BOOLEAN ORTHOGONALIZING COMBINATION METHODS

Yavuz Can[1]and Georg Fischer[2]

[1,2]Institute for Electronics Engineering (LTE),
Friedrich-Alexander University, Erlangen-Nuremberg, Germany
yavuz.can@fau.de
georg.fischer@fau.de

ABSTRACT

In this paper a new logical operation method called "orthogonalizing difference-building" is presented. It is used to calculate the difference, but also the complement of a function as well as the EXOR and EXNOR of two minterms respectively two ternary-vectors or two functions respectively two ternary-vector-lists is presented. On the basis of this new method a further logical operation method called "orthogonal OR-ing" is going to be introduced. The advantages of both methods are their results, which are already available in an orthogonal form that has an essential advantage for continuing calculations. Since it applies, an orthogonal disjunctive normal form is equal to orthogonal antivalence normal form, subsequent Boolean differential calculus will be simplified.

KEYWORDS

Difference-Building, Orthogonality, TVL, Data Memory Request, Computing Time

1. INTRODUCTION

Orthogonality is a special property of Boolean functions because an orthogonal function can be transformed in another form and so it simplifies the handling for further calculations in applications of electrical engineering. For example, the orthogonal form of a disjunctive form is equal to the orthogonal form of an antivalent form, $DNF^{orth} = ANF^{orth}$. This work shows new orthogonalizing methods, which means a new method of OR-ing as well as a method for building the difference of minterms respectively ternary-vectors (TV) or functions respectively ternary-vector-lists (TVL) of disjunctive normal form which result in orthogonal form. On the one hand, the method "othogonalizing difference-building $\ominus$" is similar to the usual difference-building derived from the set theory, but, on the other hand, it provides orthogonal results. It also enables the replacement of EXOR and EXNOR of two minterms (also TVs) or two functions (also TVLs). Thus, orthogonal results can be achieved which has essential advantage for further calculation in particular in the area of TVL-arithmetic. The other method "orthogonalizing OR-ing$\circledvee$" offers the logical disjunction of two minterms respectively two TVs with the characteristic of the resulting in orthogonal form.

2. Theoretical Foundations

2.1. Isomorphism: Set Theory as Basis for Switching Algebra

In Figure1a), the set difference from the set theory is shown. If the set difference is called$M\backslash S$, the remaining set of these two sets is the set of all elements belonging to set M, but not to set S (Mwithout S). In Figure1b), the union of set S_1 and S_2, that is$S_1 \cup S_2$, is illustrated [2][3].

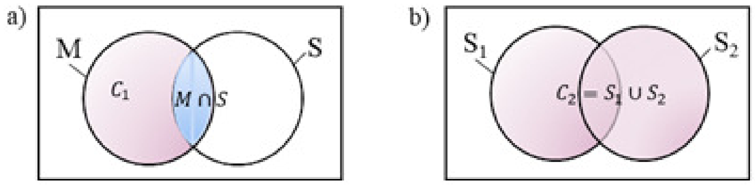


Figure 1: a) Set Difference $M\backslash S$; b) Set Union $S_1 \cup S_2$

Due to the isomorphism the set difference and the set union out of the set theory are expressed in the propositional logic as follows:

$$C_1 = M\backslash S = M \cap \bar{S} \quad \rightarrowtail \quad M \wedge \bar{S} \tag{1}$$

$$C_2 = S_1 \cup S_2 \quad \rightarrowtail \quad S_1 \vee S_2 \tag{2}$$

Thus, an Equation (3) to build the difference of two minterms in the switching algebra is defined out of (1). The minuend minterm is indentified with$m_M := \bigwedge_{m=0}^{n-1} x_{n-m}$, the subtrahend minterm is indentified with $m_S := \bigwedge_{s=0}^{n-1} x_{n-s}$ and with $n \in \mathbb{N}$follows:

$$m_M \backslash m_S = \left(\bigwedge_{m=0}^{n-1} x_{n-m}\right) \backslash \left(\bigwedge_{s=0}^{n-1} x_{n-s}\right) := \left(\bigwedge_{m=0}^{n-1} x_{n-m}\right) \wedge \overline{\left(\bigwedge_{s=0}^{n-1} x_{n-s}\right)} =$$

$$= \left(\bigwedge_{m=0}^{n-1} x_{n-m}\right) \wedge \left(\bigvee_{s=0}^{n-1} \bar{x}_{n-m}\right) = (x_n \cdot x_{n-1} \cdot\ldots\cdot x_1)_M \wedge (x_n \cdot x_{n-1} \cdot\ldots\cdot x_1)_S \tag{3}$$

An Equation (4) building the disjunction of two minterms where the first summand minterm is identified with $m_{S_1} := \bigwedge_{s_1=0}^{n-1} x_{n-s_1}$ and the second summand minterm is identified with $m_{S_2} := \bigwedge_{s_2=0}^{n-1} x_{n-s_2}$is defined out of (2). And with $n \in \mathbb{N}$ follows:

$$m_{S_1} \vee m_{S_2} = \left(\bigwedge_{s_1=0}^{n-1} x_{n-m}\right) \vee \left(\bigwedge_{s_2=0}^{n-1} x_{n-s}\right) = (x_n \cdot x_{n-1} \cdot\ldots\cdot x_1)_{S_1} \vee (x_n \cdot x_{n-1} \cdot\ldots\cdot x_1)_{S_2} \tag{4}$$

Equation (3) and (4) only indicate the value of the variable at the corresponding position in terms of its index. That means $\bar{x}_i$is the complement value of variable at the$i$-th place, whereasx_i hows the value of the variable at the i-th place. The general validity is proved by the method of mathematical induction.

