PARTITIONED BDDs VS. OTHER BDD MODELS

Beate Bollig* and Ingo Wegener*
bollig/wegener@ls2.informatik.uni-dortmund.de

FB Informatik, LS2, Univ. Dortmund, 44221 Dortmund, Germany

Abstract. Ordered binary decision diagrams (OBDDs) have found a lot of applications in the verification of combinational and sequential circuits, protocols, and the synthesis and analysis of systems. The applications are limited, since the expressive power of polynomial-size OBDDs is too restricted. Therefore, several more general BDD models are used. Partitioned OBDDs are an OBDD model allowing a restricted use of nondeterminism and different variable orderings. They are restricted enough such that the essential operations can be performed efficiently and they allow polynomial-size representations for many more functions than OBDDs. Here the expressive power of polynomial-size partitioned OBDDs is investigated.

1 INTRODUCTION

For verification we like to prove that a design meets the specification. Then we are interested in representations fulfilling many requirements. Obviously small size for many Boolean functions is an essential issue. In order to transform a formal specification (perhaps a circuit) into the representation it is necessary to have efficient algorithms to combine Boolean functions in the given representation by Boolean operators (like conjunction etc.). Moreover, one needs efficient algorithms to check whether two representations describe the same function. In order to describe our results and the organization of our paper we first define the different models of binary decision diagrams.

Definition 1. A binary decision diagram (BDD) is a directed acyclic graph with one source. Each sink is labeled by a Boolean constant and each other node by a Boolean variable from $\{x_1, \ldots, x_n\}$. These nodes have two outgoing edges one labeled by 0 and the other by 1. The BDD represents the Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ defined in the following way. An input $a \in \{0,1\}^n$ activates all edges consistent with a, i.e., the edges labeled by a_i which leave nodes labeled by x_i. The value of the sink reached by the unique path which starts at the source and is activated by a is defined as $f(a)$.

Definition 2. (i) A free BDD (FBDD) is a BDD where each path contains for each variable x_i at most one node labeled by x_i.
(ii) An ordered BDD (OBDD) is an FBDD where the orderings of the variables on all paths are consistent with one ordering.
(iii) A k-OBDD consists of k layers of OBDDs respecting the same ordering.
(iv) A k-IBDD consists of k layers of OBDDs respecting perhaps different orderings.

The size of a BDD G is the number of its nodes and is denoted by $|G|$.

OBDDs introduced by Bryant (1986) are the most popular representation for formal circuit verification. Gergov and Meinel (1994) and Sieling and Wegener (1995) independently have shown how FBDDs can be used for verification but it is difficult to find a good type or oracle graph. In k-OBDDs and k-IBDDs the condition that variables are tested only once is relaxed. For k-OBDDs and constant k the test whether two decision diagrams represent the same function can be performed in polynomial time. This problem is coNP-complete already for 2-IBDDs. On the other hand k-IBDDs allow a more compact representation of many functions but the equality test only is based on heuristic algorithms. Bollig, Sauerhoff, Sieling, and Wegener (1997) present results on the representational power of these types of BDDs and Jain, Abadir, Bitner, Fussell, and Abraham (1992) have performed and described experiments.

Even BDD models allowing restricted nondeterminism can be handled efficiently. Jain (1993) and Jain, Bitner, Fussell, and Abraham (1992) have introduced partitioned BDDs and have already described first experiments with these types of BDDs. More results on partitioned BDDs are presented by Narayan, Jain, Fujita, and Sangiovanni-Vincentelli (1996). Waack (1997) has shown how one can work with $\oplus$-OBDDs.

* Supported in part by DFG grant We 1066/7.

Definition 3. A $\oplus$-OBDD is a generalized OBDD where the number of edges leaving a node is not restricted. The output value for an input a is defined as 1 iff the number of paths between the source and the 1-sink activated by a is odd.

Definition 4. A k-PBDD (partitioned BDD with k parts where k may depend on the number of variables) consists of k OBDDs whose variable orderings may be different. The output value for an input a is defined as 1 iff at least one of the OBDDs computes 1 on a. A PBDD is a k-PBDD for some k. The size of a k-PBDD is the sum of the sizes of the k OBDDs.

