, in 1992.
Since 1993 he is with the Computer Science Department at J.W.Goethe-University Frankfurt. His research interests are logic synthesis, testing and genetic algorithms.

Paul Molitor received the Diplom, Ph.D., and Habilitation degrees in 1982, 1986, and 1992, respectively, from the University of Saarland, Germany.
Between 1982 to 1992, he was with the Sonderforschungsbereich "VLSI Design Methods and Parallelism" at the University of Saarland.
After being visiting professor at the University of Halle (1992/93) and the University of Freiburg i.Br. (1993), he was an associate professor in the Computer Science Department of the Humboldt-University Berlin in 1993/94.
Since 1994, he is a full professor at the Martin-Luther University Halle, Germany, where he is the chairman of the Institute of Computer Science.
His research interests include logic and physical synthesis, hierarchical VLSI design, algorithms, and data structures.