Extending Symmetric Variable-Pair Transitivities Using State-Space Transformations

Peter M. Maurer
Dept. of Computer Science
Baylor University

Abstract-Two-cofactor relations and their associated symmetry types have been studied for many years. While ordinary symmetries are simply transitive permitting them to be combined into clusters of variables, other types of symmetries have more complex transitivities. This paper shows how to convert the various types of symmetry into ordinary symmetries using state-space transformations. This permits the simple transitivity of ordinary symmetry to be extended to virtually any type of symmetry.

1 Introduction

Symmetric Boolean functions have been exploited in a number of different areas of design automation. [1-6] Research has progressed in a number of different directions, one of which is the identification and enumeration of the various types of symmetries that can exist between a pair of variables. [7-9] To identify these symmetries, one examines the relations between the cofactors of a Boolean function f. In virtually all cases, each relation is identified with a particular type of symmetry. Other avenues of research use group-theoretic definitions of symmetry, and include symmetries that cannot be defined in terms of variable pairs. [10-12]

There are a number of different approaches to symmetry detection, not all of which are strictly cofactor based. (See [13, 14] for example.) Although the cofactor relations still play an important role in defining symmetry types.

Several researchers have examined the idea of variable pair transitivities ([15] for example). The transitive relations between various types of symmetric variable pairs are fairly complex, except for ordinary symmetries which are simply transitive. In this paper we will show how to extend the simple transitivity of ordinary symmetry to more general types of symmetry. These extensions will be used to create clustered variables which can be used in place of simple variables for symmetry detection. The use of clustered variables speeds up symmetry detection and enables us to detect symmetric variable pairs that *do not manifest themselves as cofactor relations*. We are able to extend transitivity to all known types of symmetric variable pairs, and we can extend our algorithm to handle new types of cofactor relations that, to the best of our knowledge, have never been used for symmetry detection.

The output of our symmetry detection algorithm is a totally or partially symmetric function f and a collection of corrective functions that must be applied to the inputs and outputs of f. In many cases these corrective functions are simple, but in some cases they are not. In our practical implementations of this algorithm, we have confined ourselves to the cases with simple corrective functions, but in this paper we will include some of the details of the more complex cases.

2 Cofactor Relations

The cofactor of a Boolean function f is found by setting one or more inputs to a constant value, either 0 or 1. In this paper, we designate cofactors using subscripts consisting of 0, 1 and x. Suppose that f is a four-input function, and we have set the first two variables to 1. This will produce the cofactor f_{11xx}, which is a function of two variables. The ones and zeros designate the variables that have been replaced with constant values, and the x's represent the remaining variables. When there is no opportunity for confusion, we will drop the x's from the subscript and designate cofactors such as f_{11xx} as f_{11}.

In most cases, symmetry detection is done using relations between two-variable cofactors. There are six such relations, which listed in Figure 1 with their associated symmetry types. These relations are called the *classical relations*.

Type	Condition
Ordinary	$f_{01} = f_{10}$
Multi-Phase	$f_{00} = f_{11}$
Single-Variable a	$f_{10} = f_{11}$
Single Variable b	$f_{01} = f_{11}$
Multi-Phase Single Variable a	$f_{00} = f_{01}$
Multi-Phase Single Variable b	$f_{00} = f_{10}$

Figure 1. The Classical Relations.

As shown in [16] the classical relations can be expressed in another form using the exclusive or function, which we designate as $\oplus$. The ordinary relation, $f_{01} = f_{10}$ becomes $f_{01} \oplus f_{10} = \overline{0}$, and the multi-phase relation becomes $f_{00} \oplus f_{11} = \overline{0}$, where $\overline{0}$ represents the zero function.

A slightly more complex relation can be obtained by replacing the zero function with the constant-one function as in $f_{01} \oplus f_{10} = \overline{1}$. This gives us six new relations that are normally associated with six types of anti-symmetry. (The term anti-symmetry was introduced in [17]. Anti-symmetry is also known as skew symmetry [15] and negative symmetry [18].) These six new relations are given in Figure 2 along with their associated symmetries.

