A NEW APPROACH TO AND/OR/EXOR FACTORIZATION FOR REGULAR ARRAYS

Ning Song and Marek Perkowski,
Portland State University, Dept. of Electr. Engn., Portland, Oregon 97207,
Tel: 503-725-5411, Fax: 503-725-4882, *mperkows@ee.pdx.edu*

Abstract— The proposed factorization methods for regular arrays of two-input cells have several important advantages over the existing logic representations and methodologies: (1) The logic representation and design implementation are consistent. (2) The stages of logic synthesis and physical design are effectively merged into a single stage. (3) The structure of the mapping solution is a regular rectangle. (4) Since the connections are mainly between neighbor cells, the wire delay is reduced comparing to other design methods. (5) Since the structure is regular, the creation of the high-performance tools is significantly easier. (6) The methods can be applied to fine-grain FPGA design, standard cell, gate matrix layout and sub-micron technologies.

1. Introduction

A new methodology for designing *Logic Cell Arrays* (LCA) has been introduced in [1, 2] and extended in [7]. LCA is a rectangular array of AND, OR and EXOR cells with negated inputs and outputs, and vertical and horizontal buses, similar to Fine Grain Field Programmable Gate Arrays. Such array realizes an *Exclusive Sum of Complex Terms* (ESCT), where complex terms are cascade combinations of two-input AND, OR and EXOR gates with possible negated inputs, and assuming the same order of input variables in them. Thus, LCA is a generalization of an EXOR PLA.

The problem of LCA minimization becomes that of ESCT minimization. Our approach introduces a new logic representation for AND/OR/EXOR factorization of multi-level functions realized in a regular form. It was shown in [6] that an extended cube representation and efficient minimization algorithms can be used that generalize the ESOP minimization approach from [3, 5]. This paper is a follow-up to our previous papers, especially [6]. We present in more detail the theorems and methods of multi-level regular factorization to two-input cells.

2. Basic Definitions

Definition 1. A **complex term** is a string of literals connected by a set of Boolean operators, and no literal can appear in the string more than once. The operators in the complex term can be any combinations of AND, OR, XOR, NAND, NOR, XNOR (denoted by $\cdot$, $+$, $\oplus$, $\bar{\cdot}$, $\overline{+}$ and $\overline{\oplus}$ respectively). All operators have the same priority. All operations must be performed in a sequence from left to right.

Example 1. Each of the following rows represents a complex term: $(a\bar{b}) + c$, $(a + b)c$, $((a \oplus b) + \bar{c})$, $((c\bar{b}) + a) \oplus d$. Expression $((ab) + \bar{b})c)$ is not a complex term, because the variable b appears twice. Expression $a + (b\bar{c}) + d$ is not a complex term because the operations are not in a sequence from left to right. However, if the order of variables is changed to b, c, a, d, then $(b\bar{c}) + a + d$ becomes a complex term.

Definition 2. An **multi-level cube** (**m-cube** for short) is a cube notation to present complex terms. Given a complex term, each literal of the complex term as well as each operator of the complex term is represented by the vector:

$$c_1^0 c_1^1 ... c_1^{(p_1-1)} - d_2^0 d_2^1 ... d_2^{(m-1)} c_2^0 c_2^1 ... c_2^{(p_2-1)} - ... - d_n^0 d_n^1 ... d_n^{(m-1)} - c_n^0 c_n^1 ... c_n^{(p_n-1)}$$

where $c_i^0 c_i^1 ... c_i^{(p_i-1)}$ is a p_i bit vector representing an p_i-valued literal, and $d_i^0 d_i^1 ... d_i^{(m-1)}$ is a m bit vector representing an operator.

The operators can be encoded in many different ways. In this coding scheme, three types of operators are assumed to be used in a complex term: AND, OR and XOR. So two bits are needed to store each operator. One encoding scheme, which is referred to as *standard cording*, is: 00 - not used, 01 - AND, 10 - OR, 11 - XOR.

Example 2. Given a complex term $T = X_1^{\{001\}} \cdot X_2^{\{110\}} + X_3^{\{101\}} \oplus X_4^{\{011\}}$. By using the standard coding, the complex term is presented by the m-cube as follows.

X_1	$\cdot$	X_2	$+$	X_3	$\oplus$	X_4
001	01	110	10	101	11	011.

Definition 3. The **negation** of a complex term is assigned the value 1 if the complex term has the value 0; and is assigned the value 0 if the complex term has the value 1.

Definition 4. The operators in the complex terms are referred to as **literal operators**. The operators applied on a pair of complex terms are referred to as **term operators**.