The model of k-PBDDs in its general form has several practical drawbacks. Even if the k variable orderings are fixed, it is difficult to check whether two k-PBDDs G' and G'' represent the same function. An input a may lead in the i-th part of G' to 1 but only in the j-th part of G'' where $j \neq i$. For practical purposes it is necessary to work with so-called window functions $w_1, \ldots, w_k$ whose disjunction is the constant 1. The i-th part G_i of a k-PBDD G for f is responsible for the part $w_i^{-1}(1)$, i.e., G_i has to represent the function $f_i := f \wedge w_i$. This allows an efficient equality check.

In certain applications like combinatorial problems it is necessary to count the number of inputs satisfying a function f. This is efficiently possible for OBDDs but not for k-PBDDs, since many parts may compute 1 for an input a. The window functions are called disjoint, if $w_i \wedge w_j = 0$ for all $i \neq j$. Then the number of satisfying inputs is the sum of the number of satisfying inputs for the parts. Only in this case the notion "partitioned" is really justified. This kind of nondeterminism is known in complexity theory as unique nondeterminism. We use the notions k-PBDD^w and k-PBDD^d, if we work with fixed window functions resp. fixed disjoint window functions.

In the presence of so many different models one should ask whether we need them all, i.e., whether there are problems (functions) where one model is superior to all the other. In Section 2 we collect the properties of the different BDD models. In Section 3 it is shown that we restrict PBDDs if we ask for window functions. In Section 4 we discuss how the number of parts used in a PBDD influences its expressive power. Finally, we compare in Section 5 the expressive power of PBDDs with the expressive power of the other mentioned BDD models.

2 PROPERTIES OF THE DIFFERENT MODELS

Using BDDs we have a trade-off between expressive power and efficiency and simplicity of operations. As important operations we restrict ourselves to synthesis of two BDDs by a Boolean operation (SYN), reduction to minimal size (RED), satisfiability test (SAT), equality check (EQU), and counting the number of satisfying inputs (COU). Generalized models differ from OBDDs by allowing repeated tests of variables (REP), nondeterminism (NON), and/or the use of different variable orderings (ORD). In Table 1 we list the properties of the BDD models.

The results for FBDDs only hold if the oracle graph or type is fixed. The meaning of the abbreviations is the following:

- impl.: different orderings only can be used implicitly.
- layer: only the layers can be reduced, not the whole k-OBDD/k-IBDD.
- pol.: polynomial time but less efficient than for OBDDs.
- heur.: no polynomial time algorithm, only heuristic algorithms.
- $+/\oplus$ and +/OR: the acceptance mode of the nondeterministic models.

Since k-OBDDs can be manipulated in polynomial time only for constant k, we restrict ourselves to k-OBDDs and k-IBDDs with a constant number of layers.

	SYN	RED	SAT	EQU	COU	REP	NON	ORD
FBDD	+	+	+	+	+	−	−	impl.
k-OBDD	+	layer	pol.	pol.	pol.	+	−	−
k-IBDD	+	layer	heur.	heur.	−	+	−	+
$\oplus$-OBDD	+	pol.	pol.	pol.	−	−	$+/\oplus$	−
PBDD	−	−	+	−	−	−	+/OR	+
PBDD^w	+	+	+	+	−	−	+/OR	+
PBDD^d	+	+	+	+	+	−	+/OR	+

Table 1. Comparison of BDD models

3 THE DIFFERENT MODELS OF PARTITIONED BDDs

Table 1 shows that it is much easier to work with PBDD^{w}s, than with the more general PBDDs. Do we loose expressive power if we restrict ourselves to PBDD^{w}s? The answer is affirmative. (The examples in our paper distinguish between polynomial and exponential size. Therefore, we have to work with sequences of functions and we cannot work with benchmark circuits with a fixed number of inputs.) A permutation matrix contains in each row and in each column exactly one entry 1. The test NPERM whether a matrix X is *not* a permutation matrix easily can be performed by a 2-PBDD of linear size but the test PERM whether X is a permutation matrix needs exponential size for PBDDs and even partitioned FBDDs (Krause, Meinel and Waack (1991)). This implies that negation may lead to an exponential blow-up for general PBDDs which, therefore, are not appropriate for applications. Negations can be performed efficiently for PBDD^{w}s. Hence, NPERM needs exponential-size PBDD^{w}s.

