FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 20, no. 3, December 2007, 259-279

Most Complex Boolean Functions Detected by the Specialized Normal Form

Bernd Steinbach

Abstract: It is well known that Exclusive Sum-Of-Products (ESOP) expressions for Boolean functions require on average the smallest number of cubes. Thus, a simple complexity measure for a Boolean function is the number of cubes in its simplest ESOP. It will be shown that this structure-oriented measure of the complexity can be improved by a unique complexity measure which is based on the function. Thus, it is suggested to detect all most complex Boolean functions more precisely by means of the Specialized Normal Form (SNF). The SNF is a unique (canonical) ESOP representation of a Boolean function. In this paper properties of the most complex Boolean functions are studied. Adjacency graphs of the SNF will be used to calculate minimal ESOPs as well as to detect special properties of most complex Boolean functions. These properties affect the procedure of finding exact minimal ESOPs. A solution is given that overcomes these observed problems. Using the SNF, the number of most complex Boolean functions was found. A recursive algorithm will be given that calculates each most complex Boolean function for a given number of variables.

Keywords: Boolean function, complexity, specialized normal form, unique ESOP, exact minimal ESOP, adjacency graph, hypercube corner compaction (HCCC).

1 Introduction

Knowledge about most complex Boolean functions can be helpful as upper bound in minimization algorithms and may even control such algorithms. A very simple measure of the complexity of Boolean functions is the number of gates needed for their realization in a circuit. Unfortunately the selection of gates strongly affects this measure [1]. Alternatively the number of terms in a formula or the number of nodes in a graph representation can be taken as a measure of the complexity

Manuscript received August 18, 2007.

B. Steinbach is with Institute of Computer Science, Bernhard-von-Cotta-Str. 2, D-09596 Freiberg, Germany (e-mail: steinb@informatik.tu-freiberg.de).

of Boolean functions. Again, formulas or graphs of different size can express the same Boolean function and complicate the definition of their complexity.

Taking this observation into account, a unique representation of Boolean functions is an appropriate basis of a complexity measure of Boolean functions. Of cause, the complexity measure must be consistent with the size of circuit realizations of the evaluated Boolean function. That means, a higher complexity of a Boolean function leads basically to a larger size of the circuit.

A Binary Decision Diagram (BDD) [2], [3] is a unique representation of a Boolean function, if the order of variables is fixed. Generally, the number of nodes of a BDD can be a complexity measure of the represented Boolean function, but this number depends on the chosen order of variables in the BDD. Hence, BDDs are also not suitable as a basis for a complexity measure.

Further unique representations of Boolean functions are the classical normal forms [4]:

- disjunctive normal form,
- conjunctive normal form,
- antivalence normal form,
- equivalence normal form.

The number of terms in these forms is also a candidate for a complexity measure for the represented function. The largest number of disjunctions in a disjunctive normal form or antivalence normal form occurs for the simple function $f = 1$ which is quite a simple function. Similarly, the largest number of conjunctions in a conjunctive normal form or equivalence normal form occurs for the simple function $f = 0$, which is the simplest function at all. Hence, the classical normal forms are not suitable as a basis for a complexity measure too.

The theory of Boolean normal forms was significantly extended in [5]. Especially normal forms based on *Exclusive Sum-Of-Products* (ESOP) were studied. One class of normal forms are polynomials with a fixed polarity (negated or not negated) for each variable. Similar to BDDs, the number of disjunctions in a unique polynom depends on the chosen polarity vector. Hence, this class of normal forms is not suitable as the basis for a complexity measure too.

Ultimately, another ESOP-based unique normal form was included into the theory of Boolean normal forms in [5]. This *Specialized Normal Form* (SNF) was suggested in [6]. The SNF is a specifically selected ESOP. The research for exact minimal ESOPs leads us to a new understanding of the most complex Boolean functions. The history of research in exact ESOP minimization is much shorter than the history of research in exact Sum-Of-Products (SOP) minimization. It is

known from [7] that the ESOP representation of Boolean functions is typically more compact than the SOP representation.

In [8] the problem of exact ESOP minimization was reduced to the finding of a satisfying assignment of constraints. Unfortunately, this approach is strongly limited, because the number of variables in the used Helliwell function is equal to 3^k. This approach was generalized in [9] such that a function of n variables can be minimized if the exact minimum for all functions of $n-k$ $(k \geq 1)$ variables is known. Based on this idea in [10] a practical solution for all functions of 6 variables was presented. A slightly modified approach was suggested in [11]. The exact minimal ESOP should be found using the exact minimal ESOPs of three subfunctions.

A quite different approach was suggested in [6]. Using two very simple transformations, each ESOP of the Boolean function f can be expressed by a unique ESOP, called specialized normal form (SNF). Several helpful properties of the SNF have been proved. The basic idea for finding an exact minimal ESOP is to add the smallest number of cubes to the SNF and apply the expansion transformation in the reverse direction. The detection of the potential solution cubes was controlled by weights derived from the SNF. In [12] the method for selecting solution cubes was improved.

Using the SNF approach based on weights, the exact minimal ESOP was found for nearly all functions. In some cases the weight method does not indicate the right direction. In [13] this gap was closed using a HCCC function based on the adjacency graph of the SNF and an extended adjacency graph. HCCC is an abbreviation of *hypercube corner compaction*.

It is an interesting observation that by means of the general HCCC function the exact minimal ESOPs were found for almost all Boolean functions. The remaining very small set of Boolean functions turns out to be the set of the most complex functions. This offers a very simple complexity measure of a Boolean function based on its SFN that will be given in this paper.

Detailed studies reveal special properties of the most complex Boolean functions. These properties indicate that a slightly changed HCCC function is necessary in order to find the exact minimal ESOPs of the most complex function, too. Both the special properties of the most complex Boolean functions and their utilization in the extended HCCC function will be shown in this paper.

