SPEEDING UP SYMBOLIC MODEL CHECKING[‡]

Christoph Meinel, Christian Stangier[§]

Abstract

Model checking is a widely used technique in verification of sequential circuits, reactive systems, protocols, etc. Ordered Binary Decision Diagrams (OBDDs) allow a symbolic representation of the model that may bypass the state space explosion problem of larger systems. As the size of OBDDs and also the computation time depends on the order of the input variables, the verification may only succeed if a well suited variable order is chosen. Since the characteristics of the represented functions are changing, the variable order has to be adapted dynamically. Unfortunately, dynamic reordering strategies are often very time consuming and sometimes do not provide any improvement of the OBDD representation.

We present a technique that substantially accelerates the reordering process by using easily computable structural information about the represented function without increasing OBDD sizes significantly.

Keywords: Verification, symbolic model checking, OBDDs, reordering

[‡]In Proc. of 5th Advanced Computer Systems (ACS'98), Szczecin, Poland, pp. 548 – 554
[§]Supported by the DFG-Graduiertenkolleg "Mathematische Optimierung"

1 INTRODUCTION

Model checking [CES86] has been proven to be a powerful tool in verification of sequential circuits, reactive systems, protocols, etc. The model checking of systems with huge state spaces is possible only if there is a very efficient representation of the model. Ordered Binary Decision Diagrams (OBDDs) [Bry86] allow an efficient *symbolic* representation of the model[CM90].

Due to the huge number of operations applied to the OBDDs during model checking, the computation time is strongly related to the size of the OBDDs. As, the order of the input variables has a strong influence on the size of the OBDDs, well suited variable orders have to be found. There exist a lot of heuristics for finding reasonable initial variable orders (e. g. [MWB+88, MIY90]). In practice, techniques that improve the size of a given OBDD by changing the variable order dynamically during the computation have been proven to be more powerful. Such dynamic reordering strategies are especially useful for symbolic model checking, since the represented functions (e. g. reachable state sets) are changing during computation. As a consequence, the variable order has to be adapted to fulfil the new requirements.

Although, dynamic variable reordering may drastically reduce the OBDD size, it is often very time consuming and does not lead to substantially smaller OBDD sizes.

Our goal is to speed up the reordering process. We do this for two reasons: First, spending large fractions of the computation time of a model checking on reordering that does not lead to smaller OBDDs is very annoying. Secondly, if the reordering works faster, it can be invoked more frequently to improve the variable order.

We have developed an adaption for a dynamic reordering technique called *block restricted sifting* [MS97] intended originally for combinatorial circuits that takes into acount the special needs of symbolic model checking. The major improvement of this technique is a substantial reduction of the computation time. If these reordering techniques are applied more frequently, we obtain even smaller OBDD sizes.

The paper is structured as follows: In the next section, we recall OBDDs and symbolic model checking. In Section 3, we describe the reordering technique that we have used and its application to symbolic model checking. In Section 4, we summarize our verification experiments. The last section concludes the paper with an outlook on future work.

2 PRELIMINARIES

Model checking [CES86] is the problem to decide whether an implementation satisfies its specification given in terms of a temporal logic, e. g. the so-called *computation tree logic* (CTL).

The formulas of CTL describe properties of infinite paths of states that are traversed during the computation. In addition to the Boolean operators $\wedge$, $\vee$ and $\neg$, CTL has four temporal operators:

X The *next* operator describes a condition that is true in the next state of the computation.

G The *global* operator describes a condition that is true for all states of a path.

F The *future* operator describes a condition that is true on a path some time in the future.

U The *until* operator $a\mathbf{U}b$ is true on a path if a is true until b has been true at least once.

All temporal operators are quantified with either **A** (on all paths it holds...) or **E** (there exists a path where ...).

The **X** operator can be described in terms of the transition relation and Boolean operations. The other operators can be computed by fixpoint iterations on the **X** operator.

The model checking of many real life systems fails because of state space explosion. A possible solution is to represent the objects, relations and sets, that are used during model checking, symbolically with Ordered Binary Decision Diagrams.

An **Ordered Binary Decision Diagram** (**OBDD**) [Bry86] G for a Boolean function $f : \mathcal{B}^n \to \mathcal{B}$ over the Boolean variables $(x_1, \ldots, x_n)$ is a directed acyclic graph consisting of inner nodes labelled by Boolean variables and sinks labelled by the Boolean constants 1 and 0. Each inner node has two outgoing edges: the 1-edge and the 0-edge. The OBDD has a starting node called root. The computation of $f(a_1, \ldots, a_n)$ follows a path from the root to a sink, where on a node labelled by x_i the input bit a_i is tested. If $a_i = 1$, the path follows the 1-edge, otherwise the 0-edge. The value of the reached sink determines the value of $f(a_1, \ldots, a_n)$. On a path from the root to the sink, each variable occurs at most once. The variables on a path respect a given order, which is (possibly after renaming) $x_1, \ldots, x_n$. For an edge leading from a node labelled by x_i to a node labelled by x_j it follows that $j > i$. More than one Boolean function can be represented in a common **shared** OBDD using one root node for each function.

