On the Relation Between BDDs and FDDs

(Extended Abstract)

Bernd Becker* Rolf Drechsler†

Ralph Werchner†

TR-94-005

*Fachbereich 20 - Informatik, J.W.Goethe-Universität, D-60054 Frankfurt, and International Computer Science Institute, Berkeley, CA 94707; email: becker@informatik.uni-frankfurt.de

†Fachbereich 20 - Informatik, J.W.Goethe-Universität, D-60054 Frankfurt; email: <name>@informatik.uni-frankfurt.de

The first and second author were supported by DFG grant Be 1176/4-1.

Abstract

Data structures for Boolean functions build an essential component of design automation tools, especially in the area of logic synthesis. The state of the art data-structure is the ordered binary decision diagram (OBDD), which results from general binary decision diagrams (BDDs), also called branching programs, by ordering restrictions. In the context of EXOR-based logic synthesis another type of decision diagram (DD), called (ordered) functional decision diagram ((O)FDD) becomes increasingly important. BDDs (FDDs) are directed acyclic graphs, where a Shannon decomposition (Reed-Muller decomposition) is carried out in each node.

We study the relation between BDDs and FDDs. Both, BDDs and FDDs, result from DDs by defining the represented function in differing ways. If the underlying DD is complete, the relation between both types of interpretation can be described by a well-known Boolean transformation τ. This allows us to relate the OFDD-size of f and the OBDD-size of $\tau(f)$. We use this property to derive several results on the computational power of OFDDs and OBDDs. Symmetric functions are shown to have efficient representations as OBDDs and OFDDs as well. Classes of functions are given that have exponentially more concise OFDDs than OBDDs, and vice versa. In contrast to OBDDs, an exponential blow-up may occur in an AND-synthesis operation on two OFDDs. Finally, we demonstrate how the lower bound techniques for OBDDs can be adapted to OFDDs: We prove that the hidden weighted bit function and multiplication as well require OFDDs of exponential size independent of the ordering of the variables.

Topics: *Algorithms and data structures, complexity and computability, VLSI systems*

1 Introduction

The increasing complexity of modern VLSI circuitry is only manageable together with advanced CAD systems which as one important component contain (logic) synthesis tools. The problems to be solved, often can be formulated in terms of Boolean functions. The efficiency of the representation and the manipulation algorithms performing (synthesis) operations largely depends on the type of data structure chosen. The most popular data structure is the ordered binary decision diagram (OBDD), which is a restricted form of a binary decision diagram (BDD), also called branching program ([Lee59, Ake78, Weg87, Mei89]). OBDDs as a data structure in design automation were introduced by Bryant ([Bry86]) and in the meantime are used in many applications ([Bry92]). OBDDs allow efficient operations and manipulations, e.g. Boolean AND of two OBDDs, test of satisfiability and test of equivalence. Nevertheless, there exist functions that cannot be represented by small OBDDs ([Bry91]). This is the reason why many authors investigate the usefulness of generalizations or modifications of OBDDs for design automation (see e.g. [GM92, SW92a, LS92]).

In this context a new type of decision diagram, called ordered functional decision diagram (OFDD), has recently been introduced ([KSR92, KR93, BDT93]). BDDs (FDDs) are directed acyclic graphs, where a Shannon decomposition (Reed-Muller decomposition) is carried out in each node. While BDDs naturally correspond to AND/OR multi-level logic circuits and in this sense are suitable for synthesis tools that target on circuits realized by standard gates AND and OR, OFDDs can directly be transformed into AND/EXOR multi-level logic circuits. Lately, synthesis based on AND/EXOR realizations has gained more and more interest ([Sas93]). This stems at least partly from the fact, that, due to technological improvements, in many applications it is possible to handle EXOR gates as standard gates with the same speed and costs as OR gates (see e.g. [Xil88, WP92, PH91]). Furthermore, AND/EXOR realizations proved to be very efficient for large classes of circuits, e.g., arithmetic circuits, error correcting circuits and circuits for tele-communication ([Ber68, Sau92]), and, in addition, have nice testability properties ([Red72, DB93]). First experimental results given in [KSR92, SKR92, KR93, BDT93] demonstrate that OFDDs might be very useful for AND/EXOR based synthesis and might even be an interesting alternative data structure for the representation of Boolean functions.

Up to now a theoretical background for OFDDs as it exists for OBDDs with the huge number of results on the properties of branching programs (see e.g. [Weg87, Mei89]) has not been provided. On the other hand, this background (as well as further practical experience) is necessary to judge in which sense OFDDs might be a competitive data structure for Boolean functions. The goal of this paper is to make a step in this direction and to clarify the relation between BDDs and FDDs from the theoretical point of view.

Since FDDs and BDDs as well are based on the same underlying graph, analogously to BDDs, differing classes of FDDs, namely free, complete, and ordered FDDs, can be defined. If the Shannon expansion (Reed-Muller expansion) is used to relate the functional meaning to the DD, a BDD (FDD) results. Adapting the notion of *reductions* from BDDs, the unique minimal OFDD representing a given function with a given order of variables can easily be computed. Thus equivalence and satisfiability tests for OFDDs are as easy as for OBDDs.

