ANTISYMMETRIES IN THE REALIZATION OF BOOLEAN FUNCTIONS

J. E. Rice and J. C. Muzio

VLSI Design and Test Group, Dept. of Computer Science
University of Victoria, BC, Canada
`[jrice, jmuzio]@csr.uvic.ca`

ABSTRACT

New symmetries of degree two are introduced, along with spectral techniques for identifying these symmetries. Some applications of these symmetries are discussed, in particular their application to the construction of binary decision diagrams and the implementation of Boolean functions.

1. INTRODUCTION

Partial symmetries exist in most Boolean functions, particularly those used in practical applications. Both total and partial symmetry properties are commonly used in synthesis of digital circuits [1], [2], [3], [4], particularly in the reduction of the size of Binary Decision Diagram (BDD) representation of functions [5], [6].

Various literature on partial and total symmetries exists (see section 2.2). However, most of the documented symmetry properties depend on the identification of two identifical subfunctions within a Boolean function. In this paper we examine partial symmetries that occur when one of the subfunctions is not identical to the other, but is instead the inverse of the other. We refer to these as antisymmetries, certain types of which were introduced by Tsai and Marek-Sadowska in [3] as *skew*-symmetries.

Symmetries, whether partial or total, are useful if they can be detected and provide assistance in terms of a function's implementation or representation. We present a method for detecting any antisymmetry of degree two based on the function's spectral coefficients. We also demonstrate examples of their use in logic synthesis or minimization of function realizations, and hypothesize that the antisymmetries can be incorporated into known methods for BDD minimization.

2. PRELIMINARIES

In this section essential notations and definitions are presented. We assume throughout this paper that all Boolean functions are completely specified. We also assume that the reader is familiar with common representations of Boolean functions such as Karnaugh-maps and truth tables, and introduce only the representations pertinent to this paper.

2.1. Notation

Subfunctions based on two variables $\{x_i, x_j\}$ are used throughout this paper. Without loss of generality we label them $\{x_{n-1}, x_n\}$ allowing us to simplify the definitions and use the following notation:

$$\begin{aligned} f_0 &= f(x_1, ..., x_{n-2}, 0, 0) \\ f_1 &= f(x_1 ...., x_{n-2}, 0, 1) \\ f_2 &= f(x_1, ..., x_{n-2}, 1, 0) \\ f_3 &= f(x_1, ..., x_{n-2}, 1, 1) \end{aligned}$$

2.2. Symmetries

Many types of symmetries exist, such as total symmetries, equivalence and nonequivalence symmetries [1], quasisymmetries [7] and partial symmetries [7], [5]. In this section we define the symmetries on which our work is based.

Let $f : \{0,1\}^n$ be a completely specified Boolean function and $\mathcal{V}_n = \{x_1, ..., x_n\}$ be the corresponding set of variables. A function f is said to be symmetric with respect to a set $\lambda \subseteq \mathcal{V}_n$ if f remains unchanged for all permutations of the variables in λ. If $\lambda = \mathcal{V}_n$ then we say that the function is totally symmetric, otherwise we say that it is partially symmetric over the variables in λ [8].

A symmetry of degree two is a partial symmetry in which the two subfunctions that are identical are independent of two of the function's variables. Our antisymmetries are based on Hurst *et. al's.* definitions of equivalence, nonequivalence and single-variable symmetries, which are all symmetries of degree two [1].

A Boolean function f is said to possess an equivalence symmetry if there exist two variables $\{x_{n-1}, x_n\} \subseteq \mathcal{V}_n$ such that $f_0 = f_3$. This is written $E\{x_{n-1}, x_n\}$. Non-equivalence symmetries are written $N\{x_{n-1}, x_n\}$ and are defined as $f_1 = f_2$, and a third type of symmetry called single-variable symmetries are written $S\{x_n | x_{n-1}\}$ or $S\{x_n | \overline{x}_{n-1}\}$ and are defined as $f_2 = f_3$ or $f_0 = f_1$.

