Super Symmetric Boolean Functions

Peter M. Maurer
Dept. of Computer Science
Baylor University
Waco, Texas 76798-7356
Waco, Texas 76798

Abstract – Super symmetry is a type of matrix-based symmetry that extends the concept of total symmetry. Super symmetric functions are “even more symmetric” than totally symmetric functions. Even if a function is not super symmetric, the super symmetric transpose matrices can be used to detect partial super symmetries. These partial symmetries can be mixed arbitrarily with ordinary symmetric variable pairs to create large sets of mutually symmetric variables. In addition, one can detect subsets of super symmetric inputs, which are distinct from partial super symmetries. Super symmetry allows many new types of Boolean function symmetry to be detected and exploited.

1 Introduction

Symmetric Boolean functions have many applications in the field of Electrical Computer Aided Design (ECAD) [ref]. A symmetric Boolean function is a function of \( n \) variables, whose input variables can be rearranged in some fashion without changing the output of the function. An example is \( x_1 x_2 x_3 + x_4 \), (multiplication is AND, and addition is OR) in which the variables \( x_1, x_2 \) and \( x_3 \) can be rearranged arbitrarily.

This concept can be made more precise using permutations [1, 2]. Let \( f \) be an \( n \)-input Boolean function and \( X = \{ x_1, x_2, \ldots, x_n \} \) be its set of input variables. If \( p \) is a permutation on the set \( X \) that leaves \( f \) unchanged, then \( f \) is symmetric and is said to be invariant with respect to \( p \). Also, \( f \) and \( p \) are said to be compatible. The set of all permutations of \( X \) is called the symmetric group of \( X \), and is designated \( S_X \). The symmetry group, \( G_x \), of an \( n \)-input Boolean function, \( f \), is the set of all permutations \( p \in S_X \) that are compatible with \( f \). Because the identity permutation, which leaves \( X \) unchanged, is compatible with every function, \( G_x \) is always non-empty. A function, \( f \), is said to be symmetric if \( G_x \) contains more than one element.

The only thing that affects the structure of \( S_X \) is the size of \( X \). If \( X \) and \( Y \) are two different sets such that \( |X| = |Y| \), then \( S_X \) is isomorphic to \( S_Y \). For simplicity, we will usually assume that \( X = \{1, 2, 3, \ldots, n \} \), and will designate \( S_X \) as \( S_n \). There is a natural mapping between \( \{1, 2, 3, \ldots, n \} \) and sets of variables such as \( \{x_1, x_2, \ldots, x_n\} \) or elements of vectors such as \( (v_1, v_2, \ldots, v_n) \). When applying members of \( S_n \) to these sets, we will assume that the natural mapping between \( \{1, 2, 3, \ldots, n\} \) and the set of indices is being used.

Symmetric Boolean functions were first studied by Shannon [3], who gave us Shannon’s theorem, the basis of most symmetry detection algorithms. Shannon’s theorem is based on the cofactors of a Boolean function, \( f \), which are obtained by setting one or more input variables of \( f \) to constant values. For example, \( x_1 x_2 + x_4 \) is the cofactor obtained by setting \( x_1 = 1 \) in the function \( x_1 x_2 x_3 + x_4 \).

Cofactors can be designated in several different ways. One can specify the variable and the value in a subscript, as in \( f_{x10} \). If there is a natural ordering to the variables, one can specify a list of variable values such as \( f_{x10} \), where the \( x \) represents a variable that has not been replaced. Most often, when the variables in question are understood, we simply use lists of values as in \( f_a, f_1 \) or \( f_{01} \).

Shannon’s theorem states that two input variables, \( x_1 \) and \( x_2 \), of a function \( f \) are symmetric variable pairs if and only if \( f_0 = f_{10} \), where the cofactors are taken with respect to \( x_1 \) and \( x_2 \). The variables of a symmetric pair can be exchanged in arbitrary fashion without altering the output of the function. Symmetric variable pairs are transitive in the sense that if \((x_1, x_2)\) is a symmetric variable pair, and \((x_2, x_3)\), is a symmetric variable pair, then so is \((x_1, x_3)\).