If the underlying DD is complete, the relation between both types of interpretation can be described by a bijective Boolean transformation τ, which has already been used in theoretical computer science ([Pat86, Raz86]) and design automation ([BT92]) as well. The transformation allows us to relate the OFDD-size of f and the OBDD-size of $\tau(f)$. We use this property to derive several results on the computational power of OFDDs and OBDDs. It follows directly that variable ordering has a large influence on the size of the representations and that for some functions FFDDs are exponentially smaller than OFDDs. Symmetric functions in n variables are shown to have $O(n^2)$

size representations as OBDDs and OFDDs as well. Classes of (clique) functions are given that have exponentially more concise OFDDs than OBDDs and vice versa. In contrast to OBDDs, it turns out, that for OFDDs the AND-operation (OR-operation) may lead to an exponential blow-up, while the EXOR-operation and negation still can be performed efficiently. Finally, we demonstrate how the lower bound techniques for OBDDs can be adapted to OFDDs. We prove, that the hidden weighted bit function and multiplication as well require OFDDs of exponential size independent of the ordering of the variables.

The paper is structured as follows: In section 2 DDs, BDDs and FDDs and some important properties are presented. Section 3 studies the relation between the different data structures defined in section 2. As a first general result OBDD- and OFDD-sizes of functions can be compared. In the section 4 this result is applied to specific function classes to determine their OBDD-size and OFDD-size. The complexity of operations is analyzed and the lower bound proofs are given. We finish with a resume of the results and a discussion of further work in section 5.

2 Decision Diagrams

We provide an introduction to basic notions used in this paper and review essential definitions and properties of BDDs and FDDs. The core of the two data structures is a decision diagram (DD), which is a special type of labeled directed acyclic graph.

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, *nonterminal* and *terminal* vertices. A nonterminal vertex v is labeled with a variable from X_n, called the *decision variable* for v, and has exactly two successors denoted by $low(v), high(v) \in V$. A terminal vertex v is labeled with 0 or 1 and has no successors. G has exactly one source node called the *root* of the DD.

The size of a DD $G = (V, E)$, denoted $|G|$, is given by the number of nonterminal nodes. If DDs are to be used as a data structure in design automation, further restrictions on the structure of DDs turn out to be useful to provide efficient manipulation algorithms: A DD is *complete*, if each variable is encountered exactly once on each path in the DD from the root to a terminal vertex. 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.

Since we are interested in small representations of Boolean functions, we define methods to reduce decision diagrams. There are three reduction types, that can partially be combined:

Type 1: Identify two nodes v, v' in the DD, where the sub-DDs rooted by v, v' are isomorphic.

Type 2: 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 3: Delete a node v whose successor $high(v)$ points to the terminal 0 and connect the incoming edges of the deleted node to the successor $low(v)$.

A reduction of type s is called a (t_s)-*reduction*. In figure 1 the graphical representation of a (t_2)-reduction and a (t_3)-reduction is shown.

For a set $S \subset \{1, 2, 3\}$ we call two DDs G_1 and G_2 *(t_S)-equivalent*, iff G_2 results from G_1 by a sequence of reductions and inverse reductions with types from S. A DD is *(t_S)-reduced*, if no (t_s)-reduction with $s \in S$ can be applied to the DD. A DD G_2 is called the *(t_S)-reduction* of a DD G_1,

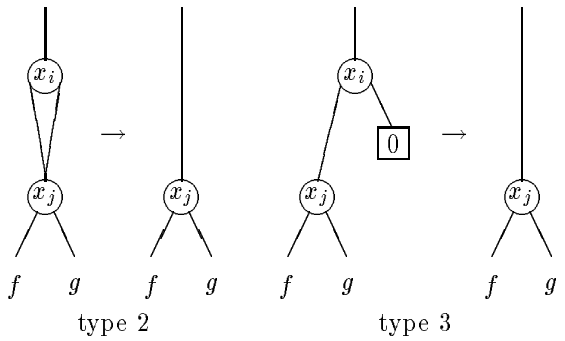

Figure 1: Reduction types

if G_2 results from G_1 by repeated applications of (t_s)-reductions and inverse (t_s)-reductions with $s \in S$ and G_2 itself is (t_S)-reduced.

In the following we only consider (t_S)-reduced DDs for $S \in \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\}\}$. $(t_{\{1,2\}})$-reductions are well-known from BDDs. In [KSR92] they have also been applied to OFDDs. In the following it turns out that for OFDDs $(t_{\{1,3\}})$-reductions are more natural. A careful analysis of the proofs in [Bry86, SW92b] shows that the following lemma is valid for ordered DDs:

Lemma 1 The (t_S)-reduction of an ordered DD G for $S \in \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\}\}$ is uniquely determined and can be computed in time $O(|G|)$.

Definition 1 Each DD G over $X_n := \{x_1, x_2, \ldots, x_n\}$ can be assigned to a Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ in two different ways, denoted by f_G^{BDD} and f_G^{FDD} respectively:

1) If G consists of a single node labeled with 0 (*resp.* 1), then $f_G^{\mathrm{BDD}} = f_G^{\mathrm{FDD}} = 0$ (*resp.* $f_G^{\mathrm{BDD}} = f_G^{\mathrm{FDD}} = 1$).

2) If the root v of G is labeled by x_i and G_0 (*resp.* G_1) is the DD rooted at $low(v)$ (*resp.* $high(v)$) then $f_G^{\mathrm{BDD}} = x_i\, f_{G_0}^{\mathrm{BDD}} + \bar{x}_i\, f_{G_1}^{\mathrm{BDD}}$ and $f_G^{\mathrm{FDD}} = f_{G_0}^{\mathrm{FDD}} \oplus x_i\, f_{G_1}^{\mathrm{FDD}}$

Note that for a given $x \in \mathbf{B}^n$ the value of $f_G^{\mathrm{BDD}}(x)$ can be computed by traversing a single path in G from the root to a terminal node, whereas for the computation of $f_G^{\mathrm{FDD}}(x)$ the Boolean function represented by each node of G may have to be evaluated.

