A Hybrid Approach Combining Symbolic and Structural Techniques 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— Up to now only monolithic approaches have been used to minimize Disjoint Sum-of-Products **(DSOPs). Here, a hybrid approach is introduced by tightly integrating** Binary Decision Diagrams **(BDDs) as a symbolic strategy with structural methods. BDDs have the advantage of an implicit representation of cubes, but not all DSOPs can directly be represented. The search problem and restrictions on the search space resulting from applying BDDs are theoretically analyzed. Afterwards the hybrid algorithm accounting for these restrictions by structural methods is explained.**

The hybrid approach performs as good as previous methods regarding the quality of the results on the one hand, but and can handle much larger functions on the other hand.

I. Introduction

A DSOP is a representation of a Boolean function as a sum of disjoint cubes. DSOPs are used in several applications in the area of CAD, e.g. the calculation of spectra of Boolean functions [4, 5, 16] 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 cubes have been introduced [6, 14] but are only applicable to small 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 logic synthesis [10, 7] and formal verification [2, 9] of integrated circuits. The advantage of this data structure is an implicit representation of cubes in the ON-set of a Boolean function.

A hybrid approach for the minimization of DSOPs relying on BDDs in combination with structural methods is introduced in this paper. At first BDDs are shown to be applicable to the problem of DSOP minimization. Given a BDD of a Boolean function it is straightforward to extract a DSOP representation, since one-paths in the BDD, i.e. paths from the root of the graph to the terminal 1, correspond to cubes. Moreover, different one-paths lead to disjoint cubes. The importance of choosing a "good" variable order to get a small DSOP is theoretically supported. Furthermore, we study the influence on the search space that results from the restriction to DSOPs that are represented by BDDs. It is shown that not for each function the minimal DSOP is reachable. By integrating the knowledge about non-minimal structures in the BDD into the minimization algorithm this restriction is compensated. Experiments show that the resulting hybrid approach often produces smaller, at least same-sized DSOPs as other techniques on the one hand and that it is applicable to much larger functions on the other hand.

The paper is organized as follows. Section II. briefly introduces the necessary notion of BDDs and one-paths and shows how DSOP-representation is related to BDDs. Section III. contains theoretical studies regarding the difficulty of the problem and constraints on the search space implied by BDDs. In Section IV. the hybrid minimization algorithm is discussed in detail. Next experimental results are given. Finally conclusions are drawn.

