Restricted Simple Disjunctive Decompositions Based on Grouping Symmetric Variables

Hiroshi Sawada, Shigeru Yamashita and Akira Nagoya
NTT Communication Science Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, JAPAN
{sawada, ger, nagoya}@cslab.kecl.ntt.co.jp

Abstract

This paper presents an efficient method for a simple disjunctive decomposition, where candidates for the bound set are restricted to sets of symmetric variables to reduce the computation cost. Symmetric variables are detected by depth-first traversals of an ordered binary decision diagram (OBDD), and decompositions are carried out by changing the variable order of the OBDD. We do not change the variable order until the change is really needed. Experimental results show that even if the decomposition form was restricted, many practical functions could be decomposed. The execution time for decomposition was very small even for functions with many variables. Combined with an exhaustive search, the method successfully decomposed some functions that could not be decomposed by exhaustive search alone in a practical amount of time.

1. Introduction

Finding a simple disjunctive decomposition form $f(X) = g(h(X^B), X^F)$ of a Boolean function helps to synthesize an optimal multi-level logic circuit. In the decomposition form, X is a set of variables, $\{X^B, X^F\}$ is a partition of X, and X^B is called a bound set. The form is called "simple" because h is a single-output function and "disjunctive" because X^B and X^F have no common variables.

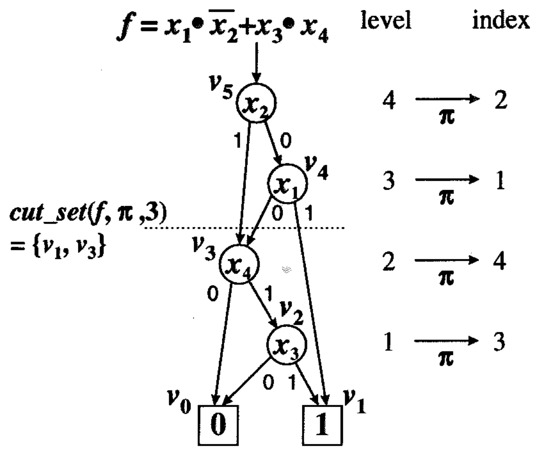
The fundamental concept of a simple disjunctive decomposition has been studied by Ashenhurst [1]. He used a decomposition chart, whose size grows exponentially to the number of variables, to check the existence of a decomposition. Roth and Karp [11] have proposed a decomposition method using a sum of product form. Recently, several researchers [3, 7, 13] have proposed decomposition algorithms based on OBDDs (Ordered Binary Decision Diagrams [2]). Although the decomposition test after a bound set was decided can be done efficiently, a difficulty still lies in finding an appropriate bound set. The number of possible bound sets grows exponentially to the number of variables, thus an exhaustive search is difficult for functions with many variables.

In this paper, we propose a restricted simple disjunctive decomposition. The decomposition form is restricted such that the bound set is a set of symmetric variables. The purpose of the restriction is to reduce the computation cost for finding an appropriate bound set, so that the decomposition can be applied to functions with many variables. We consider a set of symmetric variables to be a good candidate for a bound set that gives a simple disjunctive decomposition. In addition, if the set of symmetric variables satisfies certain properties, we can identify the existence of a simple disjunctive decomposition without doing the decomposition tests described above.

Symmetry of a Boolean function is a property that the function is invariant under any permutation among a subset of variables. Knowledge of the symmetry is useful, for example, in logic synthesis [5] and in Boolean matching [4]. Recently, several efficient techniques to detect symmetry have been proposed. Möller et al. [9] have proposed the idea of asymmetry, which can be obtained from the structure of an OBDD, to filter out the possibility of symmetry. Panda et al. [10] have combined the symmetry check of two adjacent variables and the dynamic variable ordering of an OBDD [12]. Tsai et al. [15] have used generalized Reed-Muller forms for symmetry detection. In our method, we will follow the idea of asymmetry to detect symmetric variables.

