On Variable Ordering and Decomposition Type Choice in OKFDDs *

Rolf Drechsler[1] Bernd Becker[2] Andrea Jahnke[1]

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

[2] Institute of Computer Science
Albert-Ludwigs-University
79110 Freiburg im Breisgau, Germany
email: becker@informatik.uni-freiburg.de

Abstract— We present methods for the construction of small Ordered Kronecker Functional Decision Diagrams (OKFDDs). OKFDDs are a generalization of Ordered Binary Decision Diagrams (OBDDs) and Ordered Functional Decision Diagrams (OFDDs) as well. Starting with an upper bound for the size of an OKFDD representing a tree-like circuit we develope different heuristics to find good variable orderings and decomposition types for OKFDDs representing two-level and multi-level circuits, respectively. Experimental results are presented to show the efficiency of our approaches.

I. Introduction

Decision Diagrams (DDs) are often used in CAD systems for efficient representation and manipulation of Boolean functions. The most popular data structure are *Ordered Binary Decision Diagrams (OBDDs)* [5] that are used in many applications [17, 7]. Nevertheless, some relevant classes of Boolean functions cannot be represented efficiently by OBDDs [6, 18, 2]. As one alternative *Ordered Functional Decision Diagrams (OFDDs)* [16] have been introduced and in the meantime they are used in various applications of XOR-based logic synthesis (see e.g. [20, 12]).

Recently, a new data structure, that is called *Ordered Kronecker Functional Decision Diagram (OKFDD)*, has been presented [11]. OKFDDs are a generalization of OBDDs and OFDDs as well and try to combine the advantages of both representations. The data structure allows to dynamically adapt the representation of a Boolean function to a given problem. Using OKFDDs it is possible to represent functions efficiently that only have exponentially sized OBDD and OFDD representations [1]. But as well-known for OBDDs [5] and OFDDs [2] OKFDDs are also very sensitive to the variable ordering [11, 9]. In addition to the position of a variable in the ordering a so-called decomposition type has to be chosen. Thus, there is a need for ordering heuristics for OKFDDs.

In [9] it has been shown that dynamic variable ordering methods for OBDDs [15, 18] can also be applied to OKFDDs. But in many cases it is important to choose a good starting order to avoid unnecessary sifting operations that are very time consuming.

*This work was supported in part by DFG grant Be 1176/4-2.

In this paper we present heuristics for choosing a good variable ordering and decomposition type. Many of the existing variable ordering heuristics for OBDDs [17, 14, 13] are motivated by the fact, that for tree-like circuits efficient OBDD representations are obtained, if the natural non-interleaving order of variables is used. We show a similar result for OKFDDs representing tree-like circuits. Our result gives an answer to a question posed in [8]. There it was conjectured that for arbitrary tree-like circuits including XOR gates no closed form result to predict their OBDD size is possible. In contrast, the upper bound for OKFDDs is also an upper bound for OBDDs. Then we develop methods for variable ordering and decomposition type choice starting from two-level and multi-level circuits. We perform a large set of experiments. With our two-level OKFDD heuristic we obtain better results than all previously published also for OBDDs and OFDDs although it was not designed for these special cases. The multi-level heuristic is based on a *fanin heuristic* [17] and on *interleaving of variables* [13]. Additionally, the decomposition type for each variable is determined using the polarity of the paths from all primary inputs to all primary outputs. This polarity is computed by a *depth first search.* Results for the combinational parts of some of the ISCAS89 benchmarks [4] are presented. It turns out that (with one exception) the OKFDDs obtained from our heuristic either are significantly smaller or at least have about the same size as OBDDs.

II. Kronecker Functional Decision Diagrams

In the following, basic definitions of Decision Diagrams and OKFDDs in particular are presented.

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 variable* for v, and has exactly two successors denoted by $low(v), high(v) \in V$. All nodes with label x_i are denoted as *level i*. 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 non-terminal nodes.

Further structural restrictions turn out to be important.

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.

DDs can be related to Boolean functions by using the following well-known decomposition types (given here for an arbitrary Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ over the variable set X_n):

$$f = \overline{x}_i f_i^0 + x_i f_i^1 \qquad \textit{Shannon } (S) \tag{1}$$

$$f = f_i^0 \oplus x_i f_i^2 \qquad \textit{positive Davio } (pD) \tag{2}$$

$$f = f_i^1 \oplus \overline{x}_i f_i^2 \qquad \textit{negative Davio } (nD) \tag{3}$$

where f_i^0 (f_i^1) denotes the *cofactor* of f with respect to $x_i = 0$ ($x_i = 1$) and f_i^2 is defined as $f_i^2 := f_i^0 \oplus f_i^1$, $\oplus$ being the Exclusive OR operation.

