UTILIZING BDDS FOR DISJOINT SOP MINIMIZATION

Görschwin Fey *Rolf Drechsler*

Institute of Computer Science
University of Bremen, 28359 Bremen, Germany
{fey,drechsle}@informatik.uni-bremen.de

ABSTRACT

The application of Binary Decision Diagrams (BDDs) as an efficient approach for the minimization of Disjoint Sums-of-Products (DSOPs) is discussed. DSOPs are a starting point for several applications. The use of BDDs has the advantage of an implicit representation of terms. Due to this scheme the algorithm is faster than techniques working on explicit representations and the application to large circuits that could not be handled so far becomes possible. Theoretical studies on the influence of the BDDs to the search space are carried out. In experiments the proposed technique is compared to others. The results with respect to the size of the resulting DSOP are as good or better as those of the other techniques.

1. INTRODUCTION

A DSOP is a representation of a Boolean function as a sum of disjoint monoms (or cubes). DSOPs are used in several applications in the area of CAD, e.g. the calculation of spectra of Boolean functions [4, 5, 15] or as a starting point for the minimization of *Exclusive-Or-Sum-Of-Products* (ESOPs) [11, 13]. Some techniques for minimization of DSOPs working on explicit representations of the monoms have been introduced [6, 14], but are only applicable on smaller instances of the problem.

On the other hand BDDs are well known to be an efficient representation of Boolean functions and are widely used in synthesis [8, 9, 10] and formal verification [2, 3] of integrated circuits. The advantage of this data structure is an implicit representation of monoms in the ON-set of a Boolean function.

In this paper BDDs are shown to be applicable to the problem of DSOP minimization. The idea of utilizing BDDs is due to the efficiency resulting from the implicit representation of a Boolean function. Given a BDD of a Boolean function it is straightforward to extract a DSOP representation, since one-paths in the BDD correspond to monoms. Moreover different one-paths lead to disjoint monoms. We study the influence on the search space that results from the restriction to DSOPs that are represented by BDDs and show that not for every function the minimal DSOP is reachable. Also the importance of choosing a "good" variable order to get a small DSOP is theoretically supported. Experiments show that the new technique often produces smaller at least same sized DSOPs as other techniques on the one hand and that it is applicable to much larger circuits on the other hand.

The paper is organized as follows. Section 2 briefly introduces the necessary notion of BDDs and one-paths and shows how the DSOP-minimization is related to BDDs. Section 3 contains theoretical studies regarding the constraints on the search space implied by this approach. The following section gives the results of the experiments made on circuits. Finally the paper is closed by the conclusions.

2. PRELIMINARIES

2.1. BDDs

A BDD is a directed acyclic graph $G_f = (V, E)$ that represents a Boolean function f. The Shannon decomposition $g = x_i g_{x_i} + \overline{x}_i g_{\overline{x}_i}$ is carried out in each internal node v labeled with *label*$(v) = x_i$ of the graph, therefore v has the two successors *then*(v) and *else*(v). The leafs are labeled with 0 or 1 and correspond to the constant Boolean functions. The root node *root*(G_f) corresponds to the function f. If the order of labels on internal nodes is the same on each path from the root to a leaf the BDD is called an ordered BDD. If there are no isomorphic subgraphs the ordered BDD is referred to as reduced ordered BDD [1]. In the sequel *BDD* refers to a reduced ordered BDD only.

Definition 1. *A* one-path *in a BDD $G_f = (V, E)$ is a path*

$$p = (v_0, e_0, \ldots, v_{l-1}, e_{l-1}, v_l),$$

$$v_i \in V, e_i = (v_i, v_{i+1}) \in E$$

with $v_0 = root(G_f)$ and $label(v_l) = 1$. p has length $l + 1$.
$\mathcal{P}_1(G_f)$ denotes the number of different one-paths in the BDD G_f.

2.2. BDDs and DSOP

Consider a BDD G_f representing the Boolean function $f(x_1, \ldots, x_n)$. A one-path $p = (v_0, e_0, \ldots, e_{l-1}, v_l)$ of length $l + 1$ in G_f corresponds to an $(n - l)$-dimensional cube that is a subset of $ON(f)$. The cube is described by:

$$m_p = \bigcap_{i=0}^{l-1} l_i, \text{ where}$$

$$l_i = \begin{cases} \overline{label(v_i)}, & \text{if } v_{i+1} = else(v_i) \\ label(v_i), & \text{if } v_{i+1} = then(v_i) \end{cases}$$

Two paths p_1 and p_2 in a BDD are different iff they differ in at least one edge. Since all paths originate from *root*(G_f), there is a node v where the paths separate, let *label*$(v) = x_i$. Due to the Shannon decomposition carried out in v one of the monoms includes x_i, the other $\overline{x}_i$. Therefore the monoms m_{p_1} and m_{p_2} are disjoint.

