# On the Optimal OBDD Representation of 2-XOR Boolean Affine Spaces

Anna Bernasconi
Dep. of Compure Science
Università di Pisa, Italy
anna.bernasconi@unipi.it

Valentina Ciriani
Dep. of Compure Science
Università degli Studi di Milano, Italy
valentina.ciriani@unimi.it

Marco Longhi
Dep. of Compure Science
Università degli Studi di Milano, Italy
marco.longhi2@studenti.unimi.it

***Abstract*—A Reduced Ordered Binary Decision Diagram (ROBDD) is a data structure widely used in an increasing number of fields of Computer Science. In general, ROBDD representations of Boolean functions have a tractable size, polynomial in the number of input variables, for many practical applications. However, the size of a ROBDD, and consequently the complexity of its manipulation, strongly depends on the variable ordering: depending on the initial ordering of the input variables, the size of a ROBDD representation can grow from linear to exponential. In this paper, we study the ROBDD representation of Boolean functions that describe a special class of Boolean affine spaces, which play an important role in some logic synthesis applications. We first discuss how the ROBDD representations of these functions are very sensitive to variable ordering, and then provide an efficient linear time algorithm for computing an optimal variable ordering that always guarantees a ROBDD of size linear in the number of input variables.**

***Index Terms*—Ordered Binary Decision Diagram, variable ordering, affine space representation.**

## I. Introduction

Reduced Ordered Binary Decision Diagrams (ROBDDs) are a data structure for the symbolic representation and manipulation of Boolean functions [6], [8], [13]–[16], [19]. Even if ROBDDs have often a tractable size for many practical applications, in the worst case, the ROBDD representing a Boolean function can have an exponential size in the number of input variables [6]. Moreover, several Boolean functions are very sensitive to variable ordering, i.e., their ROBDD size can range, depending on the fixed variable ordering, from linear to exponential [6]. Some other functions, as for instance totally symmetric functions, have ROBDDs of polynomial size for any variable ordering. There are also functions with exponential size ROBDDs for all possible variable orderings, e.g., integer multiplication [8].

In general, the problem of finding the optimal variable ordering for the ROBDD representation of a Boolean function $f$ is NP-hard, even when $f$ is given as a ROBDD [5]. In summary, in order to efficiently manipulate a ROBDD for a given Boolean function, it is very important to identify a good variable ordering, but the computation of the optimal variable ordering can have exponential complexity. In the literature, several classes of Boolean functions have been studied in order to find, for them, an optimal ROBDD variable ordering. Consider, for example, read-once functions, which are Boolean functions that can be expressed with a formula in which each variable occurs exactly once. An optimal variable ordering for read-once functions can be computed in a time linear in the number of input variables [20]. Moreover, methods to improve the quality of ordering heuristics for partially symmetric functions have been discussed in [21].

In this paper, we study the optimal variable ordering of another class of Boolean functions, i.e., the characteristic functions of affine spaces. Boolean affine spaces are translations of Boolean vector spaces. Their algebraic representation is an AND of XORs of literals, also called CEX expression. CEX expressions are often exploited in logic synthesis for three level logic minimization, e.g., for SPP and 2-SPP forms [1], [9], [18], [23], [24]; or for handling regular functions, e.g., autosymmetric functions [2], [18]. In all these contexts, the compact ROBDD representation of CEX expressions is crucial for guaranteeing reduced synthesis times. In this paper, we concentrate on 2-CEX expressions. A 2-CEX is an AND of XOR factors that contain at most two literals, i.e., 2-XOR. 2-CEXs are the CEX expressions that are much often exploited in logic synthesis since, for technological reasons, 2-XOR gates are preferable to unbounded XOR gates. A simple example of 2-CEX expression is given by the function $f = (x_1 \oplus x_2)(x_3 \oplus x_4)\dots(x_{2n-1} \oplus x_{2n})$. In particular, 2-CEX expressions are often encountered in the synthesis of autosymmetric functions, which represent a significant fraction of standard benchmark functions [2], [18]. Indeed, about $24\%$ of the functions in the classical ESPRESSO benchmark suite have at least one truly (i.e., non degenerate) autosymmetric output. Moreover, in the three level 2-SPP synthesis [1], the minimization algorithms manipulate 2-CEX expressions.

The main result of this paper is that we can always compute an optimal variable ordering for 2-CEX expressions in linear time. Moreover, we show that the ROBDD, with optimal variable ordering, has size linear in the number of variables. We validate the proposed approach through a set of experiments.

## II. Preliminaries

### A. Affine spaces and CEX representation

Consider the Boolean space $\{0,1\}^n$ described by $n$ variables $x_1, x_2, \dots, x_n$, where each point is described by a binary vector of $n$ *components*. Hereafter, we shall use the terms vector and point with the same meaning. In the space $\{0,1\}^n$, a ***XOR factor*** is a XOR (or modulo 2 sum), denoted by $\oplus$, of variables,

978-3-9819263-6-1/DATE22/©2022 EDAA

one of which possibly complemented (a XOR with just one literal corresponds to the literal itself). We can extend the symbol $\oplus$ to denote the elementwise XOR of two vectors.