Decomposition types are associated to the variables in X_n with the help of a *Decomposition Type List (DTL)* $d := (d_1, \ldots, d_n)$ where $d_i \in \{S, pD, nD\}$.

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

Definition 3 A *KFDD over* X_n is given by a DD over X_n together with a fixed DTL, $d = (d_1, \ldots, d_n)$. The function $f_G^d : \mathbf{B}^n \to \mathbf{B}$ represented by a KFDD G over X_n with DTL d is defined as:

1. If G consists of a single node labeled with 0 (1), then G is a KFDD for the constant 0 (1) function.
2. If G has a root v with label x_i, then G is a KFDD for

$$\begin{cases} \overline{x}_i f_{low(v)} + x_i f_{high(v)} & : \ d_i = S \\ f_{low(v)} \oplus x_i f_{high(v)} & : \ d_i = pD \\ f_{low(v)} \oplus \overline{x}_i f_{high(v)} & : \ d_i = nD \end{cases}$$

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

Of course, a KFDD is an OKFDD iff the underlying DD is ordered.

A node in an OKFDD is called an *S-node* if it is expanded by Shannon decomposition - Equation (1). It is called a *D-node* if it is expanded by Davio decompositions - Equations (2) or (3); the latter being an *nD-node* and the former a *pD-node*. According to the decomposition type list, at every node of a fixed level i in the KFDD, the same decomposition of type d_i is applied. This directly infers that KFDDs are a generalization of BDDs and FDDs: If in all levels only Shannon decomposition is applied, the KFDD will be a BDD. If only Davio decompositions are applied, i.e. $d_i \in \{pD, nD\}$ for all i, the KFDD will be an FDD. Analogously pFDDs and nFDDs are defined.

Reductions of three different types are used to reduce the size of KFDDs:

Type I: Delete a node v' whose label is identical to the label of a node v and whose successors are identical to the successors of 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 v' and connect the incoming edges of v to v'.

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

While each node in a KFDD is a candidate for the application of reduction type I, only S-nodes (D-nodes) are candidates for the application of the reduction type S (type D). A KFDD is *reduced* iff no reductions can be applied to the KFDD. Two KFDDs G_1 and G_2 (with the same DTLs) are called *equivalent* iff G_2 results from G_1 by repeated applications of reductions and inverse reductions. A KFDD G_2 is called the *reduction* of a KFDD G_1 iff G_1 and G_2 are equivalent and G_2 itself is reduced.

From [11], it is known that reductions can be used to define canonical representations for not only OBDDs and OFDDs, but also for OKFDDs.

The size of an OKFDD can be further reduced if *Complemented Edges* (CEs) are used [11]. Then a node represents a function and its complement at the same time. (The use of CEs implies that the computation of the complement can be performed in constant time.) The representation can be chosen in a way that it further remains canonical. Experiments show that CEs reduce the size of OKFDDs by 10 % on average.

III. Ordering Heuristics

The size of an OKFDDs largely depends on the variable ordering and the chosen DTL [1]:

1. For a given DTL the size of the OKFDD may vary from linear to exponential if the variable ordering is changed.
2. For a given variable ordering the size of the OKFDD may vary from polynomial to exponential if the DTL is changed.

Thus the problem to be solved is more difficult than for OBDDs and OpFDDs, since not only a good ordering must be chosen. Additionally a good DTL must be determined.

We first prove for circuits with a tree-like structure that for each given DTL an efficient OKFDD can be constructed. Then we give heuristics for two-level and multi-level circuits.

A. OKFDDs for Tree-like Circuits

Most often OKFDDs are constructed from the circuit description of a Boolean function. One important subclass are tree-like circuits, i.e. circuits without reconvergencies. A lot of OBDD heuristics use the fact that OBDDs for tree-like circuits have small size, if a good variable ordering is chosen. The variable ordering for tree-like circuits is important as the example from [5] has shown, where an exponential gap has been proven for given Boolean functions.

