Research Article

Yavuz Can*, Önder Yaz, and Dietmar Fey

Disjoint Sum of Products by Orthogonalizing Difference-Building

https://doi.org/10.1515/eng-2020-0067
Received May 21, 2019; accepted May 07, 2020

Abstract: The orthogonalization of Boolean functions in disjunctive form, that means a Boolean function formed by sum of products, is a classical problem in the Boolean algebra. In this work, the novel methodology ORTH[⊕] of orthogonalization which is an universally valid formula based on the combination technique »orthogonalizing difference-building « is presented. Therefore, the technique ⊕ is used to transform Sum of Products into disjoint Sum of Products. The scope of orthogonalization will be solved by a novel formula in a mathematically easier way. By a further procedure step of sorting product terms, a minimized disjoint Sum of Products can be reached. Compared to other methods or heuristics ORTH[⊕] provides a faster computation time.

Keywords: Sum of Products, disjoint Sum of Products, Minimization, Orthogonalization, Disjunctive Form

1 Introduction and Preliminaries

A Boolean function of \( n \) variables is defined as the mapping \( f(\mathbf{x}) : \{0, 1\}^n \rightarrow \{0, 1\} \). Four normal forms of Boolean functions exist, the disjunctive normal form (DNF), conjunctive normal form (CNF), antivalence normal form (ANF) and equivalence normal form (ENF), which consist of either product terms \( p_k(\mathbf{x}) := \bigwedge_{i=1}^n x_i = x_1 \land \ldots \land x_n \) or sum of terms \( s_k(\mathbf{x}) := \bigvee_{i=1}^n x_i = x_1 \lor \ldots \lor x_n \) (with \( n \geq 1 \) as the number of the variables; dimension) in which variables are either negated \( \overline{x}_i \) or not-negated \( x_i \) [1, 2]. The normal form is the canonical representation of the Boolean function. That means that all given variables are included in a product term or sum of term respectively. The reduced form, i.e. non-canonical representation of terms, are called disjunctive, conjunctive, antivalence and equivalence forms DF, CF, AF and EF. The disjunctive form is also considered as the Sum of Products (SOP) and notes as

\[
SOP(\mathbf{x}) := \bigvee_{k=1}^N p_k(\mathbf{x}) \quad \text{with} \quad p_k(\mathbf{x}) := \bigwedge_{i \in A} x_i \quad . \tag{1}
\]

\( A \) is the index set of the running index \( i \). The AF is a special form of Exclusive-Or Sum of Products (ESOP) and is defined as

\[
ESOP(\mathbf{x}) := \bigoplus_{k=1}^N p_k(\mathbf{x}) \quad , \tag{2}
\]

with \( N > 1 \) as the upper bound of the number of the product terms [?].

The orthogonality of a Boolean functions is a special attribute. A function is orthogonal if their terms have the characteristic of being disjoint in pairs in at least one variable. Thus, the following applies for the disjoint Sum of Products (dSOP):

\[
dSOP(\mathbf{x}) := \bigvee_{k=1}^N p_k(\mathbf{x}) \quad \text{whereby} \quad p_i(\mathbf{x}) \land p_j(\mathbf{x}) = 0 \quad . \tag{3}
\]

An orthogonal representation of a SOP, that means dSOP, is characterized by product terms which are disjoint to one another in pairs [3, 4]. Consequently, the intersection of these product terms results in 0. The orthogonal representation of a DF - disjoint Sum of Products - is equal to the orthogonal form of an AF - the disjoint Exclusive-Or Sum of Products. In this case, it applies \( dSOP(\mathbf{x}) = dESOP(\mathbf{x}) \) [3–5]. That means, the dSOP is equivalent to dESOP consisting of the same product terms and differ only in the logical connectivity between the product terms. This relationship can be explained well with the following definition out of [6], if both product terms \( p_i(\mathbf{x}) \) and \( p_j(\mathbf{x}) \) are disjoint to each other. A SOP of two product terms can be transformed into an ESOP by:

\[
p_i(\mathbf{x}) \lor p_j(\mathbf{x}) = p_i(\mathbf{x}) \oplus p_j(\mathbf{x}) \oplus (p_i(\mathbf{x}) \land p_j(\mathbf{x})) = 0 \quad , \tag{4}
\]
In the special case, that both products terms are disjoint, building their conjunction results to 0. As \( x_1 \oplus 0 = x_1 \) is, following relation follows from the Eq. (4):

\[
p_i(\overline{x}) \lor p_i(x) = p_i(x) \oplus p_i(\overline{x}).
\]  

(5)

In this case, the left side is equal to the right side which means that a dSOP is equivalent to a dESOP. It applies \( dSOP(x) = dESOP(x) \).

\[
\begin{array}{c|c|c|c|c|c}
 x_1 & x_2 & x_3 & dSOP(x) = \overline{x}_2 \lor x_1 x_3 & \overline{x}_2 \lor x_1 x_2 x_3 \\
\hline
 0 & 0 & 1 & 1 & 0 \\
 0 & 1 & 1 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 \\
 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c}
 x_1 & x_2 & x_3 & SOP(x) = \overline{x}_2 \lor x_1 x_3 & \overline{x}_2 \lor x_1 x_2 x_3 \\
\hline
 0 & 0 & 1 & 1 & 0 \\
 0 & 1 & 1 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 \\
 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

Figure 1: Difference between SOP and dSOP in a K-map.

In a K-map a dSOP is characterized by non-overlapping cubes (Figure 1). Special calculations can be easier solved in another form. It simplifies the handling of further calculations in applications of electrical engineering, e.g. calculation of suitable test patterns for combinational circuits for verifying feasible logical faults, which can mathematically be determined by Boolean Differential Calculus (BDC) \([1, 7]\). That means, the orthogonalization of a SOP facilitates the transformation into an equivalent dESOP \([1, 3, 8]\) and this characteristic simplifies the handling of BDC especially in Ternary-Vector-List (TVL) arithmetic \([3, 9–11]\). Due to the restricted number of variables the terms of SOP are not priory disjoint. However, the disjoint form can be calculated by using a novel Boolean formula based on the novel combining technique of »orthogonalizing difference-building \(\ominus««.

2 Method of Orthogonalization

2.1 Orthogonalizing Difference-Building \(\ominus\)

Orthogonalizing difference-building \(\ominus\) is the composition of two calculation steps - the usual difference-building out of the set theory and the subsequent orthogonalization, as shown in Figure 2. The result of \(\ominus\) is orthogonal in contrast to the result out of the method difference-building. Both results are different in their representations but homogenous in their covering of 1s. They only differ in their form of coverage, whereas the method \(\ominus\) constitutes the solution in already orthogonal form. This method \(\ominus\) is generally valid and equivalent to the usual method of difference-building \([3]\). The orthogonalizing difference-building \(\ominus\) corresponds to the removal of the intersection which is formed between the minuend product term \(p_m(x)\) and the subtrahend product term \(p_s(x)\) from the minuend product term \(p_m(x)\), which means \(p_m(x) - (p_m(x) \land p_s(x))\). The result consists of several product terms which are pairwise disjoint to each other. The Equation (6) applies with \(n, n' \in \mathbb{N}\) as the dimension of \(p_m(x)\) and \(p_s(x)\). In this case, the formula \((\bigvee_{i=1}^{n} x_i = x_1 \lor x_1, x_2 \lor x_4, \ldots, x_n)\) from \([4]\) is used to describe the orthogonalizing difference-building in a mathematically easier way. The method of orthogonalizing difference-building \(\ominus\) is demonstrated by the following Example 1.

\[\begin{align*}
& x_1 \ominus x_2 x_3 x_4 = x_1 x_2 \overline{x}_3 \lor x_1 x_2 x_3 \lor x_1 x_2 x_3 \overline{x}_4 \\
\end{align*}\]

Example 1: A subtrahend \(p_s(x) = x_2 x_3 x_4\) is subtracted from a minuend \(p_m(x) = x_1\) and it appears a result, which consist of pairwise disjoint product terms.