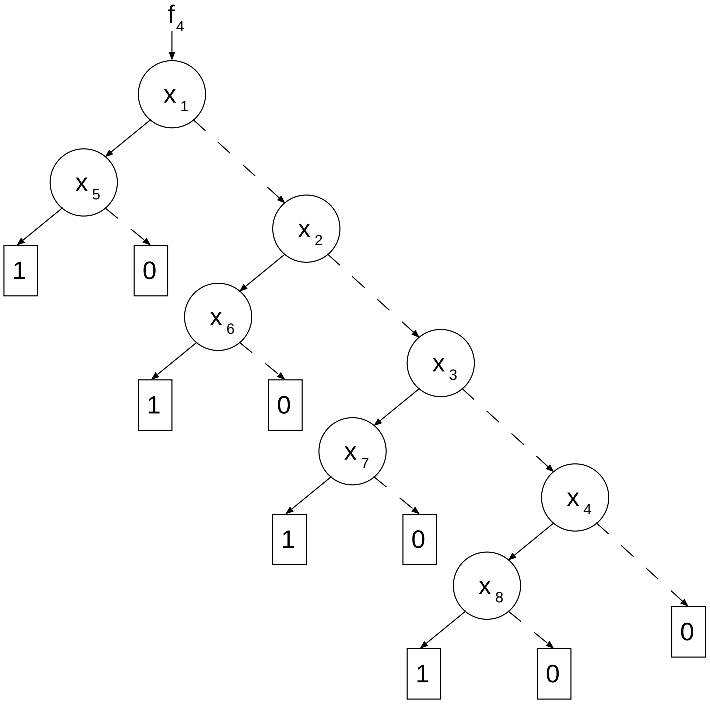
II. Preliminaries

A. Binary Decision Diagrams

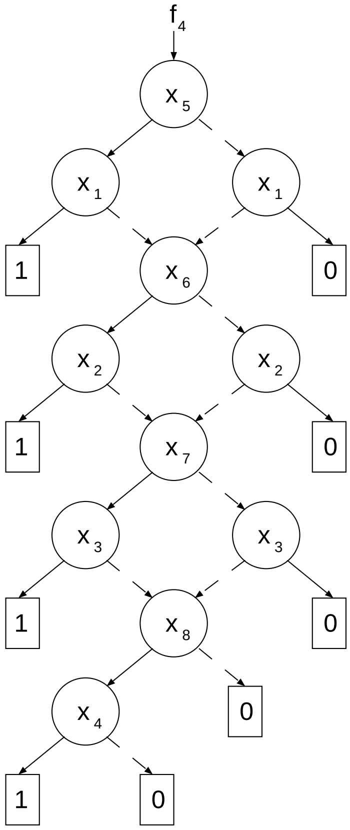
A BDD is a directed acyclic graph $G_f = (V, E)$ that represents a Boolean function f. The Shannon decomposition $g = x_i g_{x_i} + \overline{x}_i g_{\overline{x}_i}$ is carried out in each internal node v labeled with *label*$(v) = x_i$ of the graph, therefore v has the two successors *then*(v) and *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. In the sequel *BDD* refers to a reduced ordered BDD (as defined in [1]) and the size of a BDD is given by the number of nodes.

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

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



(a) $G^1_{f_4}$, variable order π_1



(b) $G^2_{f_4}$, variable order π_2

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

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

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 all different one-paths in the BDD* G_f.

B. BDDs and DSOP

Consider a BDD G_f representing the Boolean function $f(x_1, \ldots, x_n)$. A one-path $p = (v_0, \ldots, v_l)$ 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 $label(v) = x_i$. Therefore one of the cubes includes x_i, the other $\overline{x}_i$. Hence, the cubes m_{p_1} and m_{p_2} are disjoint.

Now the DSOP can easily be built by summing up all cubes 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 cube given above.*

By this, it is clear that the number of cubes in the DSOP represented by G_f is equal to $\mathcal{P}_1(G_f)$. Thus, as opposed to the usual goal of minimizing the number of nodes in a BDD, here the number of one-paths is minimized.

Known techniques to minimize the number of nodes can be used to minimize the number of paths by changing the objective function. One such technique is sifting [12]: A variable is chosen and moved to any position of the variable order based on exchange of adjacent variables. Then it is fixed at the best position (i.e. where the smallest BDD results), afterwards another variable is chosen. No variable is chosen twice during this process. An efficient technique for path-minimization based upon sifting has recently been proposed in [8].

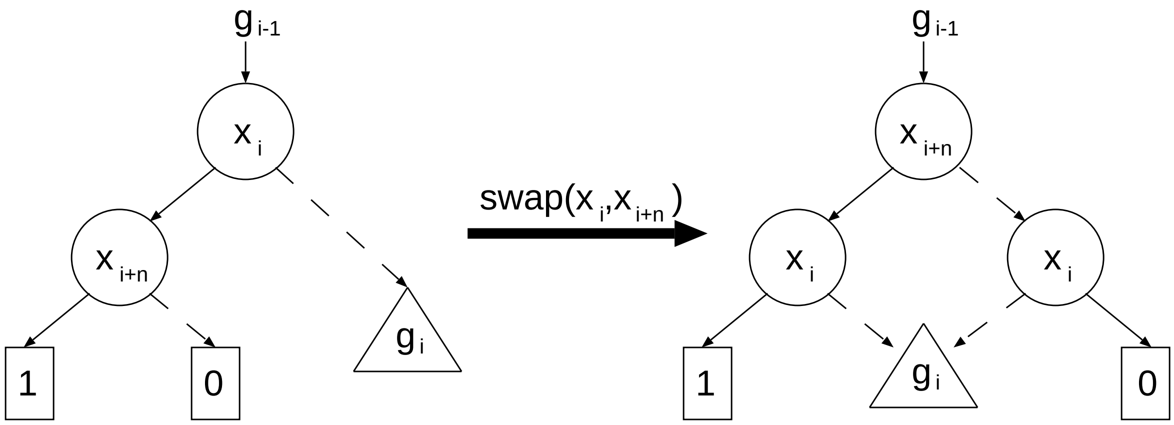


Fig. 2. The swap operation applied on a pair (x_i, x_{i+n}) of variables

III. Theory

In this section it is proven that the number of paths in a BDD heavily depends on the variable order. This establishes the need for a heuristical search algorithm to determine a "good" variable order. Second, the restrictions on the search space resulting from the usage of BDDs are investigated.

