Library Design for Symmetric Decomposition

Victor N. Kravets and Karem A. Sakallah

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109
{vkravets,karem}@eecs.umich.edu

Abstract

Using a very general symbolic decomposition template for logic synthesis we show how to pre-compute libraries for the decomposition patterns implied by the function structure. When coupled with decomposition the outfitted libraries are intended to produce improved synthesis quality. We illustrate the pre-computation process for functions that are symmetric in some inputs. For these functions we derive a set of fan-in-bounded cell libraries that guarantee a reduction in the width of the circuit being synthesized with each successive decomposition step.

I. Introduction and Motivation

In this report we take a fresh look at functional decomposition, and show how it can be used to infer appropriate decomposition patterns, and their underlying libraries, from a function structure. Functional decomposition has been studied by many authors. The original concepts were due to Ashenhurst [1], Curtis [8], and Roth and Karp [16]. These early investigations were mostly concerned with the existence of certain types of decomposition rather than the development of scalable synthesis algorithms. More recently, several authors have re-visited this early work for application in the limited domain of FPGA synthesis [13, 15, 19, 20, 23]. Practical general-purpose synthesis approaches, based on fast algebraic division algorithms, emerged in the early eighties [4, 22]. The primary motivation for these approaches was efficient decomposition and realization of large random logic functions by a two-stage process: technology-independent restructuring followed by binding to a specified library of primitive gates. This methodology enjoyed a great deal of success and is incorporated in most commercial synthesis tools in use today.

The central idea of this report is based on the premise that functional specifications have global *structural attributes*, that can be profitably used to induce a favorable structural implementation, while reducing the run time complexity of the synthesis process. These attributes can have a profound effect on the suitability of one decomposition type over another. They can be further utilized to study requirements on the functionality of library primitives

to make a particular decomposition type effective. Thus, by judiciously coupling the decomposition type with a library using structural attributes of a function we are able to merge the traditionally separate technology-independent and technology-dependent synthesis stages. The effect of the integration leads to improved synthesis quality, reflecting the global functional properties in the final circuit structure.

We begin in Section II by presenting a very general symbolic model of functional decomposition and show how it can be "solved" to determine all feasible decompositions. In Section III, we develop a model to pre-compute libraries under a set of practical constraints that are imposed under existing semiconductor technologies. This model is applied in Section IV to pre-compute symmetric libraries. Section V concludes the paper with suggestions on further extensions of this methodology.

II. Symbolic Formulation of Decomposition

In this section we propose a symbolic model for functional decomposition that allows us to pose and answer several key questions related to scalable synthesis, including the existence of a decomposition, and the existence of universal primitives that allow the decomposition of certain classes of functions.

A. Generic decomposition template

Given an *n*-variable Boolean function $f(\mathbf{x})$, and *k* *n*-variable Boolean functions $g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})$, we say that *f* has an *n*-to-*k* decomposition with respect to $g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})$ if and only if there exists a *k*-variable function h such that

$$f(\mathbf{x}) = h(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})) \tag{1}$$

A pictorial representation of this decomposition template is shown in Fig. 1; h will be referred to as the *composition function*, whereas $g_1, \ldots, g_k$ will be called the *decomposition functions*. These functions introduce intermediate variables $y_1, \ldots, y_k$ into the network that serve as the support of the composition function. The decomposition is *support-reducing* if $k < n$. The *k* decomposition functions can be viewed as a single multi-output decomposition function $\mathbf{g}(\mathbf{x}) \equiv (g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$, and the intermediate variables can be represented by a *k*-vector $\mathbf{y} \equiv (y_1, \ldots, y_k)$.

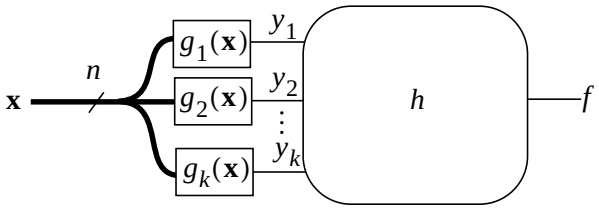


Fig. 1. Generic decomposition step

The decomposition template in (1) is sufficiently general to encompass all types of functional decomposition described in the literature, including simple and complex disjunctive and non-disjunctive decompositions [8]. As we show later, support-reducing decompositions in terms of fan-in bounded decomposition functions are particularly attractive from a practical perspective. Before such restrictions are imposed, though, we show in the remainder of this section the relations that must exist between the composition and decomposition functions for equation (1) to hold.

B. Computation of composition function

To determine if the decomposition in (1) exists, we can solve for $h(\mathbf{y})$ in terms of $\mathbf{g}(\mathbf{x})$ and $f(\mathbf{x})$. The solution, in general, is not unique and can be expressed as a function interval [5]. The interval solution for $h(\mathbf{y})$ corresponds to a partially specified function, whose flexibilities are modeled in terms of the following function:

$$c(\mathbf{x}, \mathbf{y}) = (y_1 \equiv g_1(\mathbf{x}))\ldots(y_k \equiv g_k(\mathbf{x})) \tag{2}$$

Originally introduced by Cerny [7] as an *output characteristic function*, (2) captures the consistent input-output behavior of a circuit. In recent years output characteristic functions have been used to describe the flexibility that arises in design optimization. Viewed as Boolean relations, Brayton and Somenzi [2] described how they can be used to compute the flexibility in optimizing hierarchical designs. Savoj [18] has used the output characteristic function to describe the maximum flexibility in the optimization of Boolean networks.

Function $c(\mathbf{x}, \mathbf{y})$ represents the constraints introduced by the decomposition functions which can viewed as a *care set* when selecting $h(\mathbf{y})$. Indeed, for each point $(\mathbf{x}^*, \mathbf{y}^*)$ from this set the value of $h(\mathbf{y}^*)$ must agree with the value of $f(\mathbf{x}^*)$; in all other points outside of $c(\mathbf{x}, \mathbf{y})$ we have a choice defining values of $h(\mathbf{y})$ arbitrarily. This flexibility in selecting $h(\mathbf{y})$ can be described by the means of a partially specified function $H(\mathbf{y})$ which identifies all valid selections for $h(\mathbf{y})$. We perform derivation of $H(\mathbf{y})$ in two steps: first, function $H(\mathbf{x}, \mathbf{y})$ is defined within the extended domain of function $c(\mathbf{x}, \mathbf{y})$, and then its domain is reduced to obtain $H(\mathbf{y})$.

In our first step of reasoning we show formally that $H(x, y)$ can be modeled in terms of the function interval. Formal derivation of this result is given below:

$$\begin{aligned}
& c(\mathbf{x}, \mathbf{y}) \rightarrow (H(\mathbf{x}, \mathbf{y}) \equiv f(\mathbf{x})) \\
\Leftrightarrow\ & c(\mathbf{x}, \mathbf{y}) \cdot (H(\mathbf{x}, \mathbf{y}) \oplus f(\mathbf{x})) \\
\Leftrightarrow\ & c(\mathbf{x}, \mathbf{y}) \cdot (\overline{H(\mathbf{x}, \mathbf{y})} \cdot f(\mathbf{x}) + H(\mathbf{x}, \mathbf{y}) \cdot \overline{f(\mathbf{x})}) = 0 \\
\Leftrightarrow\ & c(\mathbf{x}, \mathbf{y}) \cdot \overline{H(\mathbf{x}, \mathbf{y})} \cdot f(\mathbf{x}) + c(\mathbf{x}, \mathbf{y}) \cdot H(\mathbf{x}, \mathbf{y}) \cdot \overline{f(\mathbf{x})} = 0 \\
\Leftrightarrow\ & c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x}) \cdot \overline{H(\mathbf{x}, \mathbf{y})} = 0 \wedge c(\mathbf{x}, \mathbf{y}) \cdot \overline{f(\mathbf{x})} \cdot H(\mathbf{x}, \mathbf{y}) = 0 \\
\Leftrightarrow\ & (c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x}) \leq H(\mathbf{x}, \mathbf{y})) \wedge (H(\mathbf{x}, \mathbf{y}) \leq f(\mathbf{x}) + \overline{c(\mathbf{x}, \mathbf{y})})
\end{aligned} \tag{3}$$

The last assertion in the above derivation corresponds naturally to the interval:

$$H(\mathbf{x}, \mathbf{y}) = [c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x}), \overline{c(\mathbf{x}, \mathbf{y})} + f(\mathbf{x})] \tag{4}$$

The dependence of $H(\mathbf{x}, \mathbf{y})$ on the $\mathbf{x}$ variables in the above interval makes the solution unconditionally consistent, implying that for a given $c(\mathbf{x}, \mathbf{y})$ there is always a non-empty $H(\mathbf{x}, \mathbf{y})$. Note that when $c(\mathbf{x}, \mathbf{y}) = 1$, equation (4) reduces to $f(\mathbf{x})$; for $c(\mathbf{x}, \mathbf{y}) = 0$, on the other hand, the equation becomes the interval [0, 1] denoting an arbitrary function.

The dependence on $\mathbf{x}$ in $H(\mathbf{x}, \mathbf{y})$ is not consistent with the decomposition template in (1) however, – the template restricts composition function h to be vacuous in the $\mathbf{x}$ variables, while $H(\mathbf{x}, \mathbf{y})$ does not. To make $H(\mathbf{x}, \mathbf{y})$ vacuous in the $\mathbf{x}$ variables we must ensure that for any given assignment $\mathbf{y}^*$, values of $H(\mathbf{x}, \mathbf{y})$ agree for all assignments $\mathbf{x}^*$. This requirement is reflected in the following relation

$$H(\mathbf{y}) = \forall \mathbf{x} H(\mathbf{x}, \mathbf{y}) \tag{5}$$

or equivalently

$$H(\mathbf{y}) = \forall \mathbf{x}[(c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x})), (\overline{c(\mathbf{x}, \mathbf{y})} + f(\mathbf{x}))] \tag{6}$$

The universal quantification of $\mathbf{x}$ can be distributed inside of the interval relying on the identity for abstracting variable from an interval [12]:

$$H(\mathbf{y}) = [\exists \mathbf{x}(c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x})), \forall \mathbf{x}(\overline{c(\mathbf{x}, \mathbf{y})} + f(\mathbf{x}))] \tag{7}$$

The existential and universal quantification of $\mathbf{x}$ in the interval is consistent with the earlier result, given in [5, Th. 4.9.1], for finding redundant variables in a partially specified function.

The existential and universal quantification of **x** from the lower and upper interval bounds corresponds to the removal of these variables from the support of possible h to make it a function of just intermediate variables **y,** thereby reflecting the structure of Fig. 1. If the interval (7) is non-empty, then there is a Boolean function h which is vacuous in $\mathbf{x}$, and which belongs to this interval. We say that decomposition exists when the interval is non-empty; otherwise the decomposition does not exist. In the examples below we illustrate how choices on the decomposition functions can effect the existence of decomposition.

