Canonical Form-Based Boolean Matching and Symmetry Detection in Logic Synthesis and Verification

Afshin Abdollahi
University of Southern California
afshin@usc.edu

Massoud Pedram
University of Southern California
pedram@ceng.usc.edu

Abstract – *An efficient and compact canonical form is proposed for the Boolean matching problem under permutation and complementation of variables. In addition an efficient algorithm for computing the proposed canonical form is provided. The efficiency of the algorithm allows it to be applicable to large complex Boolean functions with no limitation on the number of input variables as apposed to previous approaches, which are not capable of handling functions with more than seven inputs. Generalized signatures are used to define and compute the canonical form while simple symmetries of variables is used to minimize the computational complexity of the algorithm. All other symmetry relations are resulted as a bi-product of the canonical form computation. Experimental results demonstrate the efficiency and applicability of the proposed canonical form.*

I Introduction

Boolean matching is the problem of determining whether a Boolean function can be functionally equivalent to another one under a permutation of its inputs and complementation of some of its inputs. Boolean matching algorithms have many applications in logic synthesis including cell-library binding where it is necessary to repeatedly determine whether some part (cluster) of a Boolean network can be realized by any of the cells in a library [1]. Boolean matching is a critical and CPU-intensive task, and therefore, there have been many efforts to effectively solve the problem [2]. Boolean functions that are equivalent under negation of inputs are N-equivalent, under permutation of inputs are P-equivalent, and under both stated conditions, are NP-equivalent [3]. If all we consider permutation of inputs and complementation of inputs and output the functions are NPN-equivalent. An exhaustive method for Boolean matching is computationally expensive since the complexity of such an algorithm for n-variable functions is $O(n!2^{n+1})$.

Boolean matching algorithms can be classified into two categories: pair-wise matching algorithms and algorithms based on canonical forms of functions. Pair-wise Boolean matching algorithms are based on a semi-exhaustive search where the search space is pruned by the use of some signatures which are computed from some properties of Boolean functions [2]. A signature in general is a description of (one or more) input variables of a Boolean function that is independent of the permutation or complementation of the variables of the function. To match a function against a cell library, pair-wise matching algorithms often need to perform pair-wise matching of the function with all the library cells. Therefore, these algorithms can only cope with libraries of modest size.

Boolean matching algorithms that belong to the second category compute some canonical form for Boolean functions [5] - [10]. These algorithms are based on the fact that two functions match if and only if their canonical forms are the same. Burch and Long introduced a canonical form for matching under complementation and a semi-canonical form for matching under permutation of the variables [5]. In their solution, in order to handle complementation and permutation of inputs simultaneously, a large number of forms for each cell are required. Other researchers, including Wu et al. [6], Debnath and Sasao [8], and Ciric and Sechen [9] have also proposed canonical forms that are applicable to Boolean matching under permutation of the variables only but do not handle complementation of inputs. Hinsberger and Kolla [7] and Debnath and Sasao [10], have introduced a canonical form for solving the general Boolean matching problem. However their approach is mainly based on manipulating the truth table of the function and employing a table look-up, which results an enormous space complexity, thus restricting their algorithm to library cells with seven or fewer input variables.

In this paper a new canonical form for representing Boolean functions is introduced. The proposed canonical form for an arbitrary Boolean function is the unique Boolean function that is obtained after applying some *canonicity-producing* (CP) transformation on the input variables. The canonical forms of NPN-equivalent Boolean functions are identical. In particular, an effective technique is presented for generating this canonical form. The proposed method is based on using *generalized signatures* (signatures of one or more variables) to find a CP phase assignment and ordering for variables. From here on, phase assignment and ordering for variables is referred to as a *transformation* on variables. For most Boolean functions, single-variable and two-variable signatures are enough to recognize all variables (i.e., to obtain a CP

transformation.) However, use of single-variable and two-variable signatures alone may not result on a canonical input transformation. In this paper it is shown that, by using generalized signatures of one or more variables, it is always possible to create a CP transformation on variables of the function.

Experimental results provided in this paper demonstrate that the proposed approach for computing the canonical form does not have the limitations of previous works; i.e., it computes the canonical form of a Boolean function with any number of variables under both permutation and complementation of variables. An important advantage of the proposed technique is the way it handles and uses the symmetry of variables to minimize the complexity of the algorithm compared to some of the previous approaches which are not able to consider symmetries [7][10]. Hence, the proposed technique is applicable to logic verification of large circuits and to technology mapping with a large ASIC library with cells of any number of inputs.

In section II, definitions and terminology are introduced. In section III, symmetry relations are discussed. In section IV signatures that are utilized in the method are described. In section V the canonical forms is defined and the details of computing the canonical form is provided followed by experimental results and conclusions in sections VI and VII.

II Preliminaries

We denote vectors and matrices in capital letters i.e., $X = (x_1, x_2, \ldots, x_n)$ where X denotes a vector of n Boolean variables. A *literal* is a variable, x, or its complement $\bar{x}$. We will refer to literal x as the positive phase of variable x and to literal $\bar{x}$ as its negative phase. In general a literal can be denoted as x^p. The phase of the literal is described using the Boolean variable $p \in B = \{0,1\}$ where $p = 1 \Rightarrow x^p = x$ (positive phase) and $p = 0 \Rightarrow x^p = \bar{x}$ (negative phase.)

For the variable vector $X = (x_1, x_2, \ldots, x_n)$ and phase vector $P = (p_1, p_2, \ldots, p_n)$ (where X and $P \in B^n$ contain the same number of variables and $p_i \in B$) the phase assignment of P to variable vector X is defined as $X^P = (x_1^{p_1}, x_2^{p_2}, \ldots, x_n^{p_n})$.

As an example for $X = (x_1, x_2, x_3)$ and $P = (0,1,0)$ the result of phase assignment is $X^P = (\bar{x}_1, x_2, \bar{x}_3)$. Also for $X = (\bar{x}_1, x_2, \bar{x}_3)$ and $P = (0,1,1)$ the result of phase assignment is $X^P = (\bar{x}_1, \bar{x}_2, x_3)$.

For a set, A, we use $|A|$ to denote the cardinality of A. Since $|B^n| = 2^n$, the number of possible different phase assignments to $X = (x_1, x_2, \ldots, x_n)$ is 2^n.

The *identity phase assignment* is denoted by $1 = (1,1,\ldots,1)$. Obviously, $X^1 = X$. The inverse of phase assignment is itself i.e., $(X^P)^P = X$.

The cascade of two phase assignments $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_n)$ is $P \overline{\oplus} Q = (p_1 \overline{\oplus} q_1, p_2 \overline{\oplus} q_2, \ldots, p_n \overline{\oplus} q_n)$ since $(X^P)^Q = X^{P \overline{\oplus} Q}$. (The '$\overline{\oplus}$' is the XNOR operation i.e., for Boolean variables x and y, $x = y \Leftrightarrow x \overline{\oplus} y = 1$.)

A permutation is a rearrangement of the elements of an ordered list or a vector. First, we define permutation on a set of the form $A_n = \{1,2,\ldots,n\}$, which will serve as indices of an ordered list or a vector.

Definition: A permutation π on set $A_n = \{1,2,\ldots,n\}$ is a bijection from A_n to A_n i.e., $\pi : A_n \to A_n$ and $i \neq j \Rightarrow \pi(i) \neq \pi(j)$. For a subset $B \subset A_n$, the range of a permutation π on domain B is defined as: $\pi(B) = \{\pi(i) | i \in B\}$.

Based on this definition, $\pi(A_n) = A_n$ which is equivalent to the reversibility of permutation π as a function i.e., $\forall j \in A_n, \exists i \in A_n, s.t. : \pi(i) = j$. The identity permutation, ι, on $A_n = \{1,2,\ldots,n\}$ is defined as follows: $\forall i \in A_n, \iota(i) = i$. We denote the set of all permutations on A_n by Π_n. The *cascade* of two permutations π_1 and π_2 on $A_n = \{1,2,\ldots,n\}$, denoted by $\pi_1\pi_2$, is defined as: $\forall i \in A_n, \pi_1\pi_2(i) = \pi_1(\pi_2(i))$. The cascade operation among permutations is not a commutative operation i.e., in general, $\forall \pi_1, \pi_2 \in \Pi_n, \pi_1\pi_2 \neq \pi_2\pi_1$. However, it is an associative operation $\forall \pi_1, \pi_2, \pi_3 \in \Pi_n, \pi_1(\pi_2\pi_3) = (\pi_1\pi_2)\pi_3$. Permutations are reversible. The inverse of a permutation π, denoted by π^{-1}, is defined as: $\pi(i) = j \Rightarrow \pi^{-1}(j) = i$.

Based on these properties, set Π_n with cascade operation creates a *group*.[1] The number of members of this group is $n!$. Any permutation $\pi \in \Pi_n$ can be applied to a vector of length n e.g., $X = (x_1, x_2, \ldots, x_n)$. The result of application of permutation $\pi \in \Pi_n$ to vector is determined

[1] A group G is a finite or infinite set of elements together with a binary operation (called the group operation) that together satisfy the four fundamental properties of closure, associativity, identity, and inverse property.

based on the relations $\pi(x_i) = x_{\pi(i)}$ and $\pi(X) = \pi(x_1, x_2, \ldots, x_n) = (x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$ which, is a rearrangement the entries of vector X. Next, we describe NP transformations comprising of phase assignment and permutations.

As an example, for the permutation $\pi \in \Pi_3$ where $\pi(1) = 2$, $\pi(2) = 3$ and $\pi(3) = 1$, we have $\pi(x_1, x_2, x_3) = (x_2, x_3, x_1)$.

Definition: An NP *transformation* on vector $X = (x_1, x_2, \ldots, x_n)$ is defined as a phase assignment followed by a permutation. In particular, for phase assignment $P = (p_1, p_2, \ldots, p_n)$ and permutation $\pi \in \Pi_n$, the NP transformation T_π^P on vector $X = (x_1, x_2, \ldots, x_n)$ is computed as follows: $T_\pi^P(X) = \pi(X^P)$.

Now then, $T_\pi^P(x_1, x_2, \ldots, x_n) = \pi(x_1^{p_1}, x_2^{p_2}, \ldots, x_n^{p_n}) = (x_{\pi(1)}^{p_{\pi(1)}}, x_{\pi(2)}^{p_{\pi(2)}}, \ldots, x_{\pi(n)}^{p_{\pi(n)}})$, or more simply, $T_\pi^P(X) = [\pi(X)]^{\pi(P)}$. The set of all NP transformations on a vector of size n is denoted by $\Gamma_n = \{T_\pi^P \,|\, P \in B^n, \pi \in \Pi_n\}$.

As an example for $P = (0,1,1)$ and $\pi(x_1, x_2, x_3) = (x_2, x_3, x_1)$, we have $T_\pi^P(x_1, x_2, x_3) = (\bar{x}_2, \bar{x}_3, x_1)$.

The number of transformations in Γ_n is $|\Gamma_n| = |B^n| \times |\Pi_n| = 2^n n!$. The identity transformation is denoted by T_ι^1 where $1 = (1,1,\ldots,1)$ is the identity phase assignment and ι is the identity permutation. Obviously, $T_\iota^1(X) = X$ for any vector X. The *cascade* of two transformations $T_{\pi_1}^{P_1}$ and $T_{\pi_2}^{P_1}$, denoted by $T_{\pi_1}^{P_1} T_{\pi_2}^{P_2}$, is defined as $T_{\pi_1}^{P_1} T_{\pi_2}^{P_2}(X) = T_{\pi_1}^{P_1}(T_{\pi_2}^{P_2}(X))$. One can verify that $T_{\pi_1}^{P_1} T_{\pi_2}^{P_2} = T_\pi^P$ where $\pi = \pi_1 \pi_2$ and $P = \pi_1(P_1) \overline{\oplus} \pi(P_2)$. The inverse of transformation T_π^P, denoted by $(T_\pi^P)^{-1}$, satisfies the relation $T_\pi^P (T_\pi^P)^{-1} = (T_\pi^P)^{-1} T_\pi^P = T_\iota^1$. Based on the relation $T_\iota^1 = T_{\pi'}^{P'} T_\pi^P = T_{\pi'\pi}^{\pi'(P') \overline{\oplus} \pi'\pi(P)}$, one can infer that $(T_\pi^P)^{-1} = T_{\pi^{-1}}^{\pi(P)}$. Set Γ_n with cascade operation creates a group.

In the remainder of this paper, when there is no ambiguity, we denote a transformation T_π^P by T for brevity. In addition, we may denote an NP transformation on vector as $T_\pi^P X$ (or TX) instead of $T_\pi^P(X)$ (or $T(X)$.) We usually denote the identity transformation by $I = T_\iota^1$. Finally, with regard to permutations, πX will refer to $\pi(X)$.

Let function $f(X)$ be a single-output completely-specified Boolean function of $X=(x_1,x_2,\ldots,x_n)$ i.e., $f:B^n\to B$. The onset of $f(X)$ is a subset of its domain, $F\subset B^n$, that results in $f(X)=1$. We will denote the size of onset of $f(X)$ by $|f(X)|$ i.e., $|f(X)|=|F|$.

The **cofactor** of $f(X)$ with respect to literal $x_i^{p_i}$ is a function $f_{x_i^{p_i}}(X_i)$ of $X_i=(x_1,x_2,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ defined as $f_{x_i^{p_i}}(X_i)=f(X)|_{x_i^{p_i}=1}=f(X)|_{x_i=p_i}$. A cube is the Boolean conjunction of some literals, $q=x_{i_1}^{p_{i_1}}x_{i_2}^{p_{i_2}}\cdots x_{i_m}^{p_{i_m}}$. The cofactor of $f(X)$ with respect to a cube q is a function of variables in X that are not present in q (positive or negative phase) is defined as $f_q(X_q)=f(X)|_{q=1}=f(X)|_{x_{i_1}=p_{i_1},x_{i_2}=p_{i_2},\ldots,x_{i_m}=p_{i_m}}$ where $X_q=X-\{x_{i_1},x_{i_2},\ldots,x_{i_m}\}$.

Consider two functions, $f(X)$ and $g(X)$, defined over the same variable set $X=(x_1,x_2,\ldots,x_n)$.

Definition: Two functions $f(X)$ and $g(X)$ are *P-equivalent*, denoted by $f\overset{P}{\equiv}g$, if there exists a permutation π such that $f(X)\overline{\oplus}\,g(\pi X)$ is a tautology.

Definition: Two functions $f(X)$ and $g(X)$ are *NP-equivalent*, denoted by $f\overset{NP}{\equiv}g$, if there exists an NP transformation T such that $f(X)\overline{\oplus}\,g(TX)$ is a tautology.

The most general type of equivalence is when we also consider phase assignment of the output. We will denote the phase assignment $p\in B$ to function $f(X)$ by $(f(X))^p$ or $f^p(X)$ for short.

Definition: Two functions $f(X)$ and $g(X)$ are *NPN-equivalent*, denoted by $f\equiv g$, if there exists an NP transformation T and an output phase assignment $p\in B$ such that $f(X)\overline{\oplus}\,g^p(TX)$ is a tautology i.e., $\exists T\in\Gamma_n,\exists p\in B,\forall X\in B^n, f(X)=g^p(TX)$.