Type	Condition
Anti	$f_{01} \oplus f_{10} = \overline{1}$
Anti Multi-Phase	$f_{00} \oplus f_{11} = \overline{1}$
Anti Single-Variable a	$f_{10} \oplus f_{11} = \overline{1}$
Anti Single Variable b	$f_{01} \oplus f_{11} = \overline{1}$
Anti Multi-Phase Single Variable a	$f_{00} \oplus f_{01} = \overline{1}$
Anti Multi-Phase Single Variable b	$f_{00} \oplus f_{10} = \overline{1}$

Figure 2. The Anti-Relations.

Cofactor relations can be expanded in several ways, only a few of which have been investigated. The constant functions $\overline{0}$ and $\overline{1}$ can be replaced with other functions to create relations of the form $f_{01} \oplus f_{10} = g$ and the XOR operation can be replaced with another two-input function such as AND and OR to give relations of the form $f_{01} f_{10} = \overline{0}$ and $f_{10} + f_{01} = g$. It is possible to use cofactors in more than two variables to detect higher-order symmetries between three or more variables. These types of relations have not been extensively studied.

Relations between three or four two-variable cofactors give rise to the Kronecker symmetries. [9]. These relations use the XOR function, and are listed in Figure 3. Three and four cofactor relations can be extended in the same way as two-cofactor relations.

3 The State-Space

In its most elementary form, the state-space of a Boolean function is an n-dimensional hypercube containing cofactors of the function with each dimension of the hypercube corresponding to an eliminated variable. In the extreme case, there will be one dimension for each input variable and the nodes will contain Boolean values. When symmetries are detected, two dimensions can be collapsed into a single dimension. The collapsed dimension corresponds to a clustered variable, and contains three or more nodes. We call this structure a hyperlinear structure to emphasize its non-cubical nature.

During symmetry detection new cofactors are computed and new symmetry tests are performed. The hyperlinear structure expands when new cofactors are computed, and collapses when symmetric variable pairs are detected. If the process runs to completion, the value of each node will be reduced to a single Boolean value. However, symmetry detection can be terminated at any point using either time or space constraints. This enables the detection process to be run in "anytime" fashion as in [18]. The process is adaptable to many existing algorithms, particularly those that already use cofactor relations for symmetry detection. It is especially well suited to techniques such as that detailed in [19], which use the cofactors that already exist in a ROBDD.

Type	Condition
K_0	$f_{01} \oplus f_{10} \oplus f_{11} = \overline{0}$
K_1	$f_{00} \oplus f_{10} \oplus f_{11} = \overline{0}$
K_2	$f_{00} \oplus f_{01} \oplus f_{11} = \overline{0}$
K_3	$f_{00} \oplus f_{01} \oplus f_{10} = \overline{0}$
K_4	$f_{00} \oplus f_{01} \oplus f_{10} \oplus f_{11} = \overline{0}$
Anti-K_0	$f_{01} \oplus f_{10} \oplus f_{11} = \overline{1}$
Anti-K_1	$f_{00} \oplus f_{10} \oplus f_{11} = \overline{1}$
Anti-K_2	$f_{00} \oplus f_{01} \oplus f_{11} = \overline{1}$
Anti-K_3	$f_{00} \oplus f_{01} \oplus f_{10} = \overline{1}$
Anti-K_4	$f_{00} \oplus f_{01} \oplus f_{10} \oplus f_{11} = \overline{1}$

Figure 3. The Kronecker Relations.

When a symmetry other than an ordinary symmetry is detected, a state-space transformation is applied to convert the symmetry to an ordinary symmetry. Corrective functions are then added to the functions input and output to compensate for the transformation. *Any* of the relations discussed in Section 2 can be converted to ordinary symmetries, but in some cases the corrective functions are too complex for these relations to be useful.

The technique described here is based on the technique presented in [20]. However in [20], symmetry is always forced to run to completion, only the classical relations are used, only input corrective functions are allowed, and the result is a state-machine used for simulating the function. Here we extend the technique to general relations, permit both input and output corrective functions, and provide the result in a general form.