Given a vector $\alpha \in \{0,1\}^n$ and a vector subspace $V$ of $(\{0,1\}^n, \oplus)$, we can build an *affine space* $A$ performing the XOR between $\alpha$ and each point of $V$. Consider the vector space $V = \{0000,\ 0110,\ 1010,\ 1100\}$ and the vector $\alpha = 0010 \in \{0,1\}^4$. The set $A = \alpha \oplus V = 0010 \oplus V = \{0010,\ 0100,\ 1000,\ 1110\}$ is an affine space over $V$.

An affine space $A$ can be represented by an algebraic expression involving AND and XOR operators, that can be derived from a base for its unique vector space $V$ as follows. Consider a $2^k \times n$ matrix $M$ whose rows correspond to the points of a vector space of dimension $k$, and whose columns correspond to the variables $x_1, x_2, \ldots, x_n$. Let the row indices of $M$ be numbered from 0 to $2^k - 1$. We say that $M$ is in *binary order* if its rows are sorted as increasing binary numbers.

*Definition 1 ( [3]):* Let $V$ be a vector space whose matrix is sorted in binary order, with the rows indexed from 0 to $2^k - 1$. And let $A = \alpha \oplus V$ be an affine space over $V$. The set of points of $V$ with indices $2^0, 2^1, \ldots, 2^{k-1}$ will be called the *canonical basis* $B_A$ of $V$ (or, equivalently, of $A$). Moreover, the $k$ variables corresponding to the first 1-component from left of each vector of $B_A$ are called *canonical variables*. The variables that are not canonical are called *non-canonical*.
The canonical basis $B_A$ corresponds to the basis derived by a matrix in *reduced row echelon form* [3], [4], [11], [17].

*Definition 2 ( [3]):* The *canonical representation* $(\alpha_A, B_A)$ of an affine space $A$ is given by the minimum point $\alpha_A$ of $A$ in binary order together with the canonical basis $B_A$.
Consider for example the affine space $A = \{0010,\ 0100,\ 1000,\ 1110\}$ and its associated vector space $V = \{0000,\ 0110,\ 1010,\ 1100\}$, that can be represented in binary order by the following matrix $M_V$:

$$M_V = \begin{matrix} 0\,0\,0\,0 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \\ 1\,1\,0\,0 \end{matrix}$$

The canonical translation point $\alpha_A$ is the vector 0010, and the canonical basis $B_A$ is given by the second and the third row of $M_V$, i.e., $B_A = \{0110, 1010\}$. The canonical variables are $x_1$ for vector 1010 and $x_2$ for vector 0110. The non-canonical variable are the remaining variables $x_3$ and $x_4$.

The canonical variables are the truly independent variables in the affine space $A$, in the sense that they can assume all possible combinations of 0-1 values. On the contrary, on $A$ the non-canonical variables are not independent because they can be defined as XORs of the canonical ones: their values are uniquely defined by the values of the canonical variables, or are constant. For instance, in our running example, while the two variables $x_1$ and $x_2$ assume all possible 0-1 combinations on $A$ and on the associated vector space $V$, $x_3$ is always equal to $x_1 \oplus x_2 \oplus 1$ on $A$, and always equal to $x_1 \oplus x_2$ on $V$, while $x_4$ is always equal to 0, both on $A$ and on $V$.

This fact is expressed by the characteristic function of an affine space, that can be derived from the canonical representation of $A$. Indeed, an affine space of dimension $k$ can be represented by a canonical expression called *CEX* [10], defined as the AND of $n - k$ XOR factors, where each XOR factor: 1) contains exactly one different non-canonical variable, which is complemented if and only if its corresponding component in $\alpha_A$ is 0; 2) expresses the dependence of the non-canonical variable on the canonical ones.

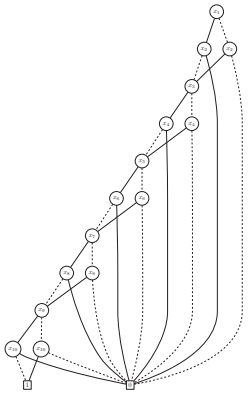

Fig. 1. ROBDD of the function $f = (x_1 \oplus x_2)(x_3 \oplus x_4)(x_5 \oplus x_6)(x_7 \oplus x_8)(x_9 \oplus x_{10})$ with optimal variable ordering $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$.

Now, observe that the CEX actually corresponds to a system of linear equations, whose solutions are exactly all vectors of the space described by the CEX. Indeed, since the CEX is a conjunction of XOR factors, it assumes the value 1 only on the vectors of $\{0,1\}^n$ that satisfy each single XOR factor. For instance, the CEX of the affine space $A$ in our running example is simply given by $(x_1 \oplus x_2 \oplus x_3)\overline{x}_4$, that is satisfied by all vectors whose components $x_1$, $x_2$, $x_3$, and $x_4$ are such that

$$\begin{cases} x_1 \oplus x_2 \oplus x_3 &= 1 \\ \overline{x}_4 &= 1\,. \end{cases}$$

Note that these equations can be rewritten as $x_3 = x_1 \oplus x_2 \oplus 1$ and $x_4 = 0$, which are precisely the linear combinations that defines the non-canonical variables $x_3$ and $x_4$ on $A$. Also, noticed that the non-canonical variable $x_3$ is not complemented as the third component of $\alpha_A$ is 1, while $x_4$ is complemented since the fourth component of $\alpha_A$ is 0.