2.2. Ternary-Vector-List (TVL)

Ternary-Vector-Lists are representations of Boolean functions which can take one of three possible values for each point. A non-negated variable is characterized by '1', a negated by '0' and not included by '-'. Boolean equations represented by TVL can be treated computationally easier [5][10]. A TVL consists of *m*-rows (number of conjunctions or disjunctions contained in the function) and *n*-columns (number of independent variables) and is characterized with$n \in \mathbb{N}$and $t \in \{0, 1, -\}$. Any form of a function, that means disjunctive form (DF), conjunctiveform (KF), equivalent form (EF) and antivalence form (AF), can be represented as a TVL. As no operators ($\vee$, $\wedge, \odot, \oplus$) in the TVL presentation are given, the designations of the matrix by *D(f), K(f), E(f)* and *A(f)* show the type of the TVL [5][10][12].

$$TVL = \begin{bmatrix} TV_1 \\ TV_2 \\ \vdots \\ TV_m \end{bmatrix} \coloneqq \begin{array}{r c} & \begin{array}{cccc} x_n & \cdots & x_2 & x_1 \end{array} \\ 1.row & \\ 2.row & \\ \vdots & \\ m.row & \end{array} \begin{bmatrix} t_{1n} \cdots & t_{12} & t_{11} \\ t_{2n} \cdots & t_{22} & t_{12} \\ \vdots \;\; \vdots & \vdots & \vdots \\ t_{mn} \cdots & t_{m2} & t_{m1} \end{bmatrix} = [t_{mn}] \tag{5}$$

The rule for AND-ing ($\wedge$) of two TVs ($TV_{i,j}$) which represent an average TV (TV_k), is defined by Table 1:

Table 1. Rule for AND-ing of two TVs

$\wedge$	0	1	-		
0	0	$\times$	0	if $\times \geq 1$	$\times$: empty set
1	$\times$	1	1		$TV_k = \times$
-	0	1	-	else	
					$TV_k = TV_i \wedge TV_j$

The rule for OR-ing ($\vee$) of two TVs ($TV_{i,j}$), which represent the union of both TVs, is defined as follows:

$$TV_i \vee TV_j = \begin{bmatrix} TV_i \\ TV_j \end{bmatrix} \tag{6}$$

The difference-building of two TVs with $TV_M = [t_n, t_{n-1}, .., t_1]_M$ and $TV_S = [t_n, t_{n-1}, .., t_1]_S$ is distinguished out of Equation (3):

$$TV_M \setminus TV_S = [t_n, t_{n-1}, .., t_1]_M \setminus [t_n, t_{n-1}, .., t_1]_S \coloneqq [t_n, t_{n-1}, .., t_1]_M \wedge \overline{[t_n, t_{n-1}, .., t_1]_S} =$$

$$= [t_n, t_{n-1}, .., t_1]_M \wedge \begin{bmatrix} \bar{t}_n & - & \cdots & - \\ - & \overline{t_{n-1}} & \cdots & - \\ \vdots & \vdots & \vdots & \vdots \\ - & - & \cdots & \bar{t}_1 \end{bmatrix}_S \tag{7}$$

2.3. Orthogonality

A function or TVL is orthogonal if its minterms respectively its TVs are disjoint to one another in pairs at least in one column. Consequently, these minterms ($m_{i,j}$) or these TVs ($TV_{i,j}$) then have no common covering after their logical conjunction. An orthogonal function has no redundant minterm. Thus, the following Equations in (8) can be formulated for the proof of orthogonality [17]:

$$m_i \wedge m_j = 0 \qquad \text{or} \qquad TV_i \wedge TV_j = \times \qquad (8)$$

3. ORTHOGONALIZING DIFFERENCE-BUILDING ⊖

3.1. Method

The method of orthogonalizing difference-building ⊖ illustrated in the Karnaugh map (Fig. 2) corresponds to the removal of the intersection which is formed between the minuend $m_M$and the subtrahend m_S, from the minuend m_M, which means $m_M \backslash (m_M \wedge m_S)$. The result consists of several disjoint minterms, which cover all of the remaining 1s and are pairwise orthogonal to each other. For this purpose, an example is shown in a K-map with 4 variables (Fig. 2),in which a group of 2 (subtrahend: $x_3x_2x_1$) is subtracted from group of 8 (minuend: x_4). The result consists of several blocks (1st Block, 2nd Block, 3rd Block) which are pairwise orthogonal to each other.

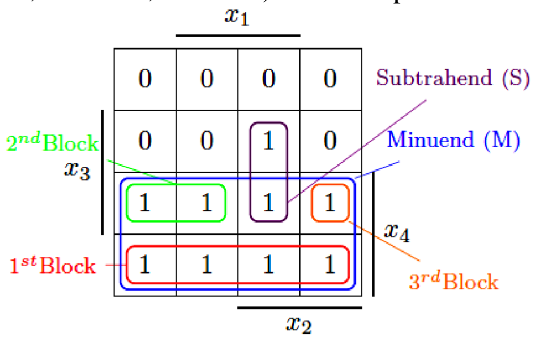


Figure 2: Example of ⊖ in a K-map

$$\underbrace{x_4}_{\text{Minuend}} \ominus \underbrace{x_3x_2x_1}_{\text{Subtrahend}} = \underbrace{\underbrace{x_4\overline{x_3}}_{1^{\text{st}}\text{Block}} \vee \underbrace{x_4x_3\overline{x_2}}_{2^{\text{nd}}\text{Block}} \vee \underbrace{x_4x_3x_2\overline{x_1}}_{3^{\text{rd}}\text{Block}}}_{\text{Difference}}$$

- The first literal of the subtrahend, here x_3, is taken complement and AND-ing to the minterm of the minuend, here x_4. Consequently, the first block of the difference is $x_4\overline{x_3}$.
- Then the second literal, herex_2, is taken complement and AND-ing to the minterm of the minuend and to the first literal x_3 of the subtrahend. Therefore, the second block is $x_4x_3\overline{x_2}$.
- Following the next literal, hier x_1, is taken complement and AND-ing to the minterm of the minuend and to the first literal x_3 and second literal x_2 of the subtrahend. Thus, the third block of the difference is $x_4x_3x_2\overline{x_1}$.
- This process is continued until all literals of the subtrahend are singly complemented and linked by AND-ing to the minuend in a separate minterm.

