Manipulation Algorithms for K*BMDs

Rolf Drechsler^{1*}, Bernd Becker¹, Stefan Ruppertz²

¹ Institute of Computer Science
Albert-Ludwigs-University
79110 Freiburg im Breisgau, Germany
² Department of Computer Science
Johann Wolfgang Goethe-University
60054 Frankfurt am Main, Germany

Abstract. *Bit-level and word-level based Decision Diagrams (DDs) have led to significant advances in the area of Computer Aided Design (CAD). Recently, a new data structure for the word-level, called Kronecker Multiplicative BMDs (K*BMDs), has been presented.*

*We study manipulation algorithms for K*BMDs: Using K*BMDs it is possible to represent functions efficiently, that have a good word-level description (like multipliers). On the the other hand K*BMDs are also applicable to verification problems at the bit-level. We clarify the relation between bit- and word-level representation which is of importance in particular in the context of verification. Experiments show that *BMDs are not well-suited for the bit-level. On the other hand OBDDs are not applicable on the word-level. We present algorithms that allow to dynamically switch between bit-level and word-level. We discuss a method for changing the decomposition type and variable order. First experiments demonstrate the efficiency of K*BMDs as a data structure that is suitable for bit-level and word-level functions as well, e.g. K*BMDs can efficiently represent all of the LGSynth91, ISCAS85, and ISCAS89 benchmarks.*

1 Introduction

One of the most important tasks during the construction and design of *Integrated Circuits* (ICs) is the proof of correctness, i.e. the check whether the design fulfills the specification. In the last few years several methods based on *Decision Diagrams* (DDs) have been proposed [26,21,9] and in the meantime have been used in industrial applications [1]. The most popular data structure in this area are *Ordered Binary Decision Diagrams* (OBDDs) [7]. OBDDs have in many cases shown to be a good compromise between compactness of representation and efficiency of the manipulation algorithms. In several applications it was for the first time possible to determine exact results for problems that have only been solved heuristically before (see e.g. [24]). Additionally, efficient implementations of DD packages are available [5,27,16]. The major drawback of OBDDs is that in some cases they cannot represent the functions efficiently; this also holds for functions with high practical relevance, like multipliers [8]. Consequently, several generalizations of OBDDs

* email: drechsle@informatik.uni-freiburg.de

have been proposed. One of them are *Ordered Kronecker Functional Decision Diagrams* (OKFDDs) [18], that have been proven to be the most general ordered DD for Boolean functions [4].

Alternatively, DDs for representation of integer-valued functions have been proposed. (The class of all these DDs is denoted by word-level DDs in the following.) The direct generalization of OBDDs leads to *Multi-Terminal BDDs* (MTBDDs) [12]. In the case of word-level DDs often no encoding has to be used for integer function representation. Additionally, these DD-types can make use of the fact that many functions that have no efficient bit-level representation can easily be expressed on the word-level, like this is the case for multipliers.

Recently, an other new data structure has been proposed, called *Multiplicative Binary Moment Diagrams* (*BMDs) [10,11] that operates on the word-level. Using *BMDs it was for the first time possible to verify multipliers of bit length up to $n = 256$. Unfortunately, *BMDs fail for the representation of Boolean functions that can easily be represented using OBDDs [20]. Thus, extensions of *BMDs are of interest.

First steps in this direction have been proposed in [15,14,17]. In [17] a method how to extend *BMDs has been presented, so that they can also handle Boolean functions efficiently. To do so, several integer-valued decomposition types (similar to OKFDDs [18,16]) are used, instead of only one. Additionally, edge values are allowed, as has been proposed in [25]. Using this generalization a data structure is obtained, called *Kronecker *BMD* (K*BMD), that is a generalization of *BMDs and OBDDs as well.