2.3. Spectral Coefficients

The spectral coefficients of a Boolean function are a representation of the function that is not restricted to the $\{0, 1\}$

domain. If the outputs of a Boolean function are encoded as $\{+1, -1\}$ then we use Y to refer to the output vector of the function[1].

The spectral coefficient vector for f is defined as

$$S = T^n \cdot Y \tag{1}$$

where T^n is some transform matrix such as the Hadamard, Walsh or Rademacher-Walsh matrix [1].

The spectral coefficient vector can be subdivided into subvectors according to which variables are related to the particular coefficient. This subdivision is as follows when based on variables x_{n-1} and x_n: S^0 is the top quarter of the coefficients[2], S^1 is the second quarter, S^2 is the third quarter, and S^3 is the last quarter of the coefficients.

Similarly, we define the spectral coefficient vectors for these subfunctions described at the beginning of this section as follows:

$$\begin{aligned} S_0 &= T^n \cdot Y_0 \\ S_1 &= T^n \cdot Y_1 \\ S_2 &= T^n \cdot Y_2 \\ S_3 &= T^n \cdot Y_3 \end{aligned}$$

where Y_0 is the output vector for f_0 and so on.

The relationship between S_0, S_1, S_2, and S_4 to S^0, S^1, S^2, and S^4 is as follows [1]:

$$\begin{aligned} 4S_0 &= S^0 + S^1 + S^2 + S^3 \\ 4S_1 &= S^0 - S^1 + S^2 - S^3 \\ 4S_2 &= S^0 + S^1 - S^2 - S^3 \\ 4S_3 &= S^0 - S^1 - S^2 + S^3 \end{aligned} \tag{2}$$

2.4. BDDs

Binary Decision Diagrams (BDDs) were first introduced by Lee [9] and later by Akers [10].

A BDD is defined as [11]

> *a binary directed acyclic graph with two leaves* TRUE *and* FALSE*, in which each non-leaf node is labeled with a variable and has two out-edges, one pointing to the subgraph that is evaluated if the node label evaluates to* TRUE *and the other pointing to the subgraph that is evaluated if the node label evaluates to* FALSE*.*

Every node in the BDD represents either a literal in the Boolean function, or its complement. Every non-leaf node has two outward edges leading to two other nodes. If the node has a value of "1" (TRUE) then, to obtain the value of the expression, one follows the edge marked "1" and evaluates that node. Similarly, if the node has a value of "0" (FALSE), one follows the edge marked "0" and evaluates that node. This process is repeated until a leaf node with the value "1" or "0" is reached reached, and the evaluation is complete. The direction of the edges from each node is not explicitly marked, but is understood to be from the root towards the leaf nodes.

BDDs are commonly used in their canonical form. This simplified form has the property that any two equivalent Boolean functions have the same canonical BDD if the same variable ordering is used. This canonical form is called a *ROBDD*, or *R*educed *O*rdered *B*inary *D*ecision *D*iagram. Again, it is generally understood that when using the term BDD one is referring to a ROBDD.

A ROBDD is a reduced BDD with a specified ordering of variables. A ROBDD meets two main specifications:

- a BDD is a reduced BDD if it contains no vertex whose left subgraph is equal to its right subgraph, nor does it contain distinct vertices v and v' such that the subgraph rooted by v and v' are isomorphic.
- a BDD is an ordered BDD if on every path from the root node to an output, the variables are encountered in the specified order.

Another commonly used method of reducing the size of a BDD is to introduce *inverters.* These are indicators on the path to a subgraph that are used to mark that the subgraph is inverted.