The form of an OBDD representing a function depends on the variable order, so it is important to find a good variable order. In a simple disjunctive decomposition test using an OBDD, all variables in the bound set are moved up to higher levels (nearer to the root node) than variables in the free set. Since changing the variable order in an OBDD is an expensive operation, we give attention to reducing the number of times the variable order is changed.

This paper is organized as follows. In Section 2, we give some definitions of Boolean functions, OBDDs, simple



$f = x_1 \bullet \overline{x}_2 + x_3 \bullet x_4$
level
index
v_5
x_2
v_4
x_1
$cut\_set(f, \pi, 3)$
$= \{v_1, v_3\}$
v_3
x_4
v_2
x_3
v_0
v_1
0
1
$4 \xrightarrow[\pi]{} 2$
$3 \xrightarrow[\pi]{} 1$
$2 \xrightarrow[\pi]{} 4$
$1 \xrightarrow[\pi]{} 3$

Figure 1. An ordered binary decision diagram

disjunctive decomposition, and symmetry. In Section 3, we discuss a procedure for the restricted simple disjunctive decomposition whose bound set is a set of symmetric variables. In Section 4, experimental results are presented. We conclude this paper in Section 5.

2. Preliminaries

2.1. Boolean functions and ordered binary decision diagrams

Let $f(x_1,\ldots,x_n)$: $\{0,1\}^n \rightarrow \{0,1\}$ be a completely specified Boolean function. The **cofactors** of f with respect to x_i=1 and x_i=0 are f_{x_i}=$f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n)$ and $f_{\overline{x}_i}$=$f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n)$, respectively. In this paper, we only consider variables that a function f essentially depends on, thus $f_{x_i} \neq f_{\overline{x}_i}$ for any variable x_i in f.

An ordered binary decision diagram (OBDD) [2] is a directed acyclic graph representing Boolean functions (Figure 1). An OBDD has two kinds of nodes: variable nodes and constant nodes. A constant node represents a Boolean constant 0 or 1. A variable node v is associated with a Boolean variable x_i and represents a function f^v. It has two outgoing edges labeled with 0 and 1, which point the nodes representing functions $f^v_{\overline{x}_i}$ and $f^v_{x_i}$, respectively.

When traversing from any variable node to a constant node according to the cofactors of variables, each variable must occur at most only once and in a given order. We define a **level** of a variable node as follows: if there exists an edge from a variable node v_i to another variable node v_j, the level of v_i is more than that of v_j. We also define a **variable order** π to be a one-to-one mapping from levels to indexes of variables.

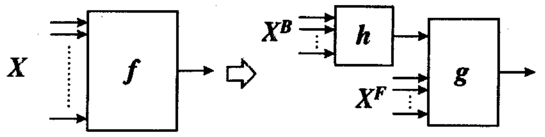


Figure 2. A simple disjunctive decomposition

2.2. Simple disjunctive decomposition

A simple disjunctive decomposition of a function $f(X)$ is of the form (Figure 2)

$$f(X) = g(h(X^B), X^F),$$

where X^B and X^F are sets of variables such that $X^B \cup X^F = X$ and $X^B \cap X^F = \emptyset$, and g: $\{0,1\}^{|X^F|+1} \rightarrow \{0,1\}$ and h: $\{0,1\}^{|X^B|} \rightarrow \{0,1\}$ are completely specified Boolean functions. The sets X^B and X^F are called the **bound set** and the **free set**, and g is called the **image**; we call h the **subfunction** in this paper.

The fundamental concept of a simple disjunctive decomposition has been studied by Ashenhurst [1]. Recently, several researchers have proposed OBDD-based algorithms for decomposition. We use the following definition and theorem found in [7].