Using the SNF, the number of most complex Boolean functions could be determined. A recursive algorithm will be given that enumerates all most complex Boolean function for a given number of variables. This knowledge about most complex Boolean functions extends the theory of Boolean rings [4].

The rest of the paper is organized as follows. The SNF is introduced in Sec-

tion 2 in a compact manner as unique basic representation of Boolean functions for all further studies. Adjacency graphs in Section 3 reveal important properties of the detection of exact minimal ESOPs and the complexity of the represented Boolean function. Peculiarities of most complex Boolean functions are studied in detail in Section 4. The extended HCCC function will be introduced here. A recursive algorithm that allows to calculate any of the most complex Boolean functions is given in Section 5. Experimental results are given in Sections 4 and 5. Finally, in Section 6 this paper will be summarized.

2 Specialized Normal Form - SNF

An algebraic property of the exclusive-or operation and the Boolean variable x is visible in the following formulas:

$$x = \overline{x} \oplus 1 \quad (1)$$
$$\overline{x} = 1 \oplus x \quad (2)$$
$$1 = x \oplus \overline{x}. \quad (3)$$

These three formulas show that each element of the set $\{x, \overline{x}, 1\}$ has the same properties. For each variable in the support of the Boolean function f, exactly one left-hand side element of (1), (2) or (3) is included in each cube of an ESOP of function 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 **compression**. Using two laws of the Boolean ring: neutral element 0 (4), and idempotency (5), a next important property of the exclusive-or operation for a Boolean function f and a cube C (6) can be concluded.

$$f = f \oplus 0 \quad (4)$$
$$0 = C \oplus C \quad (5)$$
$$f = f \oplus C \oplus C \quad (6)$$

It follows from these formulas 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 expands the cube C_j with respect to the variable V_i into the cubes C_{n1} and C_{n2} based on the fitting formula (1), (2) or (3).

Algorithm 1 Calculate Exp(f)

Require: any ESOP of a Boolean function f
Ensure: complete expansion of the Boolean function f w.r.t. all variables of its support

```
1: for all variables V_i of the support of f do
2:    for all cubes C_j of f do
3:       ⟨C_n1, C_n2⟩ ← expand(C_j, V_i)
4:       replace C_j by ⟨C_n1, C_n2⟩
5:    end for
6: end for
```

Algorithm 2 Calculate R(f)

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

```
1: for i ← 0 to n − 2 do
2:    for j ← i + 1 to n − 1 do
3:       if C_i = C_j then
4:          C_i ← C_{n−1}
5:          C_j ← C_{n−2}
6:          n ← n − 2
7:          j ← i
8:       end if
9:    end for
10: end for
```

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

Definition 1 ***- SNF(f) -*** *Take any ESOP of a Boolean function f. The resulting ESOP of*

$$SNF(f) = R(Exp(f)) \tag{7}$$

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

3 Adjacency Graphs of a SNF

The adjacency graph of an SNF emphasizes some implicit knowledge about the SNF. Edges of the graph describe structural relationships between the cubes of the SNF which are used as labels of the vertices.

Definition 2 ***- Adjacency Graph $AG^{SNF(f)}$(V,E) of the SNF(f) -*** *The vertices V of the adjacency graph $AG^{SNF(f)}(V,E)$ correspond to the cubes of the SNF(f). Each vertex carries the ternary vector of the associated cube as label. Two vertices V of $AG^{SNF(f)}(V,E)$ are connected by an edge, if the associated labels have a distance equal to one, i.e. they differ exactly in one position of the ternary vectors.*

Figure 1 (a) shows the adjacency graph of a very simple function that can be expressed by a single cube. A more complex adjacency graph is shown in Figure 2.

It was proven in [6] that the adjacency graph of a Boolean function $f : B^k \rightarrow B$ is a *k-regular graph*. Each vertex in a *k*-regular graph has a degree of *k*, which means that each vertex is connected by edges with *k* other vertices.

The regularity of the adjacency graph offers a new basic approach for calculating an exact minimal ESOP. A cube of the minimal ESOP can be calculated based on one vertex of the adjacency graph of the SNF(f) and its *k* neighbors. The elements of each pair of neighboring vertices of the adjacency graph differ in exactly one variable of the two associated cubes. These values correspond to the right-hand side of the formulas (1), (2), or (3) such that the cube for the minimal ESOP can be created from the left-hand side values. The *k* adjacent edges of the basic vertex select all *k* variables of a cube. This property exists because of the complete expansion in Algorithm 1.

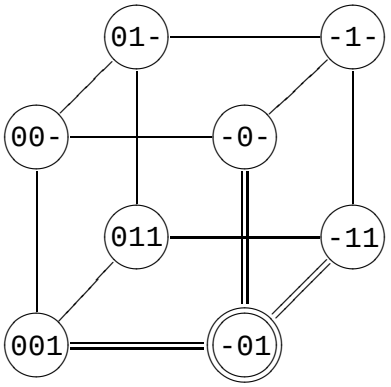
Figure 1 illustrates the procedure how a cube of the minimal ESOP can be reconstructed from a selected cube of the SNF and its neighboring vertices in the adjacency graph $AG^{SNF(f)}(V,E)$. The ternary notation uses the mapping $(x \rightarrow \texttt{1}, \overline{x} \rightarrow \texttt{0}, 1 \rightarrow \texttt{-})$. As an example the cube `(-01)` was selected in the SNF, indicated by double circles in the adjacency graph of Figure 1 (a). The three adjacent edges of the vertex `(-01)`, indicated by double lines in Figure 1 (a), end on such vertices having a different value in exactly one position. As can be seen in Figure 1, the different values in the three adjacent pairs of vertices occur once in each position. This property comes from the expansion algorithm Exp(f) and is used in the reverse direction for the reconstruction of a cube of the minimal ESOP. As shown in Figure 1 (b), each adjacent pair of vertices covers two values of the set $\{\texttt{0,1,-}\}$ in the position where they differ. The remaining third value of this set is taken in that position for the reconstructed cube. We call this procedure *hypercube corner compaction*, or shortly HCCC(AG, V_s).