Example: Let $f(x_1,x_2,x_3)=x_1x_2+\bar{x}_1x_3$ and $g(x_1,x_2,x_3)=x_1x_2+x_1x_3+x_2\bar{x}_3$. It is easy to see that $f(X)=\bar{g}(TX)$ where $X=(x_1,x_2,x_3)$ and $T(X)=(\bar{x}_2,\bar{x}_3,x_1)$. Thus, $f(X)$ and $g(X)$ are thus NPN-equivalent.

NPN-equivalence is an equivalence relation. Boolean matching is often defined in terms of P, NP or NPN-equivalence. In principle, P, NP, and NPN-equivalence can be reduced to $n!$, $2^n n!$ and $2^{n+1}n!$ tautology checks.

We use the symbol $\forall_x$ and $\exists_x$ to designate the consensus and the smoothing operators with respect to variable *x*, respectively. Recall that the consensus operation corresponds to universal quantification and is computed as $\forall_x f = f_x f_{\bar{x}}$, while the smoothing operation corresponds to existential quantification and is computed as $\forall_x f = f_x + f_{\bar{x}}$. Consensus (smoothing) with respect to an array of variables can be computed by repeated application of single-variable consensus (smoothing) operations.

III Symmetry relations

In this section we discuss variable symmetries in Boolean functions. Functional symmetries provide significant benefits for multiple tasks in synthesis and verification. As will be explained in detail later in this paper, concepts of Boolean matching and symmetry are closely related. In the Boolean matching algorithm that will be provided in this paper, this relationship manifests itself in two levels. First, simple types of symmetries (that are inexpensive to discover) are utilized to reduce the complexity of the Boolean matching algorithm. Second, the proposed Boolean matching algorithm will generate (as a bi-product) the remaining (more complicated) symmetries.

Symmetries provide insights into the structure of the Boolean function that can be used to facilitate operations on it. They can also serve as a guide for preserving that structure when the function is transformed in some way. In the context of Boolean matching problem, symmetries that we explore are variable permutations, with possible complementation that leave the function unchanged. In the presence of functional symmetries, several design problems (e.g., circuit restructuring, checking satisfiability, and computing sequential reachability) are considerably simplified. Hence, interest in functional symmetries has been keen since the early days of logic design [12]. In the context of logic synthesis which we view as a process that transforms an initial representation of the function (e.g., sum of products representation or binary decision diagram representation) into a final implementation as a multi-level Boolean network of primitive cells selected from a given ASIC cell library, when guided by knowledge of functional symmetries, such a process yields higher quality circuit realizations of the function [13].

In [14] functional symmetry is exploited to optimize a circuit implementation for low power consumption and delay under an area increase constraint. Another benefit of knowledge about functional symmetries is that it can help produce better variable orders for Binary Decision

Diagrams (BDDs) and related data structures (e.g., Algebraic Decision Diagrams). The size of the BDD of a Boolean function can be significantly reduced if symmetric variables are placed in adjacent positions. Based on this observation a specialized sifting procedure for dynamic variable ordering was proposed in [15]. This plays a crucial role in symbolic model checking.

In this paper we study symmetries in the most general from i.e., considering input permutation, input phase assignment, and output phase assignment which to the best of our knowledge has not been studied thoroughly enough in the past.

Definition: A function $f(X)$ where $X=(x_1,x_2,\ldots,x_n)$ is symmetric with respect to an NP transformation $T\in\Gamma_n$ on its inputs if there exists an output phase assignment $q\in B$ such that $f(X)=f^q(TX)$.

We will refer to such a transformation a *symmetry-producing (SP) transformation* and denote the set of all SP transformations by $S_f=\{T\in\Gamma_n \,|\, \exists q,\ f(X)=f^q(TX)\}$.

S_f creates a sub-group of Γ_n. As mentioned before, some types of symmetry are easily detectable and are discovered before the Boolean matching algorithm. We start by discussing these types of symmetries.

Definition (Simple Symmetry): For a function $f(X)$ where $X=(x_1,x_2,\ldots,x_n)$, two variables x_i and x_j are said to be *symmetric*, denoted as $x_i\equiv x_j$, if $f(X)$ is invariant under an exchange of x_i and x_j i.e., $f(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n)=f(x_1,\ldots,x_j,\ldots,x_i,\ldots,x_n)$.

In the case of simple symmetry, the phase of the output always remains unchanged. The NP transformation $T_\pi^P\in S_f$ associated with this simple symmetry has the following effect: $T_\pi^P(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n)=(x_1,\ldots,x_j,\ldots,x_i,\ldots,x_n)$.

A similar type of simple symmetry between variables x_i and x_j is when the following condition holds: $f(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n)=f(x_1,\ldots,\bar{x}_j,\ldots,\bar{x}_i,\ldots,x_n)$.

In this case we use notation $x_i\equiv\bar{x}_j$, or equivalently, $\bar{x}_i\equiv x_j$. We shall refer to x_i and x_j as being symmetric in this case as well.

Example: For the function $f(X)=(x_1+x_2)(\bar{x}_3+x_4)$ we have $x_1\equiv x_2$ and $x_3\equiv\bar{x}_4$.

To account for both types of symmetry with a unified notation, we use $x_i \equiv x_j^p$. When $p=1$, the expression indicates that $x_i \equiv x_j$ whereas $p=0$ implies that $x_i \equiv \bar{x}_j$ i.e., $x_i \equiv x_j^p \Leftrightarrow f(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n) = f(x_1,\ldots,x_j^p,\ldots,x_i^p,\ldots,x_n)$.

Variables x_i and x_j are called symmetric if $\exists_p x_i \equiv x_j^p$. We will use W_{ij}^p (or W_{ji}^p) to denote the NP transformation $W_{ij}^p(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n) = (x_1,\ldots,x_j^p,\ldots,x_i^p,\ldots,x_n)$. With this notation $x_i \equiv x_j^p \Leftrightarrow W_{ij}^p \in S_f$.

It is well known, and can be readily shown by using Boole's expansion theorem [5], that condition $x_i \equiv x_j^p$ is equivalent to $f_{x_i \bar{x}_j^p} \equiv f_{\bar{x}_i x_j^p}$. This equation serves as the computational check for first-order symmetry between variables x_i and x_j in function $f(X)$.

The symmetry relation, $\exists_p x_i \equiv x_j^p$, is an equivalence relation. Hence, it is possible to partition variables $x_1, x_2, \ldots, x_n$ into equivalence classes, which we will refer to as *symmetry classes*. An overview of such a procedure, which is composed of two nested loops that iterate on the variables, is as follows. We denote symmetry classes by $C_1, C_2, \ldots, C_m$, where m is the number of classes. The first step is to create $C_1 = \{x_1\}$ where x_1 is considered the seed variable for class C_1. Next, every variable x_i that is symmetric to x_1 will be added to C_1. The first remaining variable, say x_j, is used to initialize $C_2 = \{x_j\}$. Next, symmetric variables to x_j are added to C_2. This procedure continues until all variables are partitioned into symmetry classes $C_1, C_2, \ldots, C_m$. Symmetry classes will include all information about simple symmetries. For example, given symmetry classes $C_1, C_2, \ldots, C_m$, one can infer that if $x_i, x_j \in C_k$, then there exists a phase assignment $p \in B$ such that $x_i \equiv x_j^p$. However, the symmetry classes do not include information as to whether $p=1$ or $p=0$.

One way to include phase information in symmetry classes is to choose appropriate phases for variables while forming classes one at a time. For example, consider a class C_i with seed $x_j \in C_i$. If there exists a variable x_k that is symmetric to x_j i.e., $x_j \equiv x_k^p$ then literal x_k^p will be added to C_i. This is because we chose positive phase $x_j^1 = x_j$ for the seed of C_i. If we were to

choose $x_j^0 = \bar{x}_j$ as the seed of C_i, then in the case of $x_j \equiv x_k^p$ literal $x_k^{\bar{p}}$ will be added to C_i since $x_j \equiv x_k^p \Rightarrow \bar{x}_j \equiv x_k^{\bar{p}}$. Suppose $C_i = \{x_1^{p_1}, x_2^{p_2}, \ldots, x_k^{p_k}\}$ is a symmetry class generated in this manner. Based on the previous discussion, negating the phases of the literals $\{x_1^{\bar{p}_1}, x_2^{\bar{p}_2}, \ldots, x_k^{\bar{p}_k}\}$ will create an alternate symmetry class of the same variables. We shall denote this alternate class by $\overline{C_i}$ which introduces the notion of phase assignment for symmetry classes i.e., $C_i^q = \{x_j^{p_j \bar{\oplus} q} \,|\, x_j^{p_j} \in C_i\}$.

The algorithm for generating first-level symmetry classes is given below. In this algorithm, we choose positive phases for the seeds of all classes.

Algorithm Gen_1st_Order_Symm ( )

$i \leftarrow 1$;
while $X \neq \{\}$ do {
  $C_i \leftarrow \{\}$; select $x \in X$;
  for $\forall y \in X$ do {
    if $(x \equiv y^p)$ then $C_i \leftarrow C_i \cup \{y^p\}$;
  }
  $X \leftarrow X - C_i$;
  $i \leftarrow i + 1$;
}

For the symmetry classes generated in this manner, literals of a class do not require any phase assignment to become symmetric (the current phases of literals will be fine) i.e., $x_i^{p_i}, x_j^{p_j} \in C_k \Rightarrow x_i^{p_i} \equiv x_j^{p_j}$.

Example: For function $f(X) = (x_1 + x_2)(\bar{x}_3 + x_4)$, there exist two symmetry classes: $C_1 = \{x_1, x_2\}$ and $C_2 = \{x_3, \bar{x}_4\}$.

In the remainder of this paper we shall denote literals by simple letters such as x or y which does not necessary mean that the phase of literal is positive. With this convention the previous relation may be written as: $x, y \in C_k \Rightarrow x \equiv y$.

The classes generated by Gen_1st_Order_Symm are *maximal* in the sense that for every class C_i no other literal $y \notin C_i$ is symmetric to the literals of the class C_i i.e., $x \in C_i, x \equiv y \Rightarrow y \in C_i$

So far we have discussed simple symmetries which correspond to NP transformations that involve only two variables. In the sequel we present a key theorem, which provides a valuable insight for handling and enumerating symmetries. First, we present a lemma that will be useful in proving the main theorem.

In the following we will denote the k^{th} element of vector TX by $[TX]_k$. Furthermore, $[TX]_k^p$ will denote $([TX]_k)^p$.

Lemma 1: For any NP transformation $T_\pi^P \in \Gamma_n$ and $W_{ij}^q \in \Gamma_n$, $(T_\pi^P)^{-1} W_{ij}^q T_\pi^P = W_{\pi(i)\pi(j)}^{q'}$ where $q' = q \overline{\oplus}\, p_{\pi(i)} \overline{\oplus}\, p_{\pi(j)}$.

Proof: Based on previous discussions $[T_\pi^P X]_k = x_{\pi(k)}^{p_{\pi(k)}}$ and $[(T_\pi^P)^{-1} X]_k = x_{\pi^{-1}(k)}^{p_{\pi^{-1}(\pi(k))}} = x_{\pi^{-1}(k)}^{p_k}$. We compute $[(T_\pi^P)^{-1} W_{ij}^q T_\pi^P X]_k$ for different values of $k \in \{1,2,\ldots n\}$. First we consider values of k such that $k \notin \{\pi(i), \pi(j)\}$ and then deal with $k \in \{\pi(i), \pi(j)\}$.

For $k \notin \{\pi(i), \pi(j)\}$, $[(T_\pi^P)^{-1} W_{ij}^q T_\pi^P X]_k = [W_{ij}^q T_\pi^P X]_{\pi^{-1}(k)}^{p_k} = [T_\pi^P X]_{\pi^{-1}(k)}^{p_k} = (x_{\pi(\pi^{-1}(k))}^{p_{\pi(\pi^{-1}(k))}})^{p_k} = x_k$

The second equality follows from $\pi^{-1}(k) \notin \{i, j\}$.

For $k = \pi(i)$, $[(T_\pi^P)^{-1} W_{ij}^q T_\pi^P X]_{\pi(i)} = [W_{ij}^q T_\pi^P X]_i^{p_{\pi(i)}} = [T_\pi^P X]_j^{q \overline{\oplus} p_{\pi(i)}} = x_{\pi(j)}^{q \overline{\oplus} p_{\pi(i)} \overline{\oplus} p_{\pi(j)}} = x_{\pi(j)}^{q'}$. In addition, for $k = \pi(j)$, $[(T_\pi^P)^{-1} W_{ij}^q T_\pi^P X]_{\pi(j)} = [W_{ij}^q T_\pi^P X]_j^{p_{\pi(j)}} = [T_\pi^P X]_i^{q \overline{\oplus} p_{\pi(j)}} = x_{\pi(i)}^{q \overline{\oplus} p_{\pi(j)} \overline{\oplus} p_{\pi(i)}} = x_{\pi(i)}^{q'}$. Hence we showed that $(T_\pi^P)^{-1} W_{ij}^q T_\pi^P (x_1, \ldots, x_{\pi(i)}, \ldots, x_{\pi(j)}, \ldots, x_n) = (x_1, \ldots, x_{\pi(j)}^{q'}, \ldots, x_{\pi(i)}^{q'}, \ldots, x_n)$ which proves that $(T_\pi^P)^{-1} W_{ij}^q T_\pi^P = W_{\pi(i)\pi(j)}^{q'}$. ■

Let x_i and x_j denote two symmetric variables of function $f(X)$. Consider a general symmetry relation that involves more than two variables i.e., consider $f(X) = f(T_\pi^P X)$. In the following we will explore the effect of $T_\pi^P \in S_n$ on symmetric variables.

Every NP transformation $T_\pi^P \in \Gamma_n$ on X can be regarded as a mapping function on literals with the specification $T_\pi^P(x_i) = x_{\pi(i)}^{p_{\pi(i)}}$.

We define the effect of NP transformation T_π^P on literal x_i^q as $T_\pi^P(x_i^q) = (T_\pi^P(x_i))^q$.

Lemma 2: Let function $f(X)$ be symmetric with respect to NP transformation T_π^P i.e., $T_\pi^P \in S_f$; Mappings of two symmetric variables x_i and x_j under T_π^P are symmetric i.e., $x_i \equiv x_j^q \Leftrightarrow T_\pi^P(x_i) \equiv (T_\pi^P(x_j))^q$.

