Published at Workshop on Post Binary Ultra-Large Scale Integration (ULSI'96), Santiago de Compostela

A Note on Symbolic Simulation using Decision Diagrams

Rolf Drechsler Andreas Hett Bernd Becker

Institute of Computer Science
Albert-Ludwigs-University
79110 Freiburg im Breisgau, Germany
email: <name>@informatik.uni-freiburg.de

Abstract

If Decision Diagrams (DDs) are used for the representation of the logical behavior of a combinational logic circuit it has to be traversed in topological order. At each gate the corresponding synthesis operation is carried out. This traversal process is called symbolic simulation.

Obviously the sequence in which the operations are performed at each gate influences the number of nodes needed during the computation. In this paper we consider different traversal strategies for OBDDs and OMDDs. We compare them by means of experiments.

1 Introduction

Decision Diagrams (DDs) are used in many fields of design automation. This holds especially for *(Ordered) Binary Decision Diagrams* ((O)BDDs) [3], e.g. in the area of verification [8, 7]. In the meantime also *Ordered Multi-Valued Decision Diagrams* (OMDDs) [10] have been used for verification [4].

The basic underlying algorithm of all the verification approaches is the symbolic simulation, i.e. a traversal of the circuit to be verified where the DDs are constructed for each internal signal. Obviously the sequence in which the DDs are combined at each gate are not uniquely determined.

In this paper we study the influence of the gate evaluation sequence. We consider different circuit dependant parameters, e.g. the number of cells or the support. Most of the information can be computed efficiently that is to say only linear runtime is needed in a preprocessing phase. Most often this information can directly be computed during the reading of the circuit structure. Depending on that sequence the number of nodes needed during the construction varies. Our experiments constructing OBDDs and OMDDs for *Multi-Valued Logic Networks* (MVLNs) as considered in [5] show that there exists a 10% gap between the *best case* and the *worst case*. This gap holds for OBDDs and OMDDs as well, but the strategies for the two- and multi-valued case differ.

The paper is structured as follows: In Section 2 MVLNs, OBDDs and OMDDs are defined. The different parameters of our traversal algorithm are described in Section 3. In Section 4 experimental results are given. Finally the results are summarized.

2 Preliminaries

We provide an introduction to basic notations which are important for the understanding of this paper.

2.1 Multi-Valued Logic Networks

In general, a *Multi-Valued Logic Network* (MVLN) can be modeled as a directed acyclic graph $C = (V, E)$ with some additional properties: Each vertex $v \in V$ is labeled with the name of a basic cell or with the name of a *Primary Input* (PI) or *Primary Output* (PO). The collection of basic cells available is given by a fixed library. This library contains MIN-, MAX-, INV- and LITERAL-gates[1]. Of course, basic cells with arbitrary complexity, especially with an arbitrary number of inputs, are possible. There is an edge (u, v) in E from vertex u to v, iff an output pin of the cell associated to u is connected to an input pin of the cell associated to v, i.e. edges contain additional information to specify the pins of the source and sink node they are connected to. Vertices have exactly one incoming edge per input pin. Nodes labeled as PI (PO) have no incoming (outcoming) edges.

[1] In the binary case MIN- and MAX-gates correspond to AND- and OR-gates, respectively.

To simulate the circuit each PI may assume the values of a given ordered finite set $P = \{0, \dots, k-1\}$ where k denotes the number of elements of the logic. The complement (INV-gate) of a signal x is defined as $\overline{x} = (k-1) - x$. A LITERAL-gate (a, b) $(a, b \in P, 0 \le a \le b < k)$ has one input and one output[2]. For a given input x the behavior of such a gate is defined by:

$$f(x) = \begin{cases} k-1 & : \quad a \le x \le b \\ 0 & : \quad \textit{otherwise} \end{cases}$$

It can then be proven that the set of all gates defined above is functionally complete [9].