As example for further studies we use the following Boolean function:

$$f(x_1, x_2, x_3) = x_1\overline{x}_3 \oplus \overline{x}_1\overline{x}_2x_3 \tag{8}$$

The reduction algorithm R(f) does not destroy the property that the adjacency graph is a *k*-regular graph. Figure 2 shows both the effect of removing a pair of

(a)



(b)

x_1	x_2	x_3	x_1	x_2	x_3		x_1	x_2	x_3
-	0	1	0	0	1	$\rightarrow$	1		
-	0	1	-	1	1	$\rightarrow$		-	
-	0	1	-	0	-	$\rightarrow$			0
							1	-	0

Fig. 1. Reconstruction of the cube $f(x_1,x_2,x_3) = x_1\overline{x}_3$ of the minimal ESOP from its SNF: (a) Adjacency graph $AG^{SNF(f)}(V,E)$ of the function $f(x_1,x_2,x_3) = x_1\overline{x}_3$ (vertices are labeled by ternary values of (x_1,x_2,x_3)) (b) Reconstruction of the cube `(1-0)` of the minimal ESOP using the selected cube `(-01)` of the SNF and its neighbors

cubes creating the SNF(f) and the application of the approach to find a minimal ESOP introduced above.

The SNF in Figure 2 is created for the Boolean function (8). The expanded hypercubes are visible in Figure 2, $x_1\overline{x}_3$ in the bottom left area and $\overline{x}_1\overline{x}_2x_3$ in the top right area, respectively. Note, these cubes have a distance of 3 so that $2^{(3-3)} = 1$ pair of common cubes exist after the expansion. The pair of common cubes is indicated by dotted circles in Figure 2.

Embedded in the procedure that creates the SNF, the algorithm R(f) removes this pair of cubes. The adjacent edges indicated by dotted lines are removed from the adjacency graph too. Three new edges connect both hypercubes so that the adjacency graph $AG^{SNF(f)}(V,E)$ remains k-regular. One of the new edges and two of the removed edges form a triangle where the adjacent vertices differ in exactly one position and all three values of the set {`0,1,-`} appear.

The minimal ESOP can be calculated as described above. In Figure 2 the vertices `(-01)` and `(1-0)`, indicated by double circles, were selected in the adjacency graph of the SNF(f). The vertex `(-01)` and its neighbors lead to the

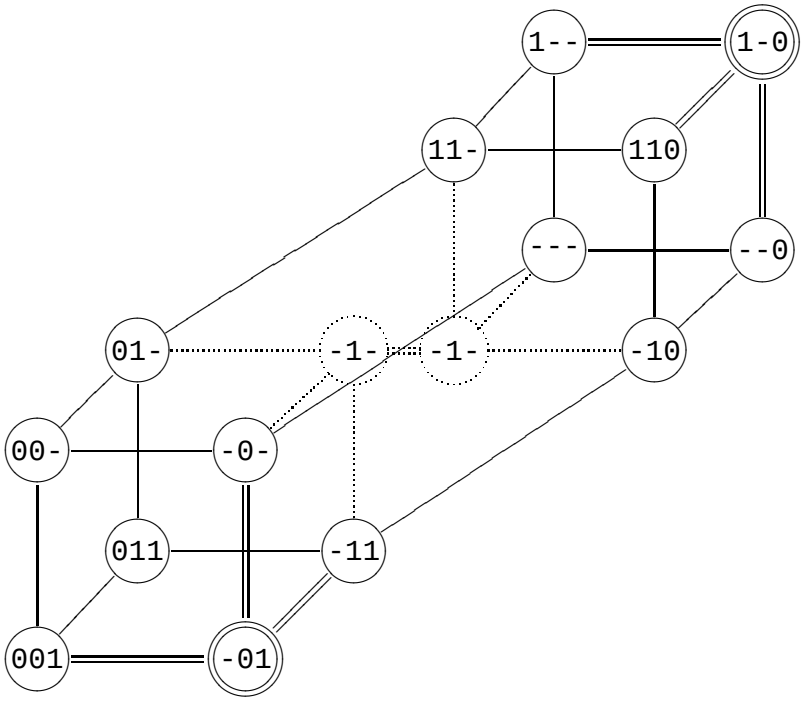


Fig. 2. Reconstruction of the minimal ESOP $f(x_1,x_2,x_3) = x_1\overline{x}_3 \oplus \overline{x}_1\overline{x}_2x_3$ from the adjacency graph $AG^{SNF(f)}(V,E)$

cube `(1-0)` that is $x_1\overline{x}_3$. The function HCCC(AG, `(1-0)`) creates the second solution cube `(001)` that is $\overline{x}_1\overline{x}_2x_3$. Based on this selection the minimal ESOP $f(x_1,x_2,x_3) = x_1\overline{x}_3 \oplus \overline{x}_1\overline{x}_2x_3$ has been found.

The adjacency graph of the SNF(f) in Figure 2 consists of 14 vertices. Each of these vertices is adjacent to three other vertices. From this point of view the vertices of the adjacency graph cannot be distinguished. The question arises, whether each vertex of the adjacency graph of the SNF(f) leads to a cube of the exact minimal ESOP of f. In order to answer this question, Table 1 enumerates all created cubes based on each selected cube of $AG^{SNF(f)}(V,E)$ of Figure 2.

