Simpler Functions for Decompositions

Bernd Steinbach

1 Introduction

There are two basic approaches to find the structure of a combinatorial circuit for a given behavior. These are either covering methods or decomposition methods. One of the decomposition methods is the bi-decomposition. The basic idea of the bi-decomposition is that the given function is built by an AND-gate, an OR-gate, or by an XOR-gate of two inputs. If one of these gates splits the given function into two simpler functions, a complete circuit structure must be found after a certain number of decomposition steps.

However, two complicate functions, which control the inputs of one of these gates, can be merged by the gate into a simpler output function. In this case, the decomposition does find a circuit structure with a finite number of gates.

The bi-decomposition [1, 3] ensures the simplification in each decomposition step because it restricts to subfunctions of the decomposition which depend on less variables than the given function. The known bi-decomposition approach even detects existing subfunctions which depend on the smallest number of variables. Of course, such very simple subfunctions should be utilized in a decomposition, because they reduce the number of required decomposition gates and contribute to short path lengths.

There are Boolean functions for which no strong bi-decomposition exists. In such a case, a weak bi-decomposition finds one simpler subfunction and extends the other subfunctions to a lattice of functions. A complete circuit structure can be synthesized for each Boolean function only using the AND-, the OR-, and the XOR-bi-decomposition, and additionally the weak AND-, and the weak OR-bi-decomposition.

The drawback of this approach is that weak bi-decompositions are needed to reach the completeness and each bi-decomposition step adds one gate to the path of the decomposed function. An interesting question is whether the weak bi-decomposition can be substituted by a bi-decomposition into simpler subfunctions which depend on the same number of variables as the given function to decompose. This paper combines the knowledge of two other approaches to answer this question.

One of these approaches is generalization of lattices of Boolean functions. The independence of a function from a single variable can be detected by a simple derivative of the Boolean Differential Calculus [4]. The $2^{2^{n-1}}$ functions of $\mathbb{B}^n$ which are independent of a single variable x_i belong to the well-known lattice. In [5, 7] the existence of more general lattices are introduced in which a vectorial derivative overtake the role of the simple derivative. Functions of such lattices depend on all variables but are simpler than other functions of $\mathbb{B}^n$. It is possible to utilize these new found lattices for the bi-decomposition?

Another source to find simpler functions utilizes the Specialized Normal Form (SNF) [6]. The SNF is a unique ESOP representation of a Boolean function and the number of cubes in the SNF indicates the complexity of the function [8]. It arises the question about the relation of these two approaches and the possibilities to utilize such information for the bi-decomposition. The latest results of the research in this field are summarized in this paper.

To make the paper self-contained, Section 2 gives the needed definitions of derivatives, Section 3 introduces the basic principle of the SNF, and Section 4 very briefly explains the bi-decomposition approach. The results of the main analysis are explored in Section 5. This analysis studies the relations between the complexity provided by the SNF and dependencies of all functions of $\mathbb{B}^4$ regarding all directions of change. An example in Section 6 demonstrates achievable benefits of the suggested extended approach of the bi-decomposition before Section 7 concludes the paper.

B. Steinbach (✉)
Freiberg University of Mining and Technology, Institute of Computer Science, D-09596 Freiberg, Germany

H. Selvaraj et al. (eds.), *Progress in Systems Engineering: Proceedings of the Twenty-Third International Conference on Systems Engineering*, Advances in Intelligent Systems and Computing 1089,
DOI 10.1007/978-3-319-08422-0_119,

2 Simple and Vectorial Derivative

The simple derivative of a Boolean function $f(\mathbf{x})$ with regard to the variable x_i describes for which patterns of the remaining variables the change of the x_i-value causes the change of the function value.

Definition 1. *Let $f(\mathbf{x}) = f(x_1, \ldots, x_i, \ldots, x_n)$ be a Boolean function of n variables, then*

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = f(x_1, \ldots, x_i, \ldots, x_n) \oplus f(x_1, \ldots, \overline{x}_i, \ldots, x_n) \quad (1)$$

is the (simple) derivative of the Boolean function $f(\mathbf{x})$ with regard to the variable x_i.

