Speeding up Variable Reordering of OBDDs*

Christoph Meinel† Anna Slobodová
ITWM-Trier, Bahnhofstr. 30-32, D-54292 Trier
e-mail: meinel/anna@itwm-trier.fhg.de

Abstract

In this paper, we suggest a block-restricted sifting strategy which is based on the restriction of Rudell's sifting to certain blocks of variables. The application of this strategy results in a considerable improvement of the time performance without causing a substantial increase of OBDD size. The blocks of variables were determined according to some structural properties reflecting the communication complexity within the considered OBDD without using any additional information about its functionality or dependency of variables. This assures the general usability of the heuristic and the possibility to use it dynamically.

1 Introduction

Ordered Binary Decision Diagrams (OBDDs) are one of the few known representations for Boolean functions that are, due their nice algorithmical properties, of a great importance in many different applications. Successful application to complex problems, however, almost invariably entails resorting to various techniques to reduce OBDD sizes. The main resource for the optimization of the OBDD size is the choice of an appropriate variable ordering. There are examples that show exponential gaps in the sizes of functionally equivalent OBDDs which differ in their variable orderings. Since the problem of finding an optimal variable ordering is NP-complete [THY93, BW94], there is a strong need for good heuristics for finding variable orderings with respect to which the OBDDs of interest are of managable size.

If the variable ordering heuristics are assumed to start from an input representation that is not an OBDD, then they may use a lot of additional information like circuit topology or formula shape. However, if the heuristics start with OBDDs, they have to work 'blind'. Usually, such heuristics proceed in greedy manner using certain local changes within the variable ordering. For this purpose, discover and investigation of those OBDD properties is required that predict how a good variable ordering looks like, e.g., which variables should be grouped together, which ones should appear on the top of the ordering, etc..

In this paper, we investigate the subfunctions' profile as a structural attribute of a given OBDD. From the profile, we derive information which variables should be kept in one block. Based on this idea, we propose an improvement of the so-called sifting introduced by Rudell [Rud93]. Sifting is based on the idea of searching for the best position for each variable. The individual variables are moved through the whole ordering, one after another. The basic operation used to implement this movement is the exchange of two neighbouring variables (the so-called swapping operation), which can be done very fast if an appropriate representation of the OBDD is used. In practical applications, sifting is known to be the state-of-the-art

*in Proceedings IWLS'97, Granlibakken, USA
†Universität Trier, FB IV – Informatik, D-54286 Trier

reordering heuristic. Since it does not require any additional information on the represented function, it is applicable whenever OBDDs can be used successfully. However, sifting has its limitation, too: Although one swap is very fast (linear in the number of nodes of the two considered levels), the total time of sifting grows very fast with the number of variables. In applications with several thousand variables, sifting of only one variable over the whole ordering takes too much time.

Several attempts have been made to improve the performance of the original sifting, e.g. in [PSP94, PS95] variables are grouped according to some symmetry properties and the entire groups are sifted instead of single variables. Another improvement of sifting can be obtained by an appropriate setting of the parameter that restricts the size growing of the OBDDs. Yet another possibility to improve the performance of sifting is an idea implemented in the CUDD-Package [S96] which consists of a pre-estimation of the possible size reduction of the considered OBDDs that could be obtained if the current variable is sifted in a given direction and to interrupt the sifting process if this estimation gives worse values than the optimum found so far.

Here we propose the method of *block-restricted sifting*. The main idea consists of a restriction of the application of swap operations to within certain blocks of variables which are determined according to some structural properties of OBDDs which are determined by communication complexity theoretic considerations. This improves the time performance of the original sifting strategy considerably, without causing a (substantial) loss in the final size. Similarly as in the case of the original sifting, no additional knowledge about the represented function, its circuit design, or 'grouping properties' of the variables are used. Hence, this method can be used in any application without spending much effort to investigate additional characteristics. In our experiments, the block-restricted sifting applied in the final sifting yields about 50% time profit and costs only 0.5% size increase.