Since [3], there have been much work on detecting and exploiting symmetric functions.[4-24]. Symmetries can be broken into three broad categories, total symmetry which allows the inputs of a function to be permuted arbitrarily, partial symmetry, which allows one or more subsets of inputs to be permuted arbitrarily, and strong symmetry, which includes everything else. Some subclasses of strong symmetry, such as hierarchical symmetry [16], and rotational symmetry [17] have been identified and studied. The Universal Symmetry Detection Algorithm [25] is capable of detecting any type of strong symmetry.

2 Super Symmetry

As pointed out in [26], permutation-based symmetry can be recast in terms of matrices over GF(2). If one views an \( n \)-input function as a function of a single \( n \)-element vector, then traditional symmetry can be defined in terms of permutation matrices on these vectors. Permutation matrices are matrices that have a single 1 in each row and in each column. A permutation matrix is so called because it permutes the elements of a vector without changing them. One can obtain any permutation matrix \( P \) by permuting the rows of the identity matrix, \( I \).

There is a one-to-one correspondence between permutations and permutation matrices. The set of all permutations on a set of \( n \) elements, \( S_n \), and the set of all \( n \times n \) permutation matrices, \( S_{R_n} \), are mathematical groups that are isomorphic to one another. Since the class of \( n \times n \) non-singular matrices is much larger than the class of permutations on \( n \) input variables, matrices can be used to define a much larger class of symmetries than permutations.

For example, matrices can be used to define conjugate symmetry. Let \( S_{R_n} \) be the set of all \( n \times n \) permutation matrices, and let \( M \) be an arbitrary non-singular \( n \times n \) matrix. Then the matrices in the set \( G = \{ M^{-1}NM \mid M \in S_{R_n} \} \) define a new type of symmetry.
called conjugate symmetry. Conjugate symmetry cannot be defined directly in terms of permutations and is a type of matrix-based symmetry.

Super symmetry is another type of matrix-based symmetry that extends the concept of total symmetry and the concept of permutation matrices. We start with $SR_n$, the $n \times n$ permutation matrices. Every matrix $M \in SR_n$ is both a row-permutation and a column-permutation of the identity matrix. For example, if $n = 4$, then every element of $SR_4$ can be constructed by arranging the rows (or columns) $0001$, $0010$, $0100$, and $1000$ in some order. We can expand $SR_n$ by adding an $n+1^{st}$ row containing all ones to the existing set of $n$ rows. Let $HR_n$ be the set of all matrices that can be formed from these $n+1$ rows, without choosing duplicates. $HR_n$ is closed under matrix multiplication, and is isomorphic to the symmetric group $S_{n+1}$. Figure 1 shows an example with $n = 3$. By the same token, we can start with the columns that contain a single 1, and add a column of all 1’s. The set of all matrices that can be formed from these columns, without choosing duplicate columns, is $VR_n$. $VR_n$ is also closed under matrix multiplication, and is isomorphic to $S_{n+1}$. If $n > 2$ then $HR_n \neq VR_n$. We call $HR_n$ and $VR_n$ the super symmetric groups of degree $n$.

$$\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}$$

Figure 1. The Super Symmetric Group $HR_3$.

To prove that $HR_n$ and $VR_n$ are groups isomorphic to $S_{n+1}$ we start with the following theorem which proves that the matrices of $HR_n$ and $VR_n$ are non-singular.

Theorem 1. Every element of $HR_n$ and $VR_n$ is non-singular.

Proof. Let $M \in HR_n$. If $M$ is singular then some subset of the rows of $M$ must sum to zero. If there is no row of all ones, then $M$ is a permutation matrix and non-singular. Let us assume that row $i$ of $M$ is all ones. Other than row $i$, there are $n-1$ rows of $M$, each containing a single 1. These rows are part of some permutation matrix, and therefore, no subset of rows that does not include row $i$ can sum to zero. Other than the 1’s in row $i$, the matrix $M$ contains exactly $n-1$ 1’s. Therefore there must be at least one column that contains no 1’s, except the 1 in row $i$. No sum of rows that includes row $i$ can have a zero in column $i$, because every other row has a zero in this column. Therefore no subset of the rows of $M$ sums to zero, and $M$ is non-singular. Now consider $N \in VR_n$. By a similar argument, we can show that no subset of the columns of $N$ can sum to zero, therefore $N$ is nonsingular.