Proof: Since S_f is a subgroup, $T_\pi^P \in S_f \Rightarrow (T_\pi^P)^{-1} \in S_f$. Furthermore, since variables x_i and x_j are symmetric, $x_i \equiv x_j^q \Rightarrow W_{ij}^q \in S_f$. Based on the previous Lemma and the fact that S_n is a subgroup, $W_{\pi(i)\pi(j)}^{q'} = (T_\pi^P)^{-1} W_{ij}^q T_\pi^P \in S_f$ where $q' = q \overline{\oplus} p_{\pi(i)} \overline{\oplus} p_{\pi(j)}$, which proves that $x_{\pi(i)} \equiv x_{\pi(j)}^{q'}$. By applying phase assignment $p_{\pi(i)}$ to both sides of $x_{\pi(i)} \equiv x_{\pi(j)}^{q'}$, one obtains $x_{\pi(i)}^{p_{\pi(i)}} \equiv x_{\pi(j)}^{q' \overline{\oplus} p_{\pi(i)}} = x_{\pi(j)}^{p_{\pi(j)}}$ or $T_\pi^P(x_i) \equiv (T_\pi^P(x_j))^q$, which proves the lemma ■

Now we will investigate the effect of NP transformation $T_\pi^P \in S_f$ on simple symmetry classes. The range of an NP transformation T on a symmetry class (or any other subset of literals) is defined as $T(C_k) = \{T(x) | x \in C_k\}$, where x in general represents a literal (with positive or negative phase) rather than a variable i.e. there is variable x_i with phase p such that $x = x_i^p$.

Theorem 1: Let function $f(X)$ be symmetric with respect to NP transformation T i.e., $T \in S_f$ and let C_k be a first order maximal symmetric class of variables of $f(X)$. The range of T on C_k (i.e., $T(C_k)$) will be a maximal symmetry class.

Proof: Based on the previous lemma, any pair of literals, x and y, of $T(C_k)$ are symmetric.

Recall that for two literals in a symmetry class, the symmetry does not require additional phase assignment since appropriate phases have already been assigned to the literals while generating the symmetry classes. Now we will prove that $T(C_k)$ is *maximal* by showing if there is a literal y symmetric to literal $x \in T(C_k)$ (i.e., $x \equiv y$), then y is a literal in $T(C_k)$.

Since S_n is subgroup, $T \in S_f \Rightarrow T^{-1} \in S_f$. From the previous lemma, $x \equiv y \Rightarrow T^{-1}(x) \equiv T^{-1}(y)$. From $x \in T(C_k)$ it can be seen that $T^{-1}(x) \in C_k$ and since C_k is maximal : $T^{-1}(x) \equiv T^{-1}(y) \Rightarrow T^{-1}(y) \in C_k \Rightarrow y \in T(C_k)$. This proves the theorem. ■

The theorem has a strong implication, that is, any NP transformation $T_\pi^P \in S_f$ maps maximal symmetry classes to other maximal symmetry classes. This result can be considered as a constraint for any $T_\pi^P \in S_f$. It is especially important in the process of identifying NP transformations of S_f since it will limit the space of transformations to be explored. In other words, to explore possible NP transformations $T_\pi^P \in S_f$, it is sufficient to only explore NP transformations that are specified in terms of higher order symmetry classes instead of individual variables. Since the number of classes is usually considerably fewer than the number of variables, this theorem tends to greatly reduce the search space.

Let $C_1, C_2, \ldots, C_m$ represent the maximal symmetry classes for variables of function $f(X)$. The corresponding NP transformation $T_\pi^P \in S_f$ must satisfy $\forall C_i \exists C_j, T(C_i) = C_j$.

IV Signatures

Conventionally, a signature (a filter or a necessary condition) is defined as some characteristics of a Boolean function with respect to one of its input variables. We shall refer to such a signature as a first order signature (or 1st-signature) since it only depends on one input variable. First order signatures have been used to identify variables that can be exchanged (permuted) without affecting the function itself, i.e., any possible correspondence between the input variables of two functions is restricted to a correspondence between variables with the same 1st-signature. So, if each variable of a function has a unique 1st-signature, then there can be at most one possible correspondence to any of the variables of some other function.

That is why the quality of any 1st-signature is characterized by its ability to be a unique identification of a variable of a function and, of course, by its ability to be computed fast. The 1st-signatures that have been introduced in literature differ in terms of their quality figure. Although the 1st order signatures have been successful in a large number of practical cases, they do not utilize the full potential of signatures in the Boolean matching problem. There is no set of 1st-signatures that can uniquely identify all the variables. However, this goal can be achieved by using higher order signatures as described below.

The 1st-signatures have been traditionally defined for variables. However, since we intend to consider phase assignment in addition to permutation of input variables, we define the 1st-signatures for literals (as opposed to variables.)

A well-known 1st-signature for a literal x of a Boolean function $f(X)$ is the "minterm" count of the ONSET of the cofactor of this function w.r.t. x i.e, $|f_x|$.

In pair-wise matching methods (for checking P-equivalence), a 1st-signature must be able to make out an input variable x_i independent of input variable permutation so that it can establish a correspondence between variable x_i of $f(X)$ with a variable x_j of some other Boolean function $g(X)$. It only makes sense to try to establish a correspondence between these two variables only if variable x_i of $f(X)$ has the same 1st-signature as variable x_j of $g(X)$.

The main idea of this pair-wise matching approach is clear: If we are able to compute a unique signature for each input variable of $f(X)$, then the variable mapping problem will have been solved – there is only one or no possible variable correspondence for P-equivalence of function $f(X)$ with any other function $g(X)$. If we find for each variable of $f(X)$ a variable of $g(X)$ that has the same unique signature, then we will have established a correspondence. Otherwise, we will know immediately that these two functions are not P-equivalent.

The main problem that arises in this paradigm is when more than one variable of a function $f(X)$ has the same 1st-signature, it is not possible to distinguish between these variables, i.e. there is no unique correspondence that can be established with the inputs of some other function.

In this paper we will generalize the concept of first order signatures to higher order signatures that have complete expressive power to handle the Boolean matching problem.

Recall that a cube (product term) is the conjunction of some literals. Let $a_1, a_2, \ldots, a_k$ be literals created from variables $x_1, x_2, \ldots, x_n$ i.e., $\forall a_i, \exists x_j, \exists p, a_i = x_j^p$ with the restriction that both phases of the same literal are not present in $a_1, a_2, \ldots, a_k$ i.e., for each variable x_i at most one of literals x_i and $\bar{x}_i$ belong to $\{a_1, a_2, \ldots, a_k\}$.

Definition: The k^{th} order signature of function $f(X)$ with respect to literals $a_1, a_2, \ldots, a_k$ is the minterm count of cofactor of $f(X)$ with respect to cube $q = a_1 a_2 \ldots a_k$ i.e., $|f_q(X_q)|$ where X_q denotes those variables of X that are not in $\{a_1, a_2, \ldots, a_k\}$ in any phase. The 0^{th} order signature is $|f(X)|$.

We will refer to a cube $q = a_1 a_2 \ldots a_k$ as a *positive cube* if literals $a_1, a_2, \ldots, a_k$ are in their positive phases i.e., $\forall a_i, \exists x_j, a_i = x_j$, or equivalently, $\{a_1, a_2, \ldots, a_k\} \subset \{x_1, x_2, \ldots, x_n\}$.

We also refer to signatures associated with positive cubes as positive signatures.

With respect to variables $x_1, x_2, \ldots, x_n$, the number of *k*-literal cubes (or the number of k^{th}-order signatures) is $\binom{n}{k}2^k$ whereas the number of positive k^{th}-order signatures is $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

In the following we introduce a method to enumerate positive signatures of function $f(X)$.

A *k*-literal cube can be presented as $q = x_{\alpha_1} x_{\alpha_2} \cdots x_{\alpha_k}$ where $\{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset \{1,2,\ldots n\}$.

We impose the constraint that $\alpha_1 < \alpha_2 < \ldots < \alpha_k$. Obviously, for each *k*-literal cube it is always possible to find $\alpha_1, \alpha_2, \ldots, \alpha_k$ that satisfy this constraint.

Before we proceed further, we must define the lexicographical comparison of two vectors.

Definition: Consider two vectors $A = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ and $B = (\beta_1, \beta_2, \ldots, \beta_k)$ where $\{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset \{1,2,\ldots n\}$ and $\{\beta_1, \beta_2, \ldots, \beta_k\} \subset \{1,2,\ldots n\}$. Let i be the smallest index such that $\alpha_i \neq \beta_i$. Then the order relation '$\prec$' between A and B is defined as $\alpha_i < \beta_i \Rightarrow A \prec B$.

With this definition, an order relation can be defined between *k*-literal positive cubes.

Definition: Consider two cubes $q_A = x_{\alpha_1} x_{\alpha_2} \cdots x_{\alpha_k}$ and $q_B = x_{\beta_1} x_{\beta_2} \cdots x_{\beta_k}$ where $1 \le \alpha_1 < \alpha_2 < \ldots < \alpha_k \le n$ and $1 \le \beta_1 < \beta_2 < \ldots < \beta_k \le n$. The order relation '$\prec$' between q_A and q_B is defined as $(\alpha_1, \alpha_2, \ldots, \alpha_k) \prec (\beta_1, \beta_2, \ldots, \beta_k) \Rightarrow q_A \prec q_B$.

We denote the set of all *k*-literal positive cubes by $Q^k = \{x_{\alpha_1} x_{\alpha_2} \cdots x_{\alpha_k} \,|\, 1 \le \alpha_1 < \alpha_2 < \ldots < \alpha_k \le n\}$. Since we have defined an ordering for members of Q^k, these members (which correspond to *k*-literal positive cubes) can be represented as $q_1^k, q_2^k, \ldots, q_{n_k}^k$ where the superscript k indicates the number of literals in each cube and the following ordering is satisfied by their indices: $q_1^k \prec q_2^k \prec \ldots \prec q_{n_k}^k$. Notice that $n_k = |Q^k| = \binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Every possible positive cube can be denoted as q_i^k. The set of all positive cubes is denoted by $Q = Q^0 \cup Q^1 \cup Q^2 \cup \cdots \cup Q^k = \{q_i^k \,|\, 1 \le k \le n, 1 \le i \le n_k\}$.

Now we extend the definition of ordering to members of Q. For two cubes q_i^k and q_j^l if $k \neq l$ then $k < l \Rightarrow q_i^k \prec q_j^l$. Now, if $k = l$ then $i < j \Rightarrow q_i^k \prec q_j^l$.

In this ordering, single literal cubes appear first followed by two literal and higher literal cubes i.e., $q_1^0 \prec \overbrace{q_1^1 \prec q_2^1 \prec \ldots \prec q_{n_1}^1}^{Q^1} \prec \overbrace{q_1^2 \prec q_2^2 \prec \ldots \prec q_{n_1}^2}^{Q^2} \prec \ldots \prec \overbrace{q_1^{n-1} \prec q_2^{n-1} \prec \ldots \prec q_{n_{n-1}}^{n-1}}^{Q^{n-1}} \prec \overbrace{q_{n_n}^n}^{Q^n}$

which can also be represented as (except for $q_1^0 = \{\}$):

$$\overbrace{x_1 \prec x_2 \prec \ldots \prec x_n}^{Q^1} \prec \overbrace{x_1x_2 \prec x_1x_3 \prec \ldots \prec x_1x_n}^{Q^2} \prec \ldots \prec \overbrace{x_1x_2\cdots x_{n-1} \prec \ldots \prec x_2x_3\cdots x_n}^{Q^{n-1}} \prec \overbrace{x_1x_2\cdots x_n}^{Q^n}$$

We are now ready to introduce the signature vector for a function $f(X)$.

Definition: For the function $f(X)$ where $X = (x_1, x_2, \ldots, x_n)$ with positive cubes q_i^k, the *signature vector* denoted by V^f is defined as follows:

$$V^f = (|f|, \overbrace{|f_{q_1^1}|, |f_{q_2^1}|, \ldots, |f_{q_{n_1}^1}|}^{1^{st}-signatures}, \overbrace{|f_{q_1^2}|, |f_{q_2^2}|, \ldots, |f_{q_{n_2}^2}|}^{2^{nd}-signatures}, \ldots, \overbrace{|f_{q_1^{n-1}}|, |f_{q_2^{n-1}}|, \ldots, |f_{q_{n_{n-1}}^{n-1}}|}^{(n-1)^{nd}-signatures}, |f_{q_{n_n}^n}|)$$

which can equivalently be presented as:

$$V^f = (|f|, \overbrace{|f_{x_1}|, |f_{x_2}|, \ldots, |f_{x_n}|}^{1^{st}-signatures}, \overbrace{|f_{x_1x_2}|, |f_{x_1x_3}|, \ldots, |f_{x_{n-1}x_n}|}^{2^{nd}-signatures}, \ldots, \overbrace{|f_{x_1\ldots x_{n-1}}|, \ldots, |f_{x_2\ldots x_n}|}^{(n-1)^{th}-signatures}, \overbrace{|f_{x_1\ldots x_n}|}^{n^{th}-signature})$$

In the following we present an important theorem, which proves that the signature vector of a function is unique i.e., two different functions have different signature vectors i.e., $V^f = V^g \Leftrightarrow f(X) = g(X)$.

To prove this claim, we first prove that values of all signatures of all orders can be obtained from the signature vector (which only include positive signatures.) This vector eventually specifies the function $f(X)$ uniquely for all $X \in B^n$.

Lemma 3: Values of all k^{th}-signatures can be uniquely obtained from the $(k\text{-}1)^{th}$-signatures and positive k^{th}-signatures.

Proof: Given all $(k\text{-}1)^{th}$-signatures and positive k^{th}-signatures we want to prove that all k^{th}-signatures can be computed. A k^{th}-signature in general can be denoted by $|f_q|$ where $q = x_{\alpha_1}^{p_1} x_{\alpha_2}^{p_2} \cdots x_{\alpha_k}^{p_k}$, $\alpha_i \in \{1,2,\ldots,n\}$ and $p_i \in B$. The proof is by induction on the number of negative

literals of $q = x_{\alpha_1}^{p_1} x_{\alpha_2}^{p_2} \cdots x_{\alpha_k}^{p_k}$ denoted by n_q. The value of n_q ranges from 0 to k. For $n_q = 0$, the cube is positive and since positive k^{th}-signatures are given, the claim is correct for $n_q = 0$. Assume that it is correct for $n_q = m-1$ i.e., all k^{th}-signatures associated with cubes of $m-1$ negative literals are computed. Now we will compute the k^{th}-signatures associated with cubes of m negative literals. Let's denote such a cube by $x_{\alpha_1}^{p_1} x_{\alpha_2}^{p_2} \cdots x_{\alpha_{i-1}}^{p_{i-1}} x_{\alpha_i}^{p_i} x_{\alpha_{i+1}}^{p_{i+1}} \cdots x_{\alpha_k}^{p_k}$. Assume that $x_{\alpha_i}^{p_i}$ is one of the negative literals i.e., $p_i = 0$. Let's denote this cube by $q_0 = x_{\alpha_1}^{p_1} x_{\alpha_2}^{p_2} \cdots x_{\alpha_{i-1}}^{p_{i-1}} x_{\alpha_i}^{0} x_{\alpha_{i+1}}^{p_{i+1}} \cdots x_{\alpha_k}^{p_k}$. Accordingly we create the cube $q_1 = x_{\alpha_1}^{p_1} x_{\alpha_2}^{p_2} \cdots x_{\alpha_{i-1}}^{p_{i-1}} x_{\alpha_i}^{1} x_{\alpha_{i+1}}^{p_{i+1}} \cdots x_{\alpha_k}^{p_k}$ in which the phase of x_{α_i} is positive. One can easily verify that $| f_{q_1} | + | f_{q_0} | = | f_q |$ where $q = x_{\alpha_1}^{p_1} x_{\alpha_2}^{p_2} \cdots x_{\alpha_{i-1}}^{p_{i-1}} x_{\alpha_{i+1}}^{p_{i+1}} \cdots x_{\alpha_k}^{p_k}$. Hence $| f_{q_0} |$ can be computed as $| f_{q_0} | = | f_q | - | f_{q_1} |$ Since q is a (k-1)-literal cube and q_1 contains $m-1$ negative literals the claim is proven by induction. ■

Theorem 2: For a function $f(X)$, *signature vector* V^f *uniquely and completely* specifies function $f(X)$.