A DD G with $f = f_G^{\mathrm{BDD}}$ (*resp.* $f = f_G^{\mathrm{FDD}}$) will be called a BDD (*resp.* FDD) for f. If G is free or ordered, it is called a FBDD or OBDD (*resp.* FFDD or OFDD).

Definition 2 For a Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ and a total order π on the set of variables X_n let OBDD-SIZE$_\pi(f)$ denote the minimum size of an OBDD representing f respecting the order π. Set OBDD-SIZE$(f) = \min_\pi$ OBDD-SIZE$_\pi(f)$ and denote by FBDD-SIZE(f) the minimum size of a free BDD for f. Define the measures OFDD-SIZE$_\pi(f)$, OFDD-SIZE(f) and FFDD-SIZE(f) analogously considering FDDs for f.

For each Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ and each order of the variables in X_n a complete OBDD (*resp.* OFDD) can be constructed by repeated application of the Shannon expansion (*resp.* Reed-Muller expansion) to f and its subfunctions. The resulting DD is a complete binary tree of height n with 2^n terminal nodes as its leaves.

If G_1 and G_2 are $(t_{\{1,2\}})$-equivalent DDs, then $f_{G_1}^{\mathrm{BDD}} = f_{G_2}^{\mathrm{BDD}}$. It is well-known that for a given ordering the $(t_{\{1,2\}})$-reduced OBDD of a Boolean function f is uniquely determined ([Bry86]). On the other hand it is easy to verify that for two $(t_{\{1,3\}})$-equivalent DDs G_1 and G_2 we have $f_{G_1}^{\mathrm{FDD}} = f_{G_2}^{\mathrm{FDD}}$. Combining this with lemma 1 we conclude that, starting with any OFDD for f with some order π on the variables, repeated applications of (t_1)- and (t_3)-reductions finally result in the unique minimal OFDD for f with order π. Thus, equivalence and satisfiability can efficiently be decided.

On the other hand, SAT-COUNT ,i.e. computing $\left(f_G^{\mathrm{FDD}}\right)^{-1}(1)$, is possible in linear time for FBDDs, while it is NP-hard for FFDDs:

Theorem 1 SAT-COUNT$_{FFDD}$ is NP-hard.

Proof: We give a reduction from 3SAT (satisfiability of a CNF with clauses of 3 literals) to SAT-COUNT$_{FFDD}$. Let λ be a CNF with m clauses, each containing exactly 3 literals, over X_n. Denote by l_i $(1 \leq i \leq n)$ the number of occurences of x_i or $\bar{x_i}$ in λ. Define the Boolean function f_1 over the set of variables

$$X = \{x_{i,j} | 1 \leq i \leq n\ ,\ 1 \leq j \leq l_i\}$$

by the formula obtained from λ by replacing the j-th occurence of x_i in λ by $x_{i,j}$. Define the function f_0 over X as

$$f_0 = \bigwedge_{1 \leq i \leq n} (x_{i,j_1} = x_{i,j_2} = \ldots = x_{i,j_{l_i}})$$

Let the Boolean function f, defined on the variables of $X \cup \{y\}$, be

$$f = f_0 \oplus y \cdot f_1$$

It is easy to verify that λ is satisfiable iff

$$\begin{aligned} & f_0^{-1}(1) \cap f_1^{-1}(1) \neq \emptyset \\ \Leftrightarrow\ & |f^{-1}(1)| < 2 \cdot |f_0^{-1}(1)| + |f_0^{-1}(1)| \\ \Leftrightarrow\ & |f^{-1}(1)| < 2 \cdot 2^n + 7^m \end{aligned}$$

Given the CNF λ, an FFDD for f can easily by constructed in polynomial time. And this FFDD has less than $2 \cdot 2^n + 7^m$ satisfying assignments iff λ is satisfiable. □

3 Relation between BDDs and FDDs

In this section we will establish a close relation between the functions f_G^{BDD} and f_G^{FDD} for complete DDs G. For an intuitive approach to this problem, we will first give a graph-theoretic interpretation of f_G^{BDD} and f_G^{FDD}.

Fix a DD G and an assignment $a = (a_1, \ldots, a_n)$ to the variables $x_1, \ldots, x_n$. Let v be any nonterminal node of G labeled with a variable x_i. We say that the edge $(v, low(v))$ (*resp.* $(v, high(v))$) is *BDD-admissable* iff $a_i = 0$ (*resp.* $a_i = 1$). The edge $(v, high(v))$ is *FDD-admissable* iff $a_i = 1$ whereas the edge $(v, low(v))$ is always *FDD-admissable*. A path in G is called *BDD-admissable* (*resp. FDD-admissable*) iff it leads from the root to a terminal node and contains only BDD-admissable (*resp.* FDD-admissable) edges.

Obviously, there is exactly one BDD-admissable path in G and this path leads to a terminal node labeled by $f_G^{\mathrm{BDD}}(a)$. Unfolding G by inverse (t_1)-reductions to a tree does not change f_G^{FDD} or the FDD-admissable paths and shows that $f_G^{\mathrm{FDD}}(a) = 1$ iff the number of FDD-admissable paths

leading to a terminal node labeled 1 is odd. Furthermore, if G is complete, each FDD-admissable path for $(a_1, \ldots, a_n)$ corresponds exactly to a BDD-admissable path for an assignment $(b_1, \ldots, b_n) \leq (a_1, \ldots, a_n)$. Thus, $f_G^{\mathrm{FDD}}(a) = \bigoplus_{b \leq a} f_G^{\mathrm{BDD}}(b)$.

