Disjoint-Support Boolean Decomposition Combining Functional and Structural Methods

Andrés Martinelli
andres@imit.kth.se

René Krenz
rene@imit.kth.se

Elena Dubrova
elena@imit.kth.se

Royal Institute of Technology, IMIT/KTH, Stockholm, Sweden

Abstract— This paper presents an algorithm for disjoint-support decomposition of Boolean functions which combines functional and structural approaches. First, a set of proper cut points is identified in the circuit by using dominator relations (structural method). Then, the circuit is partitioned along these cut points and a BDD-based decomposition is applied to the resulting smaller functions (functional method). Previous work on Boolean decomposition used only single methods and did not integrate a combined strategy. The experimental results show that the presented technique is more robust than a pure BDD-based approach and produces better-quality decompositions.

I. Introduction

Boolean decomposition is a technique used in many applications, including multi-level logic synthesis, testing, formal verification, combinatorial optimization problems over graphs and networks.

In general terms, the problem of decomposition of functions can be formulated as follows: Given a function f, express it as a composite function of some set of new functions. Often, a composite expression can be found in which the new functions are significantly simpler than f.

The basic type of decomposition is the *simple disjoint* decomposition where a function $f(X)$ is expressed as a composite function of two functions g and h, namely

$$f(X) = h(g(Y), Z) \tag{1}$$

where Y and Z are sets of variables forming a partition of the set of variables $X = \{x_1, x_2, \ldots, x_n\}$ of f. Every set of variables X for which a decomposition like (1) exists is called a *bound set* for f.

The fraction of all Boolean functions of n variables possessing simple disjoint decompositions of type (1) approaches zero as n approaches infinity [1, p. 90]. Therefore, a more general type of decomposition, known as *disjoint-support* (or *Roth-Karp* [2]) decomposition is usually considered. Disjoint-support decomposition has the form

$$f(X) = h(g(Y), Z)$$

with $f : \{0,1\}^n \rightarrow \{0,1\}$, $g : \{0,1\}^{|Y|} \rightarrow \{0,1,...,m-1\}$ and $h : \{0,1,...,m-1\} \times \{0,1\}^{|Z|} \rightarrow \{0,1\}$. The m-valued function h can be encoded by $k = \lceil \log_2 m \rceil$ Boolean functions $g_1, g_2, \ldots, g_k$, giving a representation of the form

$$f(X) = h(g_1(Y), g_2(Y), \ldots, g_k(Y), Z) \tag{2}$$

with all functions being Boolean. Disjoint-support decomposition includes as a special case non-disjoint decomposition. For example, if V is the set of overlapping variables of h and g_1, then the non-disjoint decomposition $f(X) = h(g_1(Y,V), V, Z)$ can be treated as a disjoint-support decomposition $f(X) = h(g_1(Y,V), g_2(V), Z)$, with g_2 being the identity function.

It is possible to extend algorithms for simple disjoint decomposition to the disjoint-support case. For example, [3] presents such an extension of the algorithm proposed in [4]. It is a BDD-based heuristic algorithm which quickly finds many disjoint-support decompositions and can handle large functions. One problem with this approach is that the decompositions found in this way do not necessarily simplify the function. For example, a circuit implemented as the two cofactors of a Shannon decomposition joined by a multiplexor is usually not optimal. Shannon decomposition is a special case of the decomposition (2), with $Z = x_1$, $g_1(Y) = f(0, x_2, \ldots, x_n)$, $g_2(Y) = f(1, x_2, \ldots, x_n)$, and $h = x_1' g_1(Y) + x_1 g_2(Y)$.

Another problem is that, in contrast to the case of simple disjoint decompositions, that are "too few", disjoint-support decompositions are "too many". So, an algorithm which first generates all disjoint-support decompositions and then checks which of them simplify the function is not feasible.

Our approach to overcome these problems is the following. First, a set of *proper cut* points is identified in a circuit representation of the function by applying a structural decomposition method. The circuit is partitioned along these cut points into a set of smaller sub-circuits which are treated independently. This allows us to reduce the search space for disjoint-support decompositions at the next stage, since all bound sets which overlap proper cut partitions are pruned. Finally, the overall decomposition is determined by combining the intermediate results.