4 Symmetry Detection

The detection algorithm begins by computing four cofactors of the function and placing them in a two-dimensional hypercube. Figure 4 illustrates how this would be done for the function $abc + abd + acd + bcd$ and its four cofactors, 0, cd, cd, and $c + d$. The physical structure is a single-dimensional array which is indexed using a private indexing function to make it appear multi-dimensional. The use of a private indexing function permits the number of dimensions in the logical structure to change during the symmetry detection process. Furthermore, the logical structure

can be indexed in unconventional ways to perform certain state-space transformations.

It is obvious from Figure 4 that there is a an ordinary symmetry between variables a and b. This causes the logical structure to collapse into a single dimension as in Figure 5. The vertices of Figure 5 are indexed by the number of ones in the input variables a and b.

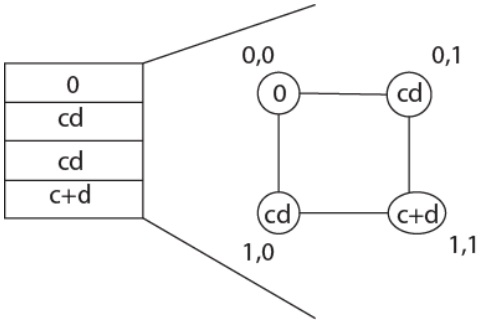

Figure 4. The Array and Logical Structure.

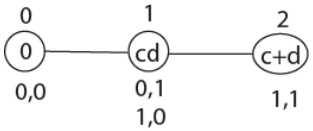

Figure 5. A Collapsed Logical Structure.

Collapsing reduces the size and the number of dimensions, and it permits symmetric variable pairs to be combined into a single composite variable. Composite variables can be treated more or less like simple variables, thus simplifying the detection process.

Testing for the other five classical relations can be done by selecting various pairs of nodes from the structure of Figure 5. However, rather than doing this, we modify the private indexing function and test for ordinary symmetry. For multi-phase symmetry, we modify the private indexing function to index one dimension in reverse order and test for ordinary symmetry. Figure 6 shows how reversing the second dimension of Figure 4 repositions the nodes. (The comparison is always between the node labeled "0,1" and the node labeled "1,0", wherever these indices may appear.) Reversing the dimension is the same as negating the associated input variable. We call this the *phase* transformation.

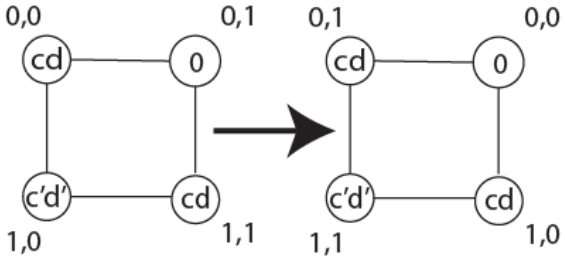

Figure 6. Reverse Indexing the Second Dimension.

When a multi-phase symmetry is detected, the structure is collapsed using the phase transformation. This makes the transformation permanent for all future operations. To correct for the phase transformation it is necessary to place an inverter on the input representing the reversed dimension.

To detect the single-variable symmetries it is necessary to reverse the indexing of odd-numbered rows or the odd-numbered columns. (Rows and columns are numbered starting with zero.) We call this type of transformation the *conjugate* transformation. It is equivalent to transforming the inputs with a matrix is over the field GF2. (In GF2 there are two values 0, and 1 with AND replacing multiplication and XOR replacing addition. See [20] for the theoretical development.) Figure 7 gives a matrix and shows the effect of multiplying the inputs by the given matrix. This results in testing for a single-variable symmetry when comparing the nodes labeled "0,1" and "1,0". Strictly speaking, the type of symmetry detected using the conjugate transformation is *conjugate symmetry*, not single-variable symmetry. Every single-variable symmetry is also a conjugate symmetry, but there are conjugate symmetries that do not manifest themselves as single-variable symmetries. (As in the function $ab' + ac + b'c$, which has an ordinary symmetry between a and b and a conjugate symmetry between the composite pair ab and the variable c.) The conjugate transformation can detect these types of symmetries as well as single-variable symmetries.