The created cubes and the associated conjunctions in the first 4 lines and in the last 4 lines of Table 1 show the cubes of the minimal ESOP; several other cubes can be created as well. This example shows that in general not each vertex of the adjacency graph of the SNF(f) can be used to reconstruct the exact minimal ESOP of f. On the other hand 8 of 14 cubes lead by means of the suggested HCCC approach to the cubes of the exact minimal ESPO.

Table 1. All reconstructed cubes of the adjacency graph of the SNF(f) of Figure 2

selected cube	neighbor cube 1	neighbor cube 2	neighbor cube 3	created cube	associated conjunction
001	-01	011	00-	1-0	$x_1\overline{x}_3$
-01	001	-11	-0-	1-0	$x_1\overline{x}_3$
011	-11	001	01-	1-0	$x_1\overline{x}_3$
00-	-0-	01-	001	1-0	$x_1\overline{x}_3$
01-	11-	00-	011	--0	$\overline{x}_3$
-0-	00-	---	-01	110	$x_1x_2\overline{x}_3$
-11	011	-01	-10	1--	x_1
-10	110	--0	-11	00-	$\overline{x}_1\overline{x}_2$
---	1--	-0-	--0	011	$\overline{x}_1x_2x_3$
11-	01-	1--	110	-01	$\overline{x}_2x_3$
--0	1-0	-10	---	001	$\overline{x}_1\overline{x}_2x_3$
110	-10	1-0	11-	001	$\overline{x}_1\overline{x}_2x_3$
1--	---	11-	1-0	001	$\overline{x}_1\overline{x}_2x_3$
1-0	--0	110	1--	001	$\overline{x}_1\overline{x}_2x_3$

It seems that the regularity of the adjacency graph of the SNF(f) inhibits the selection of suitable cubes for the reconstruction of the exact minimal ESOP of f. Table 2 enumerates this uniform information for the adjacency graph of the Boolean function (8).

Table 2. Degrees of the vertices of the adjacency graph of the SNF(f) of Figure 2

Vertex	Degree
001	3
-01	3
011	3
00-	3
01-	3
-0-	3
-11	3
-10	3
---	3
11-	3
--0	3
110	3
1--	3
1-0	3

An approach was suggested in [13] that overcomes this restriction. Its idea is derived from the basic method to create a minimal ESOP from a given SNF(f) that consists of two steps:

1. add a minimal number of pairs of cubes to the SNF which fill up the partial hypercubes covered by the SNF,
2. compress the hypercubes inversely to the expansion algorithm.

The pairs of cubes to be added are not part of the SNF. At least some of the cubes to be added have a distance one to some of the SNF cubes. Each cube of an ESOP has $2 * k$ cubes of a distance one, due to formulas (1), (2), or (3) and the number k of variables in the function. The SNF covers exactly k of these cubes. The other k cubes are located in a distance-one wrapper outside of the SNF and can be used to find suitable basic cubes for HCCC in the adjacency graph of the SNF(f).

Definition 3 ***- Distance-one wrapper cubes of the SNF(f) -*** *Each cube from the same Boolean space like f that does not belong to SNF(f), but has a distance of one to at least one cube of the SNF(f) is a distance-one wrapper cube of the SNF(f).*

Figure 3 shows all distance-one wrapper cubes of the SNF(f) of the function (8) as nodes visualized in dotted circles around the adjacency graph depicted by solid lines and circles.

Definition 4 ***- Extended adjacency graph of the SNF(f):***
EAG$^{SNF(f)}$(V,E) - *The extended adjacency graph of the SNF(f) consists of the adjacency graph AG$^{SNF(f)}$(V,E) of the SNF(f) as core extended by vertices of all distance-one wrapper cubes of the SNF(f) and edges between these wrapper vertices and the core vertices of the SNF cubes with a distance of one. There are no edges between the wrapper vertices.*

Figure 3 shows the extended adjacency graph EAG$^{SNF(f)}$(V,E) of the SNF(f) of the function (8) as example. Notice: each node of the embedded adjacency graph is connected with 6 nodes, 3 nodes of the embedded adjacency graph and 3 nodes of distance-one wrapper cubes, because the represented function (8) depends on $k = 3$ variables.

The degree of a vertex is the number of edges ending at that vertex. In contrast to the degree of the adjacency graph AG$^{SNF(f)}$(V,E) the degree of the wrapper vertices of the extended adjacency graph EAG$^{SNF(f)}$(V,E) is not unique. Table 3 enumerates the degrees of the wrapper vertices of EAG$^{SNF(f)}$(V,E) of the Boolean function (8).