In the following we show that tree-like circuits over basic gates with ≤ 2 inputs can also be represented efficiently by OKFDDs independent of the chosen decomposition types. For the proof we use OKFDDs without CEs. Obviously, OKFDDs with and without CEs differ in size at most by a factor of 2. Furthermore we will construct OKFDDs with so-called 'isolated Davio low-paths':

Let G be any OKFDD with fixed DTL and ordering. Consider an interval $I = [k, l]$ of maximal length with the property, that all variables from position k to l have type pD or nD. I uniquely defines the part of G consisting of exactly the nodes labeled with variables from position k to l. We call this part of G the *D-interval* I of G. Analogously, *S-intervals* of G consisting only of Shannon nodes are defined. The whole OKFDD can be partitioned into a uniquely defined sequence of non-empty D- and S-intervals. A node is called a root node of a D-interval (S-interval) I, if it is labeled with a variable from I and none of its predecessors has a label from I. Now consider a root v in a D-interval I. Then the *Davio Low-Path* (DLP) in the interval I starting in v is the path (within I) 'determined' by the input combination that sets all variables in I to 0. A path from the root of the OKFDD to a leaf is called *complete DLP in G*, if it is composed from DLPs and S-paths. (S-paths are paths within S-intervals.) A complete DLP is called *isolated*, if the following holds: The incoming edges of all nodes on the path belong to (other) complete DLPs. Now, an OKFDD G is an *OKFDD with isolated DLPs*, if each complete DLP is isolated.

It is easy to see, that OKFDDs with isolated DLPs can be negated by negating the terminal nodes of the complete DLPs and that the property *OKFDD with isolated DLPs* is not destroyed. In the following we show inductively how this structural property can be maintained during the construction of OKFDDs for tree-like circuits.

Lemma 1 Let $B = \{$AND, NAND, OR, NOR, EQUIV, XOR, INVERTER$\}$ be the set of basic gates with ≤ 2 inputs and $d = (d_1, d_2, \ldots, d_n)$ a DTL for n variables. Then it holds:

> Each tree-like circuit SK over B with n inputs can be represented by an OKFDD with DTL d of size $\leq n^{\lg 3}$.

Proof: We prove the lemma by induction on n, the number of inputs of SK. For each n we construct a small OKFDD with isolated Davio low-paths. For $n = 1$, the resulting OKFDD clearly has ≤ 1 non-terminal nodes.

If $n > 1$, we may assume that SK consists of two tree-like sub-circuits SK_f and SK_g and a gate, that combines SK_f and SK_g. SK_f (SK_g) realizes the Boolean function f (g) defined over n_f (n_g) variables. (Of course, the variable sets are disjoint and $n_f + n_g = n$.)

Assume inductively, that F (G) is an OKFDD with isolated Davio low-paths for f (g) with $\leq n_f^{\lg 3}$ ($\leq n_g^{\lg 3}$) nodes. Without loss of generality $n_f \geq n_g$. We have to prove for all binary operations that they can be realized within the bounds given by the lemma and that the structural property of isolated DLPs is maintained.

Let $*$ be any of these operations and define the variable order for the OKFDD of $h = f * g$ to be given as the concatenation of the order for F and the order for G. Consider the root node v of the OKFDD for $f * g$. v is labeled with x_i, the first variable of F. We recursively compute the OKFDDs rooted at $low(v)$ and $high(v)$. Since g is independent of x_i, these are OKFDDs for the functions

$$\begin{array}{lcll} f_0 * g & \text{and} & (f_0 * g) \oplus (f_1 * g) & : \ d_i = pD \\ f_1 * g & \text{and} & (f_0 * g) \oplus (f_1 * g) & : \ d_i = nD \\ f_0 * g & \text{and} & (f_1 * g) & : \ d_i = S \end{array}$$

Replacing $*$ by AND, OR and XOR we conclude the following for nodes with type pD:

$$\begin{array}{rcl} f \cdot g & = & (f_0 \cdot g) \oplus x_i (f_2 \cdot g) \\ f + g & = & (f_0 + g) \oplus x_i (f_2 \cdot \overline{g}) \\ f \oplus g & = & (f_0 \oplus g) \oplus x_i (f_2) \end{array}$$

Similar equations hold for nodes with type nD and S, respectively. We conclude, that the OKFDD H for $f * g$ is obtained from the OKFDD for f by replacing some of the terminals by G and some of them by the complement of G.

Using the induction hypothesis we get

$$\begin{array}{rcl} |H| & \leq & |F| + 2|G| \\ & \leq & n_f^{\lg 3} + 2 n_g^{\lg 3} \\ & \leq & n^{\lg 3} \end{array}$$

(Notice, that for the last inequation $n_g \leq n/2$ is essential.) □

Our result gives an answer to a question posed in [8]. There it was conjectured that for arbitrary tree-like circuits including XOR gates no closed form result to predict their OBDD size is possible. In contrast the upper bound from Lemma 1 for OKFDDs is also an upper bound for OBDDs[1].