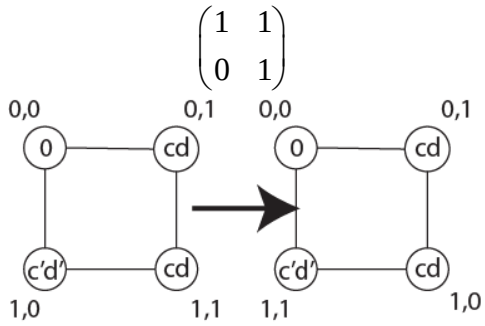

Figure 7. Reverse Indexing the Odd Rows.

Again, the conjugate transformation is used during the collapse of the logical structure, making the transformation permanent. The matrix used during the detection is retained and used to construct the corrector function. If more than one conjugate symmetry is detected, the matrices are multiplied together in the order they are first used. To correct for conjugate transformations it is necessary to add one or more XOR gates to the inputs of the function. The final matrix is used to determine the inputs of these XOR gates. Each column of the matrix corresponds to an input of the transformed function, and each row corresponds to one of the original inputs. The outputs of these gates replace the original inputs. The columns of the transformation matrix determine the inputs of the XOR gates.

There are two types of single-variable symmetries. Reversing the indexing of the odd rows gives one type, reversing the odd columns gives the other type. The multi-phase single-variable symmetries can be tested by combining a phase transformation with a conjugate transformation.

Detecting anti symmetries requires a transformation of the node contents, not just a transformation of indices. Correcting these sorts of transformations requires an output corrective function rather than an input corrective function. Figure 8 shows the *anti* transformation required to detect an anti-symmetry. Odd numbered rows of the hyperlinear structure are inverted. In Figure 8 it is not technically necessary to complement node 1,1, but doing so simplifies the corrector function. Like the previous transformations, anti transformations are temporary until a symmetry is detected, at which time the transformation becomes permanent.

To correct the transformation of Figure 8, the output of the function must be XORed with variable corresponding to the vertical dimension. The anti transformation can be combined with the phase and conjugate transformations to detect all conditions listed in Figure 2.

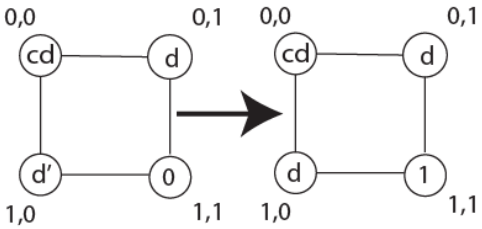

Figure 8. Anti-Symmetry Detection.

Although our symmetry detection algorithms do not check for relations of the form $f_{01} \oplus f_{10} = g$ we could easily adapt them to do so. Figure 9 shows the state-space transformation for detecting this type of relation. The function g is XORed with the odd rows of the logical structure. Once a symmetry is detected, the logical structure is collapsed and the transformation becomes permanent. The corrective function for $f_{01} \oplus f_{10} = g$ relations is shown in Figure 10, where F is the result of the state-space transformation and a is the input variable corresponding to the vertical dimension. The corrector function requires the computation of the function g so it should be easy to compute, or a function that is already computed elsewhere in the circuit.

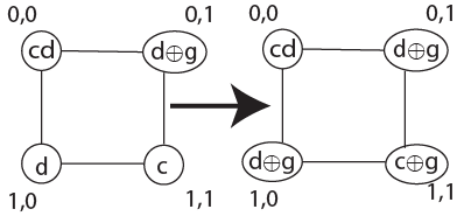

Figure 9. Detection for Relation $f_{01} \oplus f_{10} = g$.

For relations of the form $f_{01} \oplus f_{00} \oplus f_{10} = \overline{0}$ the situation is more complex because the output corrector function must include at least one cofactor of the function. The two-dimensional structure and its state-space transformation, the *K transformation*, are shown in Figure 11. The corrector function is given in Figure 12. This function could be simplified by XORing node 1,1 with f_{00}, but this trick does not scale up to the more complex structures described in the next section. Combining the phase, conjugate, and anti transformations with the K transformation permits all relations given in Figure 3 to be tested.