Proof: Since the signature vector includes all positive signatures, it can be seen that the values of $| f_q |$ can be computed for all possible cubes q by using induction on the number of literals in q and the previous Lemma. However, for this proof, we are only interested in *n*-literal cubes since they deliver sufficient information to specify function f for all points $(p_1, p_2, \ldots, p_n) \in B^n$ i.e., $f(p_1, p_2, \ldots, p_n) = | f_q |$ where $q = x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}$. ■

Based on this theorem, the necessary and sufficient condition for two functions $f(X)$ and $g(X)$ to be equal is that $| f_q | = | g_q |$ for all positive cubes q (which are presented in the signature vectors of $f(X)$ and $g(X)$.)

In this part we investigate the implication of this theorem on the general symmetry relation. Consider NP transformation T which corresponds to the symmetry relation $f(X) = f(TX)$. Let's denote $g(X) = f(TX)$. Since functions $f(X)$ and $g(X)$ are equal, their signature vectors are equal i.e., $| f_q | = | g_q |$ for all $q = x_{\alpha_1} x_{\alpha_2} \cdots x_{\alpha_k}$. The cofactors f_q and g_q can be expressed as $f_q = f(X)\Big|_{x_{\alpha_1} = x_{\alpha_2} = \cdots = x_{\alpha_k} = 1}$ and $g_q = g(X)\Big|_{x_{\alpha_1} = x_{\alpha_2} = \cdots = x_{\alpha_k} = 1} = f(TX)\Big|_{x_{\alpha_1} = x_{\alpha_2} = \cdots = x_{\alpha_k} = 1} = f(X)\Big|_{T(x_{\alpha_1}) = T(x_{\alpha_2}) = \cdots = T(x_{\alpha_k}) = 1}$.

Let's denote $t_i = T(x_i)$ and $T(q) = t_{\alpha_1} t_{\alpha_2} \cdots t_{\alpha_k}$ for $q = x_{\alpha_1} x_{\alpha_2} \cdots x_{\alpha_k}$. The necessary and sufficient condition for NP transformation T to be symmetric is $f(X) = f(TX) \Leftrightarrow | f_q | = | f_{T(q)} | \quad \forall q \in Q$.

Notice that if $f(X) = f(TX)$, then $| f_q | = | f_{T(q)} |$, where q denotes any cube (not only a positive cube.) A result of the previous theorem is that it proves that a sufficient condition for $f(X) = f(TX)$ is that $| f_q | = | f_{T(q)} |$ for positive cubes only. We will revisit these results in future sections.

V Canonical form

In this section we present a canonical form for Boolean matching under NPN-equivalence. As was proved in previous chapters, NPN-equivalence is an equivalence relation that partitions the set of all single output Boolean functions into equivalence classes. Let's consider an NPN-equivalence class by $E = \{f_1(X), f_2(X), \ldots, f_m(X)\}$. Every two functions in E are symmetric to each other, i.e., $f_i(X) \in E, f_j(X) \in E \Rightarrow f_i(X) \equiv f_j(X)$ and any function symmetric to some function in E is also in E. The Boolean matching problem under NPN-equivalence is reduced to that of verifying whether or not two target Boolean functions, $f(X)$ and $g(X)$, belong to the same NPN-equivalence class.

In the canonical form based Boolean matching, a unique representative is selected for every class called the NPN-representative of the class. Let's denote the NPN-representative of a class E by $F(X)$.

Definition: The NPN-representative $F(X)$ of a class E is defined as the NPN-representative (or the canonical form) of all functions $f_1(X), f_2(X), \ldots, f_m(X)$ in E. Let's denote the canonical form of a function $f(X)$ by $F(X)$ (i.e., we use capital letters for canonical forms.) We have: $F(X) = F_1(X) = F_2(X) = \ldots = F_m(X)$. Notice that $F(X)$ is one of $f_1(X), f_2(X), \ldots, f_m(X)$ i.e., $F(X) \in E$.

The NPN-representative $F(X)$ is selected among $f_1(X), f_2(X), \ldots, f_m(X)$ based on some criteria that makes $F(X)$ unique. For example, one way is to define a total ordering for functions $f_1(X), f_2(X), \ldots, f_m(X)$ and select the maximum or minimum (with respect to the defined order) as the NPN-representative (canonical form.)

Observation: Two functions $f(X)$ and $g(X)$ are NPN-equivalent if and only if they have the same canonical form i.e., $f(X) \equiv g(X) \Leftrightarrow F(X) = G(X)$.

The NPN-equivalence class that includes a function $f(X)$, denoted by E_f is the set of all functions that are NPN-equivalent to $f(X)$. Hence E_f can be created by applying all NP transformations and output phase assignments to $f(X)$ i.e., $E_f = \{f^q(TX) | q \in B, T \in \Gamma_n\}$.

The number of different NP transformations and output phase assignments to $f(X)$ is $|B| \times |\Gamma_n| = 2^{n+1} n!$; However $|E_f|$ is in general less than $|B| \times |\Gamma_n|$ because of the symmetry relations discussed in previous chapters.

Since $f(X)$ and $F(X)$ are NPN-equivalent, there is an NP transformation T such that $\exists_q f^q(TX) = F(X)$; however, T is not the only such NP transformation.

Definition: We call the set of NP transformations T such that $\exists_q f^q(TX) = F(X)$, the canonicity-producing (CP) transformations : $C_f = \{T \in \Gamma_n | \exists q, f^q(TX) = F(X)\}$.

We present an algorithm to compute the canonical form of a given NPN-equivalence class as well as the set of all CP transformations C_f. We will show that the set of symmetry-producing (SP) transformations S_f can be easily obtained from C_f. The importance of identifying all NP transformations in S_f was explained in the previous section.

For any set $S \subset \Gamma_n$ of NP transformations and transformation $T \in \Gamma_n$, we define TS and ST as follows: $TS = \{TT' | T' \in S\}$ and $ST = \{T'T | T' \in S\}$. If $S \subset \Gamma_n$ is a subgroup, then TS is called the *left coset* of S determined by T and ST is called the *right coset* of S determined by T.

Lemma 4: For function $f(X)$, let T and T' be two CP transformations. $T'T^{-1}$ is a SP transformation i.e., $T \in C_f, T' \in C_f \Rightarrow T'T^{-1} \in S_f$.

Proof: Clearly, $T \in C_f \Rightarrow \exists q, f^q(TX) = F(X)$ and $T' \in C_f \Rightarrow \exists q', f^{q'}(T'X) = F(X)$ which result in $f^q(TX) = f^{q'}(T'X) \Rightarrow f(X) = f^{q \overline{\oplus} q'}(T'T^{-1}X) \Rightarrow T'T^{-1} \in S_f$. ■

Theorem 3: For function $f(X)$ and any CP transformation T, C_f is the right coset of S_f determined by T i.e., $T \in C_f \Rightarrow C_f = S_f T$.

Proof: First we prove that $T_1 \in S_f T \Rightarrow T_1 \in C_f$:

$$T_1 \in S_f T \Rightarrow \exists T' \in S_f, T_1 = T'T$$

$$T' \in S_f \Rightarrow \exists q', f(T'X) = f^{q'}(X) \Rightarrow f(T_1 X) = f(T'TX) = f^{q'}(TX)$$

$$T \in C_f \Rightarrow \exists q, f^q(TX) = F(X)$$

$$f^{q \overline{\oplus} q'}(T_1 X) = f^q(TX) = F(X) \Rightarrow T_1 \in C_f$$

Now we prove that $T_1 \in C_f \Rightarrow T_1 \in S_f T$. Based on the previous Lemma $T_1 \in C_f, T \in C_f \Rightarrow T_1 T^{-1} \in S_f \Rightarrow T_1 T^{-1} T \in S_f T \Rightarrow T_1 \in S_f T$; hence, $C_f = S_f T$. It can be easily verified that $C_f = S_f T \Rightarrow C_f T^{-1} = S_f T T^{-1} = S_f$; hence, for any $T \in C_f$, $S_f = C_f T^{-1}$ which shows that S_f can be easily obtained from C_f. ■

The set of SP transformations, S_f, includes transformations corresponding to simple symmetries. In the algorithm that we will present next to identify CP transformations C_f, first simple symmetries are identified since the computational complexity is relatively lower than that of general symmetries. This information is used efficiently to compute C_f. Next based on C_f the remaining SP transformations of S_f are computed.

V.1 The Proposed Canonical Form

In this part the canonical form used in this paper is formally defined. As mentioned earlier, among functions of an NPN-equivalence class the NPN-representative is selected based on a criterion that makes the representative unique among all functions in the class.

We previously defined the signature vector for a function and proved that it is unique for every function. We will define a total ordering for functions based on the lexicographical comparison of their signature vector.

Definition: Consider two functions $f(X)$ and $g(X)$ with signature vectors V^f and V^g, respectively. The order relation '$\prec$' between $f(X)$ and $g(X)$ is defined as: $V^f \prec V^g \Leftrightarrow f(X) \prec g(X)$. The relation $f(X) \succeq g(X)$ means that $f(X) \succ g(X)$ or $f(X) = g(X)$.

This ordering is well defined since we proved that $V^f = V^g \Leftrightarrow f(X) = g(X)$. An important aspect of the signature vector is that it enables us to compare functions even if they are not

functions of the same variable vector. The only requirement is the size of their variable vectors should be the same. Consider functions $f(X)$ and $g(Y)$ where $X=(x_1,x_2,\ldots,x_n)$ and $X=(y_1,y_2,\ldots,y_n)$, then f and g are equal (or equivalent) if their signature vectors are equal i.e., $f=g \Leftrightarrow V^f=V^g$. Also for the order relation '$\prec$', $f \prec g \Leftrightarrow V^f \prec V^g$. Using this order relation, the NPN-representative (canonical form) is defined as follows.

Definition: The NPN-representative of a class $\{f_1(X), f_2(X),\ldots, f_m(X)\}$ of functions is a function $F(X)$ which is maximal with respect to the order relation, '$\succ$' i.e., $\forall i \in \{1,2,\ldots,m\}, F(X) \succeq f_i(X)$.

V.2 Properties of the Canonical Form

In this section we will observe some important properties of the proposed canonical form that are used for the purpose of computing the canonical form.

Theorem 4: Let $F(X)$ be the canonical form of an NPN-equivalence class E. $F(X)$ is greater than or equal to that of its complement $\overline{F}(X)$ (which may also be denoted by $\overline{F(X)}$) i.e., $F(X) \succ \overline{F}(X)$.

Proof: Obviously $F(X)$ and $\overline{F}(X)$ are NPN-equivalent, Therefore, they belong to the same NPN-equivalence class i.e., $F(X) \in E, F(X) \equiv \overline{F}(X) \Rightarrow \overline{F}(X) \in E$.

Since $F(X)$ is the NPN-representative of class E, it is maximal i.e., $F(X) \succeq \overline{F}(X)$. Clearly, the equality in not possible; hence, $F(X) \succ \overline{F}(X)$. ■

Corollary: $|F(X)| \geq |\overline{F}(X)|$.

Proof: Clearly $F(X) \succ \overline{F}(X) \Rightarrow V^F \succ V^{\overline{F}}$ where V^F and $V^{\overline{F}}$ are the signature vectors of $F(X)$ and $\overline{F}(X)$ respectively and $|F(X)|$ is the zeroth signature and the first entry of signature vector V^F and $|\overline{F}(X)|$ is the first entry of $V^{\overline{F}}$. Since $V^F \succeq V^{\overline{F}}$ is based on lexicographic comparison, $V^F \succ V^{\overline{F}} \Rightarrow |F(X)| \geq |\overline{F}(X)|$. ■

Let F_{x_i} denote the cofactor of function $F(X)$ with respect to x_i where $X=(x_1,x_2,\ldots,x_n)$. (F_{x_i} is regarded as a function of $X_i=(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$.) Signatures in the signature vector $V^{F_{x_i}}$ of F_{x_i} are in the form of $|F_q|$ where cube q always include literal x_i. (We will use the

notation $x_i \in q$ to indicate that literal x_i is included in cube q.) Obviously, $x_i \in q \Leftrightarrow |F_q| \in V^{F_{x_i}}$. All signatures that exist in the signature vector of F_{x_i} will also exist in the signature vector of F i.e., $|F_q| \in V^{F_{x_i}} \Rightarrow |F_q| \in V^F$.

It is important to point out that the order in which signatures $|F_q|$ appear in $V^{F_{x_i}}$ is maintained in V^F. To state this fact formally let's denote the location of $|F_q|$ in V^F by $L(V^F, |F_q|)$. We have $L(V^F, |F_q|) \in \{1,2,3,\ldots,2^n\}$. For example $L(V^F, |F_{x_i}|) = i+1$ and $L(V^{F_{x_i}}, |F_{x_i}|) = 1$ since $|F_{x_i}|$ is the $i+1^{\text{st}}$ entry of V^F and the first entry of $V^{F_{x_i}}$.

Recall that in the previous chapter, when defining the signature vector, we defined an ordering '$\prec$' between cubes q of a given function $F(X)$. One can verify that this ordering has the following important property. For any two cubes q_1 and q_2 that contain literal x_i i.e., $q_1 = x_i q_1'$ and $q_2 = x_i q_2'$, we have $q_1' \prec q_2' \Leftrightarrow q_1 \prec q_2$. If the cubes are ordered based on the ordering '$\prec$', then $L(F, q)$ will be location of q in that ordering. From this argument it can be resulted that $q \prec q' \Leftrightarrow L(V^F, |F_q|) < L(V^F, |F_{q'}|)$. Therefore, if two signatures $|F_q|$ and $|F_{q'}|$ exist in both V^F and $V^{F_{x_i}}$ (i.e., $x_i \in q$ and $x_i \in q'$) then the order in which they appear in V^F and $V^{F_{x_i}}$ is the same i.e., if $|F_q|$ appears before $|F_{q'}|$ in $V^{F_{x_i}}$, then $|F_q|$ will appear before $|F_{q'}|$ also in $V^{F_{x_i}}$ and vice versa i.e., $L(V^{F_{x_i}}, |F_q|) < L(V^{F_{x_i}}, |F_{q'}|) \Leftrightarrow L(V^F, |F_q|) < L(V^F, |F_{q'}|)$.