The Equation (9) is applied to calculate the orthogonalizing difference-building of two minterms. In this case, the formula does not show the value of the individual literals, but it shows whether the associated literal exists complement or not. The indices declare only the order of literals which have to be calculated. In this case, the formula ($\bigvee_{i=1}^{n_j} \overline{x_i} = \overline{x_1} \vee x_1\overline{x_2} \vee .. \vee x_1x_2 \cdot .. \cdot \overline{x_n}$) formed from [18] is applied to describe the orthogonalizing difference-building in a mathematically easier way:

$$m_M \ominus m_S = \left(\bigwedge_{m=0}^{n-1} x_{n-m}\right) \ominus \left(\bigwedge_{s=0}^{n-1} x_{n-s}\right) \coloneqq \left(\bigwedge_{m=0}^{n-1} x_{n-m}\right) \wedge \left(\bigvee_{s=0}^{n_j-1} \bar{x}_{n-s}\right) =$$
$$= (x_n \cdot x_{n-1} \cdot\cdot\cdot x_1)_M \ \wedge (\bar{x}_n \ \vee x_n\bar{x}_{n-1} \vee..\vee x_n \cdot x_{n-1} \cdot\cdot\cdot x_1)_S \tag{9}$$

The result may differ depending on the starting literal. There are many equivalent options. They only differ in the form of coverage. The minterms of the difference are pairwise disjoint to each other, so that the result is already availabe in an orthogonal form. The the result corresponds to the minuend if the subtrahend is already orthogonal to the minuend ($m_M \perp m_S$):

$$\textbf{if}\colon m_S \not\subseteq m_M \textbf{then}\colon m_M \ominus m_S = m_M \tag{10}$$

The number of the minterms (blocks) in the result called n corresponds to the number of the variables presented in the subtrahend and are not presented in the minuend at the same time. The number of the possible results can be defined by $n!$ for $n > 0$. The orthogonalizing difference-building of two TVs (TV_M, TV_S), is defined by a corresponding Equation (11):

$$TV_M \ominus \ TV_S = [t_n, t_{n-1}, .., t_1]_M \ominus \ [t_n, t_{n-1}, .., t_1]_S \coloneqq$$
$$= \ [t_n, t_{n-1}, .., t_1]_M \wedge \begin{bmatrix} \bar{t}_n & - & \cdots & - \\ t_n & \overline{t_{n-1}} & \cdots & - \\ \vdots & \vdots & \vdots & \vdots \\ t_n & t_n & \cdots & \bar{t}_1 \end{bmatrix}_S \tag{11}$$

By the use of this new method, two calculation procedures - building the difference and the subsequent orthogonalization - can be performed in one step. That means that the orthogonalizing difference-building $\ominus$represents the composition of the difference-building \and the subsequent orthogonalization *Orth* [6].

3.2. Analysis

3.2.1. Mathematical

The general validity is proved by mathematical induction:

Basis: n = 1

$$\left(\bigwedge_{m=0}^{1-1} x_{1-m}\right) \wedge \left(\bigvee_{s=0}^{1-1} \bar{x}_{1-s}\right) = (x_1)_M \wedge (\bar{x}_1)_S$$
$$(x_1)_M \wedge (\bar{x}_1)_S = (x_1)_M \wedge (\bar{x}_1)_S$$

Statement is true:

Inductive step: n = n + 1

$$\left(\bigwedge_{m=0}^{n} x_{(n+1)-m}\right) \wedge \left(\bigvee_{s=0}^{n_j} \bar{x}_{(n+1)-s}\right) = \left(x_{(n+1)} x_n x_{n-1} .. x_1\right)_M \wedge \left(\bar{x}_{(n+1)} \vee x_{(n+1)} (\bar{x}_n \ \vee x_n \bar{x}_{n-1} \vee..\vee x_n \cdot\cdot\cdot x_1)\right)_S$$
$$\left(x_{(n+1)} \bigwedge_{m=0}^{n-1} x_{n-m}\right) \wedge \left(\bar{x}_{(n+1)} \vee x_{(n+1)} \bigvee_{s=0}^{n_j-1} \bar{x}_{n-s}\right) = \left(x_{(n+1)} \bigwedge_{m=0}^{n-1} x_{n-m}\right) \wedge \left(\bar{x}_{(n+1)} \vee x_{(n+1)} \bigvee_{s=0}^{n_j-1} \bar{x}_{n-s}\right)$$

The new Equation (3) is equated with the usual Equation (9) to show the equivalence. Since the term of the minuend ($\bigwedge_{m=0}^{n-1} x_{n-m}$) is equal on both sides, it can be neglected:

$$\left(\bigvee_{s=0}^{n-1} \bar{x}_{n-s}\right) = \left(\bigvee_{s=0}^{n_j-1} \bar{x}_{n-s}\right)$$

$$\bar{x}_n \vee \bar{x}_{n-1} \vee .. \vee \bar{x}_1 = \bar{x}_n \vee x_n\bar{x}_{n-1} \vee .. \vee x_n x_{n-1} \cdot .. \cdot x_2\bar{x}_1$$

Due to the axiom of absorption with $\bar{x_\iota} \vee x_i\bar{x_J} = \bar{x_\iota} \vee \bar{x_J}$, the equivalence between two methods is confirmed. They only differ in their form of coverage. The right side is the orthogonal form of the left side. This proves that the orthogonalizing difference-building supplies equivalent results as the difference-building derived out of the set theory.

3.2.2. Computing Time

A study between usual and new method in computing time depending on the length (dimension$dim[i]$) of the minuend and subtrahend indicates almost identical computing times. The new method (*orth_Diff*) has a slightly longer computing time with increasing dimension than the method of difference-building (*Diff*). The comparison of the average values of both methods shows a maximum difference of just 1.2μs, which can be explained by the consideration complexity. The supplementation instruction per loop iteration results in a multiplicative constant which explains the minimal difference. Against it, the results are orthogonal. The comparison in Figure 3 illustrates that the method *orth_Diff* has faster computing time with increasing dimension $dim[i]$ as the composition of method *Diff* and the subsequent method of orthogonalizing *Orth* [19].