In general, define the transformation[1] τ by

$$\tau(f)(x) = \bigoplus_{y \leq x} f(y)$$

for $f \in \mathbf{B}_n$. A straightforward computation shows, that τ is its own inverse and thus bijective. Altogether, we then conclude

Theorem 2 For each complete DD G it holds that $f_G^{\mathrm{FDD}} = \tau\left(f_G^{\mathrm{BDD}}\right)$ and $f_G^{\mathrm{BDD}} = \tau\left(f_G^{\mathrm{FDD}}\right)$.

Since theorem 2 is valid only for complete DDs we have to provide a transformation from the DDs we are mainly interested in, *e.g.* ordered and free DDs, to equivalent complete DDs. Fortunately, the size of the DDs does not grow too much by this transformation:

Lemma 2 Each free DD G over the set of variables X_n can be transformed into a $(t_{\{2\}})$-equivalent complete DD G_1 and into a $(t_{\{1,3\}})$-equivalent complete DD G_2. If G is ordered, then G_1 and G_2 can be made to respect the same order of variables as G. In each of these transformations the size of the DD grows at most by a factor $O(n)$.

Proof: First, we give the idea how to convert a free DD G into a complete DD G_2 applying inverse (t_3)-reductions and (t_1)-reductions: Assign to each node v of G a set of variables $A_v \subseteq X_n$ so that $x_k \in A_v$ iff v is a terminal node or there is a predecessor of v in G (including v itself) labeled with x_k. Let (u, v) be an edge of G where v is labeled with x_j. Then, if $A_v \neq A_u \cup \{x_j\}$, new nodes labeled with the variables in $A_v \setminus (A_u \cup \{x_j\})$ have to be inserted on the edge (u, v). And, additionally, the outgoing *high*-edges of the inserted nodes are collected in a chain consisting of one node for each variable (except one) in $X_n \setminus A_u$.

The same transformation can be applied to DDs ordered with respect to some order π of the variables. The only difference is that for a node v labeled with x_j we set $A_v = \{x_i | x_i \leq_\pi x_j\}$.

The construction of $(t_{\{2\}})$-equivalent DDs is even simpler. Instead of applying an inverse (t_3)-reduction in the procedure above, we apply an inverse (t_2)-reduction and the additional chains can be ommitted. □

Combining theorem 2 with the previous lemma yields:

Theorem 3 For each Boolean function $f \in \mathbf{B}_n$ and each order π on X_n:

$$\begin{array}{rcccl}
\Omega(1/n) \cdot \text{OBDD-SIZE}_\pi(\tau(f)) & \leq & \text{OFDD-SIZE}_\pi(f) & \leq & O(n) \cdot \text{OBDD-SIZE}_\pi(\tau(f)) \\
\Omega(1/n) \cdot \text{OBDD-SIZE}(\tau(f)) & \leq & \text{OFDD-SIZE}(f) & \leq & O(n) \cdot \text{OBDD-SIZE}(\tau(f)) \\
\Omega(1/n) \cdot \text{FBDD-SIZE}(\tau(f)) & \leq & \text{FFDD-SIZE}(f) & \leq & O(n) \cdot \text{FBDD-SIZE}(\tau(f))
\end{array}$$

4 Transfering Results on BDDs to FDDs

By theorem 3 it is possible to transfer results about BDDs to FDDs and vice versa. Of course, BDD representations for a family F of Boolean functions directly gives FDD representations for the family

[1] The transformation was also used in [Raz86] (see also [Pat86]) to analyse circuits over $\{\wedge, \oplus\}$ and in [BT92] (as the Reed-Muller Transform) to synthesize two level circuits.

$\tau(F) := \{\tau(f) | f \in F\}$. An exponential gap between any two of the BDD representations considered in the previous theorem directly translates into an exponential gap between the corresponding FDD representations. If the gap for BDDs occurs for a family F of Boolean functions, then the corresponding gap for FDDs occurs for $\tau(F)$.

We start with rather simple, but useful applications of this concept. Afterwards exponential gaps between FDDs and BDDs are derived. We finish this section with exponential lower bounds for FDDs.

Consider $f = (x_1x_2 + .. + x_{2n-1}x_{2n}) \wedge \bigwedge_{(i \ mod \ 2)=1}(x_i = x_{i+1})$. A straightforward computation shows that $f = \tau(x_1x_2 + .. + x_{2n-1}x_{2n})$. From [Bry86] and theorem 2 we conclude that there are two orders π and π' of X_{2n} so that OFDD-SIZE$_\pi(f) = O(n^2)$ but OFDD-SIZE$_{\pi'}(f) = \Omega(\frac{1}{n}2^n)$, i.e. variable ordering is important.

In [Bry91] it is shown that the *hidden weighted bit* function (HWB) (which will be investigated in more detail in subsection 4.2) requires OBDDs of exponential size, but it is known that there are polynomial size FBDDs for HWB [SW92a]. Thus, τ(HWB) is a function with polynomial size FFDDs but only exponential size OFDDs.

Now we show that OFDDs are suitable for the representation of symmetric functions as well as OBDDs. This can be concluded as follows: A complete OBDD of size $n(n-1)/2$ for a symmetric function in n variables is easy to construct (see [Bry86]). Furthermore, it follows from the definition of τ that the family of symmetric functions in n variables is invariant under τ. Thus, we have

Theorem 4 Each symmetric function of n variables can be represented by an OFDD of size $n(n-1)/2$ for any order of the variables.