Example 2.1 Let $f(x_1, x_2) = x_1 \oplus x_2$, $g_1(x_1, x_2) = x_1$, $g_2(x_1, x_2) = \bar{x}_1 + \bar{x}_2$, and $g_3(x_1, x_2) = x_2$. The input-output characteristic function of the decomposition functions is:

$$c(\mathbf{x}, \mathbf{y}) = (y_1 \equiv x_1) \cdot (y_2 \equiv \bar{x}_1 + \bar{x}_2) \cdot (y_3 \equiv x_2)$$

Using this function we compute the lower bound for the interval (7):

$$\exists \mathbf{x}(c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x})) = \bar{y}_1 y_2 \bar{y}_3 \cdot 0 + y_1 y_2 \bar{y}_3 \cdot 1 + \bar{y}_1 y_2 y_3 \cdot 1 + y_1 \bar{y}_2 y_3 \cdot 0$$

Similarly for the upper bound:

$$\forall \mathbf{x}(\overline{c(\mathbf{x}, \mathbf{y})} + f(\mathbf{x})) = (y_1 + \bar{y}_2 + y_3 + 0) \cdot (\bar{y}_1 + \bar{y}_2 + y_3 + 1) \cdot (y_1 + \bar{y}_2 + \bar{y}_3 + 1) \cdot (\bar{y}_1 + y_2 + \bar{y}_3 + 0)$$

These bounds define an interval of sixteen composition functions h which can be equally selected for the decomposition of f.

$$H = [y_1y_2\bar{y}_3 + \bar{y}_1y_2y_3, \bar{y}_1\bar{y}_2 + y_1y_2 + \bar{y}_2\bar{y}_3 + y_2y_3]$$

For instance, $h_1(y_1, y_2, y_3) = \bar{y}_1y_3 + y_1y_2$ and $h_2(y_1, y_2, y_3) = y_1y_2 + y_2y_3$ are two functions from this interval that represent two possible decompositions of f as can be readily verified by substitution in (1). ■

Example 2.2 Let $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$, $g_1(x_1, x_2, x_3) = \bar{x}_1 + \bar{x}_2$, and $g_2(x_1, x_2, x_3) = x_3$. The H interval in this case is:

$$H = [y_1 + y_2, \bar{y}_1y_2]$$

which is empty since its upper bound is less than its lower bound. Thus, f cannot be decomposed in terms of the given decomposition functions. If, however, the first decomposition function is replaced with $g_1(x_1, x_2, x_3) = x_1 \oplus x_2$, then (1) yields the following interval:

$$H = [y_1 \oplus y_2, y_1 \oplus y_2]$$

representing a unique composition function. ■

In general, it is always possible to find k decomposition functions that make the interval in (7) non-empty. For example, when $k \geq n$ the decomposition functions can be selected by assuming that at least n of these functions are trivial pass through wires corresponding to the support of f. The remaining decomposition functions can be selected arbitrarily since their output signals can be assumed redundant in $h(\mathbf{y})$. Similarly, when $k < n$ decomposition functions always make the interval non-empty letting one of them correspond to $f(\mathbf{x})$. Such trivial selections of decomposition functions have little practical value though, and additional constraints on decomposition functions must be imposed to make decomposition useful in synthesis. These additional constraints, and their implication on synthesis quality are addressed in this work.

C. Computation of decomposition functions

Equation (7) can be used to compute sets of decomposition functions that will guarantee the existence of a decomposition according to the template in (1). Computation of such decomposition functions forms the starting point for the problem of library construction which we discuss later in Section III. To solve for the decomposition functions, we begin by noting that an arbitrary *n*-variable Boolean function can be expressed in terms of 2^n binary coefficients that denote the function value at each point in its variable space. Thus, we can express $g_j(\mathbf{x})$ as:

$$g_j(\mathbf{x}) = \sum_{i=0}^{2^n-1} \gamma_{ij} \cdot m_i(\mathbf{x}) \tag{8}$$

where $\gamma_{ij} \in \{0, 1\}$ and $m_i(\mathbf{x})$ is the minterm on $\mathbf{x}$ whose bits form decimal value i. Using $\Gamma \equiv [\gamma_{ij}]$ to denote the $2^n \times k$ matrix of coefficients representing the k decomposition functions, the care set $c(\mathbf{x}, \mathbf{y})$ can be re-written as:

$$C(\mathbf{x}, \mathbf{y}, \Gamma) = \prod_{j=1}^{k} \left[y_j \equiv \sum_{i=0}^{2^n - 1} \gamma_{ij} \cdot m_i(\mathbf{x}) \right] \tag{9}$$

A decomposition exists if the interval (7) is non-empty, landing the following derivation:

$$\begin{aligned} & [\exists \mathbf{x}(c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x})), \forall \mathbf{x}(\overline{c(\mathbf{x}, \mathbf{y})} + f(\mathbf{x}))] \\ \Leftrightarrow\ & \exists \mathbf{x}(c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x})) \leq \forall \mathbf{x}(\overline{c(\mathbf{x}, \mathbf{y})} + f(\mathbf{x})) \\ \Leftrightarrow\ & \overline{\exists \mathbf{x}(c(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{x}))} + \forall \mathbf{x}(\overline{c(\mathbf{x}, \mathbf{y})} + f(\mathbf{x})) = 1 \end{aligned}$$

Substituting (9) in the last step of the above derivation, and universally quantifying **y,** we have

$$G(\Gamma) = \forall \mathbf{y}(\overline{\exists \mathbf{x}(C(\mathbf{x}, \mathbf{y}, \Gamma) \cdot f(\mathbf{x}))} + \forall \mathbf{x}(\overline{C(\mathbf{x}, \mathbf{y}, \Gamma)} + f(\mathbf{x}))) \tag{10}$$

which is a Boolean function that encodes all feasible decomposition functions **g**. The universal quantification of the $\mathbf{y}$ variables in (10) ensures that computed decomposition functions remain valid in the decomposition for all combinations of their output values.

Example 2.3 We apply formula (10) to compute 3-to-2 decomposition solutions for the $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ function. The space of two 3-input decomposition functions is encoded as:

$$\begin{aligned} C(\mathbf{x}, \mathbf{y}, \Gamma) = & (y_1 \equiv (\gamma_{01}\bar{x}_1\bar{x}_2\bar{x}_3 + \gamma_{11}\bar{x}_1\bar{x}_2x_3 + \gamma_{21}\bar{x}_1x_2\bar{x}_3 + \gamma_{31}\bar{x}_1x_2x_3 + \\ & \gamma_{41}x_1\bar{x}_2\bar{x}_3 + \gamma_{51}x_1\bar{x}_2x_3 + \gamma_{61}x_1x_2\bar{x}_3 + \gamma_{71}x_1x_2x_3)) \\ & \times (y_2 \equiv (\gamma_{02}\bar{x}_1\bar{x}_2\bar{x}_3 + \gamma_{12}\bar{x}_1\bar{x}_2x_3 + \gamma_{22}\bar{x}_1x_2\bar{x}_3 + \gamma_{32}\bar{x}_1x_2x_3 + \\ & \gamma_{42}x_1\bar{x}_2\bar{x}_3 + \gamma_{52}x_1\bar{x}_2x_3 + \gamma_{62}x_1x_2\bar{x}_3 + \gamma_{72}x_1x_2x_3)) \end{aligned}$$

Together with f this function is then used to compute formula (10). For the part corresponding to the lower bound in the formula we have:

$$\begin{aligned} \exists \mathbf{x}(C(\mathbf{x}, \mathbf{y}, \Gamma) \cdot f(\mathbf{x})) = & (y_1 \equiv \gamma_{01}) \cdot (y_2 \equiv \gamma_{02}) \cdot 0 + (y_1 \equiv \gamma_{11}) \cdot (y_2 \equiv \gamma_{12}) \cdot 1 + \\ & (y_1 \equiv \gamma_{21}) \cdot (y_2 \equiv \gamma_{22}) \cdot 1 + (y_1 \equiv \gamma_{31}) \cdot (y_2 \equiv \gamma_{32}) \cdot 0 + \\ & (y_1 \equiv \gamma_{41}) \cdot (y_2 \equiv \gamma_{42}) \cdot 1 + (y_1 \equiv \gamma_{51}) \cdot (y_2 \equiv \gamma_{52}) \cdot 0 + \\ & (y_1 \equiv \gamma_{61}) \cdot (y_2 \equiv \gamma_{62}) \cdot 0 + (y_1 \equiv \gamma_{71}) \cdot (y_2 \equiv \gamma_{72}) \cdot 1 \end{aligned}$$

Similarly for the upper bound part:

$$\begin{aligned} \forall \mathbf{x}(\overline{C(\mathbf{x}, \mathbf{y}, \Gamma)} + f(\mathbf{x})) = & ((y_1 \oplus \gamma_{01}) + (y_2 \oplus \gamma_{02}) + 0) \cdot ((y_1 \oplus \gamma_{11}) + (y_2 \oplus \gamma_{12}) + 1) \cdot \\ & ((y_1 \oplus \gamma_{21}) + (y_2 \oplus \gamma_{22}) + 1) \cdot ((y_1 \oplus \gamma_{31}) + (y_2 \oplus \gamma_{32}) + 0) \cdot \\ & ((y_1 \oplus \gamma_{41}) + (y_2 \oplus \gamma_{42}) + 1) \cdot ((y_1 \oplus \gamma_{52}) + (y_2 \oplus \gamma_{52}) + 0) \cdot \\ & ((y_1 \oplus \gamma_{61}) + (y_2 \oplus \gamma_{62}) + 0) \cdot ((y_1 \oplus \gamma_{71}) + (y_2 \oplus \gamma_{72}) + 1) \end{aligned}$$

Dropping 0-terms in above two expansions computation of function $G(\Gamma)$ now has form:

$$
\begin{aligned}
G(\Gamma) \;=\; \forall \mathbf{y}[&((y_1 \oplus \gamma_{11}) + (y_2 \oplus \gamma_{12})) \cdot ((y_1 \oplus \gamma_{21}) + (y_2 \oplus \gamma_{22})) \cdot \\
&((y_1 \oplus \gamma_{41}) + (y_2 \oplus \gamma_{42})) \cdot ((y_1 \oplus \gamma_{71}) + (y_2 \oplus \gamma_{72})) \\
+\,&((y_1 \oplus \gamma_{01}) + (y_2 \oplus \gamma_{02})) \cdot ((y_1 \oplus \gamma_{31}) + (y_2 \oplus \gamma_{32})) \cdot \\
&((y_1 \oplus \gamma_{52}) + (y_2 \oplus \gamma_{52})) \cdot ((y_1 \oplus \gamma_{61}) + (y_2 \oplus \gamma_{62}))]
\end{aligned}
$$