Now we can prove that $HR_n$ and $VR_n$ are groups isomorphic to $S_{n+1}$.

Theorem 2: $HR_n$ and $VR_n$ are closed under matrix multiplication, and are isomorphic to $S_{n+1}$.

Proof. Let $M, N \in HR_n$ and consider the form of $K = M \times N$. Because $M$ and $N$ are nonsingular, $K$ must be nonsingular. If no row of $M$ is all ones, then $M$ is a permutation matrix. In this case, $K$ is a row-permutation of $N$, and $K \in HR_n$. So let us assume that row $i$ of $M$ is all 1’s. Now, suppose $N$ is a permutation matrix. Because every row of $N$ has a single 1, every row, except row $i$, of $K$ has a single 1. Row $i$ of $K$ is the sum of all rows of $N$, which is a row of all 1’s. Therefore $K \in HR_n$. If $N$ is not a permutation matrix, then it must have a row, $j$ of all 1’s. In this case, the rows of $K$, except for row $i$ must be a permutation of the rows of $N$, not including row $i$. Row $i$ of $K$ must be the sum of the rows of $N$. Every column of $N$, except one, must have exactly 2 ones. The remaining column must have a single one. Therefore the sum of the rows of $N$ must contain a single one in some position, and zeroes elsewhere. Because the product is non-singular, row $i$ cannot duplicate any other row of $K$. Therefore, $K$ must either be a permutation matrix, or a permutation matrix with one row replaced by a row of all ones. Thus $K \in HR_n$, and $HR_n$ is closed under multiplication. A similar argument shows that $VR_n$ is also closed under multiplication. To show that $HR_n$ is isomorphic to $S_{n+1}$, it suffices to show that $HR_n$ is the set of permutations of a set of size $n+1$. This follows from the fact that every matrix in $HR_n$ is a permutation of the $n+1$ rows used to form the elements of $HR_n$, each element, $M$, of $HR_n$ has $n$ rows from the set of $n+1$ rows. The missing row is always unique, and we can imagine it as being appended as the $n+1^{th}$ row of $M$. Thus $HR_n$ is isomorphic to $S_{n+1}$.

A similar argument on the columns of the elements of $VR_n$ shows that $VR_n$ is also isomorphic to $S_{n+1}$.

Any finite set of non-singular matrices that is closed under multiplication is a group. Because $HR_n$ and $VR_n$ are groups, they can serve as the symmetry group of certain functions. We say that a function $f$ is super symmetric if either $HR_n$ or $VR_n$ leaves $f$ invariant. If we wish to be more specific, we will call $f$ $H$-super symmetric or $V$-super symmetric.

3. Boolean Orbits

Let $G$ be a group of $n \times n$ matrices. Two $n$-element vectors $v$ and $w$ are said to be in the same Boolean orbit of $G$ if there is a matrix $M \in G$ such that $v \times M = w$. Being in the same Boolean orbit is an equivalence relation that breaks the set of all $n$-element vectors into a collection of disjoint subsets. The Boolean orbits of a group can be used to determine whether a group $G$ is the compatible with
a function \( f \). The function \( f \) is compatible with \( G \) if and only if \( f \) maps every element of each Boolean orbit of \( G \) to the same value. For example, the symmetric group \( S_3 \) has the Boolean orbits \{\{0(0,0)\}, \{0(0,1),(1,0),(1,0)\}, \{0(1,1),(0,1),(1,0)\}, \{0(1,1)\}\}. A 3-input function \( f \) is totally symmetric if and only if \( f \) maps the three vectors \{\{0(0,1),(0,1),(1,0)\}\} to the same value, and the three vectors \{\{0(1,1),(0,1),(1,0)\}\} to the same value.

The Universal Symmetry Detection algorithm can detect any type of symmetry as long as the Boolean orbits of that symmetry are known. The Boolean orbits of V and H super symmetry are relatively easy to compute. Since every super symmetric function is also totally symmetric, all vectors of the same weight must be contained in a single orbit. The Boolean orbits of V and H super symmetry can be obtained by combining the Boolean orbits of total symmetry.