All the term operators and literal operators are:

	AND	OR	XOR	NAND	NOR	XNOR
Term Term Operator	$\wedge$	$\vee$	$\uplus$	$\overline{\wedge}$	$\overline{\vee}$	$\overline{\uplus}$
Term Literal Operator	$\cdot$	$+$	$\oplus$	$\bar{\cdot}$	$\overline{+}$	$\overline{\oplus}$

Note, that each literal operator has its counterpart of term operator, and they have the same logic meaning. For instance, $\wedge = \cdot$. The difference is their priority. The literal operators are evaluated before the term operators.

Example 3. $a + b \cdot c \oplus d \vee \bar{a} \cdot \bar{b} + d \uplus \bar{a} \oplus c + d = [(((a + b) \cdot c) \oplus d) \vee ((\bar{a} \cdot \bar{b}) + d)] \uplus ((\bar{a} \oplus c) + d)$.

Definition 5. The **tail literal of a complex term** (**tail literal** for short), is the last literal of the complex term. The **tail operator of a complex term** (**tail operator** for short), is the last operator of the complex term. The tail literal and the tail operator together is referred to as the **tail of the complex term** (**tail** for short). The **head of a complex term**, (**head term** or **head** for short), are all the literals and operators of the complex term except its tail.

Example 4. $a \cdot b + c \oplus d \cdot \bar{e}$ is a complex term. The tail literal is e, the tail operator is AND. the tail is $\cdot \bar{e}$ and the head is $a \cdot b + c \oplus d$.

When a complex term has only one literal, *e.g.* a, the literal is referred to as the tail. In this case, the complex term has no head.

Definition 6. If the number of non-universal literals in a complex term is 1, the complex term is referred to as a **single literal complex term**.

Definition 7. A **subterm of a complex term** is defined as **(1)** the head of the complex term; or **(2)** the head of a subterm of the complex term.

Example 5. In Example 4, the following are subterms: $a \cdot b + c \oplus d$, $a \cdot b + c$, $a \cdot b$, a.

Please note that b is not a subterm of a complex term $a \cdot b + c$, because it is not the head of the complex term nor the head of any subterms of the complex term. In sequel, P, Q, or P_1, P_2 will be used to denote complex terms. P^n will be used to denote the complex term P that has n input variables. Some of the literals in P may be **universal**, denoted by I. Consequently P^i will be used to denote a subterm of the complex term P^n, $(i < n)$.

Example 6. Given a complex term $P = X_1^{S_1} \odot X_2^{S_2} \odot \cdots \odot X_n^{S_n}$. It can be presented by its subterms as

$$P = P^{n-1} \odot X_n^{S_n}$$
$$= P^{n-2} \odot X_{n-1}^{S_{n-1}} \odot X_n^{S_n}$$
$$\vdots$$
$$= P^1 \odot X_2^{S_2} \odot \cdots \odot X_n^{S_n}.$$

Since programmable inverters at each cell input are assumed to be available, not only $P(i-1) \odot \overline{X}_i^{S_i}$ can be implemented, but also $\overline{P(i-1)} \odot \overline{X}_i^{S_i}$. In other words, the head of the complex term can be negated or non-negated.

Example 7. $P = (a \cdot b) + \bar{c}$ is a complex term, $Q = \overline{a \cdot b} + \bar{c}$ is also a complex term.

Lemma 1. All the Boolean properties, defined on two complex terms, can be applied on the head and the tail of a complex term.

Definition 8. A **positive complex term** is a complex term without NAND, NOR and XNOR operators. An ESCT contains only positive terms are referred to as in **positive term form**.

By the above definition, no literal operators are negated in a positive complex term. Please note that the negations on input variables are allowed.

Example 8. $a \cdot \bar{b}$ is a positive complex term. $a \bar{\cdot} b$ is not a positive complex term, because there is a NAND operator in the term.

Definition 9. An **or-free complex term** is a complex term without OR and NOR operators. An ESCT that contains only or-free complex terms is called an **or-free form**.

In an or-free complex term, its literal operators may be any combinations of AND, NAND and XOR operators. XNOR operators will be normalized to XOR, as discussed in more detail in [4].

Example 9. $a \oplus b \bar{\cdot} c$ is an or-free complex term. $a + b \bar{\cdot} c$ is not a or-free complex term; it contains an OR.