2.2 Decision Diagrams

As well-known each Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ can be represented by an *Ordered Binary Decision Diagram* (OBDD) [3], i.e. a directed acyclic graph where a Shannon decomposition is carried out in each node.

It is straightforward to extend OBDDs to *Multi-Valued Decision Diagrams* (MDDs) [10] representing functions $f : \{0, .., k-1\}^n \to \{0, .., k-1\}$. Each internal node has k outgoing edges and there exist k terminal nodes, one for each logical value[3]. In [10] it has been shown that the efficient operations known for BDDs can also be carried out on MDDs using a *case*-operator instead of the *ite*-operator [1].

A DD is called *ordered* if each variable is encountered at most once on each path from the root to a terminal node and if the variables are encountered in the same order on all such paths. A DD is called *reduced* if it does not contain vertices either with isomorphic sub-graphs or with all successors pointing to the same node.

In the following we only consider reduced, ordered MDDs, i.e. reduced OMDDs. If $k = 2$ holds for the OMDD, the OMDD is an OBDD.

3 Parameters

Before we give the definition of the different parameters a first (trivial) example will demonstrate the influence of the sequence.

Example 1 Let $k = 2$ and g be an AND-gate with three inputs a, b and c. If c has the constant logical behavior 0 the OBDD only consists of the terminal 0. Let us assume that a and b are represented by "large" OBDDs. Obviously the output of g also represents the logic value 0. But since ITE is only able to handle two operands for synthesis at a time two out of the three operands have to be chosen. Combining a and b first is the worst case, since "many" useless nodes are created that are not needed for the representation of the final result.

If c is chosen first no additional nodes are created.

[2] These LITERAL-gates are also called *window literals*.

[3] In our application we restrict ourselves w.l.o.g. to the case that all variables are defined over the same set of values.

In the following we consider several different alternatives how to choose the sequence at a gate with more than 2 inputs. In the following section we will see that the choice of the ordering affects the number of nodes that are needed during the construction by approximately 10%. (Obviously the number of nodes for the final result is not influenced, since OMDDs are a canonical representation.)

We consider for each gate:

Predecessor path (PP): Number of paths in the input cone of the gate.

Successor path (SP): Number of paths in the output cone of the gate.

Predecessor cell (PC): Number of cells in the input cone of the gate.

Successor cell (SC): Number of cells in the output cone of the gate.

Support (S): Number of PIs in the input cone of the gate.

Output (O): Number of POs in the output cone of the gate.

Height (H): Height of the gate.

Depth (D): Depth of the gate.

All these informations can easily be obtained from the initial circuit description by a traversal in linear time.

For the above structures we consider in the following the increasing (<) and decreasing (>) order at each gate. E.g. if the incoming gates are considered in increasing order with respect to the number of paths in the input cone this is denoted by PP<.

As an "on-line" measure we also consider the number of BDD nodes used for the representation of the incoming gates. This method is denoted as BDD (again in combination with < and >).

name	PP>	PP<	SP>	SP<	S>	S<
s641	3360	**2847**	2947	3215	3360	**2847**
s713	3419	**2965**	2976	3275	3416	**2965**
s820	1785	1619	1649	1802	1801	**1594**
s832	1830	1648	1680	1842	1840	**1624**
s838	5462	**5315**	5357	5432	5462	**5315**
s953	2916	2857	2868	2899	2916	**2853**
s1238	11548	10658	11009	11387	11378	**10268**

name	PC>	PC<	SC>	SC<	O>	O<
s641	3358	2848	2947	3219	2926	3246
s713	3418	2966	2976	3255	2985	3284
s820	1801	1619	1653	1796	1649	1791
s832	1839	1648	1682	1841	1680	1829
s838	5335	**5315**	5357	5401	5388	5432
s953	2917	2855	2868	2899	2868	2899
s1238	11560	10391	10989	11388	11014	11356

name	BDD>	BDD<	H>	H<	D>	D<
s641	3333	**2847**	2925	3208	3358	3208
s713	3392	**2965**	2984	3246	3417	3246
s820	1801	1613	1657	1771	1802	1771
s832	1840	1642	1689	1812	1840	1812
s838	5462	**5315**	5340	5413	5418	5413
s953	2915	2860	2873	2897	2919	2897
s1238	11599	**10268**	10884	11360	11547	11360