Now the DSOP can easily be built by summing up all monoms corresponding to a one-path.

Remark 1. *Let G_f be a BDD of $f(x_1, \ldots, x_n)$ and $\mathcal{M}_1$ be the set of one-paths in G_f. Then G_f* represents the DSOP

$$\sum_{p \in \mathcal{M}_1} m_p,$$

where m_p is the monom given above.

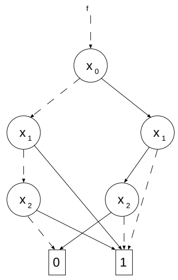


Figure 1: BDD of $f = \neg(x_0x_1x_2 + \overline{x}_0\ \overline{x}_1\ \overline{x_2})$

From this it is clear that the number of monoms in the DSOP represented by G_f is equal to the number of one-paths $\mathcal{P}_1(G_f)$. Thus, as opposed to the usual goal of minimizing the number of nodes in a BDD, the minimization of the one-paths is aimed at. A technique that is based upon the well-known Sifting algorithm [12] has recently been proposed in [7].

3. THEORY

When utilizing BDDs to find a DSOP it is of interest whether restrictions on the search space result from this technique. One question arising is, if the minimal DSOP of a function can always be represented by a BDD.

Lemma 1. *There exist functions that have no BDD representing the minimal DSOP.*

Proof. Consider the totally symmetric function $f = \neg(x_0x_1x_2 + \overline{x}_0\ \overline{x}_1\ \overline{x}_2)$. Due to the symmetry all BDDs of f are isomorphic regardless of the variable order. As can be seen in Figure 1 a BDD of f has four one-paths. On the other hand the function can be represented by a DSOP with three monoms: $f = \overline{x}_0x_1 + x_0\overline{x}_2 + \overline{x}_1x_2$ ∎

It is known that the variable order heavily influences the number of nodes of a BDD. Bryant [1] gave the function

$$b = x_1x_{n+1} + x_2x_{n+2} + ... + x_nx_{2n}$$

and the two variable orders

$$\begin{aligned}\pi_1 &= (x_1, x_{n+1}, \ldots, x_n, x_{2n})\\ \pi_2 &= (x_1, x_2, x_{2n-1}, x_{2n})\end{aligned}$$

as an example. The BDD of f has $2n$ nodes when π_1 is used, but more than 2^n nodes when π_2 is used. The number of one-paths in the corresponding BDDs G_b^1 and G_b^2 is exponential in n in both cases.

Nonetheless the same influence of the variable order can be proven for the number of one-paths in a BDD as the following lemma shows.

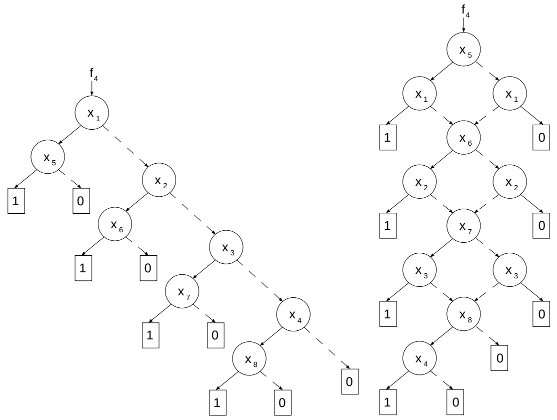
Lemma 2. *Let the Boolean function $f_n(x_1, \ldots, x_{2n}), n > 1$ be recursively defined:*

- $f_1 = x_1x_{n+1}$
- $f_j = f_{j-1} + (\bigcap_{i=1}^{j-1} \overline{x}_i)x_jx_{n+j}$

And let π_1 and π_2 be two variable orders:

- $\pi_1 = (x_1, x_{n+1}, x_2, x_{n+2}, \ldots, x_n, x_{2n})$
- $\pi_2 = (x_{n+1}, x_1, x_{n+2}, x_2, \ldots, x_{2n}, x_n)$

The BDD $G_{f_n}^1$ with respect to π_1 has $2n$ nodes and n one-paths. But $G_{f_n}^2$ with respect to π_2 has $3n$ nodes and $2^n - 1$ one-paths.