The simple derivative $\frac{\partial f(\mathbf{x})}{\partial x_i}$ is again a Boolean function. If

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 0 \quad (2)$$

then the function $f(\mathbf{x})$ is independent of the variable x_i. From Definition (1) follows the welcome property that the result function of the simple derivative $\frac{\partial f(\mathbf{x})}{\partial x_i}$ does not depend on the variable x_i anymore.

$$\frac{\partial}{\partial x_i}\left(\frac{\partial f(\mathbf{x})}{\partial x_i}\right) = 0 \quad (3)$$

holds for all Boolean functions $f(\mathbf{x})$ of $\mathbb{B}^n$.

The vectorial derivative of function $f(\mathbf{x})$ has a similar meaning like the simple derivative. The difference is that in the case of the vectorial derivative several variables change their values at the same point in time.

Definition 2. *Let $\mathbf{x}_0 = (x_1, x_2, \ldots, x_k)$, $\mathbf{x}_1 = (x_{k+1}, x_{k+2}, \ldots, x_n)$ be two disjoint sets of Boolean variables, and $f(\mathbf{x}_0, \mathbf{x}_1) = f(x_1, x_2, \ldots, x_n) = f(\mathbf{x})$ a Boolean function of n variables, then*

$$\frac{\partial f(\mathbf{x}_0, \mathbf{x}_1)}{\partial \mathbf{x}_0} = f(\mathbf{x}_0, \mathbf{x}_1) \oplus f(\overline{\mathbf{x}}_0, \mathbf{x}_1) \quad (4)$$

is the *vectorial derivative of the Boolean function $f(\mathbf{x}_0, \mathbf{x}_1)$ with regard to the variables of $\mathbf{x}_0$.*

The vectorial derivative $\frac{\partial f(\mathbf{x}_0, \mathbf{x}_1)}{\partial \mathbf{x}_0}$ is also a Boolean function, but differently to the simple derivative, a vectorial derivative depends in general on all variables $(\mathbf{x}_0, \mathbf{x}_1)$ like the given function $f(\mathbf{x}_0, \mathbf{x}_1)$. However, a vectorial derivative is also simpler than the given function, because:

$$\frac{\partial}{\partial \mathbf{x}_0}\left(\frac{\partial f(\mathbf{x}_0, \mathbf{x}_1)}{\partial \mathbf{x}_0}\right) = 0 \quad (5)$$

holds for all Boolean functions $f(\mathbf{x}_0, \mathbf{x}_1)$.

3 Specialized Normal Form - SNF

The Specialized Normal Form was found in a research for minimal Exclusive-OR Sum Of Products (ESOPs) in [6]. The number of cubes in the SFN allows us to distinguish several complexity classes of functions in $\mathbb{B}^n$. Further subclasses were detected in [8] using the Hamming distance δ between the cubes of an SNF.

The SNF utilizes the following algebraic property of the exclusive-or operation ($\oplus$) and the Boolean variable x:

$$x = \overline{x} \oplus 1 \quad (6)$$

$$\overline{x} = 1 \oplus x \quad (7)$$

$$1 = x \oplus \overline{x}. \quad (8)$$

These three formulas show that each element of the set $\{x, \overline{x}, 1\}$ has isomorphic properties. For each variable in the support of the Boolean function f, exactly one left-hand side element of (6), (7), or (8) is included in each cube of an ESOP of f. An application of these formulas from the left to the right doubles the number of cubes and is called expansion. The reverse application of these formulas from the right to the left halves the number of cubes and is called compaction.

The procedure to construct the SNF utilizes one more property the Boolean function f, a cube C, and the exclusive-or operation:

$$f = f \oplus 0 \quad (9)$$

$$0 = C \oplus C \quad (10)$$

$$f = f \oplus C \oplus C. \quad (11)$$

From these formulas follows that two identical cubes can be added to or removed from any ESOP without changing the represented function. The SNF can be defined using two simple algorithms based on the properties mentioned above.

