Vectorial Bi-Decompositions for Lattices of Boolean Functions

Bernd Steinbach
Freiberg University of Mining and Technology
Institute of Computer Science
D-09596 Freiberg, Germany
email: steinb@informatik.tu-freiberg.de

Christian Posthoff
The University of The West Indies
St. Augustine Campus
Trinidad & Tobago
email: christian@posthoff.de

Abstract

This paper generalizes the design approach of vectorial bi-decompositions for lattices of Boolean functions. It is shown how it can be found whether a lattice contains at least one function for which an AND-, an OR-, or an EXOR-bi-decomposition exists. Generalized lattices from which the decomposition functions can be chosen are specified. The rank of the independency matrix of the decomposition functions is increased by 1 if the functions of the decomposed lattice depend on the chosen direction of change. Weak bi-decompositions complete strong bi-decompositions, but lead to unbalanced circuits with longer paths. Vectorial bi-decompositions lead to balanced circuits and avoid certain weak bi-decompositions.

1 Introduction

Bi-decomposition is an approach that utilizes the properties of the given Boolean function to design circuit structures of a small area, low power consumption, and low delay [1]. The decomposition functions of the strong bi-decomposition are simpler than the given function because they depend on fewer variables.

Unfortunately, there are functions for which no strong bi-decomposition exists. Le [4] bridged this gap by means of the weak bi-decomposition. A simplified proof of the completeness of the strong and weak bi-decomposition is given in [6]. An implementation that reuses already decomposed blocks outperforms other synthesis approaches [5]. Drawbacks of the weak bi-decompositions are circuits having a large difference between the shortest and the longest path (unbalanced circuits) and increased longest paths.

All bi-decomposition approaches mentioned above utilize the Boolean differential calculus to find optimal bi-decompositions. There are several other approaches of bi-decomposition. In [7] a method for disjoint bi-decompositions with an extension to non-disjoint bi-decompositions for a single common variable are suggested. A graph based approach for bi-decompositions was suggested in [3]. Unfortunately the used benchmarks in [5] and [3] overlap only partially. One common benchmark is t481 where our approach from [5] outperforms the graph based approach from [3], e.g., in the number of gates (17/25). Recently, semi-tensor products of matrices were suggested for bi-decompositions of Boolean and multi-valued functions [2]. However, this paper does not contain experimental results of benchmark circuits.

Hence, we extend in this paper the knowledge of the bi-decomposition based on the Boolean differential calculus. Vectorial bi-decompositions were recently suggested as supplement to strong


and weak bi-decompositions [11]. The decomposition functions of these bi-decompositions are simpler than the given function because they are independent of the simultaneous change of several variables. Vectorial bi-decompositions can exist for functions without any strong bi-decomposition. Benefits of the vectorial bi-decomposition are their contribution to balanced circuits and the increased number of decomposition possibilities in comparison to the strong bi-decomposition.

Using lattices of Boolean functions in the synthesis process, the synthesized function can be chosen from a larger set of functions, and the possibilities of strong bi-decompositions are increased. Incompletely specified function were traditionally used as a source of a lattice of Boolean functions. The ON-set-function \( f_o(x) \) and the OFF-set-function \( f_r(x) \) are the preferred mark function of these lattices. More general lattices were recently found [9]. The introduction of derivative operations for lattices of Boolean functions [8] facilitates the application of these more general lattices in circuit design.

Lattices of Boolean functions are used so far only for the strong and weak bi-decompositions. The introduction of generalized lattices [9, 14] and the specification of derivative operations [8] of such lattices established the basis to find and generate vectorial bi-decompositions for generalized lattices of Boolean functions. This generalization is the main aim of this paper.

The rest of the paper is organized as follows. Making this paper self-contained we repeat in Section 2 the definitions of the used derivative operations of the Boolean Differential Calculus [8, 12, 13] and summarize in Section 3 Definitions and Algorithms needed to describe the vectorial bi-decomposition for generalized lattices of Boolean functions. Section 4 contains the new contributions of vectorial bi-decompositions for a generalized lattice. Section 5 applies the vectorial bi-decompositions to a lattice of 32 functions and compares the fundamental circuit structure with other synthesis approaches.