All important operations on Boolean functions like synthesis, satisfiability test, equivalence test etc. have efficient algorithms based on OBDDs. For a survey see [MT98]. Due to the fact, that there are 2^{2^n} functions on n variables, many functions have an OBDD representation of exponential size, but most functions of practical relevance have only small or moderate OBDD size.

The size of an OBDD depends heavily on the variable order. Therefore, one may experience an exponential gain in size by choosing a good variable order. Since it is NP-complete to find the optimal variable order for a given function (see [THY93]), much effort is spent on finding reasonable good orders or improving given ones. A promising approach is based on swapping the position of neighbored variables in a given OBDD. This operation can be performed locally and thus can be carried out efficiently. The *sifting* algorithm [Rud93] that is based on this idea moves each variable by swap operations to the top and to the bottom of the order to find its best position. This algorithm has proven to be one of the fastest reordering strategies, it does not require much space and may lead to much smaller OBDDs.

The idea to combine model checking with the symbolic OBDD algorithms was first introduced by Coudert/Madre [CM90]. Using this way of *symbolic* model checking, real life systems up to 10^{100} states can be verified. SMV written by Ken McMillan [McM93] is a wellknown tool that uses OBDDs.

3 SPEEDING UP MODEL CHECKING

OBDD based model checking tools like SMV use variable reordering techniques for the reduction of OBDD sizes. SMV for example uses Rudells sifting algorithm. Indeed, variable reordering is strongly recommended since the huge amount of computations like iterations or fixpoint computations requires too much time or is impossible due to memory limitations, if the underlying OBDDs are too large. Also, the represented functions, like reachable state sets, are changing during computation and thus, the variable order has to be adapted to avoid explosion of OBDD sizes.

Although, sifting is the fastest common reordering technique, it is often too time consuming for an application during model checking. It emerges that a large fraction of the computation time is spent on sifting without a gain in OBDD size. Another fact is that sifting is to costly to be invoked as often as required if functions are changing.

Our goal is to accelerate the variable reordering process but retaining good OBDD sizes. To obtain this, we adapted a method called *block restricted sifting* (BRS) to the needs of model checking. BRS was introduced by [MS97]. The idea behind BRS is to move the variables during reordering only within fixed blocks instead of moving them through the complete order. From theory it is known, that changing the variable order of a block does not affect the size of the other blocks.

The determination of the block boundaries follows from a communication complexity argument. A small information flow between two parts of an OBDD indicates a good candidate for a block boundary. If there is only little information flow between two blocks the distribution of variables to these blocks is well chosen. Improving the variable order inside the blocks will lead to a significant reduction of the OBDD size. The information flow is best indicated by the number of subfunctions that cross one level. With the aid of a *subfunction profile* we get easily computable structure information of the represented function.

The approach for block restricted sifting is the following:

1. Compute the subfunction profile,
2. detect the local minima of the profile,
3. specify the block boundaries,
4. sift within the blocks.

Block restricted sifting is well suited for symbolic model checking, since it substantially accelerates the reordering. This is possibly due to the observed fact, that the subfunction profiles have distinct “waists” that serve as boundaries.

4 EXPERIMENTAL RESULTS

We integrated the block restricted sifting strategy in the SMV tool (version 2.5). Each time sifting is invoked, block restricted sifting is performed instead. Furthermore, we had to take care about present and next state variables, because some OBDD routines of SMV rely on restricted positioning of the variables. Therefore, each next state variable is always placed directly beside its corresponding present state variable and never in another block.

There are several parameters that may be adjusted:

MINBLOCK denotes the minimum fraction of the number of variables, that a block must contain. MINBLOCK is usually set to 0.1 .

MINUP/MINDOWN denote the gradient of the levels beside a local minimum. These parameters ensure that only destinctive waists are chosen as boundaries. Their default values are 0.01 resp. 0.02 .

OVERLAPPING An overlapping of blocks may be specified, too. We implemented this overlapping, to allow a variable placed on the border of two blocks to move to the other block. The overlapping is set to 2 variables.

For an experimental basis we used the benchmark suite of Yang [Yan98] and the examples that come with SMV. We used the models that require less than one hour of computation time.

For experimental results see Figure 1 and Figure 2. It emerged that for symbolic model checking larger block sizes are required than for combinational verification, i. e. up to 25% instead of 10%.

In most cases an overlapping of 2 variables was sufficient. Only for the `abp8` model an overlapping of 4 variables produced better results (12%: 90.96s, 14700 nodes).

We also performed experiments, where the block restricted sifting is invoked earlier. This resulted in most cases to better OBDD sizes without requiring more computation time for the reordering. Furthermore it was possible to verify the `pci4p` model in less than one hour (3179.2s, 81787 nodes).

We performed all experiments on a Intel PentiumPro 200Mhz Linux Workstation with datasize limited to 200MByte and CPU time limited to 60 minutes.