The `expand()` function in line 3 of Algorithm 1 expands the cube C_j with regard to the variable V_i into the cubes C_{n1} and C_{n2} based on the fitting formula (6), ..., (8). Algorithm 1 realizes this expansion for all variables of all cubes of a given ESOP. Assuming n variables in the given ESOP, this Algorithm distributes the information about each given cube to 2^n cubes, similar to the creation of a hologram

Algorithm 1 Exp(*f*)

Input: any ESOP of a Boolean function f
Output: complete expansion of the Boolean function f with regard to all variables of its support

```
for all variables V_i of the support of f do
    for all cubes C_j of f do
        ⟨C_n1, C_n2⟩ ← expand(C_j, V_i)
        replace C_j by ⟨C_n1, C_n2⟩
    end for
end for
```

Algorithm 2 R(*f*)

Input: any ESOP of a Boolean function f containing n cubes
Output: reduced ESOP of f containing no cube more than once

```
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        if C_i = C_j then
            C_i ← C_{n−1}
            C_j ← C_{n−2}
            n ← n − 2
            j ← i
        end if
    end for
end for
```

of an object. Algorithm 2 removes all pairs of cubes using the formulas (9), (10), and (11) so that a unique ESOP of the Boolean function *f* remains.

Using Algorithms Exp(*f*) and R(*f*) it is possible to create a special ESOP having a number of remarkable properties which are specified and proven in [6].

Definition 3 (SNF(*f*)). *Take any ESOP of a Boolean function f. The ESOP resulting from*

$$SNF(f) = R(Exp(f)) \tag{12}$$

is called the Specialized Normal Form (SNF) of the Boolean function.

4 Bi-Decomposition

A bi-decomposition (see left part of Figure 1) decomposes a function $f(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ into two subfunctions $g(\mathbf{x}_a, \mathbf{x}_c)$ and $h(\mathbf{x}_b, \mathbf{x}_c)$. Both subfunctions are simpler than the given function $f(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ due to the missing variables $\mathbf{x}_b$ in the subfunction $g(\mathbf{x}_a, \mathbf{x}_c)$ and the missing variables $\mathbf{x}_a$ in the subfunction $h(\mathbf{x}_b, \mathbf{x}_c)$.

There are three types of by bi-decompositions shown in the left part of Figure 1. It is a property of the function $f(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ whether a bi-decomposition exists with regard to one of the decomposition-gates. An empty set of variables $\{\mathbf{x}_c\}$ and the split of the set of all variables $\{\mathbf{x}\}$ into the subsets $\{\mathbf{x}_a\}$ and $\{\mathbf{x}_b\}$ of the same size contributes best to the synthesis of a circuit by bi-decomposition. However, only few functions have this welcome property.

A necessary condition of the bi-decomposition [4] is that both the set of variables $\{\mathbf{x}_a\}$ and the set of variables $\{\mathbf{x}_b\}$ contains at least one variable. In the limit case of single variables in the sets $\{\mathbf{x}_a\}$ and $\{\mathbf{x}_b\}$, we can assume $x_i = \mathbf{x}_a$ and $x_j = \mathbf{x}_b$; nevertheless both subfunctions of the bi-decomposition are simpler than the given function $f(x_i, x_j, \mathbf{x}_c)$ because:

$$\frac{\partial g(x_i, \mathbf{x}_c)}{\partial x_j} = 0 \quad \text{and} \quad \frac{\partial h(x_j, \mathbf{x}_c)}{\partial x_i} = 0. \tag{13}$$

Unfortunately, there are functions for which no bi-decomposition exists. Le [2] additionally suggested for such cases the weak bi-decomposition. The function to decompose must hold a certain condition [4] for the weak AND-bi-decomposition (Figure 1 (d)) and the weak OR-bi-decomposition (Figure 1 (e)). Only the subfunction $h(\mathbf{x}_c)$ is simpler due to the missing variables $\mathbf{x}_a$. The function $g(\mathbf{x}_a, \mathbf{x}_c)$ of a weak bi-decomposition depends on the same