2 Block-Restricted Reordering

We look for a simple structural property of an OBDD that can be easily computed and tells something about certain properties of the represented function with respect to the current variable ordering. Here we use the shape of an OBDD representation of the function under consideration to determine certain blocks of variables. Then each variable is sifted within its block only. The underlying idea is that an 'indentation' in the shape of an OBDD indicates a weak information flow from the part above the indentation to the part below it. Indeed, the amount of communication between the top part and the bottom part of an OBDD is reflected by the number of nodes (i.e., subfunctions) in the bottom part referred from the top part. Hence, 'cutting' an OBDD in an indentation divides it into two parts that do not communicate too much. Consequently, the restriction of sifting on such parts should not result in an essential loss in optimization, while time could be decreased considerably.

For each i, $1 \leq i \leq n$, we can compute the number of subfunctions in P with respect to the first $i-1$ variables in the order π. An indentation in the subfunctions' profile of P indicates a weak information flow between those parts of P defined by cutting it at the corresponding level. This can be used in such a way that the two parts of the OBDD can be reordered separately. The idea described above can be condensed into four steps:

Block-restricted Sifting.

1. Compute a subfunctions' profile for the given OBDDs.
2. Find local minima in this profile.
3. Choose some of them to define blocks to restrict sifting.

4. Reorder OBDDs within the chosen blocks by means of sifting.

3 Computing the Number of Subfunctions

The efficient computation of the number of subfunctions for all levels of OBDDs depends on the used OBDD package. The procedure described here was designed for the CUDD-package.

```
ComputeSubfunctions(table,roots)
begin
   for (i=0; i<nroots; i=i+1) {
      node=roots[i];
      if (NotMarked(node)) {
         for (j=1; j<=node.var.index; j++)
            sf[j]=sf[j]+1;
         SetMark(node);
      }
   }
   for (i=1; i<=nvars; i=i+1)
      for each node in table[i]
         for both sons of the node
            if (NotMarked(son)) {
               for (j=i+1; j<=son.var.index; j=j+1)
                   sf[j]=sf[j]+1;
               SetMark(son);
            }
end
```

Although the computation of the subfunctions takes negligible time, we need to have fast access to all roots. If an implementation does not support this, we can approximate the subfunctions' profile of the OBDDs by its nodes' profile. We have done some experiments with this rought approximation as well. However, the power of the method decreased substantially.

4 Some Experimental Results

Circuit	sifting (1)		block-restricted sifting (2) (subfunctions' profile)					block-restricted sifting (3) (nodes' profile)			
	size (nodes)	time (sec)	size (nodes)	time (sec)	Δsize (%)	Δtime (%)	bounds (sec)	size (nodes)	time (sec)	Δsize (%)	Δtime (%)
C1355	121,264	206.3	121,264	74.0	0.0	-64.1	0.61	121,264	73.0	0,0	-64.6
C1908	26,767	29.0	29,993	12.1	12.1	-58.3	0.13	30,112	8.9	12.5	-69.3
C3540	124,926	132.2	125,008	82.0	0.1	-38.0	0.86	134,314	46.0	7.5	-65.2
C499	36,603	107,6	36,836	54.5	0.6	-49.4	0.30	36,836	54.4	0.6	-49.4
C7552	115,474	456.9	108,833	170.4	-5.8	-62.7	0.77	117,604	101.4	1.8	-77.9
C880	35,990	42.5	39,460	32.8	9.6	-22.8	0.27	39,938	9.9	11.0	-76.7
i10	83,028	221.9	88,083	98.8	6.1	-55.5	0.78	89,237	59.3	7.5	-73.3
mm30a	982,326	3,158.5	982,326	1,482.4	0.0	-53.1	7.39	982,326	1,478.2	0.0	-53.2
s15850.1	39,615	135.7	39,865	44.6	0.6	-67.1	0.40	40,059	52.5	1.1	-61.3
s38584.1	36,311	95.7	36,311	97.7	0.0	2.1	1.25	38,295	39.9	5.5	-58.3
s4863	337,277	555.5	341,436	280.2	1.2	-49.6	2.73	341,433	183.9	1.2	-66.9
s6669	187,249	206.5	187,249	206.9	0.0	0.2	1.29	187,465	146.9	0.1	-28.9
Total	2,126,830	5,348.3	2,136,664	2,636.4	0.46	-50.71		2,158,883	2,254.0	1.51	-57.86