The explanation of Eq. (6) is given by the following points:

- The first literal of the subtrahend, here \(x_2\), is taken complementary and build the intersection with the minuend, here \(x_1\). Consequently, the first term of the difference is \(x_1 \overline{x}_2\).
- Then the second literal, here \(x_3\), is taken complementary and the intersection with the minuend and the first literal \(x_2\) of the subtrahend is built. Therefore, the second term is \(x_1 x_2 \overline{x}_3\).
\[ p_m(x) \ominus p_s(x) = \bigcap_{m=1}^{n} x_m \bigcap_{s=1}^{n'} x_s := \bigcap_{m=1}^{n} x_m \cap \bigvee_{s=1}^{n'} \overline{x_s} = (x_1 \cdot \ldots \cdot x_n)_m \land (\overline{x_{1j}} \lor \ldots \lor x_{1j} \cdot \ldots \cdot x_{(n-1)j} \overline{x_{n'}}) \] \hfill (6)

- Following the next literal, here \( x_a \), is taken complementary and the intersection with the minuend and the first literal \( x_3 \) and second literal \( x_3 \) of the subtrahend is built. Thus, the third term of the difference is \( x_1 x_2 x_3 \overline{x_4} \).
- This process is continued until all literals of the subtrahend are singly complemented and linked by building the intersection with the minuend in a separate term.

\( n_{orth} \) as the number of product terms in the orthogonal result corresponds to the number of literals presented in the subtrahend \( p_s(x) \) and are not presented in the minuend \( p_m(x) \) at the same time. Following rules must be followed to get correct results for the application of \( \ominus \):

1. If the subtrahend is already orthogonal to the minuend (\( p_s(x) \perp p_m(x) \)) the result corresponds to the minuend:
   
   \[ p_m(x) \ominus p_s(x) = p_m(x) |_{p_s(x) \perp p_m(x)} \] \hfill (7)

2. The difference between 0 and the subtrahend is the subtrahend itself:
   
   \[ 0 \ominus p_s(x) = p_s(x) \] \hfill (8)

3. The result between 1 and subtrahend is the complement of the subtrahend which results in a dSOP:
   
   \[ 1 \ominus p_s(x) = dSOP(x) \bigcap_{p_s(x)} \] \hfill (9)

4. Thereby, the subset symbol \( \subseteq \) of the set theory is transferred to switching algebra. The result between subtrahend and minuend is empty if the subtrahend is already completely contained in the minuend. If the subtrahend is the subset of the minuend (\( p_s(x) \subseteq p_m(x) \)), the result is 0:
   
   \[ p_m(x) \ominus p_s(x) = 0 |_{p_s(x) \subseteq p_m(x)} \] \hfill (10)

### 2.2 Orthogonalization of SOP

#### 2.2.1 Mathematical Methodology

The orthogonalization of every SOP(\( x \)) consisting of at least two product terms \( (N > 1) \) can be performed by Eq. (11) which bases on the Eq. (6), which is based on the combination technique of \( \ominus \) [3]. The order of the calculation is important. That means, the first two product terms must be calculated and then the third product term must be calculated with the result of the two product terms before, and so on. The result of \( dSOP(x) \) can diversify depending on the starting product term. As a SOP has the characteristic of being commutative, the order of their product terms can be changed for getting results with fewer number of disjoint product terms called as \( N_{orth} \). To obtain better result is often reached by ordering the product terms from higher number of variables to fewer number of variables. Following Example 2 gives an overview about the procedure orthogonalizing by Eq. (11) and afterwards the Example 3 with an additional process of sorting.