Definition 1 In the OBDD of a function f with a variable order π, let $cut\_set(f,\pi,l)$ denote the set of nodes whose levels are less than l, and that have edges from nodes in levels greater than or equal to l. □

Theorem 1 For an n-variable function $f(x_1,\ldots,x_n)$ and a variable order π, if $|cut\_set(f,\pi,l)| \leq 2$, there exists a simple disjunctive decomposition of f whose bound set and free set are $X^B = \{x_{\pi(n)},\ldots,x_{\pi(l)}\}$ and $X^F = \{x_{\pi(l-1)},\ldots,x_{\pi(1)}\}$. □

For the function f in Figure 1, $cut\_set(f,\pi,3) = \{v_1, v_3\}$. Since $|cut\_set(f,\pi,3)| = 2$, there exists a simple disjunctive decomposition of the form $f = g(h(x_2,x_1),x_4,x_3)$.

2.3. Symmetry

A function $f(x_1,\ldots,x_n)$ is **symmetric** in $\{x_i,x_j\}$ (or $\{x_i,\overline{x}_j\}$) if the interchange of x_i and x_j (or $\overline{x}_j$) leaves the function invariant. For example, the function f in Figure 1 is symmetric in $\{x_3,x_4\}$ and in $\{x_1,\overline{x}_2\}$. A function f is symmetric in $\{x_i,x_j\}$ (or $\{x_i,\overline{x}_j\}$) if and only if $f_{x_i\overline{x}_j} = f_{\overline{x}_i x_j}$ (or $f_{x_i x_j} = f_{\overline{x}_i\overline{x}_j}$).

A function f is symmetric in a subset X' of input variables or their complements if any permutation of the subset leaves the function invariant. The set X' is called a **maximal symmetry group** [8] (a maximal set of symmetric variables) of f if there is no variable or its complement $x_i \notin X'$ or $\overline{x}_i \notin X'$ such that f is also symmetric in $x_i \cup X'$ or $\overline{x}_i \cup X'$.

For a complete specified function, symmetry is an equivalence relation. Thus, symmetry in a subset of variables can be calculated from a set of symmetries in two variables. Examples are listed below.

If a function f is symmetric in $\{x_i, x_j\}$ and $\{x_j, x_k\}$, then it is symmetric in $\{x_i, x_j, x_k\}$.
If a function f is symmetric in $\{x_i, \overline{x}_j\}$ and $\{x_j, x_k\}$, then it is symmetric in $\{x_i, \overline{x}_j, \overline{x}_k\}$.
If a function f is symmetric in $\{x_i, \overline{x}_j\}$ and $\{x_j, \overline{x}_k\}$, then it is symmetric in $\{x_i, \overline{x}_j, x_k\}$. □

In addition, we use the following definitions found in [5]. A function f is **multiform symmetric** in variables x_i and x_j if f is symmetric in both $\{x_i, x_j\}$ and $\{x_i, \overline{x}_j\}$. A function f is **single-variable symmetric** in x_j in the space x_i=0 if x_j's two cofactors of $f_{\overline{x}_i}$ are identical; $f_{\overline{x}_i\overline{x}_j} = f_{\overline{x}_i x_j}$. A function f is **single-variable symmetric** in x_j in the space x_i=1 if x_j's two cofactors of f_{x_i} are identical; $f_{x_i\overline{x}_j} = f_{x_i x_j}$.

Proposition 1 Let a function f have a maximal symmetry group. If f is multiform symmetric in a pair of variables in the maximal symmetry group, then f is multiform symmetric in all pairs of variables in the maximal symmetry group. □

Proposition 2 Let a function f have a maximal symmetry group. If f is single-variable symmetric in a pair of variables in the maximal symmetry group, then f is single-variable symmetric in all pairs of variables in the maximal symmetry group. □

3. Decomposition procedure

3.1. The overall procedure

In this section, we show our procedure for the restricted simple disjunctive decomposition. The decomposition form is restricted such that the bound set is a maximal symmetry group. Because of the restriction, we can offer an efficient procedure based on symmetry detection and variable reordering in an OBDD. The point of our procedure is that we do not change the variable order until the change is really needed. This is because variable reordering is an expensive operation. Overall, the procedure is as follows.