3. The ESCT Minimization Problem.

The starting point to ESCT minimization is a minimized *Exclusive Sum of Products* (ESOP) expression. The main idea of the ESOP minimization, as implemented in programs EXORCISM-MV-2 and EXORCISM-MV-3, [3, 4], is to *link* (reshape) a pair of *product terms.* This is done using *exorlink operations* [3, 4]. The product terms, after reshaping, may be able to merge with other product terms. Thus the total number of product terms in an ESOP is minimized. The same idea is used in the ESCT minimization [6]. The key operation in the ESCT minimization is to link (reshape) a pair of *complex terms.* We found that the *complex term linking* could be done is a way similar to the product term (exor)linking. As shown in [4], any two product terms in an ESOP can be linked and can generate a number of resultant product terms. The number of resultant product terms depends on the *distance* of two product terms. We found that this property exists also in ESCTs. Since the linking rules for complex terms are much more complicated than the linking rules for product terms, all the different linking cases are discussed [6]. Several special cases are also discussed [6]. In section 4 a new cube operation, *m-link,* which is an extension of exorlink, is introduced. Just like for the exorlink, we proved that the m-link operation can be applied on any two complex terms in a ESCT, and the number of resultant complex terms depends on the distance of two complex terms [4]. Then in section 5 the distance of two complex terms is defined with a thorough analysis of all different linking cases. Section 6 presents briefly our experimental results on MCNC benchmarks. Section 7 concludes the paper.

A. Different Linking Cases

It was shown in [6] that linking of two complex terms can be done step-by-step linking the heads and tails of the complex terms and their subterms. In this section, linking of heads and tails of two complex terms is discussed. In general the two complex terms can be written in the form: $P_1 = P_1^{n-1} \odot_1 X_n^{S_n}$, $P_2 = P_2^{n-1} \odot_2 X_n^{R_n}$ where $\odot_1$ and $\odot_2$ are two *tail operators* in P_1 and P_2 respectively. There are four cases to be investigated for heads:

[1] $P_1^{n-1} = P_2^{n-1}$;
[2] $P_1^{n-1} = \overline{P_2^{n-1}}$;
[3] $P_1^{n-1} \neq P_2^{n-1}$, $P_1^{n-1} = I^{n-1}$;
[4] $P_1^{n-1} \neq P_2^{n-1}$, $P_1^{n-1} \neq I^{n-1}$, $P_2^{n-1} \neq I^{n-1}$;

Note that case 1 includes the case that both heads are universal. The case 2 is that one of the heads is the negation of the other. Case 3 is the case that one of the heads is universal. Since the commutative law holds, when either one of the heads is universal, belongs to this case. The last case is that the two heads are different and none of them is universal. Thus all the different cases are included in the above four cases.

Similarly, there are four different cases for tail literals:

[1] $X_n^{S_n} = X_n^{R_n}$;
[2] $X_n^{S_n} = \overline{X_n^{R_n}}$;
[3] $X_n^{S_n} \neq X_n^{R_n}$, $X_n^{R_n} = I$;
[4] $X_n^{S_n} \neq X_n^{R_n}$, $X_n^{S_n} \neq I$, $X_n^{R_n} \neq I$.

Please note that in case 3, one of the tail literals is universal. If the head is also in case 3, the universal head and the universal tail literal are in different complex terms. In other words, the case is either P_1^{n-1} and $X_n^{R_n}$ are universal, or P_2^{n-1} and $X_n^{S_n}$ are universal. If both P_1^{n-1} and $X_n^{S_n}$ are universal, or both P_2^{n-1} and $X_n^{R_n}$ are universal, the case should be taken care of at a previous stage by evaluating $P_1^n \odot X_{n+1}^{S_{n+1}}$ and $P_2^n \odot X_{n+1}^{R_{n+1}}$. So at the current stage, if both the head and the tail are in case 3, we only need to consider the case that the universal head and the universal tail are in different complex terms.

There are six cases of positive literal operators: **(1)** both operators are AND; **(2)** both operators are OR; **(3)** both operators are XOR; **(4)** one operator is AND and the other is OR; **(5)** one operator is AND and the other is XOR; **(6)** one operator is OR and the other is XOR. There are also the cases that the literal operators are **negative**. However, all the negative operators can be converted to the positive operators without increasing the number of complex terms. So only the positive operators are considered here. There are totally 4 (case on heads) $\times$ 4 (case on tail literals) $\times$ 6 (cases on tail operators) = 96 different cases.