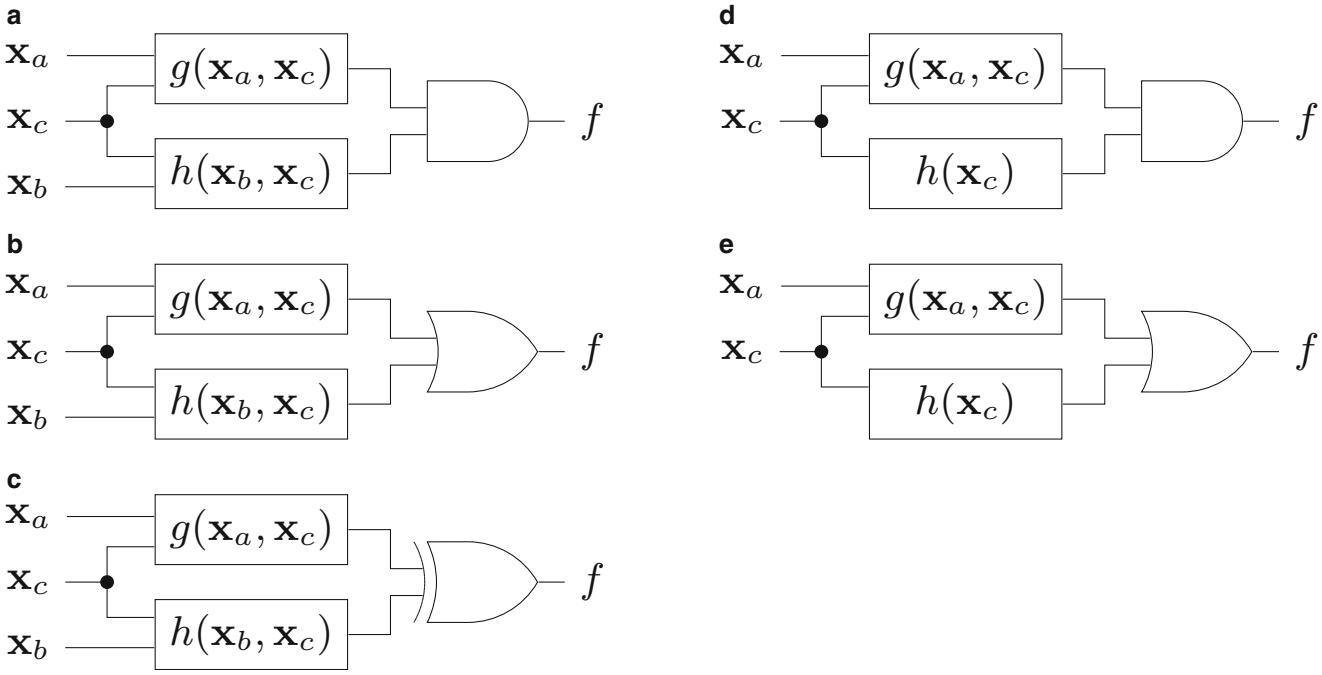

Fig. 1 Circuit structures of bi-decompositions: (a) AND-bi-decomposition; (b) OR-bi-decomposition; (c) XOR-bi-decomposition; (d) weak AND-bi-decomposition; (e) weak OR-bi-decomposition

variables as the given function $f(\mathbf{x}_a, \mathbf{x}_c)$, however, this function can be chosen from a larger lattice of functions.

A weak XOR-bi-decomposition can be realized for each pair of functions $f(\mathbf{x}_a, \mathbf{x}_c)$, $h(\mathbf{x}_c)$. However, the subfunction $g(\mathbf{x}_a, \mathbf{x}_c)$ can be more complicated than the given function $f(\mathbf{x}_a, \mathbf{x}_c)$. For that reason the weak XOR-bi-decomposition is excluded from the synthesis approach by bi-decomposition.

The recursive application of one of the three types of the bi-decomposition together with the weak OR-bi-decomposition and weak AND-bi-decomposition enables a complete multilevel design of each function. This completeness follows from Theorem 1 found by Le [2].

Theorem 1. *If the function* $f(\mathbf{x}_a, \mathbf{x}_c)$ *is neither weakly OR-bi-decomposable nor weakly AND-bi-decomposable with regard to a single variable* $x_i = \mathbf{x}_a$, *then the function* $f(x_i, \mathbf{x}_b)$ *is disjointly XOR-bi-decomposable with regard to the single variable* x_i *and the set of variables* $\{\mathbf{x}_b\} = \{\mathbf{x}_c\}$.
The proof of Theorem 1 is also given in [4].

5 Experimental Results

The key of the bi-decomposition is that each created subfunction is either independent of at least one variable x_i, which can be checked by (2); or the created subfunction belongs to a larger lattice than the given function $f(\mathbf{x})$. After several extensions, such a lattice contains a function $f(\mathbf{x})$ that also holds (2). The larger the number of variables from which all functions of the lattice are independent the simpler circuits can be synthesized by bi-decomposition.