Table 1: Influence for $k = 2$

4 Experimental Results

An OMDD package parameterized in k has been implemented in C++ [4] and the methods discussed in this paper have been applied. For the experimental results some of the benchmarks from [2] have been used analogously to [5]. All measurements were performed on a *Sun Sparc Station 4*. A node limit of 250.000 nodes and a limit of 1 CPU hour was set. Variable interleaving [6] has been applied for all benchmarks to determine a good variable ordering.

In a first series of experiments we compare the different methods for the case $k = 2$. The results are given in Table 1. (The best results are given in bold.) One might expext that the "on-line" method BDD gives the best result, but this is not the case for the examples considered here. BDD< achieves good results, S< is even better. PP> gets the worst results. Thus, functions with small suppport should be combined first.

In a second series of experiments we applied all methods for $k = 5$ to benchmarks *s820* and *s832*. (For all other circuits the OMDD could not be constructed within reasonable time bounds [4].) As can be seen from these two examples the "good" methods are completely different than in the binary case, i.e. PP> (which was the worst method in the binary case) is one of the best methods for $k = 5$.

Therefore we finally compared the method PP> and S< for varying k in Table 3. We can observe that for $k = 3$ S< ist still better, but the situation changes for increasing k. Further experiments (not included in the paper) have shown that for large k no "unique" method exists that is better than all others in all cases.

5 Conclusions

In this paper we considered the problem of input ordering during symbolic simulation. By experiments it has been shown that the influence of the ordering is 10% between best case and worst case on average. The

name	PP>	PP<	SP>	SP<	S>	S<
s820	33867	35879	34381	34327	34037	35279
s832	34797	36309	34923	34942	34981	35653
name	**PC>**	**PC<**	**SC>**	**SC<**	**O>**	**O<**
s820	33947	35637	34391	34311	34387	34294
s832	34864	35961	34931	34934	34929	34898
name	**BDD>**	**BDD<**	**H>**	**H<**	**D>**	**D<**
s820	33835	35647	34407	34270	**33794**	35713
s832	34759	36083	34950	34882	**34722**	36014

Table 2: Influence for $k = 5$

name	PP>			S<		
	$k=2$	$k=3$	$k=5$	$k=2$	$k=3$	$k=5$
s820	1785	5865	**33867**	**1594**	**5759**	35279
s832	1830	6076	**34797**	**1624**	**5887**	35653

Table 3: Influence of varying k

ordering for the best case for binary valued circuits can easily be determined by a circuit traversal in linear time.

For MVLNs with $k > 2$ the situation becomes more difficult. It is still an open problem how to determine a good sequence in this case.

References

[1] K.S. Brace, R.L. Rudell, and R.E. Bryant. Efficient implementation of a BDD package. In *Design Automation Conf.*, pages 40–45, 1990.

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

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

[4] R. Drechsler. Verification of multi-valued logic networks. In *Int'l Symp. on multi-valued Logic*, pages 10–15, 1996.

[5] R. Drechsler, R. Krieger, and B. Becker. Random pattern fault simulation in multi-valued circuits. In *Int'l Symp. on multi-valued Logic*, pages 98–103, 1995.

[6] 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.

[7] 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.

[8] 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.

[9] J.C. Muzio and T.C. Wesselkamper. *Multiple-Valued Switching Theory.* Adam Hilger, Bristol and Boston, 1986.

[10] A. Srinivasan, T. Kam, S. Malik, and R.E. Brayton. Algorithms for discrete function manipulation. In *Int'l Conf. on CAD*, pages 92–95, 1990.