In fact, the upper bound on the OBDD-size of symmetric functions can be improved to $n^2/2 - \Theta(n \log n)$ [Weg84] and this bound also applies to OFDDs.

4.1 Exponential Gaps between FDDs and BDDs

In the sequel it is shown that there exist functions, for which OFDDs (*resp.* OBDDs) are exponentially better than OBDDs and FBDDs as well (*resp.* OFDDs and FFDDs as well).

To prove this, we consider the clique-function $\oplus$-$cl_{n,3}$ defined over $n(n-1)/2$ variables x_{ij} $(1 \leq i < j \leq n)$. Let $\oplus$-$cl_{n,3}(x) = 1$, iff the undirected graph $G(x) = (V, E)$, with $V = \{1, .., n\}$ and $E = \{\{i, j\} \mid x_{ij} = 1\}$ contains an odd number of 3-cliques. It has been proven in [ABH+86] that each FBDD representing $\oplus$-$cl_{n,3}$ requires size $2^{\Omega(n^2)}$.

The above definition of $\oplus$-$cl_{n,3}$ can naturally be reformulated in terms of a τ-transformation: $\oplus$-$cl_{n,3}(x) = 1$ iff $G(x)$ contains an odd number of subgraphs $G(y)$ consisting only of a single 3-clique and $n-3$ isolated nodes. Thus, we have $\oplus$-$cl_{n,3}(x) = \bigoplus_{y \leq x} 1$-$cl_{n,3}(y) = \tau(1\text{-}cl_{n,3})(x)$, where 1-$cl_{n,3}(x) = 1$ iff $G(x)$ consists only of a single 3-clique and $n-3$ isolated nodes.

Comparing the representations by free DDs we show that 1-$cl_{n,3}$ can efficiently be described only using the BDD interpretation and $\oplus$-$cl_{n,3}$ only with the FDD interpretation.

Theorem 5 For any order π of the variables x_{ij} $(1 \leq i < j \leq n)$ it holds:

1. OBDD-SIZE$_\pi$ (1-$cl_{n,3}$) $= O(n^5)$, but FFDD-SIZE (1-$cl_{n,3}$) $= 2^{\Omega(n^2)}$.

2. OFDD-SIZE$_\pi$ ($\oplus$-$cl_{n,3}$) $= O(n^3)$, but FBDD-SIZE ($\oplus$-$cl_{n,3}$) $= 2^{\Omega(n^2)}$.

Proof: We first prove the upper bounds. Let T be the ordered DD obtained by labeling the complete tree of height $n(n-1)/2$ according to the order π and labeling the $2^{n(n-1)/2}$ leaves of T according to the values of 1-$cl_{n,3}$. Then $f_T^{\mathrm{BDD}} = 1\text{-}cl_{n,3}$ and $f_T^{\mathrm{FDD}} = \oplus\text{-}cl_{n,3}$.

Applying (t_1)- and (t_2)-reductions to T, each nonterminal node v for that the subtree rooted at v contains no leaf labeled 1 can be deleted. Since T has only $\binom{n}{3}$ leaves labeled 1 and each leaf has $n(n-1)/2$ predecessors, the $(t_{\{1,2\}})$-reduction of T is of size at most $\frac{1}{2}n(n-1)\binom{n}{3}$.

Analogously, by repeated applications of (t_3)-reductions to T each nonterminal node v can be deleted for that the subtree rooted at $high(v)$ contains no leaf labeled 1. Thus, if a node v of T is still present in the $(t_{\{3\}})$-reduction of T, the *high*-edge leaving v lies on a path from the root of T to a leaf labeled 1. Since each path from the root of T to a leaf labeled 1 contains exactly 3 *high*-edges, the size of the $(t_{\{3\}})$-reduction of T is at most $3\binom{n}{3}$.

The lower bound for FBDD-SIZE ($\oplus$-$cl_{n,3}$) was derived in [ABH+86] and the lower bound for FFDD-SIZE (1-$cl_{n,3}$) is an application of theorem 3. □

For OBDDs it is well-known that synthesis operations can be performed efficiently. Given two DDs G_1 and G_2 respecting the same order of variables an ordered DD G with $f_G^{\mathrm{BDD}} = f_{G_1}^{\mathrm{BDD}} \circ f_{G_2}^{\mathrm{BDD}}$ can be constructed in $O(|G_1|\,|G_2|)$ steps for each binary Boolean operator $\circ$. The EXOR-synthesis and the negation can also be performed efficiently for OFDDs ([BDT93]). On the other hand, the following theorem shows that the AND- and the OR-synthesis of OFDDs can not be computed in polynomial time since the resulting OFDD may have exponential size. In fact, even changing the order of variables or allowing the result to be any FFDD does not make the synthesis operation easier.

Theorem 6 There is a family of Boolean functions $(f_n)_{n\in\mathbf{N}}$ with FFDD-SIZE$(f_n) = 2^{\Omega(n)}$ so that each f_n is the product of two functions represented by OFDDs of polynomial size for any order of the variables.