For general circuits such polynomial upper bounds cannot be given [2]. In the following we present heuristics that in many cases allow the construction of 'small' OKFDDs.

B. OKFDDs for Two-level Circuits

Often the circuit description is given as a *Sum Of Product expression (SOP)*. In this paper we present a new ordering heuristic that is also compared to OBDD

[1] The same result for OBDDs has been obtained independently in [21].

```
pla_heuristic (benchmark) {
    read_literals () ;
    for ( i=1; i ≤ inputs; i++) {
        if (plit == 0 && nlit == 0 || k = 0)
            {node_i = S; var_weight_i = plit_i + nlit_i; }
        else if (plit_i/(plit_i+nlit_i) ≤ k)
            {node_i = nD; var_weight_i = plit_i/(plit_i+nlit_i); }
        else if (nlit_i/(plit_i+nlit_i) ≤ k)
            {node_i = pD; var_weight_i = nlit_i/(plit_i+nlit_i); }
        else {node_i = S; var_weight_i = plit_i + nlit_i; }
    }
    max_weight = max_{d_i=S}(var_weight_i);
    for (all S-nodes with var_weight_i ≠ 0)
        var_weight_i = (max_weight−var_weight_i)/max_weight;
    sort_variables () ;
}
```

Fig. 1. Algorithm for SOP heuristic

[15] and OFDD [19] heuristics by restricting OKFDDs to these special cases. Although our heuristic was not designed for OBDDs and OFDDs it performs better for *all* benchmarks investigated (see Section IV.).

We consider PLAs with an AND-plane that is followed by an OR-plane. For a given circuit a parameter $k \in [0, 0.5]$ must be chosen. This parameter influences the choice of the decomposition type as will be discussed later.

For each variable x_i the number of positive literals $plit_i$ and the number of negative literals $nlit_i$ is determined as follows: Define the multiplicity of a literal in row l to be the number of 1's in row l of the OR-plane. Then $plit_i$ ($nlit_i$) is given by summing up over all rows the occurrences of x_i ($\overline{x}_i$) multiplied with their multiplicities. Then for each variable x_i a weight is determined by distinguishing the following cases for a fixed paramter k:

$$var\_weight_i = \begin{cases} plit_i + nlit_i & : \; nlit_i{=}plit_i{=}0 \vee k{=}0 \\ \frac{plit_i}{plit_i+nlit_i} & : \; \frac{plit_i}{plit_i+nlit_i} \leq k \\ \frac{nlit_i}{plit_i+nlit_i} & : \; \frac{nlit_i}{plit_i+nlit_i} \leq k \\ plit_i + nlit_i & : \; else \end{cases}$$

In the first and last case the nodes will be S-nodes. (The difference between $plit_i$ and $nlit_i$ is 'small'.) In the second (third) case the node will be an nD-node (pD-node). Then the maximal weight $max\_weight = max_{d_i=S}(var\_weight_i)$ is determined and for all S-nodes with $var\_weight_i \neq 0$ the weight is scaled by

$$var\_weight_i := \frac{max\_weight - var\_weight_i}{max\_weight}.$$

Then the variables are sorted in increasing order according to their weights. If there is a tie the variables are ordered as they occur in the benchmark. For S-nodes (D-nodes) this means that the most binate (unate) variables are ordered first. A sketch of the algorithm is given in Fig. 1.

As special cases we obtain: For $k = 0$ the OKFDD is an OBDD and for $k = 0.5$ the OKFDD is an OFDD. The best results on average for OKFDDs have been obtained for $k = 0.25$.

Example 1 Consider a function $f : \mathbf{B}^3 \rightarrow \mathbf{B}^2$ given by the following SOP:

110 01
100 01
111 11

Then we obtain the following output from the function read_literals():

i	$nlit_i$	$plit_i$
1	0	4
2	1	3
3	2	2

Thus for $k = 0.25$ x_1 and x_2 will become pD-nodes and x_3 will become an S-node. Since x_1 and x_3 have the same weight x_1 is before x_3. The last variable in the ordering is x_2.

C. OKFDDs for Multi-level Circuits

Our heuristic for multi-level circuits first determines a good variable ordering and afterwards determines the decomposition type for each variable.