A. Dependence on the Variable Order

The relation between the size and number of paths is not well understood. It is known that there exist functions that have a BDD larger in size with less paths compared to a BDD of the same function with respect to another variable order [3].

It is also known that the variable order heavily influences the size of a BDD [1]. 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 1. *Let the Boolean function*

$$f_n(x_1, \ldots, x_{2n}), n > 1$$

be recursively defined:

- $f_1 = x_1 x_{n+1}$
- $f_j = f_{j-1} + (\bigcap_{i=1}^{j-1} \overline{x}_i) x_j x_{n+j}$

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

Proof. At first the variable order π_1, i.e. the BDD $G^1_{f_n}$ is considered (for an easier understanding of the proofs' flow see the example for $n = 4$ in Figure 1(a)). The nodes in $G^1_{f_n}$ 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:

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

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

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 $f_n(\overline{x}_1, \ldots, \overline{x}_j, x_{j+1})$ 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 $G^1_{f_n}$ and the BDD has $2n$ nodes. Because only cofactors of the form $f_n(\overline{x}_1, \ldots, \overline{x}_{j-1}, x_j, x_{n+j})$ lead to '1' there are n one-paths in $G^1_{f_n}$.

The transformation of a BDD $G^1_{f_n}$ with respect to π_1 to a BDD $G^2_{f_n}$ (see Figure 1(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 2. Before the swap operation there are

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

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

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

one-paths from the topmost node involved in the swap as can be seen in the figure.

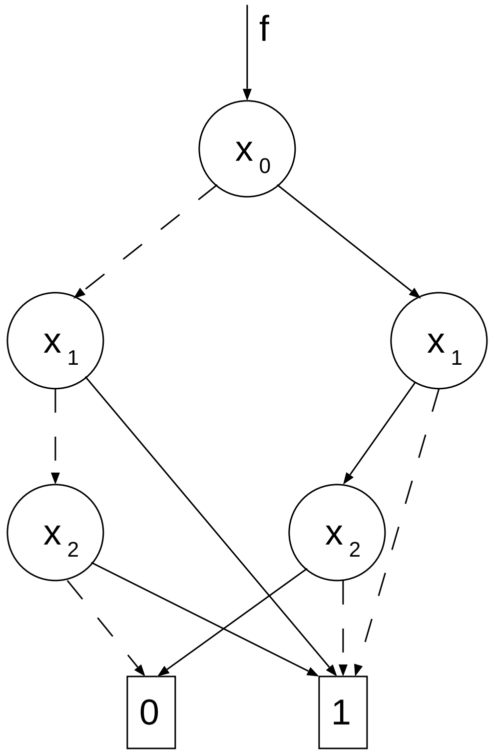


Fig. 3. BDD of $f = \neg(x_0 x_1 x_2 + \overline{x}_0\ \overline{x}_1\ \overline{x_2})$

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. ∎

B. Restrictions of the Search Space

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

Proof. Consider the totally symmetric function $f = \neg(x_0 x_1 x_2 + \overline{x}_0\ \overline{x}_1\ \overline{x}_2)$. Due to the symmetry of f all BDDs for f are isomorphic regardless of the variable order. As can be seen in Figure 3 a BDD of f has four one-paths. On the other hand the function can be represented by a DSOP with three cubes: $f = \overline{x}_0 x_1 + x_0 \overline{x}_2 + \overline{x}_1 x_2$ ∎

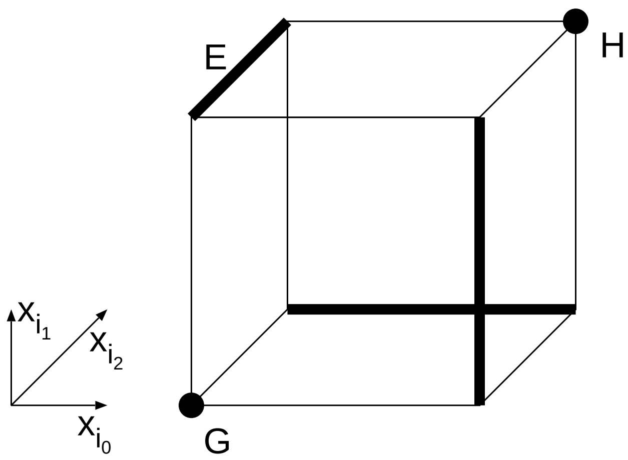


Fig. 4. Function F is determined by sub-functions E, G and H. Bold edges are determined by E.