In [4], the concept of **simplified complex terms** is introduced. In a simplified complex term, some of the above conditions would not occur. For instance, $P^{n-1} \oplus I$ is not a simplified complex term. Table 1 lists all the different linking cases. Since the simplified complex terms are assumed, the tail operators adjacent to universal literals are AND only, there are 23 cases which do not exist. So there are 93 - 23 = 73 cases listed in Table 1.

In Table 1, the top left block shows the cases that the two complex terms have the identical heads and identical tails. There are six cases in this block, which are all combinations of tail operators. In the first row, the second block from the left has three cases: **(1)** both operators are AND, **(2)** one operator is AND and the other is OR, **(3)** one operator is AND and the other is XOR. This block includes the cases that the two complex terms have the identical heads but different tail literals, and one of the tail literals is universal. According to the simplification rules from [4], if the tail literal is universal, the tail operator is AND. So only three cases instead of six cases are in this block.

Table 1: Linking Cases for the Positive Term Form

	$X_{i+1}^{S_{i+1}} = X_{i+1}^{R_{i+1}}$	$X_{i+1}^{S_{i+1}} = I^i$	$X_{i+1}^{S_{i+1}} = \overline{X_{i+1}^{R_{i+1}}}$	$X_{i+1}^{S_{i+1}} \neq X_{i+1}^{R_{i+1}}$
$P_1^i = P_2^i$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$	$(\cdot,\cdot)$ $(\cdot,+)$ $(\cdot,\oplus)$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$
$P_2^i = I$	$(\cdot,\cdot)$ $(\cdot,+)$ $(\cdot,\oplus)$	$(\cdot,\cdot)$	$(\cdot,\cdot)$ $(\cdot,+)$ $(\cdot,\oplus)$	$(\cdot,\cdot)$ $(\cdot,+)$ $(\cdot,\oplus)$
$P_1^i = \overline{P_1^i}$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$	$(\cdot,\cdot)$ $(\cdot,+)$ $(\cdot,\oplus)$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$
$P_1^i \neq P_2^i$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$	$(\cdot,\cdot)$ $(\cdot,+)$ $(\cdot,\oplus)$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$	$(\cdot,\cdot)$ $(+,+)$ $(\oplus,\oplus)$ $(\cdot,+)$ $(\cdot,\oplus)$ $(+,\oplus)$

The above discussion is based on the *positive term form.* Comparing with the *or-free form*, we found that the latter one has two advantages: **(1)** If the or-free form is used, the number of linking cases is less than in the positive term form. This is because the OR operators are eliminated from the ESCT. The combinations of literal operators are reduced. **(2)** $P_1^i = \overline{P_2^i}$ is one of the conditions to be checked before the linking. Checking this condition is more convenient by using the or-free form than the positive term form.

There are 40 cases in Table 2. This number is significantly smaller than in the positive form, which is 73. All these cases were proved in [4] and discussed also in [6].

Two Complex Terms with Identical Tails. We consider here the case that the tails of the two complex terms are the same, and the two heads are different. In this case, the tail operators can be AND, OR or XOR. The linking of such two complex terms are shown by the following three equations:

$$(P_1^{n-1} \cdot X_n^{S_n}) \uplus (P_2^{n-1} \cdot X_n^{S_n}) = (P_1^{n-1} \uplus P_2^{n-1}) \cdot X_n^{S_n} \quad (1)$$

$$(P_1^{n-1} + X_n^{S_n}) \uplus (P_2^{n-1} + X_n^{S_n}) = (P_1^{n-1} \uplus P_2^{n-1}) \cdot \overline{X_n^{S_n}} \quad (2)$$

$$(P_1^{n-1} \oplus X_n^{S_n}) \uplus (P_2^{n-1} \oplus X_n^{S_n}) = P_1^{n-1} \uplus P_2^{n-1} \quad (3)$$

Above three equations show that if the tails of two positive terms are the same, then the tail can be extracted out. To extract an identical tail, the following operations are performed: **(1)** if the tail operator is AND, the tail remains the same; **(2)** if the tail operator is OR, the tail is negated; **(3)** if the tail operator is XOR, the tail becomes an universal tail.

As proved in [4], the above three equations can be then applied on the subterms if the tails of the two subterms are also the same. Note that if the or-free form is used, then the OR operators have been converted to AND operators and Equation (2) above is not needed. However, converting to or-free form does not change the linking results, as shown below.

Example 10. Given are two complex terms $P_1 = P_1^{n-1} + X_n^{S_n}$ and $P_2 = P_2^{n-1} + X_n^{S_n}$. The two complex terms have identical tails. Applying Equation (2) on the two complex terms generates:

$P_1^{n-1} + X_n^{S_n} \uplus P_2^{n-1} + X_n^{S_n} = (P_1^{n-1} \uplus P_2^{n-1}) \cdot \overline{X_n^{S_n}}$.