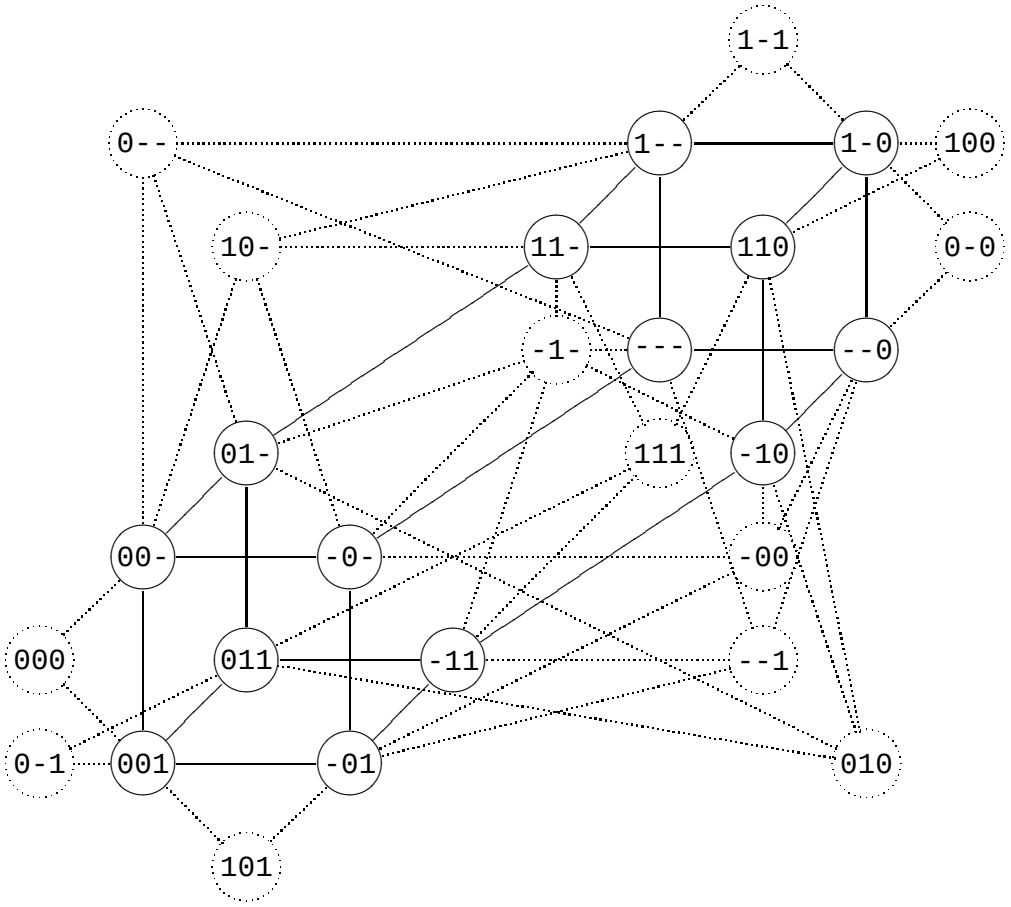


Fig. 3. Extended adjacency graph $AG^{SNF(f)}(V,E)$ of $f(x_1,x_2,x_3) = x_1\overline{x}_3 \oplus \overline{x}_1\overline{x}_2x_3$

Weights of the core vertices can be calculated using the degrees of the wrapper vertices of EAG$^{SNF(f)}$(V,E) as described in the algorithm 3. The `degree()` function in line 2 counts the number of edges ending at the given vertex.

The weights of the wrapper vertices of the Boolean function (8) are shown in the second column of Table 3. The weights of the core vertices are enumerated in Table 4. The minimal weights of core vertices in the EAG$^{SNF(f)}$(V,E) indicate cubes to be selected for the reconstruction procedure of the exact minimal ESOP introduced above.

The minimal weight in Table 4 is 6 for the SNF vertices labeled by (`001`) and (`1-0`), respectively. In Table 1 the associated cubes $x_1\overline{x}_3$ and $\overline{x}_1\overline{x}_2x_3$ of the minimal ESOP of the function $f(x_1,x_2,x_3) = x_1\overline{x}_3 \oplus \overline{x}_1\overline{x}_2x_3$ are given.

Table 3. Degree of the wrapper vertices of the extended adjacency graph of the SNF(f) of Figure 2. In the third column the adjacent vertices of the wrapper vertices in AG$^{SNF(f)}$(V,E) are enumerated.

Vertex	Degree	Adjacent Vertices of Wrapper Vertices
101	2	001 -01
0-1	2	001 011
000	2	001 00-
--1	4	-01 -11 --0 ---
-00	4	-01 -0- -10 --0
0--	4	01- --- 1-- 00-
-1-	6	01- -0- -11 11- -10 ---
010	4	011 01- 110 -10
111	4	011 -11 11- 110
10-	4	-0- 11- 1-- 00-
0-0	2	1-0 --0
100	2	1-0 11-
1-1	2	1-0 1--

Algorithm 3 Calculate Weights for the vertices of the extended adjacency graph EAG$^{SNF(f)}$(V,E)

Require: extended adjacency graph EAG$^{SNF(f)}$(V,E) of a Boolean function f
Ensure: weights of all vertices of EAG$^{SNF(f)}$(V,E)

1: **for all** wrapper vertices $V_w[i]$ of EAG$^{SNF(f)}$(V,E) **do**
2: $\quad$ *weight*$(V_w[i]) \leftarrow$ `degree`$(V_w[i])$
3: **end for**
4: **for all** core vertices $V_c[j]$ of EAG$^{SNF(f)}$(V,E) **do**
5: $\quad$ *weight*$(V_c[j]) \leftarrow 0$
6: $\quad$ **for all** adjacent wrapper vertices $V_w[i]$ of the extended adjacency graph EAG$^{SNF(f)}$(V,E) **do**
7: $\quad\quad$ *weight*$(V_c[j]) \leftarrow$ *weight*$(V_c[j])$ + *weight*$(V_w[i])$
8: $\quad$ **end for**
9: **end for**

4 Peculiarities of Most Complex Boolean Functions

There are 2^{2^n} Boolean function of n variables. These functions can be expressed by ESOPs. The number of different cubes in an ESOP is given by the interval $[0, 3^n]$. Thus, the number of cubes of an ESOP is a simple complexity measure. We restrict to ESOPs that do not include any cube twice. There are 2^{3^n} such ESOPs of n variables.

Table 4. Weights (right column) of the core vertices (left column) of the extended adjacency graph of the SNF(f) of Figure 2

Vertex	Weight
001	6
-01	10
011	10
00-	10
01-	14
-0-	14
-11	14
-10	14
---	14
11-	14
--0	10
110	10
1--	10
1-0	6

Using the SNF, it was shown in [6] that there are, for each Boolean function, exactly $2^{3^n}/2^{2^n}$ different ESOPs that do not include any cube more than once. A more precise measure of the complexity of a Boolean function is therefore the number of cubes in its shortest ESOP. The problem with this measure is the difficulty of finding the shortest ESOP, and several shortest ESOPs of the same function can exist.