Table 1: One time reordering. Comparison of the unrestricted sifting (1) with the block-restricted sifting. The bounds of blocks are set according to subfunctions' profile in method (2) and according to the nodes' profile of the OBDDs in the method (3).

Circuits	sifting		block-restricted sifting			
	size (nodes)	total time (sec)	size (nodes)	total time (sec)	Δ size (%)	Δ time (%)
C1355	121,264	464.34	121,264	343.40	0.00	-26.05
C1908	26,767	92.24	29,993	77.96	12.05	-15.48
C3540	124,926	447.63	146,760	399.54	17.48	-10.74
C499	36.603	293.64	36,836	241.57	0.64	-17.73
C7552	115,474	2557.25	155,270	1166.48	34.46	-54.39
C880	35,990	146.48	39,460	137.94	9.64	-5.83
i10	83,028	555.82	83,888	766.21	1.04	37.85
mm30a	982,326	5947.84	982,326	3228.96	0.00	-45.71
s15850.1	39,615	383.51	39,865	285.51	0.63	-25.55
s38584.1	36,311	549.62	36,311	552.40	0.00	0.51
s4863	337,277	1328.27	341,436	1067.90	1.23	-19.60
s6669	187,249	581.85	202,935	551.31	8.38	-5.25
Total	2,126,830	13,348.49	2,216,344	8,819.18	4.21	-33.93

Table 2: Dynamic reordering completed by final reordering. Comparison of the unrestricted sifting with the block-restricted sifting. The bounds of blocks are set according to subfunctions' profile of the OBDDs.

All experiments were done on a Sun SPARC 20. The time limit was set to 10,0000s, the memory limit to 100 Mbyte.

References

[BMS95] J. Bern, Ch. Meinel, A. Slobodová: Global Rebuilding of OBDDs – Avoiding Memory Requirement Maxima. *In Proc. of CAV'95. LNCS 939 (Springer Verlag 1995) 299-308.*

[Bry86] R.E. Bryant: Graph Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Computers, C-35 (1986) 677-691.*

[BW94] B. Bollig, I. Wegener: Improving the Variable Ordering of OBDDs is NP-complete. *Manuscript.*

[FYBS93] E. Felt, G. York, R. Brayton, A. Sangiovanni-Vincentelli: Dynamic Variable Reordering for BDD Minimization. *In Proc. EURO-DAC'93, 130–135.*

[Lon93] D. Long: OBDD-package, Carnegie Mellon University, 1993.

[MS9] Ch. Meinel, A. Slobodová: Global Changes in Variable Ordering. - Comparison of Different Methods - *Jessi Report, June 1996.*

[PS95] F. Somenzi, S. Panda: Who are the Variables in Your Neighbourhood? *In Proc. ICCAD'95, San Jose, CA, Nov. 1995.*

[PSP94] S. Panda, F. Somenzi, B.F. Plessier: Symmetry Detection and Dynamic Variable Ordering of Decision Diagrams. *In Proc. ICCAD'94, San Jose, CA, Nov. 1994, 628-631.*

[Rud93] R. Rudell: Dynamic Variable Ordering for Ordered Binary Decision Diagrams. *In Proc. IEEE/ACM ICCAD'93, 42-47.*

[S96] F. Somenzi: CUDD: CU Decision Diagram Package. Release 1.1.1. Technical Report, University of Colorado, 1996.

[THY93] S. Tani, K. Hamaguchi, S. Yajima: The Complexity of the Optimal Variable Ordering Problem of Shared Binary Decision Diagrams. *In Proc. of ISAAC'93, LNCS 762, 389-398.*