Quantifying out $\mathbf{y}$ we have:

$$
\begin{aligned}
G(\Gamma) \;=\; &((\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72}) + \\
&(\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62})) \\
\times\,&((\bar{\gamma}_{11} + \gamma_{12}) \cdot (\bar{\gamma}_{21} + \gamma_{22}) \cdot (\bar{\gamma}_{41} + \gamma_{42}) \cdot (\bar{\gamma}_{71} + \gamma_{72}) + \\
&(\bar{\gamma}_{01} + \gamma_{02}) \cdot (\bar{\gamma}_{31} + \gamma_{32}) \cdot (\bar{\gamma}_{52} + \gamma_{52}) \cdot (\bar{\gamma}_{61} + \gamma_{62})) \\
\times\,&((\gamma_{11} + \bar{\gamma}_{12}) \cdot (\gamma_{21} + \bar{\gamma}_{22}) \cdot (\gamma_{41} + \bar{\gamma}_{42}) \cdot (\gamma_{71} + \bar{\gamma}_{72}) + \\
&(\gamma_{01} + \bar{\gamma}_{02}) \cdot (\gamma_{31} + \bar{\gamma}_{32}) \cdot (\gamma_{52} + \bar{\gamma}_{52}) \cdot (\gamma_{61} + \bar{\gamma}_{62})) \\
\times\,&((\gamma_{11} + \gamma_{12}) \cdot (\gamma_{21} + \gamma_{22}) \cdot (\gamma_{41} + \gamma_{42}) \cdot (\gamma_{71} + \gamma_{72}) + \\
&(\gamma_{01} + \gamma_{02}) \cdot (\gamma_{31} + \gamma_{32}) \cdot (\gamma_{52} + \gamma_{52}) \cdot (\gamma_{61} + \gamma_{62}))
\end{aligned}
$$

This is a function of 1812 ON-set minterms, each corresponding to a feasible 3-to-2 decomposition. However, only 99 of them define non-trivial (i.e. with no decomposition function, or its complement, corresponding to f) decomposition solutions invariant under complementation of the decomposition functions. This number can be further reduced by discarding solutions whose decomposition functions have redundant signals. Some of the more interesting solutions are:

Solution A:	**Solution B:**	**Solution C:**
$g_1 = x_1 \oplus x_2$	$g_1 = \bar{x}_1 x_2 + x_1 \bar{x}_3$	$g_1 = x_1 \bar{x}_2$
$g_2 = x_3$	$g_2 = \bar{x}_1 x_3 + x_1 x_2$	$g_2 = x_1 x_3 + \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3$

Example 2.4 Similarly to the previous example, application of (10) to $f(x_1, x_2, x_3) = \bar{x}_1 x_2 + x_1 x_3$ yields also 99 3-to-2 non-trivial decomposition solutions invariant under output complementation. We list some of them below:

Solution A:	**Solution B:**	**Solution C:**
$g_1 = x_1 + x_2$	$g_1 = \bar{x}_1 x_2$	$g_1 = \bar{x}_1 x_2$
$g_2 = x_1 x_3$	$g_2 = x_1 x_3 + x_2 x_3$	$g_2 = x_1 x_3$

■

Equation (10) encompasses all the decomposition solutions for a given function *f*. To find the decomposition solutions for an arbitrary *n*-variable function *f*, we introduce a vector of 2^n encoding coefficients $\Phi \equiv [\varphi_i]$ to express the universe of *n*-variable functions as:

$$
F(\mathbf{x}, \Phi) \;=\; \sum_{i=0}^{2^n - 1} \varphi_i \cdot m_i(\mathbf{x}) \tag{11}
$$

Note that a complete assignment to Φ represents a particular completely specified function f; partial assignments to Φ denote families of functions. Re-writing (10) in terms of $F(\mathbf{x}, \Phi)$ we obtain:

$$
G(\Phi, \Gamma) \;=\; \forall \mathbf{y}(\overline{\exists \mathbf{x}(C(\mathbf{x}, \mathbf{y}, \Gamma) \cdot F(\mathbf{x}, \Phi))} + \forall \mathbf{x}(\overline{C(\mathbf{x}, \mathbf{y}, \Gamma)} + F(\mathbf{x}, \Phi))) \tag{12}
$$

which is a Boolean function that encodes all feasible decomposition functions **g** for any given function *f*. We show next how this encoding function can be used to derive libraries of primitive decomposition functions suitable for large-scale synthesis.

Example 2.5 We can use equation (12) to compute 3-to-2 decompositions for all 3-variable functions. The universe of these functions can be encoded as:

$$\begin{aligned} F(\mathbf{x}, \Phi) = {} & \varphi_0 \cdot \bar{x}_1\bar{x}_2\bar{x}_3 + \varphi_1 \cdot \bar{x}_1\bar{x}_2 x_3 + \varphi_2 \cdot \bar{x}_1 x_2\bar{x}_3 + \varphi_3 \cdot \bar{x}_1 x_2 x_3 \\ & + \varphi_4 \cdot x_1\bar{x}_2\bar{x}_3 + \varphi_5 \cdot x_1\bar{x}_2 x_3 + \varphi_6 \cdot x_1 x_2\bar{x}_3 + \varphi_7 \cdot x_1 x_2 x_3 \end{aligned}$$

Using this encoding in (12) we can identify all 3-to-2 decompositions for every assignment to the Φ variable. Indeed, computation of $G(\Phi, \Gamma)$ shows that there are non-trivial decompositions for every function in $F(\mathbf{x}, \Phi)$ induced by the assignments to Φ variable. In particular, assignments $[011010001]$ and $[00110101]$ to the $\Phi \equiv [\varphi_0\varphi_1\varphi_2\varphi_3\varphi_4\varphi_5\varphi_6\varphi_7]$ variables induce Example 2.3 decomposition solutions computed for $x_1 \oplus x_2 \oplus x_3$ and $\bar{x}_1 x_2 + x_1 x_3$, respectively. Note that assignments to that assignments to the Φ variables for the two functions have the same number of 0's and 1's, implying their equivalent structures. This structural equivalence provides explanation for the equal number of decompositions generated in Example 2.3 for these two functions. ■

III. Enforcing Practical Decomposition Constraints

Although decomposition template in (1) is very general, the associated computational complexity and lack of qualitative constraints makes its use difficult in scalable synthesis. In this section we address the complexity problems by imposing practical fan-in constraints on the decomposition functions. These constraints are reflected in the modified decomposition template, which is used to define appropriate decomposition patterns deduced from a function structure.

A. Fan-in constraint

Equation (10) requires, in the worst case, the construction of a Boolean characteristic function of $2^n \cdot k$ encoding variables. Our first decomposition constraint for the equations (10) has therefore an objective of reducing exponential in n number of encoding variables Γ. We satisfy this objective by requiring the support of the $\mathbf{g}(\mathbf{x})$ functions to be bounded by s, where s is the maximum allowable fan-in of the underlying implementation technology; in current CMOS processes, s is typically four. When the support of each function in $\mathbf{g}(\mathbf{x})$ is known, and is bounded by s, we are effectively eliminating $k \cdot 2^{n-s}$ encoding variables in (11), thereby reducing exponentially its computational effort. In general, the fan-in bound restriction on the decomposition functions introduces additional algorithmic component whose goal is to identify suited support for the functions of $\mathbf{g}(\mathbf{x})$.

Although for the fan-in of at least 2 it is always possible to find decomposition functions (e.g. trivial wire functions) which make computation (10) non-empty, the number of such decomposition functions can vary greatly depending on a functional structure of a given $f(\mathbf{x})$. In general, the number of decomposition functions required to make (10) non-empty increases significantly as the fan-in constrain becomes stricter. The example below illustrates the effect of fan-in constraint on the existence of decomposition.

Example 3.1 Suppose that we would like to decompose function of a second sum bit in an n-bit adder with two decomposition functions. This function is given in terms of the following factored form:

$$s_1 = a_1 \oplus b_1 \oplus (a_0 b_0 + c_0(a_0 \oplus b_0))$$

Let $C(\mathbf{x}, \mathbf{y}, \Gamma)$ encode the space of two decomposition functions (as in Example 2.3). We can then compute the set of all decomposition solutions for s_1 using (10). Indeed, function $G(\Gamma)$, computed in (10), contains 1,116,591,939 non-trivial pairs of decomposition functions assuming invariance under complementation of their outputs. Restricting the fan-in of these solutions to four, we find that there is total of 795 solutions. Further restriction of three on the fan-in shows only four such solutions. Their decomposition functions are listed in the table below:

Solution A:	**Solution B:**	**Solution C:**	**Solution D:**
$g_1 = a_1 \oplus b_1$	$g_1 = c_0 \oplus a_1 \oplus b_1$	$g_1 = c_0 \oplus a_1 \oplus b_1$	$g_1 = c_0 \oplus a_1 \oplus b_1$
$g_2 = c_0 a_0 + c_0 b_0 + a_0 b_0$	$g_2 = \bar{c}_0 a_0 b_0 + c_0 \bar{a}_0 \bar{b}_0$	$g_2 = \bar{c}_0 a_0 \bar{b}_0 + c_0 \bar{a}_0 b_0$	$g_2 = \bar{c}_0 a_0 \bar{b}_0 + c_0 \bar{a}_0 b_0$

■

The general decomposition template in the presence of a fan-in bound s has the form:

$$f(\mathbf{x}) = h(g_1(\mathbf{x}_1), \ldots, g_k(\mathbf{x}_k)) \tag{13}$$

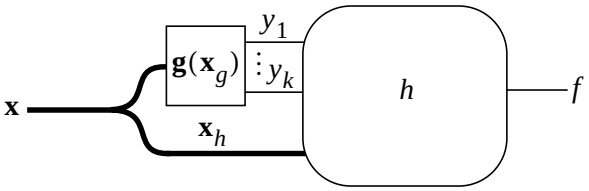
where $\mathbf{x}_i$'s are composed from the s-subsets of $\mathbf{x}$. When the decomposition variables $\mathbf{x}_i$ for each g_i are given to us we can now use computational form in (10) to find all feasible sets of k decomposition functions. Solving (10) simultaneously for all k decomposition functions however, still requires a large number of encoding variables Γ – in the worst case $k \cdot 2^s$. This makes computation of $G(\Gamma)$ a formidable task even for a small problem.