We have chosen to use at the first stage of our algorithm a circuit-based technique rather than a BDD-based one, because manipulating circuits is much faster. Therefore, for functions with no proper cuts, the presented technique does not bring a significant overhead. The running time of our algorithm is normally similar, or even faster, than the running time of a BDD-based algorithm.

II. Previous work

The concept of *proper cuts* was first introduced in combinational equivalence checking [5]. Later it was applied to test-

ing [6, 7] and design for low power [8]. A vertex v is a proper cut if every path from any primary input in the cone of influence of v to the root contains v. The presented algorithm for finding proper cuts is based on the concept of *reduced dominator tree* constructed by using an extension of the Lengauer-Tarjan algorithm [9] for finding dominators in a graph. A proper cut is required to dominate all the primary inputs in its cone of influence. This guarantees that all re-converging paths are completely enclosed within the cone and, therefore, that those primary inputs belong to a bound set.

Disjoint-support decomposition was introduced by Roth and Karp [10]. They defined the notion of compatible classes describing the conditions for the existence of bound sets. Two assignments $x_1, x_2 \in B^{|Y|}$ are said to be *compatible with respect to the reference function* $f(Y, Z)$ if, for all $y \in B^{|Z|}$ such that $f(x_1, y)$ and $f(x_2, y)$ are defined, $f(x_1, y) = f(x_2, y)$. The set Y is a bound set if and only if $B^{|Y|}$ can be partitioned into $k \leq 2$ mutually compatible classes. If $f(X)$ is completely specified, then compatibility is an equivalence relation and k is the number of equivalence classes.

A number of BDD-based decomposition algorithms have been developed. Karplus [11] presented a technique for AND- and OR-type decomposition based on dominators in BDDs. It was extended by Yang et al [12] to XOR-type decompositions. Stanion and Sechen [13] target *quasi-algebraic* decomposition of the form $f(X) = g(Y) \odot h(Z)$, where "$\odot$" is any binary Boolean operation and $|Y \cup Z| = k$ for some $k \geq 0$. This type of decomposition is often referred to as *bi-decomposition* Bengtsson [4] developed a fast heuristic for simple disjoint decomposition which iteratively examines all linear intervals of variables of a ROBDD, and for every interval checks whether it is a bound set. This algorithm has been extended to disjoint-support decompositions in [3]. Minato and De Micheli [14] presented an algorithm which computes simple disjoint decompositions by generating irreducible sum-of-product for the function from its BDD and applying factorization. The algorithm of Bertacco and Damiani [15] makes a single traversal of the BDD to identify the simple disjoint decomposition of the co-factors and then combine them to obtain the decomposition for the entire function. The algorithm is impressively fast; however, as Sasao has observed in [16], it fails to compute some of the disjoint decompositions. This problem was corrected by Matsunaga [17], who added the missing cases in [15] allowing to treat the OR/XOR functions correctly. The algorithm [17] appears to be the fastest of existing exact algorithms for finding all simple disjoint decompositions.

III. Experimental results

All experiments were performed on a PC with a 2GHz Pentium4 CPU and 1024MByte main memory, running Linux Mandrake 8.2. We used a set of 188 combinational circuits from IWLS'02 benchmark set which comprises a total of 17633 outputs.

Due to space restrictions, the reader is referred to [18] and [3], for details on the algorithms used for the respective phases of the combined strategy.

Fig.1 shows a comparison of the running times of the pure BDD-based approach against the combined one. Each cross

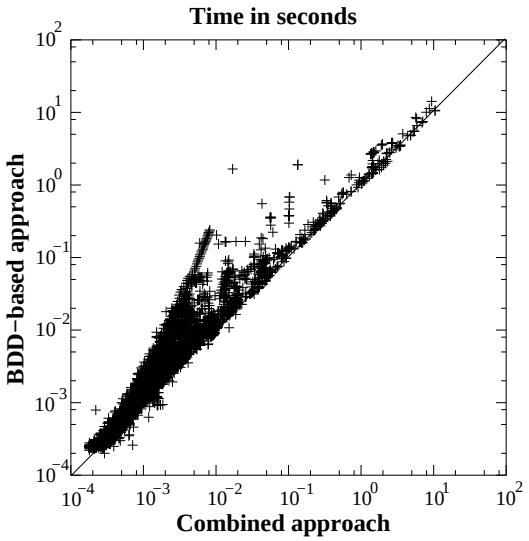

Fig. 1. Runtime comparison for the combined versus BDD-based approaches.