If the two complex terms have been converted to or-free form before hand, then $P_1 = \overline{P_1^{n-1}} \cdot \overline{X_n^{S_n}}$ and $P_2 = \overline{P_2^{n-1}} \cdot \overline{X_n^{S_n}}$. Applying Equation (1): on the two complex terms generates:

$\overline{P_1^{n-1}} \cdot \overline{X_n^{S_n}} \uplus \overline{P_2^{n-1}} \cdot \overline{X_n^{S_n}} = (\overline{P_1^{n-1}} \uplus \overline{P_2^{n-1}}) \cdot \overline{X_n^{S_n}} = (P_1^{n-1} \uplus P_2^{n-1}) \cdot \overline{X_n^{S_n}}$.

For both the positive term form and or-free form, the results are the same.

Theorem 1. The identical trailing part of two complex terms does not increase the number of resultant complex terms in complex term linking.

Example below shows the Theorem's 1 application.

Example 11. $(P_1^{n-1} \oplus X_{n-2}^{S_{n-2}} + X_{n-1}^{S_{n-1}} \cdot X_n^{S_n}) \uplus (P_2^{n-1} \oplus X_{n-2}^{S_{n-2}} + X_{n-1}^{S_{n-1}} \cdot X_n^{S_n}) = [(P_1^{n-1} \oplus X_{n-2}^{S_{n-2}} + X_{n-1}^{S_{n-1}}) \uplus (P_2^{n-1} \oplus X_{n-2}^{S_{n-2}} + X_{n-1}^{S_{n-1}})] \cdot X_n^{S_n} = [(P_1^{n-1} \oplus X_{n-2}^{S_{n-2}}) \uplus (P_2^{n-1} \oplus X_{n-2}^{S_{n-2}}] \cdot \overline{X_{n-1}^{S_{n-1}}} \cdot X_n^{S_n} = (P_1^{n-1} \uplus P_2^{n-1}) \cdot \overline{X_{n-1}^{S_{n-1}}} \cdot X_n^{S_n}$.

In product term linking, the number of resultant cubes is determined by the number of different literals in the product terms. A pair of identical literals in the product terms do not increase the resultant cubes. Theorem 1 shows that the complex term linking has a similar property. The number of resultant complex terms is determined by the difference between the two complex terms, not the identical portion.

Two Complex Terms with Identical Heads and Identical Tail Operators. In these cases, the differences between the two complex terms are in the tail literals. There are three cases on tail operators; AND, OR and XOR, as in the following three equations.

(1) $(P^{n-1} \cdot X_n^{S_n}) \uplus (P^{n-1} \cdot X_n^{R_n}) = (P^{n-1} \wedge X_n^{S_n}) \uplus (P^{n-1} \wedge X_n^{R_n}) = P^{n-1} \cdot (X_n^{S_n} \uplus X_n^{R_n}) = P^{n-1} \cdot X_n^{S_n \uplus R_n}$.

(2) $(P^{n-1} + X_n^{S_n}) \uplus (P^{n-1} + X_n^{R_n}) = (\overline{P^{n-1}} \cdot \overline{X_n^{S_n}}) \uplus (\overline{P^{n-1}} \cdot \overline{X_n^{R_n}}) = \overline{P^{n-1}} \cdot (X_n^{S_n} \uplus X_n^{R_n}) = \overline{P^{n-1}} \cdot X_n^{S_n \uplus R_n}$.

(3) $(P^{n-1} \oplus X_n^{S_n}) \uplus (P^{n-1} \oplus X_n^{R_n}) = (P^{n-1} \uplus P^{n-1}) \oplus X_n^{S_n \uplus R_n}$.

Above three equations show that if both the heads and the tail operators are identical, the heads can be factored out.

Table 2: Linking Cases for the Or-Free Form

	$X_{i+1}^{S_{i+1}} = X_{i+1}^{R_{i+1}}$	$X_{i+1}^{S_{i+1}} = I^i$	$X_{i+1}^{S_{i+1}} = \overline{X_{i+1}^{R_{i+1}}}$	$X_{i+1}^{S_{i+1}} \neq X_{i+1}^{R_{i+1}}$
$P_1^i = P_2^i$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$
$P_2^i = I$	$(\cdot, \cdot)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$	$(\cdot, \cdot)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\cdot, \oplus)$
$P_1^i = \overline{P_1^i}$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$
$P_1^i \neq P_2^i$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$	$(\cdot, \cdot)$ $(\oplus, \oplus)$ $(\cdot, \oplus)$

Two Complex Terms with Identical Heads and Identical Tail Literals. Two complex terms differ by the literal operators. There are three such cases: AND to OR, AND to XOR and OR to XOR.