1. Construct an OBDD representing a function f to be decomposed.
2. Examine the symmetry, including multiform symmetry. We can then check a decomposition whose subfunction is implemented in an XOR gate. If it can be decomposed, stop here.
3. Change the variable order of the OBDD such that variables in a maximal symmetry group are adjacent in the new order, and examine the single-variable symmetry. We can now check a decomposition whose subfunction is implemented in an AND gate and some inverters. If it can be decomposed, stop here.
4. Move all variables in a maximal symmetry group up to higher levels than the other variables. Then, we can check a decomposition whose bound set is the maximal symmetry group.

3.2. Symmetry detection and decomposition with an XOR gate

We firstly examine the symmetry of f in an OBDD representation whose variable order is π. Let n be the number of variables of f. Since symmetry is an equivalence relation, all we have to do for a complete check of the symmetry is the following. For all $x_{\pi(i)}$ $(n \geq i \geq 2)$, examine whether there exists a variable $x_{\pi(j)}$ such that f is symmetric in $\{x_{\pi(i)}, x_{\pi(j)}\}$ and/or $\{x_{\pi(i)}, \overline{x}_{\pi(j)}\}$ and j is maximum under the condition $i > j$.

To check symmetry of two variables, we use the idea of asymmetry proposed in [9, 10] to filter out the possibility of symmetries.

Theorem 2 Let a function f be represented in an OBDD whose variable order is π. f is asymmetric in two variables $x_{\pi(i)}$ and $x_{\pi(j)}$ $(i > j)$, if a node at level i does not have any successor at level j, or if a node at level j can be reached from the root node traversing a path that does not contain any node at level i. □

The above asymmetry conditions can be examined by depth-first traversals in the OBDD. At the same time, we can examine the symmetry of two adjacent variables $\{x_{\pi(i)}, x_{\pi(i-1)}\}$ or $\{x_{\pi(i)}, \overline{x}_{\pi(i-1)}\}$ by checking $f^v_{x_{\pi(i)}\overline{x}_{\pi(i-1)}} = f^v_{\overline{x}_{\pi(i)}x_{\pi(i-1)}}$ or $f^v_{x_{\pi(i)}x_{\pi(i-1)}} = f^v_{\overline{x}_{\pi(i)}\overline{x}_{\pi(i-1)}}$ for all nodes v at level i.

As for pairs of variables that are not filtered out by the asymmetry check, we examine the symmetries according to the following theorem. The examination can also be done by depth-first traversals in the OBDD.

Theorem 3 Let a function f be represented in an OBDD whose variable order is π. f is symmetric in two variables $\{x_{\pi(i)}, x_{\pi(j)}\}$ or $\{x_{\pi(i)}, \overline{x}_{\pi(j)}\}$ $(i > j)$, if every node v at level i satisfies $f^v_{x_{\pi(i)}b_{i-1}\cdots b_{j+1}\overline{x}_{\pi(j)}} = f^v_{\overline{x}_{\pi(i)}b_{i-1}\cdots b_{j+1}x_{\pi(j)}}$ or $f^v_{x_{\pi(i)}b_{i-1}\cdots b_{j+1}x_{\pi(j)}} = f^v_{\overline{x}_{\pi(i)}b_{i-1}\cdots b_{j+1}\overline{x}_{\pi(j)}}$ for all