in the figure represents a single output function. Those above the line mark an improvement in the running time. As one can see, in the majority of cases, the combined approach is faster. Crosses below the line, representing cases where the running time of the combined tool is slower, are primarily circuits with no simple disjoint decomposition (i.e. no proper cuts), where the time spent on circuit exploration simply adds up as an overhead on the BDD-based algorithm.

Some representative results, aiming to show the number of disjoint-support decompositions computed by the combined approach, are given in Table III. The first three columns show information about the benchmarks: their name, the number of primary inputs and the number of primary outputs. Column 4 shows the number of proper cuts found in the first phase of the algorithm. Columns 5 to 7 show the number of k-bound sets found in the second phase, for different values of k, as the total sum of the results for individual outputs.

Notice that although columns 4 and 5 both show simple disjoint decompositions, the results they report do not overlap and should be considered separately. They respectively represent those decompositions found during the structural and the BDD-based phases of the algorithm, respectively. Since the heuristics used in each phase may not find all bound sets, and since they are dependent on the structure of the circuit and BDD ordering, the combination of the two can result in one finding bound sets which cannot be found by the other.

Also notice the cases like `cm42a`, `decod` or `parity`: in these, only zeroes are reported for the second phase of the algorithm. This is because after the partitioning along the cut points found in the first phase, the resulting functions only contain trivial disjoint-support decompositions, so the BDD-based algorithm is not invoked at all. This is one the reasons for the running time improvement.

TABLE I
EXPERIMENTAL RESULTS. NOTICE THAT 'PROPER CUTS' AND DISJOINT-SUPPORT CASE 'k=1' REPRESENT DIFFERENT SIMPLE DISJOINT DECOMPOSITIONS, FOUND IN THE FIRST AND THE SECOND PHASE RESPECTIVELY, AND SHOULD BE COUNTED SEPARATELY.

			bound sets			
benchmarks			classical		Roth-Karp	
name	in	out	proper cuts	k=1	k=2	k=3
9symml	9	1	0	0	10	17
alu2	10	6	1	3	55	89
alu4	14	8	0	2	141	263
apex2	39	3	0	9	57	119
apex6	135	99	229	9056	32520	36084
apex7	49	37	104	2814	8406	9435
b9	41	21	37	335	1320	1465
C1355	41	32	0	0	11624	21708
C1908	33	25	0	2758	5337	8279
C3540	50	22	18	79	676	1378
C432	36	7	16	0	97	259
C499	41	32	0	0	12404	22428
C880	60	26	57	204	1574	3150
cm150a	21	1	1	0	5	11
cm42a	4	10	20	0	0	0
cm85a	11	3	9	20	90	100
cmb	16	4	20	325	325	325
comp	32	3	7	108	520	696
cordic	25	2	0	18	71	95
count	35	16	136	136	544	544
decod	5	16	48	0	0	0
des	256	245	640	30527	202664	327911
e64	65	65	2016	0	0	0
f51m	8	8	0	26	85	123
frg2	143	139	1	14	29	31
lal	26	19	598	32927	142542	169797
misex2	25	18	50	237	1522	1550
mux	21	1	1	0	5	11
pair	173	137	889	28416	113431	173717
parity	16	1	14	0	0	0
rot	135	107	177	9159	45070	65873
seq	41	35	34	3951	15762	28193
term1	34	10	26	340	822	870
too_large	38	3	0	7	43	109
ttt2	24	21	10	843	2583	2665
x3	135	99	129	11980	31663	36259
x4	94	71	66	11066	30591	34785

IV. CONCLUSION

We present a decomposition technique which integrates circuit-based and BDD-based decompositions. The combination of the two approaches results in an algorithm which is more robust than the pure BDD-based method, regarding both quality of the result and running time.