We now turn our attention to the special case of affine spaces whose CEX contains XOR factors of at most two literals:

*Definition 3:* A *2-CEX* is an AND of XOR factors that contain at most two variables: a non-canonical variable and possibly a canonical one.
For example, the CEX $(x_1 \oplus x_2)(x_1 \oplus \overline{x}_5)x_6(x_3 \oplus x_7)\overline{x}_8$ in $\{0,1\}^9$ is a 2-CEX. When the 2-CEX is actually a simple product of literals, the system contains only the non-canonical variables. In this case, the CEX describes an affine space where all non-canonical variables assume a constant 0 or 1 value.

### B. Binary Decision Diagrams

A *Binary Decision Diagram* (BDD) over a set of Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ is a rooted, connected direct acyclic graph, where each non-terminal (internal) node $N$ is labeled by a Boolean variable $x_i$ and has exactly two outgoing

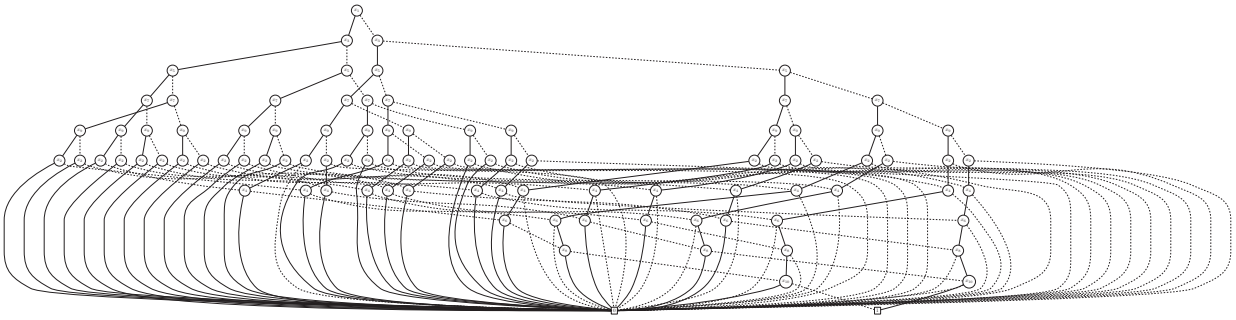

Fig. 2. ROBDD of the function $f = (x_1 \oplus x_2)(x_3 \oplus x_4)(x_5 \oplus x_6)(x_7 \oplus x_8)(x_9 \oplus x_{10})$ with non-optimal variable ordering $x_1, x_3, x_5, x_7, x_9, x_2, x_4, x_6, x_8, x_{10}$. The corresponding ROBDD with optimal variable ordering is depicted in Figure 1.

edges, the 0-edge and the 1-edge, pointing to two nodes called the 0-child and the 1-child of node $N$, respectively. Terminal nodes (leaves) are labeled 0 or 1. For example, consider the BDD in Figure 1 with variables $x_1, ..., x_{10}$. Each pointer to a 1-child is depicted with a solid line, while each pointer to a 0-child is depicted with a dashed line. Binary decision diagrams are typically used to efficiently represent and manipulate Boolean functions [6], [7].

A BDD is *ordered* if there exists a total order $<$ over the set $X$ of variables such that if an internal node is labeled by $x_i$, and its 0-child and 1-child have labels $x_{i_0}$ and $x_{i_1}$, respectively, then $x_i < x_{i_0}$ and $x_i < x_{i_1}$. A BDD is *reduced* if there exist no nodes whose 1-child is equal to the 0-child and there not exist two distinct nodes that are roots of isomorphic subgraphs. A reduced and ordered BDD is called *ROBDD*.

ROBDDs are usually quite compact, but there are functions whose ROBDD representation has a size, i.e., number of nodes, exponential in the number of input variables. Moreover, depending on the variable ordering, the size of the ROBDD representing a function can drastically change, from linear to exponential in the number of the input variables. It is therefore very important to properly select the variable ordering when applying ROBDDs in practice. However, the problem of finding the best variable ordering is NP-hard [5]. Note that the representation of Boolean functions with ROBDDs allows to perform operations that do not depend on the number of inputs that are equal to 1 or 0; for this reason, algorithms based on ROBDDs are usually defined implicit algorithms.