Analogously to OKFDDs, K*BMDs are very sensitive to the variable ordering and decomposition type choice. In this paper we discuss algorithms for (dynamic) minimization of K*BMDs. We clarify the relation between bit- and word-level verification. It is shown that *BMDs are not well-suited for the bit-level. On the other hand OBDDs are not applicable on the word-level. We give algorithms that allow to dynamically switch between bit-level and word-level representation e.g. during one call of a symbolic simulation procedure. Experimental results are given to demonstrate the usefulness of K*BMDs, even in the case, that the decision for bit- or word-level is made before the run of the algorithm. K*BMDs can efficiently represent all of the ISCAS85 and ISCAS89 benchmarks.

The paper is structured as follows: In Section 2 we briefly review the definitions of *BMDs, EVBDDs and K*BMDs. In Section 3 we show the need for the extension of *BMDs, since they are not well-suited for bit-level representation. We give a recursive algorithm to switch between bit- and word-level (and between word-levels of differing width) in Section 4. In Section 5 we consider the problem of finding good variable orderings and decomposition types. Experimental results are given in Section 6. Finally, the results are summarized.

2 The Data Structure

In this section we review the basic definitions of the data structure *Kronecker Multiplicative Binary Moment Diagrams* (K*BMDs) [17]¹. We first briefly review the basic notations of *BMDs and EVBDDs, since this simplifies the understanding of the paper. (For more details see [25,10,11].)

2.1 *BMDs and EVBDDs

We use *Multiplicative Binary Moment Diagrams* (*BMDs) as introduced in [10,11] and *Edge-Valued Binary Decision Diagrams* (EVBDDs) as introduced in [25]. Both data structures are graph-based representations, where at each (non leaf) node a decomposition of the function (represented by this node) into two subfunctions (the *low*-function and the *high*-function) is performed.

EVBDDs make use of the Shannon decomposition. Additionally, a constant a is added to the function being represented. Thus, an edge with weight a to a node v in the EVBDD represents the function $\langle a, f \rangle = a + (1-x)f_{low(v)} + xf_{high(v)}$. (f is the function represented at node v , $f_{low(v)}$ ($f_{high(v)}$) denotes the function represented by the *low*-edge (*high*-edge) of v .)

In *BMDs the values at the edges are multiplied with the functions represented. Thus, an edge with weight m to a node v in a *BMD represents the function $\langle m, f \rangle = m(f_{low(v)} + xf_{high(v)})$.

Using the notations from [18] we call the EVBDD decomposition the *Shannon decomposition* (S) and the one used by *BMDs the *positive Davio decomposition* (pD).

*Example 1. An EVBDD and *BMD for the unsigned integer encoding (4 bits)*

$$f_{enc}(x_3, x_2, x_1, x_0) := [x_3, x_2, x_1, x_0] := \sum_{i=0}^3 2^i x_i$$

is given in Figure 1. The left outgoing edge of each node represents the *low*-edge. At the edges the additive or multiplicative values are displayed. Quadratic nodes represent the leaf nodes.

2.2 Kronecker *BMDs (K*BMDs)

K*BMDs [17] are also a graph-based representation where the underlying graph is ordered. The handling of decomposition types and edge values is different to the two DD types described above. We allow additive **and** multiplicative weights in parallel (see also [30]) and, analogously to OKFDDs, use

¹ In contrast to [17] the K*BMDs considered in the following have integer (instead of rational) edge weights. This has shown advantages with respect to runtime for dynamic minimization (see Section 5).

Fig. 1. EVBDD and *BMD for unsigned integer encoding

three different decomposition types, i.e. the *Shannon decomposition* and the *positive* and *negative Davio* decomposition:

$$\langle (a, m), f \rangle = a + m((1 - x)f_{low(v)} + xf_{high(v)}) \quad (S)$$

$$\langle (a, m), f \rangle = a + m(f_{low(v)} + xf_{high(v)}) \quad (pD)$$

$$\langle (a, m), f \rangle = a + m(f_{low(v)} + (1 - x)f_{high(v)}) \quad (nD)$$

To make K*BMDs a canonical representation we need some further restrictions on the graph with respect to the edge values. Let a_l, m_l (a_h, m_h) denote the weights at the *low*-edge (*high*-edge) leaving v . Let f_{v_l} (f_{v_h}) denote the function represented at the endpoint of the *low*-edge (*high*-edge). Then the representation is *normalized*, iff the following holds:

1. There exists only one leaf and this leaf is labeled 0.
2. The *low*-edge of a node has additive weight $a_l = 0$.
3. The greatest common divisor of weights m_l , a_h and m_h is 1.