## 2 Vectorial and Single Derivative Operations

**Definition 1 (Vectorial Derivative Operations)** Let \( x_0 = (x_1, \ldots, x_k), x_1 = x \setminus x_0 \) be two disjoint sets of Boolean variables, \( 1 \leq k \leq n \), and \( f(x_0, x_1) = f(x_1) = f(x) \) be a Boolean function of \( n \) variables, then

\[
\frac{\partial f(x)}{\partial x_0} = f(x_0, x_1) \oplus f(x_0, x_1)
\]

is the **vectorial derivative**, \( \min_{x_0} f(x_0, x_1) \land f(x_0, x_1) \) \hspace{1cm} (1)

the **vectorial minimum**, and

\[
\max_{x_0} f(x_0, x_1) \lor f(x_0, x_1)
\]\n
the **vectorial maximum**

of the Boolean function \( f(x) \) with regard to the variables \( x_0 \).

The results of these derivative operations are again Boolean function that depend in general on all variables \( x \). In the special case of \( k = 1 \) we get the single derivative operations.

**Definition 2 (Single Derivative Operations)** Let \( f(x) = f(x_i, x_1) \) be a Boolean function of \( n \) variables, then

\[
\frac{\partial f(x)}{\partial x_i} = f(x_i, x_1) \oplus f(x_i, x_1) = f(x_0 = 0, x_1) \oplus f(x_i = 1, x_1)
\]

is the (single) derivative,

\[
\min_{x_i} f(x_i, x_1) \land f(x_i, x_1) = f(x_0 = 0, x_1) \land f(x_i = 1, x_1)
\] \hspace{1cm} (4)

the (single) **minimum**, and

\[
\max_{x_i} f(x_i, x_1) \lor f(x_i, x_1) = f(x_0 = 0, x_1) \lor f(x_i = 1, x_1)
\] \hspace{1cm} (5)

the (single) **maximum**

of the Boolean function \( f(x) \) with regard to the variable \( x_i \).
All single derivative operations have the welcome property that they do not depend on the variable \( x_i \) anymore. This property can be verified by means of the single derivative for an arbitrary function.

**Theorem 1 (Independent of \( x_i \))** The function \( f(x) = f(x_i, x_1) \) does not depend on the change of \( x_i \) if

\[
\frac{\partial f(x)}{\partial x_i} = 0 .
\]

(7)

Such a function can be expressed by the simpler function \( f(x_1) \). Hence, the value 0 of a single derivative is an indicator that this function is simpler than other functions \( f : \mathbb{B}^n \rightarrow \mathbb{B}^1 \).

Strong bi-decompositions based on this kind of simplification: both decomposition functions depend on less variables than the given function to decompose. The key idea of vectorial bi-decompositions is that a function \( f(x) \) of \( n \) variables that satisfies

\[
\frac{\partial f(x)}{\partial x_0} = 0
\]

(8)
is also simpler than other functions \( f(x) \) which does not satisfy this condition.

### 3 Generalized Lattices of Boolean Functions

Incompletely specified functions are widely used in circuit design. There is a direct relationship between an incompletely specified function and a lattice of Boolean functions. The set of all \( 2^n \) input patterns \( x = (x_1, x_2, \ldots, x_n) \) of an incompletely specified function can be divided into three disjoint sets:

- \( x \in \) don’t-care-set \( \iff f_\varphi(x_1, \ldots, x_n) = 1 \)
  \( \iff \) it is allowed to choose the function value of \( f(x) \) without any restrictions,

(9)

- \( x \in \) ON-set \( \iff f_q(x_1, \ldots, x_n) = 1 \)
  \( \iff (f_\varphi(x_1, \ldots, x_n) = 0) \land (f(x_1, \ldots, x_n) = 1) \),

(10)

- \( x \in \) OFF-set \( \iff f_r(x_1, \ldots, x_n) = 1 \)
  \( \iff (f_\varphi(x_1, \ldots, x_n) = 0) \land (f(x_1, \ldots, x_n) = 0) \).