## III. Variable ordering for affine spaces

In this section we show how to derive a variable ordering for the optimal ROBDD representation of Boolean affine spaces represented by 2-CEX algebraic expressions. We first give several definitions in order to better describe the proposed solution. In a Boolean space $\{0,1\}^n$ described by $n$ variables $x_1$, ..., $x_n$, let a 2-*XOR* be a XOR with *at most* 2 input variables, one of which possibly complemented. Given two distinct Boolean variables $x_i, x_j$, all the possible 2-XORs are essentially $x_i$, $\overline{x}_i$, $x_j$, $\overline{x}_j$, $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$. (Notice that $\overline{x}_i \oplus x_j = x_i \oplus \overline{x}_j$, and $\overline{x}_i \oplus \overline{x}_j = x_i \oplus x_j$.) For sake of clarity, we now consider 2-CEX expressions without complemented variables in the 2-XORs (all results can be extended to general 2-CEXs, as it will be shown in the full version of the paper).

We consider affine spaces whose CEX expressions contain only 2-XORs (i.e., 2-CEX). This choice enables the definition of a simple partition of the input variables that can be exploited to determine an optimal variable ordering for the ROBDD construction. Recall that each 2-CEX corresponds to a linear system as described in Section II-A. Any linear system $S$, corresponding to a 2-CEX $C$ with non-canonical variables $\{x_{l_1}, \ldots, x_{l_m}, x_{j_1}, \ldots, x_{j_k}\}$, can be rewritten in the form of an *equality system* $E$, as follows:

$$S = \begin{cases} x_{l_1} & = 1 \\ & \vdots \\ x_{l_m} & = 1 \\ x_{i_1} \oplus x_{j_1} & = 1 \\ & \vdots \\ x_{i_k} \oplus x_{j_k} & = 1\,. \end{cases} \qquad E = \begin{cases} 1 & = x_{l_1} \\ & \vdots \\ 1 & = x_{l_m} \\ \overline{x}_{i_1} & = x_{j_1} \\ & \vdots \\ \overline{x}_{i_k} & = x_{j_k}\,. \end{cases} \tag{1}$$

Note that the systems $S$ and $E$ are equivalent since $x_i \oplus x_j = 1$ if and only if $\overline{x}_i = x_j$.

*Definition 4:* Let $E$ be the equality system in the Boolean space described by the set of variables $\{x_1, x_2, \ldots, x_n\}$ in Equation (1) where $x_{i_h}, x_{j_h} \in \{x_1, x_2, \ldots, x_n\}$ for $1 \leq h \leq k$ and $x_{l_h} \in \{x_1, x_2, \ldots, x_n\}$ for $1 \leq h \leq m$. The *partition* derived from $E$ is the partition $P$ of the set $\{x_1, x_2, \ldots, x_n\}$ where, for any $x$ and $y$ in $\{x_1, x_2, \ldots, x_n\}$, $x$ and $y$ are in the same subset of the partition if and only if the equality $x = y$ can be derived from the system $E$.

*Example 1:* Consider the following 2-CEX containing 2-XORs only: $x_1(x_2 \oplus x_4)(x_3 \oplus x_5)x_6(x_3 \oplus x_7)$. In this 2-CEX the non-canonical variables are $x_1$, $x_4$, $x_5$, $x_6$ and $x_7$, and the canonical variables are $x_2$ and $x_3$. The 2-CEX corresponds to the following linear and equality systems:

$$\begin{cases} x_1 = 1 \\ x_2 \oplus x_4 = 1 \\ x_3 \oplus x_5 = 1 \\ x_6 = 1 \\ x_3 \oplus x_7 = 1 \end{cases} = \begin{cases} 1 = x_1 \\ \overline{x}_2 = x_4 \\ \overline{x}_3 = x_5 \\ 1 = x_6 \\ \overline{x}_3 = x_7 \end{cases} = \begin{cases} 1 = x_1 \\ 1 = x_6 \\ \overline{x}_2 = x_4 \\ \overline{x}_3 = x_5 \\ \overline{x}_3 = x_7 \end{cases}$$

From these systems we can derive the following variable equalities: $1 = x_1 = x_6$; $\overline{x}_2 = x_4$; $\overline{x}_3 = x_5 = x_7$.

In particular, $x_1$ and $x_6$ must be always equal to 1, $x_2$ and $x_4$ must have complemented values, and $\overline{x}_3$, $x_5$,

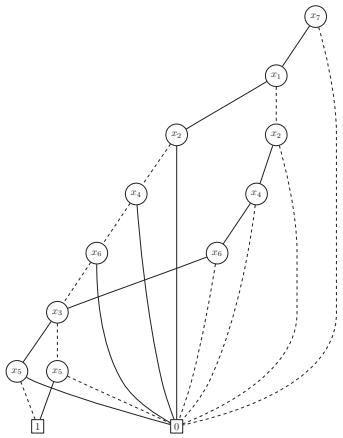

Fig. 3. ROBDD of $(x_1 \oplus x_2)(x_1 \oplus x_4)(x_3 \oplus x_5)(x_1 \oplus x_6)x_7$ with optimal variable ordering $x_7, x_1, x_2, x_4, x_6, x_3, x_5$.

and $x_7$ must have the same value. These equalities imply the following partition of the set $\{x_1, x_2, \ldots, x_7\}$: $\{\{x_1\}, \{x_6\}, \{x_2, x_4\}, \{x_3, x_5, x_7\}\}$. Notice that $x_1$ and $x_6$ are in separate subsets of the partition because their value does not depend on a canonical variable (i.e., it is fixed to 1).