5 CONCLUSION

Variable reordering during symbolic model checking is very time consuming and often does not achieve an improvement of OBDD size. We used easily computable structural and implicit semantical information about the represented functions to speed up the reordering process without losing too much efficiency in the representation. Due to the faster computation, the reordering may be invoked more frequently and leads to even smaller OBDD sizes.

What has to be done next, is to combine the standard and the new method. For example the first and the final reordering can be done by standard sifting and the intermediate reorderings can be done by block restricted sifting for the reason of faster computation.. Another possibility for improvement is to use the blocks to keep strongly related variables (e. g. variables representing a communication channel) close together.

REFERENCES

[Bry86] Bryant R. E., Graph-Based Algorithms for Boolean Function Manipulation, IEEE Transactions on Computers, 1986, C-35, pp. 677–691

[Bry91] Bryant R. E., On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication, IEEE Transactions on Computers, 1991, 40, pp. 205–213

[CES86] Clarke E. M., Emerson E. A., and Sistla A. P., Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications, ACM Transactions on Programming Languages and Systems, 1986, 8, pp. 244–263

[CM90] Coudert O., and Madre J. C., A Unified Framework for the Formal Verification of Sequential Circuits, in Proc. IEEE International Conference on Computer-Aided Design, 1990, pp. 126–129

[McM93] McMillan K. L., Symbolic model checking, Kluwer Academic Publishers, 1993

[MIY90] Minato S., Ishiura N., and Yahima S., Shared Binary Decision Diagrams with Attributed Edges for Efficient Boolean Function Manipulation, in Proc. of the 27th ACM/IEEE Design Automation Conference, 1990, pp. 52–57

[MS97] Meinel C., and Slobodová A., Speeding up Variable Reordering of OBDDs, in Proc. of the International Conference on Computer Design, 1997, pp. 338–343

[MT98] Meinel C., and Theobald T., Algorithms and datastructures in VLSI-Design, Springer, 1998

[MWB+88] Malik S., Wang A. R., Brayton R. K., and Sangiovanni-Vincentelli A., Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment, in Proc. of the 25th Design Automation Conference, 1992, pp. 268–333

[Rud93] Rudell R., Dynamic Variable Ordering for Ordered Binary Decision Diagrams, in Proc. IEEE International Conference on Computer-Aided Design , 1993, pp. 42–47

[SM98] Slobodová A., and Meinel C., Sample Method for Minimization of OBDDs. In Proc. of the International Workshop on Logic Synthesis, 1998, pp. 311–316

[THY93] Tani S., Hamaguchi K., and Yajima S., The Complexity of the Optimal Variable Ordering Problem of Shared Binary Decision Diagrams, in Proc. ISAAC'93, volume 762 of Lecture Notes in Computer Science, Springer, 1993, pp. 389–398

[Yan98] Yang B., SMV Models, `http://www.cs.cmu.edu/~bwolen/software/` .

	SMV Original	SMV with Block Restricted Sifting			
Min. Blocksize		10%	15%	20%	25%
a7spec	$> 1h$	196.79	$> 1h$	1990.34	$> 1h$
abp4	20.32	**19.49**	19.68	19.65	19.98
abp8	**90.17**	152.19	151.20	858.42	948.38
brp	150.01	255.25	119.88	142.29	**119.46**
counter16	4.52	4.51	4.52	**4.48**	4.59
gigamax	4.41	4.22	**3.36**	3.49	3.85
guidance	1233.69	277.08	**238.72**	305.15	376.34
my_abp_ch	127.90	344.20	141.39	**60.84**	95.94
p-queue	2137.38	**5.13**	7.72	9.72	17.13
pci3p	13.82	4.17	3.73	**3.39**	6.34
periodic	32.70	24.26	25.34	**23.24**	23.57
prod-cons	35.50	145.64	175.98	41.36	**31.09**
production-cell	318.47	217.17	190.58	**91.62**	107.60
semaphore10	13.68	**11.11**	11.35	11.31	12.77
syncarb10	4.82	**2.04**	2.38	2.24	3.22
$\sum$	> 7787.39	1663.25	> 4695.83	3567.54	> 5370.26

Figure 1: Comparison of Computation Time in CPU Seconds for Standard Sifting and Block Restricted Sifting.

	SMV Original	SMV with Block Restricted Sifting			
Min. Blocksize		10%	15%	20%	25%
a7spec	–	77412	–	77483	–
abp4	6540	6540	6540	6540	6540
abp8	12670	26170	27477	802372	816415
brp	241184	606955	265143	312794	260057
counter16	5012	5012	5012	5012	5012
gigamax	31587	31587	31587	31587	31587
guidance	30938	48894	28096	29227	29512
my_abp_ch	21394	35297	25438	21394	21394
p-queue	14923	37478	19247	23434	35714
pci3p	21041	28878	15673	20883	15280
periodic	21926	21926	21926	21926	21926
prod-cons	18601	78245	81472	20709	19199
production-cell	53711	93576	71878	42139	42139
semaphore10	13831	13831	13831	13831	13831
syncarb10	19321	19486	20045	19321	19321

Figure 2: Comparison of OBDD Size for Standard Sifting and Block Restricted Sifting.