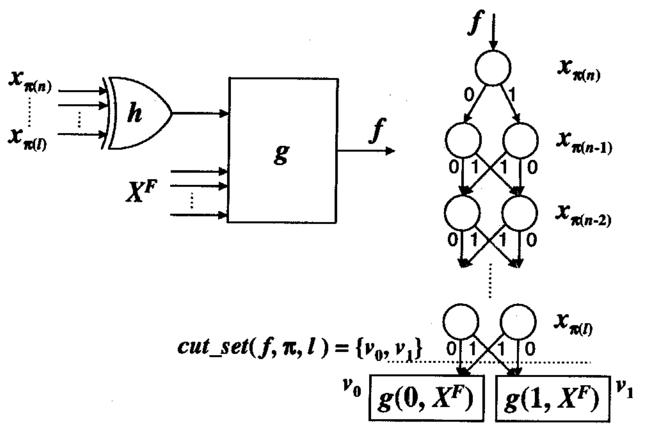


Figure 3. Decomposition with an XOR Gate

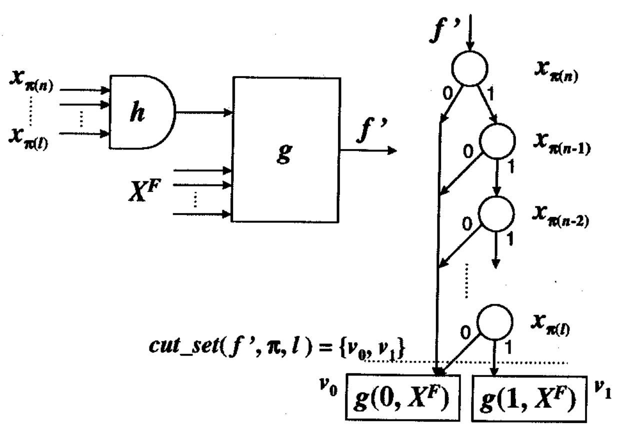


Figure 4. Decomposition with an AND Gate

cofactors of variables in levels ranging from $i-1$ to $j+1$ (b_k is either $x_{\pi(k)}$ or $\overline{x}_{\pi(k)}$). □

Now we obtain information about symmetry, including multiform symmetry. We can then check whether there exists a restricted simple disjunctive decomposition whose subfunction is implemented in an XOR gate.

Proposition 3 If a function f has a maximal symmetry group containing multiform symmetric variables, there exists a simple disjunctive decomposition of f whose bound set is the maximal symmetry group and the subfunction is implemented in an XOR gate. □

Figure 3 shows the interpretation of this proposition. Suppose that the variables in the maximal symmetry group are moved up to levels ranging from n to l. Because the variables are multiform symmetric, $f^v_{x_{\pi(i)}\overline{x}_{\pi(i-1)}} = f^v_{\overline{x}_{\pi(i)}x_{\pi(i-1)}}$ and $f^v_{x_{\pi(i)}x_{\pi(i-1)}} = f^v_{\overline{x}_{\pi(i)}\overline{x}_{\pi(i-1)}}$ for any node v at level i ($n \geq i \geq l$+1), which lead to $|cut\_set(f, \pi, l)| = 2$. According to Theorem 1, there exists a simple disjunctive decomposition. The OBDD of the subfunction h is obtained by replacing the nodes representing $g(0, X^F)$ and $g(1, X^F)$ into Boolean constants 0 and 1, respectively. We can see that the subfunction is implemented with an XOR gate.

Note that we do not change the variable order while checking for multiform symmetry. After we find the existence of multiform symmetric variables, we change the variable order and decompose the function.

3.3. Grouping symmetric variables and decomposition with an AND gate

If there exist no pair of multiform symmetric variables, we change the variable order of the OBDD such that variables in a maximal symmetry group are adjacent in the new order. When two variables $x_{\pi(i)}$ and $x_{\pi(j)}$ are symmetric ($i > j$), we apply $jumpup(j, i-1)$ operation, which moves the variable at level j to level $i-1$ and decreases the level of every variable from level $i-1$ to level j+1 by 1.