IV. Hybrid Minimization Algorithm

The hybrid algorithm integrates the symbolic approach of calculating a minimal DSOP from a path-minimized BDD with structural methods. Structural methods use a predefined set of transformations to cross the search space by replacing a set of cubes with an equivalent set. In [6] operations like intersection and absorption are applied to retrieve disjoint cubes. In [11] exclusive SOPs are minimized by the use of the EXOR-Link operation that replaces a set of cubes by an equivalent set. Therefore, this structural methods carry out local transformations only. Our hybrid algorithm incorporates knowledge about local non-optimal structures in a BDD and replaces them with their optimal solution.

Figure 4 shows a 3-dimensional cube of a function F that has no BDD representing its optimal DSOP. One of the bold edges determined by E will be cut when carrying out the Shannon decomposition, regardless which variable is chosen to be first in the variable order. The cut edge has to be represented by two paths in the BDD, the two remaining edges by two more paths. Other functions that lead to non-optimal structures of three adjacent variables in the BDD have the same structure: All but two opposing vertices of the three-dimensional cube are determined by the same sub-function.

Remark 2. *All non-optimal structures for functions of three variables are isomorphic to the graph in Figure 5 with the exception that the function G may be equal to H (i.e. rooted at the same node).*

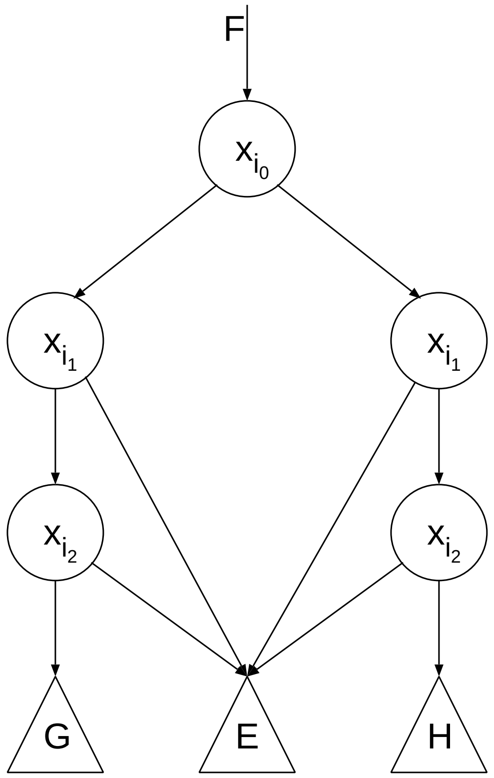


Fig. 5. Non-optimal structure

When traversing the BDD levelwise from top to bottom to count the number of paths, non-optimal structures are detected by graph-matching and the path-count is corrected to the known optimal value.

Example 1. *Consider function $g = x_3 f + \overline{x}_3 x_0 x_1 x_2$, where f is given by the proof of Lemma 2. All BDDs of this function have five one-paths. The BDD with variable order (x_2, x_0, x_1, x_3) is shown in Figure 6. When the x_2-node is reached during counting the number of paths a non-optimal structure is detected. Subfuction E is rooted at the left x_3-node, subfunctions G and H are rooted at the 0- and right x_3-node, respectively. Thus, the path-count of the left x_3-node is increased by three, the path-count of the 0- and right x_3-node by one, each. The path-count of intermediate nodes labeled x_0 and x_1 remains unchanged, i.e. zero. Because there are no other nodes labeled x_0 or x_1, counting proceeds with the x_0-level of the BDD.*

Functions of more than three variables that lead to non-optimal structures in the BDD do also exist. For example g under variable order (x_0, x_3, x_1, x_2) does not lead to a BDD with a structure isomorphic to the graph in Figure 5. Detection of these structures could further improve the result. This would be payed by a longer runtime for the additional checks needed and for this is not further considered in our implementation.