B. Modified decomposition template

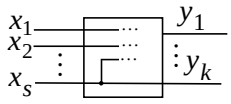
We reflect the fan-in bound restriction on the decomposition functions by defining a new decomposition template which partitions the input variables into two sets, $\mathbf{x}_g$ and $\mathbf{x}_h$, with $|\mathbf{x}_g| = s$ (see Fig. 2-a):

$$f(\mathbf{x}_g, \mathbf{x}_h) = h(g_1(\mathbf{x}_g), \ldots, g_t(\mathbf{x}_g), \mathbf{x}_h) \tag{14}$$

The $g_1(\mathbf{x}_g), \ldots, g_t(\mathbf{x}_g)$ functions, or collectively $\mathbf{g}(\mathbf{x}_g)$, can be assumed to be library primitives, possibly pass-through wires.



a) partition of the support $f(\mathbf{x})$ into $\mathbf{x}_g$ and $\mathbf{x}_h$



b) accommodation of non-disjoint decomposition within $\mathbf{g}(\mathbf{x}_g)$

Fig. 2. Illustration of the decomposition template $f(\mathbf{x}) = h(\mathbf{g}((\mathbf{x}_g), \mathbf{x}_h))$

Within this template the generality of decomposition in (13) still can be preserved if we change the algorithmic flow in which the equation is been solved. Instead of finding simultaneously all k decomposition functions for which (13) holds we find them iteratively for the subsets of $\mathbf{x}$. These subsets are required to have size of at most s, which is the fan-in bound on the decomposition functions. A particular solution for each of the subsets presents us with a collection of t decomposition functions, which can be viewed as a multi-output module. Encoding of a complete set of such solutions uses at most $t \cdot 2^s$ variables Γ.

Letting some of the decomposition functions to be trivial wire functions we are able to accommodate non-disjoint decomposition, allowing supports of the distinct subsets $\mathbf{x}_g$ to overlap during decomposition flow. This is illustrated in Fig. 2-b, where signal x_s is used not only in the non-trivial functions of $\mathbf{g}(\mathbf{x}_g)$ but also becomes available to h as a wire. This wire can be reused in the subsequent decomposition steps. The effect of such reuse is a preserved generality of the decomposition template (13).

Example 3.2 Using template (14) let us decompose function

$$f(a, b, c, d) = ac + \bar{a}d + b\bar{d} + \bar{b}d$$

using three non-trivial decomposition functions at the first level of logic, and the fan-in bound $s = 3$. The first step of decomposition finds a subset of decomposition functions whose $\mathbf{x}_g$ support is $\{a, b, c\}$. Letting decomposition functions to be

$$g_1(a, b, c) = \bar{a}b + ac$$
$$g_2(a, b, c) = b$$
$$g_3(a, b, c) = c$$

we have $f(a, b, c, d) = h(\bar{a}b + ac, b, c, d)$. The f function can be further decomposed using another subset of decomposition functions whose support is $\{b, c, d\}$. The set of decomposition functions for this support is

$$g_1(b, c, d) = b \oplus d$$
$$g_2(b, c, d) = cd$$

These two decomposition functions are now used to complete decomposition of f at the first level:

$$f(a, b, c, d) = h(\bar{a}b + ac, b \oplus d, cd)$$

Observe that all of the decomposition functions have overlapping variables in their supports. This is achieved by letting some of the decomposition functions at the first step of decomposition to be trivial wires. These wires are then used as support to the decomposition functions introduced at the next step. ■

When $\mathbf{x}_g$ variables are given, we find their sets of feasible decomposition functions by modifying computation of $G(\Gamma)$ in (13). The modification is based on the observation that variables in $\mathbf{x}_h$ can be viewed as trivial wire functions, which we use to relabel their corresponding output signals in $\mathbf{y}$. The new variant on the computation of $G(\Gamma)$ now has form:

$$G(\Gamma) = \forall \mathbf{y}_g, \mathbf{x}_h(\overline{\exists \mathbf{x}_g(C(\mathbf{x}_g, \mathbf{y}_g, \Gamma) \cdot f(\mathbf{x}))} + \forall \mathbf{x}_g(\overline{C(\mathbf{x}_g, \mathbf{y}_g, \Gamma)} + f(\mathbf{x})))$$

where $\mathbf{y}_g$ variables correspond to the output signals of decomposition functions $\mathbf{g}(\mathbf{x}_g) \equiv (g_1(\mathbf{x}_g), \ldots, g_t(\mathbf{x}_g))$. The g subscript in $\mathbf{y}_g$ is superfluous however, and we therefore re-write the above formula as:

$$G(\Gamma) = \forall \mathbf{y}, \mathbf{x}_h(\overline{\exists \mathbf{x}_g(C(\mathbf{x}_g, \mathbf{y}, \Gamma) \cdot f(\mathbf{x}))} + \forall \mathbf{x}_g(\overline{C(\mathbf{x}_g, \mathbf{y}, \Gamma)} + f(\mathbf{x}))) \tag{15}$$

The computational effort in (15) can be reduced significantly taking into account decomposition properties of template (14). Its partition of the function support into $\mathbf{x}_g$ and $\mathbf{x}_h$ can be used to write $f(\mathbf{x})$ in the expanded form as [8]:

$$f(\mathbf{x}_g, \mathbf{x}_h) = \sum_{i=0}^{2^s - 1} m_i(\mathbf{x}_g) \cdot f_i(\mathbf{x}_h) \tag{16}$$

where $f_i(\mathbf{x}_h)$'s are the cofactors [3] of $f(\mathbf{x})$ with respect to $\mathbf{x}_g$. The $f_i(\mathbf{x}_h)$ cofactors can be arbitrarily complex functions of the $\mathbf{x}_h$ variables. However, their significance in the decomposition lies in the relation to each other rather than their individual dependence on $\mathbf{x}_h$. The dependence on $\mathbf{x}_h$ can, therefore, be abstracted away allowing us to replace cofactors by a set of variables $\mathrm{Z} \equiv [\zeta_i]$ such that the same variable is associated with the identical cofactors. Thus, for any value of *n*, we encode *n*-variable functions in terms of their *s* decomposition variables $\mathbf{x}_g$:

$$F(\mathbf{x}_g, \mathrm{Z}) = \sum_{i=0}^{r-1} M_i(\mathbf{x}_g) \cdot \zeta_i \tag{17}$$

Each of the $M_i(\mathbf{x}_g)$'s in the above equation denotes a largest subset of minterms whose cofactors are identical, thereby defining a factor. The factors are often referred to as *equivalence classes* of minterms [16], which are induced by the equality relation of their cofactors.

Example 3.3 Given function

$$f(a, b, c, d, e) = a\bar{b}d + a\bar{e} + adc + b\bar{e}d + \bar{b}c\bar{e}$$

we can write it in the form of (17) letting $\mathbf{x}_g$ to be a set of three variables $\{a, b, c\}$. The minterm space of these variables induces a set four distinct cofactors, which are listed below together with their factors $M_i(\mathbf{x}_g)$:

i	**Factor,** $M_i(a, b, c)$	**Cofactor,** $f_i(d, e)$
0	$\{\bar{a}\bar{b}\bar{c}\}$	0
1	$\{a\bar{b}\bar{c}, a\bar{b}c, abc\}$	$d + \bar{e}$
2	$\{\bar{a}b\bar{c}, \bar{a}bc\}$	$d\bar{e}$
3	$\{ab\bar{c}, \bar{a}\bar{b}c\}$	$\bar{e}$

Each of the distinct cofactors f_i from the above table can be replaced by a single variable ζ_i, yielding $F(\mathbf{x}_g, \mathrm{Z})$ for the function f:

$$F(a, b, c, \zeta_1, \zeta_2, \zeta_3, \zeta_4) = \bar{a}\bar{b}\bar{c} \cdot \zeta_0 + (a\bar{b}\bar{c} + a\bar{b}c + abc) \cdot \zeta_1 + (\bar{a}b\bar{c} + \bar{a}bc) \cdot \zeta_2 + (ab\bar{c} + \bar{a}\bar{b}c) \cdot \zeta_3$$

■

Substituting $F(\mathbf{x}_g, \mathrm{Z})$ for $f(\mathbf{x})$ in (15) we finally obtain a computational form for all feasible decompositions of $F(\mathbf{x}_g, \mathrm{Z})$ (and subsequently of $f(\mathbf{x})$):

$$G(\Gamma) = \forall \mathrm{Z} \forall \mathbf{y} (\overline{\exists \mathbf{x}_g (C(\mathbf{x}_g, \mathbf{y}, \Gamma) \cdot F(\mathbf{x}_g, \mathrm{Z}))} + \forall \mathbf{x}_g (\overline{C(\mathbf{x}_g, \mathbf{y}, \Gamma)} + F(\mathbf{x}_g, \mathrm{Z}))) \tag{18}$$

Together with the substitution of $F(\mathbf{x}_g, \mathrm{Z})$ this formula replaces abstraction of the $\mathbf{x}_h$ variables in (15) with the abstraction of Z variables. This transformation of (15) is made possible due to the following relation:

$$\forall \mathbf{x}_h \left(f(\mathbf{x}_g, \mathbf{x}_h) = \sum_{i=0}^{2^s - 1} M_i(x_g) \cdot f_i(x_h) \right) \Leftrightarrow \forall \mathrm{Z} \left(F(\mathbf{x}_g, \mathrm{Z}) = \sum_{i=0}^{r-1} M_i(x_g) \cdot \zeta_i \right)$$

This relation signifies importance of the *relative* values that the cofactors assume in the decomposition, and not their individual dependence on $\mathbf{x}_h$. The universal quantification of the Z variables in (18) ensures validity of its computed decomposition functions for all output values of the $f_i(\mathbf{x}_h)$ cofactors.

Example 3.4 We illustrate the application of (18) continuing with the decomposition of function $f(a, b, c, d, e)$ from Example 3.3. The space of two decomposition functions for its $\mathbf{x}_g$ variables $\{a, b, c\}$ can be encoded as:

$$\begin{aligned} C(\mathbf{x}_g, \mathbf{y}, \Gamma) = {} & (y_1 \equiv (\gamma_{01}\bar{a}\bar{b}\bar{c} + \gamma_{11}\bar{a}\bar{b}c + \gamma_{21}\bar{a}b\bar{c} + \gamma_{31}\bar{a}bc + \gamma_{41}a\bar{b}\bar{c} + \gamma_{51}a\bar{b}c + \gamma_{61}ab\bar{c} + \gamma_{71}abc)) \times \\ & (y_2 \equiv (\gamma_{02}\bar{a}\bar{b}\bar{c} + \gamma_{12}\bar{a}\bar{b}c + \gamma_{22}\bar{a}b\bar{c} + \gamma_{32}\bar{a}bc + \gamma_{42}a\bar{b}\bar{c} + \gamma_{52}a\bar{b}c + \gamma_{62}ab\bar{c} + \gamma_{72}abc)) \end{aligned}$$