Now we can examine whether all pairs of variables in a maximal symmetry group are single-variable symmetric by checking the following condition. For a pair of two adjacent variables $x_{\pi(i)}$ and $x_{\pi(i-1)}$ in the maximal symmetry group, $f^v_{\overline{x}_{\pi(i)}\overline{x}_{\pi(i-1)}} = f^v_{\overline{x}_{\pi(i)}x_{\pi(i-1)}}$ or $f^v_{x_{\pi(i)}\overline{x}_{\pi(i-1)}} = f^v_{x_{\pi(i)}x_{\pi(i-1)}}$ for all nodes v at level i. We can then check a decomposition whose subfunction is implemented in an AND gate and some inverters.

Let us invert some of the variables in the maximal symmetry group such that the variable x_i has the property that f is single-variable symmetric in a variable x_j in the space x_i=0. That is, if f is single-variable symmetric in a variable x_j in the space x_i=1, invert the variable x_i. Let the resultant function be f'.

Proposition 4 If a function f' has a maximal symmetry group containing single-variable symmetric variables, there exists a simple disjunctive decomposition of f' whose bound set is the maximal symmetry group and the subfunction is implemented in an AND gate. □

Figure 4 shows the interpretation of this proposition. Suppose that the variables in the maximal symmetry group are moved up to levels ranging from n to l. Because the variables are single-variable symmetric, $f^v_{\overline{x}_{\pi(i)}\overline{x}_{\pi(i-1)}} = f^v_{\overline{x}_{\pi(i)}x_{\pi(i-1)}} = f^v_{x_{\pi(i)}\overline{x}_{\pi(i-1)}}$ for any node v in level i ($n \geq i \geq l$+1), which leads to $|cut\_set(f, \pi, l)| = 2$. According to Theorem 1, there exists a simple disjunctive decomposition. The OBDD of the subfunction h is obtained by replacing the nodes representing $g(0, X^F)$ and $g(1, X^F)$ into Boolean constants 0 and 1, respectively. We can see that the subfunction is implemented with an AND gate.

Note that we do not change the variable order such that the maximal symmetry group is in levels ranging from n to l while checking for single-variable symmetry. After we find that there exist single-variable symmetric variables, we change the variable order and decompose the function.

3.4. Moving up variables in a maximal symmetry group

If there exists neither multiform symmetric variables nor single-variable symmetric variables in any maximal symmetry group, we move the variables in a maximal symmetry group up to higher levels than the other variables. Let *gtop* be the highest level of variables in the maximal symmetry group and $gsize$ be the size of the group. Until *gtop* becomes n, we repeatedly apply $jumpdown(gtop + 1, gtop + 1 - gsize)$ operation, which moves the variable at level *gtop*+1 to level $gtop{+}1{-}gsize$ and increase the level of every variable in the maximal symmetry group by 1. Then, according to Theorem 1, we can examine whether there exists a simple disjunctive decomposition whose bound set is the maximal symmetry group.

4. Experimental results

The procedure for the restricted simple disjunctive decomposition presented so far has been implemented. We performed experiments on combinational (two-level or multi-level) circuits of the MCNC benchmark [16]. From the circuit description, we constructed OBDDs of primary outputs in terms of primary inputs with the initial variable order obtained by heuristics [6]. We then applied the procedure to each of the single-output functions. If a decomposition form was found, we applied the procedure recursively to the image and the subfunction of the decomposition.

Table 1 shows the experimental results. Due to space limitations, we had listed the results of only one primary output for each circuit. We selected the primary output whose function had the largest number of variables. The columns "name", "po" and "#in" show the circuit name, the index of the selected primary output and the number of variables of the function.

We conducted three experiments on each function for comparison. The column "restricted" corresponds to the restricted simple disjunctive decomposition shown in Section 3. The column "exhaustive" corresponds to the method in which an exhaustive search [14] was used for finding a bound set that gives a decomposition. The column "restricted+exhaustive" corresponds to the combined method in which the exhaustive search was applied after no more decomposition was found by the "restricted" method.