This scheme to calculate the path-count is incorporated

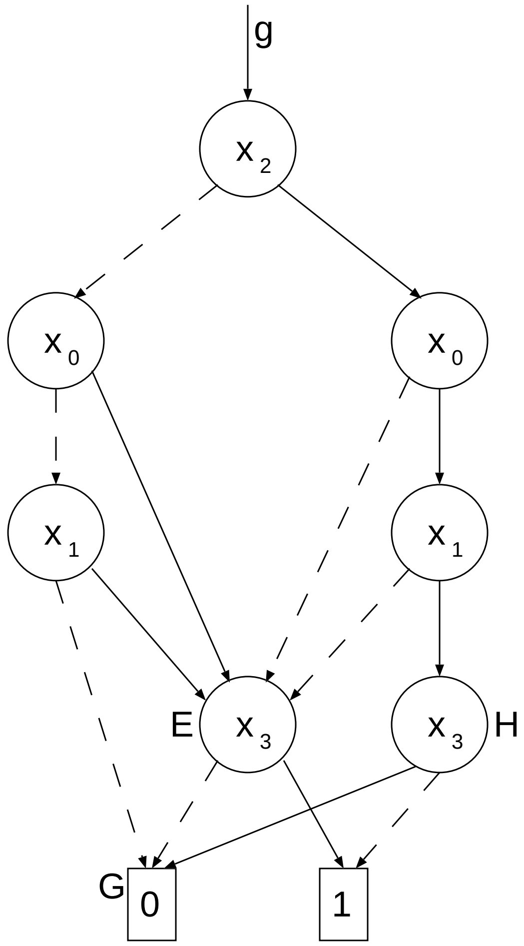


Fig. 6. BDD of $g = x_3 f + \overline{x}_3 x_0 x_1 x_2$ with variable order (x_2, x_0, x_1, x_3)

into sifting as the objective function. Sifting in this context is a heuristic that works globally. It detects a variable order for a recursive decomposition of the search space that leads to a small number of cubes to cover the given function. But here also knowledge about non-optimal structures is applied. Opposed to a pure structural approach this hybrid method is not directly restricted by the number of cubes that have to be represented.

V. Experimental Results

In our experiments we integrated the proposed approach as sifting algorithm into CUDD [15]. All non-optimal structures of three variables were considered. Experiments were carried out on a Pentium II system at 450 MHz with 256 MB of physical memory. The machine was running under Linux.

The experiments show the usability of the approach in DSOP minimization. At first the hybrid approach was compared to three other techniques. Finally, to show the scalability, the new algorithm was applied to larger benchmarks. Most benchmarks are taken from the LGSynth93 set.

TABLE IV
Circuits from the LGSynth93-set

Circuit	#in	#out	initial size		after minimization		time
			Nodes	Cubes	Nodes	Cubes	
C432	36	7	115684	2.56e+07	19799	1.38e+06	102 sec
apex3	54	50	53365	2.80e+06	958	727	30 sec
comp	32	3	203198	1.54e+06	3617	166606	33 sec
cordic	23	2	653	92672	121	19763	364 ms
dalu	75	16	11178	989363	2620	2968	26 sec
des	256	245	10771	499611	4788	381927	5.2 min
frg1	28	3	16817	323049	154	167	2.8 sec
misex3	14	14	1301	7261	946	2255	722 ms
my_adder	33	17	178963	6.59e+06	83	655287	32 sec
pair	173	137	42083	6.71e+07	9624	106134	14.2 min
phase_decoder	59	65	2936	9911	1080	1735	7 sec
s1423	92	79	272109	1.39e+08	11122	409238	22 min
seq	41	35	49080	8.69e+06	1372	1905	31 sec
too_large	38	3	7083	65520	937	4200	7.9 sec
x1	51	35	1916	10417	815	2226	4.5 sec

TABLE I
Comparison to ESPRESSO

Circuit	#in	#out	PLA	ESP	disjoint	hybrid
add6	12	7	1092	355	1092	517
alu4	14	8	1028	575	3551	1545
b12	15	9	431	43	691	60
clip	9	5	167	120	359	262
co14	14	1	47	14	14	14
inc	7	9	34	30	56	66
majority	5	1	5	5	10	5
max1024	10	6	1024	274	775	444
misex1	8	7	32	12	18	34
misex2	25	18	29	28	29	30
mlp4	8	8	256	128	206	203
rd53	5	3	32	31	31	35
rd73	7	3	141	127	127	147
rd84	8	4	256	255	255	294
root	8	5	256	57	154	79
sand	17	14	258	154	508	298
sao2	10	4	58	58	199	96
sym10	10	1	837	210	367	240
t481	16	1	481	481	2139	1009
xor5	5	1	16	16	16	16
x7dn	66	15	622	538	1697	1077
z9sym	9	1	420	86	190	148
5xp1	7	10	75	65	99	82
9sym	9	1	87	86	209	148