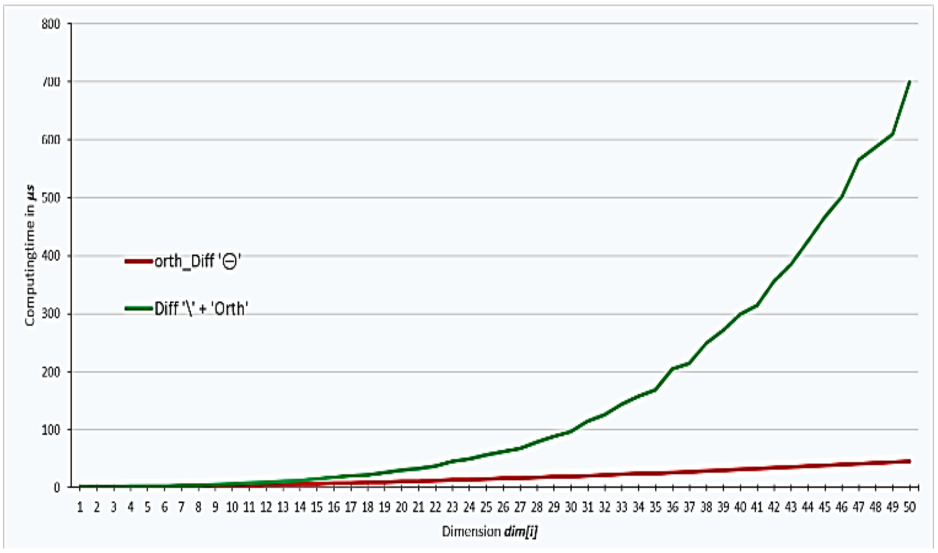


Figure 3: Comparison of computing time of *orth_Diff*⊖ and composition of *Diff* and *Orth*

In Diagram (Fig. 4) the explanation for the higher computing time of the composition is provided. With increasing dimension the computing time of the method *Orth* increases, whereas the method *Diff* decreases with respect to the composition. In dimension $dim[14]$ both functions have the same percentage value of the whole computing time. Thereafter, the percentage value of *Diff*runs low and at the same time the percentage value of *Orth*increases.

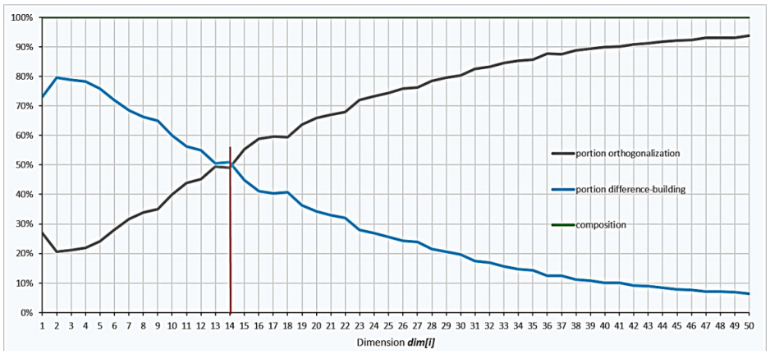


Figure 4: Percentage value of whole computing of *Diff* and *Orth*

At the dimension$dim[50]$the share of computing time of $Orth$ is already 93.7 $\%$ and the percentage value of *Diff*is 6.7 $\%$. Finally, the measurement of computing time demonstrates that the new method *orth_Diff* brings a significant advantage due to the orthogonal results [19].

3.2.3. Complexity

The complexity of both methods is analyzed by the evaluation of their implemented functions. The dimension of the inserted minterms respectively ternary-vectors is determined as the input size j. Figure 5 shows the pseudocode of *orth_Diff*. For the function *orth_Diff* is obtained according to the rules of the O-calculus many complexities of $O(1)$ for instructions and accordingly comparisons and two interleaved nested complexities of$O(1)$. As the result, the total complexity is $O(j^2)$ [19]. The complexity of the function *Diff* is $O(j)$ because the inner loops are not required and may be replaced by three linear operations with the complexity of $O(1)$ (Fig. 6).

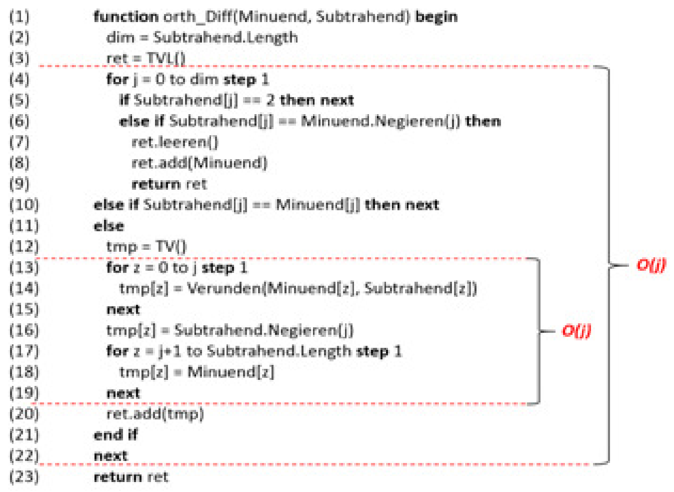
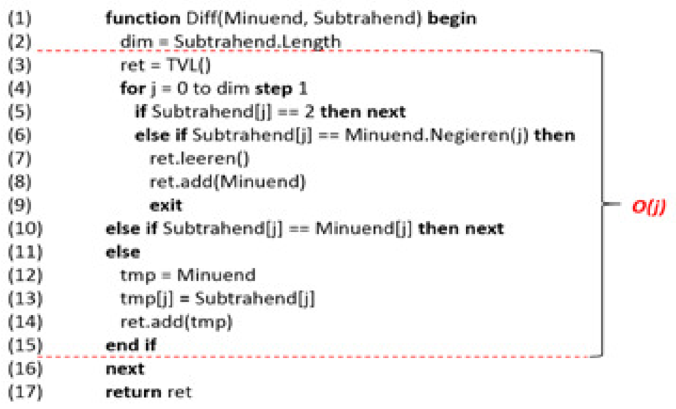

```
function orth_Diff(Minuend, Subtrahend) begin
  dim = Subtrahend.Length
  ret = TVL()
  for j = 0 to dim step 1
    if Subtrahend[j] == 2 then next
    else if Subtrahend[j] == Minuend.Negieren(j) then
      ret.leeren()
      ret.add(Minuend)
      return ret
else if Subtrahend[j] == Minuend[j] then next
else
  tmp = TV()
  for z = 0 to j step 1
    tmp[z] = Verunden(Minuend[z], Subtrahend[z])
  next
  tmp[z] = Subtrahend.Negieren(j)
  for z = j+1 to Subtrahend.Length step 1
    tmp[z] = Minuend[z]
  next
  ret.add(tmp)
end if
next
return ret
```