**Example 2:** Function SOP(\( x \)) = \( x_1 \lor x_1 x_2 \lor x_1 x_3 \) has to be orthogonalized by Eq. (11).

\[ dSOP(1) = (\overline{x_3} \lor x_1 x_2 \lor x_1 x_3) \lor (x_1 x_2 \lor x_1 x_3) \lor x_1 x_3 \]

Eq. (7)

\[ = x_2 x_3 \lor x_1 x_2 x_3 \lor x_1 x_2 x_3 \lor x_1 x_3 \]

Function \( dSOP(1) \) consists of four disjoint product terms \( (N_{orth} = 4) \) and is the orthogonalized form of SOP(\( x \)). Both are equivalent. They only differ in their form of coverage which is illustrated in the K-maps as shown in Figure 3.

**Example 3:** Now, the sorted function \( sort\ dSOP(1) = x_1 x_2 \lor x_1 x_3 \lor \overline{x_3} \) of Example 2 has to be orthogonalized by Eq. (11).

\[ sort\ dSOP(1) = (x_1 x_2 \lor x_1 x_3 \lor \overline{x_3}) \lor (x_1 x_3 \lor \overline{x_3}) \lor \overline{x_3} \]

Eq. (7)

\[ = (x_1 x_2 \lor \overline{x_3}) \lor x_1 x_3 \lor \overline{x_3} = 0 \]

Eq. (10)

\[ = x_1 x_3 \lor \overline{x_3} \]

Function \( sort\ dSOP(1) \) is another equivalent orthogonal form of SOP(\( x \)) which consists of two disjoint product terms \( (N_{orth} = 2) \) that is also illustrated in third K-map (Figure 3). The coverage of is done by two cubes. By sorting, a minimized dSOP can be reached. The comparison of the three functions shows their equivalence. They are homogenous and only differ in their form of superimposition.
\[ dSOP(x) := \bigvee_{k=0}^{N-1} \left( \bigotimes_{i=k+1}^{N} p_i(x) \right) = (p_1(x) \lor \ldots \lor p_N(x)) \lor (p_2(x) \lor \ldots \lor p_N(x)) \lor \ldots \lor (p_{N-1}(x) \lor p_N(x)) \lor p_N(x) \] (11)

2.2.2 Algorithm

The corresponding Algorithm ORTH[\circ], whose pseudo code is shown in the Table 1 outlines the computational procedure of orthogonalization of a SOP according to the formula in Eq. (11). To obtain dSOP with fewer number of product terms, the sub-functions absorb() and sort() are additionally used. absorb() is a function which reduces the number of product terms of the SOP, which serves as the input of the algorithm. The reduction is achieved by absorption of smaller product terms, which consists of higher number of variables, by larger product terms, which consists of lower number of variables if those are already covered by the larger ones (following example):

\[ \ldots \lor x_1x_2x_3 \lor x_1x_2 \overline{x}_3 \lor x_1 \lor \ldots \xrightarrow{\text{absorb}()} \ldots \lor x_1 \lor \ldots \]

The product term \( x_1 \) is absorbing the other two product terms. Additionally, absorb() reduces duplicated product terms to a single term which is demonstrated with the following example:

\[ \ldots \lor x_2x_3 \lor x_3 \lor x_2 \overline{x}_3 \lor \ldots \xrightarrow{\text{absorb}()} \ldots \lor x_2x_3 \lor x_3 \lor \ldots \]

Consequently, by using absorb() the number of product terms, that have to be treated decreases. With the optionally function sort() follows the resorting of the product terms from smaller product terms to larger product terms. After proceeding these two sub-functions absorb() and sort() the process of orthogonalization ORTH[\circ] according to the method \( \circ \) is performed.