The ordering is determined by a *fanin heuristic* similar to [17]. For each output an ordering is computed. Then the outputs are sorted in decreasing order with respect to the number of cells. If there is a tie the second sorting criterion is the number of inputs on which the output depends. Then the different orderings are merged by *variable interleaving* [13].

For the computation of the decomposition type we first determine the number of paths pos_path_i (neg_path_i) with positive (negative) polarity from input i to all outputs. This can simply be done by a depth first search algorithm. For all variables the decomposition type is then determined as follows:

$$type_i = \begin{cases} pD & : \; pos\_path_i - neg\_paths_i \geq 10 \\ nD & : \; neg\_path_i - pos\_paths_i \geq 10 \\ S & : \; else \end{cases}$$

Thus S-nodes are used, if the number of positive and negative polarity paths only differs by a small number. The bound 10 has been determined by experimental results. Additionally, experiments have shown that the node types in adjacent levels should not change too often. Thus, alternating decomposition types should be avoided. For this we finally perform an operation that is called *smoothing* of the node types. The operation works as follows: If both neighbours of a node are of the same type then this node is also set to this type.

Example 2 Let $d = (pD, pD, S, S, nD, S)$ be a DTL. Then the smoothed DTL d_f is given by:

$$d_f = (pD, pD, S, S, S, S)$$

A sketch of the algorithm is given in Fig. 2.

```
ml_heuristic (benchmark) {
   read_circuit () ;
   for (all outputs)
       compute_variable_ordering();
   variable_interleaving();
   compute_node_type();
   apply_smoothing();
}
```

Fig. 2. Algorithm for multi-level heuristic

TABLE I
Comparison to OBDD heuristic

name	*in*	*out*	FMK	our method
apex1	45	45	4892	1523
apex2	39	3	12580	2840
apex3	54	50	5621	1015
planet	14	25	2247	444
sand	17	14	2315	436
styr	15	15	2428	447
seq	41	35	7424	1528

IV. Experimental Results

In this section we present experimental results that confirm the advantage of the heuristics of the previous section in comparison to other approaches. All experiments were carried out on the OKFDD package presented in [10, 9] on an *HP 9000/710* workstation with a node limit of 250000 nodes.

A. Two-level Circuits

In a first series of experiments we compare our two-level heuristic (for $k = 0$ and $k = 0.5$) to previously published results for OBDDs [15] (FMK) in Table I and for OFDDs [19] (SKR) in Table II, respectively. The benchmarks were first optimized using ESPRESSO [3]. *in* (*out*) denotes the number of inputs (outputs) of the corresponding benchmark. (The run times for the two-level circuits are not given, since they are smaller than one CPU second for almost all examples.)

Notice that our heuristic is better for *all* considered benchmarks although it was not designed for this special case.

Finally we, give a comparison how the heuristic works for different choices of the parameter k. The results are given in Table III. In the last row the sum over all considered benchmarks is given: For OKFDDs with $k = 0.25$ we obtain the best results. Thus OKFDDs are superior to OBDDs and OFDDs on average. On the other hand, for specific examples $k = 0$ (circuit *chkn*) or $k = 0.5$ (circuit *add6*) might be superior. Thus OKFDDs also allow to construct the graphs with various values of k and to choose the best result. This is especially applicable to small benchmarks, since the time for the construction is neglible.

TABLE II
Comparison to OFDD heuristic

name	*in*	*out*	SKR	our method
alu4	14	8	1890	1435
apex4	9	19	1825	1008
cps	24	109	5205	1846
duke2	22	29	2160	616
ex5	8	63	952	377
pdc	16	40	2783	842
spla	16	46	2386	925
vg2	25	8	2415	318

TABLE III
Comparison for different parameters

name	*in*	*out*	k=0	k=0.25	k=0.5
add6	12	7	173	173	40
addm4	9	8	178	178	146
adr4	8	5	41	41	24
alu4	14	8	1041	1034	1435
b12	15	9	83	102	88
bc0	26	11	803	693	954
bw	5	28	113	107	113
chkn	29	7	370	636	671
cps	24	109	1663	1469	1846
e64	65	65	2145	2041	2040
ex5	8	63	295	295	377
f51m	8	8	60	60	43
intb	15	7	929	663	861
misex1	8	7	41	44	40
mlp4	8	8	159	159	113
radd	8	5	42	42	24
rd53	5	3	16	16	13
rd73	7	3	30	30	21
tial	14	8	1140	1040	1441
ts10	22	16	352	165	165
vg2	25	8	431	280	318
Z5xp1	7	10	76	76	65
Z9sym	9	1	24	24	26
$\sum$			10205	9368	10864

B. Multi-level Circuits

For the multi-level circuits we consider the combinational parts of some of the ISCAS89 benchmarks [4]. Since efficient OFDD heuristics for multi-level circuits have not been presented we restrict ourselves in the following to a comparison between heuristics for OBDDs and OKFDDs.