(11)

The three functions \( f_q(x) \), \( f_r(x) \), and \( f_\varphi(x) \) can be used to mark the set of all functions of a lattice; therefore we refer to these factions as mark functions. All three mark functions cover the whole Boolean space

\[
f_q(x) \lor f_r(x) \lor f_\varphi(x) = 1
\]

(12)

for all vectors \( x \), and they are also mutually disjoint:

\[
f_q(x) \land f_r(x) = 0 ,
\]

(13)

\[
f_q(x) \land f_\varphi(x) = 0 ,
\]

(14)

\[
f_r(x) \land f_\varphi(x) = 0 .
\]

(15)

A function \( f_i(x) \) belongs to the lattice of Boolean functions characterized by the mark functions \( f_q(x) \) and \( f_r(x) \), in short \( f_i(x) \in L (f_q(x), f_r(x)) \), if

\[
f_q(x) \leq f_i(x) \leq f_r(x) .
\]

(16)

The inequality (16) can be transformed into the equation

\[
(f_q(x) \land \overline{f_i(x)}) \lor (f_i(x) \land f_r(x)) = 0 .
\]

(17)

Exactly one of \( 2^{\|f_\varphi(x)\|} \) different Boolean functions coincide in the fixed values of an incompletely specified Boolean function with \( |f_\varphi(x)| \) don’t cares. One of these \( 2^{\|f_\varphi(x)\|} \) functions can be implemented in a circuit.
There are sets of Boolean functions that satisfy the axioms of a lattice \( L \) and even a Boolean Algebra, but cannot be expressed by an incompletely specified function, see [8, 9, 14]. All functions \( f_i(x) \in L \) that satisfy (17) are specified by the ON-set \( f_q(x) \) and the OFF-set \( f_r(x) \); for short we write \( L(f_q(x), f_r(x)) \). A subset of functions \( g_i(x) \in L_g \subseteq L \) can be created such that the ON-set \( g_q(x) = f_q(x) \) and the OFF-set \( g_r(x) = f_r(x) \), i.e., the mark functions of \( L_g \) and \( L \) are the same, but the functions \( g_i(x) \) satisfies the stronger restriction
\[
(g_q(x) ∧ g_i(x)) ∨ (g_r(x) ∧ g_i(x)) ∨ \frac{∂g(x)}{∂x_i} = 0 .
\]

A short characteristic of this lattice is \( L_g \left\langle g_q(x), g_r(x), \frac{∂g(x)}{∂x_i} \right\rangle \). Due to the vectorial derivative in (18) the set of functions \( g_i(x) \in L_g \) is smaller than \( L \), but satisfies the axioms of a lattice and a Boolean Algebra as well.

The vectorial derivative in (18) specifies that all functions of the lattice are independent of the simultaneous change of all variables of \( x_0 \). A lattice can satisfy such a restriction for several directions of change. Hence, the number of directions of change, in which all functions of a lattice are independent, is an important information to characterize a lattice. This information can be stored within an independency matrix.

**Definition 3 (Independency Matrix)** The independency matrix \( IDM(f) \) of a Boolean function \( f(x_1, x_2, \ldots, x_n) \) is a Boolean matrix of \( n \) rows and \( n \) columns. The columns of the independency matrix are associated with the \( n \) variables of the Boolean space in the fixed order \( (x_1, x_2, \ldots, x_n) \). The independency matrix has the shape of an echelon; all elements below the main diagonal are equal to 0. Values 1 of a row of the independency matrix indicate a set of variables for which the vectorial derivative of the function \( f(x_1, x_2, \ldots, x_n) \) is equal to 0. The following rules ensure uniqueness of the independency matrix:

1. Values 1 can only occur to the right of a value 1 in the main diagonal of the independency matrix.
2. All values above a value 1 in the main diagonal of the independency matrix are equal to 0.

The actual number of independent directions of change is implicitly specified by the independency matrix \( IDM \).

**Definition 4 (Rank)** The rank of an independency matrix \( IDM(f) \) describes the number of independent directions of change of the Boolean function \( f(x_1, x_2, \ldots, x_n) \). The rank(\( IDM(f) \)) is equal to the number of values 1 in the main diagonal of the unique \( IDM(f) \).

A lattice of Boolean functions can be restricted to functions which do not change their values in the case of the simultaneous change of several variables or even of a set of such directions of change. These directions of change can be expressed by a disjunction of appropriate vectorial derivatives which are uniquely indicated in the independency matrix. For a short notation we define the independency function \( f^{id}(x) \).