### 3 Comparison and Measurement

#### 3.1 \( N_{\text{orth}} \) before and after Sorting

However, to make a statement about the optimized form, the optimum minimization would have to be defined, which has not yet been clarified. Table 2 illustrates the percentage of reduced terms by the use of subsequent procedure of sorting. The procedure of sorting brings an advantage for gaining minimized dSOP. Firstly, a list of ten non-orthogonal functions in respect to \( N = \{5, 10, 15\} \) and dimension \( x_n = \{5, 6, \ldots, 50\} \) were created. Consequently, per each \( N \) has produced 50 different non-orthogonal SOPs. Subsequently, each SOP was orthogonalized according to the method \( \circ \) before and after sorting. The resulting number of product terms \( N_{\text{orth}} \) in dSOP and \( sort \) \( N_{\text{orth}} \) in \( sort \) dSOP in respect to \( N \) and \( x_n \) were determined (Figure 4) and compared. The number of product terms of a sorted SOP is fewer than the unsorted SOP. It follows \( sort N_{\text{orth}}(N, x_n) < N_{\text{orth}}(N, x_n) \). An average value of these values was calculated for each dimension \( x_n \). Thereby, the results of the quotients of the average number of disjoint product terms were obtained. Furthermore,
Table 2: Percentage Value of \( \frac{N_{orth}}{N_{orth}} \)

| \( x_n \) | \( \frac{N_{orth}}{N_{orth}} \) (5) | \( \frac{N_{orth}}{N_{orth}} \) (10) | \( \frac{N_{orth}}{N_{orth}} \) (15) |
|---|---|---|---|
| 5 | 0.0 | 8.6 | — |
| 6 | 23.6 | 29.0 | — |
| 7 | 28.8 | 11.8 | 24.6 |
| 8 | 30.2 | 27.0 | 28.7 |
| 9 | 73.3 | 52.0 | 32.8 |
| 10 | 6.1 | 24.6 | 29.6 |
| … | … | … | … |
| 28 | 13.6 | 10.5 | 40.9 |
| 29 | 2.7 | 58.6 | 22.3 |
| 30 | 38.6 | 25.8 | 50.6 |
| 31 | 17.8 | 8.5 | 18.0 |
| … | … | … | … |
| 45 | —1.6 | 9.4 | —2.8 |
| 46 | —5.4 | 6.5 | 13.1 |
| 47 | 14.8 | 0.6 | 38.4 |
| 48 | 8.8 | 20.2 | 16.4 |
| 49 | 1.2 | —13.1 | 18.6 |
| 50 | —2.5 | 30.6 | 8.8 |
| \( \emptyset \) | 17.3 | 21.7 | 27.6 |

The average values of \( N_{orth} \) and \( \text{sort} N_{orth} \) in percentage value were gained. The minus signifies that in that case the number of \( N_{orth} \) is fewer. Finally, a total average percentage value per each \( N \) was determined out of these average values (Table 2). Three in their number. Consequently, an additional procedure of sorting leads to a dSOP with fewer number of product terms. Minimalization of approximately 17% till 28% is obtained in comparison to a dSOP which has not been sorted before.