Let us first derive the Boolean orbits of H super symmetry. We will designate the set of all vectors of weight \( k \) as \( W_k \). The sets \( W_k \) are just the Boolean orbits of total symmetry. We will designate the Boolean orbits of H super symmetry as \( O_k \), where \( k \) is the weight of the lightest vector in \( O_k \). Note that if an orbit \( O_k \) contains any vector of weight \( k \), then \( W_k \subseteq O_k \). In particular, \( W_k \subseteq O_k \). Consider the orbit \( O_k \). This orbit must contain a single vector, since every linear transformation maps the zero vector onto itself. The orbit \( O_k \) contains all vectors of weight 1 and must also contain the vector of all 1’s. Let \( v \) be an n-element vector of weight 1, and let \( M \in HR_n \).

When a vector \( v = (a_1,...,a_n) \) is multiplied by a matrix \( V \), the result is \( v' = (a_1,...,a_{i-1},p,a_{i+1},...,a_n) \), where \( p \) is the parity of \( v \). (i.e., \( p = 1 \) if the number of bits in \( v \) is odd.) If \( a_i = 0 \) and \( p = 0 \), or if \( a_i = 1 \) and \( p = 1 \), then \( v = v' \). If \( a_i = 0 \) and \( p = 1 \) then the weight of \( v' \) is one larger than that of \( v \). If \( a_i = 1 \) and \( p = 0 \) then the weight of \( v' \) is one smaller than that of \( v \). Note that the weight of \( v \) can increase only if it is odd, and can decrease only if it is even. Thus \( Q = W = W_{1} \), where \( i \) is odd, \( i \) running from 1 to \( m \) where \( m \) is the largest odd number less than or equal to \( n \). The other matrices of \( VR_n \) will not affect these orbits because they are either permutation matrices that do not change the weight of a vector, or they are permutation matrices with a single column set to ones. Such matrices combine a permutation of \( v \) with parity insertion, and do not change the orbits described above.

We have created a super symmetry detection module to the universal symmetry detector using the Boolean orbits described above.

### 4 Symmetric Variable Pairs

Although the universal symmetry detection algorithm can detect super symmetry, super symmetric functions are comparatively rare. The same is true, of course, for totally symmetric functions. However, when a function is not totally symmetric, it may be partially symmetric, and using symmetric variable pairs, we can detect such partial symmetries. By the same token, we can detect super symmetric variable pairs and partial super symmetries. The super symmetric variable pairs can be mixed arbitrarily with ordinary symmetric variable pairs.

Ordinary symmetric variable pairs correspond to a type of a permutation called a transposition. A transposition of a set, \( X \), is a permutation that swaps two elements of \( X \), leaving everything else fixed. In the matrix domain, a transposition corresponds to a transpose matrix. A transpose matrix swaps two elements of an input vector, leaving all other elements fixed. We designate a transpose matrix that swaps elements \( i \) and \( j \) of a vector as \( T_{ij} \). Every row, \( k \), of \( T_{ij} \) except rows \( i \) and \( j \), is identical to row \( k \) of the identity matrix. Row \( i \) of \( T_{ij} \) has a 1 in column \( j \) and zeros elsewhere. Row \( j \) has a 1 in column \( i \) and zeros elsewhere. Figure 2 has several examples of transpose matrices.

\[
\begin{pmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
\end{pmatrix}
\quad
\begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\quad
\begin{pmatrix}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\quad
\begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
\end{pmatrix}
\quad
\begin{pmatrix}
0 & 1 \\
1 & 0 \\
0 & 0 \\
\end{pmatrix}
\]

Figure 2. Some transpose matrices.

Super symmetry introduces \( 2n \) new transpose matrices known as the super symmetric transpose matrices. Half of these matrices are taken from \( HR_n \) and half are taken from \( VR_n \).