Proof: For the proof it is sufficient to construct two functions with polynomial OFDDs for any order π whose product is 1-$cl_{n,3}$. The first of these two functions is $\oplus$-$cl_{n,3}$. The other one is the inverted threshold function $T^{\leq 3}_{n(n-1)/2}$ that computes 1 iff at most 3 of the x_{ij}'s $(1 \leq i < j \leq n)$ have value 1. Since $T^{\leq 3}_{n(n-1)/2}$ is a symmetric function it can be represented by an OFDD of polynomial size (theorem 4). Obviously, the assertion of the theorem follows from 1-$cl_{n,3} = \oplus\text{-}cl_{n,3} \cdot T^{\leq 3}_{n(n-2)/2}$ □

4.2 Exponential Lower Bounds for OFDDs

In this subsection we prove that the two Boolean functions for which Bryant gave exponential lower bounds on the OBDD-SIZE also require exponentially large OFDDs. In [Bry91] lower bounds for the size of OBDDs have been proven using lower bound techniques from VLSI design. We now show how to adapt these arguments to OFDDs.

We first consider the *hidden wighted bit function* (HWB) defined as follows:

HWB : For $x = (x_1, x_2, \ldots, x_n)$ we define the weight $wt(x)$ of a vector x as the number of variables set to 1. Then $HWB : \mathbf{B}^n \to \mathbf{B}$ is given by:

$$HWB(x) = \begin{cases} 0, & wt(x) = 0 \\ x_{\mathrm{wt}(x)}, & wt(x) > 0 \end{cases}$$

We shortly review the argument of [Bry91] giving a lower bound on OBDD-SIZE(f) for $f : \mathbf{B}^n \to \mathbf{B}$: Let (L, R) be a partition of X_n. Then, any input assignment $x : X_n \to \mathbf{B}$ can be split into a *left input assignment* $l : L \to \mathbf{B}$ and a *right input assignment* $r : R \to \mathbf{B}$. We use the notation $x = lr$. A set $\mathbf{F}_{(L,R)}$ of left input asignments is called a *fooling set* for the partition (L, R) iff for each two distinct $l, l' \in L$ there is a right input assignment r with $f(lr) \neq f(l'r)$. Fixing a real parameter $\omega \in (0, 1)$ and a subset $Y \subseteq X_n$ a partition (L, R) is called a *balanced partition* iff $\lfloor \omega \cdot |Y| \rfloor \leq |Y \cap L| \leq \lceil \omega \cdot |Y| \rceil$.

The main tool in [Bry91] for proving lower bounds on the size of OBDDs is:

Lemma 3 ([Bry91]) If, for each balanced partition (L, R) (with respect to some ω and Y), f has a fooling set $\mathbf{F}_{(L,R)}$ of size c, then OBDD-SIZE(f) $\geq c - 2$.

We now prove:

Theorem 7 Any OFDD representing HWB has $2^{\Omega(n)}$ nodes.

Proof: W.l.o.g. assume that n is a multiple of 10. We show that OBDD-SIZE($\tau(HWB)$) $\geq \binom{0.2n}{0.1n} - 2$. Then, the assertion follows from theorem 3.

In general, we follow [Bry91] except in the last step, where the argument for HWB has to be modified appropriately to work for $\tau(HWB)$. Set $Y = X_n$, $\omega = 0.6$ and define

$$\begin{aligned} X_H &= \{x_{0.5n+1}, \ldots, x_{0.9n}\} \\ X_L &= \{x_{0.1n+1}, \ldots, x_{0.5n}\} \end{aligned}$$

For each balanced partition (L, R) there is a set W of $0.2\,n$ variables with $W \subseteq X_H \cap L$ (case 1) or $W \subseteq X_L \cap L$ (case 2) since $|L \cap (X_H \cup X_L)| \geq 0.4\,n$. Let $\mathbf{F}_{(L,R)}$ be the set of all left input assignments l assigning 1 to all variables of $L \setminus W$ and to $0.1\,n$ variables of W. This set contains $\binom{0.2\,n}{0.1\,n}$ assignments. It remains to prove for both cases that $\mathbf{F}_{(L,R)}$ is a fooling set for $\tau(HWB)$. We give the proof of case 1. Case 2 follows analogously:

Case 1 : $W \subset X_H$

Let $l, l' \in \mathbf{F}_{(L,R)}$ be two different input assignments for the variables in L. Choose $0.5n < i \leq 0.9n$ minimal with $l(x_i) \neq l'(x_i)$. Choose r by setting any $i - 0.5\,n$ variables of R to 1. Then

$$HWB(lr) = l(x_i) \neq l'(x_i) = HWB(l'r)$$

Thus, $\mathbf{F}_{(L,R)}$ is a fooling set for HWB. To get the result for $\tau(HWB)$ observe that

$$\begin{aligned} \tau(HWB)(lr) &= \bigoplus_{y<lr} HWB(y) \quad \oplus \quad HWB(lr) \\ \tau(HWB)(l'r) &= \bigoplus_{y<l'r} HWB(y) \quad \oplus \quad HWB(l'r) \end{aligned}$$

It remains to show that

$$\bigoplus_{y<lr} HWB(y) = \bigoplus_{y<l'r} HWB(y) \tag{1}$$

There is a permutation of the variables $x_i, \ldots, x_n$ that maps lr to $l'r$. Therefore this permutation maps all y with $y < lr$ bijectively on all y with $y < l'r$. But this permutation does not change the value of HWB since it does not change the weight and leaves $y_{wt(y)}$ fixed for $y < lr$. That shows (1) and completes the proof in case 1.

□

Since FDDs are a data structure based on the $\oplus$-operation they might be practical in the field of arithmetical functions. In this context the multiplication is especially interesting as it has no polynomial size representation by OBDDs. Denote by MULT_i^n the function mapping two n-bit numbers $a_{n-1}\dots a_1a_0$ and $b_{n-1}\dots b_1b_0$ to the i-th bit of their product. In [Bry91] it is shown that OBDD-SIZE(MULT_{n-1}^n) $= 2^{\Omega(n)}$. But again, by a slight modification of Bryant's lower bound argument the same applies to OFDDs as we will prove now.