(1) $(P^{n-1} \cdot X_n^{S_n}) \uplus (P^{n-1} + X_n^{S_n}) = P^{n-1} \uplus X_n^{S_n}$

(2) $(P^{n-1} \cdot X_n^{S_n}) \uplus (P^{n-1} \oplus X_n^{S_n}) = P^{n-1} + X_n^{S_n}$

(3) $(P^{n-1} + X_n^{S_n}) \uplus (P^{n-1} \oplus X_n^{S_n}) = P^{n-1} \cdot X_n^{S_n}$

Two Complex Terms with different Tails. The heads of the two complex terms are the same, and both tail operators and tail literals are different.

(1) $(P^{n-1} \cdot X_n^{S_n}) \uplus (P^{n-1} + X_n^{R_n}) = (P^{n-1} \cdot X_n^{S_n}) \overline{\uplus}\, \overline{(P^{n-1} + X_n^{R_n})} = (P^{n-1} \cdot X_n^{S_n}) \overline{\uplus} (\overline{P^{n-1}} \cdot \overline{X_n^{R_n}}) = (P^{n-1} \cdot X_n^{S_n \uplus \overline{R_n}}) \overline{\uplus}\, \overline{X_n^{R_n}} = (P^{n-1} \cdot X_n^{S_n \uplus \overline{R_n}}) \uplus X_n^{R_n}$.

(2) $(P^{n-1} \cdot X_n^{S_n}) \uplus (P^{n-1} \oplus X_n^{R_n}) = (P^{n-1} \cdot X_n^{S_n}) \uplus P^{n-1} \uplus X_n^{R_n}$ $[P^{n-1} \cdot (X_n^{S_n} \uplus I)] \uplus X_n^{R_n}$ $(P_1^{n-1} \cdot \overline{X_n^{S_n}}) \uplus X_n^{R_n}$.

(3) $(P^{n-1} + X_n^{S_n}) \uplus (P^{n-1} \oplus X_n^{R_n}) = \overline{(P^{n-1} + X_n^{S_n})} \uplus \overline{P^{n-1}} \uplus X_n^{R_n} = (\overline{P^{n-1}} \cdot \overline{X_n^{S_n}}) \uplus \overline{P^{n-1}} \uplus X_n^{R_n} = [\overline{P^{n-1}} \cdot (\overline{X_n^{S_n}} \uplus I)] \uplus X_n^{R_n} = (\overline{P^{n-1}} \cdot X_n^{S_n}) \uplus X_n^{R_n}$.

Linking of heads and tails of a pair of complex terms is one step of the complex term linking. Based on this discussion, the multi-level cube linking will be now presented.

4. Multi-level Cube Linking

In Table 3 the top row lists the conditions of tail literals. The left column lists the conditions of heads. $(\cdot, \cdot)$ indicates both the tail operators are AND. There are 40 linking cases listed in the table. Most of linking cases have one linking rule to apply. In these cases, one group of complex terms is generated by the rule. For some linking cases, like case 17, there are two linking rules, 17-1 and 17-2, which generate two groups of resultant complex terms. There is one special case, case 40, which has four linking rules, thus generates four groups of resultant complex terms.

Given two complex terms in an ESCT, their identical tails, if any, are first extracted out. Then one of the linking rules is applied on the two complex terms (or subterms, if the identical tails are removed). The linking rules usually generate a single complex term plus a pair of complex terms with identical tails. The single complex term is saved for the final result. The complex term pair will be processed by the above procedure – remove identical tails and perform linking rules. In this procedure, which linking rule to apply depends on the type of heads, tail literals and tail operators. The following example is given to show this procedure.

Example 12. Given are two complex terms $P_1 = a \cdot b \cdot d$ and $P_2 = \bar{a} \cdot \bar{c} \cdot d$. Linking of P_1 and P_2 consists of the following steps.

[1.] Move out identical tail AND d
$(a \cdot b \cdot d) \uplus (\bar{a} \cdot \bar{c} \cdot d) = [(a \cdot b) \uplus (\bar{a} \cdot \bar{c})] \cdot d$.