TABLE II
Comparison to [6]

Circuit	[6]	hybrid
b12	57	60
clip	162	262
misex1	15	34
misex2	28	30
rd53	31	35
rd73	127	147
sao2	98	96
5xp1	70	82
9sym	166	148

TABLE III
Comparison to [14]

Circuit	[14]	hybrid
co14	14	14
majority	5	5
sym10	240	240
t481	2139	1009
xor5	16	16
9sym	148	148

Table I gives the results for some Benchmarks and compares them to ESPRESSO. Besides the number of inputs and outputs, the number of terms needed by the PLA-realization given from 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 cubes 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 "disjoint" states the size of the DSOP generated by ESPRESSO when using ESPRESSOs "-Ddisjoint"-option. Column "hybrid" is the result of our approach. In most cases the DSOP is smaller than that generated by "ESPRESSO -Ddisjoint". Especially regard "b12", where the result of ESPRESSO is 10 times larger. In total only 53.5 % of the cubes were needed when using the new approach. Furthermore, ESPRESSO took several minutes on some of the circuits while the new approach took at most 4 seconds on "x7dn", 0.5 seconds on "alu4", but less than 100ms on most of the benchmarks.

The comparison to two other methods, [6] and [14], is given by Tables II and III, respectively. For these two methods only results on the benchmarks shown in the tables were available. In neither case exact runtimes were given. Both methods explicitly enumerate the cubes. In [6] an algorithm was introduced that works on an explicit representation of cubes by arrays. It then heuristically chooses cubes to merge. As shown in Table II this algorithm often generates smaller DSOP-representations, but still the sizes compared to the new approach differ slightly only. For "9sym" the result of the hybrid approach is even smaller.

The technique presented in [14] makes use of the Shannon decomposition as well. The binateness of a variable is used as a measure for choosing variables. But still the algorithm works on an explicit representation of cubes as lists. In all cases but "t481" the hybrid method yielded the same size for the DSOP, for "t481" our result was only half the size. It took 44ms to calculate this result.

Finally, Table IV gives the results on some larger benchmarks from the LGsynth93 set. Obviously enumerative methods would run into problems because of the large number of cubes. The processing time to execute the minimization is given in the last column of the table.

As can be seen there are up to 1.39e+08 cubes (initial size of "s1423") and after minimization roughly 400000 remain. Also consider "apex3" which is reduced to only 0.026 % of the initial size. But the technique can also handle circuits that can not be reduced to a small DSOP (consider "C432").

VI. Conclusions and Future Work

A hybrid approach to minimize the DSOP-representation of a Boolean function was introduced. A structural optimization is tightly integrated with a symbolic strategy. The benefit from the hybrid approach is a wider search space that can reduce the resulting DSOP significantly. Furthermore, the hybrid technique is applicable to circuits that are several orders of magnitude larger than those optimized by previous methods.

It is focus of current work to integrate the structural optimization within the efficient method for BDD minimization with respect to one-paths in [8]. This integration is not straightforward, since no information about incoming edges is easily available in an efficient implementation of a BDD package.

Taking the new method as a starting point for ESOP-minimization in the fashion of [11] is another promising approach.

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] E.V. Dubrova and D.M. Miller. On disjoint covers and ROBDD size. In *Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 162–164, 1999.

[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] F. Ferrandi, A. Macii, E. Macii, M. Poncino, R. Scarsi, and F. Somenzi. Symbolic algorithms for layout-oriented synthesis of pass transistor logic circuits. In *Int'l Conf. on CAD*, pages 235–241, 1998.

[8] G. Fey and R. Drechsler. Minimizing the number of paths in BDDs. In *Symposium on Integrated Circuits and System Design*, pages 359–364, 2002.

[9] Th. Kropf. *Introduction to Formal Hardware Verification.* Springer, 1999.

[10] 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.

[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] F. Somenzi. *CUDD: CU Decision Diagram Package Release 2.3.1.* University of Colorado at Boulder, 2001.

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