Our experiments on benchmark circuits suggest that the developed algorithm has a significant potential for a large number of circuits. However, there are also limitations. The main one is that our method depends on dominator relations of the circuit. If the circuit under consideration has no internal dominators, the presented technique reduces to a BDD-based decomposition. We have found that the majority of practical circuit graphs contain a substantial number of internal dominator vertices (between 5 and 0.5 per input) which warrants an efficient performance of our algorithm. For circuits with no internal dominators, in the future we plan to use complementary methods for structuring the decomposition process, such as generalized dominators [19] and min-cut [20].

REFERENCES

[1] C. E. Shannon, "The synthesis of two-terminal switching circuits," *Bell Systems Technical J.*, no. 28, pp. 59–98, 1949.

[2] R. M. Karp, "Functional decomposition and switching circuit design," *Journal of Soc. Indust. Appl. Math.*, vol. 11, pp. 291–335, June 1963.

[3] A. Martinelli, T. Bengtsson, and A. J. Sullivan, "Roth-Karp decomposition of large Boolean functions with application to logic design," in *Proceedings of NORCHIP'02*, (Copenhagen, Denmark), November 2002.

[4] T. Bengtsson and A. Martinelli, "A BDD-based fast heuristic algorithm for disjoint decomposition," in *Proceedings of Asia and South Pacific Design Automation Conference, ASP-DAC03*, (Kitakyushu, Japan), January 2003.

[5] W. Donath and H. Ofek, "Automatic identification of equivalence points for boolean logic verification," *IBM Technical Disclosure Bulletin*, vol. 18, no. 8, pp. 2700–2703, 1976.

[6] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT-probabilistic estimation of digital circuit testability," in *Proceeding of International Symposium on Fault-Tolerant Computing*, pp. 220–225, June 1985.

[7] B. B. Bhattacharya and S. C. Seth, "On the reconvergent structure of combinational circuits with applications to compact testing," in *Proceeding of International Symposium on Fault-Tolerant Computing*, pp. 264–269, 1987.

[8] D. Cheng, "Power estimation of digital CMOS circuits and the application to logic synthesis for low power," December 1995. Pd.D. Thesis, University of California at Santa Barbara.

[9] T. Lengauer and R. E. Tarjan, "A fast algorithm for finding dominators in a flowgraph," *Transactions of Programming Languages and Systems*, vol. 1, pp. 121–141, July 1979.

[10] J. P. Roth and R. M. Karp, "Minimization over Boolean graphs," *IBM Journal*, vol. 6, pp. 227–238, April 1962.

[11] K. Karplus, *Using if-then-else DAGs for multi-level logic minimization*. University of California Santa Cruz: Technical Report UCSC-CRL-88-29, 1988.

[12] C. Yang, V. Singhal, and M. Ciesielski, "Bdd decomposition for efficient logic synthesis," in *Proceedings of International Conference on Computer Design*, pp. 626–631, 1999.

[13] T. Stanion and C. Sechen, "Quasi-algebraic decompositions of switching functions," in *Proceedings of Sixteenth Conference on Advanced Research in VLSI*, pp. 358–367, IEEE, 1995.

[14] S. Minato and G. D. Micheli, "Finding all simple disjunctive decompositions using irredundant sum-of-products forms," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 111–117, 1998.

[15] V. Bertacco and M. Damiani, "The disjunctive decomposition of logic functions," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 78–82, 1997.

[16] T. Sasao and M. Matsuura, "DECOMPOS: An integrated system for functional decomposition," in *Proceedings of ACM/IEEE International Workshop on Logic Synthesis*, 1998.

[17] Y. Matsunaga, "An exact and efficient algorithm for disjunctive decomposition," in *Proceedings of SASIMI'98*, pp. 44–50, 1998.

[18] R. Krenz, "On-the-fly proper cut recognition based on circuit graph analysis," in *Proceedings of NORCHIP'02*, (Copenhagen, Denmark), November 2002.

[19] R. Gupta, "Generalized dominators and post-dominators," in *Proceedings of 19th Annual ACM Symposium on Principles of Programming Languages*, pp. 246–257, 1992.

[20] J. Cong, H. P. Li, S. K. Lim, T. Shibuya, and D. Xu, "Large scale circuit partitioning with loose/stable net removal and signal flow based clustering," in *Inetrantional Conference on Computer-Aided Design*, pp. 441–446, 1997.