### 3.2 Comparison in Number of Terms \( N_{orth} \)

The number of the disjoint product terms \( N_{orth} \) in a dSOP produced by \( \text{ORTH} [\text{-}] \) is analyzed in comparison to the heuristic \( \text{ORTH}[\text{DSOP}] \) in [5], the method \( \text{ORTH}[m1] \) in [12] and a varied form of \( \text{ORTH}[\text{DSOP}] \) as \( \text{ORTH}[\text{DSOP}] \). In the varied form \( \text{ORTH}[\text{DSOP}] \) the minimization function “espresso.exe” was replaced by \( \text{absorb}() \). The comparisions in respect to \( N = \{2, 5, 10, 15, 20, 25\} \) and \( x_n = \{1, 2, \ldots, 50\} \) are shown in Figure 5. The corresponding average values \( N_{orth} \) and the ratios to the method \( \text{ORTH}[\text{-}] \) are given in Table 3. The average value \( N_{orth} \) for each \( N \) are formed of 50 calculated tasks for each dimension \( x_n \). Out of these average values of \( N_{orth} \) regarding to a method an average value \( \overline{N_{orth}} \) is built. The corresponding charts, as shown in the Figures 5a) - f), illustrate that the method \( \text{ORTH}[\text{-}] \) offers results with fewer number of terms \( N_{orth} \) in respect to growing \( x_n \) and \( N \) in comparison to the methods \( \text{ORTH}[m1] \) and \( \text{ORTH}[\text{DSOP}] \). The number of terms are 1.67 times fewer by \( \text{ORTH}[\text{-}] \) in comparison to \( \text{ORTH}[m1] \). Against this, the heuristic \( \text{ORTH}[\text{DSOP}] \) provides approximately 50% fewer number of terms \( N_{orth} \) in contrast to the method \( \text{ORTH}[\text{-}] \). Therefore the relationship \( N_{orth}[\text{DSOP}] < N_{orth}[\text{-}] < N_{orth}[m1] \) can be deduced from this. Probably, this benefit is given by the use of “espresso.exe”, which is a program used as heuris-
Disjoint Sum of Products by Orthogonalizing Difference-Building

(a) SOPs with 2 product terms

(b) SOPs with 5 product terms

(c) SOPs with 10 product terms

(d) SOPs with 15 product terms

(e) SOPs with 20 product terms

(f) SOPs with 25 product terms

Figure 5: Average number of the $N_{orth}$ in the dSOP

Table 3: Average of the number of terms $N_{orth}$ and relation to ORTH[$\ominus$]

| $x_n$ | in $N_{orth}$ | in $N_{orth}$ | in $N_{orth}$ | in $N_{orth}$ | ORTH[$\ominus$] | ORTH[DSOP] | ORTH[DSOP2] | ORTH[m1] |
|-------|---------------|---------------|---------------|---------------|-----------------|------------|-------------|----------|
| 2     | 6,22          | 6,30          | 6,30          | 9,45          | 1,01            | 1,01       | 1,52        |
| 5     | 29,52         | 18,57         | 29,11         | 43,22         | 0,63            | 0,99       | 1,46        |
| 10    | 86,72         | 44,96         | 88,32         | 127,35        | 0,52            | 1,02       | 1,47        |
| 15    | 164,05        | 77,82         | 165,90        | 249,31        | 0,47            | 1,01       | 1,52        |
| 20    | 248,18        | 115,97        | 258,58        | 431,67        | 0,47            | 1,04       | 1,47        |
| 25    | 369,09        | 162,79        | 393,47        | 650,57        | 0,44            | 1,07       | 1,76        |
| $\emptyset$ | 150,63 | 71,07 | 156,95 | 251,93 | 0,47 | 1,04 | 1,67 |
Figure 6: Comparison in computation time

tic logic minimizer. In contrast to our function \textit{absorb()}, the number of terms in the minimized result is fewer by “\texttt{espresso.exe}”. This step of minimization is important because a further calculation of a dSOP with fewer number of product terms needs fewer operations and is carried out by reduced computation time. In this case, a further calculation of a dSOP such as the Boolean Differential Calculus (BDC) is performed with fewer number of product terms and thus reduces the number of further operations and the number of calculation steps which will certainly affect the computation time.