**Definition 5 (Independency Function)** The independency function \( f^{id}(x) \) of a Boolean function corresponds to the independency matrix \( IDM(f) \) such that
\[
f^{id}(x) = \sum_{i=1}^{n} \frac{∂f(x)}{∂x_i} ,
\]
where
\[
\frac{∂f(x)}{∂x_i} = 0
\]
if all elements of the row \( i \) in \( IDM(f) \) are equal to 0, and
\[
x_j \in x_0 \text{ if } IDM(f)[i,j] = 1 ,
\]
where \( IDM(f)[i,j] \) is the value of the row \( i \) and the column \( j \) of the independency matrix \( IDM(f) \).
In this way, the Boolean Differential Calculus facilitates the more general definition of lattices of Boolean functions \(f_i(x) \in L\{f_q(x), f_r(x), f^{id}(x)\}:\)

\[
(f_q(x) \land \neg f(x)) \lor (f_i(x) \land f_r(x)) \lor f^{id}(x) = 0.
\] (22)

Definition 3 requires a unique specification of the independent directions of change. For easy dealing with directions of change we introduce the binary vector \(s\).

**Definition 6 (Binary Vector (BV))** Let \(f(x) = f(x_1, x_2, \ldots, x_n)\) be a Boolean function and \(x_0 \subseteq x\) be a subset of variables; then

\[
s_0 = BV(x_0)
\]

(23)

is a binary vector of \(n\) elements where \(s_0[i] = 1\) indicates that \(x_i \in x_0\).

**Algorithm 1** \(s_{\text{min}} = \text{MIDC(IDM}(f), x_0)\): Minimal Independent Direction of Change

\[
\begin{align*}
\text{Input} : & \quad x_0 \subseteq x: \text{evaluated subset of variables}, \\
\text{Input} : & \quad \text{IDM}(f): \text{unique independency matrix of } n \text{ rows and } n \text{ columns of } f(x) \\
\text{Output} : & \quad s_{\text{min}}: \text{minimal direction of change} \\
1: & \quad j \leftarrow 1 \\
2: & \quad s_{\text{min}} \leftarrow BV(x_0) \\
3: & \quad \text{while } j \leq n \text{ do} \\
4: & \quad \quad \text{if } (s_{\text{min}}[j] = 1) \land (\text{IDM}(f)[j, j] = 1) \text{ then} \\
5: & \quad \quad \quad s_{\text{min}} \leftarrow s_{\text{min}} \oplus \text{IDM}(f)[j] \\
6: & \quad \quad \text{end if} \\
7: & \quad j \leftarrow j + 1 \\
8: & \quad \text{end while}
\end{align*}
\]

Algorithm 1 calculates the vector \(s_{\text{min}}\) that indicates the minimal direction of change not covered by the given independency matrix \(\text{IDM}(f)\) and the set of variables \(x_0\). The function \(BV(x_0)\) in line 2 maps the variables into a binary vector, where the bit \(s_{\text{min}}[j]\) indicates that the variable \(x_j\) belongs to \(x_0\). The initial vector \(s_{\text{min}}\) is modified in line 5 if both the bit \(s_{\text{min}}[j]\) and the bit in the main diagonal \(\text{IDM}(f)[j, j]\) are equal to 1. The EXOR-operation between \(s_{\text{min}}\) and the \(j\)-th row \(\text{IDM}(f)[j]\) of the independency matrix \(\text{IDM}(f)\) in line 5 of Algorithm 1 ensures that the created vector \(s_{\text{min}}\) satisfies Rule 2 of Definition 3 when this vector is included in the independency matrix \(\text{IDM}(f)\). The result of Algorithm 1 is a uniquely specified vector \(s_{\text{min}}\) that also satisfies Rule 1 of Definition 3 when it is included into the independency matrix \(\text{IDM}(f)\) such that the most significant bit belongs to the main diagonal. A result vector \(s_{\text{min}} = 0\) of Algorithm 1 indicates that all functions \(f(x)\) of the used \(\text{IDM}(f)\) already satisfy:

\[
\frac{\partial f(x)}{\partial x_0} = 0.
\] (24)

Algorithm 2 realizes the unique merge of a give direction of change specified by \(x_0\) into the independency matrix \(\text{IDM}(f)\). Initial steps copy \(\text{IDM}(f)\) to \(\text{IDM}(g)\) and calculate the unique vector \(s_{\text{min}}\) for the given \(x_0\) using Algorithm 1. The copied independency matrix \(\text{IDM}(g)\) must changed only if \(s_{\text{min}} > 0\). The index of the most significant bit \(j\) of \(s_{\text{min}}\) indicates within \(\text{IDM}(g)\) both

- the row where \(s_{\text{min}}\) must be stored to satisfy due to Rule 1 of Algorithm 1 and
- the column which must be evaluated due to Rule 2 of Definition 3.