(a) $G_{f_4}^1$, variable order π_1

(b) $G_{f_4}^2$, variable order π_2

Figure 2: The BDDs of f_4 (for readability the BDDs are not totally reduced, so there are several '0'- and '1'-nodes)

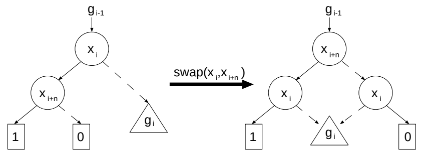


Figure 3: The swap operation applied on a pair (x_i, x_{i+n}) of variables

Proof. At first the variable order π_1, i.e. the BDD $G_{f_n}^1$ is considered (for an easier understanding of the proofs flow see the example for $n = 4$ in Figure 2(a)). The nodes in $G_{f_n}^1$ correspond to cofactors in f_n. By construction the positive cofactor of f_n with respect to x_j only depends on x_{n+j} and hence:

$$f_n(\overline{x}_1 \ldots \overline{x}_{j-1}x_jx_{n+j}) = 1 \text{ and}$$

$$f_n(\overline{x}_1 \ldots \overline{x}_{j-1}x_j\overline{x}_{n+j}) = 0$$

Thus, for a node v labeled x_j it always holds $label(then(v)) = x_{n+j}$, i.e. the subgraph representing the positive cofactor has only one node labeled x_{n+j}. The negative cofactor with respect to x_j, $f_n(\overline{x}_1 \ldots \overline{x}_{j-1}\overline{x}_j)$, is independent of x_{n+j}, so $label(else(v)) = x_j + 1$. This cofactor can be constructed from $f_n(\overline{x}_1 \ldots \overline{x}_jx_{j+1})$ and $f_n(\overline{x}_1 \ldots \overline{x}_j\overline{x}_{j+1})$. Now the same argumentation can be applied for cofactorizing with respect to x_{j+1}. As a result to each variable corresponds a node in $G_{f_n}^1$, the BDD has $2n$ nodes. Since only cofactors of the form $f_n(\overline{x}_1 \ldots \overline{x}_{j-1}x_jx_{n+j})$ lead to '1' there are n one-paths in $G_{f_n}^1$.

The transformation of a BDD $G_{f_n}^1$ with respect to π_1 to a BDD $G_{f_n}^2$ (see Figure 2(b) for an example) with respect to π_2 takes n swap operations of pairs (x_i, x_{i+n}). Due to the canonicity of BDDs the order of swaps is arbitrary. Let therefore be (x_n, x_{2n}), ..., (x_1, x_{n+1}) the order of pairs. The swap operations are local operations and are of the type that is shown in Figure 3. Before the swap operation there are

$$\mathcal{P}_1(G_{g_{i-1}}^1) = 1 + \mathcal{P}_1(G_{g_i}^1)$$

one-paths from the topmost node involved, afterwards there are

$$\mathcal{P}_1(G_{g_{i-1}}^2) = 1 + 2 * \mathcal{P}_1(G_{g_i}^1)$$

Table 1: Comparison to other techniques

Benchmark	#in	#out	PLA	ESPRESSO	ESPRESSO -Ddisjoint	[6]	[14]	$\mathcal{P}_1$
alu4	14	8	1028	575	3551	na.	na.	1545
b12	15	9	431	43	691	57	na.	60
clip	9	5	167	120	359	162	na.	262
co14	14	1	47	14	14	na.	14	14
majority	5	1	5	5	10	na.	5	5
max1024	10	6	1024	274	775	na.	na.	444
misex1	8	7	32	12	18	15	na.	34
misex2	25	18	29	28	29	28	na.	30
mlp4	8	8	256	128	206	na.	na.	203
rd53	5	3	32	31	31	31	na.	35
rd73	7	3	141	127	127	127	na.	147
rd84	8	4	256	255	255	na.	na.	294
sao2	10	4	58	58	199	98	na.	97
sym10	10	1	837	210	367	na.	240	240
xor5	5	1	16	16	16	na.	16	16
x7dn	66	15	622	538	1697	na.	na.	1091
5xp1	7	10	75	65	99	70	na.	82
9sym	9	1	87	86	209	166	148	148
t481	16	1	481	481	2139	na.	2139	1009

one-paths from the topmost node involved in the swap as can be seen in the figure.
Swapping (x_n, x_{2n}) does not change the number of paths, since g_n is the '0'-function (i.e. $G^1_{g_n} = G^2_{g_n} = G_{g_n}$), the number of one-paths in the subgraph is $\mathcal{P}_1(G^2_{g_{n-1}}) = 1$. Proceeding with (x_{n-1}, x_{2n-1}) results in

$$\mathcal{P}_1(G^2_{g_{n-2}}) = 1 + 2 * \mathcal{P}_1(G^2_{g_{n-1}}).$$

Recursively applying this scheme $\mathcal{P}_1(G^2_{g_0})$ can be calculated:

$$\mathcal{P}_1(G^2_{g_0}) = 1 + 2 * (\ldots 1 + 2 * (1 + \mathcal{P}_1(G_{g_n}))\ldots)$$