3.3 Comparison in Computation Time

In this section the comparison of all four approaches relating to the computation time in respect to $N = \{2, 5, 10, 15, 20, 25\}$ and $x_n = \{1, 2, \ldots, 50\}$ as shown in the Figures 6a) - f) is given. The corresponding average values of the calculation times and the ratios to
the method $ORTH[\ominus]$ are given in the Table 4. The computation times of method $ORTH[\ominus]$ is faster in comparison to the heuristic $ORTH[DSOP]$, $ORTH[m1]$ and the varied form $ORTH[DSOP2]$. The complexity class of $ORTH[\ominus]$ totals up to $\Theta(n^3)$. The distinction is that the novel method $ORTH[\ominus]$ calculates the orthogonalizing difference-building $\ominus$ consistently no matter if two product terms are orthogonal or not. As this consideration takes place in method $ORTH[m1]$ in [1], the computation time is likely to be deteriorated. Thus unnecessary calculations are not to be carried out in the method $ORTH[\ominus]$. By the use of “espresso.exe” in $ORTH[DSOP]$ the procedure time of orthogonalization is more slowly. This is also confirmed by replacing the function with $absorb()$, which is shown by the charts of $ORTH[DSOP2]$. Therefore, its computation time is decelerated. Due to the replacement of the minimization function by $absorb()$ in $ORTH[DSOP2]$ the calculation time gets faster in comparison to $ORTH[DSOP]$. However, it is still higher than the computation time of the novel method $ORTH[\ominus]$. In summary, it has to be clarified here that the new method has faster computation time than the other approaches. $ORTH[\ominus]$ is approximately 1000 times faster in comparison to $ORTH[DSOP]$, approximately 25 times faster than $ORTH[DSOP2]$ and twice as fast than $ORTH[m1]$. Even if the two sub-functions $absorb()$ and $sort()$ are excluded, the method $ORTH[\ominus]$ provides computation time, which are reduced, as shown in Figure 6a - f). The measurements are limited to the dimension $x_n = 50$. Against this, for dimension $x_n > 50$ similar results are expected.

| $x_n$ | $ORTH[\ominus]$ | $ORTH[DSOP]$ | $ORTH[DSOP2]$ | $ORTH[m1]$ | $ORTH[DSOP]$ | $ORTH[DSOP2]$ | $ORTH[m1]$ |
|------|-----------------|--------------|--------------|------------|--------------|--------------|------------|
| 2    | 7.02            | 196545.63    | 100.48       | 33.09      | 27996.18     | 14.31        | 4.71       |
| 5    | 65.79           | 338287.65    | 988.83       | 191.86     | 5142.02      | 15.03        | 2.92       |
| 10   | 297.52          | 580726.60    | 5586.03      | 667.70     | 1951.86      | 18.78        | 2.24       |
| 15   | 720.68          | 714265.14    | 15323.50     | 1520.70    | 991.10       | 21.26        | 2.11       |
| 20   | 1298.12         | 948014.46    | 31221.00     | 3023.73    | 730.30       | 24.05        | 2.33       |
| 25   | 2211.61         | 1443192.99   | 65237.83     | 5149.57    | 652.55       | 29.50        | 2.33       |
| $\Theta$ | 766.79       | 703505.41     | 19742.95      | 1764.44    | 917.47       | 25.75        | 2.30       |

the varied form $\ominus$ difference-building takes place in method terms are orthogonal or not. As this consideration noval method $ORTH[\ominus]$ are not to be carried out in the method $ORTH[\ominus]$. In contrast, its computation time is decelerated. Due to the replacement of the minimization function by $absorb()$ in $ORTH[DSOP]$ the calculation time gets faster in comparison to $ORTH[DSOP]$. However, it is still higher than the computation time of the novel method $ORTH[\ominus]$. In summary, it has to be clarified here that the new method has faster computation time than the other approaches. $ORTH[\ominus]$ is approximately 1000 times faster in comparison to $ORTH[DSOP]$, approximately 25 times faster than $ORTH[DSOP2]$ and twice as fast than $ORTH[m1]$. Even if the two sub-functions $absorb()$ and $sort()$ are excluded, the method $ORTH[\ominus]$ provides computation time, which are reduced, as shown in Figure 6a - f). The measurements are limited to the dimension $x_n = 50$. Against this, for dimension $x_n > 50$ similar results are expected.