For this we consider a *fanin heuristic* similar to [17] and a heuristic that is based on variable interleaving [13].

TABLE IV
COMPARISON TO OBDD HEURISTICS

name	*in*	*out*	fanin	inter	our method	time
cs953	45	32	501	500	500	0.1
cs1196	32	32	4017	4017	3833	0.7
cs1238	32	32	3921	3920	3711	0.7
cs1423	91	79	14886	14481	30004	8.1
cs1488	14	25	500	500	500	0.1
cs1494	14	25	500	500	500	0.1
cs5378	214	228	11595	9967	4539	1.2
cs9234	247	250	-	37322	26707	7.7
cs13207	700	790	13327	5796	5814	1.4
cs15850	611	684	73305	68421	70281	39.2
cs35932	1763	2048	6190	6501	6501	5.2
cs38417	1664	1742	-	-	-	-
cs38584	1464	1730	30995	30371	31978	6.9

The results for the *fanin heuristic* (column fanin) and the heuristic based on interleaving (column inter) with a comparison to our appraoch are given in Table IV. In the last column the run time of our approach in CPU seconds is given. It is important to mention that although the OBDD and the OKFDD size is equal for some benchmarks the DTLs are always different. The experiments show that on average OKFDDs allow smaller representation of the circuits. There is only one example, i.e. *cs1423*, for which the OKFDD size is significantly larger than the OBDD size. In all other cases the size of the OKFDD is either comparable to that of the OBDD or significantly smaller, e.g. more than 50 % for *cs5378*.

V. Conclusions

We presented heuristics for OKFDDs to choose a good variable ordering and a good decomposition type. Heuristics for two-level and multi-level circuits were given. Additionally we proved an upper bound on the size of OKFDDs for tree-like circuits. Obviously this also includes upper bounds for OBDDs and OFDDs in contrast to the conjecture in [8].

We performed a large set of experimental results to show the efficiency of our approach. It has been shown that OKFDDs are often more efficient than OBDDs and allow more succinct representations.

References

[1] B. Becker, R. Drechsler, and M. Theobald. OKFDDs versus OBDDs and OFDDs. In *ICALP, LNCS*, 1995.

[2] B. Becker, R. Drechsler, and R. Werchner. On the relation between BDDs and FDDs. In *LATIN, LNCS 911*, pages 72–83, 1995.

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

[4] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Int'l Symp. Circ. and Systems*, pages 1929–1934, 1989.

[5] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 8:677–691, 1986.

[6] 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.

[7] R.E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM, Comp. Surveys*, 24:293–318, 1992.

[8] K.M. Butler and M.R. Mercer. *Assessing Fault Model and Test Quality.* Kluwer Academic Publishers, 1992.

[9] R. Drechsler and B.Becker. Dynamic minimization of OKFDDs. In *Int'l Conf. on Comp. Design*, 1995.

[10] R. Drechsler and B. Becker. PUMA: An OKFDD-package and its implementation. In *European Design & Test Conf.*, 1995. University Booth.

[11] 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.

[12] 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.

[13] H. Fujii, G. Ootomo, and C. Hori. Interleaving based variable ordering methods for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 38–41, 1993.

[14] M. Fujita, H. Fujisawa, and Y. Matsunaga. Variable ordering algorithms for binary decision diagrams and their evolution. *IEEE Trans. on CAD*, 12:6–12, 1993.

[15] M. Fujita, Y. Matsunga, 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.

[16] 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.

[17] 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.

[18] Richard Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.

[19] E. Schubert, U. Kebschull, and W. Rosenstiel. Some optimizations for functional decision diagrams. *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design, Hamburg*, pages 176–180, 1993.

[20] C.C. Tsai and M. Marek-Sadowska. Boolean matching using generalized Reed-Muller forms. In *Design Automation Conf.*, pages 339–344, 1994.

[21] I. Wegener. Comments on "A characterization of binary decision diagrams". *IEEE Trans. on Comp.*, 43:383–384, 1994.