The operations within the while-loop in lines 6 to 11 perform the needed changes by conditional \(\oplus\)-operations where \(\text{IDM}(g)[i, j]\) indicates the value of the \(i\)-th row and the \(j\)-th column and \(\text{IDM}(g)[i, i]\) specifies the \(i\)-th row of the independency matrix \(\text{IDM}(g)\). The new vector \(s_{\text{min}}\) is included into the independency matrix \(\text{IDM}(g)\) in line 12. If \(s_{\text{min}} \neq 0\) then all functions

\[
g_i(x) \in L\{g_q(x), g_r(x), g^{id}(x)\}.
\]
are simpler than

\[ f_i(x) \in L\{f_q(x), f_r(x), f^d(x)\} \]

because

\[ \text{rank}(\text{IDM}(g)) > \text{rank}(\text{IDM}(f)) \].

**Example 1 (Sequence of extensions of the IDM(f) by four directions of change)**

We assume that all functions of a lattice depend basically on all directions of change. Step by step simpler function are found by means of bi-decompositions. The found directions of change from which all functions are independent are represented by the unique independency matrix IDM(f).

Figure 1 (a) shows that we start with a zero matrix IDM(f) with \( \text{rank}(\text{IDM}(f)) = 0 \) and add the first direction of change \((x_2, x_4)\) from which the decomposition function does not depend anymore. Due to the given matrix no change from \( s_0 \) to \( s_{\text{min}} \) is necessary, the vector \( s_{\text{min}} = (0101) \) is stored in row 2 of IDM(f), and the new \( \text{rank}(\text{IDM}(f)) = 1 \).

Figure 1 (b) shows that next all functions are also independent of \((x_1, x_3)\), the \( s_0 = (1010) \) is not influenced by IDM(f), the unchanged vector \( s_{\text{min}} = (1010) \) is stored as row 1, and \( \text{rank}(\text{IDM}(f)) \) is increased to 2.

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( s_0 = (0 1 0 1) \)

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( s_0 = (1 0 1 0) \)

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( s_0 = (0 1 1 1) \)

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( s_0 = (1 1 0 1) \)

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( s_0 = (0 0 1 0) \)

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**(d)**

Figure 1: Adding several directions of change to the independency matrix IDM(f).
Figure 1 (c) assumes that all further simplified functions do not depend on the simultaneous change of \((x_2, x_3, x_4)\). Due to the second row of IDM(\(f\)) Algorithm 1 finds the minimal direction of change \(s_{min} = (0010)\), and Algorithm 2 creates the unique IDM(\(f\)) by setting IDM(\(f\))[1,3]=0, and storing \(s_{min}\) as row 3, so that the new \(\text{rank}(\text{IDM}(f)) = 3\).

As can be seen in Figure 1 (d), the final detected direction of change from which all functions of the lattice are independent is \((x_1, x_2, x_4)\). Algorithm 1 finds that this is no new direction of change because \((1101) \oplus (1000) \oplus (0101) = (0000)\), and Algorithm 2 consequently does not change the IDM(\(f\)). The Rank remains unchanged: \(\text{rank}(\text{IDM}(f)) = 3\).

4 Vectorial Bi-Decompositions

4.1 Definition

A recursive decomposition approach terminates if the two decomposition functions \(g\) and \(h\) of a bi-decomposition are simpler than the given function \(f\). A lattice functions increases the possibility to find a bi-decomposition. The condition for simplification is satisfied if

\[
\text{rank}(g(x)) > \text{rank}(f(x)) \quad \text{(25)}
\]

\[
\text{rank}(h(x)) > \text{rank}(f(x)) \quad \text{(26)}
\]

Figure 2 shows the structure of the three types of vectorial bi-decompositions and summarizes the conditions for simplifications.

**Definition 7 (Vectorial Bi-Decomposition for Generalized Lattices)** A lattice

\[
L \langle f_0(x), f_r(x), f_{id}(x) \rangle
\]

contains at least one function \(f_i(x)\) that is OR-, AND-, or EXOR-bi-decomposable with regard to the subsets of variables \(x_a \subseteq x\) and \(x_b \subseteq x, x_a \neq x_b\), if