### A. Optimal Variable Ordering

We now show how we can derive a variable ordering for a ROBDD of a 2-CEX starting from the linear systems and the corresponding partitioning of Boolean variables. From the partition in Definition 4 we can describe the following variable ordering:

*Definition 5:* Let $P$ be a partition derived from an equality system $E$ such that $P = \{p_1, p_2, \ldots, p_s\}$ and each $p_i$ $(1 \leq i \leq s)$ is a subset of $\{x_1, x_2, \ldots, x_n\}$. An *ordering $O$ derived from $P$* is $v_1, v_2, \ldots, v_s$ where each $v_i$ $(1 \leq i \leq s)$ contains the variables of $p_i$ in any order.

The overall algorithm for deriving a minimal variable ordering is then:

1) derive the equality system corresponding to the 2-CEX expression;
2) order the equalities with respect to the canonical variables and the constant 1 (consider the constant 1 first in the ordering; or, alternatively, last in the ordering);
3) perform a partition $P = \{p_1, p_2, \ldots, p_s\}$ of the set $\{x_1, x_2, \ldots, x_n\}$, such that in each subset $p_i$ $(1 \leq i \leq s)$ we have all the variables that are equal to the same canonical variable in the equality system;
4) output a variable ordering $O$ derived from $P$.

Note that Step 2) requires a sorting phase. Since the indices of the variables are in a finite set of $n$ elements, we can use the Counting Sort algorithm, which requires linear computational time [12]. It is simple to see that all other steps of the algorithm are also linear. It follows that the time complexity of this algorithm is in $O(n)$.

Example 1 shows the first three steps of the algorithm. The computed partition is $\{\{x_1\}, \{x_6\}, \{x_2, x_4\}, \{x_3, x_5, x_7\}\}$. The last step of the algorithm gives us the variable ordering: $x_1, x_6, x_2, x_4, x_3, x_5, x_7$ . The variable ordering described in Definition 5 is indeed an *optimal variable ordering* for the ROBDD representation of the corresponding 2-CEX, as we show in the sequel of this section. We first give the intuition of this optimality starting from the following example, which shows that even ROBDDs representing strong regular structures as affine spaces, are very sensitive to variable ordering.

*Example 2:* Consider, for instance, the 2-CEX expression $(x_1 \oplus x_2)(x_3 \oplus x_4)(x_5 \oplus x_6)(x_7 \oplus x_8)(x_9 \oplus x_{10})$ . The corresponding equality system is:

$$\begin{cases} \overline{x}_1 &= x_2 \\ \overline{x}_3 &= x_4 \\ \overline{x}_5 &= x_6 \\ \overline{x}_7 &= x_8 \\ \overline{x}_9 &= x_{10}\,. \end{cases}$$

Therefore, the partition is $P = \{\{x_1, x_2\}, \{x_3, x_4\}, \{x_5, x_6\}, \{x_7, x_8\}, \{x_9, x_{10}\}\}$ and the variable ordering is $O = x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$. Figure 1 shows the ROBDD with this variable ordering, while Figure 2 shows the ROBDD with a different variable ordering $O' = x_1, x_3, x_5, x_7, x_9, x_2, x_4, x_6, x_8, x_{10}$. We can notice that the variable ordering in Figure 1 guarantees a linear ROBDD, while the ROBDD in Figure 2 has an exponential behavior.

More precisely, the ROBDD in Figure 1 contains three nodes for each subset $\{x_{2i-1}, x_{2i}\}$ (with $1 \leq i \leq 5$) of the partition $P$. Intuitively, this is due to the fact that each non-canonical variable (i.e., $x_{2i}$) must have a complemented valued with respect to the corresponding canonical variable (i.e., $x_{2i-1}$). Therefore, if the variables in the same subset of $P$ are all near each other in the ordering, the ROBDD returns 0 if the non-canonical variables have the same value of the canonical one, and it evaluates the next subset of variables, otherwise.

On the contrary if we put the variables, contained in the same subset of $P$, very distant in the ordering, the ROBDD increases in size. This is due to the fact that the value of each $x_{2i}$ depends on the value of $x_{2i-1}$. Thus, if we put the variables $x_{2i}$ and $x_{2i-1}$ distant in the ROBDD ordering, in each path from the root, we have to keep trace of the value of the first variable till we reach the node containing the other. For this reason, the top part of the ROBDD in Figure 2 is a complete binary tree composed by all the variables with odd indices. This tree has an exponential size.

The former considerations imply the following theorem (a complete proof will be given in the full version of the paper).

*Theorem 1:* Let $C$ be a 2-CEX expression on the Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$, and let $P = \{p_1, p_2, \ldots, p_m\}$ the partition of $X$ with respect to $C$. The corresponding ordering $O$ is an optimal ordering for the ROBDD of $C$ ($B_C$), which contains $|B_C| = 2 + \Sigma_{i=1}^{m}(2|p_i| - 1)$ nodes, where $|p_i|$ is the number of variables in the subset $p_i$ $(1 \leq i \leq m)$.