Using this function $C(\mathbf{x}_g, \mathbf{y}, \Gamma)$, and function $F(\mathbf{x}_g, \mathrm{Z})$ described in the earlier example, the computation for the lower bound part of (18) has the form:

$$\exists \mathbf{x}_g (C(\mathbf{x}_g, \mathbf{y}, \Gamma) \cdot F(\mathbf{x}_g, \mathbf{y})) = (y_1 \equiv \gamma_{01}) \cdot (y_2 \equiv \gamma_{02}) \cdot z_0 + (y_1 \equiv \gamma_{11}) \cdot (y_2 \equiv \gamma_{12}) \cdot z_1$$

$$
\begin{aligned}
&+ (y_1 \equiv \gamma_{21}) \cdot (y_2 \equiv \gamma_{22}) \cdot z_2 + (y_1 \equiv \gamma_{31}) \cdot (y_2 \equiv \gamma_{32}) \cdot z_3 \\
&+ (y_1 \equiv \gamma_{41}) \cdot (y_2 \equiv \gamma_{42}) \cdot z_3 + (y_1 \equiv \gamma_{51}) \cdot (y_2 \equiv \gamma_{52}) \cdot z_1 \\
&+ (y_1 \equiv \gamma_{61}) \cdot (y_2 \equiv \gamma_{62}) \cdot z_2 + (y_1 \equiv \gamma_{71}) \cdot (y_2 \equiv \gamma_{72}) \cdot z_1
\end{aligned}
$$

Its upper bound part is computed analogously:

$$
\begin{aligned}
\forall \mathbf{x}_g(\overline{C(\mathbf{x}_g, \mathbf{y}, \Gamma)} + F(\mathbf{x}_g, \mathbf{y})) \;=\; & ((y_1 \oplus \gamma_{01}) + (y_2 \oplus \gamma_{02}) + z_0) \cdot ((y_1 \oplus \gamma_{11}) + (y_2 \oplus \gamma_{12}) + z_1) \\
& \times ((y_1 \oplus \gamma_{21}) + (y_2 \oplus \gamma_{22}) + z_2) \cdot ((y_1 \oplus \gamma_{31}) + (y_2 \oplus \gamma_{32}) + z_3) \\
& \times ((y_1 \equiv \gamma_{41}) + (y_2 \equiv \gamma_{42}) + z_3) \cdot ((y_1 \oplus \gamma_{52}) + (y_2 \oplus \gamma_{52}) + z_1) \\
& \times ((y_1 \oplus \gamma_{61}) + (y_2 \oplus \gamma_{62}) + z_2) \cdot ((y_1 \oplus \gamma_{71}) + (y_2 \oplus \gamma_{72}) + z_1)
\end{aligned}
$$

Applying lower and upper bound computations in (18) we have:

$$
\begin{aligned}
G(\Gamma) \;=\; & \forall \mathrm{Z}(((\bar{\gamma}_{01} + \bar{\gamma}_{02} + \bar{z}_0) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12} + \bar{z}_1) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22} + \bar{z}_2) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32} + \bar{z}_3) \cdot \\
& (\bar{\gamma}_{41} + \bar{\gamma}_{42} + \bar{z}_3) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52} + \bar{z}_1) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62} + \bar{z}_2) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72} + \bar{z}_1) + \\
& (\bar{\gamma}_{01} + \bar{\gamma}_{02} + z_0) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12} + z_1) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22} + z_2) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32} + z_3) \cdot \\
& (\bar{\gamma}_{41} + \bar{\gamma}_{42} + z_3) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52} + z_1) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62} + z_2) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72} + z_1)) \\
& \times ((\bar{\gamma}_{01} + \gamma_{02} + \bar{z}_0) \cdot (\bar{\gamma}_{11} + \gamma_{12} + \bar{z}_1) \cdot (\bar{\gamma}_{21} + \gamma_{22} + \bar{z}_2) \cdot (\bar{\gamma}_{31} + \gamma_{32} + \bar{z}_3) \cdot \\
& (\bar{\gamma}_{41} + \gamma_{42} + \bar{z}_3) \cdot (\bar{\gamma}_{52} + \gamma_{52} + \bar{z}_1) \cdot (\bar{\gamma}_{61} + \gamma_{62} + \bar{z}_2) \cdot (\bar{\gamma}_{71} + \gamma_{72} + \bar{z}_1) + \\
& (\bar{\gamma}_{01} + \gamma_{02} + z_0) \cdot (\bar{\gamma}_{11} + \gamma_{12} + z_1) \cdot (\bar{\gamma}_{21} + \gamma_{22} + z_2) \cdot (\bar{\gamma}_{31} + \gamma_{32} + z_3) \cdot \\
& (\bar{\gamma}_{41} + \gamma_{42} + z_3) \cdot (\bar{\gamma}_{52} + \gamma_{52} + z_1) \cdot (\bar{\gamma}_{61} + \gamma_{62} + z_2) \cdot (\bar{\gamma}_{71} + \gamma_{72} + z_1)) \\
& \times ((\gamma_{01} + \bar{\gamma}_{02} + \bar{z}_0) \cdot (\gamma_{11} + \bar{\gamma}_{12} + \bar{z}_1) \cdot (\gamma_{21} + \bar{\gamma}_{22} + \bar{z}_2) \cdot (\gamma_{31} + \bar{\gamma}_{32} + \bar{z}_3) \cdot \\
& (\gamma_{41} + \bar{\gamma}_{42} + \bar{z}_3) \cdot (\gamma_{52} + \bar{\gamma}_{52} + \bar{z}_1) \cdot (\gamma_{61} + \bar{\gamma}_{62} + \bar{z}_2) \cdot (\gamma_{71} + \bar{\gamma}_{72} + \bar{z}_1) + \\
& (\gamma_{01} + \bar{\gamma}_{02} + z_0) \cdot (\gamma_{11} + \bar{\gamma}_{12} + z_1) \cdot (\gamma_{21} + \bar{\gamma}_{22} + z_2) \cdot (\gamma_{31} + \bar{\gamma}_{32} + z_3) \cdot \\
& (\gamma_{41} + \bar{\gamma}_{42} + z_3) \cdot (\gamma_{52} + \bar{\gamma}_{52} + z_1) \cdot (\gamma_{61} + \bar{\gamma}_{62} + z_2) \cdot (\gamma_{71} + \bar{\gamma}_{72} + z_1)) \\
& \times ((\gamma_{01} + \gamma_{02} + \bar{z}_0) \cdot (\gamma_{11} + \gamma_{12} + \bar{z}_1) \cdot (\gamma_{21} + \gamma_{22} + \bar{z}_2) \cdot (\gamma_{31} + \gamma_{32} + \bar{z}_3) \cdot \\
& (\gamma_{41} + \gamma_{42} + \bar{z}_3) \cdot (\gamma_{52} + \gamma_{52} + \bar{z}_1) \cdot (\gamma_{61} + \gamma_{62} + \bar{z}_2) \cdot (\gamma_{71} + \gamma_{72} + \bar{z}_1) + \\
& (\gamma_{01} + \gamma_{02} + z_0) \cdot (\gamma_{11} + \gamma_{12} + z_1) \cdot (\gamma_{21} + \gamma_{22} + z_2) \cdot (\gamma_{31} + \gamma_{32} + z_3) \cdot \\
& (\gamma_{41} + \gamma_{42} + z_3) \cdot (\gamma_{52} + \gamma_{52} + z_1) \cdot (\gamma_{61} + \gamma_{62} + z_2) \cdot (\gamma_{71} + \gamma_{72} + z_1)))
\end{aligned}
$$

Abstraction of four Z variables gives 16 product terms, each of which is listed on a separate line below:

$$
\begin{aligned}
G(\Gamma) = & 1 \\
& \times ((\bar{\gamma}_{01} + \bar{\gamma}_{02}) + (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72})) \\
& \times ((\bar{\gamma}_{11} + \bar{\gamma}_{12}) + (\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72})) \\
& \times ((\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72}) + (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72})) \\
& \times ((\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) + (\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72})) \\
& \times ((\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) + (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72})) \\
& \times ((\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72}) + (\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42})) \\
& \times ((\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72}) + (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42})) \\
& \times ((\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) + (\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72})) \\
& \times ((\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) + (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72})) \\
& \times ((\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72}) + (\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62})) \\
& \times ((\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72}) + (\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62})) \\
& \times ((\bar{\gamma}_{21} + \bar{\gamma}_{22}) \cdot (\bar{\gamma}_{31} + \bar{\gamma}_{32}) \cdot (\bar{\gamma}_{41} + \bar{\gamma}_{42}) \cdot (\bar{\gamma}_{61} + \bar{\gamma}_{62}) + (\bar{\gamma}_{01} + \bar{\gamma}_{02}) \cdot (\bar{\gamma}_{11} + \bar{\gamma}_{12}) \cdot (\bar{\gamma}_{52} + \bar{\gamma}_{52}) \cdot (\bar{\gamma}_{71} + \bar{\gamma}_{72}))
\end{aligned}
$$

$$\times((\bar{\gamma}_{01}+\bar{\gamma}_{02})\cdot(\bar{\gamma}_{21}+\bar{\gamma}_{22})\cdot(\bar{\gamma}_{31}+\bar{\gamma}_{32})\cdot(\bar{\gamma}_{41}+\bar{\gamma}_{42})\cdot(\bar{\gamma}_{61}+\bar{\gamma}_{62})+(\bar{\gamma}_{11}+\bar{\gamma}_{12})\cdot(\bar{\gamma}_{52}+\bar{\gamma}_{52})\cdot(\bar{\gamma}_{71}+\bar{\gamma}_{72}))$$
$$\times((\bar{\gamma}_{11}+\bar{\gamma}_{12})\cdot(\bar{\gamma}_{21}+\bar{\gamma}_{22})\cdot(\bar{\gamma}_{31}+\bar{\gamma}_{32})\cdot(\bar{\gamma}_{41}+\bar{\gamma}_{42})\cdot(\bar{\gamma}_{52}+\bar{\gamma}_{52})\cdot(\bar{\gamma}_{61}+\bar{\gamma}_{62})\cdot(\bar{\gamma}_{71}+\bar{\gamma}_{72})+(\bar{\gamma}_{01}+\bar{\gamma}_{02}))$$
$$\times 1$$

This function has total of 24 ON-set minterms, each corresponding to a pair of decomposition functions. Assuming order-invariance of the decomposition functions, and invariance of their complements the number of solutions reduces to three:

Solution A:	**Solution B:**	**Solution C:**
$g_1 = a\bar{b}+\bar{a}b+bc$	$g_1 = a\bar{b}+\bar{a}b+bc$	$g_1 = a+\bar{b}c$
$g_2 = \bar{a}b+b\bar{c}+\bar{a}c$	$g_2 = a+\bar{b}c$	$g_2 = \bar{a}b+b\bar{c}+\bar{a}c$