Some computations show that for the three decomposition types the normalization per node can be done along the following lines:

The normalized values for Shannon decomposition are computed according to the subsequent formula² (where *gcd* denotes the *greatest common divisor*).

$$\begin{aligned} f &= (a_l + m_l f_{v_l})(1 - x) + (a_h + m_h f_{v_h})x \\ &= a_l + \gcd(m_l, m_h, (a_h - a_l)) \left(\frac{m_l}{\gcd(m_l, m_h, (a_h - a_l))} f_{v_l}(1 - x) \right. \\ &\quad \left. + \left(\frac{a_h - a_l}{\gcd(m_l, m_h, (a_h - a_l))} + \frac{m_h}{\gcd(m_l, m_h, (a_h - a_l))} f_{v_h} \right) x \right) \end{aligned}$$

Due to these rules, the weights at the edges are integer numbers³. In K*BMDs for Boolean functions consisting of only Shannon nodes all inner nodes also represent only Boolean functions (in contrast to *BMDs) [17]. It follows, that K*BMDs are isomorphic to OBDDs, if the Shannon decomposition is carried out in each node. Thus, our hybrid data structure avoids the drawback of *BMDs that there exist functions that cannot be represented by *BMDs but have simple OBDD descriptions [20]. It follows from our normalization rule that K*BMDs have only one node for the representation of a single variable. In EVBDDs and *BMDs several nodes might be used. This is one of the explanations for the differences in the number of nodes (see Section 6).

In the following we always consider *reduced* K*BMDs. The reductions are performed analogously to OKFDDs. In addition, the weights at the edges have to be checked.

² The formulae for positive and negative Davio are very similar.

³ This is a modification of K*BMDs as introduced in [17], where rational edge weights have been used.

Example 2. A K^ BMD for the unsigned integer encoding is given in Figure 2. At the edges the additive and multiplicative values are displayed by (a, m) . The decomposition type of each node is also given.*

Fig. 2. K^* BMD for unsigned integer encoding

3 Bit- and Word-Level Representation

In [10,11] it has been claimed that * BMDs can also be used for bit-level verification. In this section we show by some experiments that OBDDs in general are much better suited. This especially shows that an extension of * BMDs to a data structure that is also a superset of OBDDs is useful.

In [13] it has been observed that the pD decomposition carried out in *Ordered Functional Decision Diagrams* (OFDDs) [23,19] is the same as the one carried out in BMDs, when a modulo 2 operation is performed on the terminals of the BMD. An analogous result holds for * BMDs if a modulo 2 operation is carried out in each node. Thus, the size of OFDDs is always smaller for bit-level representation than the size of the corresponding $(^*)$ BMD.

In the following we compare the size of OFDDs and OBDDs for the representation of some of the ISCAS85 benchmarks [6]. The OBDDs and OFDDs have been constructed and then a *sifting* like method [29] has been applied until no further improvement could be obtained. (All experiments have been carried out with the OKFDD package PUMA [16].)

The results are shown in Table 1. *name* denotes the name of the benchmark. OBDD (OFDD) denotes the number of nodes needed for the representation of the outputs of the corresponding function. As can easily be seen OBDDs are even better suited than OFDDs, and OFDDs are only a lower bound for *BMDs. Thus, *BMDs seem to be not very well suited for bit-level verification.

<i>name</i>	OBDD	OFDD
C432	1209	4011
C499	29561	480058
C880	5555	8995
C1355	29561	481505
C1908	9171	72682
C2670	6488	4124
C5315	2554	5882
C6288	-	-

Table 1. OBDDs versus OFDDs

On the other hand OBDDs and OFDDs both fail for functions that have simple word-level descriptions, like multipliers (C6288 in Table 1). Therefore a “good” data structure should support both types of representation.