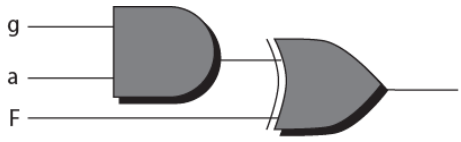

Figure 10. Output Correction for $f_{01} \oplus f_{10} = g$ Relations.

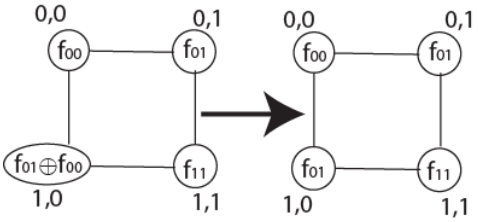

Figure 11. Structure for $f_{01} \oplus f_{00} \oplus f_{10} = \overline{0}$ Relations.

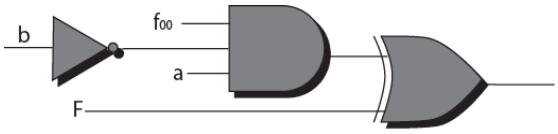

Figure 12. Corrector for $f_{01} \oplus f_{00} \oplus f_{10} = \overline{0}$ Relations.

5 More Complex Logical Structures

As the number of variables represented in the logical structure increases, so does the complexity of the structure. In some algorithms, the search for symmetric variable pairs is done from the top down by starting over with the original function f and computing new cofactors. Instead of doing this, our algorithm proceeds by computing the cofactors of the cofactors already in the structure. Figure 12 shows how the structure of Figure 4 can be expanded to three dimensions if no symmetry is detected between a and b.

In Figure 13, two comparisons are necessary to detect the symmetry between variables a and b. Node 0,1,0 must be compared to node 1,0,0, and node 0,1,1 must be compared to node 1,0,1. There are now three pairs of variables that must be compared, (a,b), (a,c) and (b,c). Dimensions must be examined two at a time, and all planes in that pair of dimensions must be examined.

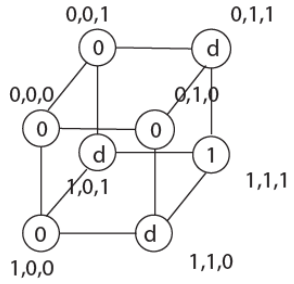

Figure 13. Three Dimensions.

When a symmetric pair is detected in a three-dimensional cube, it is collapsed as shown in Figure 14. In all further operations the two variables a and b will be treated as a composite variable taking the values 0, 1 and 2, corresponding to the number of ones in the two inputs.

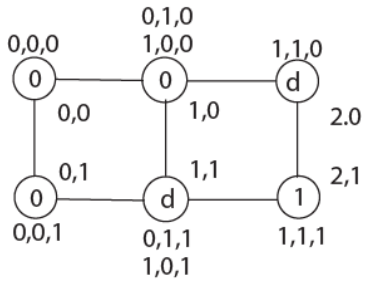

Figure 14.Collapsed 3-Dimensional Structure.

Since all detected symmetries are converted to ordinary symmetries a complex analysis of symmetry transitivities is not needed. The primary problem is detecting symmetries with respect to composite variables. When comparing a composite variable to another variable, there will be more than one anti-diagonal, as in Figure 14. To check for ordinary symmetry, it is necessary that all vertices along each diagonal have the same value. Node 0,1 will be compared to node 1,0 and node 1,1 will be compared to node 2,0. It is permissible for different diagonals to have different values.

When testing for skew and conjugate symmetries, reverse-indexing the dimension corresponding to a composite variable is equivalent to negating each individual variable in the composite. Thus reverse-indexing can be used to detect multi-phase and conjugate symmetries with respect to composite variables.

For anti-symmetries and more general relations, we need the following theorem which clarifies the situation with respect to composite variables.

Theorem 1. Given a Boolean function f, if the relation $f_{01} \oplus f_{10} = \overline{0}/\overline{1}$ is true for variables a and b and if there is an ordinary symmetry between variables b and c then the relation same is true for variables a and c.