In permutation matrices, we consider a row containing a 1 in position \( i \) and zeros elsewhere to represent the \( i^{th} \) input variable. Alternatively, we could consider a column containing a 1 in the \( i^{th} \) position to represent the \( i^{th} \) input variable. In the super symmetric matrices, we consider the row of all 1’s or a column of all 1’s to represent an \( n+1 \)’ “invisible” variable. In \( HR_n \), a super symmetric transpose matrix is a matrix that is identical to the identity matrix except for row \( i \), which is a row of all 1’s. In \( VR_n \) a super
symmetric transpose matrix is identical to the identity matrix except for column \( i \) which is a column of all 1’s. We designate these matrices as \( H_i \) and \( V_i \) respectively. Figure 3 gives some examples of such matrices.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{pmatrix} \quad \begin{pmatrix}
1 & 1 & 1 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
\end{pmatrix} \quad \begin{pmatrix}
1 & 1 & 1 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
\end{pmatrix} \quad \begin{pmatrix}
1 & 1 \\
0 & 1 \\
1 & 0 \\
1 & 0 \\
\end{pmatrix}
\]

Figure 3. Super Symmetric Transpose Matrices.

For any ordinary transpose matrix \( T_{ij} \), the matrix is self-inverting. That is, \( T_{ij} T_{ij} = I \). As the Theorem 3 shows, the same is true for the matrices \( V_i \) and \( H_i \).

Theorem 3. \( H_i H_i = I \) and \( V_i V_i = I \) for all \( 1 \leq i \leq n \).

Proof: Since \( H_i \) is identical to the identity matrix, except for row \( i \), every row of \( H_i H_i \) is identical to the identity matrix, except for row \( i \). Because row \( i \) of \( H_i \) is all ones, row \( i \) of \( H_i H_i \) is the sum of the rows of \( H_i \). Every column of \( H_i \) contains exactly two 1’s, except for column \( j \) which contains exactly one 1. Thus the sum of the rows of \( H_i \) has a 1 in column \( i \) and zeroes elsewhere, and is equal to row \( i \) of the identity matrix.

Similarly, since every column \( k \) of \( V_i \), except for column \( i \), is identical to column \( k \) of the identity matrix, every column \( k \) of \( V_i V_i \) is identical to column \( k \) of the identity matrix. Because column \( i \) of \( V_i \) is all ones, column \( i \) of \( V_i V_i \) is the sum of the columns of \( V_i \). Every row of \( V_i \), except row \( i \) has exactly two 1’s. Row \( i \) has exactly one 1. Therefore the sum of the columns of \( V_i \) has a 1 in row \( i \) and zeroes elsewhere, and column \( i \) of \( V_i V_i \) is identical to column \( i \) of the identity matrix.

It is convenient to think of the matrices \( H_i \) and \( V_i \) as being transpose matrices between \( x_i \) and the “invisible” \( n+1 \)th variable, \( x_{n+1} \). This makes the transitivity of the new matrices more obvious. For example, because of transitivity, a function is H super symmetric if it is compatible with \( T_{i,1}, T_{i,3}, \ldots, T_{i,n} \) and \( H_i \). For V super symmetry, we substitute \( V_i \) for \( H_i \).

Another important and useful property of the super symmetric transpose matrices is that the conjugate of any matrix \( H_j \) with another matrix \( H_j \) \((i \neq j)\) is an ordinary transpose matrix. The same is true for matrices \( V_j \) and \( V_j \), as the following theorem shows.

Theorem 4. Suppose \( i \neq j \). Then \( H_j^{-1} H_j H_j = T_{i,j} \) and \( V_j^{-1} V_j V_j = T_{i,j} \).