A unique ESOP of each Boolean function is defined by its SNF. Thus, the number of cubes in the SNF is a next candidate for a complexity measure of a Boolean function. It is known from [6], [12], and [13] that all Boolean functions with the largest SNF belong to the set of Boolean functions that need the largest number of cubes in their shortest ESOP. This statement does not hold for the reverse direction. Evidently, in the set of such Boolean functions that require the largest number of cubes in their shortest ESOP, there is a subset of functions that belong to the set of functions with the largest number of cubes in its SNF.

See for example Table 8. There are 66 functions of three variables that require three cubes in their shortest ESOP. 54 of these functions have 16 cubes in their SNF, and the remaining 12 functions have 18 cubes in their SNF. Obviously, the number of cubes in a minimal ESOP conceals the complexity of a Boolean function which can be detected by the SNF. This leads to the following definition.

Definition 5 ***- Most complex Boolean functions*** f ***-*** *The set of Boolean functions of n variables that needs the largest number of cubes in their unique SNF(f) is called the set of most complex Boolean functions.*

There are peculiarities of most complex Boolean function that affect the HCCC function mentioned above. These peculiarities will be discussed in the following. Initially it is amazing that the same peculiarities occur for certain classes of simpler functions of a Boolean space. The reason for that observation will be explained too.

The HCCC function evaluates the weight of the extended adjacency graph $EAG^{SNF(f)}$(V,E) calculated by algorithm 3. In the simple case where the function can be expressed by a single cube, all SNF cubes carry the same weight. It is irrelevant which cube of the SNF is selected for the HCCC function. Based on each of them the HCCC function creates the same correct solution cube. This case can be detected by formula (9). Such an adjacency graph can be called *completely convex*, because it describes an hypercube, and the HCCC function finds the same solution for each SNF cube.

$$|SNF(f(x_1,\ldots,x_n)| = 2^n. \tag{9}$$

Generally, if more than one cube is required to express a function by a minimal ESOP, the smallest weights calculated by algorithm 3 indicate the basic cubes required for the HCCC function. The adjacency graph does not include any complete hypercube in this case. The smallest weights indicate remaining hypercube cube corners of cubes belonging to the minimal ESOP. For that reason such adjacency graphs are called *partially convex*.

An interesting observation is that the larger the number of cubes in the SNF the smaller is the difference between the values of the weights calculated by algorithm 3. One exception of this rule is the simple case of a complete convex adjacency graph mentioned above. The second more important peculiarity occurs in the case of a most complex function.

All weights calculated by algorithm 3 have the same value if a most complex function is evaluated. However, there is not any remaining hypercube cube corner of cubes belonging to the minimal ESOP. Such an adjacency graph is therefore called *completely concave*. This peculiarity in the associated adjacency graph of a most complex function requires a modification in the HCCC function.

Each cube in the adjacency graph of a most complex function can be taken to create its minimal ESOP. Due to the completely concave adjacency graph all pseudo-hypercube cube corners point in the inverse direction. For that reason this direction must change by changing exactly one value in the solution cube. The last column in Table 5 enumerates the values that must be taken in this special case. A precondition is that for a chosen variable all three possible values appear in the SNF.

In a first experiment the exact minimal ESOPs of all Boolean functions of a few

Table 5. Assignment of values in the HCCC function

value of x_j in the selected basic cube	value of x_j in the adjacent SNF cube	default value of x_j in the solution cube	peculiar value of x_j in the solution cube
0	1	–	1
0	–	1	–
1	0	–	0
1	–	0	–
–	0	1	0
–	1	0	1

Boolean variables were calculated. For that purpose each Boolean function f of a fixed number of variables was first created, in a second step transformed into the SFN(f), and in a third step minimized by the algorithm described above. Finally, each function was associated to the class characterized by the number of cubes of the SNF in combination with the number of cubes of the exact minimal ESOP of f. The cubes of all exact minimal ESOPs were selected by the modified HCCC function.

The Tables 6, 7, 8, and 9 show protocols of the calculations of all exact minimal ESOPs of $f : B^k \to B, k = 1, \ldots, 4$ using the modified HCCC function explained above. In these Tables the following symbols have been used:

# ALLBF	the number of all Boolean functions in the Boolean space,
# SNF	the number of cubes of the SNF,
# EMIN	the number of cubes of the exact minimal ESOP, and
# BF	the number of Boolean functions, specified by # SNF and # EMIN.

Generally the Tables 6, 7, 8, and 9 show both the dependency of most complex functions on the associated Boolean space and a classification of all Boolean function by the number of cubes in the SNF and in the exact minimal ESOP. These Tables confirm that the maximal number of cubes in an exact minimal ESOP occur in most complex functions. However, Table 8 shows that there exist some func-

Table 6. Complexity classes of all exact minimal ESOPs of 1 variable

```
# ALLBF = 4
# SNF = 0    # EMIN = 0  # BF 1
# SNF = 2    # EMIN = 1  # BF 3
```

Table 7. Complexity classes of all exact minimal ESOPs of 2 variables

```
# ALLBF = 16
# SNF = 0     # EMIN = 0  # BF 1
# SNF = 4     # EMIN = 1  # BF 9
# SNF = 6     # EMIN = 2  # BF 6
```

Table 8. Complexity classes of all exact minimal ESOPs of 3 variables

```
# ALLBF = 256
# SNF = 0      # EMIN = 0  # BF 1
# SNF = 8      # EMIN = 1  # BF 27
# SNF = 12     # EMIN = 2  # BF 54
# SNF = 14     # EMIN = 2  # BF 108
# SNF = 16     # EMIN = 3  # BF 54
# SNF = 18     # EMIN = 3  # BF 12
```

tions not belonging to the most complex functions, but having the same number of cubes in their minimal ESOPs.