We conclude that the restriction to PBDD^{w}s is necessary to be able to work efficiently with partitioned OBDDs, although we loose expressive power.

4 HOW MANY PARTS DO WE NEED FOR PBDDs?

If we work with PBDD^{w}s the set of window functions has to be fixed. Is the number of parts a crucial parameter?

It is easy to see that we efficiently can reduce the number of parts by each constant factor if all parts use the same variable ordering. More precisely, if f has polynomial-size PBDD^{w}s with p parts and a fixed variable ordering, then f has for each constant c polynomial-size PBDD^{w}s with $\lceil p/c \rceil$ parts and the same variable ordering.

The situation changes if we use the full power of PBDD^{w}s, namely different variable orderings in the different parts. We only state the idea of the construction of our function $P_{k,n}$. First we have chosen a function with polynomial-size OBDDs but with the additional property that OBDDs may have small size only if a certain set of variables is tested at the beginning. Then we consider k variants of this function where for each variant another set of variables has to be tested first in small size OBDDs. Then these k variants are combined by a multiplexer. For this function we can prove the following:

- It has polynomial-size PBDD^{d}s (the weakest model) with k parts.
- PBDDs (even without window functions) need exponential size if the number of parts is bounded by $k-1$.

This result does not hold only for all constants k but even if k is increasing slowly with the number of variables.

We conclude that the choice of the number of parts (and the corresponding window functions) is crucial for small size PBDD^{w}s. Moreover, we have obtained a lower bound technique to determine sometimes the minimal number of parts for small size PBDD^{w}s.

5 PARTITIONED BDDs VS. OTHER BDD MODELS

There are three functions with exponential OBDD size which are well studied. The hidden weighted bit function HWB (Bryant (1991)) computes x_{sum} where sum is the number of ones in the input. The indirect storage access function ISA works in two multiplexer phases. First the control vector c of length $k = \log n$ is used to choose a position pos in the data vector x. Then the k positions in x starting at pos (if necessary mod n) are used as control vector of a second multiplexer choosing a position pos^* and the output is x_{pos^*}. The functions HWB and ISA are easy for all generalized BDD models. They have polynomial size FBDDs (Sieling and Wegener (1997)), 2-OBDDs, 2-IBDDs, $\oplus$-OBDDs, and PBDD^{d}s. On the other hand multiplication has exponential size for FBDDs (Ponzio (1995)), k-OBDDs, $\oplus$-OBDDs (Dietzfelbinger and Savický (1997)), and for k-IBDDs and PBDDs with k parts and constant k which follows from a result of Gergov (1994) for more general BDDs. Hence, these typical examples cannot be used to distinguish the generalized BDD models.

The function $P_{k,n}$ from Section 4 has further interesting properties. It has polynomial size representations not only for k-PBDD^{d}s but also for 2-OBDDs, FBDDs, and $\oplus$-OBDDs. Nevertheless, this function needs exponential size for $(k-1)$-PBDDs. Another function is only easy for PBDDs. For an $n \times n$ Boolean matrix X we have to output 1 iff the matrix X contains an odd number of ones and a row consisting of ones only or an even number of ones and a column consisting of ones only. It is easy to see that this function can be represented in linear size with 2-PBDD^{d}s using the window functions deciding whether the number of ones in the output is even or odd. With lower bound techniques due to Krause (1991) and Dietzfelbinger and Savický (1997) we have shown that this function needs exponential size for FBDDs, k-OBDDs, and $\oplus$-OBDDs.

One may ask whether PBDDs where all parts are based on the same variable ordering are useful. Then the function XOR-HWB computing the $\oplus$-sum of all x_i where $i \leq sum$ (see above) can be represented in polynomial size. With the algorithm of Sieling and Wegener (1995) for the HWB function it can be shown that this function also can be represented by FBDDs of polynomial size. We present another example. It is a variant of a multiplexer. The input $x = (x_0, \ldots, x_{n-1})$ is partitioned to $k = \log n$ blocks $X_1, \ldots, X_k$. Let s^2 be the largest square not larger than n/k. Then we arrange s^2 variables of each block as $s \times s$-matrix. The i-th bit of the control vector equals 1 iff the i-th matrix contains in each row at least a 1. This leads to a control vector of length k which together with x is the input of a multiplexer. This function has the following properties:

- PBDD^ds with n parts can represent the function with the same variable ordering (blockwise and in each block rowwise) in polynomial size.
- The i-th window function is responsible for the i-inputs where the control vector takes the value $i-1$.
- FBDDs need exponential size 2^{s-1}, since they have to start with a complete binary tree of depth $s-1$.