Proof: By Theorem 3, \( H_j^{-1} = H_j \) and \( V_j^{-1} = V_j \), so \( H_j^{-1} H_j H_j = H_j H_j \) and \( V_j^{-1} V_j V_j = V_j V_j \) have the following form. Since every row of \( H_j \), except row \( j \), is identical to the corresponding row of the identity matrix, every row, except row \( j \) of \( H_j H_j \) is identical to the corresponding row of \( H_j \). Because row \( j \) of \( H_j \) is all ones, row \( j \) of \( H_j H_j \) is the sum of the rows of \( H_j \). Every column of \( H_j \) has exactly two 1’s, except for column \( i \), which has exactly one 1. Thus row \( j \) of \( H_j H_j \) has a one in column \( i \) and zeroes elsewhere. Row \( i \) of \( H_j H_j \) contains all 1’s. We can use the structure of \( H_j H_j \) to deduce the structure of \( H_j H_j H_j \). Because every row of \( H_j H_j \) except rows \( i \) and \( j \) is identical to the corresponding row of the identity matrix, every row of \( H_j H_j H_j \), except rows \( i \) and \( j \), is identical to the corresponding row of the identity matrix. Because row \( j \) has a 1 in column \( i \) and zeroes elsewhere, row \( j \) of \( H_j H_j H_j \) is identical to row \( i \) of \( H_j \), and has a 1 in column \( i \) and zeroes elsewhere. Because row \( i \) of \( H_j H_j \) contains all 1’s, row \( i \) of \( H_j H_j H_j \) is the sum of the rows of \( H_j \). Every column of \( H_j \) has exactly two ones, except for column \( j \) which has exactly one 1. Therefore row \( i \) of \( H_j H_j H_j \) has a 1 in column \( j \) and zeroes elsewhere. Therefore \( H_j H_j H_j \) is the transpose matrix \( T_{i,j} \).

Now consider the structure of \( V_j V_j \). Because every column of \( V_j \) is identical to the corresponding column of the identity matrix, except for column \( j \), every column of \( V_j V_j \) is identical to the corresponding column of \( V_j \), except for column \( j \). Because column \( j \) of \( V_j \) is all ones, column \( j \) of \( V_j V_j \) is equal to the sum of the columns of \( V_j \). Every column of \( V_j \) contains exactly two 1’s, except for column \( i \) which has exactly one 1. Thus column \( j \) of \( V_j V_j \) has a 1 in row \( i \) and zeroes elsewhere. Column \( i \) of \( V_j V_j \) contains all ones. We can now deduce the structure of \( V_j V_j V_j \). Since every column of \( V_j V_j \), except for columns \( i \) and \( j \), is identical to the corresponding column of the identity matrix, every column of \( V_j V_j V_j \), except for columns \( i \) and \( j \), is identical to the corresponding column of \( V_j \). But these columns are identical to the corresponding columns of the identity matrix, so every column of \( V_j V_j V_j \), except for columns \( i \) and \( j \), is identical to the corresponding column of the identity matrix. Column \( j \) of \( V_j V_j V_j \) is equal to column \( i \) of \( V_j \), which has a 1 in row \( i \) and zeroes elsewhere. Because column \( i \) of \( V_j V_j V_j \) is all ones, column \( i \) of \( V_j V_j V_j \) is the sum of the columns of \( V_j \). Every row of \( V_j \) has exactly two 1’s, except for row \( j \), which has exactly one 1. Thus the sum of the columns of \( V_j \) has a 1 in row \( j \) and zeroes elsewhere. Thus \( V_j V_j V_j \) is the transpose matrix \( T_{i,j} \).

Let \( f \) be an \( n \)-input function with input variables \( \{x_1, x_2, \ldots, x_n\} \). To determine whether \( f \) is compatible with \( H_i \), we select some variable other than \( x_i \), say \( x_j \) with \( i \neq j \), and conditionally invert every variable except \( x_j \) itself with respect to \( x_j \). These conditional inversions can be done simultaneously using the matrix \( H_j \). We compute \( f'(v) = f(H_j(v)) \). The function \( f \) is compatible with \( H_i \) if and only if \((x_i, x_j)\) is a symmetric variable.
pair of $f'$. The correctness of this procedure stems from the fact that if $i \neq j$ then $H_i^T H_j = T_{ij}$. Super symmetry can be viewed as a type of conjugate symmetry requiring multiple simultaneous conditional inversions.

Given the same function, $f$, we can determine whether $f$ is compatible with $V_i$ by selecting any input variable other than $x_i$, say $x_j$ with $i \neq j$, and conditionally invert $x_j$ with respect to every variable other than $x_i$ itself. This gives us the new function, $f'(v) = f(V_i(v))$. The function $f$ is compatible with $V_i$ if and only if $(x_i, x_j)$ is a symmetric variable pair of $f'$. Again, the correctness of this procedure depends on the fact that $V_i^T V_j = T_{ij}$.