3. ANTISYMMETRIES OF DEGREE TWO

Based on the the equivalence, nonequivalence and single-variable symmetries we introduce some additional types of symmetries of degree two. We call these symmetries *anti-symmetries.*

A Boolean function f is said to possess an anti-equivalence symmetry $\overline{E}\{x_{n-1}, x_n\}$ if

$$f(x_1, ..., x_{n-2}, 0, 0) = \overline{f(x_1, ..., x_{n-2}, 1, 1)}$$

Table 1 summarizes the various types of antisymmetries and their definitions.

Antisymmetry	Definition
$\overline{E}\{x_{n-1}, x_n\}$	$f(x_1, ..., x_{n-2}, 0, 0) = \overline{f(x_1, ..., x_{n-2}, 1, 1)}$
$\overline{N}\{x_i, x_j\}$	$f(x_1, ..., x_{n-2}, 0, 1) = \overline{f(x_1, ..., x_{n-2}, 1, 0)}$
$\overline{S}\{x_j \vert x_i\}$	$f(x_1, ..., x_{n-2}, 1, 0) = \overline{f(x_1, ..., x_{n-2}, 1, 1)}$
$\overline{S}\{\overline{x}_j \vert x_i\}$	$f(x_1, ..., x_{n-2}, 0, 0) = \overline{f(x_1, ..., x_{n-2}, 0, 1)}$

Table 1. Definitions and notation for the antisymmetries.

4. CONDITIONS AND TESTS FOR THE ANTISYMMETRIES

[3], [6], and [12] each present methods of (anti)symmetry detection based on BDDs. In this section we present both conditions and tests for the antisymmetries based on the function's spectral coefficients. Until recently, the spectral coefficients were considered too expensive to compute and so

[1]the $\{0, 1\}$ encoded output vector is generally referred to as Z.

[2]Note that this assumes the Hadamard ordering of the spectral vector

this technique was not popular. However, Thornton *et. al.* have presented methods for efficient calculation of the spectral coefficients based on BDDs [13], thus making spectral techniques feasible for many practical functions.

Based on the definitions of the various antisymmetries we recall that $\overline{E}\{x_{n-1}, x_n\}$ can be defined as $f_0 = \overline{f_3}$, or, since the effect of negating a function is to change the sign of all the spectral coefficients, we have $S_0 = -S_3$. This is the necessary condition for the anti-equivalence symmetry on x_n, x_{n-1} to exist.

Using Equation 2 we get $S^0 = -S^3$. This is the test for the existence of the anti-equivalence symmetry in a Boolean function. This process can be extended to any variables x_i, x_j by selecting the appropriate subvectors S^0 and S^3.

The table below summarizes the conditions and tests we have derived for each of the antisymmetries:

Symmetry	Condition	Test
$\overline{E}\{x_{n-1}, x_n\}$	$S_0 = -S_3$	$S^0 = -S^3$
$\overline{N}\{x_{n-1}, x_n\}$	$S_0 = S_3$	$S^0 = S^3$
$\overline{S}\{x_{n-1} \vert x_n\}$	$S_1 = -S_3$	$S^0 = S^1$
$\overline{S}\{x_n \vert x_{n-1}\}$	$S_2 = -S_3$	$S^0 = S^2$
$\overline{S}\{x_{n-1} \vert \overline{x}_n\}$	$S_0 = -S_2$	$S^0 = -S^1$
$\overline{S}\{x_n \vert \overline{x}_{n-1}\}$	$S_0 = -S_1$	$S^0 = -S^2$

Table 2. Spectral conditions and tests for the antisymmetries of degree two.

Detection of the antisymmetries is also relatively straightforward with the use of representations such as Karnaugh-maps and BDDs. However, there is some dependency on the chosen ordering of variables, particularly with the use of Karnaugh-maps.

5. APPLICATIONS

5.1. BDD Minimization

The use of symmetries to minimize Binary Decision Diagram (BDD) or related representations is well-documented [6], [14], [15], [16], [17]. Much research has demonstrated that a function's symmetry properties may reduce the size of the BDD or related data structure such as Functional Decision Diagrams (FDDs) [5], [15], [18], [19].