Proof. Let f_{000} designate the cofactors of f with respect to the variables a b and c in that order. By assumption, $f_{01x} \oplus f_{10x} = \overline{0}/\overline{1}$ which implies that $f_{010} \oplus f_{100} = \overline{0}/\overline{1}$ and $f_{011} \oplus f_{101} = \overline{0}/\overline{1}$. Because there is an ordinary symmetry between b and c, $f_{x01} = f_{x10}$, which implies that $f_{001} = f_{010}$ and $f_{101} = f_{110}$. Substituting these into the previous two equations, we get $f_{001} \oplus f_{100} = \overline{0}/\overline{1}$ and $f_{011} \oplus f_{110} = \overline{0}/\overline{1}$ which implies that $f_{0x1} \oplus f_{1x0} = \overline{0}/\overline{1}$, the desired condition.

Theorem 1 says that if a condition exists between any element of a composite variable c and another variable v, then that condition exists between v and every other member of c. Theorem 1 also applies to relations of the form $f_{01} \oplus f_{10} = g$ if g is symmetric or if the variables b and c are adjacent. Theorem 2 gives us a similar result for the Kronecker relations.

Theorem 2. Given a Boolean function f, if any of the relations $f_{01} \oplus f_{10} \oplus f_{11} = \overline{0}/\overline{1}$, $f_{01} \oplus f_{10} \oplus f_{00} = \overline{0}/\overline{1}$ $f_{01} \oplus f_{11} \oplus f_{00} = \overline{0}/\overline{1}$, $f_{11} \oplus f_{10} \oplus f_{00} = \overline{0}/\overline{1}$ or $f_{00} \oplus f_{01} \oplus f_{10} \oplus f_{11} = \overline{0}/\overline{1}$ exists between variables a and b, and if an ordinary symmetry exists between variables b and c, then the same relation also exists for variables a and c.

The proof of Theorem 2 is essentially identical to that for Theorem 1.

Theorems 1 and 2 are helpful in determining how to test for non-classical symmetries with composite variables. In Figure 14 we are comparing two composite variables, each of which consists of three mutually symmetric variables. Five diagonals must be compared.

If there is an anti-symmetry between any two variables from the respective composites, Theorem 1 states that there must be a total of nine anti-symmetries, one for each pair of variables. If we break out each of these nine symmetries in the manner of the Theorem 1 proof, we see that the relation $f_x \oplus f_y = \overline{1}$ must exist between every pair of consecutive nodes along each diagonal. Another way of expressing the relation $f_x \oplus f_y = \overline{1}$ is $f_x = f'_y$, which implies that a cofactor and its complement must alternate along each diagonal. General relations of the form $f_{01} \oplus f_{10} = g$ are more complicated because the comparison involves cofactors of g, and if g is not symmetric, the comparisons along a diagonal may not be uniform.

For Kronecker symmetries, applying Theorem 2 shows that the relation must exist for every "square" of nodes along a

diagonal. For example, for the relation $f_{01} \oplus f_{10} \oplus f_{00} = \overline{0}$ to be true between the composite variables of Figure 14, it is necessary that $node_{01} \oplus node_{10} \oplus node_{00} = \overline{0}$, $node_{20} \oplus node_{21} \oplus node_{10} = \overline{0}$, $node_{21} \oplus node_{02} \oplus node_{01} = \overline{0}$, etc. There are nine comparisons in all.

In our comparison algorithm, we can detect Kronecker symmetries between composite variables, but we make no use of these symmetries in practical applications because the output-corrective functions are too complex. Not only do they involve cofactors of the function, but they are non-uniform. The corrective function for Figure 14 would require the computation of nine cofactors, one for each of the variable pairs.

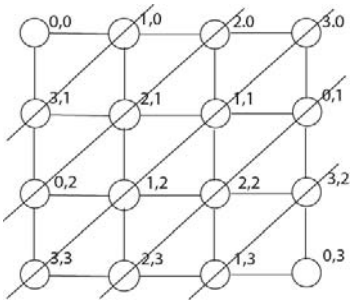

Figure 14. Comparing Composite variables.

For anti-symmetries, the corrective function is simple. Any time an anti-symmetry is detected, the odd rows of the hyperlinear structure are inverted. The variables corresponding to these rows are added to an inversion list. Regardless of the number of anti-symmetries, a single XOR gate is used on the output of the function. The inputs to this gate are the output of the function and the variables on the inversion list.