1. \(f_i(x)\) can be expressed by

\[
f_i(x) = g(x) \bullet h(x)
\]

where \(\bullet \in \{\lor, \land, \oplus\}\),

2. the decomposition functions \(g(x)\) and \(h(x)\) satisfy

\[
\frac{\partial g(x)}{\partial x_b} = 0 \quad \text{(29)}
\]

\[
\frac{\partial h(x)}{\partial x_a} = 0 \quad \text{(30)}
\]
and the functions of the given lattice depend on the simultaneous change of both $x_a$ and $x_b$:

$$MIDC(IDM(f), x_a) \neq 0, \quad (31)$$

$$MIDC(IDM(f), x_b) \neq 0. \quad (32)$$

### 4.2 Vectorial OR-Bi-Decompositions

**Theorem 2 (Vectorial OR-Bi-Decomposition for a Generalized Lattice)**

A lattice of (27) contains at least one logic function $f(x)$ that is vectorially OR-bi-decomposable with regard to the dedicated sets $x_a \subseteq x$ and $x_b \subseteq x$, $x_a \neq x_b$, that satisfy (31), (32) if and only if

$$f_q(x) \land \max_{x_a} f_r(x) \land \max_{x_b} f_r(x) = 0. \quad (33)$$

Possible decomposition functions $g_i(x)$ can be chosen from the lattice $L\langle g_q(x), g_r(x), g^{id}(x) \rangle$ with

$$g_q(x) = \max_{x_a} (f_q(x) \land \max_{x_a} f_r(x)), \quad (34)$$

$$g_r(x) = \max_{x_a} f_r(x), \quad (35)$$

$$g^{id}(x) = f^{id}(x) \lor \frac{\partial f(x)}{\partial x_a}, \quad (36)$$

so that

$$IDM(g) = UM(IDM(f), x_b). \quad (37)$$

Assuming that the function $g(x)$ is chosen from this lattice to realize the circuit, possible decomposition functions $h_j(x)$ can be chosen from the lattice $L\langle h_q(x), h_r(x), h^{id}(x) \rangle$ with

$$h_q(x) = \max_{x_a} (f_q(x) \land g(x)), \quad (38)$$

$$h_r(x) = \max_{x_a} f_r(x), \quad (39)$$

$$h^{id}(x) = f^{id}(x) \lor \frac{\partial f(x)}{\partial x_a}, \quad (40)$$

so that

$$IDM(h) = UM(IDM(f), x_a). \quad (41)$$

**Proof 1 (Vectorial OR-Bi-Decomposition for a Generalized Lattice)**

The directions of change of a vectorial bi-decomposition are specified by the subsets of variables $x_a$ and $x_b$. These two directions of change determine quadruples of nodes in the Boolean space. Such quadruples of nodes are also evaluated for strong bi-decompositions with regard to single variables $x_i$ and $x_j$. The formulas of Theorem 2 transform the known condition of the strong OR-bi-decomposition with regard to $x_i$ and $x_j$ to the new vectorial OR-bi-decomposition with regard to $x_a$ and $x_b$. The complexities of the decomposition lattices $L_g$ and $L_h$ are reduced due to the general requirements (29) and (30) of any vectorial bi-decomposition.

### 4.3 Vectorial AND-Bi-Decompositions

**Theorem 3 (Vectorial AND-Bi-Decomposition for a Generalized Lattice)**

A lattice of (27) contains at least one logic function $f(x)$ that is vectorially AND-bi-decomposable with regard to the dedicated sets $x_a \subseteq x$ and $x_b \subseteq x$, $x_a \neq x_b$, that satisfy (31), (32) if and only if

$$f_r(x) \land \max_{x_a} f_q(x) \land \max_{x_b} f_q(x) = 0. \quad (42)$$
Possible decomposition functions $g_t(x)$ can be chosen from the lattice $L\langle g_q(x), g_r(x), g^{id}(x)\rangle$ with
\[
g_q(x) = \max_{x_b} f_q(x) ,
\]
\[
g_r(x) = \max_{x_a} (f_r(x) \land \max_{x_b} f_q(x)) ,
\]
\[
g^{id}(x) = f^{id}(x) \lor \frac{\partial f(x)}{\partial x_b} ,
\]
so that
\[
IDM(g) = UM(IDM(f), x_b) .
\]

Assuming that the function $g(x)$ is chosen from this lattice to realize the circuit, possible decomposition functions $h_j(x)$ can be chosen from the lattice $L\langle h_q(x), h_r(x), h^{id}(x)\rangle$ with
\[
h_q(x) = \max_{x_b} f_q(x) ,
\]
\[
h_r(x) = \max_{x_a} (f_r(x) \land g(x)) ,
\]
\[
h^{id}(x) = f^{id}(x) \lor \frac{\partial f(x)}{\partial x_a} ,
\]
so that
\[
IDM(h) = UM(IDM(f), x_a) .
\]

**Proof 2 (Vectorial AND-Bi-Decomposition for a Generalized Lattice)**

The duality between the Boolean Algebras $(\mathbb{B}^n, \land, \lor, \neg, 0, 1)$ and $(\mathbb{B}^n, \lor, \land, \neg, 1, 0)$ is the basis for the similarity between the vectorial OR- and AND-bi-decomposition. Hence, the proof of Theorem 3 is done in the same manner as before for Theorem 2.