In [7, 5] was shown, that

1. there are lattices of Boolean functions which do not depend on a certain number of variables;
2. the constant value 0 of the simple derivative (1) of such a function with regard to a respective variable indicates this property;
3. the constant value 0 of a vectorial derivative (1) also indicates a simpler function;
4. there are $2^n - 1$ directions of change in $\mathbb{B}^n$;
5. the number of independent directions of change for lattices in $\mathbb{B}^n$ is restricted to n.

From these findings arises the question whether vectorial derivatives can be utilized to find simpler subfunctions of a bi-decomposition. An alternative measure of the complexity of a Boolean function $f(\mathbf{x})$ is the number of cubes in the SNF ($f(\mathbf{x})$) [6, 8]. An experiment allows us to evaluate these properties form a more general point of view.

For that reason we calculated for all 65,536 Boolean functions $f(\mathbf{x})$ of $\mathbb{B}^4$ the number of cubes in the SNF($f(\mathbf{x})$) and all simple and vectorial derivatives. Table 5 summarizes these experimental results as follows:

- column 1 contains the numbers of functions of one complexity class of the SNF;
- column 2 lists the numbers of cubes of the SNF class as measure of the complexity;

Table 1 Evaluation of all functions of $\mathbb{B}^4$ regarding the SNF and vectorial derivatives

number of functions	number of cubes in the SNF	functions with $\frac{\partial f}{\partial \mathbf{x}} = 0$ and $\lvert\mathbf{x}\rvert = 1$	$\lvert\mathbf{x}\rvert = 2$	$\lvert\mathbf{x}\rvert = 3$	$\lvert\mathbf{x}\rvert = 4$	number
1	0	4	6	4	1	1
81	16	0	0	0	0	16
	16	1	0	0	0	32
	16	2	1	0	0	24
	16	3	3	1	0	8
	16	4	6	4	1	1
324	24	0	0	0	0	96
	24	1	0	0	0	96
	24	0	1	0	0	48
	24	1	1	1	0	48
	24	2	1	0	0	24
	24	2	2	2	1	12
1,296	28	0	0	0	0	832
	28	1	0	0	0	416
	28	0	0	1	0	32
	28	1	0	1	1	16
648	30	0	0	0	0	640
	30	0	0	0	1	8
648	32	0	0	0	0	320
	32	1	0	0	0	160
	32	0	1	0	0	96
	32	1	1	1	0	48
	32	0	3	0	0	16
	32	1	3	3	0	8
3,888	34	0	0	0	0	3,888
6,732	36	0	0	0	0	6,064
	36	1	0	0	0	32
	36	0	1	0	0	480
	36	0	0	1	0	128
	36	1	0	1	1	16
	36	0	2	0	1	12
7,776	38	0	0	0	0	7,776
9,234	40	0	0	0	0	8,704
	40	0	1	0	0	240
	40	0	0	1	0	192
	40	0	0	0	1	56
	40	0	1	2	0	24
	40	0	3	0	0	16
	40	0	6	0	1	2
14,472	42	0	0	0	0	14,416
	42	0	0	0	1	56
12,636	44	0	0	0	0	12,144
	44	0	0	1	0	288
	44	0	1	0	0	192
	44	0	2	0	1	12
5,184	46	0	0	0	0	5,136
1,944	48	0	0	0	0	1,776
	48	0	0	1	0	96
	48	0	1	0	0	48
	48	0	1	2	0	24
648	50	0	0	0	0	640
24	54	0	0	0	0	16
	54	0	0	0	1	8

- column 3 enumerates the numbers of simple derivatives which are equal to 0 for the evaluated functions (number of independent variables);
- in $\mathbb{B}^4$ there are six vectorial derivatives with regard to two variables; column 4 specifies how many of these vectorial derivatives are equal to 0;
- in $\mathbb{B}^4$ there are four vectorial derivatives with regard to three variables; column 5 specifies how many of these vectorial derivatives are equal to 0;
- in $\mathbb{B}^4$ there is one vectorial derivative with regard to all four variables; a value 1 in column 6 specifies that this vectorial derivative is equal to 0;
- the most right column 7 gives the numbers of functions with the properties introduced above.