4 Transformation between Bit- and Word-Level

In this section we present manipulation algorithms for K*BMDs that allow to transform bit- to word-level representations and vice versa. Thus, there is no longer the need to determine the encoding of the output functions in advance as it has been done in all previous work where word-level DDs have been used. The basic underlying principle of our two transformation operators *Contract* and *Split* is a recursive decomposition similar to the *Apply*-Operation used for OBDDs (see also [2]). The basic idea of the functionality of *Contract* and *Split* is illustrated in Figure 3.

To do so, consider two words

$$s := [s_{k-1}, s_{k-2}, \dots, s_0]$$

and

$$t := [t_{l-1}, t_{l-2}, \dots, t_0]$$

Fig. 3. *Contract* and *Split*

defined by $k \geq 1$ binary signals s_i and $l \geq 1$ binary signals t_j , respectively. (s_0 (t_0) is the least significant bit.) Let G_s and G_t be K*BMDs (with identical variable ordering and decomposition types per variable) representing the integer valued functions f_s and f_t for the words s and t , respectively.

We define the *Contract*-Operation by setting

$$f_{st} := \text{Contract}(f_s, f_t) := (2^l f_s + f_t),$$

i.e. f_{st} is the function realized on the concatenated word

$$s \cdot t := [s_{k-1}, s_{k-2}, \dots, s_0, t_{l-1}, t_{l-2}, \dots, t_0].$$

Using the equation⁴

$$\begin{aligned} \text{Contract}(f_s, f_t) &= (2^l f_s + f_t) \\ &= 2^l((1-x)f_{s,x=0} + xf_{s,x=1}) + ((1-x)f_{t,x=0} + xf_{t,x=1}) \\ &= (1-x)(2^l f_{s,x=0} + f_{t,x=0}) + x(2^l f_{s,x=1} + f_{t,x=1}) \\ &= (1-x)\text{Contract}(f_{s,x=0}, f_{t,x=0}) \\ &\quad + x\text{Contract}(f_{s,x=1}, f_{t,x=1}) \end{aligned}$$

it can be shown for the Shannon decomposition that *Contract* can be computed on K*BMDs by recursively computing *Contract* on the left and right subfunctions of G_s and G_t . (For the other decompositions analogous equations hold.)

Conversely, given a function f_r realized on the word

$$r := s \cdot t := [s_{k-1}, s_{k-2}, \dots, s_0, t_{l-1}, t_{l-2}, \dots, t_0]$$

⁴ $f_{s,x=i}$ is used to denote the cofactor of f_s with respect to $x = i$ ($i \in \{0, 1\}$).

we define the *Split*-Operation by setting

$$Split(f_r, s, t) := \{f_s, f_t\}$$

with f_s and f_t being the functions realized on words s and t , respectively. It follows (analogously to the *Contract* operation) that *Split* can also be computed by using recursive computations on the subfunctions of the K*BMD G_r for f_r .

Taken together, *Split* allows to dynamically switch from larger words to smaller words (or even bits), while *Contract* gives the possibility to group e.g. signal lines that belong together “semantically” and represent them in one K*BMD.

Thus, these operators allow to algorithmically describe the transformation between bit- and word-level and thus allow to dynamically switch between the two different forms of description. Obviously, these operations can easily be included in a dynamic minimization procedure (see next section).

5 Variable Ordering and Decomposition Type Choice

As well-known for OBDDs (and similar data structures) the ordered graphs discussed here are sensitive to the variable ordering. Thus, this problem has also to be considered. Additionally, as for OKFDDs [16] one out of three decomposition types has to be chosen.

One of the most promising methods for finding good variable orderings is *Dynamic Variable Ordering* (DVO) [22,29].