### 4.4 Vectorial EXOR-Bi-Decompositions

**Theorem 4 (Vectorial EXOR-Bi-Decomposition for a Generalized Lattice)**

A lattice of (27) contains at least one logic function $f(x)$ that is vectorially EXOR-bi-decomposable with regard to the dedicated sets $x_a \subseteq x$ and $x_b \subseteq x$, $x_a \neq x_b$, that satisfy (31), (32) if and only if
\[
\max_{x_a} (\max_{x_a} f_q(x) \land \max_{x_b} f_r(x)) \land (\min_{x_a} f_q(x) \lor \min_{x_b} f_r(x)) = 0 .
\]

Possible decomposition functions $g_t(x)$ can be chosen from the lattice $L\langle g_q(x), g_r(x), g^{id}(x)\rangle$ with $x_{ba} \in x_b$
\[
g_q(x) = x_{ba} \land \max_{x_a} (\max_{x_a} f_q(x) \land \max_{x_b} f_r(x)) ,
\]
\[
g_r(x) = \min_{x_a} f_q(x) \land \min_{x_a} f_r(x) ,
\]
\[
g^{id}(x) = f^{id}(x) \lor \frac{\partial f(x)}{\partial x_b} ,
\]
so that
\[
IDM(g) = UM(IDM(f), x_b) .
\]

Assuming that the function $g(x)$ is chosen from this lattice to realize the circuit, possible decomposition functions $h_j(x)$ can be chosen from the lattice $L\langle h_q(x), h_r(x), h^{id}(x)\rangle$ with
\[
h_q(x) = \max_{x_a} \left( (\max_{x_a} f_q(x) \land g(x)) \lor (f_r(x) \land g(x)) \right) ,
\]
\[
h_r(x) = \max_{x_a} \left( (\min_{x_a} f_q(x) \land g(x)) \lor (f_r(x) \land g(x)) \right) ,
\]
\[
h^{id}(x) = f^{id}(x) \lor \frac{\partial f(x)}{\partial x_a} ,
\]
so that
\[
IDM(h) = UM(IDM(f), x_a) .
\]
Proof 3 (Vectorial EXOR-Bi-Decomposition for a Generalized Lattice)

Condition (51) is the generalization of the known condition

\[ \frac{\partial}{\partial x_5} \left( \frac{\partial f(x)}{\partial x_a} \right) = 0 \]

of the vectorial bi-decomposition of a single function to the more general case of lattice. All \(3^4 \times 2^{n-2}\) possible assignments are precisely evaluated by Condition (51). The first part

\[ \max(\max f_q(x) \land \max f_r(x)) \]

is equal to 1 for all four nodes of a region specified by the directions of change \(x_a\) and \(x_b\), if function value changes in this region are caused by the simultaneous change of variables of \(x_a\). The second part

\[ (\min f_q(x) \lor \min f_r(x)) \]

is equal to 1 if the function value must remain unchanged for the same direction of change \((x_b)\). In this case an odd number of function values 1 occurs in the evaluated region and a vectorial EXOR-bi-decomposition is not possible.

The remaining formulas of Theorem 4 adapt the known formulas for the strong EXOR-bi-decomposition of a lattice with regard to single variables \(x_i\) and \(x_j\) to the new vectorial EXOR-bi-decomposition of a generalized lattice with regard to \(x_a\) and \(x_b\).