In particular, Scholl *et. al.* present a method of BDD minimization based on symmetries [15]. This method is based on heuristics which identify partial symmetries within a function. It is our hypothesis that such heuristics can be expanded to incorporate the antisymmetries that we have defined. This would allow identification of situations where an antisymmetric portion of a function could be shared within the structure of the BDD.

Figure 1 shows a 4 variable Boolean function possessing $\overline{E}\{x_3, x_4\}$.

Figure 2 shows the BDD for this function. The branches which could be shared due to the anti-equivalence symmetry have been indicated by the boxes.

$^{x_4x_3}_{x_2x_1}$	00	01	11	10
00	0	0	1	1
01	1	0	0	0
11	1	1	0	1
10	0	0	1	0

Fig. 1. The Karnaugh map for a Boolean function possessing $\overline{E}\{x_3, x_4\}$.

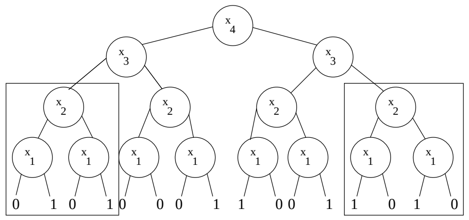

Fig. 2. A BDD showing two branches which display an anti-equivalence symmetry. Note that the left edge from each node is the 0 edge while the right is the 1 edge.

5.2. Logic Synthesis Application

An example of how the antisymmetries in a function can reduce the complexity of the function's logic implementation is shown below.

$^{x_4x_3}_{x_2x_1}$	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	0	0	1	1
10	0	1	0	0

a)

$^{x_4x_3}_{x_2x_1}$	00	01	11	10
00	0	0	–	0
01	1	1	–	1
11	0	0	–	1
10	0	1	–	0

b)

Fig. 3. **a)** the Karnaugh map for $F = \overline{x}_1\overline{x}_2x_3x_4 + x_1\overline{x}_2\overline{x}_4 + x_1\overline{x}_3x_4 + x_1x_2x_4 + \overline{x}_1x_2x_3\overline{x}_4$. **b)** the Karnaugh map for $F* = x_1\overline{x}_2 + x_1x_4 + \overline{x}_1x_2x_3$.

Function F, shown on the left in Figures 3 and 4, possesses the antisymmetry $\overline{S}\{x_4|x_3\}$. Knowing this, we can manipulate the function in such a way that results in a function of a reduced size, plus some additional logic to convert the reduced function into the desired function. The reduced function is shown on the right in Figures 3 and 4.

The advantage of using $F*$ to implement the function is that there is a greatly reduced number of blocks in the Karnaugh-map. This leads to a smaller number of overall inputs being required, as well as possibly improving routing requirements on an FPGA-type implementation. Additionally, $F*$ clearly has many more symmetries that can be identified, and repeating this process allows us to realize a circuit that has

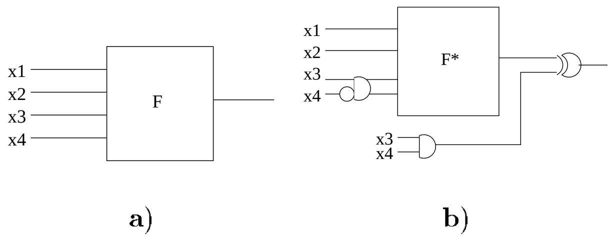

Fig. 4. **a)** Representation of F. **b)** Representation of F in terms of a reduced function, $F*$.

approximately one-half the complexity of that resulting directly from the original sum-of-products description.

6. CONCLUSIONS

We have presented symmetries of degree two that are based on the inverse of a portion of a function. Prior symmetry definitions have not incorporated this possibility, and thus possible advantages may have been overlooked. We suggest that making use of our definitions of the antisymmetries may improve even further known heuristics for ROBDD ordering that are based on partial symmetries of a function, as well as leading to improved synthesis techniques.