From these complete enumerations further information about most complex functions can be discovered. First, $n_{max}^{SNF}(k)$ is the number of cubes of the most complex SNF(f) of k variables

$$n_{max}^{SNF}(k) = 2 * 3^{k-1}, \tag{10}$$

and second, $n_{max}^{BF}(k)$ is the number of the most complex SNF(f) of k variables

$$n_{max}^{BF}(k) = 3 * 2^{k-1}. \tag{11}$$

Both formulas (10), and (11) generalize information for the last lines of Tables 6, 7, 8, and 9.

The set of all Boolean functions of k variables, $k > 1$, includes all Boolean functions that can be expressed by $k-1$ variables. This property influences the HCCC function such that completely concave adjacency graphs occur not only for the most complex functions of k variables, but also for SNF classes that include

Table 9. Complexity classes of all exact minimal ESOPs of 4 variables

```
# ALLBF = 65536
# SNF = 0        # EMIN = 0  # BF 1
# SNF = 16       # EMIN = 1  # BF 81
# SNF = 24       # EMIN = 2  # BF 324
# SNF = 28       # EMIN = 2  # BF 1296
# SNF = 30       # EMIN = 2  # BF 648
# SNF = 32       # EMIN = 3  # BF 648
# SNF = 34       # EMIN = 3  # BF 3888
# SNF = 36       # EMIN = 3  # BF 6624
# SNF = 36       # EMIN = 4  # BF 108
# SNF = 38       # EMIN = 3  # BF 7776
# SNF = 40       # EMIN = 3  # BF 2592
# SNF = 40       # EMIN = 4  # BF 6642
# SNF = 42       # EMIN = 3  # BF 216
# SNF = 42       # EMIN = 4  # BF 14256
# SNF = 44       # EMIN = 4  # BF 12636
# SNF = 46       # EMIN = 4  # BF 3888
# SNF = 46       # EMIN = 5  # BF 1296
# SNF = 48       # EMIN = 5  # BF 1944
# SNF = 50       # EMIN = 5  # BF 648
# SNF = 54       # EMIN = 6  # BF 24
```

the most complex functions of smaller Boolean spaces. Table 10 enumerates such classes of SNF by its cube numbers, in which the minimal and the maximal weights calculated by algorithm 3 are equal. These classes can be calculated for each k by (12).

$$n^{SNF}_{w_{min}=w_{max}}(k,i) = 2^{(k-i)} * 3^i \,|k > 1, i = 1,\ldots,k-1 \tag{12}$$

5 Calculation of Most Complex Boolean Functions

Table 10. Classes of SNFs, where $weight_{min} = weight_{max}$

number of variables	#SNF, where $weight_{min} = weight_{max}$		
k	$i=1$	$i=2$	$i=3$
2	$2^1 * 3^1 = 6$		
3	$2^2 * 3^1 = 12$	$2^1 * 3^2 = 18$	
4	$2^3 * 3^1 = 24$	$2^2 * 3^2 = 36$	$2^1 * 3^3 = 54$

Algorithm 4 Calculate most complex function MCF(k,s)

Require: k: number of variables in the function, $k \geq 1$
s: selection index of the most complex function, $0 \leq s \leq 3 * 2^{k-1}$
Ensure: function vector mcf_s of the most complex function, defined by k and s

```
1:  if k = 1 then
2:     mcf_s ← s + 1
3:  else
4:     remainder ← s mod 3
5:     selection ← s/3
6:     direction ← selection mod 2
7:     base ← (selection/2) * 3
8:     len2 ← 2^(k-1)
9:     if direction = 0 then
10:       mcf_s ← [MCF(k-1, base + ((remainder + 1) mod 3))] * 2^len2
                  + MCF(k-1, base + ((remainder + 2) mod 3))
11:    else
12:       mcf_s ← [MCF(k-1, base + ((remainder + 3 - 1) mod 3))] * 2^len2
                  + MCF(k-1, base + ((remainder + 3 - 2) mod 3))
13:    end if
14: end if
```

A detailed evaluation of all most complex function showed that these functions can be calculated by a recursive algorithm. The required base cases are given by the most complex functions of one variable. The last line in Table 6 shows that there are three most complex function of one variable. These functions can be enumerated by their function vectors: (01), (10), and (11). These base cases are implemented in line 2 of algorithm 4. The algorithm 4 calculates each most complex function recursively.

The lines 4 to 8 of algorithm 4 prepare the recursion. The Boolean value *direction* decides about the direction in which basic functions from a lower level of recursion have to be combined. The multiplication by 2^{len2} in lines 10 and 12 move the associated function vector by its size bits so that the most outer additions in these lines combines the basic functions.

Of cause, the simple operations $3-1$ and $3-2$ in line 12 of the algorithm can be substituted by the values 2 and 1, respectively. The used style emphasizes the inverse direction in comparison to line 10 of algorithm 4. Two recursive calls of the function MFC(k,s) either in line 10 or in line 12 generate the required basic function.

Table 11 enumerates all most complex Boolean functions depending on $k = 1,\ldots,4$ variables. The content of this Table was calculated using the algorithm 4. It can be seen in Table 11 that the root of all most complex functions are the trivial functions $f_1(x) = \overline{x}$, $f_2(x) = x$, and $f_3(x) = 1$. Using the recursive algorithm 4 any most complex Boolean function can be calculated very quickly.