Figure 5: Pseudocode of *orth_Diff* with complexity

However, the additive and multiplicative constants have no influence on the complexity and are omitted for this reason. Although the total complexity of the usual method is minor but it increases in total by an additional code for orthogonalization.



```
function Diff(Minuend, Subtrahend) begin
  dim = Subtrahend.Length
  ret = TVL()
  for j = 0 to dim step 1
    if Subtrahend[j] == 2 then next
    else if Subtrahend[j] == Minuend.Negieren(j) then
      ret.leeren()
      ret.add(Minuend)
      exit
else if Subtrahend[j] == Minuend[j] then next
else
  tmp = Minuend
  tmp[j] = Subtrahend[j]
  ret.add(tmp)
end if
next
return ret
```

Figure 6: Pseudocode of *Diff* with complexity

3.2.4. Data Memory Request

First, the memory request of the data structures TV and TVL is determined for this comparison [19]. A TV consists of an array of $m \cdot$ byte elements. Each byte represents an element of the ternary variables. This means that the memory request of the data structures depends on the dimension of the ternary variables. In *C#* the data type *Byte* consists of 8Bits. For a 64 bit system a minimum memory request $Sb_{TV}(m)$for the data structure of TV with an addressing pointer is calculated by:

$$Sb_{TV}(m) = \underbrace{m \cdot 8\,Bits}_{m \cdot Byte-Elements} + \underbrace{32\,Bits}_{Pointer\ at\ TV} \tag{12}$$

Since a TVL consists of a list of TVs. The memory request of a TVL depends on the number of ternary variables and also the number of the consisting TVs called k. The easiest way to implement a list is the use of a linked list. Each entry in the list has a pointer at its follower. Thus, a minimal memory request $Sb_{TVL}(k,m)$is calculated by:

$$Sb_{TVL}(k,m) = k \cdot Sb_{TV}(m) + \underbrace{32\,Bits}_{Pointer\ at\ TVL} \tag{13}$$

For this reason, the minimal total memory requestSb_{tot} for the operation of two TVs, the difference-building and orthogonalizing difference-building in this case is:

$$Sb_{tot} = \underbrace{2 \cdot Sb_{TV}(m)}_{TV_M \& TV_S} + \underbrace{Sb_{TVL}(k,m)}_{Result-TVL} \tag{14}$$

The theoretical memory request has to be calculated equally for both function. Because of that, they have the same minimal memory request. Addional memory request for a function for orthogonalization is not needed, because an orthogonal TVL of the orthogonalizing difference is already provided. That shows that the new method, which is the composition of two functions has the advantage of reducing the memory request in addition to faster computing time. The minimal theoretical memory request depending on the dimension $dim[i]$and the number of n which primarily affects the memory usage is illustrated in Figure 7. Thereby, $dim[i]$ is varied at a constant n and afterwards in the reverse case and analyzed on memory request. For constant$dim[i]$and changing $n$the memory usage is higher than in the reverse case. It applies$\Delta Sb_{tot}(n) = \Delta Sb_{tot}(dim[i])$.

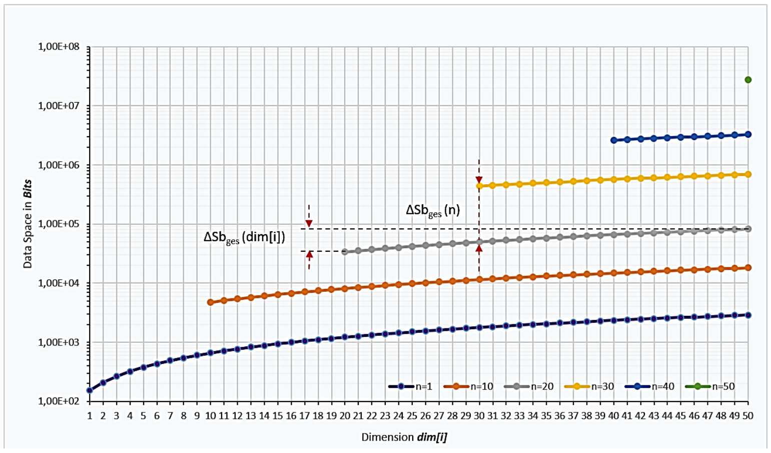


Figure 7: Minimal theoretical memory request

3.3. Applications of Orthogonalizing Difference-Building

3.3.1. Orthogonalizing Difference-Building of two Functions

By Equation (14) (presented in TVL-arithmetic) based on the orthogonalizing difference-building it is possible to calculate an orthogonal difference of two functions respectively two TVLs. One of them is the minuend function f_M^{orth} which has to be orthogonal and the other one is the subtrahend function f_S. The minuend is subducted the common set of subtrahend and minuend; the result is represented in an orthogonal form. The associated minterms (TVs) have the corresponding index-notation, which means M stands for the minuend and S stands for the subtrahend:

$$D\left(f_M^{orth}\right) \ominus D(f_S) = \begin{bmatrix} TV_{1M} \\ TV_{2M} \\ \vdots \\ TV_{mM} \end{bmatrix} \ominus \begin{bmatrix} TV_{1S} \\ TV_{2S} \\ \vdots \\ TV_{mS} \end{bmatrix} :=$$

$$= \begin{bmatrix} (TV_{1M} \ominus TV_{1S}) \wedge (TV_{1M} \ominus TV_{2S}) \wedge .. \wedge (TV_{1M} \ominus TV_{mS}) \\ (TV_{2M} \ominus TV_{1S}) \wedge (TV_{2M} \ominus TV_{2S}) \wedge .. \wedge (TV_{2M} \ominus TV_{mS}) \\ \vdots \\ (TV_{mM} \ominus TV_{1S}) \wedge (TV_{mM} \ominus TV_{2S}) \wedge .. \wedge (TV_{mM} \ominus TV_{mS}) \end{bmatrix} \quad (14)$$

In this case it is important to stress that any outcome of each individual $\ominus$-linkings has to be considered. That means, if the combination of $(TV_{1M} \ominus TV_{1S}) = 0$ arises for example, this will complete the appropriate row to 0 because of: $x_i \wedge 0 = 0$. Since the orthogonalizing difference-building has already been proved in general validity, there is no need proof for generel validity in this case, because all the single links are generally valid. Therefore, out of logical conclusion, Equation (14) is generally valid. If two functions are disjoint to each other $(f_M \perp f_S)$ then a difference cannot be formed and it follows:

$$\textbf{if}: f_S \not\subseteq f_M \textbf{then}: f_M \ominus f_S = f_M \quad (15)$$