Theorem 5: Let $F(X)$ be the canonical form of an NPN-equivalence class E. The cofactor of $F(X)$ with respect to the positive phase literals x_i is greater than or equal to that of negative literals $\bar{x}_i$ i.e., $i \in \{1,2,\ldots,n\}, F_{x_i} \succ F_{\bar{x}_i}$.

Proof: The proof is by contradiction. Assuming that $F_{x_i} \prec F_{\bar{x}_i}$ for some $i \in \{1,2,\ldots,n\}$, we prove that $F(X)$ cannot be the canonical form. We will show that if $F_{x_i} \prec F_{\bar{x}_i}$ then negating x_i will transform $F(X)$ to another function $F'(X)$ such that $F' \succ F$ which is clearly a contradiction. Consider the NP transformation:

$$T(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) = (x_1, x_2, \ldots, x_{i-1}, \bar{x}_i, x_{i+1}, \ldots, x_n).$$

We create a new function $F'(X) = F(TX)$. Clearly $F(X)$ and $F'(X)$ are NPN-equivalent, thus $F'(X) \in E$ since $F(X) \in E$. Previously, we defined the NP transformation on cubes. Using this definition, $F'(X) = F(TX) \Rightarrow |F'_{x_i q}| = |F_{T(x_i q)}|$. Based on the definition of transformation T, if a cube q does not include x_i i.e., $x_i \notin q$, then $T(q) = q$. Therefore, $T(x_i q) = T(x_i)T(q) = \bar{x}_i q \Rightarrow |F'_{x_i q}| = |F_{\bar{x}_i q}|$. We made the assumption that $F_{x_i} \prec F_{\bar{x}_i}$. The signatures of $V^{F_{\bar{x}_i}}$ are of the form $|F_{\bar{x}_i q}|$ where q is a positive cube not including x_i ($x_i \notin q$). Let q_1 be the first cube (not including x_i i.e., $x_i \notin q_1$) for signature vectors $V^{F_{x_i}}$ and $V^{F_{\bar{x}_i}}$ that $|F_{x_i q_1}| < |F_{\bar{x}_i q_1}|$. Based on the previous Theorem for signature vectors V^F and $V^{F'}$, $x_i q_1$ will also be the first cube (including x_i) such that $|F_{x_i q_1}| < |F'_{x_i q_1}|$ and for all cubes $x_i q$ before $x_i q_1$ i.e., $x_i q \prec x_i q_1$ (or equivalently $q \prec q_1$), $|F_{x_i q}| = |F'_{x_i q}|$. (Notice that since $F(X)$ and $F'(X)$ are defined over the same variable vector X for all cubes q, $L(V^F, |F_q|) = L(V^{F'}, |F_q|)$.) Signatures in $V^{F'}$ are of the form $|F'_{q'}|$. In addition, $F'(X) = F(TX) \Rightarrow |F'_{q'}| = |F_{T(q')}|$. If a cube q' does not include x_i then $T(q') = q'$; Thus $|F'_{q'}| = |F_{T(q')}| = |F_{q'}|$. Now if a signature $|F'_{q'}|$ from $V^{F'}$ is different from the corresponding signature $|F_{q'}|$ from V^F, then q' must include x_i i.e., $q' = x_i q$. Based on these arguments, for signature vectors V^F and $V^{F'}$, $x_i q_1$ will be the first cube among all cubes (including and not including x_i) that $|F_{x_i q_1}| < |F'_{x_i q_1}|$ (for all cubes q before $x_i q_1$ i.e., $q \prec x_i q_1$, $|F_q| = |F'_q|$.) This condition results in $F \prec F'$ which is a contradiction since $F(X)$ is the NPN-representative of its class E that also contains $F'(X)$ (the relation $F \succeq F'$ must be satisfied.) Hence the assumption $F_{x_i} \prec F_{\bar{x}_i}$ must be wrong which proves that $F_{x_i} \succeq F_{\bar{x}_i}$. However if $F_{x_i} = F_{\bar{x}_i}$ then $F(X)$ will be independent of x_i. Therefore, $F_{x_i} \succ F_{\bar{x}_i}$. ■

Corollary: $|F_{x_i}| \geq |F_{\bar{x}_i}|$.

Proof: Since $|F_{x_i}|$ and $|F_{\bar{x}_i}|$ are the first entries of the signature vectors of F_{x_i} and $F_{\bar{x}_i}$, $F_{x_i} \succ F_{\bar{x}_i} \Rightarrow |F_{x_i}| \geq |F_{\bar{x}_i}|$. ■

Let's use Q to denote the set all positive cubes created from conjunction of some literals among $x_1, x_2, \ldots, x_n$. Also let Q_i denote the set of cubes that do not include x_i i.e., $Q_i = \{q \in Q \big| x_i \notin q\}$. The set Q_{ij} is defined as the set of cubes that do not include x_i and x_j i.e., $Q_{ij} = \{q \in Q \big| x_i \notin q, x_j \notin q\}$. It is easy to see that $Q_{ij} = Q_i \bigcap Q_j$. Similarly Q_{ijk} is defined as $Q_{ijk} = Q_i \bigcap Q_j \bigcap Q_k$.

Since we include $|f|$ in the signature vector of function $f(X)$, we define the cube $q_0 = \{\}$ which contains no literals. Notice that $f_{q_0} = f$ and $\forall i, q_0 \in Q_i$.

In the following we study the symmetry of variables in the canonical form.

Theorem 6: Let $F(X)$ be the canonical form of an NPN-equivalence class E and q' represent a cube that does not include literals x_i and x_j ($q' \in Q_{ij}$). For every such cube, $q' \in Q_{ij}$, $|F_{x_i q'}|$ and $|F_{x_j q'}|$ are equal if and only if x_i and x_j are symmetric i.e., $(\forall q' \in Q_{ij}, |F_{x_i q'}| = |F_{x_j q'}|) \Leftrightarrow x_i \equiv x_j$.

Proof: We will prove that if signatures $|F_{x_i q}|$ and $|F_{x_j q}|$ are equal, then swapping x_i and x_j will not change the value of the signature vector of $F(X)$ and vice versa. Let's denote the NP transformation that swaps x_i and x_j by T i.e., $T(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) = (x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n)$ Furthermore, let's denote $F'(X) = F(TX)$. We will investigate the relation between signature vectors of functions $F(X)$ and $F'(X)$. The cubes q appearing in the signature vector of $F(X)$ (or $F'(X)$) can be classified into four types. For $q' \in Q_{ij}$, the following relations are satisfied: $T(q') = q'$, $T(x_i q') = x_j q'$, $T(x_j q') = x_i q'$ and $T(x_i x_j q') = x_i x_j q'$. As we can see, cubes of the form q' and $x_i x_j q'$ are invariant under NP transformation T. Recall that for a cube q, $F'(X) = F(TX) \Rightarrow |F'_q| = |F_{T(q)}|$; therefore, the signatures of the signature vectors V^F and $V^{F'}$ that correspond to cubes of the form q' and $x_i x_j q'$ are always equal. The signature vectors V^F and $V^{F'}$ are equal if and only if $|F_q| = |F'_q|$ for every cube q; therefore, $|F_{x_i q'}| = |F'_{x_i q'}| \,\&\, |F_{x_j q'}| = |F'_{x_j q'}| \Leftrightarrow V^F = V^{F'}$. However, from $|F'_q| = |F_{T(q)}|$, it can be concluded that $|F'_{x_i q'}| = |F_{x_j q'}|$ and $|F'_{x_j q'}| = |F_{x_i q'}|$, which combined with the previous relation, results in

$F = F' \Leftrightarrow |F_{x_i q'}| = |F_{x_j q'}|$. Since $F'(X) = F(TX)$ and T only swaps x_i and x_j, $F = F'$ means that x_i and x_j are symmetric. Therefore, $|F_{x_i q'}| = |F_{x_j q'}| \Leftrightarrow x_i \equiv x_j$. ■

Theorem 7: Let $F(X)$ be the canonical form of an NPN-equivalence class E. Assume that x_i and x_j (with $i < j$) are not symmetric. There exists a cube $q_1 \in Q_{ij}$ such that $|F_{x_i q_1}| > |F_{x_j q_1}|$ and for every cube $q \in Q_{ij}$ before q_1 (i.e., $q \prec q_1$), $|F_{x_i q}|$ and $|F_{x_j q}|$ are equal i.e., $\exists q_1 \in Q_{ij}, |F_{x_i q_1}| > |F_{x_j q_1}|$ and $\forall q \in Q_{ij}, q \prec q_1 \Rightarrow |F_{x_i q}| = |F_{x_j q}|$.

Proof: Let $q_1 \in Q_{ij}$ be the first cube in Q_{ij} that $|F_{x_i q_1'}| \neq |F_{x_j q_1'}|$. From the previous Theorem since x_i and x_j are not symmetric, such a cube exists. We will prove that $|F_{x_i q_1}| > |F_{x_j q_1}|$. The proof is by contradiction. We will prove that if $|F_{x_i q_1}| < |F_{x_j q_1}|$, then swapping x_i and x_j in $F(X)$ will result in another function $F'(X) \in E$ with $F' \succ F$ which is a contradiction since $F(X)$ is the NPN-representative of E. Let's denote the NP transformation that swaps x_i and x_j by $T(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n) = (x_1,\ldots,x_j,\ldots,x_i,\ldots,x_n)$. Also let's denote $F'(X) = F(TX)$. Now we compare signatures $|F_q|$ and $|F'_q|$ of signature vectors V^F and $V^{F'}$. Assuming that q' generally denotes a cube not including x_i and x_j i.e., $q' \in Q_{ij}$, signatures $|F_q|$ and $|F'_q|$ are equal for cubes q of the form of $q = q'$ or $q = x_i x_j q'$ i.e., $|F'_{q'}| = |F_{T(q')}| = |F_{q'}|$ and $|F'_{x_i x_j q'}| = |F_{T(x_i x_j q')}| = |F_{x_i x_j q'}|$. Hence, V^F and $V^{F'}$ may only be different in signatures with cubes of the form $q = x_i q'$ or $q = x_j q'$. Notice that $|F'_{x_i q'}| = |F_{T(x_i q')}| = |F_{x_j q'}|$ and $|F'_{x_j q'}| = |F_{T(x_j q')}| = |F_{x_i q'}|$. Hence, $|F_{x_i q_1}| < |F_{x_j q_1}| \Rightarrow |F_{x_i q_1}| < |F'_{x_i q_1}|$ and $\forall q \in Q_{ij}, q \prec q_1, |F_{x_i q}| = |F_{x_j q}| \Rightarrow |F_{x_i q}| = |F'_{x_i q}|$ which proves that $V^{F'} \succ V^F$. Since $V^{F'} \succ V^F \Rightarrow F' \succ F$ results in a contradiction; Thus, $|F_{x_i q_1}| > |F_{x_j q_1}|$ must be correct. ■

Corollary: If for the canonical form $F(X)$, x_i and x_j $(i < j)$ are not symmetric, then $|F_{x_i}| \geq |F_{x_j}|$.

Proof: Since $q_0 \in \{\}$ is the first cube in $Q_{ij}, |F_{x_i q_0}| \geq |F_{x_j q_0}|$ In addition, since $F_{x_i q_0} = F_{x_i}$ and $F_{x_j q_0} = F_{x_j}$, it can be seen that $|F_{x_i}| \geq |F_{x_j}|$. ■

Corollary: For the case that $|F_{x_i}| = |F_{x_j}|$, let $k \notin \{i, j\}$ be the first number that $|F_{x_i x_k}| \neq |F_{x_j x_k}|$; if such k exists, $|F_{x_i x_k}| > |F_{x_j x_k}|$.

Proof: Notice that cubes of the form $q_k = x_k$ where $k \notin \{i, j\}$ belong to Q_{ij}. For $k, l \notin \{i, j\}$, $k < l \Rightarrow q_k \prec q_l$. ■

Another way of describing this result is, $V^F(i, j) \succ V^F(j, i)$ where $V^F(i, j)$ is a partial signature vector defined as,

$$V^F(i, j) = (|F_{x_i}|, |F_{x_i x_1}|, |F_{x_i x_2}|, \ldots, |F_{x_i x_{i-1}}|, |F_{x_i x_{i+1}}|, \ldots, |F_{x_i x_{j-1}}|, |F_{x_i x_{j+1}}|, \ldots, |F_{x_i x_n}|)$$

Similarly $V^F(j, i)$ is defined as,

$$V^F(j, i) = (|F_{x_j}|, |F_{x_j x_1}|, |F_{x_j x_2}|, \ldots, |F_{x_j x_{i-1}}|, |F_{x_j x_{i+1}}|, \ldots, |F_{x_j x_{j-1}}|, |F_{x_j x_{j+1}}|, \ldots, |F_{x_j x_n}|)$$

Theorem 8: Let $F(X)$ be the canonical form of an NPN-equivalence class E. If for $i < j$, x_i and x_j are symmetric in function $F(X)$, then all other x_k where $i < k < j$ are also symmetric to x_i and x_j i.e., $x_i \equiv x_j \Rightarrow x_i \equiv x_{i+1} \equiv \cdots \equiv x_j$.