For example, let us consider the 2-CEX expression $(x_1 \oplus x_2)(x_1 \oplus x_4)(x_3 \oplus x_5)(x_1 \oplus x_6)x_7$. The corresponding partition of $\{x_1, x_2, \ldots, x_7\}$ is $P = \{\{x_7\}, \{x_1, x_2, x_4, x_6\}, \{x_3, x_5\}\}$. An optimal ordering is then $O = x_7, x_1, x_2, x_4, x_6, x_3, x_5$, as shown in Figure 3. In this example, consider the subset $\{x_1, x_2, x_4, x_6\}$. Once the value of $x_1$ (i.e., 0 or 1) is fixed, all

TABLE I
ROBDD SIZE GAP AND SYNTHESIS TIMES FOR EXPONENTIAL AND OUR OPTIMAL VARIABLE ORDERING ALGORITHMS. THE NUMBER OF CONSIDERED VARIABLES IS ONLY UP TO 8, DUE TO THE HIGH COMPUTATIONAL COST OF THE EXPONENTIAL ALGORITHM.

| | Number of nodes | | Computational time (s) | |
|---|---|---|---|---|
| **2-CEX** | **Minimum** | **Maximum** | **Exponential** | **Our algorithm** |
| $(x_1 \oplus x_5)(x_1 \oplus x_4)(x_0 \oplus x_7)(x_1 \oplus x_3)(x_0 \oplus x_6)(x_1 \oplus x_2)$ | 16 | 27 | 258.71146 | 0.00471 |
| $(x_0 \oplus x_7)(x_2 \oplus x_6)(x_1 \oplus x_5)(x_2 \oplus x_3)(x_0 \oplus x_4)$ | 15 | 39 | 233.03524 | 0.00463 |
| $(x_0 \oplus x_1)(x_2 \oplus x_3)(x_4 \oplus x_5)(x_6 \oplus x_7)$ | 14 | 47 | 223.15385 | 0.00451 |
| $(x_2 \oplus x_7)x_4(x_0 \oplus x_5)(x_1 \oplus x_6)x_3$ | 13 | 39 | 183.21888 | 0.00402 |
| $(x_0 \oplus x_3)x_6x_2x_7x_5x_1x_4$ | 11 | 17 | 145.53098 | 0.00116 |
| $(x_1 \oplus x_4)(x_0 \oplus x_5)(x_0 \oplus x_7)(x_1 \oplus x_6)(x_1 \oplus x_3)$ | 14 | 23 | 18.12630 | 0.00156 |
| $(x_0 \oplus x_7)(x_2 \oplus x_6)(x_2 \oplus x_4)x_3(x_2 \oplus x_5)$ | 13 | 23 | 15.82440 | 0.00246 |
| $x_6(x_1 \oplus x_5)(x_3 \oplus x_7)(x_2 \oplus x_4)$ | 12 | 31 | 14.27452 | 0.00258 |
| $x_5(x_1 \oplus x_7)x_3(x_2 \oplus x_6)(x_2 \oplus x_4)$ | 12 | 23 | 13.59286 | 0.00266 |
| $(x_2 \oplus x_5)x_7(x_2 \oplus x_6)(x_3 \oplus x_4)$ | 11 | 19 | 1.27687 | 0.00167 |
| $(x_1 \oplus x_3)(x_1 \oplus x_7)x_5(x_1 \oplus x_6)x_4$ | 11 | 13 | 1.24278 | 0.00406 |
| $(x_0 \oplus x_5)x_6x_7x_4(x_0 \oplus x_3)$ | 10 | 13 | 1.08048 | 0.00163 |
| $x_5(x_2 \oplus x_7)(x_1 \oplus x_6)$ | 9 | 15 | 0.12989 | 0.00108 |
| $x_5x_6x_7(x_2 \oplus x_4)$ | 8 | 11 | 0.11211 | 0.00103 |
| $x_6x_3x_5x_4x_7$ | 7 | 7 | 0.10327 | 0.00106 |
| $x_6(x_3 \oplus x_7)x_5$ | 7 | 9 | 0.02324 | 0.00073 |
| **Total average** (on the entire set of 1000 2-CEXs tested) | 11 | 18 | 68.21228 | 0.00251 |

the other variables in the subset must have the complemented value (i.e., 1 or 0, respectively). Figure 3 shows the two chains with root $x_1$. One is pointed by the 0-edge of $x_1$ and it is composed by the nodes containing the variables $x_2$, $x_4$, and $x_6$, which point to 0 with their 0-edge because $x_2$, $x_4$, and $x_6$ must have value 1 (since $x_1$ has value 0). The second chain is pointed by the 1-edge of $x_1$ and it is analogously composed by the nodes containing the variables $x_2$, $x_4$, and $x_6$, pointing to 0 with their 1-edge. Notice that the unique requirement for the optimality is that the variables in the same subset of the partition are next to each other in the ordering. Therefore, we can have several optimal ordering, each one corresponding to a different ROBDD with the same minimum number of nodes. In our example, $O' = x_7, x_3, x_5, x_1, x_2, x_4, x_6$ and $O'' = x_7, x_6,\ x_1, x_2, x_4, x_3, x_5$ are optimal. The ROBDDs contain $2 + (2 * 1 - 1) + (2 * 4 - 1) + (2 * 2 - 1) = 13$ nodes.