### Summary and Conclusions

This work introduced a generally valid method of »orthogonalizing difference-building $\ominus$« which is used to calculate the orthogonal difference of two product terms. Furthermore, rules for this method were explained which must be followed to get correct results. By a novel formula based on the combining technique $\ominus$ every Sum of Products (SOP) can easily be orthogonalized mathematically. Thus, we get disjoint Sum of Products (dSOP). A minimized dSOP can also be reached by two additional procedures of sorting and absorbing of terms before the process of orthogonalization. By sorting the product terms of a SOP are resorted from smaller number of variables to higher number of variables. This resorting brings an advantage of approximately 17% and 26% depending on $N$ to reach minimized dSOP. The corresponding Algorithm $ORTH[\ominus]$ was compared to other algorithms $ORTH[DSOP]$, $ORTH[DSOP2]$ and $ORTH[m1]$ in their number of product terms in the calculated dSOP and the computation time. $ORTH[DSOP]$ determines fewer number of product terms in contrast to $ORTH[\ominus]$. However, the reduction of the product terms by $ORTH[\ominus]$ is about 50% in contrast to $ORTH[m1]$. Furthermore, the novel method $ORTH[\ominus]$ provides approximately 1000 times faster computation in comparison to $ORTH[DSOP]$ and is approximately 25 times faster in comparison to $ORTH[m1]$. The number of terms in the orthogonalized result by the method $ORTH[\ominus]$ can probably reduced by an additional absorption of the disjoint product terms. For that, a postfunction for absorption could be developed, which retains the property of orthogonality.

### References

[1] D. Bochmann, *Binäre Systeme - Ein Boolean Buch*. Hagen, Germany: LiLoLe-Verlag, 2006.

[2] B. Steinbach and C. Posthoff, “An extended theory of boolean normal forms,” in Proc. 6th Annual Hawaii International Conference on Statistics, Mathematics and Related Fields, (Hawaii, USA), pp. 1124–1139, 2007.

[3] Y. Can, *Neue Boolesche Operative Orthogonalisierende Methoden und Gleichungen*. Erlangen, Germany: FAU University Press,
1 ed., 2016.

[4] Y. Crama and P. Hammer, *Boolean Functions - Theory, Algorithms, and Applications*. Cambridge, UK: Cambridge University Press, 2011.

[5] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "New Heuristic for DSOP Minimization," in *Proc. 8th International Workshop on Boolean Problems (IWSBP)*, (Freiberg (Sachsen), Germany), 2008.

[6] H. J. Zander, *Logischer Entwurf binärer Systeme*. Berlin, DDR: Verlag Technik, 1989.

[7] B. Steinbach, "The Boolean Differential Calculus – Introduction and Examples," in *Proc. Reed-Muller Workshop 2009*, (Naha, Okinawa, Japan), pp. 107–117, 2009.

[8] Y. Can, H. Kassim, and G. Fischer, "New Boolean Equation for Orthogonalizing of Disjunctive Normal Form based on the Method of Orthogonalizing Difference-Building," *Journal of Electronic Testing. Theory and Application (JETTA)*, vol. 32, no. 2, pp. 197–208, 2016.

[9] Y. Can, H. Kassim, and G. Fischer, “Orthogonalization of DNF in TVL-Arithmetic,” in *Proc. 12th International Workshop on Boolean Problems (IWSBP)*, (Freiberg (Sachsen), Germany), 2016.

[10] C. Dorotska and B. Steinbach, “Orthogonal Block Change & Block Building Using Ordered Lists of Ternary Vectors,” in *Proc. 5th International Workshop on Boolean Problems (IWSBP)*, (Freiberg (Sachsen), Germany), pp. 91–102, 2002.

[11] C. Posthoff and B. Steinbach, *Logikentwurf mit XBOOLE. Algorithmen und Programme*. Berlin, Germany: Verlag Technik GmbH, 1991.

[12] C. Posthoff and B. Steinbach, *Binäre Gleichungen - Algorithmen und Programme*. Karl-Marx-Stadt (Chemnitz), DDR: Technische Universität Karl-Marx-Stadt, 1979.