As could be expected, functions of $\mathbb{B}^4$ which do not depend on all four variables have small values of $|\mathrm{SNF}(f(\mathbf{x}))|$. Column 3 of Table 5 shows the number of variables the evaluated functions do not depend on. The complete evaluation of all Boolean functions of $\mathbb{B}^4$ reveals that there are simple functions which depend on all four variables, but are independent of the common change of more than one variable; e.g., 48 functions with $|\mathrm{SNF}(f(\mathbf{x}))| = 24$ and one vectorial derivative with regard to two variables which is equal to 0, 32 functions with $|\mathrm{SNF}(f(\mathbf{x}))| = 28$ and one vectorial derivative with regard to three variables which is equal to 0, 8 functions with $|\mathrm{SNF}(f(\mathbf{x}))| = 30$ and one vectorial derivative with regard to all four variables which is equal to 0, and many more.

An interesting result is that there are also simple functions, which depend on all variables, and which also depend on all other directions of change. An example is the function $f(\mathbf{x}) = x_1\, x_2\, x_3\, x_4$ with $|\mathrm{SNF}(f(\mathbf{x}))| = 16$ for which neither any simple derivative nor any vectorial derivative is equal to 0.

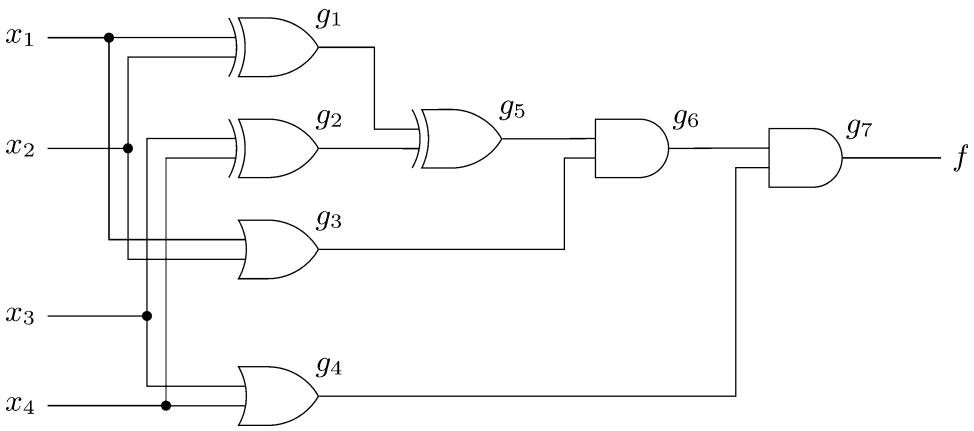

Fig. 2 Circuit structure of the function (14) designed by bi-decomposition controlled by the independence of variables.

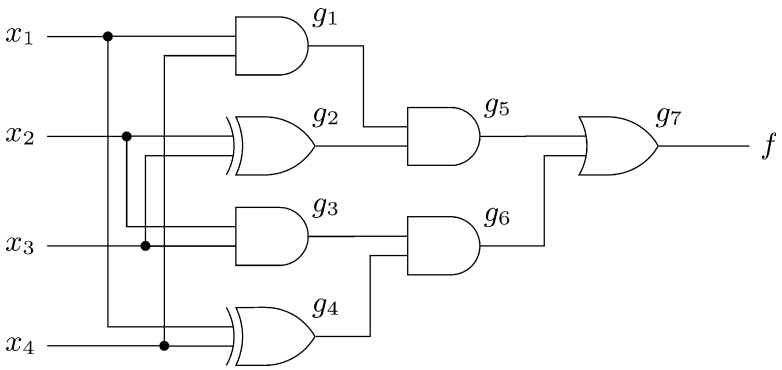

Fig. 3 Circuit structure of the function (14) designed by bi-decomposition controlled by $|\text{SNF}(f)|$.