The ROBDD with the optimal ordering, described by Theorem 1, contains a number of nodes that is linear in the number of variables, as stated by the following corollary.

*Corollary 1:* Let $C$ be a 2-CEX expression on the Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$, and let $P = \{p_1, p_2, \ldots, p_m\}$ be the partition of $X$ induced by $C$. The corresponding optimal ordering $O$ contains a number of nodes $|B_C| \leq 2n + 2$ (i.e., $|B_C| \in O(n)$).

We finally mention that optimal variable ordering does not depend on variable negations (the proof will be given in the extended version of the paper).

## IV. EXPERIMENTAL RESULTS

In this section we report the experimental results related to the algorithm for deriving a minimal variable ordering for ROBDDs of 2-CEXs. The experiments have been run on a MacOS Apple M1 Chip with 16 GB of main memory. The Optimal Variable Ordering is computed with *Python* with *BDD Interface* of *PBL Interface* [22].

The theoretical results show that we can derive an optimal variable ordering, starting from a 2-CEX expression of an affine space, in linear time. The aim of our experiments is then twofold: 1) first, we are interested in understanding if the ROBDD with optimal variable ordering is much more compact than the one with worst variable ordering; 2) second, we are interested in evaluating the gain in computational time with respect to an exhaustive algorithm that computes the number of nodes of the ROBDD, for any variable ordering.

In order to conduct these experiments, we implement a naive strategy that generates all the possible variable orderings, which are exponential in number, and computes the corresponding ROBDDs. This exhaustive strategy is obviously exponential and cannot handle 2-CEX expressions with more than 8 distinct variables. Therefore, we concentrate our first set of experiments on this subset of 2-CEXs. We then compare the exponential algorithm with our linear algorithm described in Section III-A. Obviously, the linear algorithm can handle 2-CEXs with an extremely higher number of variables (as shown in the following), but for these 2-CEXs we would not be able to make a comparison with respect to the exhaustive algorithm.

In order to compare our strategy with the exhaustive one, we randomly generate $1000$ 2-CEX Boolean expressions with up to $8$ variables. These 2-CEXs are then used as inputs to the exponential algorithm where the minimum and maximum number of nodes for each 2-CEX are computed. The same 2-CEXs are then used as input for our algorithm, which calculates the minimum number of nodes, in order to compare the execution times. The ROBDDs build using our strategy are always composed by the minimum number of nodes (validating the theoretical result in Section III-A).

Table I reports a subset of the experimental results showing the gap in size of the best and worst ordering for 2-CEX expressions. The first column reports the input 2-CEXs, the second (resp., third) column shows the minimum (resp., maximum) number of nodes. The last two columns show the execution times, reported in seconds, for the exponential and for our linear

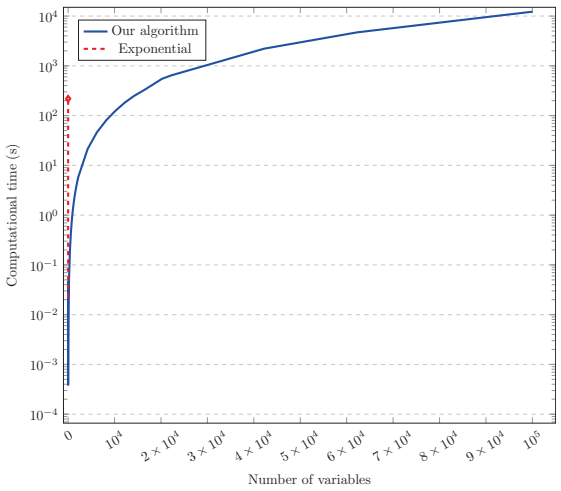


Fig. 4. Execution times for computing the optimal variable ordering for the 2-CEX $f = (x_1 \oplus x_2)(x_3 \oplus x_4)\dots(x_{2n-1} \oplus x_{2n})$ up to $10^5$ variables (in logarithmic scale).

algorithm. Finally, the last row of the table reports the average values for all the 1000 2-CEXs in the experiment.

We observe that the difference in execution time between the exhaustive and the proposed linear algorithm is up to 5 orders of magnitude. We can also note that several 2-CEXs show a high difference between minimum and maximum number of nodes (e.g., $(x_0 \oplus x_1)(x_2 \oplus x_3)(x_4 \oplus x_5)(x_6 \oplus x_7)$ as discussed in Section III-A), while others have always the same number of nodes in any variable ordering (e.g., $x_6x_3x_5x_4x_7$). In average, for our random experiments, the worst variable ordering increases the ROBDD dimension of about 64%. Moreover, the proposed method provides a speedup of about 27,000x with respect to the exhaustive approach.