3.3.2. Orthogonal Complement of a Function

A further application is the building of an orthogonal complement of a function of the disjunctive normal form. In the set theory the set difference of universal set $G$and a set Acorresponds to set $\bar{A}$which is the complement of the given set A:

$$G \setminus A = \bar{A} \quad \rightarrow \quad G \cap \bar{A} = \bar{A} \qquad (16)$$

Transferred to the switching (Boolean) algebra the complement of a function $\bar{f}(\underline{x})$ can be determined from the difference of a unit function $f(\underline{x}) = 1$ and the function $f(\underline{x})$:

$$f(1) \setminus f(\underline{x}) = \bar{f}(\underline{x}) \quad \rightarrow \quad f(1) \wedge \bar{f}(\underline{x}) = \bar{f}(\underline{x}) \qquad (17)$$

As both methods are equivalent, Equation (17) can be formulated with $\ominus$ to find out an orthogonal complement:

$$f(1) \ominus f(\underline{x}) = \bar{f}(\underline{x})^{orth} \text{or} \quad D(1) \ominus D(f) = D(\bar{f}^{orth}) \qquad (18)$$

3.3.3. Orthogonal EXOR of two Functions

The EXOR-operation of two minterms $m_{i,j}$and also of two functions $f_{i,j}$can be calculated as:

$$\begin{aligned} f_i \oplus f_j &= f_i\bar{f}_j \vee \bar{f}_i f_j = \\ &= (f_i \vee f_j) \wedge (\bar{f}_i \vee \bar{f}_j) = \\ &= (f_i \vee f_j) \wedge \overline{(f_i \wedge f_j)} = \\ &= (f_i \vee f_j) \setminus (f_i \wedge f_j) = \end{aligned} \qquad (19)$$

It may be formulated with $\ominus$ due to the equivalence to get orthogonal result:

$$f_i \oplus f_j = (f_i \vee f_j) \ominus (f_i \wedge f_j) \qquad (20)$$

By AND-ing and OR-ing of the functions $f_{i,j}$an inherent relation between these two functions is constructed.Thus, the minuend function does not need to be orthogonal. The advantage is the orthogonal result again. By using $\ominus$ the difficulties which arise with the building of the complement of a function are circumvented. The form of the function changes by building the complement and the transformation back to its original form requires a more sophisticated calculation.

3.3.4. Orthogonal EXNOR of two Functions

The EXNOR-operation of two minterms $m_{i,j}$or two function$f_{i,j}$is basically the complement of the EXOR-operation of the same minterms or functions. The EXNOR can be expressed by the complement of EXOR, which also can be formulated by using $\ominus$. Accordingly, the result is orthogonal:

$$f_i \odot f_j = \overline{f_i \oplus f_j} = f(1) \ominus (f_i \oplus f_j) = f(1) \ominus [(f_i \vee f_j) \ominus (f_i \wedge f_j)] \qquad (21)$$

3.3.5. Orthogonalizing OR-ing

Another method, the orthogonalizing OR-ing, which is based on the orthogonalizing difference-building, is formed in the following. The orthogonalizing OR-ing $\ovee$ is a variant of building the disjunction of two summand-minterms (S_1, S_2) whereby the result is orthogonal. Orthogonalizing OR-ing is going to be explained by an example in a K-map with 4 variables (Fig. 8). Two summand-minterms (a group of 8 and a group of 4) are orthogonalizing OR-ed and a result consisting of several blocks appears; the several blocks are pairwise orthogonal to each other.

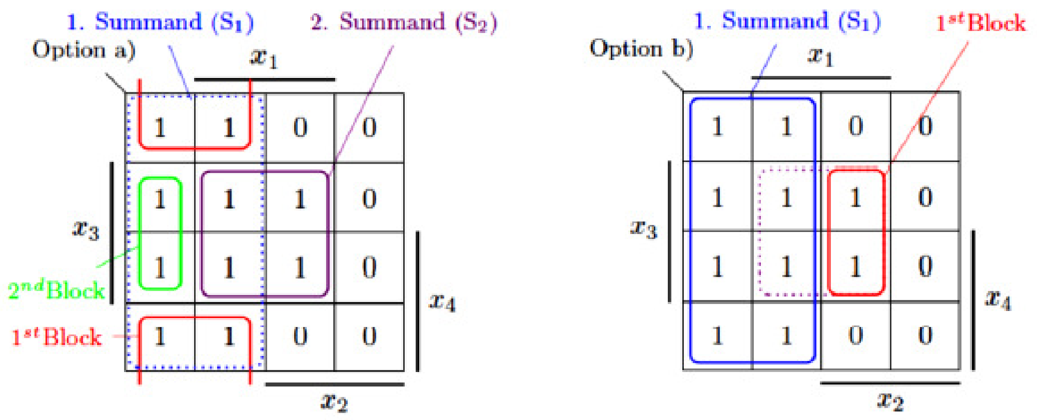


Figure 8: Orthogonalizing OR-ing in K-map

Option a):

$$\bar{x}_2 \ovee x_3x_1 = \underbrace{\bar{x}_3\bar{x}_2 \vee x_3\bar{x}_2\bar{x}_1 \vee x_3x_1}_{\text{Sum of Blocks}}$$

Option b):

$$x_3x_1 \ovee \bar{x}_2 = \underbrace{x_3x_2x_1 \vee \bar{x}_2}_{\text{Sum of Blocks}}$$

The idea out of the K-map is noted as propositional logic which a Boolean form is going to be derived. By the use of orthogonalizing OR-ing, the intersection set of first and second summand-minterm (S_1, S_2) is removed from the first summand-minterm $S_1$and the second summand-minterm S_2 is linked by a disjunction to that subtraction:

$$S_1 \ovee S_2 = [S_1 \setminus (S_1 \wedge S_2)] \vee S_2 = [S_1 \wedge \overline{(S_1 \wedge S_2)}] \vee S_2 =$$
$$= [S_1 \wedge (\overline{S_1} \vee \overline{S_2})] \vee S_2 = S_1\overline{S_2} \vee S_2 \qquad (22)$$