6 Example of a Bi-Decomposition Controlled by $|\text{SNF}(f)|$

A simple example shows the advantage of the bi-decomposition controlled by $|\text{SNF}(f)|$. The function (14) is a symmetric function. It is known that there is no classical bi-decomposition for this function.

$$f = x_1\,x_2\,x_3\,\overline{x}_4 \vee x_1\,x_2\,\overline{x}_3\,x_4 \vee x_1\,\overline{x}_2\,x_3\,x_4 \vee \overline{x}_1\,x_2\,x_3\,x_4 \tag{14}$$

The classical bi-decomposition approach requires two weak AND-bi-decompositions (realized by the AND-gates g_6 and g_7 of Figure 2) before an XOR-bi-decomposition (realized by the XOR-gate g_5 of Figure 2) can be applied. Due to the weak AND-bi-decompositions there is no balanced path length. The shortest path contain only two gates (g_4 and g_7) and the longest path contains even four gates (g_1, g_5, g_6, and g_7). It should be mentioned that $|\text{SNF}(f)| = 40$ and the first weak AND-bi-decomposition increases the complexity: $|\text{SNF}(g_6)| = 44$, but utilizes the independence: $\frac{\partial g_6(\mathbf{x})}{\partial(x_3,x_4)=0}$.

The new bi-decomposition controlled by $|\text{SNF}(f)|$ decomposes the same function $f(\mathbf{x})$ (14) with $|\text{SNF}(f)| = 40$ into two subfunctions g_5 and g_6 (see Figure 3). Each of these two subfunctions depends on all four variables. Hence, the condition of the classical bi-decomposition is not achieved. However, these two subfunctions are simpler measured by the number of cubes in the SNF:

$$|\text{SNF}(g_5)| = |\text{SNF}(g_6)| = 24 \tag{15}$$

and one vectorial derivative of theses functions with regard to two variables is equal to 0. The number of gates remains 7, but the benefit of the circuit structure of Figure 3 is that all paths contain the same number of only two gates. Hence, the proof of concept of the bi-decomposition controlled by $|\text{SNF}(f)|$ is achieved.

7 Conclusions

The experimental results in Table 5 confirm that not only zero-functions of simple derivatives but also zero-functions of vectorial derivatives indicate simple functions. The number of cubes in the SNF is an additional indicator for a more

general bi-decomposition approach. The comparison of the circuit structures of the same function (14) using the classical bi-decomposition in Figure 2 and the new extended bi-decomposition in Figure 3 confirm that the proof of concept of the bi-decomposition controlled by $|\mathrm{SNF}(f)|$ is achieved.

References

1. Bochmann, D., Dresig, F., and Steinbach, B.: *A New Decomposition Method for Multilevel Circuit Design*. European Design Automation Conference, Amsterdam, The Netherlands, 1991, pp. 374–377.
2. Le, T.Q. Testbarkeit kombinatorischer Schaltungen—Theorie und Entwurf (in German). Dissertation thesis, Technical University Karl-Marx-Stadt, Karl-Marx-Stadt, Germany, 1989.
3. Mishchenko, A., Steinbach, B., and Perkowski, M.: *An Algorithm for Bi-Decomposition of Logic Functions*. in: Proceedings of the 38th Design Automation Conference 2001. June 18-22, 2001, Las Vegas (Nevada), USA, 2001, pp. 103–108.
4. Posthoff, Ch. and Steinbach, B.: Logic Functions and Equations - Binary Models for Computer Science. Springer, Dordrecht, The Netherlands, 2004.
5. Steinbach, B.: *Generalized Lattices of Boolean Functions Utilized for Derivative Operations*. in: Materiały konferencyjne KNWS'13, Łagów, Poland, 2013, pp. 1–17.
6. Steinbach, B. and Mishchenko, A.: *SNF: A Special Normal Form for ESOPs*. in: Proceedings of the 5th International Workshop on Application of the Reed-Muller Expansion in Circuit Design (RM 2001), August 10–11, 2001, Mississippi State University, Starkville (Mississippi), USA, 2001, pp. 66–81.
7. Steinbach, B. and Posthoff, Ch.: *Derivative Operations for Lattices of Boolean Functions*. in: Proceedings Reed-Muller Workshop 2013, Toyama, Japan, 2013, pp. 110–119.
8. Steinbach, B. and De Vos, A.: The *Shape of the SNF as a Source of Information*. in: Steinbach, B. (Ed.): Boolean Problems, Proceedings of the 8th International Workshops on Boolean Problems, September 18–19, 2008, Freiberg University of Mining and Technology, Freiberg, Germany, 2008, pp. 127–136.