In order to test the scalability of our approach, we finally considered the specific 2-CEX: $f = (x_1 \oplus x_2)(x_3 \oplus x_4)\dots(x_{2n-1} \oplus x_{2n})$. We computed this 2-CEX expression for increasing number of variables. Figure 4 reports the results up to $10^5$ variables in logarithmic scale. The blue solid line represents the computational time (in seconds) of the proposed algorithm and the red dashed line reports the times for the exhaustive approach. Note that the exhaustive algorithm can handle up to 8 variables. This set of experiments shows that the proposed method has a high scalability, thanks to the linear complexity of the optimization algorithm.

## V. Conclusion

In this paper we have proposed a linear time algorithm for computing an optimal variable ordering for the ROBDD representations of Boolean functions representing affine spaces whose characteristic function only involves XOR factors of at most two literals. As formally proved, and experimentally evaluated, this ordering always guarantees a ROBDD of size linear in the number of input variables.

Future work includes the analysis of the implicit representation of general affine subspaces, starting from those described by functions involving XOR factors of at most $k$ literals, for $k \geq 3$. In this regard, our first impression is that the complexity of the problem of finding the best variable ordering may increase already going from 2-XOR to 3-XOR factors.

## References


[1] A. Bernasconi, V. Ciriani, R. Drechsler, and T. Villa, "Logic Minimization and Testability of 2-SPP Networks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1190–1202, 2008.

[2] A. Bernasconi, S. Cimato, V. Ciriani, and M. C. Molteni, "Multiplicative complexity of autosymmetric functions: Theory and applications to security," in *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*. IEEE, 2020.

[3] A. Bernasconi and V. Ciriani, "Dimension-reducible boolean functions based on affine spaces," *ACM Trans. Design Autom. Electr. Syst.*, vol. 16, no. 2, p. 13, 2011.

[4] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Synthesis of autosymmetric functions in a new three-level form," *Theory Comput. Syst.*, vol. 42, no. 4, pp. 450–464, 2008.

[5] B. Bollig and I. Wegener, "Improving the variable ordering of obdds is np-complete," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 993–1002, 1996.

[6] R. Bryant, "Graph Based Algorithm for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. 35, no. 9, pp. 667–691, 1986.

[7] ——, "Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, 1992.

[8] R. E. Bryant, "Binary decision diagrams," in *Handbook of Model Checking*, E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds. Springer, 2018, pp. 191–217.

[9] D. Buchfuhrer, "The complexity of SPP formula minimization," in *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, ser. Lecture Notes in Computer Science, S. Hong, H. Nagamochi, and T. Fukunaga, Eds., vol. 5369. Springer, 2008, pp. 580–591.

[10] V. Ciriani, "Synthesis of SPP Three-Level Logic Networks using Affine Spaces," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1310–1323, 2003.

[11] P. Cohn, *Algebra Vol. 1*. John Wiley & Sons, 1981.

[12] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. MIT Press, 2009.

[13] R. Drechsler and B. Becker, *Binary Decision Diagrams - Theory and Implementation*. Springer, 1998.

[14] R. Ebendt, G. Fey, and R. Drechsler, *Advanced BDD optimization*. Springer, 2005.

[15] S. Fröhlich, S. Shirinzadeh, and R. Drechsler, "Multiply-accumulate enhanced bdd-based logic synthesis on RRAM crossbars," in *IEEE International Symposium on Circuits and Systems, ISCAS 2020, Sevilla, Spain, October 10-21, 2020*. IEEE, 2020, pp. 1–5.

[16] D. Knuth, *The Art of Computer Programming: Sorting and Searching*, Second ed. Addison Wesley, 1998, vol. 3.

[17] R. Liebler, *Basic Matrix Algebra with Algorithms and Applications*. Chapman & Hall/CRC., 2003.

[18] F. Luccio and L. Pagli, "On a New Boolean Function with Applications," *IEEE Transactions on Computers*, vol. 48, no. 3, pp. 296–310, 1999.

[19] C. Meinel and T. Theobald, *Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Applications*. Springer, 1998.

[20] M. Sauerhoff, I. Wegener, and R. Werchner, "Optimal ordered binary decision diagrams for read-once formulas," *Discrete Applied Mathematics*, vol. 103, no. 1, pp. 237–258, 2000.

[21] C. Scholl, D. Möller, P. Molitor, and R. Drechsler, "Bdd minimization using symmetries," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 18, no. 2, pp. 81–100, 1999.

[22] T. Sorensen, "BDD interface," http://formal.cs.utah.edu:8080/pbl/BDD.php, 2014.

[23] L. Tran, A. Gronquist, M. A. Perkowski, and J. S. C. IV, "An improved factorization approach to reversible circuit synthesis based on exors of products of exors," in *46th IEEE International Symposium on Multiple-Valued Logic, ISMVL 2016, Sapporo, Japan, May 18-20, 2016*. IEEE Computer Society, 2016, pp. 37–43.

[24] L. Tran, B. Schaeffer, A. Gronquist, M. A. Perkowski, and P. Kerntopf, "Synthesis of reversible circuits based on exors of products of exors," *Trans. Comput. Sci.*, vol. 24, pp. 111–128, 2014.