■

It is possible to simplify the process of instantiating computation in (18) observing that terms corresponding to the lower and upper bounds in the formula can be simplified, and that they also have virtually identical structure. The first step of simplification multiplies out two functions corresponding to the interval's lower bound. Let vector γ_i denote the ith row of variables in the $\Gamma \equiv [\gamma_{ij}]$ matrix, and let function $\sigma(i)$ identify the index of a variable in Z corresponding to a cofactor $f_i(\mathbf{x}_h)$. We then have:

$$\begin{aligned} C(\mathbf{x}_g,\mathbf{y},\Gamma)\cdot F(\mathbf{x}_g,\mathrm{Z}) &= \left(\prod_{j=1}^{k}\left[y_j \equiv \sum_{i=0}^{2^n-1}\gamma_{ij}\cdot m_i(\mathbf{x})\right]\right)\cdot\left(\sum_{i=0}^{r-1}M_i(\mathbf{x}_g)\cdot\zeta_i\right) \\ &= \left(\sum_{i=0}^{2^s-1}m_i(\mathbf{x}_g)\cdot(\mathbf{y}\equiv\gamma_i)\right)\cdot\left(\sum_{i=0}^{2^s-1}m_i(\mathbf{x}_g)\cdot\zeta_{\sigma(i)}\right) \\ &= \sum_{i=0}^{2^s-1}m_i(\mathbf{x}_g)\cdot(\mathbf{y}\equiv\gamma_i)\cdot\zeta_{\sigma(i)} \end{aligned} \tag{19}$$

We similarly rewrite the upper bound part of (18):

$$\overline{C(\mathbf{x}_g,\mathbf{y},\Gamma)}+F(\mathbf{x}_g,\mathrm{Z}) = \overline{C(\mathbf{x}_g,\mathbf{y},\Gamma)\cdot\overline{F(\mathbf{x}_g,\mathrm{Z})}} = \overline{\sum_{i=0}^{2^s-1}m_i(\mathbf{x}_g)\cdot(\mathbf{y}\equiv\gamma_i)\cdot\bar{\zeta}_{\sigma(i)}} \tag{20}$$

Substituting results of (19) and (20) into (18) we have:

$$G(\Gamma) = \forall\mathrm{Z}\forall\mathbf{y}\left(\overline{\exists\mathbf{x}_g\left(\sum_{i=0}^{2^s-1}m_i(\mathbf{x}_g)\cdot(\mathbf{y}\equiv\gamma_i)\cdot\zeta_{\sigma(i)}\right)}+\overline{\exists\mathbf{x}_g\left(\sum_{i=0}^{2^s-1}m_i(\mathbf{x}_g)\cdot(\mathbf{y}\equiv\gamma_i)\cdot\bar{\zeta}_{\sigma(i)}\right)}\right) \tag{21}$$

The two summands in the above formula are identical with the exception of complemented Z variables, suggesting that during computation of $G(\Gamma)$ one summand can be constructed from the other by simple complementation of Z variables.

C. Support constraints in composition function

Introducing a fan-in constraint on the decomposition functions enabled us to define decomposition in terms of the computationally efficient, yet very general, template (14). The generality of this template however, also offers a great variety of the decomposition patterns which become available during the synthesis process. In addition to the fan-in bound s we can classify these decomposition patterns according to the number of decomposition functions t used during a decomposition step. Such s-to-t classification of the decomposition patterns provides us with a simple estimation of a decomposition quality, and links it to the structure of function.

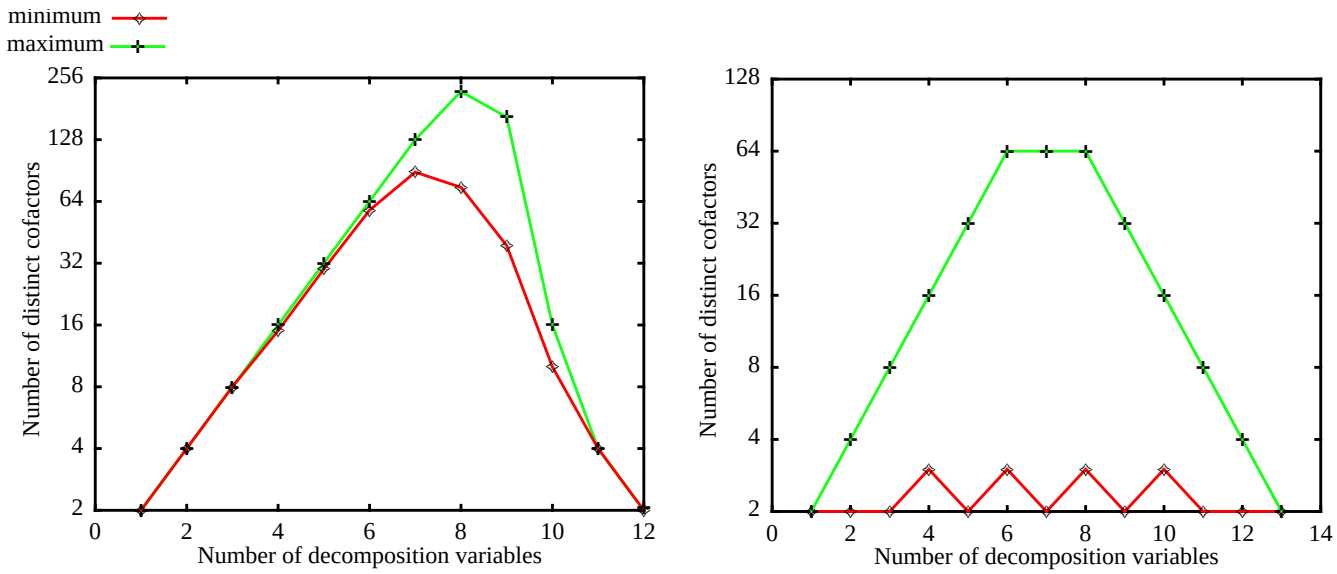
The decomposition template (14) establishes a strong relation between the number of decomposition variables s and the number of decomposition functions t, as their relative values significantly determine flexibility in selecting the composition function h. Indeed, letting t to be smaller than s may imply non-existence of h, and therefore non-existence decomposition. On the other hand, letting t to be larger than s may present us with a vast number of choices for h, making it difficult to find a good composition function which improves synthesis quality.

According to the decomposition template (14) the s-to-t decomposition pattern determines the difference in the support sizes between functions f and the composition function h. Whenever $s > t$, the decomposition is support-reducing as it implies that the number of variables in h is less than the number of variables f. Similarly, the decomposition is support-maintaining or support-increasing whenever, respectively, $s = t$ or $s < t$. In this work we are primarily interested in the support-reducing decomposition, and in the enabling it conditions. This type of decomposition defines a particularly attractive class of circuits whose width for each of its outputs decreases at the successive levels of logic.

Although the support-reducing decomposition under a fan-in constraint produces very attractive circuits, it also places a restriction on the function classes for which such decomposition pattern is feasible. Depending on a function structure, and the s-to-t parameters of the decomposition pattern, template (14) may have no feasible decomposition. In general the existence of the s-to-t decomposition requires that the number of distinct cofactors induced by the minterm space of the decomposition variables is $\leq 2^t$. Otherwise, it is impossible to find t decomposition functions whose 2^t products $\dot{g}_1(\mathbf{x}_g) \cdot \ldots \cdot \dot{g}_t(\mathbf{x}_g)$ [1] do not contain minterms from two distinct factors $M_i(\mathbf{x}_g)$, making these minterms indistinguishable in the decomposition. The cofactor count argument has been used extensively in the classical theory for disjoint decomposition relying on the notion of column multiplicity in partition tables [8].

The support-reducing decomposition implied by the s-to-t pattern translates into a requirement on $f(\mathbf{x})$: for a support reduction of one, i.e. $t = s - 1$, the number of *f*'s distinct cofactors must be $\leq 2^{s-1}$. The following two examples illustrate how the relation between cofactors of f impacts existence of decomposition.

1. The dot above functions denotes their fixed phase, either complemented or not complemented.



a) the p_7 output function of a 6-bit multiplier

b) the s_5 output function of an adder

Fig. 3. Dependence of the distinct cofactor counts on the number of decompositions variables

Example 3.5 We use function $f(a, b, c, d, e) = a\bar{b}d + a\bar{e} + adc + b\bar{e}d + \bar{b}c\bar{e}$ from Example 3.3 to illustrate how existence of decomposition depends on the choice of decomposition variables. The three decomposition variables in the earlier example induce four distinct cofactors 0, $d + \bar{e}$, $d\bar{e}$, and $\bar{e}$, implying existence of 3-to-2 reduction. This fact re-confirmed in Example 3.4 applying our symbolic computation of decomposition.

On the other hand, selecting a, b, e as decomposition variables $\mathbf{x}_g$ for the function f we can quickly determine the non-existence of 3-to-2 reduction based on the distinct cofactor counts for these variables. Indeed, function f has total of 5 distinct cofactors with respect to these variables – 0, 1, d, e, and de, prohibiting support-reducing decomposition. ■

In general the number of distinct cofactors induced by the decomposition variables can vary greatly depending on the structure of a given function. For a set of non-vacuous in f decomposition variables the number of distinct cofactors is at least 2. There is also an upper bound on the maximum number of distinct cofactors for any given set of decomposition variables. For an arbitrary function of n variables the count of its distinct cofactors induced by s decomposition variables must satisfy the following two conditions:

- it is bounded by 2^s, the total number of [need not be distinct] cofactors that can be possibly induced by s decomposition variables
- it is bounded by $2^{2^{n-s}}$, the total number of distinct cofactor functions that can be possibly created from $n - s$ non-decomposition variables

As follows from the above two constraints, the potential number of distinct cofactors increases exponentially in s, while being bounded by $2^{2^{n-s}}$. This relation is used below as a starting point to derive a simpler form relating s to n:

$$(2^s \le 2^{2^{n-s}}) \Leftrightarrow (s \cdot 2^s \le 2^n) \Leftrightarrow (\log(s \cdot 2^s) \le n) \Leftrightarrow (\log(s) + s \le n)$$

Whenever s and n satisfy $\log(s) + s \le n$ we can find a function whose s decomposition variables induce 2^s distinct cofactors. Thus, the number of distinct cofactors for the s decomposition variables can theoretically be between 2 and 2^s, even for the large s approaching n.

The theoretical bound on the maximum number of distinct cofactors implies that for some large sets of decomposition variables we may not be able to achieve support-reducing threshold requiring the number of distinct cofactors to be 2^{s-1}. Among functions for which no small set of support-reducing decomposition variables exists are functions of a multiplier circuit. To illustrate this fact Fig. 3-a plots minimum and maximum numbers of distinct cofactors among all possible sets of s $(1 \le s \le 12)$ of decomposition variables for an 8-th bit product function in a 6-bit multiplier. In this figure the smallest set of decomposition variables for which support-reducing decomposition exists has 8 variables inducing 75 distinct cofactors. (Indeed the $75 \le 2^{8-1}$ relation enables support reduction.)