Theorem 8 OFDD-SIZE(MULT_{n-1}^n) $= 2^{\Omega(n)}$.

Proof: By theorem 2, it is sufficient to show OBDD-SIZE($\tau(\text{MULT}_{n-1}^n)$) $= 2^{\Omega(n)}$. We briefly review the proof of [Bry91] giving the lower bound for MULT_{n-1}^n and point out how to change Bryant's argument to work for $\tau(\text{MULT}_{n-1}^n)$.

A set $\mathbf{F}$ of input assignments lr for a partition (L, R) of X_n is called a *VLSI-fooling set* for a function $f : \mathbf{B}^n \to \mathbf{B}$ iff f is a fixed constant $z \in \{0, 1\}$ on $\mathbf{F}$, but for any distinct $lr, l'r' \in \mathbf{F}$:

$$(f(l'r) \neq z) \quad \vee \quad (f(lr') \neq z)$$

The left input assignments of a VLSI-fooling set constitute a fooling set for f. Thus, applying lemma 3, the existence of a VLSI-fooling set of cardinality c for each balanced partition implies OBDD-SIZE(f) $\geq c - 2$.

We apply this method to the function $\tau(\text{MULT}_{n-1}^n)$. Let (L, R) be any balanced partition (L, R) for $Y = \{a_0, \dots, a_{n-1}\}$ and $\omega = 0.5$. Then, there are two disjoint equally sized substrings $U = u_{k-1}\dots u_0$ and $V = v_{k-1}\dots v_0$ of the a-operand so that the set $S = \{i | 1 \leq i \leq k-2, |\{u_i, v_i\} \cap L| = 1\}$ contains at least $n/8 - 2$ elements. Let $\mathbf{F}$ be the set of assignments with the following properties:

1. Assign 0 to all a_i which are not part of U or V.
2. Assign 1 to u_0 and v_0. Assign 0 to u_{k-1} and v_{k-1}.
3. For each $i \notin (S \cup \{0, k-1\})$, assign 0 to u_i and 1 to v_i.
4. For each $i \in S$ assign complementary values to u_i and v_i (there are $2^{|S|}$ possibilities).
5. If $u_0 = a_m$ and $v_0 = a_{m'}$. Assign 1 to b_{n-k-m} and $b_{n-k-m'}$ and assign 0 to all other variables b_i.

By properties 1 and 5 the value of $\text{MULT}_{n-1}^n(a, b)$ is the bit c_{k-1} of the sum $U + V = c_k \dots c_0$. From the other properties follows that $U + V = 2^{k-1}$, thus $\text{MULT}_{n-1}^n(lr) = 1 \ \forall lr \in \mathbf{F}$.

Now, let lr and $l'r'$ be two different assignments in $\mathbf{F}$. Since lr and $l'r'$ differ, they differ on a variable u_j or v_j for $j \in S$. Choose such a j maximal. By property 4:

$$lr(u_j) = \overline{lr(v_j)} = l'r'(v_j) = \overline{l'r'(u_j)}$$

W.l.o.g. assume that $lr(u_j) = 0$ and $u_j \in L$. Then

$$lr'(u_j) = l(u_j) = 0 \qquad \text{and} \qquad lr'(v_j) = r'(v_j) = 0$$

Thus, $\text{MULT}_{n-1}^n(lr') = 0$ since in the addition of U and V any carry is eliminated at the j-th position and no carry is generated to the left of that position. This proves that $\mathbf{F}$ is a VLSI-fooling

set for MULT^n_{n-1}. To achieve the result for $\tau(\mathrm{MULT}^n_{n-1})$ we show that both functions agree on lr, $l'r'$ and lr':
Let y be any assignment with $y < lr$ or $y < l'r'$ or $y < lr'$. If y sets only a single variable b_i to 1 then clearly $\mathrm{MULT}^n_{n-1}(y) = 0$. Otherwise $\mathrm{MULT}^n_{n-1}(y)$ is again the bit c_{k-1} of the sum $U + V = c_k \ldots c_0$. But this sum must be less than 2^{k-1}, thus $\mathrm{MULT}^n_{n-1}(y) = 0$.

□

The result can be applied to each output of the n-bit mulitplier, since an i-bit multiplier can be embedded in an n-bit multiplier.

Corollary 1 Any OFDD representing MULT^n_{i-1} and MULT^n_{2n-i-1}, for $1 \leq i \leq n$, has $\Omega(2^{\frac{i}{8}})$ nodes.

5 Conclusions and Open Problems

We investigated functional decision diagrams (FDDs) and their relation to binary decision diagrams (BDDs). For complete DDs G we established a close relation between the BDD-function f^{BDD}_G and the FDD-function f^{FDD}_G. The relation is given by a Boolean transformation τ, also called the Reed-Muller Transform. Taking advantage of some nice properties of τ we could derive several structural similarities between OFDDs and OBDDs.

Investigating the functions $\oplus$-$cl_{n,3}$ and 1-$cl_{n,3}$ we showed that FDDs might be appropriate for functions where an efficient BDD representation does not exist, and vice versa. Subsequently, we proved that for OFDDs the AND-operation (OR-operation) may lead to an exponential blow-up, while the EXOR-operation and negation still can be performed efficiently. This is a drawback of OFDDs compared to OBDDs, at least from the theoretical point of view. Experiments have to be performed to answer the question whether this property becomes relevant for practical circuits.