Proof: The proof is by contradiction. Assume that there exist x_k (where $i < k < j$) that is not symmetric to x_i and x_j. Let's denote the NP transformation that swaps x_i and x_j by $T(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) = (x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n)$. Since x_i and x_j are symmetric, $F(TX) = F(X)$. Let $q_1 \in Q_{ik}$ be the first cube in Q_{ik} such that $|F_{x_i q_1}| \neq |F_{x_k q_1}|$. Based on the previous Theorem, since $i < k$, $|F_{x_i q_1}| > |F_{x_k q_1}|$ and $\forall q \in Q_{ik}, q \prec q_1 \Rightarrow |F_{x_i q_1}| = |F_{x_k q_1}|$. In addition, let $q_2 \in Q_{ik}$ be the first cube in Q_{kj} that $|F_{x_j q_2}| \neq |F_{x_k q_2}|$. Based on the previous Theorem, since $k < j$, $|F_{x_k q_2}| > |F_{x_j q_2}|$ and $\forall q \in Q_{kj}, q \prec q_2 \Rightarrow |F_{x_k q_2}| = |F_{x_j q_2}|$. Cube $q_1 \in Q_{ik}$ does not include x_i and x_k. However, it may or may not include x_j. Similarly cube $q_2 \in Q_{kj}$ may or may not include x_i. Hence four possibilities exist. We will investigate all cases. First let's assume that $x_j \notin q_1$ and $x_i \notin q_2$ i.e., $q_1, q_2 \in Q_{ijk}$ which results in $T(q_1) = q_1$ and $T(q_2) = q_2$ since $q_1, q_2 \in Q_{ij}$. Now since $F(TX) = F(X)$, $|F_{x_i q_1}| = |F_{T(x_i q_1)}| = |F_{x_j q_1}|$. From the definition of q_1, $|F_{x_i q_1}| > |F_{x_k q_1}| \Rightarrow |F_{x_j q_1}| > |F_{x_k q_1}|$. Therefore, based on the definition of q_2 and since $q_1 \in Q_{jk}$, q_1

should satisfy the relation $q_2 \prec q_1$ because if $q_1 \prec q_2$, then $|F_{x_j q_1}|$ and $|F_{x_k q_1}|$ must be equal. Therefore, again based on the definition of q_1, $q_2 \prec q_1, q_2 \in Q_{ik} \Rightarrow |F_{x_i q_2}| = |F_{x_k q_2}|$. However, $T(q_2) = q_2$ since $q_2 \in Q_{ij}$, and therefore, $|F_{x_i q_2}| = |F_{T(x_i q_2)}| = |F_{x_j q_2}|$ which combined with the previous relation results in $|F_{x_j q_2}| = |F_{x_k q_2}|$. This is clearly a contradiction since based on the definition of q_2, $|F_{x_k q_2}| > |F_{x_j q_2}|$. Hence our last assumption (i.e., $x_j \notin q_1$ and $x_i \notin q_2$) must be incorrect. Next we investigate the case $x_j \in q_1$ and $x_i \notin q_2$. Clearly, $x_j \in q_1 \Rightarrow \exists q_1' \in Q_{ijk}, q_1 = x_j q_1'$. From the definition of q_1, $|F_{x_i q_1}| > |F_{x_k q_1}| \Rightarrow |F_{x_i x_j q_1'}| > |F_{x_k x_j q_1'}|$. Since $q_1' \in Q_{ij}$, $T(q_1') = q_1'$, $|F_{x_k x_j q_1'}| = |F_{T(x_k x_j q_1')}| = |F_{x_k x_i q_1'}|$ which combined with the previous relation results in $|F_{x_j x_i q_1'}| > |F_{x_k x_i q_1'}|$. Therefore, based on the definition of q_2 and since $x_i q_1' \in Q_{jk}$, $x_i q_1'$ should satisfy the relation $q_2 \prec x_i q_1'$. Since $i \prec j$, $x_i q_1' \prec x_j q_1'$ which combined with $q_2 \prec x_i q_1'$ results in $q_2 \prec x_j q_1'$. Notice that $x_j q_1' = q_1$; hence, $q_2 \prec q_1$. Therefore, again based on the definition of q_1, $q_2 \prec q_1, q_2 \in Q_{ik} \Rightarrow |F_{x_i q_2}| = |F_{x_k q_2}|$. However $T(q_2) = q_2$ since $q_2 \in Q_{ij}$, and therefore, $|F_{x_i q_2}| = |F_{T(x_i q_2)}| = |F_{x_j q_2}|$ which combined with the previous relation results in $|F_{x_j q_2}| = |F_{x_k q_2}|$. This is a contradiction since based on the definition of q_2, $|F_{x_k q_2}| > |F_{x_j q_2}|$. Hence the assumption (i.e., $x_j \in q_1$ and $x_i \notin q_2$) is also incorrect. Similarly one can prove that $x_j \notin q_1$ and $x_i \in q_2$ leads to contradiction. Hence the only remaining option is $x_j \in q_1$ and $x_i \in q_2$ which in the following we will show that results in a contradiction. Clearly, $x_j \in q_1 \Rightarrow \exists q_1' \in Q_{ijk}, q_1 = x_j q_1'$ and $x_i \in q_2 \Rightarrow \exists q_2' \in Q_{ijk}, q_2 = x_i q_2'$. From the definition of q_1, $|F_{x_i q_1}| > |F_{x_k q_1}| \Rightarrow |F_{x_i x_j q_1'}| > |F_{x_k x_j q_1'}|$. Since $q_1' \in Q_{ij}$, $T(q_1') = q_1'$ and therefore $|F_{x_k x_j q_1'}| = |F_{T(x_k x_j q_1')}| = |F_{x_k x_i q_1'}|$ which combined with the previous relation results in $|F_{x_j x_i q_1'}| > |F_{x_k x_i q_1'}|$. Therefore, based on the definition of q_2 and since $x_i q_1' \in Q_{jk}$, $x_i q_1'$ should satisfy the relation $x_i q_1' \succ q_2$ because if $x_i q_1' \prec q_2$ then $|F_{x_j x_i q_1'}|$ and $|F_{x_k x_i q_1'}|$ must be equal. Therefore, since $q_2 = x_i q_2'$, $x_i q_1' \succ q_2 \Rightarrow x_i q_1' \succ x_i q_2'$. Based on one of important properties of the order relation that we defined between cubes, $x_i q_1' \succ x_i q_2' \Rightarrow q_1' \succ q_2' \Rightarrow x_j q_1' \succ x_j q_2'$ which

combined with $q_1 = x_j q_1'$ results in $q_1 \succ x_j q_2'$. Therefore, again based on the definition of q_1, $q_1 \succ x_j q_2', x_j q_2' \in Q_{ik} \Rightarrow |F_{x_i x_j q_2'}| = |F_{x_k x_j q_2'}|$. However, $|F_{x_i x_j q_2'}| = |F_{x_j q_2}|$; also $T(q_2') = q_2'$ since $q_2' \in Q_{ij}$. Therefore, $|F_{x_k x_j q_2'}| = |F_{T(x_k x_j q_2')}| = |F_{x_k x_i q_2'}| = |F_{x_k q_2}|$ which combined with the previous relation results in $|F_{x_j q_2}| = |F_{x_k q_2}|$. This is clearly a contradiction since based on the definition of q_2, $|F_{x_k q_2}| > |F_{x_j q_2}|$. Hence, the statement that "there exist x_k where $i < k < j$ that is not symmetric to x_i and x_j" is incorrect, which proves that $x_i \equiv x_{i+1} \equiv \cdots \equiv x_j$. ■

This Theorem indicates that for the canonical form $F(X)$, symmetric variables are positioned consecutively in $X = (x_1, x_2, \ldots, x_n)$ i.e., variables will be arranged as: $\overbrace{x_1, x_2, \cdots, x_{n_1}}^{C_1}, \overbrace{x_{n_1+1}, x_{n_1+2}, \cdots, x_{n_1+n_2}}^{C_2}, \cdots, \overbrace{x_{n-n_k+1}, x_{n-n_k+2}, \cdots, x_n}^{C_k}$ where $C_1, C_2, \ldots, C_k$ are symmetry classes and $n_i = |C_i|$.

VI Computing the canonical form

Given a function $f(X)$ the objective is to find its canonical form $F(X)$ and the corresponding set of CP transformations $C_f = \{T \in \Gamma_n \,|\, \exists q, f^q(TX) = F(X)\}$. Equivalently the set of CP transformations can be expressed as: $C_f = \{T \in \Gamma_n \,|\, \forall T' \in \Gamma_n, \exists q, f^q(TX) = f(T'X)\}$.

Previous Theorems impose some conditions on the canonical form $F(X)$. For a CP transformation T' such that $\exists_q f^q(T'X) = F(X)$, we project the conditions on $F(X)$ to conditions on T'. These conditions significantly limit the search space (for C_f) since $f^q(T'X) = F(X) \Rightarrow f^q(X) = F(T'^{-1}X)$.

Let's denote the inverse of T' by $T_\pi^P = T'^{-1}$ where $P = (p_1, p_2, \ldots, p_n)$. Let's also denote the variable after phase assignment by $\widetilde{X} = X^P$ (i.e. $\widetilde{x}_i = x_i^{p_i}$) for simplicity. With this notation, $T_\pi^P X = \pi(X^P) = \pi(\widetilde{X}) = (\widetilde{x}_{\pi(1)}, \widetilde{x}_{\pi(2)}, \ldots, \widetilde{x}_{\pi(n)})$. Based on the relation $f^q(X) = F(T_\pi^P X)$ the conditions of previous Theorems are translated as follows.

The first condition is on the output phase assignment q. Since $F \succ \overline{F}$, phase assignment q should satisfy the condition $f^q \succ f^{\bar{q}}$. Again for simplicity we denote $\widetilde{f} = f^q$. The second

condition is on phase assignment P which follows from $F_{x_i} \succ F_{\bar{x}_i}$ and translates to $\tilde{f}_{\tilde{x}_i} \succ \tilde{f}_{\bar{\tilde{x}}_i}$ (or $\tilde{f}_{x_i^{p_i}} \succ \tilde{f}_{x_i^{\bar{p}_i}}$). The next condition is as follows. If for $i<j$, $\tilde{x}_{\pi(i)}$ and $\tilde{x}_{\pi(j)}$ are symmetric in function $f(X)$ (or $\tilde{f}(X)$), then all other $\tilde{x}_{\pi(k)}$ where $i<k<j$ are also symmetric to $x_{\pi(i)}$ and $x_{\pi(j)}$ i.e., $\tilde{x}_{\pi(i)} \equiv \tilde{x}_{\pi(j)} \Rightarrow \tilde{x}_{\pi(i)} \equiv \tilde{x}_{\pi(i+1)} \equiv \cdots \equiv \tilde{x}_{\pi(j)}$.

Another important constraint is that if $\tilde{x}_{\pi(i)}$ and $\tilde{x}_{\pi(j)}$ are symmetric in function $f(X)$ (or $\tilde{f}(X)$), then $V_\pi^{\tilde{f}}(i,j) \succ V_\pi^{\tilde{f}}(j,i)$ where $V_\pi^{\tilde{f}}(i,j)$ is a partial signature vector defined as

$$V_\pi^{\tilde{f}}(i,j) = (|\tilde{f}_{\tilde{x}_{\pi(i)}}|, |\tilde{f}_{\tilde{x}_{\pi(i)}\tilde{x}_{\pi(1)}}|, \ldots, |\tilde{f}_{\tilde{x}_{\pi(i)}\tilde{x}_{\pi(i-1)}}|, |\tilde{f}_{\tilde{x}_{\pi(i)}\tilde{x}_{\pi(i+1)}}|, \ldots, |\tilde{f}_{\tilde{x}_{\pi(i)}\tilde{x}_{\pi(j-1)}}|, |\tilde{f}_{\tilde{x}_{\pi(i)}\tilde{x}_{\pi(j+1)}}|, \ldots, |\tilde{f}_{\tilde{x}_{\pi(i)}\tilde{x}_{\pi(n)}}|) \quad ,$$

$$V_\pi^{\tilde{f}}(j,i) = (|\tilde{f}_{\tilde{x}_{\pi(j)}}|, |\tilde{f}_{\tilde{x}_{\pi(j)}\tilde{x}_{\pi(1)}}|, \ldots, |\tilde{f}_{\tilde{x}_{\pi(j)}\tilde{x}_{\pi(i-1)}}|, |\tilde{f}_{\tilde{x}_{\pi(j)}\tilde{x}_{\pi(i+1)}}|, \ldots, |\tilde{f}_{\tilde{x}_{\pi(j)}\tilde{x}_{\pi(j-1)}}|, |\tilde{f}_{\tilde{x}_{\pi(j)}\tilde{x}_{\pi(j+1)}}|, \ldots, |\tilde{f}_{\tilde{x}_{\pi(j)}\tilde{x}_{\pi(n)}}|)$$

Symmetry classes of variables will be arranged as:

$$\overbrace{\tilde{x}_{\pi(1)}, \cdots, \tilde{x}_{\pi(n_1)}}^{C_1}, \overbrace{\tilde{x}_{\pi(n_1+1)}, \cdots, \tilde{x}_{\pi(n_1+n_2)}}^{C_2}, \cdots, \overbrace{\tilde{x}_{\pi(n-n_k+1)}, \cdots, \tilde{x}_{\pi(n)}}^{C_k}$$

where C_i with n_i members is a maximal symmetry class for function $\tilde{f}(X)$. The first order signatures $|\tilde{f}_{\tilde{x}_{\pi(i)}}|$ are sorted non-increasingly.

$$|\tilde{f}_{\tilde{x}_{\pi(1)}}| = \cdots = |\tilde{f}_{\tilde{x}_{\pi(n_1)}}| > |\tilde{f}_{\tilde{x}_{\pi(n_1+1)}}| = \cdots = |\tilde{f}_{\tilde{x}_{\pi(n_1+n_2)}}| > \cdots > |\tilde{f}_{\tilde{x}_{\pi(n-n_k+1)}}| = \cdots = |\tilde{f}_{\tilde{x}_{\pi(n)}}|$$

One can use this property to reduce the complexity of identifying symmetry classes since a necessary condition for symmetry of $\tilde{x}_{\pi(i)}$ and $\tilde{x}_{\pi(j)}$ is the equality of $|\tilde{f}_{\tilde{x}_{\pi(i)}}|$ and $|\tilde{f}_{\tilde{x}_{\pi(j)}}|$ i.e., $\tilde{x}_{\pi(i)} \equiv \tilde{x}_{\pi(j)} \Rightarrow |\tilde{f}_{\tilde{x}_{\pi(i)}}| = |\tilde{f}_{\tilde{x}_{\pi(j)}}|$.

These conditions on $T_\pi^P = T'^{-1}$ are necessary conditions for $(T_\pi^P)^{-1} \in C_f$. Hence, if we define inverse of C_f as $C_f^{-1} = \{T \in \Gamma_n \,|\, T^{-1} \in C_f\}$, then these are necessary conditions for $T_\pi^P \in C_f^{-1}$. An important property of the CP transformation set C_f is as follows. If for two NP transformations T and T' satisfy the inequality $\tilde{f}(TX) \succ \tilde{f}(T'X)$, then T' is definitely not a CP transformation i.e., $\tilde{f}(TX) \succ \tilde{f}(T'X) \Rightarrow T' \notin C_f$.

These constraints are effectively used to identify CP transformations C_f as described in the following.

VI.1 The *Compute_C_f* Algorithm

The proposed algorithm, called *Compute_C_f*, uses signatures of function $f(X)$ to compute the CP transformations on inputs and corresponding output phase assignments. However, at the beginning, only 0^{th}, 1^{st} and if necessary 2^{nd} signatures are used. In most cases, 1^{st}-signatures alone determine a considerable portion of the inequalities required to identify the desired NP transformation. Otherwise, as will be explained later, the remaining comparisons are performed using 2^{nd}-signatures. Similarly higher order signatures are used only if they become necessary.

Experimental evidence shows that in the great majority of cases, a signature inequality occurs for the low order signatures (0^{th}, 1^{st} and 2^{nd} signatures.) Intuitively, the reason is that the lower order signatures depend on higher number of minterms of the function, and thus, contain more information about the function. For example, a 1^{st}-signature depends on 2^{n-1} minterms, which is one half of the whole Boolean space (2^n minterms) whereas a 2^{nd}-signature depends on one-forth of all minterms (2^{n-2} minterms.) Hence, the 1^{st}-signatures are the most powerful and effective signatures. The 2^{nd}-signatures are the next most effective signatures and so on. The reader will observe that this arrangement of the proposed signature vector minimizes the computational complexity.

The first step of the *Compute_C_f* algorithm is to identify the output phase assignment. If $|f| \neq |\bar{f}|$, then the output phase q can be uniquely determined from the relation $|f^q| > |f^{\bar{q}}|$. However, if $|f| = |\bar{f}|$, the output phase cannot be determined by using only 0^{th} signatures. In this case ($|f| = |\bar{f}|$) the output phase is considered *undecided*, which is denoted by $q = u$. (Thus we have extended the set of acceptable phases $\{0,1\}$ to $\{0,1,u\}$.) If the output phase is undecided, it will be determined in subsequent steps of the algorithm. In the end, the algorithm will return a value in $\{0,1\}$ for the phase. For simplicity we will denote $\tilde{f} = f^q$.