Despite the exponential bound on the number of distinct cofactors, our analysis of the MCNC benchmarks [24] shows that the vast majority of the functions do enable support-reducing decomposition for small values of s, usually less than 5. Existence of a support-reducing decomposition in a typical function from this suit of benchmarks is very sensitive to a given set of decomposition variables, necessitating their careful selection during decomposition process. In Fig. 3-b we illustrate this fact for a sixth sum bit function of an adder. Analogous to Fig. 3-a, the two sets of data points plotted in the figure are for the minimum and the maximum numbers of distinct cofactors induced by the sets of s $(1 \le s \le 12)$ decomposition variables. The gap between these data points is much wider though than in the case of a multiplier function. In fact, as Fig. 3-b for the adder function suggests for any s there is a set of decomposition variables of this size inducing at most 3 cofactors. These small numbers of the cofactor counts enable support-reducing decomposition for any $s \ge 2$.

IV. Symmetric Libraries Construction

Functions that satisfy requirement of support-reducing decomposition under a practical fan-in constraint include the class of symmetric functions [9, 10], namely functions that remain invariant under certain permutations of their inputs. In this section we focus on the library computation required to achieve decomposition function classes of this type.

A. Class-universal decomposition primitives

Functions that are symmetric in the s decomposition variables can have at most $s+1$ distinct cofactors implying the existence of the decomposition in (14) whenever s is a solution to the relation $s+1 \le 2^{s-1}$, namely $s \ge 3$. Denoting such functions by f_s, we can express them by the following $(s+1)$-term sum:

$$f_s(\mathbf{x}_g, \mathbf{x}_h) = \sum_{i=0}^{s} S_i(\mathbf{x}_g) \cdot \zeta_i(\mathbf{x}_h) \tag{22}$$

where S_i represents a class of equivalent minterms induced by the symmetry relation between variables in $\mathbf{x}_g$; it is a totally symmetric function which is equal to 1 for those minterms on $\mathbf{x}_g$ whose weight (number of positive literals) is equal to i. The $\zeta_i(\mathbf{x}_h)$ functions in (22) represent cofactors of f_s with respect to $\mathbf{x}_g$.

Example 4.1 Suppose we would like to decompose function

$$f = \bar{a}\bar{b}\bar{c}de + \bar{a}\bar{b}cd + \bar{a}b\bar{c}d + \bar{a}bc\bar{d}e + a\bar{b}\bar{c}d + a\bar{b}c\bar{d}e + ab\bar{c}\bar{d}e + abc\bar{d}$$

with respect to mutually symmetric decomposition variables a, b, c. The equivalent minterms of the decomposition variables induced by symmetry relation along with their cofactors are given in the table below:

i	**Factor,** $S_i(a, b, c)$	**Cofactor,** $\xi_i(d, e)$
0	$\{\bar{a}\bar{b}\bar{c}\}$	de
1	$\{\bar{a}\bar{b}c, \bar{a}b\bar{c}, a\bar{b}\bar{c}\}$	d
2	$\{\bar{a}bc, a\bar{b}c, ab\bar{c}\}$	$\bar{d}e$
3	$\{abc\}$	$\bar{d}$

According to the table we can write f in the factored form as

$$f = (\bar{a}\bar{b}\bar{c})de + (\bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c})d + (\bar{a}bc + a\bar{b}c + ab\bar{c})\bar{d}e + (abc)\bar{d}$$

There is total of four distinct cofactors, and therefore there is a 3-to-2 support-reducing decomposition. ■

Abstracting away the dependence on $\mathbf{x}_h$ we replace cofactors in (22) by a set of independent variables $\mathrm{Z} \equiv [\zeta_i]$ that, for any value of *n*, encode *n*-variable functions in terms of their symmetric core with respect to the s $\mathbf{x}_g$ variables, yielding:

$$F_s(\mathbf{x}_g, \mathrm{Z}) = \sum_{i=0}^{s} S_i(\mathbf{x}_g) \cdot \zeta_i \tag{23}$$

Instantiating (18) with $F_s(\mathbf{x}_g, \mathrm{Z})$ we obtain a computational form for all feasible support-reducing decompositions of *n*-variable functions that are symmetric in *s* or fewer variables:

$$G(\Gamma) = \forall \mathrm{Z} \forall \mathbf{y} (\overline{\exists \mathbf{x}_g (C(\mathbf{x}_g, \mathbf{y}, \Gamma) \cdot F_s(\mathbf{x}_g, \mathrm{Z}))} + \forall \mathbf{x} (\overline{C(\mathbf{x}_g, \mathbf{y}, \Gamma)} + F_s(\mathbf{x}_g, \mathrm{Z}))) \tag{24}$$

The above computational form provides us with the decomposition solutions which hold universally for any function represented in (23). Each of these solutions provide us with a set of decomposition primitives which serve as a pre-computed symmetric library.

As stated, equation (24) yields a non-empty set of solutions, $G(\Gamma) \neq 0$, for the function classes when the number of decomposition functions t encoded by $C(\mathbf{x}_g, \mathbf{y}, \Gamma)$ is $\geq \log(s+1)$. This effectively implies that the universal support-reducing decomposition primitives exist only when $s \geq 3$ in (23).

B. Subclass-universal decomposition primitives

To perform computation of the sets of decomposition primitives for the class of functions when support-reducing decomposition in (24) yields $G(\Gamma) = 0$ we compute decomposition primitives for the subclasses of $F_s(\mathbf{x}_g, \mathrm{Z})$. These subclasses of functions are defined according to the weakest relations between ζ_i variables which make decomposition feasible. A symmetric library for the decomposable functions is then constructed as a union of representative libraries taken from each subclass.

To derive libraries for the case when (24) yields the empty set, we first encode the universe of possible equality relations between Z variables in terms of the following function:

$$\cup_F(\mathrm{Z}, A) = \prod_{i<j}^{s} (\zeta_i \equiv \zeta_j + \bar{\alpha}_{ij}) \tag{25}$$

An assignment to the A variables in this function induces an equality relation between variables Z. The non-trivial equality relation between variables in Z corresponds to a reduction of distinct cofactors in (22), and therefore reduced decomposition constraints. By restricting computation of (24) to the conditions which satisfy such an induced relation we can relax computational form (24), and provide decomposition solutions for subclass of $F_s(\mathbf{x}_g, \mathrm{Z})$. Such decomposition solutions can be computed simultaneously for all assignment to the A variables by means of the following extension to (24):

$$G(\Gamma, A) = \forall \mathrm{Z} \forall \mathbf{y}_g (\overline{\exists \mathbf{x}_g (C(\mathbf{x}_g, \mathbf{y}_g, \Gamma) \cdot F_s(\mathbf{x}_g, \mathrm{Z}))} + \forall \mathbf{x} (\overline{C(\mathbf{x}_g, \mathbf{y}_g, \Gamma)} + F_s(\mathbf{x}_g, \mathrm{Z})) + \overline{\cup_F(\mathrm{Z}, A)}) \tag{26}$$

For each of the assignments to A function $G(\Gamma, A)$ now gives sets of feasible decomposition functions. Note that a 0 assignment to all of the A variables in the above formula induces a set completely independent variables Z, reducing it to (24).

Table 1: Characteristics of cell libraries necessary for support-reducing symmetric decomposition

Symmetry type		Possible *s*-to-*t* reduction	Cell library characteristics		
s	**Max #distinct cofactors**		**#Libraries**	**Size**	**Example library (composed of *s*-input symmetric primitives)**
2	2	2-to-1	1	3	
3	2	3-to-1	1	7	 S_1 S_2 $S_{0,3}$ $MAJ3$
3	4	3-to-2	3	2	 $MAJ3$
4	2	4-to-1	1	15	$S_0, \ldots, S_4 S_{0,1}, \ldots, S_{0,4}$ $S_{1,2}, \ldots, S_{1,4}$ $S_{2,3}$ $S_{2,4}$ $S_{3,4}$
4	4	4-to-2	15	8	$S_{0,1}$ $S_{0,3}$ $S_{0,4} S_{1,2}, \ldots, S_{1,4} S_{2,3}$ $S_{2,4}$
4	5	4-to-3	140	3	  $MAJ4$
5	2	5-to-1	1	31	$S_0, \ldots, S_5$ $S_{0,1}, \ldots, S_{0,5}$ $S_{1,2}, \ldots, S_{1,5}$ $S_{2,3}, \ldots, S_{2,5}$ $S_{3,4}$ $S_{3,5}$ $S_{4,5}$ $S_{0,1,2}, \ldots, S_{0,1,5}$ $S_{0,2,3}, \ldots, S_{0,2,5}$ $S_{0,3,4}$ $S_{0,3,5}$ $S_{0,4,5}$
5	4	5-to-2	105	22	$S_{0,1}, \ldots, S_{0,3}$ $S_{0,5}$ $S_{1,2}, \ldots, S_{1,4}$ $S_{2,3}, \ldots, S_{2,5}$ $S_{3,4}$ $S_{3,5}$ $S_{4,5}$ $S_{0,1,2}, \ldots, S_{0,1,5}$ $S_{0,2,3}$ $S_{0,2,5}$ $S_{0,3,4}$ $S_{0,3,5}$ $S_{0,4,5}$
5	6	5-to-3	420	3	$S_{0,3}$  $MAJ5$
5	6	5-to-4	14385	4	$S_{0,3}$  $MAJ5$ $ANY5$

We must observe however, that some of the assignments to the A variables are superfluous in the encoding $U_F(\mathrm{Z}, A)$ – they are subsumed by other assignments due to the transitive property of the equality relation. We discard such solutions defining a universe of assignments for which the transitive property of the equality is satisfied:

$$T(A) = \prod_{i<k<j} (\alpha_{ik}\alpha_{kj} \rightarrow \alpha_{ij}) \tag{27}$$

The above equation provides us with a characteristic function for all assignments valid under the transitive relation between Z variables. We can therefore restrict our analysis to $G(\Gamma, A) \cdot T(A)$.

To reduce further the number of assignments needed to be considered another observation analogous to the transitive relation between ζ_i variables can be made. It is based on the fact that some of the assignments derive "weaker" equality relation constraints than the other. Formally, an assignment to variables A^* is *weaker* than an assignment B^* if and only if relation $\alpha_{ij} \leq \beta_{ij}$ holds componentwise between values of these assignments, and there exists at least one pair of components such that $\alpha_{ij} \neq \beta_{ij}$. Symbolically we define such relation as:

$$LT(A, B) = \prod_{i<j} (\alpha_{ij} \leq \beta_{ij}) \cdot \overline{\prod_{i<j} (\alpha_{ij} \equiv \beta_{ij})} \tag{28}$$

The above equation encodes the *less-than* relation between two domains A and B, which can be used to extract the subset of weakest assignments from a given set $E_F(A) = \exists\Gamma(G(\Gamma, A))$:

$$E_W(B) = \forall A(E_F((A) \rightarrow E_F(B) \cdot \overline{LT(A, B)})) \tag{29}$$

$E_W(B)$ encodes in terms of B variables all weakest functional structures for which decomposition exists. It can be easily brought to the form which depends on the A variables, $E_W(A)$. Constraining $G(\Gamma, A)$ with $E_W(A)$ we have all feasible decompositions. For each of the assignments to A, such that $E_W(A) = 1$, we now have a set of decompositions. These sets provide decomposition primitives which can be used to construct libraries covering all function subclasses.