It is thought that these techniques will improve the existing heuristics for BDD minimization, and lead to the possible development of new heuristics for ROBDD ordering based on the antisymmetries. Additionally, this work may be able to be extended to symmetries of degree n for $n > 2$, incompletely specified functions, and multi-valued logic.

7. REFERENCES

[1] S. L. Hurst, D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, Inc., Orlando, Florida, 1985.

[2] B. Kim and D. L. Dietmeyer, “Multilevel Logic Synthesis of Symmetric Switching Functions,” *IEEE Trans. on CAD*, pp. 436–446, Apr. 1991.

[3] C.-C. Tsai and M. Marek-Sadowska, “Generalized Reed-Muller Forms as a Tool to Detect Symmetries,” *IEEE Trans. on Comp.*, pp. 33–40, Jan. 1996.

[4] V. N. Kravets and K. A. Sakallah, “Constructive Library-Aware Synthesis Using Symmetries,” in *DATE*, 2000.

[5] L. Heinrich-Litan and P. Molitor, “Least Upper Bounds for the Size of OBDDs Using Symmetry Properties,” *IEEE Trans. on Comp.*, pp. 360–368, Apr. 2000.

[6] S. Panda, F. Somenzi, and B. Plessier, “Symmetry detection and dynamic variable ordering of decision diagrams,” in *ICCAD*, 1994.

[7] C. Meinel and T. Theobald, *Algorithms and Data Structures in VLSI Design*, Springer-Verlag, 1998.

[8] L. Litan, P. Molitor, and D. Möller, “Least Upper Bounds on the Sizes of Symmetric Variable Order based OBDDs,” in *Proceedings of the Great Lakes Symposium on VLSI*, 1996, pp. 680–684.

[9] C. Y. Lee, “Representation of Switching Circuits by Binary Decision Diagrams,” *Bell System Technical Journal*, pp. 958–999, 1959.

[10] S. Akers, “Binary Decision Diagrams,” *IEEE Trans. on Comp.*, vol. C-27, no. 6, pp. 509–516, June 1978.

[11] K. Karplus, “Using if-then-else DAGs for Multi-Level Logic Minimization,” Tech. Rep. UCSC-CRL-88-29, University of California Santa Cruz, Nov. 1988, ftp’d from ftp.cse.ucsc.edu, 20 pages.

[12] D. Möller, J. Mohnke, and M. Weber, “Detection of symmetry of boolean functions represented by ROBDDs,” in *ICCAD*, 1993, pp. 680–684.

[13] M. A. Thornton and V. S. S. Nair, “Efficient Calculation of Spectral Coefficients and Their Applications,” *IEEE Trans. on CAD*, Nov. 1995.

[14] R. Drechsler and D. Sieling, “Binary Decision Diagrams in Theory and Practice,” *STTT*, pp. 112–136, May 2001.

[15] C. Scholl, D. Möller, P. Molitor, and R. Drechsler, “BDD Minimization using Symmetries,” *IEEE Trans. on CAD*, pp. 81–99, Feb. 1999.

[16] Randal E. Bryant, “On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication,” *IEEE Trans. on Comp.*, vol. 40, no. 2, pp. 205–213, Feb. 1991.

[17] R. E. Bryant, “Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams,” in *ACM Computing Surveys*, vol. 24. Sept. 1992.

[18] C. Scholl, S. Melchior, G. Hotz, and P. Molitor, “Minimizing ROBDD Sizes of Incompletely Specified Boolean Functions by Exploiting Strong Symmetries,” in *ED&TC*, 1997, pp. 229–234.

[19] R. Drechsler and B. Becker, “Sympathy: Fast Exact Minimization of Fixed Polarity Reed-Muller Expressions for Symmetric Functions,” in *ED&TC*, 1995.