[2.] The linking is on two subterms $a \cdot b$ and $\bar{a} \cdot \bar{c}$. In this case, the heads are different $(a \cdot b \neq \bar{a} \cdot I))$, one of the tail literals is I, and both the tail operators are AND. We can see that case 17 in Table 3 meets these conditions, so linking rule 17 in Table 3 could be applied, which generates two groups of resultant complex terms.

Rule 17-1:
$(a \cdot b) \uplus (\bar{a} \cdot \bar{c}) = [(a \cdot b) \uplus (\bar{a} \cdot I) \cdot \bar{c}] \uplus [a \cdot b \cdot \bar{c}]$.
The single subterm $a \cdot b \cdot \bar{c}$ plus its tail AND d is one of the resultant complex terms in group 1. The subterm pair $(a \cdot b) \uplus (\bar{a} \cdot I)$ will be carried over to the next linking step.

Rule 17-2:
$(a \cdot b) \uplus (\bar{a} \cdot \bar{c}) = [(a \cdot b) \uplus (\bar{a} \cdot I)] \uplus [\bar{a} \cdot \bar{c}]$.
The single subterm $\bar{a} \cdot \bar{c}$ plus the identical tail AND d is in the group 2.

[3.] Check identical tails on the pairs of subterms. For the first pair, $(a \cdot b \uplus \bar{a} \cdot I) \cdot \bar{c}$, the tail $\cdot\bar{c}$ is generated at the previous step, and has been moved out. The heads, $a \cdot b$ and $\bar{a} \cdot I$ have no identical tails. The second pair, $(a \cdot b \uplus \bar{a} \cdot I)$ is always the same as the heads of the first pair, and there are no identical tails either.

[4.] Apply linking rules on the subterm pairs. Since $P_1^1 = a$, $P_2^1 = \bar{a}$, $P_1 = \overline{P_2}$, one of the tail literals is universal and both tail operators are AND, linking rule 15 is applied, which generates one complex term: $a \cdot b \uplus \bar{a} \cdot I = \overline{\bar{a} \cdot \bar{b}}$

[5.] The final results are two groups of complex terms:
Group 1 = { $a \cdot b \cdot \bar{c} \cdot d$ (from step 2),
$\overline{\bar{a} \cdot \bar{b}} \cdot \bar{c} \cdot d$ (from step 4)}
and Group 2 = { $\bar{a} \cdot \bar{c} \cdot d$ (from step 2),
$\overline{\bar{a} \cdot \bar{b}} \cdot d$ (from step 4)}.

Above example showed that the whole procedure contains iterations of extracting the identical tail and applying the linking rules.

Next the linking operation is defined. The two complex terms are written in the following forms:

$$P_1^n = X_1^{S_1} \odot_1^2 X_2^{S_2} \odot_1^3 \cdots \odot_1^i X_i^{S_i} \cdots \odot_1^{i+1} X_{i+1}^{S_{i+1}} \odot_1^{i+2} \cdots \odot_1^n X_n^{S_n},$$
$$P_2^n = X_1^{R_1} \odot_2^2 X_2^{R_2} \odot_2^3 \cdots \odot_2^i X_i^{R_i} \cdots \odot_2^{i+1} X_{i+1}^{R_{i+1}} \odot_2^{i+2} \cdots \odot_2^n X_n^{R_n}.$$

In the equation, $\odot_i^j$ indicates the jth literal operator of the ith complex term. Each $\odot_i^j$ can be AND, OR or XOR.

Definition 10. Given two complex terms P_1^n and P_2^n in an ESCT, the tail, $Tail(P_1^n, P_2^n)$, is an m-cube operation defined by the following formula:
$Tail(P_1^n, P_2^n) =$

$$\begin{cases} Tail(P_1^{n-1}, P_2^{n-1}) \cdot X_n^{S_n} & \text{if } X_1^n = X_2^n \text{ and } \odot_1^n = \odot_2^n \text{ are AND} \\ Tail(P_1^{n-1}, P_2^{n-1}) \cdot \overline{X_n^{S_n}} & \text{if } X_1^n = X_2^n \text{ and } \odot_1^n = \odot_2^n \text{ are OR} \\ Tail(P_1^{n-1}, P_2^{n-1}) & \text{if } X_1^n = X_2^n \text{ and } \odot_1^n = \odot_2^n \text{ are XOR} \\ P_1^n, P_2^n & \text{if } X_1^n \neq X_2^n \text{ or } \odot_1^n \neq \odot_2^n. \end{cases}$$

The **Tail operation** can be applied on a pair of complex terms or a pair of subterms. If the tails of the two complex terms are different, then the the tail operation has no effect, and the outputs are the two input complex terms.