C. Computed libraries

Table 1 summarizes the results of solving (24) for all *s*-to-*t* support-reducing decompositions where $s \leq 5$. The first two columns in the table characterize the symmetry of the function being decomposed in terms of the number of symmetric decomposition variables s and the maximum number of distinct cofactors with respect to those variables. Column 3 indicates the corresponding achievable support reduction. The remaining columns characterize the cell libraries required to realize these decompositions: column 4 is the number of possible minimal-size libraries, column 5 is the number of required *s*-input cells in each library, and column 6 shows a sample library. The counts in column 4 include only libraries of symmetric cells and assume that libraries consisting of the same cells up to complementation of their functions are indistinguishable. Some of the library cells listed in column 6 are expressed

using the S_a notation, where a is a set of integers identifying the weights of the minterms for which the function evaluates to 1.

We can make the following observations about the results in Table 1:

- The libraries in this table represent pre-computed decomposition primitives that map a structural property of the functions being decomposed (symmetry) into a structural property of the circuit implementation (width reduction). Indeed, the complexity of the function being synthesized is reflected directly in the implementation: the support of the most complex symmetric functions (with s+1 distinct cofactors) can only be reduced by one, whereas the support of the least complex symmetric functions (with 2 distinct cofactors) can be maximally reduced to 1.
- Whenever $s \geq 3$ and $t \geq \lceil \log(s+1) \rceil$, the libraries for s-to-t reduction are universal in the sense that they will yield the desired decomposition for all functions that are symmetric in s or more variables; they are "functionally complete" for the class of symmetric functions. For instance, there are exactly three universal libraries that enable 3-to-2 decomposition, one of which is shown in the table. The other two are $\{XOR3, SAME3\}$ and $\{MAJ3, SAME3\}$, where $SAME3(x_1, x_2, x_3) = x_1x_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3$. Some of the decomposition functions in the s-to-t pattern may become redundant however. The 5-to-4 pattern is an example of such redundancy, where the fourth function (denoted by ANY5 in the library) can be defined arbitrarily.
- For a given s the number of libraries decreases and their size (number of cells) increases with stronger support reduction (smaller t.) For $t = 1$, the libraries become unique, up to complementation, and contain $2^s - 1$ cells.
- The libraries in this table can be extended to handle the class of functions that are invariant with respect to both the permutation and complementation of their inputs [12] by placing corresponding inversions on the respective primitive inputs.

When simple symmetries do not exist other structural attributes of a function might be present. In particular, a multiplexer-like symmetry of the form

$$f(\bar{x}_1, x_2, x_3, \ldots, x_{2m}, x_{2m+1}, \ldots, x_{n-1}, x_n) = f(x_1, x_3, x_2, \ldots, x_{2m+1}, x_{2m}, \ldots, x_{n-1}, x_n) \quad (30)$$

often arises in datapath circuits. The invariance described by this relation swaps two ordered groups of variables of size m while complementing one variable outside these groups:

$$\{\langle \bar{x}_1, x_2, x_4, \ldots, x_{2m}\rangle, \langle x_1, x_3, x_5, \ldots, x_{2m+1}\rangle\} \quad (31)$$

In benchmark circuits the most common functions of this type can be described as a sum:

$$f_M(\mathbf{x}_g, \mathbf{x}_h) = \sum_{i=0}^{2^m-1} [\bar{x}_1 \cdot m_i(\mathbf{x}_g^L) + x_1 \cdot m_i(\mathbf{x}_g^R)] \cdot \zeta_i(\mathbf{x}_h) \quad (32)$$

where $\mathbf{x}_g$ is composed of x_1, $\mathbf{x}_g^L = \langle x_2, x_4, \ldots, x_{2m} \rangle$ and $\mathbf{x}_g^R = \langle x_3, x_5, \ldots, x_{2m+1} \rangle$.

Using the decomposition template (32) we can now pre-compute corresponding 3-to-2 libraries using a procedure similar to the one we used in the case of simple symmetries. Specifically, we first express all functions admitting template (32) using a suitable encoding function F_M in which the dependence on the non-decomposition variables is eliminated through the introduction of a set of binary encoding coefficients $\mathrm{Z} \equiv [\zeta_i]$:

$$F_M(x_1, x_2, \ldots, x_{2m+1}, \mathrm{Z}) = \sum_{i=0}^{2^m-1} [\bar{x}_1 \cdot m_i(x_2, x_4, \ldots, x_{2m}) + x_1 \cdot m_i(x_3, x_5, \ldots, x_{2m+1})] \cdot \zeta_i$$

Next, noting that this decomposition template is independent of the "datapath width" *m*, we reduce it to a width of one by choosing a single representative from each group of "left" and "right" variables $\mathbf{x}_g^L$ and $\mathbf{x}_g^R$. Without loss of generality, choosing x_2 to represent $\mathbf{x}_g^L$ and x_3 to represent $\mathbf{x}_g^R$ we obtain:

$$F_M(x_1, x_2, x_3, \mathrm{Z}) = \sum_{i=0}^{1} \bar{x}_1 \cdot m_i(x_2) \cdot \zeta_i + \sum_{i=0}^{1} x_1 \cdot m_i(x_3) \cdot \zeta_{i+2}$$

which can be rewritten as $F_M(x_1, x_2, x_3, \mathrm{Z}) = \bar{x}_1\bar{x}_2 \cdot \zeta_0 + \bar{x}_1 x_2 \cdot \zeta_1 + x_1\bar{x}_3 \cdot \zeta_2 + x_1 x_3 \cdot \zeta_3$. Substituting $F_M(x_1, x_2, x_3, \mathrm{Z})$ in (12) we obtain a computational form similar to (24). The solution for this 3-to-2 decomposition yields three possible 2-output modules: $\{\bar{x}_1 x_2 + x_1\bar{x}_3, x_1\}$, $\{\bar{x}_1 x_2 + x_1\bar{x}_3, \bar{x}_1 x_2 + x_1 x_3\}$ and $\{\bar{x}_1 x_2 + x_1 x_3, x_1\}$. As follows from the last module, we can accommodate this type of decomposition by means of a 2-to-1 multiplexer and a wire. (Note that whenever $m = 1$ template (32) admits 3-to-1 reduction using just a 2-to-1 multiplexer.) It is interesting to note that restricting our decomposition pattern to three decomposition variables as we did above allows us to avoid computing an exact symmetry structure of the form (31) to identify x_1, x_2 and x_3. These three decomposition variables can be identified with the help of quantifying out control signal x_1 from $f_M(\mathbf{x})$ which gives a function symmetric in x_2 and x_3.

V. Conclusions and Future Work

For the functions that are symmetric in some inputs we have pre-computed libraries required for their decomposition patterns. These decomposition patterns capture the structural properties of a function and reflect them in the implementation structure. We are currently studying other forms of structure-aware functional decomposition which might yield "natural" decomposition patterns for the efficient synthesis of control logic. Among these decomposition patterns we are studying implications of functional structure on a library required by the composition functions to improve synthesis quality.

References

[1] R. L. Ashenhurst. The decomposition of switching functions. *Ann. Computation Lab.*, Harvard University, vol. 29, pages 74-116, 1959.

[2] R. K. Brayton and F. Somenzi. Boolean relations and the incomplete specification of logic networks. In *VLSI'89*, August 1989.

[3] R. K. Brayton, J. D. Cohen *et al.* Fast recursive boolean function manipulation. In *Proc. IEEE Int. Symp. Circ. and Syst.*, pages 58-62, May 1982

[4] R. K. Brayton and C. McMullen. The decomposition and factorization of Boolean expressions. In *Proc. IEEE Int. Symp. Circ. and Syst.*, pages 29-54, May 1982.

[5] F. M. Brown. *Boolean Reasoning*. Kluwer Academic Publishers, Boston, 1990.

[6] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE TC*, C-35(6):677–691, August 1986.

[7] E. Cerny. An approach to unified methodology of combinational switching circuits. *IEEE TC*, 27(8), 1977.

[8] A. Curtis. *New Approach to the Design of Switching Circuits.* Van Nostrand, Princeton, NJ, *1962.*

[9] C. R. Edwards and S. L. Hurst. A digital synthesis procedure under function symmetries and mapping methods. *IEEE TC*, C-27:985–997, 1978.

[10] B.-G. Kim and D. L. Dietmeyer. Multilevel logic synthesis of symmetric switching functions. *IEEE TCAD IC*, 10(4):436–446, April 1991.

[11] V. N. Kravets and K. A. Sakallah. Generalized symmetries in Boolean functions. Technical Report CSE-TR-420-00, February 2000.

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

[13] Y. T. Lai, M. Pedram, and Sarma B. K. Vrudhula. BDD based decomposition of logic functions with application to FPGA synthesis. In *Proc. 30th DAC*, pages 642–647, June 1993.

[14] E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness. Logic decomposition during technology mapping. In *Proc. ICCAD*, pages 264–271, November 1995.

[15] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli. Optimum functional decompositions using encoding. In *Proc. 31st DAC*, pages 408–414, June 1994.

[16] J. P. Roth and R. Karp. Minimization over boolean graphs. *IBM J. Res. and Develop.*, 6(2):227–238, April 1962.

[17] K. A. Sakallah. Functional abstraction and partial specification of Boolean functions. Technical Report CSE-TR-255-95, University of Michigan, August 1995.

[18] H. Savoj. *Don't cares in Multi-Level network Optimization*. Ph.D. thesis, University of California, Berkeley, 1992

[19] H. Sawada, T. Suyama, and A. Nagoya. Logic synthesis for look-up table based FPGAs using functional decomposition and support minimization. In *Proc. ICCAD*, pages 353–358, November 1995.

[20] C. Scholl, Multi-output functional decomposition with exploration of don't cares. In *Proc. DATE*, pp. 755-759, February 1998.

[21] C. Scholl, D. Moller, P. Molitor, and R. Drechsler. BDD minimization using symmetries. *IEEE TCAD IC*, 18(2):81–100, February 1999.

[22] E. M. Sentovich. SIS: A system for sequential circuit synthesis. Technical Report UCB/ERL M92/41, UC Berkeley, May 1992.

[23] B. Wurth, K. Eckl, and K. Antreich. Functional multiple-output decomposition: theory and an implicit algorithm. In *Proc. 32nd DAC,* pages 54–59, June 1995.

[24] S. Yang. *Logic synthesis and optimization benchmarks user guide – ver. 3.0*. MCNC, Res. Triangle Park, NC, Jan. 1991