We remark that functions with polynomial-size PBDD^ds using only one variable ordering have also $\oplus$-OBDDs of polynomial size.

Finally, we compare PBDDs with k-OBDDs and k-IBDDs. We can prove that functions with polynomial-size k-OBDDs and constant k have also PBDD^ds of polynomial size. On the other hand each PBDD with k parts can be replaced by a k-IBDD of the same size. One may conclude that PBDD^ws are the most general submodel of IBDDs which can be handled efficiently.

CONCLUSION

Partitioned BDDs are special cases of IBDDs which can be handled much more efficiently. The importance of partitioned BDDs has been proved by experiments (Jain et al. (1992) and Narayan et al. (1996)) and now also theoretically. Partitioned BDDs of polynomial size form a tight hierarchy with respect to the number k of their parts. There are functions which can be handled efficiently already with 2-PBDDs but not with FBDDs, $\oplus$-OBDDs, or k-OBDDs. The other models still have the right to exist, since there are functions which can be handled efficiently by them but not by k-PBDDs for small k.

References

Bollig, B., Sauerhoff, M., Sieling, D., and Wegener, I.: Hierarchy theorems for kOBDDs and kIBDDs. Accepted for publication in Theoretical Computer Science (1997). (Preliminary version: On the power of different types of restricted branching programs. Electronic Colloquium on Computational Complexity. TR 94-026).

Bryant, R. E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Computers **35** (1986), 677 – 691.

Bryant, R. E.: On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication. IEEE Trans. on Computers **40** (1991), 205 – 213.

Dietzfelbinger, M. and Savický, P.: Parity OBDDs cannot represent the multiplication succinctly. Manuscript, Univ. Dortmund (1997).

Gergov, J.: Time-space trade-offs for integer multiplication on various types of input oblivious sequential machines. Information Processing Letters **51** (1994), 265 – 269.

Gergov, J. and Meinel, C.: Efficient Boolean manipulation with OBDDs can be extended to FBDDs. IEEE Trans. on Computers **43** (1994), 1197 – 1209.

Jain, J.: On Analysis of Boolean Functions. Ph. D. Dissertation, Dept. of Electrical and Computer Engineering, University of Texas at Austin, (1993).

Jain, J., Abadir, M., Bitner, J., Fussell, D. S., and Abraham, J. A.: IBDDs: an efficient functional representation for digital circuits. Proc. European Design Automation Conference (1992), 440 – 446.

Jain, J., Bitner, J., Fussell, D. S., and Abraham, J. A.: Functional partitioning for verification and related problems. Brown/MIT VLSI Conference (1992), 210 – 226.

Krause, M.: Lower bounds for depth-restricted branching programs. Information and Computation **91** (1991), 1 – 14.

Krause, M., Meinel, C., and Waack, S.: Separating the eraser Turing machine classes L_e, NL_e, co-NL_e and P_e. Theoretical Computer Science **86** (1991), 267 –275.

Narayan, A., Jain, J., Fujita, M., and Sangiovanni-Vincentelli, A.: Partitioned ROBDDs - a compact, canonical and efficiently manipulable representation for Boolean functions. Proc. of IEEE/ACM Int. Conf. on Computer Aided Design ICCAD'96 (1996), 547 – 554.

Ponzio, S.: A lower bound for integer multiplication with read-once branching programs. Proc. of 27th Annual Symp. on Theory of Computing STOC'95 (1995), 130 – 139.

Sieling, D. and Wegener, I.: Graph driven BDDs - a new data structure for Boolean functions. Theoretical Computer Science **141** (1995), 283 – 310.

Waack, S.: On the descriptive and algorithmic power of parity ordered binary decision diagrams. Proc. of STACS'97, LNCS 1200 (1997), 201 – 212.