Finally, we demonstrated that the lower bound techniques for OBDDs can be adapted to OFDDs by proving that the hidden weighted bit function (HWB) and multiplication as well require OFDDs of exponential size independent of the ordering of the variables. It is commonly believed that EXOR-based circuits are well suited for arithmetical functions. In the case of multiplication this is not true at least for OFDDs. It might still be true for FFDDs.

If FFDDs and OFDDs are to be applied in design automation, efficient representation and manipulation algorithms for concrete functions will be required. Thus, at least heuristics for good orderings are needed. First experiences indicate that the methods known from BDDs might not work satisfactorily. If heuristics tailored to FDDs have been developed, a fair comparison between the optimized OBDD- and OFDD-size of realistic functions can be carried out. Of course, also manipulation of FDDs and BDDs has to be compared for practical examples. As mentioned above, the complexity of the AND-operation offers a major problem for FDDs. Also the feasibility of other desirable operations has to be studied. As an example consider the computation satisfiability count, i.e. the computation of the number of satisfying assignments: It is possible in linear time for FBDDs, while it can be shown to be NP-hard for FFDDs. It is open, whether this is also true for OFDDs.

References

[ABH+86] M. Ajtai, L. Babai, P. Hajnal, J. Komlos, P. Pudlak, V. Rödl, E. Szemeredi, and G. Turan. Two lower bounds for branching programs. In *Proceedings of 18th ACM STOC*, pages 30–38, 1986.

[Ake78] S.B. Akers. Binary decision diagrams. *IEEE Trans. on Comp.*, C-27:509–516, 1978.

[BDT93] 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 Applications of the Reed-Muller Expansion in Circuit Design, Hamburg*, 1993.

[Ber68] E. R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill Book Company, 1968.

[Bry86] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 8:677–691, 1986.

[Bry91] 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.

[Bry92] R.E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM, Comp. Surveys*, 24:293–318, 1992.

[BT92] Ph.W. Besslich and E.A. Trachtenberg. A three-valued quasi-linear transformation for logic synthesis. In C. Moraga and R. Creutzburg, editors, *Spectral Techniques: Theory and Applications.* Elsevier, North Holland, 1992.

[DB93] R. Drechsler and B. Becker. Rapid prototyping of fully testable multi-level and/exor networks. *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design, Hamburg*, 1993.

[GM92] J. Gergov and C. Meinel. *Analysis and Manipulation of Boolean Functions in Terms of Decision Graphs*, volume 657 of *LNCS.* Symp. on Theoretical Aspects of Comp. Science, 1992.

[KR93] U. Kebschull and W. Rosenstiel. Efficient graph-based computation and manipulation of functional decision diagrams. In *Proceedings of EDAC'93*, pages 278–282, 1993.

[KSR92] U. Kebschull, E. Schubert, and W. Rosenstiel. Multilevel logic based on functional decision diagrams. In *Proceedings of EDAC'92*, pages 43–47, 1992.

[Lee59] C.Y. Lee. Representation of switching circuits by binary decision diagrams. *Bell System Technical Jour.*, 38:985–999, 1959.

[LS92] Y.-T. Lai and S. Sastry. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Proceedings of 29th Design Automation Conference*, pages 608–613, 1992.

[Mei89] C. Meinel. *Modified Branching Programs and their Computational Power*, volume 370 of *LNCS.* Springer Verlag, 1989.

[Pat86] M.S. Paterson. On Razborov's result for bounded depth circuits over $\{\oplus, \wedge\}$. Technical report, University Warwick, 1986.

[PH91] D. Pellerin and M. Holley. *Practical Design Using Programmable Logic.* Prentice-Hall, New Jersey, 1991.

[Raz86] A.A. Razborov. Lower bounds on the size of bounded depth networks over the basis $\{\oplus, \wedge\}$. Technical report, Moscow State University, Moscow, 1986.

[Red72] S. M. Reddy. Easily testable realizations for logic functions. In *IEEE Transaction on Computers*, volume c-21, pages 1183–1188, 1972.

[Sas93] T. Sasao. *Logic Synthesis and Optimization*. Kluwer Academic Publisher, 1993.

[Sau92] J. Saul. Logic synthesis for arithmetic circuits using the reed-muller representation. In *Proceedings of EDAC'92*, pages 109–113, 1992.

[SKR92] E. Schubert, U. Kebschull, and W. Rosenstiel. FDD based technology mapping for FPGA. In *Proceedings of EUROASIC*, pages 14–18, 1992.

[SW92a] D. Sieling and I. Wegener. Graph driven BDDs - a new data structure for Boolean functions. Technical report, Universität Dortmund, 1992.

[SW92b] D. Sieling and I. Wegener. Reduction of BDDs in linear time. Technical report, Universität Dortmund, 1992.

[Weg84] I. Wegener. Optimal decision trees and one-time-only branching programs for symmetric Boolean functions. *Information and Control*, 62:129–143, 1984.

[Weg87] I. Wegener. *The Complexity of Boolean Functions*. John Wiley & Sons Ltd., and B.G. Teubner, Stuttgart, 1987.

[WP92] W. Wan and M. A. Perkowski. A new approach to the decomposition of imcompletely specified multi-output functions based on graph coloring and local transformations and its application to fpga mapping. In *European Design Automation Conf.*, pages 230–235, 1992.

[Xil88] Xilinix Inc. *The Programmable Gate Array Data Book*. , 1988.