5 Application of the Vectorial Bi-Decomposition

We assume that one Boolean function of a given lattice has to be implemented by a circuit of two-input AND-, OR-, or EXOR-gates where negated inputs are possible. The mark functions of this lattice are

\[ f_q(x_1, x_2, x_3, x_4, x_5) = \overline{\overline{\overline{\overline{x_1}}}} \overline{\overline{\overline{x_2}}} \overline{\overline{\overline{x_3}}} x_4 x_5 \lor x_1 x_2 x_3 \overline{x_4} x_5 \lor x_1 x_2 \overline{\overline{x_3}} x_4 x_5 \lor \overline{\overline{x_1}} x_2 \overline{\overline{x_3}} x_4 x_5 \lor x_1 x_2 x_3 x_4 x_5 \lor x_1 x_2 \overline{\overline{x_3}} x_4 \overline{x_5} , \]

\[ f_r(x_1, x_2, x_3, x_4, x_5) = \overline{x_1} x_2 \overline{x_3} x_4 x_5 \lor x_2 x_3 \overline{x_4} x_5 \lor x_1 x_2 x_3 x_4 \overline{x_5} \lor x_1 \overline{x_2} x_3 x_4 \overline{x_5} \lor \overline{x_1} \overline{x_2} x_3 x_4 \overline{x_5} \lor \overline{x_1} x_2 x_3 \overline{x_4} x_5 \lor x_1 x_2 \overline{x_3} x_4 \overline{x_5} \lor x_1 x_2 x_3 x_4 x_5 \lor x_1 x_2 \overline{x_3} x_4 x_5 \lor \overline{x_1} x_2 x_3 \overline{x_4} x_5 \lor \overline{x_1} x_2 x_3 x_4 \overline{x_5} \lor x_1 x_2 \overline{x_3} x_4 \overline{x_5} \lor x_1 x_2 x_3 x_4 x_5 . \]

Figure 3 (a) shows the associated Karnaugh-map.

\[
\begin{array}{cccccc|cccccc}
\text{x}_4 & \text{x}_5 & f_q(x) & \text{x}_4 & \text{x}_5 & f(x) \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc|cccccc}
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Figure 3: Karnaugh-maps of (a) the given lattice and (b) the realized Boolean function.

The result of an exhaustive analysis is that there exists neither and strong AND-, a strong OR-, nor a strong EXOR-bi-decomposition. However, the condition (33) of the vectorial OR-bi-decomposition with regard to \(x_a = (x_1, x_4)\) and \(x_b = (x_1, x_2, x_4)\) is satisfied. Using the rules (34) and (35) we get in this special case the completely specified function

\[ g(x) = g_q(x) = g_r(x) = \overline{x_1} \overline{x_2} \overline{x_3} \overline{x_4} x_5 \lor x_1 x_2 \overline{x_3} x_4 x_5 \]

This function does not depend on the simultaneous change of \((x_1, x_2, x_4)\) and has a strong and even disjoint AND-bi-decomposition with regard to \(x_a = (x_3, x_5)\) and \(x_b = (x_1, x_2, x_4)\).
The decomposition function $h(x)$ does not depend on the simultaneous change of $x_1$ and $x_4$. Hence, we can take advantage of the general property (reported in [10]) that a strong EXOR-bi-decomposition with regard to these variables exists. Using the rules (38) and (39) we also get in this special case a completely specified function:

$$h(x) = h_q(x) = h_r(x) = (x_1 x_2 x_3 \lor x_1 x_5 (x_2 \lor x_3)) \oplus (x_2 x_3 (x_4 \oplus x_5) \lor x_4 x_5 (x_2 \oplus x_3)) .$$ \hspace{1cm} (63)

The sub-expression before the central $\oplus$-operation does not depend on $x_4$ and sub-expression thereafter is independent of $x_1$. Figure 3 (b) shows the selected function of the given lattice of $2^5 = 32$ functions. It can be see that three of the five don’t cares are set to 1 and the remaining two don’t cares are set to 0.

Figure 4 shows the synthesized circuit structure where the output OR-gate realizes the vectorial OR-bi-decomposition. Table 1 summarizes the synthesis results for alternative approaches. The combination of the vectorial and strong bi-decomposition outperforms the alternative approaches in both the number of gates and the delay roughly expressed by the number of levels.

### 6 Conclusions

Using the known theory about both the strong and the vectorial bi-decompositions for single functions, the known generalization of strong bi-decompositions for lattices of functions, as well as the generalization of the Boolean differential calculus for lattices of Boolean functions, we were able to specify the conditions and rules to derive the decomposition lattices for all types of vectorial bi-decompositions of generalized lattices of Boolean functions. A synthesis example confirms the benefit of the common use of the vectorial and strong bi-decomposition.

As future task remains the experimental evaluation of the unified decomposition approach that takes into account strong, vectorial, and weak bi-decompositions for generalized lattices of Boolean functions.
References