In the next step, input phase assignment is performed by using 1^{st}-signatures. First assume that the output phase is decided i.e., $q \neq u$. The opposite case will be discussed later. For variable x_i if $|\tilde{f}_{x_i}| \neq |\tilde{f}_{\bar{x}_i}|$, then p_i, the phase of x_i, can be uniquely determined by using the relation $|\tilde{f}_{x_i^{p_i}}| > |\tilde{f}_{x_i^{\bar{p}_i}}|$. However, if $|\tilde{f}_{x_i}| = |\tilde{f}_{\bar{x}_i}|$, p_i cannot be determined by using only 1^{st}-signatures in which case the phase of x_i is considered *undecided* (which is again denoted by $p_i = u$.)

Undecided input phases will be determined in subsequent steps of the algorithm. If the output phase is undecided i.e., $q = u$, then 1st-signatures are used as follows to determine output and input phases. The are two possibilities for the final value of q i.e., $q = 0$ or $q = 1$. Each possibility for q results in input phase assignments $p_i(q)$ (where q in the parenthesis indicates the dependence of p_i on q) based on the following relation, $| f^{q}_{x_i^{p_i(q)}} | > | f^{q}_{\overline{x_i^{p_i(q)}}} |$ and $| f^{q}_{x_i} | = | f^{q}_{\overline{x}_i} | \Rightarrow p_i(q) = u$. Notice that even if $p_i(q) = u$, the value of $| f^{q}_{x_i^{p_i(q)}} |$ will be unique i.e., $p_i(q) = u \Rightarrow | f^{q}_{x_i^{p_i(q)}} | = | f^{q}_{x_i} | = | f^{q}_{\overline{x}_i} |$

Next we sort signatures $| f^{q}_{x_i^{p_i(q)}} |$ in a non-increasing order. Let π_q be a permutation that results in such an ordering i.e., $| f^{q}_{\tilde{x}_{\pi_q(1)}} | \geq | f^{q}_{\tilde{x}_{\pi_q(2)}} | \geq \cdots \geq | f^{q}_{\tilde{x}_{\pi_q(n)}} |$ where $\tilde{x}_i = x_i^{p_i(q)}$ for simplicity. Let $V(q)$ denote the vector comprising of the ordered 1st-signatures i.e., $V(q) = (| f^{q}_{\tilde{x}_{\pi_q(1)}} |, | f^{q}_{\tilde{x}_{\pi_q(2)}} |, \ldots, | f^{q}_{\tilde{x}_{\pi_q(n)}} |)$. Assuming that q is the output phase assignment that results in the canonical form, one can easily see that $V(q)$ consists of 2nd to $(n+1)^{st}$ entries of the signature vector of the canonical form of $f(X)$. (The first entry is $| f | = | \bar{f} |$.) Now if $V(q) \neq V(\overline{q})$, then q must satisfy the following relation: $V(q) \succ V(\overline{q})$ since the canonical form has the maximal signature vector. This relation uniquely determines the output phase assignment q unless $V(q) = V(\overline{q})$ in which case the computation of q is again postponed to the subsequent steps of the algorithm.

Let's denote a variable after phase assignment as $\tilde{x}_j = x_j^{p_j}$. In the next step, symmetry classes of variables are determined. A necessary condition for $\tilde{x}_i \equiv \tilde{x}_j$ is $| \tilde{f}_{\tilde{x}_i} | = | \tilde{f}_{\tilde{x}_j} |$; therefore the symmetry check is performed for $\tilde{x}_i$ and $\tilde{x}_j$ only if $| \tilde{f}_{\tilde{x}_i} | = | \tilde{f}_{\tilde{x}_j} |$. If the phases of variables x_i and x_j are undecided, we will determine x_i and x_j to be symmetric only if they are symmetric independent of their phases i.e. $x_i \equiv x_j$ and $x_i \equiv \overline{x}_j$. An example of this situation is when $\tilde{f}(X)$ depends on $x_i \oplus x_j$.

Based on these symmetry relations, we form the maximal symmetry classes of variables $C_1, C_2, \ldots, C_m$. Function $\tilde{f}(X)$ will remain invariant under permutations inside a symmetry

class. Based on this fact and since symmetric variables are positioned consecutively, instead of finding NP transformations on the variables, it is sufficient to search for NP transformations on classes $C_1, C_2, ..., C_m$, which will in turn greatly reduce the size of search space. In fact, this is a major advantage of the proposed technique compared to previous approaches.

The objective is to determine all members of C_f. Obviously, if an NP transformation T is a member of C_f and $T' \in S_f$, then the transformation TT' is also a member of C_f.

(Recall that S_f is the set of SP transformations i.e., $T' \in S_f \Leftrightarrow \exists_q f(X) = f^q(T'X)$.)

Let's denote the subgroup of SP transformations that correspond to simple symmetry of variables by W. An example of a member of W is the NP transformation that swaps variables x_i and x_j. Notice that W also includes the cascades of NP transformations that correspond to simple symmetry of variables i.e., W is closed with respect to the cascade operation: $T \in W, T' \in W \Rightarrow TT' \in W$. Obviously $W \subset S_f$ and also $T \in C_f \Rightarrow TW \subset C_f$. Thus, to avoid computational redundancy, the proposed algorithm will return only one member of TW since given an NP transformation $T \in C_f$, one can easily obtain all members of TW.

To explain this matter formally, we define a relation '$\approx$' between $T, T' \in C_f$ where $T \approx T'$ means that there exist a $T'' \in W$ such that $T = T'T''$. This relation breaks C_f into equivalence classes. The proposed algorithm returns the set $C'_f \subset C_f$ which contains exactly one member of each such class i.e., $T \in C'_f, T' \in W \Rightarrow TT' \notin C'_f$ or equivalently, $T \in C'_f, T \approx T' \Rightarrow T' \notin C'_f$.

Now we will discuss the concept of NP transformations on classes. The phases of classes that contain variables with decided phases are defined as 1. The phase assignment for classes that contain variables with undecided phases is defined as: $C_i^{p_i} = \{x_j^{p_i} \mid x_j \in C_i\}$ where p_i is to be determined.

An NP transformation on classes $C_1, C_2, ..., C_m$ signifies a transformation on variables $X = (x_1, x_2, \ldots, x_n)$. The cofactor of function $f(X)$ with respect to any member of a class C_i is a unique function; hence the cofactor of $f(X)$ with respect to the class C_i can be defined as $f_{C_i} = f_{x_j}$ for $x_j \in C_i$.

Let's denote the classes after phase assignment as $\widetilde{C}_1, \widetilde{C}_2, ..., \widetilde{C}_m$. In the next step, classes are ordered based on their first signatures. Let π be a permutation on classes that respects the ordering: $|\widetilde{f}_{\widetilde{C}_{\pi(1)}}| \geq |\widetilde{f}_{\widetilde{C}_{\pi(2)}}| \geq \cdots \geq |\widetilde{f}_{\widetilde{C}_{\pi(m)}}|$ (even when the phase of a class, $\widetilde{C}_i$, is undecided, the 1st-signature can be defined since $|\widetilde{f}_{C_i}| = |\widetilde{f}_{\overline{C}_i}|$.)

If the 1st-signatures are distinct values for $\widetilde{C}_1, \widetilde{C}_2, ..., \widetilde{C}_m$, then a unique ordering can be achieved since $|\widetilde{f}_{\widetilde{C}_{\pi(1)}}| > |\widetilde{f}_{\widetilde{C}_{\pi(2)}}| > \cdots > |\widetilde{f}_{\widetilde{C}_{\pi(m)}}|$ in which case the algorithm terminates returning the NP transformation resulting from π and the corresponding phase assignment as a member of $C_f'^{-1} = \{T \in \Gamma_n \,|\, T^{-1} \in C_f'\}$. Otherwise, the classes are placed in *k* groups such that all classes inside a group have the same 1st-signature: $\overbrace{\widetilde{C}_{\pi(1)}, \cdots, \widetilde{C}_{\pi(n_1)}}^{G_1}, \overbrace{\widetilde{C}_{\pi(n_1+1)}, \cdots, \widetilde{C}_{\pi(n_1+n_2)}}^{G_2}, \cdots, \overbrace{\widetilde{C}_{\pi(n-n_k+1)}, \cdots, \widetilde{C}_{\pi(m)}}^{G_k}$

where $|\widetilde{f}_{\widetilde{C}_{\pi(1)}}| = \cdots = |\widetilde{f}_{\widetilde{C}_{\pi(n_1)}}| \succ |\widetilde{f}_{\widetilde{C}_{\pi(n_1+1)}}| = \cdots = |\widetilde{f}_{\widetilde{C}_{\pi(n_1+n_2)}}| \succ \cdots \succ |\widetilde{f}_{\widetilde{C}_{\pi(n-n_k+1)}}| = \cdots = |\widetilde{f}_{\widetilde{C}_{\pi(m)}}|$.

We refer to a group as *unresolved* if it contains more than one class or the phases of classes in that group are undecided. If all the groups are *resolved*, a unique ordering has been obtained and the algorithm terminates. The objective of next steps is to resolve all unresolved groups.

Let $G_j = \{\widetilde{C}_{\pi(j)}, \widetilde{C}_{\pi(j+1)}, \ldots, \widetilde{C}_{\pi(l)}\}$ be the first unresolved group. Since all groups $G_1, G_2, \ldots, G_{j-1}$ are resolved, (i.e., they contain a single class with decided phase,) the ordering of classes up to G_j is identified. (The case that G_1 is unresolved is discussed later.)

Now the 2nd-signatures are used to specify the ordering inside the unresolved groups starting with G_j. Since G_1 is resolved, $G_1 = \{\widetilde{C}_{\pi(1)}\}$, 2nd-signatures with respect to $\widetilde{C}_{\pi(1)}$ and $\widetilde{C}_{\pi(i)}$ for $j \leq i \leq l$ (i.e., $|\widetilde{f}_{\widetilde{C}_{\pi(1)}\widetilde{C}_{\pi(i)}}|$) can be used for phase assignment (if needed) and ordering classes $\widetilde{C}_{\pi(j)}, \widetilde{C}_{\pi(j+1)}, \ldots, \widetilde{C}_{\pi(l)}$ (later on this step will be referred to as iteration 1.)

If phases of classes $C_{\pi(i)}$ in G_j is undecided, the 2nd-signatures $|\widetilde{f}_{\widetilde{C}_{\pi(1)}C_{\pi(i)}}|$ and $|\widetilde{f}_{\widetilde{C}_{\pi(1)}\bar{C}_{\pi(i)}}|$ are compared and phase p for $C_{\pi(i)}$ is decided based on $|\widetilde{f}_{\widetilde{C}_{\pi(1)}C^{p}_{\pi(i)}}| > |\widetilde{f}_{\widetilde{C}_{\pi(1)}C^{\bar{p}}_{\pi(i)}}|$. In case of equality of the 2nd-signatures, the phase of $C_{\pi(i)}$ remains undecided.

Next, new values of 2nd-signatures $|\widetilde{f}_{\widetilde{C}_{\pi(1)}\widetilde{C}_{\pi(i)}}|$ after phase assignment are used to order classes $\widetilde{C}_{\pi(j)},\widetilde{C}_{\pi(j+1)},\ldots,\widetilde{C}_{\pi(l)}$ and subsequently regroup these classes i.e., the values of $\pi(j),\pi(j+1),\ldots,\pi(l)$ are updated to respect the following ordering: $|\widetilde{f}_{\widetilde{C}_{\pi(1)}\widetilde{C}_{\pi(j)}}| \geq |\widetilde{f}_{\widetilde{C}_{\pi(1)}\widetilde{C}_{\pi(j+1)}}| \geq \cdots \geq |\widetilde{f}_{\widetilde{C}_{\pi(1)}\widetilde{C}_{\pi(l)}}|$. Subsequently, G_j is split into smaller groups such that inside each group the 2nd-signatures, $|\widetilde{f}_{\widetilde{C}_{\pi(1)}\widetilde{C}_{\pi(i)}}|$, are equal. The same procedure (phase assignment, ordering and regrouping based on $|\widetilde{f}_{\widetilde{C}_{\pi(1)}\widetilde{C}_{\pi(i)}}|$) is applied to all other unresolved groups.

Finally, the indices of new groups are properly updated. If after these steps, there still exists some unresolved group, G_l, a similar procedure (called iteration 2) is applied based on 2nd-signatures with respect to $\widetilde{C}_{\pi(2)}$ and $\widetilde{C}_{\pi(i)} \in G_l$ (i.e., $|\widetilde{f}_{\widetilde{C}_{\pi(2)}\widetilde{C}_{\pi(i)}}|$ assuming $l > 2$). If needed iterations $3,4,\ldots,j-1$ are applied. If at iteration *j*, there still exists some unresolved groups and G_j itself is also unresolved, the procedure described below will be used. (This case includes the case where G_1 is unresolved.)

At this point, the values of $\pi(1),\pi(2),\ldots,\pi(j-1)$ have been finalized (so is the value of π for any other group with only one class). However, the values of $\pi(j),\pi(j+1),\ldots,\pi(l)$ (group G_j) is not final (since 1st and 2nd signature have not made them distinct.) The final value for $\pi(j)$ can be any value among $\pi(j),\pi(j+1),\ldots,\pi(l)$. Finalizing the value of $\pi(j)$ to one of $\pi(j),\pi(j+1),\ldots,\pi(l)$ is equivalent to splitting $G_j = \{\widetilde{C}_{\pi(j)},\widetilde{C}_{\pi(j+1)},\ldots,\widetilde{C}_{\pi(l)}\}$ to groups $\{\widetilde{C}_{\pi(i)}\}$ and $\{\widetilde{C}_{\pi(j)},\ldots,\widetilde{C}_{\pi(i-1)},\widetilde{C}_{\pi(i+1)},\ldots,\widetilde{C}_{\pi(l)}\}$ and updating π and indices of groups so these two groups are represented by $G_j = \{\widetilde{C}_{\pi(j)}\}$ and $G_{j+1} = \{\widetilde{C}_{\pi(j+1)},\widetilde{C}_{\pi(j+2)},\ldots,\widetilde{C}_{\pi(l)}\}$. Therefore, there are $l-j+1$ ways to do this split ($l-j+1$ ways to specify new G_j) and if the phase of $C_{\pi(j)}$ is undecided,

then there will be two ways to resolve the group, new G_j (two phases $p=1$ and $p=0$.) Consequently, there are $r=l-j+1$ (or $r=2(l-j+1)$) ways to specify and resolve the new group, G_j. All these r cases need to be tracked, since it is unknown which one(s) will result in a maximal transformation. For each case, the 2nd-signatures, $|\widetilde{f}_{\widetilde{C}_{\pi(j)}\widetilde{C}_{\pi(i)}}|$, are used to first order classes inside the unresolved groups among $G_{j+1},G_{j+2},\ldots,G_m$ and then split them based on the outcome of ordering.