The columns "XOR", "AND" and "SYM" correspond to XOR gates, AND gates and the subfunctions produced in the decomposition shown in Subsection 3.4; the column "random" corresponds to functions that could not be decomposed by the "restricted" method. These columns show the input sizes of functions of the type; the number in a parenthesis indicates the number of functions of the same input size. In "XOR" and "AND", we merged successive functions if we could. For example, a decomposition result $((x_1 \cdot \overline{x}_2) \cdot x_3) \cdot (x_4 \oplus x_5)$ (XOR: 2 and AND: 2(3)) was converted to $x_1 \cdot \overline{x}_2 \cdot x_3 \cdot (x_4 \oplus x_5)$ (XOR: 2 and AND: 4). The columns "random" show the difference of the decomposition results between "restricted" and "exhaustive". For example, in the circuit "x4", the "random" function with 9 inputs that could not be decomposed by the "restricted" method was decomposed into functions with 2 and 8 inputs by the "exhaustive" method. Note that the decomposition results of "exhaustive" and "restricted+exhaustive" are the same. "Time" is the CPU seconds on a Sun Ultra 1 Model 170E, where "$>$1000" means that the execution did not finish in 1000 seconds. We limited the maximum number of usable OBDD nodes to 1,000,000.

From the results of the "restricted" method, we can observe the following. Even if we restricted a bound set to a maximal symmetry group, we found simple disjunctive decomposition forms in many practical functions. The execution time was negligible even for functions with many variables. The "exhaustive" method decomposed the "random" functions that could not be decomposed in the "restricted" method, but it required an expensive execution time and generally failed for functions with more than 24 variables. The "restricted+exhaustive" method successfully decomposed some functions that could not be decomposed by the "exhaustive" method in a practical amount of time. This is because the restricted simple disjunctive decomposition reduced the number of variables in the function to be decomposed before applying the exhaustive search.

5. Conclusion

We have presented an efficient procedure for a restricted simple disjunctive decomposition. It is based on symmetry detection and variable reordering of an OBDD. The decomposition form is restricted such that the bound set is a maximal symmetry group. Because of the restriction, the computation cost is very small. Even if we restricted a bound set to a set of symmetric variables, we found simple disjunctive decomposition forms in many practical functions. Thus the procedure will be useful as a preprocess for multi-level logic synthesis.

As a future work, we would like to develop an

Table 1. Experimental results

circuit			restricted					exhaustive		restricted+exhaustive	
name	po	#in	XOR	AND	SYM	random	time(sec)	random	time(sec)	random	time(sec)
C880	24	45	0	2,4	0	41	0.70	—	>1000	—	>1000
apex1	23	39	0	5,6	0	30	0.18	—	>1000	—	>1000
apex2	1	36	0	2(4),3	0	30	1.59	—	>1000	—	>1000
apex3	8	45	0	2(3)	0	42	0.23	—	>1000	—	>1000
apex5	77	25	0	4,6,11	0	7	1.03	—	>1000	2,6	1.11
apex6	3	24	0	2(2)	0	22	0.20	2(2),7,14	419.22	2(2),7,14	426.44
apex7	4	24	0	3,7	0	16	0.12	2(2),6,9	2.76	2(2),6,9	1.31
cordic	1	23	4(2)	2,3	3(3)	8	0.40	8	74.37	8	0.40
count	16	20	2	16	0	4	0.09	2,3	0.11	2,3	0.08
cps	1	22	0	5	3	16	0.26	2(2),7,8	330.77	2(2),7,8	0.80
dalu	2	47	0	2(2),14(2)	3	17	0.50	—	>1000	17	34.26
des	65	19	2(6)	3	0	11	0.81	2,3(2),6	28.69	2,3(2),6	0.83
duke2	7	18	0	0	0	18	0.09	2,17	15.83	2,17	15.98
frg2	135	25	0	2(5),3(2),4	0	13	0.36	2(3),10	6.99	2(3),10	0.80
misex2	10	14	0	2,3,7	0	5	0.07	2,4	0.18	2,4	0.07
o64	1	130	0	2(65),65	0	0	1.25	0	0.40	0	1.23
pdc	30	13	0	3	0	11	1.22	11	2.05	11	1.54
seq	6	38	0	2(2),3	0	34	0.61	—	>1000	—	>1000
spla	1	16	0	2(2),14	0	0	1.08	0	3.05	0	1.08
t481	1	16	2(5)	2(10)	0	0	0.16	0	9.74	0	0.17
term1	8	20	2	2(6),4,5,6	0	0	0.18	0	0.83	0	0.17
too_large	1	36	0	2(4),3	0	30	1.61	—	>1000	—	>1000
vda	15	17	0	0	0	17	0.23	17	20.30	17	20.26
vg2	2	25	0	2	0	24	0.08	—	>1000	—	>1000
x4	26	15	0	3,5	0	9	0.16	2,8	0.98	2,8	0.17