Knowing that $g_n = 0$ the result is $\mathcal{P}_1(G^2_{g_0}) = 2^n - 1$. Since $g_0 = f_n$ the BDD of f_n with respect to π_2 has $2^n - 1$ one-paths. ■

4. EXPERIMENTAL RESULTS

In our experiments we used the algorithm for minimization of one-paths in BDDs that has been introduced in [7]. This algorithm effectively carries out the sifting algorithm with the goal of minimizing the number of one-paths instead of nodes of a BDD. It keeps track of changes in the number of one-paths due to a swap operation and propagates them thereafter.
The experiments show the usability of BDDs in DSOP minimization. At first the application of BDDs to find a DSOP representation was compared to other techniques. In a second series of experiments a set of larger circuits was investigated.
To minimize the BDDs with respect to one-paths we executed the algorithm described in [7] on a Pentium II system at 400 MHz with 128 MB of physical memory under Linux.
Table 1 gives the results for some Benchmarks. Besides the number of inputs and outputs the number of terms needed by the PLA-realization given by the circuits specification through the file is shown. The next column shows the results of optimizing the circuit with ESPRESSO, i.e. the number of monoms in a "Sum of Product"-representation. The smallest SOP-representation is always smaller or equal in size to the smallest DSOP-representation of a Boolean function. ESPRESSO heuristically minimizes the SOP.
Column "ESPRESSO -Ddisjoint" states the post-processing with ESPRESSOs "Ddisjoint"-option. This makes ESPRESSO generate a DSOP from the previously found SOP. "$\mathcal{P}_1$" is the result of our approach. In most cases the DSOP derived from the BDD is smaller than that generated by ESPRESSO. Especially regard "b12", where the result of ESPRESSO is 10 times larger. In total only 53 % of the cubes are needed when using the new approach. Furthermore, ESPRESSO needed several minutes on some of the circuits while the BDD approach needed at most 1159ms on "alu4", 517ms on "x7dn" but less than 100ms on most of the benchmarks.
In [6] an algorithm was introduced that works on an explicit representation of cubes by arrays, the results are given in Table 1 as well. This algorithm often generates smaller DSOP representations, but still the sizes compared to the BDD approach differ slightly only. For "9sym" the result of the BDD approach is even smaller. The calculation took at most 43ms ("clip").
Another recent approach presented in [14] makes use of the Shannon decomposition as well and of the binateness of a variable. But still the algorithm works on an explicit representation of cubes as lists. In all cases but "t481" the BDD yielded the same size for the DSOP, for "t481" our result was only half the size. It took 22ms to calculate the result.
In a second series of experiments we applied the BDD approach to some of the circuits from the LGSynth93 set. Due to the implicit representation of cubes the BDD approach can handle larger examples than the other approaches, Table 2 states the results. The table gives the size of the initial BDD that is built from the network in the BLIF-file in column "initial". Then this initial BDD is reduced within the time shown to a BDD of the size in column "minimized". For BDDs the size in number of nodes as well as number of one-paths is printed.
Most of the Benchmarks are reduced in size, a special case is "i3" which does not change in the number of one-paths. Quite some of the circuits have a large number of monoms in the initial representation but this can be significantly reduced. Consider "frg1" for an example, after minimization only 0.05% of the monoms are needed. On average only 20% the monoms are used for the minimized DSOP.
The efficiency of the implicit representation of monoms by one-paths becomes clear when regarding those benchmarks with a DSOP representation of more than one million monoms. Though "i4" is