DVO is based on the exchange of two adjacent variables. For exchanging two adjacent variables in OBDDs only pointers must be redirected (see Figure 4).

Basically, the exchange for K*BMDs works in a similar way, but in contrast to OBDDs, it might happen that nodes and edge values in the upper levels have to be considered in order to normalize the representation. In this case an additional “repair” run is needed [17].

In [17] it has been shown that usual synthesis operations on K*BMDs can be carried out analogously to other data structures like OBDDs and *BMDs. Thus, changing the decomposition type can be carried out using synthesis operations analogously to OKFDDs [18].

In this paper we generalize a method known from OKFDDs [16], where the exchange of the decomposition type is integrated in the DVO process: A variable is sifted down to the bottom level. There the exchange can easily be performed, since in this level only one node exists. Since the exchange is integrated in the sifting process no additional costs result.

For a sketch of the algorithm see Figure 5.

Fig. 4. Exchange of adjacent variables

```

minimize_K*BMD (benchmark) {
  for (each variable) {
    for (all three decomposition types)
      sift_variable ();
    position_variable_locally_optimal ();
  }
  return ;
}

```

Fig. 5. Sketch of minimization algorithm

6 Experimental Results

In this section we present experimental results that were carried out on a *Sun Sparc 1+* workstation. We implemented the data structure K*BMDs in C++. For all experiments we set a node limit of 100.000 nodes.

In a first series of experiments we compare K*BMDs to other data structures, like EVBDDs [25], OFDDs [23] and *BMDs [10,11], on some of the *clique functions* for graphs from [20,4,3]:

1. $1-cl_{n,3}$ checks for the existence of exactly one clique.
2. $\#-cl_{n,3}$ counts the number of cliques.
3. $\oplus-cl_{n,3}$ determines the parity of the number of cliques.

The clique-functions have intensively been studied for proving exponential trade-offs between different types of DDs for the bit-level [4] as well as for the word-level [20,3]. For this they are also considered here. The number of nodes needed for the representation of the functions depending on n are given in Table 2. (An 'X' symbolizes that the function has an integer range and the data structure is Boolean only. A '-' symbolizes that the construction exceeded the given node limit.) Only the "pure" DDs are considered for K*BMDs, i.e. mixing of decomposition types is not yet considered. For all functions we used the initial variable ordering. Our experiments show, that K*BMDs can easily handle functions for which no efficient *BMD exists. Nevertheless, the function $\oplus-cl_{n,3}$ can only be represented efficiently using OFDDs. But they can easily be derived from (K)*BMDs, if a modulo 2 operation is performed in each node and then a reduce operation is carried out.

In a next series of experiments we compare the sizes of different types of DDs for some of the benchmarks from LGSynth91 [31] and ISCAS85 [6] using dynamic variable ordering. In Table 4 (Table 3) the results for benchmarks are given for which (no) dynamic reordering during the construction is needed. OBDD (*BMD) denotes the OBDD (*BMD) size. The sizes for the K*BMDs are given in the last column. The best results are given in bold. A dash symbolizes that the corresponding graph could not be constructed within the given node limit. As can easily be seen the K*BMDs in Table 3 are nearly always smaller. For only two examples the sizes are slightly larger. (The differences in Table 4 in comparison to Table 1) result from the "random" starting point of the sifting procedure.) For none of the ISCAS85 benchmarks the *BMD could be constructed, since too many nodes were needed during the synthesis procedures. The construction of the K*BMDs took only a few CPU seconds.