6 BDDs from State-Spaces

At the end of the symmetry detection process we are left with the state space of a symmetric function and a list of elementary corrective functions. The state-space is converted to a BDD one dimension at a time. The first dimension of Figure 14 would be converted as shown in Figure 15. The leaf-nodes in this diagram are the columns of the state-space of Figure 14. Duplicate leaves are combined, nodes pointing to duplicate leaves are removed and the conversion proceeds in a recursive manner at the leaf nodes. This process is repeated until the leaves contain elementary Boolean values.

7 Experimental Data

To determine the effectiveness of our symmetry detection algorithms, we ran our algorithms on the ISCAS85 benchmarks. [21] Each circuit was first partitioned into a collection of fanout-free networks, each one of which represents a single-output function. Because some of these fanout-free networks are extremely large, they were further partitioned into circuits with no more than eight inputs. Our partitioning algorithm is capable of using any number of inputs as a limit, but we find that a limit of eight tends to maximize the number of symmetries for most circuits. (This issue is discussed in greater depth in [20]). Figure 16 shows the total number of symmetries of each type detected by our algorithms. The K3 column contains the results for the positive three-cofactor relations, and K4 gives the results for the positive four-cofactor relation.

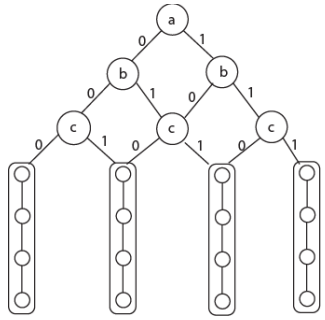

Figure 15. Converting to BDDs.

Although Figure 16 shows a number of Kronecker symmetries for some circuits, these results are misleading because of a phenomenon we call *symmetry masking*. Sometimes detecting one type of symmetry will prevent other types of symmetry from being detected. (Although sometimes the reverse is true, such as for c7552.) Figure 17 shows the number of classical symmetries detected when detection of Kronecker symmetries is suppressed. In several cases, the Kronecker symmetries are being detected at the cost of detecting classical symmetries (see c5315, for example). Because the corrective functions for classical symmetries are far simpler than those for Kronecker symmetries, it is clearly undesirable to detect Kronecker symmetries for these circuits.

This phenomenon is far less pronounced for anti-symmetries, as Figure 18 shows. In most cases the anti-symmetries are detected with little or no suppression of classical symmetries. This is especially true for c6288, for which the number of detected symmetries almost doubles, without suppressing any classical symmetries.

8 Conclusion

We have presented a symmetry detection technique that permits the natural transitivity of classical symmetries to be extended to virtually any cofactor relation. For some types of relations, namely the multi-phase, single-variable and anti symmetries, the cost of this transitivity is negligible. For others, especially the Kronecker relations, the cost is higher.

However, our experimental data suggests that detecting Kronecker symmetries may not provide a significant advantage, even if simple corrective functions could be found.

One area that would bear further investigation is relations of the type $f_{01} \oplus f_{10} = g$ which have corrective functions that are not significantly more complex than those for anti-symmetry.

Circuit	Classical	Anti	K3	K4
c432	103	0	0	0
c499	158	0	0	24
c880	144	20	19	8
c1355	142	104	0	0
c1908	146	0	3	0
c2670	356	34	0	1
c3540	461	7	1	15
c5315	588	32	12	82
c6288	480	464	0	0
c7552	959	89	32	0

Figure 16. Results for all symmetry types.

Circuit	Classical
c432	103
c499	174
c880	152
c1355	142
c1908	146
c2670	358
c3540	464
c5315	669
c6288	480
c7552	953

Figure 17. Classical Symmetries Alone.

Circuit	Classical	Anti
c432	103	0
c499	174	0
c880	152	19
c1355	142	104
c1908	146	0
c2670	357	34
c3540	463	5
c5315	664	32
c6288	480	464
c7552	951	99

Figure 18. Classical and Anti-Symmetries.