As $S_1 \setminus (S_1 \wedge S_2) = S_1 \setminus S_2$ applies, $\ominus$ is substituted. Consequently, the orthogonalizing OR-ing $\ovee$ can also be expressed by the orthogonalizing difference-building $\ominus$:

$$S_1 \ovee S_2 = [S_1 \ominus S_2] \vee S_2 \qquad (23)$$

The Boolean form of orthogonalizing OR-ing of two minterms $m_{S_1} = \bigwedge_{S_1=0}^{n-1} x_{n-S_1}$and $m_{S_1} = \bigwedge_{S_2=0}^{n-1} x_{n-S_2}$is defined by Equation (24), which does not give the value of separate literals, but it shows whether the value of the variable is complement or not. The indices only indicate the order of the variables to be calculated:

$$m_{S_1} \circledvee m_{S_2} = \left(\bigwedge_{S_1=0}^{n-1} x_{n-S_1} \right) \circledvee \left(\bigwedge_{S_2=0}^{n-1} x_{n-S_2} \right) :=$$

$$= \left[\left(\bigwedge_{S_1=0}^{n-1} x_{n-S_1} \right) \wedge \left(\bigvee_{S_2=0}^{n_j-1} \bar{x}_{n-S_2} \right) \right] \vee \left(\bigwedge_{S_2=0}^{n-1} x_{n-S_2} \right) = \qquad (24)$$

By swapping the two summands in their position, the result changes. Both solutions, however, are equivalent because the same set is covered. They only differ in the form of coverage which also be seen in Figure 8, in which both possible solutions are presented. But in order to represent the orthogonal result with a minimum of minterms to work in the TVL-representation with low memory request, the summand-minterm with more literals has to be accepted as the first summand-minterm,because the commutativity applies for $\circledvee$ and also for $\ominus$. The following Equation (25) is used for the orthogonalizing OR-ing of two ternary vectors
$TV_{S_1} = [t_n, t_{n-1}, .., t_1]_{S_1}$and $TV_{S_2} = [t_n, t_{n-1}, .., t_1]_{S_2}$:

$$TV_{S_1} \circledvee TV_{S_2} = [t_n, t_{n-1}, .., t_1]_{S_1} \circledvee [t_n, t_{n-1}, .., t_1]_{S_2} =$$

$$= \left[[t_n, t_{n-1}, .., t_1]_{S_1} \wedge \begin{bmatrix} \bar{t}_n & - & \cdots & - \\ t_n & \overline{t_{n-1}} & \cdots & - \\ \vdots & \vdots & \vdots & \vdots \\ t_n & t_n & \cdots & \bar{t}_1 \end{bmatrix}_{S_2} \right] \vee [t_n, t_{n-1}, .., t_1]_{S_2} \qquad (25)$$

The number of the minterms in the result called n corresponds to the number of the variables presented in the second summand m_{S_2} and are not presented in the first summand m_{S_1}at the same time; plus*1* for the second summand m_{S_2} as the last linked minterm. The number of the possible results can be charged by $n!$for $n > 0$. Depending on the starting literal the result may differ. There are many equivalent options which only differ only in the form of coverage. If both minterms are disjoint (orthogonal) to each other, the result corresponds to the disjunction of both minterms:

$$\textbf{if: } m_{S_1} \not\subset m_{S_2} \qquad \textbf{then: } m_{S_1} \circledvee m_{S_2} = m_{S_1} \vee m_{S_2} \qquad (26)$$

By the use of orthogonalizing OR-ing $\circledvee$, two calculation procedures - OR-ing and subsequent orthogonalizing - can be performed in one step. That means that the orthogonalizing OR-ing $\circledvee$ is the composition of OR-ing ($\vee$) and the subsequent orthogonalization Orth.

It is not necessary to prove this method for general validity because it includes the already general method of orthogonalization difference-building. The Equations (23) and (2) are equalized to indicate the equivalence of orthogonalization OR-ing and usual OR-ing:

$$S_1 \overline{S_2} \vee S_2 = S_1 \vee S_2 \qquad (27)$$

Due to the axiom of the absorption the equivalence is verified. The right side is the orthogonal form of the left side which means they only differ in the form of coverage. So, the results of both sides are equal.

4. Conclusion

This work shows that the method of the orthogonalizing difference-building is generally valid and is also equivalent to the usual method of difference-building. In contrast to the composition, orthogonalizing difference-building has faster computing time with increasing dimension. In addition, the method does not require additional memory request for an additional function for orthogonalization because this method already provides orthogonal results. The orthogonalizing difference-building is used to calculate the orthogonal difference of two minterms respectively two TVs, two functions or two TVLs. It is also employed to determine the complement of a function as well as the EXOR and EXNOR of two functions to achieve an orthogonal result. Another method, the orthogonalizing OR-ing of two minterms or TVs, is developed out of the orthogonalizing difference-building. The application of ternary-vector-list is amplified by these new methods to implement simple and quick elementary functions. Due to the inner orthogonalization further processing steps in the TVL arithmetic are considerably simplified because the orthogonal form of disjunctive normal form has the advantage to consider it as antivalence normal form. Thus, additional calculation such as the differential calculus are remarkably facilitated.

References

[1] Zander, H. J.: Logischer Entwurf binärer Systeme. 3. bearb. Auflage. Berlin, Germany: Verl.Technik, 1989, ISBN 3-341-00526-9.

[2] Bronstein, I.N.; Semendjajew, K.A.; Musiol, G.; Mühlig, H.:Taschenbuch der Mathematik. 7. vollständig überarbeitete und ergänzte Auflage. Frankfurt am Main, Germany: wissenschaftlicher Verl. Harri Deutsch GmbH, 2008, ISBN 978-3-8171-2007-9.

[3] Popula, L.: Mathematik für Ingenieure und Naturwissenschaften, Band 1. 13. durchgelesene Auflage. Wiesbaden, Germany: Viewer + Teubner Verlag | Springer Fachmedien. Wiesbaden GmbH, 2011, ISBN 978-3-8348-1749-5.

[4] Matthes, W.: Datenzugriffsprinzipien in objektorientierten Rechnerarchitekturen. Preprint. Technische Universität Karl-Marx-Stadt (Chemnitz), 1989.

[5] Matthes, W.: Spezielle Hardware zur Verarbeitung von Ternärvektorlisten. Dissertation. Technische Universität Karl-Marx-Stadt (Chemnitz), 1987.