Finally, we consider multiplier circuits. The results for multipliers are given in Table 5. For OBDDs only circuits with a small number of inputs could be constructed, due to the exponential lower bound [8]. (K)*BMDs succeeded, if the word-level description of the function was given.

name	n	inp	OBDD	EVBDD	OFDD	*BMD	K*BMD		
							S	pD	nD
1-cl _{n,3}	5	10	47	48	114	171	47	154	65
	6	15	106	107	757	1610	106	1532	144
	7	21	207	208	6248	19676	207	19399	275
	8	28	367	368	-	-	367	-	477
	10	45	947	948	-	-	947	-	1185
#-cl _{n,3}	5	10	X	112	X	22	108	22	33
	6	15	X	764	X	40	757	40	59
	7	21	X	6464	X	65	6452	65	94
	8	28	X	-	X	98	-	98	139
	10	45	X	-	X	192	-	192	263
⊕-cl _{n,3}	5	10	94	115	22	123	94	112	217
	6	15	540	735	40	1360	540	1283	3040
	7	21	4005	6113	65	25410	4005	24647	-
	8	28	36392	-	98	-	36392	-	-
	10	45	-	-	192	-	-	-	-

Table 2. Comparison between different DD types.

name	OBDD	*BMD	K*BMD
apex6	571	612	531
apex7	315	312	264
frg1	82	160	81
frg2	1419	760	912
alu2	157	270	157
alu4	428	1076	349
cc	59	42	39
c8	79	79	68
k2	1270	2045	1097
lal	85	124	87
too_large	456	681	332
vda	479	715	439
x1	466	709	426

Table 3. DD sizes for small benchmarks

<i>name</i>	OBDD	*BMD	K*BMD
C432	1209	-	1209
C499	29561	-	29561
C880	7718	-	4048
C1355	29561	-	29561
C1908	5028	-	5944
C2670	4093	-	3939
C5315	2270	-	2504
des	3022	4425	3022
pair	3210	4833	2911
rot	7702	11191	5808

Table 4. DD sizes for large benchmarks

<i>name</i>	OBDD	*BMD	K*BMD
mult-4	134	8	8
mult-8	10283	16	16
mult-16	-	32	32
mult-30	-	60	60

Table 5. DD sizes for multipliers with word-level description

All in all, seen over the whole set of benchmarks K*BMDs are the only data structure that could represent all functions.

7 Conclusions and Future Work

Algorithms for dynamic manipulation of *Kronecker Multiplicative Binary Moment Diagrams* (K*BMDs) have been discussed. A method for changing the decomposition type and variable order known for OKFDDs has been shown to also work for K*BMDs. *Contract* and *Split* have been shown to allow the dynamic switch between word-level representations of differing width.

We presented experimental results that show the ability of K*BMDs to represent functions more compact than other data structures, like OBDDs, EVBDDs, OKFDDs and *BMDs. In [28] it has been shown that using an improved sifting method it is possible to construct OBDDs for all the ISCAS85 and ISCAS89 benchmarks except the multiplier C6288. Since K*BMDs are a superset of OBDDs all these results can directly be transferred. Additionally, K*BMDs can easily represent multipliers due to the Davio decomposition and thus are a data structure that can represent all ISCAS85 and ISCAS89 benchmarks.

The operations *Contract* and *Split* have not yet been integrated in the dynamic minimization process, i.e. our minimization algorithm does not make

use of the ability to switch (on the fly) from bit-level to word-level, and vice versa, although the algorithmical problem is solved. Currently we are working on heuristics to decide when and where such a switch during a symbolic simulation run is useful.