efficient procedure for a general (not restricted) simple disjunctive decomposition. We also have to evaluate the effectiveness of a simple disjunctive decomposition by integrating the procedure into a practical multi-level logic synthesis system.

References

[1] R. L. Ashenhurst. The Decomposition of Switching Functions. In *Proc. of an International Symposium on the Theory of Switching*, Apr. 1957.

[2] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.*, C-35(8):667–691, Aug. 1986.

[3] S. Chang and M. Marek-Sadowska. Technology Mapping via Transformations of Function Graphs. In *Proc. ICCD*, pages 159–162, Oct. 1992.

[4] D. I. Cheng and M. Marek-Sadowska. Verifying Equivalence of Functions with Unknown Input Correspondence. In *Proc. EDAC*, pages 81–85, Feb. 1993.

[5] C. R. Edwards and S. L. Hurst. A Digital Synthesis Procedure Under Function Symmetries and Mapping Methods. *IEEE Trans. Comput.*, c-27(11):985–997, Nov. 1978.

[6] M. Fujita, Y. Matsunaga, and T. Kakuda. On Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis. In *Proc. EDAC*, pages 50–54, Feb. 1991.

[7] Y.-T. Lai, M. Pedram, and S. Vrudhula. BDD Based Decomposition of Logic Functions with Application to FPGA Synthesis. In *Proc. DAC*, pages 642–647, June 1993.

[8] J. Mohnke, P. Molitor, and S. Malik. Limits of using Signatures for Permutation Independent Boolean Comparison. In *Proc. ASP-DAC*, pages 459–464, Aug. 1995.

[9] D. Möller, J. Mohnke, and M. Weber. Detection of Symmetry of Boolean Functions Represented by ROBDDs. In *Proc. ICCAD*, pages 680–684, Nov. 1993.

[10] S. Panda, F. Somenzi, and B. F. Plessier. Symmetry Detection and Dynamic Variable Ordering of Decision Diagrams. In *Proc. ICCAD*, pages 628–631, Nov. 1994.

[11] J. P. Roth and R. M. Karp. Minimization Over Boolean Graphs. *IBM Journal*, pages 227–238, Apr. 1962.

[12] R. Rudell. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *Proc. ICCAD*, pages 42–47, Nov. 1993.

[13] T. Sasao. FPGA Design by Generalized Functional Decomposition. In T. Sasao, editor, *Logic Synthesis and Optimization*, pages 233–258. Kluwer Academic Publishers, 1993.

[14] H. Sawada, T. Suyama, and A. Nagoya. Logic Synthesis for Look-Up Table based FPGAs using Functional Decomposition and Support Minimization. In *Proc. ICCAD*, pages 353–358, Nov. 1995.

[15] C. Tsai and M. Marek-Sadowska. Generalized Reed-Muller Forms as a Tool to Detect Symmetries. *IEEE Trans. Comput.*, 45(1):33–40, Jan. 1996.

[16] S. Yang. *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0.* MCNC, Jan. 1991.