Table 11. Enumeration of most complex Boolean functions by hexadecimal function vectors

k = 1	k = 2	k = 3	k = 4
3 MCF of	6 MCF of	12 MCF of	24 MCF of
1 variable	2 variables	3 variables	4 variables
mcf[0] = 1	mcf[0] = b	mcf[0] = d6	mcf[0] = 6bbd
mcf[1] = 2	mcf[1] = d	mcf[1] = 6b	mcf[1] = bdd6
mcf[2] = 3	mcf[2] = 6	mcf[2] = bd	mcf[2] = d66b
	mcf[3] = e	mcf[3] = 6d	mcf[3] = bd6b
	mcf[4] = 7	mcf[4] = b6	mcf[4] = d6bd
	mcf[5] = 9	mcf[5] = db	mcf[5] = 6bd6
		mcf[6] = 79	mcf[6] = b6db
		mcf[7] = 9e	mcf[7] = db6d
		mcf[8] = e7	mcf[8] = 6db6
		mcf[9] = 97	mcf[9] = dbb6
		mcf[10] = e9	mcf[10] = 6ddb
		mcf[11] = 7e	mcf[11] = b66d
			mcf[12] = 9ee7
			mcf[13] = e779
			mcf[14] = 799e
			mcf[15] = e79e
			mcf[16] = 79e7
			mcf[17] = 9e79
			mcf[18] = e97e
			mcf[19] = 7e97
			mcf[20] = 97e9
			mcf[21] = 7ee9
			mcf[22] = 977e
			mcf[23] = e997

6 Conclusion

This paper investigates most complex Boolean functions. The complexity was measured by the number of cubes in an ESOP. Among several possibilities in order to specify most complex Boolean functions precisely, a definition of most complex

Boolean functions has been given. This definition has been based on the SNF which was introduced briefly.

In the context of most complex functions a peculiarity of adjacency graphs was detected. While the adjacency graphs of all other Boolean functions are at least partially convex, all most complex Boolean functions lead to concave adjacency graphs. This affects the selection of cubes belonging to the exact minimal ESOP and requires a modification of the HCCC function. This modification was explained in detail.

The most complex functions of a given Boolean space are members of the set of all function of any larger Boolean space. Although such a function does not belong to the class of most complex functions in the larger Boolean space, all of its weights calculated by algorithm 3 have the same value. This indicates that the property of a concave adjacency graphs is strongly connected to the function which belongs in any Boolean space to the set of the most complex functions. The modified HCCC function selects also in this case a right column for the particular association of the value based on Table 5.

The experimental results confirm that the modified HCCC function creates exact minimal ESOPs. Using these results, formulas for both the number of cubes in the SNF for most complex Boolean functions and the number of most complex Boolean functions were explored. Based on these quantities an recursive algorithm was found that generates any most complex Boolean function depending on the number of variables and an order number of the most complex function. This algorithm generates very quickly any most complex Boolean function. For the Boolean spaces $B^k, k = 1,\ldots,4$, all most complex function was calculated and enumerated in Table 11.

The classification of Boolean functions based on both the number of cubes in their SNFs and the number of cubes in their exact minimal ESOPs extends the knowledge about Boolean functions and may be a root for further theoretical results. The knowledge about peculiarities of most complex Boolean functions supports these research.

References

[1] I. Wegener, *The Complexity of Boolean Functions*. Stuttgart, Germany: John Wiley & Sons Ltd, and B. G. Teubner, 1987.

[2] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Computers*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.

[3] R. Drechsler and B. Becker, *Graphenbasierte Funktionsdarstellung - Boolesche und Pseudo-Boolesche Funktionen*. Stuttgart, Germany: Teubner Verlag, 1998.

[4] C. Posthoff and B. Steinbach, *Logic Functions and Equations - Binary Models for Computer Science*. Dordrecht, The Netherlands: Springer, 2004.

[5] B. Steinbach and C. Posthoff, "Extended Theory of Boolean Normal Forms," in *Proceedings of the 6th Annual Hawaii International Conference on Statistics*, Honolulu, Hawaii, 2007, p. 1124 1139.

[6] B. Steinbach and A. Mishchenko, "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)*, Mississippi State University, Starkville (Mississippi) USA, 2001, pp. 66 – 81.

[7] T. Sasao and M. Fujita, *Representation of Discrete Functions*. Boston, US; London, UK; Dordrecht, The Netherlands: Kluwer Academic Publishers, 1996.

[8] M. Helliwell and M. A. Perkowski, "A Fast Algorithm to Minimize Multi-Output Mixed-Polarity Generalized Reed-Muller Forms," in *Proceedings of the 25th ACM/IEEE conference on Design automation*, Atlantic City, New Jersey, US, 1988, pp. 427–432.

[9] T. Sasao, "An Exact Minimization of AND-EXOR Expressions Using BDDs," in *Proceedings of IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design (RM 1993)*, Wilhelm Schickard-Institute für Informatik, Hamburg, Germany, 1993, pp. 91 – 98.

[10] A. Gaidukov, "Algorithm to derive minimum ESOP for 6-variable function," in *Proceedings of the 5th International Workshops on Boolean Problems*, Freiberg University of Mining and Technology, Freiberg, Germany, 2002, pp. 141– 148.

[11] S. Stergios and G. Papakonstantinou, "Towards a General Novel Exact ESOP Minimization Methodology," in *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technology (RM 2003)*, University of Trier, Germany, 2003, pp. 19 – 26.

[12] B. Steinbach, V. Yanchurkin, and M. Lukac, "On SNF Optimization: a functional comparison of methods," in *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technology (RM 2003)*, University of Trier, Germany, 2003, pp. 11 – 18.

[13] B. Steinbach, "Adjacency Graph of the SNF as Source of Information," in *Proceedings of the 7th International Workshops on Boolean Problems*, Freiberg University of Mining and Technology, Freiberg, Germany, 2006, pp. 19 – 28.