References

[1] C. E. Shannon, "The synthesis of two-terminal switching circuits," *Bell System Technical Journal,* Vol.28, *No.1,* pp. 59-98, 1949.

[2] C. R. Edwards and S. L. Hurst, "A digital synthesis procedure under function symmetries and mapping methods," *IEEE Transactions on Computers,* Vol.27, *No.11,* pp. 985-997, 1978.

[3] D. Moller, P. Molitor, R. Drechsler and J. W. G. U. Frankfurt, "Symmetry based variable ordering for ROBDDs," *IFIP Workshop on Logic and Architecture Synthesis,* pp. 47-53, 1994.

[4] C. Scholl, D. Moller, P. Molitor and R. Drechsler, "BDD minimization using symmetries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol.18, *No.2,* pp. 81-100, 1999.

[5] T. Sasao, "A new expansion of symmetric functions and their application to non-disjoint functional decompositions for LUT type FPGAs," *IEEE International Workshop on Logic Synthesis,* pp. 105-110, 2000.

[6] V. N. Kravets and K. A. Sakallah, "Constructive library-aware synthesis using symmetries," *Design Automation and Test in Europe,* pp. 208-213, 2000.

[7] C. C. Tsai and M. Marek-Sadowska, "Boolean functions classification via fixed polarity Reed-Muller forms," *IEEE Transactions on Computers,* Vol.46, *No.2,* pp. 173-186, 1997.

[8] M. Chrzanowska-Jeske, "Generalized symmetric variables," *The 8th IEEE International Conference on Electronics, Circuits and Systems,* pp. 1147-1150, Vol. 3, 2001.

[9] M. Chrzanowska-Jeske, A. Mishchenko and J. R. Burch, "Linear Cofactor Relationships in Boolean Functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol.25, *No.6,* pp. 1011-1023, 2006.

[10] P. M. Maurer, "An application of group theory to the analysis of symmetric gates," Department of Computer Science, Baylor University, Waco, TX 76798, 2009.

[11] V. N. Kravets and K. A. Sakallah, "Generalized symmetries in boolean functions," *IEEE International Conference on Computer Aided Design,* pp. 526-532, 2000.

[12] J. Mohnke, P. Molitor and S. Malik, "Limits of using signatures for permutation independent Boolean comparison," *Formal Methods Syst. Des.,* Vol.21, *No.2,* pp. 167-191, 2002.

[13] C. C. Tsai and M. Marek-Sadowska, "Detecting symmetric variables in boolean functions using generalized reed-muller forms," *IEEE International Symposium on Circuits and Systems,* pp. 287-290, Vol. 1, 1994.

[14] S. L. Hurst, "Detection of symmetries in combinatorial functions by spectral means," *IEE Journal on Electronic Circuits and Systems,* Vol.1, *No.5,* pp. 173-180, 1977.

[15] C. C. Tsai and M. Marek-Sadowska, "Generalized Reed-Muller forms as a tool to detect symmetries," *IEEE Transactions on Computers,* Vol.45, *No.1,* pp. 33-40, 1996.

[16] M. Chrzanowska-Jeske, "Generalized symmetric and generalized pseudo-symmetric functions," *Proceedings of the 6th IEEE International Conference on Electronics, Circuits and Systems,* pp. 343-346, Vol. 1, 1999.

[17] J. Rice and J. Muzio, "Antisymmetries in the realization of boolean functions," *IEEE International Symposium on Circuits and Systems,* pp. 69-72, Vol. 4, 2002.

[18] N. Kettle and A. King, "An anytime algorithm for generalized symmetry detection in ROBDDs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol.27, *No.4,* pp. 764-777, 2008.

[19] D. Moller, J. Mohnke and M. Weber, "Detection of symmetry of boolean functions represented by ROBDDs," *IEEE International Conference on Computer-Aided Design*, pp. 680-684, 1993.

[20] P. M. Maurer, "Conjugate Symmetry," *Formal Methods Syst. Des.*, Vol.38, *No.3,* pp. 263-288, 2011.

[21] F. Brglez, P. Pownall and R. Hum. Accelerated ATPG and fault grading via testability analysis. Presented at Proceedings of IEEE Int. Symposium on Circuits and Systems.