This process will continue for all r cases, recursively (cf. the **recursive-resolve** algorithm), until all groups are resolved. All resulting NP transformations $T_1,T_2,\ldots,T_s$ resulting from different cases are stored (where in general $s\geq r$ since each returns more than one transformation as a result of the recursion process.)

Because of the way they have been constructed, the inverses of CP transformations are among $\{T_1,T_2,\ldots,T_s\}$, or equivalently, $C'_f\subset\{T_1^{-1},T_2^{-1},\ldots,T_s^{-1}\}$. Hence by using 1st and 2nd signatures, we have limited the search for a C'_f among all $2^n n!$ transformations of Γ_n to search among $\{T_1^{-1},T_2^{-1},\ldots,T_s^{-1}\}$ which is a significantly smaller size space than Γ_n. Indeed, our experimental results confirm that most of the time all members of $\{T_1^{-1},T_2^{-1},\ldots,T_s^{-1}\}$ are CP transformations i.e., $C'_f=\{T_1^{-1},T_2^{-1},\ldots,T_s^{-1}\}$, which implies that most of the time only 1st and 2nd signatures are capable of determining the canonical form.

Members of C'_f among $\{T_1^{-1},T_2^{-1},\ldots,T_s^{-1}\}$ are identified based on the fact that $T_i^{-1}\in C_f$ if and only if $\widetilde{f}(T_i^{-1}X)\succeq\widetilde{f}(T_j^{-1}X)$ for $1\leq j\leq s$. This task requires the performing comparison $\widetilde{f}(T_i^{-1}X)\succeq\widetilde{f}(T_j^{-1}X)$ repeatedly. The comparison is done based on their signature vectors. However, before using the signature vectors, the possibility equivalency of T_i and T_j should be considered i.e., first the relation $\widetilde{f}(T_i^{-1}X)=\widetilde{f}(T_j^{-1}X)$ should be checked since in case of equality their signature vectors will be equal as well. On the other hand, if $\widetilde{f}(T_i^{-1}X)\neq\widetilde{f}(T_j^{-1}X)$ as we proved in a theorem earlier, their signature vectors are different i.e., either $\widetilde{f}(T_i^{-1}X)\succ\widetilde{f}(T_j^{-1}X)$ or $\widetilde{f}(T_i^{-1}X)\prec\widetilde{f}(T_j^{-1}X)$.

The algorithm will return C'_f which in general may contain more than one transformation.

As a result of the way in which these NP transformations are obtained, they all have the same set of 0th and 1st signatures and some of their 2nd signatures are equal as well. Hence, to avoid redundancy in comparing $\widetilde{f}(T_i^{-1}X)$ and $\widetilde{f}(T_j^{-1}X)$, only signatures that are not already determined to be equal are generated and compared. Since comparison is done based on lexicographic comparison of signature vectors, signatures are generated one by one based on their significance order. Furthermore, only in case of equality, the subsequent signatures are generated and compared. Experimental results show that for nearly all functions, 1st and 2nd signatures conclude the comparison of $\widetilde{f}(T_i^{-1}X)$ and $\widetilde{f}(T_j^{-1}X)$.

Different steps of the proposed techniques are summarized in the following pseudo-code descriptions of **Compute_**C_f and **Recursive_Resolve** algorithms.

Algorithm *Compute*_C_f ($f(X)$)

Input: A Boolean function $f(X)$

Output: The canonical form $F(X)$ and CP transformations C'_f.

Using the 0th-signature perform output phase assignment ;
Using the1st-signatures,
    Assign phases ;
    Create symmetry classes ;
    Order and group classes to groups $G_1, G_2, \ldots, G_k$;

Recursive_Resolve($G_1, G_2, \ldots, G_k; C_f$) ;

Set the canonical form: $F(X) = \widetilde{f}(TX)$ where $T \in C_f$;

Algorithm Recursive_Resolve ($G_1, G_2, \ldots, G_k; C_f$ **)**

Input: Ordered groups ($G_1, G_2, \ldots, G_k$)

Output: The CP transformations C'_f

$i = 1; C'_f = \{\}$;

while ($i \le m$) { // *m* is the number of classes

    if (G_i is resolved) { // $G_i = \{\widetilde{C}_{\pi(i)}\}$

        for (all unresolved groups G_j){

            use signatures | $\widetilde{f}_{\widetilde{C}_{\pi(i)}\widetilde{C}_{\pi(l)}}$ | ($\widetilde{C}_{\pi(l)} \in G_j$) to assign phase, order and split G_j ;

            update indices of groups and classes;

}

$i = i + 1$;

} else { // $G_i = \{\widetilde{C}_{\pi(i)}, \widetilde{C}_{\pi(i+1)}, \ldots, \widetilde{C}_{\pi(l)}\}$ is not resolved

// for space limitation assume the phase of $G_i = \{\widetilde{C}_{\pi(i)}, \widetilde{C}_{\pi(i+1)}, \ldots, \widetilde{C}_{\pi(l)}\}$ is decided

for ($j = i$; $j \le l$; $j{+}{+}$) {

split G_i to groups $\{\widetilde{C}_{\pi(j)}\}$ and $\{\widetilde{C}_{\pi(i)}, \ldots, \widetilde{C}_{\pi(j-1)}, \widetilde{C}_{\pi(j+1)}, \ldots, \widetilde{C}_{\pi(l)}\}$;

update indices of groups and classes: ($G_1, G_2, \ldots, G_k, G_{k+1}$);

recursive-resolve ($G_1, G_2, \ldots, G_k, G_{k+1}; C_f'^{TEMP}$) ;

if ($C_f' = \{\}$ or $\widetilde{f}(T^{TEMP}X) \succ \widetilde{f}(TX)$) {// $T \in C_f', T^{TEMP} \in C_f'^{TEMP}$

$C_f' = C_f'^{TEMP}$;

} else if ($\widetilde{f}(T^{TEMP}X) = \widetilde{f}(TX)$) {

$C_f' = C_f' \bigcup C_f'^{TEMP}$;

}

}

return;

}

}

// At this point there are *m* groups and all of them are resolved

T =Transformation resulted from current phase assignment and π;

$C_f' = \{T^{-1}\}$;

return;

In the above description of the algorithm we have not included the case where the output phase may not be decided by the 0^{th} signature. In such a case, the following steps will have to be performed for both output phases. Let's assume that C_f' is the set of NP transformations returned by the algorithm for f and C_f'' is returned for $\bar{f}$ then if $f(T'X) \succ \bar{f}(T''X)$ (where $T' \in C_f'$ and $T'' \in C_f''$) then output phase is $q = 1$, if $f(T'X) \prec \bar{f}(T''X)$ then $q = 0$ and if $f(T'X) = \bar{f}(T''X)$ then both phases can result in the canonical form i.e., $F(X) = f(T'X) = \bar{f}(T''X)$. And the set of CP transformations is set to $C_f' \cup C_f''$.

The algorithm will return C_f' which in general may contain more than one NP transformations. Based on the members of C_f' SP transformations (other than transformations corresponding to simple symmetries) are detected i.e., $T \in C_f', T' \in C_f' \Rightarrow T^{-1}T' \in S_f$. Equivalently, for an NP

transformation $T \in C'_f$, $T^{-1}C'_f \subset S_f$. The remaining member of S_f can be generated by cascading NP transformation $T^{-1}C'_f$ of with transformations of W. By cascading we mean generation every transformation that can be generated by members of $T^{-1}C'_f$ and W which might require repeated cascading since the cascade operation is not commutative.

VII Experimental Results

The technique presented above has been implanted as part of the SIS logic synthesis environment. To reveal the effectiveness of the proposed technique, the proposed canonical form is computed for all cells in a cell library, containing a large number of complex cells with up to 20 inputs. To asses the efficiency of the method, a large number of randomly generated logic cells with different input counts were added to the library. Figure 1 shows the worst-case and average run-times required for computing the canonical form in terms of the number of inputs; i.e., the height of the n^{th} bars are the worst-case and average runtimes for all n-input cells. Yellow bars present results provided in reference [10].

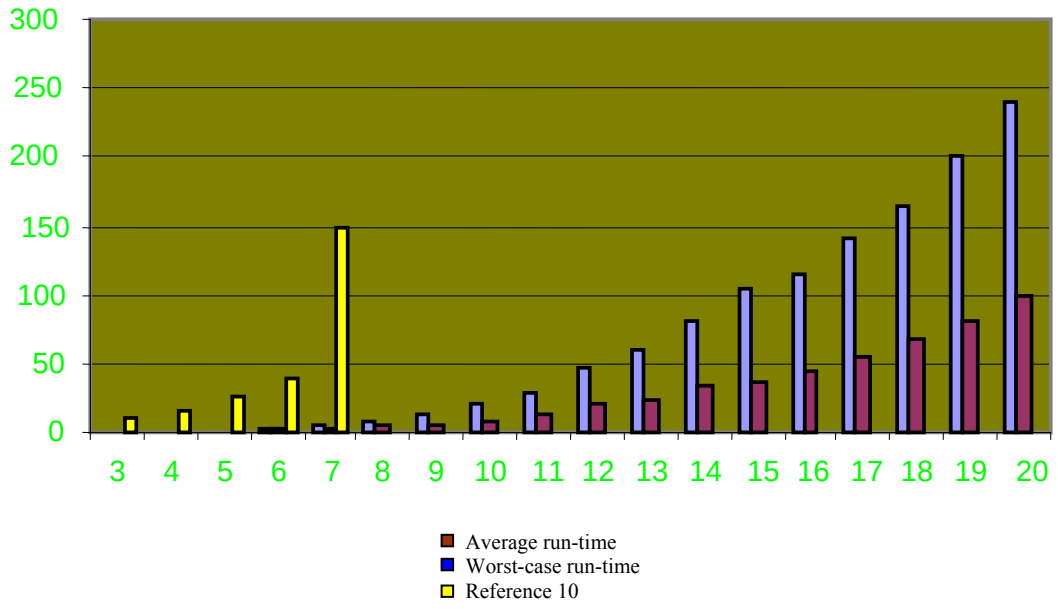

Figure 1. Worst-case and average runtimes to compute canonical forms

The run-times in this Figure 1 (Y-axis) are in microseconds and include data for cells with more than five inputs. This is because run-times for cells with as few as five inputs are too small (less than 1μ-sec) to be discernible. As an example the worst-case 20-input cell was a multiplexer

with four select inputs for which the algorithm takes 240 microseconds to compute its canonical form.

These results show a major improvement in run-time over previous approaches [9][10]. (Notice that reference [9] does not handle complementation of inputs and output and reference [10] entails enormous space complexity.) For nearly all of the cells in the library, the canonical forms were computed using only the zeroth, 1st and 2nd signatures. Only one of the cells required the use of the 3rd signatures and none of them required the use of higher order signatures. However, the algorithm given above is complete and able to handle functions that may require the use of higher order signatures for computing the canonical form.

As mentioned in the paper, one of the advantages of this technique is identifying all symmetry relations for the given function.

The runtime of the algorithm for any function has a direct relation with the number of non-simple symmetry relations of the function. More precisely, the higher the number of non-simple SP transformations, the higher the runtime. Figure 2 demonstrates the number of non-simple SP transformations for functions that correspond to the worst-case runtimes in the previous experiment. Note that the number of non-simple SP transformations is equal to the number of non-trivial CP transformations i.e., $|C'_f|$.

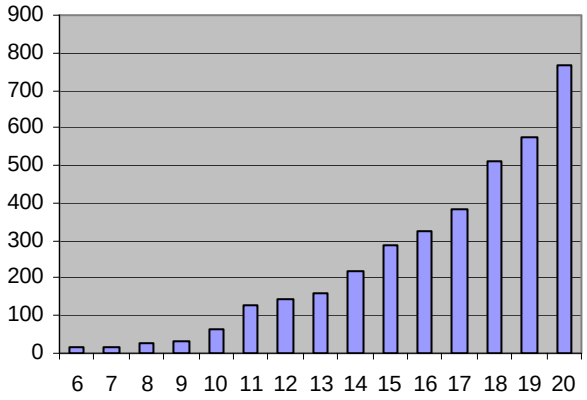

Figure 2. Number of non-simple SP transformations

VIII Conclusions

A new efficient and compact canonical form was defined and an effective algorithm for computing the proposed canonical form was provided in this paper. The compactness and efficiency of the presented methods enables the approach to be applicable to a wide range of Boolean networks as apposed to previous approaches that either do not solve the problem generally or only handle functions with limited number of inputs. This paper addresses the general Boolean matching problem in which both permutation and complementation of inputs and output are considered. The proposed canonical form was based on using generalized signatures to obtain all CP transformations on inputs. Signatures were defined very effectively and first, most powerful signatures (that include more information about the function) are generated and used followed by less significant signatures, only if necessary. Using the resulted CP transformations the information about all symmetry relations are provided. Experimental results demonstrate the efficiency of the proposed approach and it was observed, in nearly all cases zeroth, 1st and 2nd signatures are enough to provide the canonical form and since these signatures is performed efficiently by ordering variables, the proposed approach is associated with a very low computational complexity.

References

[1] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

[2] L. Benini and G. De Micheli, "A survey of Boolean matching techniques for library binding," *ACM Trans. Design Automation of Electronic Systems*, vol. 2, no. 3, pp. 193–226, July 1997.

[3] M. A. Harrison, Introduction to Switching and Automata Theory, McGraw-Hill, 1965.

[4] J. Mohnke, P. Molitor, and S. Malik, "Limits of using signatures for permutation independent Boolean comparison," *Proc. of ASP Design Automation Conf.*, pp. 459-464, 1995.

[5] J. R. Burch and D. E. Long, "Efficient Boolean function matching," in *Proc. Int. Conf. on Computer-Aided Design*, pp. 408–411, Nov. 1992.

[6] Q. Wu, C. Y. R. Chen, and J. M. Acken, "Efficient Boolean matching algorithm for cell libraries," *Proc. IEEE Int. Conf. on Computer Design*, pp. 36–39, Oct. 1994.

[7] U. Hinsberger and R. Kolla, "Boolean matching for large libraries," *Proc. of Design Automation Conf.*, pp. 206–211, June 1998.

[8] D. Debnath and T. Sasao, "Fast Boolean matching under permutation using representative," *Proc. ASP Design Automation Conf.*, pp. 359–362, Jan. 1999.

[9] J. Ciric and C. Sechen, "Efficient canonical form for Boolean matching of complex functions in large libraries," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 5, pp. 535–544, May 2003.

[10] D. Debnath and T. Sasao, "Efficient computation of canonical form for Boolean matching in large libraries," Proc. *ASP Design Automation Conf.*, pp. 591–596, Jan. 2004.

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

[12] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *AIEE Trans*., 57:713-723, 1938.

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

[14] K. S. Chung and C. L. Liu, "Local transformation techniques for multi-level logic circuits utilizing circuit symmetries for power reduction," in Proc. of *Internat. Symp. on Low Power Electronics and Design*, pp. 215–220, August 1998.

[15] S. Panda, F. Somenzi, and B. F. Plessier, "Symmetry detection and dynamic variable ordering of decision diagrams," in Proc. *International Conference on Computer-Aided Design*, pp. 628–631, November 1994.