Table 2: Some LGSynth93-benchmarks

			initial		minimized		
Benchmark	#in	#out	#nodes	$\mathcal{P}_1$	#nodes	$\mathcal{P}_1$	time
C432	36	7	115684	2.55982e+07	19799	1.38274e+06	1548658 ms
apex2	39	3	13418	412174	2056	8286	39516 ms
cordic	23	2	653	92672	121	26330	250 ms
dalu	75	16	11178	989363	2620	2968	26505 ms
des	256	245	10771	499779	4788	381927	83391 ms
frg1	28	3	16817	323049	154	167	19162 ms
frg2	143	139	5104	22311	2165	6010	5084 ms
i3	132	6	30303	262148	7259	262148	112951 ms
i4	192	6	85855	1.96102e+08	1783	2.3053e+06	266375 ms
i8	133	81	10366	115186	2235	2230	6812 ms
i9	88	63	6687	20000	1723	2591	5087 ms
misex3	14	14	1301	7267	946	2303	1944 ms
misex3c	14	14	1104	9255	1287	3886	1114 ms
pair	173	137	42083	6.71276e+07	9634	106532	128306 ms
pdc	16	40	4087	65374	1002	9345	4091 ms
seq	41	35	49080	8.69103e+06	1372	1905	220867 ms
spla	16	46	1204	11002	793	708	667 ms
too_large	38	3	7083	65520	937	4200	8846 ms
x1	51	35	1916	10417	815	2226	523 ms

reduced by a factor of 85 still 2.3 million paths have to be represented by the data structure, obviously the iterations necessary for reduction would not be possible on an explicit representation. Another example is "seq" which is collapsed from more than 8 million to roughly 2000 monoms.
Still for most circuits the runtime is moderate with less than half an hour in the worst case ("C432") and less than five minutes in all other cases.

5. CONCLUSIONS AND FUTURE WORK

The application of BDDs was introduced as a technique to minimize the DSOP-representation of a Boolean function. The restrictions on the search space were shown to be of minor influence on the investigated benchmarks, since the algorithm performed as good or better than other methods. The efficiency of the BDD approach due to the implicit representation allows to minimize larger circuits than before.
It is focus of current work to use the new technique as a preprocessor, further reductions in the size of a DSOP may be gained by using structural techniques as in [16]. Taking the result as a starting point for ESOP-minimization in the fashion of [11] is another promising approach.

6. REFERENCES

[1] R.E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[2] R.E. Bryant. Binary decision diagrams and beyond: Enabeling techniques for formal verification. In *Int'l Conf. on CAD*, pages 236–243, 1995.

[3] R. Drechsler. *Formal Verification of Circuits*. Kluwer Academic Publishers, 2000.

[4] B.J. Falkowski. Calculation of rademacher-walsh spectral coefficients for systems of completely and incompletely specified boolean functions. In *IEEE Proceedings on Circuits*, pages 1698–1701, 1993.

[5] B.J. Falkowski and C.-H. Chang. Paired haar spectra computation through operations on disjoint cubes. In *IEEE Proceedings on Circuits, Devices and Systems*, pages 117–123, 1999.

[6] B.J. Falkowski, I. Schäfer, and C.-H. Chang. An effective computer algorithm for the calculation of disjoint cube representation of boolean functions. In *Midwest Symposium on Circuits and Systems*, pages 1308–1311, 1993.

[7] G. Fey and R. Drechsler. Minimizing the number of paths in BDDs. In *Int'l Workshop on Boolean Problems*, 2002.

[8] W. Günther and R. Drechsler. ACTion: Combining logic synthesis and technology mapping for MUX based FPGAs. *Journal of Systems Architecture*, 46(14):1321–1334, 2000.

[9] Y.-T. Lai, S. Sastry, and M. Pedram. Boolean matching using binary decision diagrams with applications to logic synthesis and verification. In *Int'l Conf. on CAD*, pages 452–458, 1992.

[10] C. Meinel, F. Somenzi, and T. Theobald. Linear sifting of decision diagrams and its application in synthesis. *IEEE Trans. on CAD*, 19(5):521–533, 2000.

[11] A. Mischenko and M. Perkowski. Fast heuristic minimization of exclusive-sums-of-products. In *Int'l Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 242–250, 2001.

[12] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.

[13] T. Sasao. EXMIN2: A simplification algorithm for Exclusive-OR-Sum-of products expressions for multiple-valued-input two-valued-output functions. *IEEE Trans. on CAD*, 12:621–632, 1993.

[14] L. Shivakumaraiah and M.Thornton. Computation of disjoint cube representations using a maximal binate variable heuristic. In *Southeastern Symposium on System Theory*, pages 417–421, 2002.

[15] M. Thornton, R. Drechsler, and D.M. Miller. *Spectral Techniques in VLSI CAD*. Kluwer Academic Publisher, 2001.

[16] Y. Ye and K. Roy. Graph-based synthesis algorithms for AND/XOR networks. In *Design Automation Conf.*, pages 107–112, 1997.