Because super symmetric transpose matrices can be equated with a type of conjugate symmetry, they can be detected and utilized by the hyperlinear algorithm for digital simulation [26, 27], and by other algorithms that detect symmetry using symmetric variable pairs.

5 Sub-Symmetries

For a Boolean function $f$ to possess $X$ symmetry in variables $\{x_1, x_2, \ldots, x_n\}$ every cofactor of the form $f_{x_{i_1}, x_{i_2}, \ldots, x_{i_n}}$ must possess $X$ symmetry. We usually do this by ensuring the symmetry relations exist between cofactors of the form $f_{x_{i_1}, x_{i_2}, \ldots, x_{i_n}}$. It is possible for an $n$-input function to be super symmetric in any proper subset of its input variables, and it is possible for a function to have several subsets of variables in which it is super symmetric.

This is not the same as partial symmetry, because the super symmetric variable pairs involve all inputs of a function, while sub-super symmetric variables involve only a subset of variables. It is possible to test a subset of variables for super symmetry, and to test the same subset for compatibility with the super symmetric transpose matrices of the sub-symmetry. This gives us many more opportunities to detect symmetries in a Boolean function, because there are $2^n - 2$ proper subsets of variables, and $\sum_{i=1}^{n-1} 2^n i = 2 \left(\frac{n(n-1)}{2} - 1\right) = n^2 - n - 2$ additional super symmetric transpose matrices.

6 Experimental Data

To determine the prevalence of super symmetry in real circuits, we tested the ISCAS 85 benchmarks for the presence of super symmetries. We tested for total super symmetry, for super symmetric variable pairs, and for sub symmetries. The results of our tests are given in Figure 4. These results show that super symmetries do indeed exist in real circuits, and are, in fact, quite numerous. The results for super symmetric variable pairs and for sub symmetries are especially encouraging. Because, in several cases, the number of symmetries exceeds the number of functions, it is clear that there are many functions that exhibit multiple sub-super symmetries and that there are functions that are compatible with many super symmetric variable pairs.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Super Sym.</th>
<th>Var. Pairs</th>
<th>Sub-Sym.</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432</td>
<td>78</td>
<td>213</td>
<td>1097</td>
</tr>
<tr>
<td>c499</td>
<td>0</td>
<td>56</td>
<td>728</td>
</tr>
<tr>
<td>c880</td>
<td>122</td>
<td>33</td>
<td>902</td>
</tr>
<tr>
<td>c1355</td>
<td>288</td>
<td>44</td>
<td>704</td>
</tr>
<tr>
<td>c1908</td>
<td>158</td>
<td>59</td>
<td>5326</td>
</tr>
<tr>
<td>c2670</td>
<td>276</td>
<td>90</td>
<td>3145</td>
</tr>
<tr>
<td>c3540</td>
<td>710</td>
<td>1310</td>
<td>2093</td>
</tr>
</tbody>
</table>

Figure 4. Experimental Results.

7 Conclusion

The various aspects of super symmetry allow many different types of Boolean function symmetry to be detected and exploited. In addition to super symmetry itself we have partial super symmetries which are generated by the super symmetric transposition matrices. These partial symmetries can be mixed and matched in an arbitrary fashion with ordinary symmetric variable pairs. In addition, there are super-super symmetries and partial sub-super symmetries which greatly expand the opportunity for detecting and exploiting symmetries in a Boolean function.

What is even more exciting, super symmetry allows us to exploit more of the full power of matrix-based symmetry. For example, for 4-input functions, there are 24 permutations of the inputs, but 20160 non-singular $4 \times 4$ matrices. There are obviously many more kinds of matrix-based symmetry than permutation-based symmetry, and super symmetry is only one of these.

We expect this work to be the basis of much more extended work in the future.

8 References

[12] Y. Hu, V. Shih, R. Majumdar and L. He, "Exploiting Symmetries to Speed Up SAT-Based Boolean Matching for