[6] Posthoff, Ch.; Steinbach, B.: Binäre Gleichungen - Algorithmen und Programme.wissenschaftliche Schriftreiche. Technische Universität Karl-Marx-Stadt (Chemnitz), 1979.

[7] Posthoff, Ch.; Steinbach, B.: Binäre dynamische Systeme - Algorithmen und Programme. wissenschaftliche Schriftenreihe. Technischen Hochschule Karl-Marx-Stadt (Chemnitz), 1979.

[8] Posthoff, Ch.; Steinbach, B.: Binäre dynamische Systeme. Berlin: Oldenbourg R. Verlag GmbH, 1981, ISBN 348625071X.

[9] Bochmann, D.; Zakrevskij, A.D.; Posthoff, Ch.: Boolesche Gleichungen. Theorie - Anwendungen - Algorithmen.Berlin: VEB Verlag Technik, 1984, ISBN 3211958150.

[10] Kühnrich, M.: Ternärvektorlisten und deren Anwendung auf binäre Schaltnetzwerke. Dissertation. Technische Hochschule Karl-Marx-Stadt (Chemnitz), 1979.

[11] Posthoff, C.; Bochmann, D.; Haubold, K.: Diskrete Mathematik. 1. Auflage. Leipzig, DDR: BSB Teubner, 1986, ISSN 0465-3769.

[12] Kempe, G.: Tupel von TVL als Datenstruktur für Boolesche Funktionen. Dissertation (A). Technische Universität Bergakademie Freiberg, 2003.

[13] Whitesitt, J.E.: Boolesche Algebra und Ihre Anwendungen. Band 3. Braunschweig, Germany: Friedr. Vieweg + Sohn GmbH, 1969.

[14] Pomberger, G.; Dobler, H.: Algorithmen und Datenstrukturen - eine systematische Einführung in die Programmierung. 1. Auflage. München, Germany: Addison-Wesley Verlag, 2008, ISBN-10: 3827372682.

[15] Saake, G.; Sattler, K.-U.: Algorithmen und Datenstrukturen - Eine Einführung mit Java. 3. Auflage. Heidelberg, Germany: dpunkt.verlag GmbH, 2006, ISBN-10: 3898643859.

[16] Wagenknecht, Ch.: Algorithmen und Komplexität. 1. Auflage. Leipzig, Germany: Carl Hanser Verlag GmbH \& Co. KG, 2003, ISBN-10: 3446223142.

[17] Bochmann, D.: Binäre Systeme. Ein BOOLEAN Buch. Hagen, Germany: LiLoLe-Verlag GmbH, 2006, ISBN 3-934447-10-4.

[18] Crama, Y.; Hammer, P.L.: Boolean Functions. Theory, Algorithms, and Applications.New York, USA: Cambridge University Press, 2011, ISBN 978-0-521-84751-3.

[19] Kassim,H.; Can, Y.; Sattler, M.S.:Untersuchung eines neuen Algorithmus zur Berechnung orthogonalisierter Differenz. Bachelor-Thesis, Lehrstuhl für Zuverlässige Schaltungen und Systeme, Friedrich-Alexander-Universität Erlangen Nürnberg, Germany, 2014.

[20] Can, Y.; Fischer, G.: Orthogonalizing Boolean Subtraction of Minterms or Ternary Vektors. International Conference on Computational and Experimental Science and Engineering (ICCESEN2014), 24-29 October, 2014, Antalya-Turkey.

AUTHORS

Yavuz Can was born in Erlangen, Germany, in 1979. He received his Diploma (Dipl.-Ing.) degree in me-chatronics from Friedrich-Alexander-University Erlan-gen-Nürnberg, Germany, in 2010. He is currently a Research Assistant of Prof. Georg Fischer working toward his Ph.D. degree in the Institute for Electronics Engineering at Friedrich-Alexander University in Erlangen. His research interest include orthogonality of Boolean functions and Ternary-Vector-List.

Georg Fischer was born in Lower Rhine region, Germany, in 1965. He received the Diploma degree in electrical engineering with focus on communications, micro-wave and electro-dynamics from RWTH Aachen University, Aachen, Germany, in 1992, and the Dr.-Ing. degree in electrical engineering from the University of Paderborn, Paderborn, Germany, in 1997. From 1993 to 1996, he was a Research Assistant with the University of Paderborn, where he was involved with adaptive antenna array systems for mobile satellite communications. From 1996 to 2008, he performed research with Bell Laboratories, Lucent (later Alcatel-Lucent), where he focused on the RF and digital architecture of mobile communication base stations for global system for mobile communications (GSM), Universal Mobile Telecommunications System (UMTS), and features for network coverage and capacity enhancements. In 2000, he became a Bell Labs Distinguished Member of Technical Staff (DMTS), and in 2001, a Bell Labs Consulting Member of Technical Staff (CMTS). He was also a Chairman with the European Telecommunications Standards Institute (ETSI) during the physical layer standardization of the GSM-EDGE system. From 2001 to 2007, he was a Part-Time Lecturer with the University of Erlangen–Nuremberg, Erlangen, Germany, during which time he lectured on base station RF and digital technology. Since April 2008, he has been a Professor of electronics engineering with the University of Erlangen–Nuremberg. He holds over 50 patents concerning microwave and communications technology. His research interests are in transceiver design, analog/digital partitioning, digital signal processing, converters, enhanced amplifier architectures, duplex filters, metamaterial structures, GaN transistor technology and circuit design, and RF microelectromechanical systems (MEMS) with specific emphasis on frequency agile, tunable, and reconfigurable RF systems for software-define radio (SDR) and cognitive radio (CR) applications. His new research interests concentrate on medical electronics like using microwaves for detection of vital parameters. Georg Fischer is a Senior Member of the IEEE Microwave Theory and Techniques Society (MTT-S)/Antennas and Propagation Society (AP-S)/Communications Society (COMSOC)/ Vehicular Technology Society (VTC) and Engineering in Medicine and Biology Society (EMBS). He is a member of VDE-ITG and the European Microwave Association (EUMA). He was the co-chair of the European Conference on Wireless Technology (ECWT) of European Microwave Week Conference (EUMW 2007). For EUMW 2013, Nuremberg, Germany and GeMiC 2015, Nuremberg, Germany he has served as the General Technical Program Committee (TPC) chairman.