References

1. D.P. Appenzeller and A. Kuehlmann. Formal verification of a PowerPC micro-processor. In *Int'l Conf. on Comp. Design*, pages 79–84, 1995.
2. L. Arditi. *bmds can delay the use of theorem proving for verifying arithmetic assembly instructions. In *FMCAD*, 1996.
3. B. Becker, R. Drechsler, and R. Enders. On the computational power of bit-level and word-level decision diagrams. In *ASP Design Automation Conf.*, 1997.
4. B. Becker, R. Drechsler, and M. Theobald. OKFDDs versus OBDDs and OFDDs. In *ICALP, LNCS 944*, pages 475–486, 1995.
5. K.S. Brace, R.L. Rudell, and R.E. Bryant. Efficient implementation of a BDD package. In *Design Automation Conf.*, pages 40–45, 1990.
6. F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational circuits and a target translator in fortran. In *Int'l Symp. Circ. and Systems, Special Sess. on ATPG and Fault Simulation*, pages 663–698, 1985.
7. R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.
8. R.E. Bryant. On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication. *IEEE Trans. on Comp.*, 40:205–213, 1991.
9. R.E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM, Comp. Surveys*, 24:293–318, 1992.
10. R.E. Bryant and Y.-A. Chen. Verification of arithmetic functions with binary moment diagrams. Technical report, CMU-CS-94-160, 1994.
11. R.E. Bryant and Y.-A. Chen. Verification of arithmetic functions with binary moment diagrams. In *Design Automation Conf.*, pages 535–541, 1995.
12. E. Clarke, M. Fujita, P. McGeer, K. McMillan, J. Yang, and X. Zhao. Multi terminal binary decision diagrams: An efficient data structure for matrix representation. In *Int'l Workshop on Logic Synth.*, pages P6a:1–15, 1993.
13. E.M. Clarke, M. Fujita, and X. Zhao. Application of multi-terminal binary decision diagrams. *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 21–27, 1995.
14. E.M. Clarke, M. Fujita, and X. Zhao. Hybrid decision diagrams - overcoming the limitations of MTBDDs and BMDs. In *Int'l Conf. on CAD*, pages 159–163, 1995.
15. E.M. Clarke and X. Zhao. Word level symbolic model checking - a new approach for verifying arithmetic circuits. Technical Report CMU-CS-95-161, 1995.
16. R. Drechsler and B. Becker. Dynamic minimization of OKFDDs. In *Int'l Conf. on Comp. Design*, pages 602–607, 1995.
17. R. Drechsler, B. Becker, and S. Ruppertz. K*BMDs: A new data structure for verification. In *European Design & Test Conf.*, pages 2–8, 1996.
18. R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski. Efficient representation and manipulation of switching functions based on Ordered

- Kronecker Functional Decision Diagrams. In *Design Automation Conf.*, pages 415–419, 1994.
19. R. Drechsler, M. Theobald, and B. Becker. Fast OFDD based minimization of Fixed Polarity Reed-Muller expressions. In *European Design Automation Conf.*, pages 2–7, 1994.
 20. R. Enders. Note on the complexity of binary moment diagram representations. *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 191–197, 1995.
 21. M. Fujita, H. Fujisawa, and N. Kawato. Evaluation and improvements of boolean comparison method based on binary decision diagrams. In *Int'l Conf. on CAD*, pages 2–5, 1988.
 22. M. Fujita, Y. Matsunaga, and T. Kakuda. On variable ordering of binary decision diagrams for the application of multi-level synthesis. In *European Conf. on Design Automation*, pages 50–54, 1991.
 23. U. Kebschull, E. Schubert, and W. Rosenstiel. Multilevel logic synthesis based on functional decision diagrams. In *European Conf. on Design Automation*, pages 43–47, 1992.
 24. R. Krieger, B. Becker, and R. Sinković. A BDD-based algorithm for computation of exact fault detection probabilities. In *Int'l Symp. on Fault-Tolerant Comp.*, pages 186–195, 1993.
 25. Y.-T. Lai and S. Sastry. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Design Automation Conf.*, pages 608–613, 1992.
 26. S. Malik, A.R. Wang, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Logic verification using binary decision diagrams in a logic synthesis environment. In *Int'l Conf. on CAD*, pages 6–9, 1988.
 27. S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagrams with attributed edges for efficient boolean function manipulation. In *Design Automation Conf.*, pages 52–57, 1990.
 28. S. Panda and F. Somenzi. Who are the variables in your neighborhood. In *Int'l Conf. on CAD*, pages 74–77, 1995.
 29. R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.
 30. P. Tafertshofer and M. Pedram. Factored edge-valued binary decision diagrams. to appear in *Formal Methods in System Design*, 1996.
 31. S. Yang. Logic synthesis and optimization benchmarks user guide. Technical Report 1/95, Microelectronic Center of North Carolina, Jan. 1991.