Example 13. Given two complex terms, $P_1 = a \cdot b + c \oplus d$, $P_2 = a \cdot b + c + d$ the $Tail(P_1, P_2) = P_1 \uplus P_2$.

Please note that the two complex terms P_1 and P_2 are in an ESCT, the term operator between P_1 and P_2 is XOR, $P_1 \uplus P_2$ remains the same before and after the **Tail operation**.

If the two complex terms have the same tail, the tail can be extracted out. Since the $Tail$ operation is defined recursively, if the last k literals and last k operators are all the same, in a pair of complex terms, then those k literals and operators can be extracted by the $Tail$ operation.

Example 14. Given two complex terms, $P_1 = a \cdot \bar{b} + c + d \oplus e \cdot f$, $P_2 = a \cdot b + c + d \oplus e \cdot f$
$Tail(a \cdot \bar{b} + c + d \oplus e \cdot f, a \cdot b + c + d \oplus e \cdot f) = Tail(a \cdot \bar{b} + c + d \oplus e, a \cdot b + c + d \oplus e) \cdot f = Tail(a \cdot \bar{b} + c + d, a \cdot b + c + d) \cdot f = Tail(a \cdot \bar{b} + c, a \cdot b + c) \cdot \bar{d} \cdot f = Tail(a \cdot \bar{b}, a \cdot b) \cdot c \cdot \bar{d} \cdot f = Tail(a \cdot \bar{b}, a \cdot b) \cdot c \cdot \bar{d} \cdot f = [(a \cdot \bar{b}) \uplus (a \cdot b)] \cdot c \cdot \bar{d} \cdot f$.

In the last step of the example, $a \cdot \bar{b}$ and $a \cdot b$ are two subterms. Since $b \neq \bar{b}$, the tails of the subterms are different, the $Tail$ operation stops. After the $Tail$ operation, the complex terms (or subterms) have different tails.

Definition 11. Given two complex terms P_1^n and P_2^n in an ESCT, the $Link(P_1^n, P_2^n)$ is an m-cube operation which applies one of the linking rules based on the conditions from Table 3.

Based on the above discussion, the main m-cube operation can be defined below:

Definition 12. The **multi-level cube linking (m-link** for short), denoted as $\otimes_{CT}$, is an m-cube operation on two complex (sub) terms defined by the following procedure:

1. $Tail(P_1^n, P_2^n)$,
2. $Link(P_1^n, P_2^n)$,
3. $Tail(P_1^i, P_2^i)$,
4. $Link(P_1^i, P_2^i)$,
5. repeat steps 3 and 4 until no more subterm pairs.

Compare this new operation to exorlink operation on two product terms. Section 3 listed 4 cases on literals. We found that in exorlink, only two cases for each pair of literals are considered, case 1 and case 4. In other words, we only need to check if a pair of literals are the same or they are different. Case 2 does not exist in exorlink because there are no negations on a subterm of a product. Now turning to operations on complex terms, case 3 is separated from case 4, because the universal subterm or the universal literal may reduce the distance of two complex

Table 3: The Linking Rules of Two Complex Terms

	$X_{i+1}^{S_{i+1}} = X_{i+1}^{R_{i+1}}$			$X_{i+1}^{S_{i+1}} = I^i$		
	case	tail opt	Rule	case	tail opt	Rule
$P_1^i = P_2^i$	1	$(\cdot, \cdot)$	1	12	$(\cdot, \cdot)$	12
	2	$(\oplus, \oplus)$	2			
	3	$(\cdot, \oplus)$	3	13	$(\cdot, \oplus)$	13
$P_2^i = I$	4	$(\cdot, \cdot)$	4	14	$(\cdot, \cdot)$	14
	5	$(\oplus, \cdot)$	5			
$P_1^i = \overline{P_1^i}$	6	$(\cdot, \cdot)$	6	15	$(\cdot, \cdot)$	15
	7	$(\oplus, \oplus)$	7			
	8	$(\cdot, \oplus)$	8	16	$(\cdot, \oplus)$	16
$P_1^i \neq P_2^i$	9	$(\cdot, \cdot)$	9	17	$(\cdot, \cdot)$	17-1 and 17-2
	10	$(\oplus, \oplus)$	10			
	11	$(\cdot, \oplus)$	11	18	$(\cdot, \oplus)$	18

	$X_{i+1}^{S_{i+1}} = \overline{X_{i+1}^{R_{i+1}}}$			$X_{i+1}^{S_{i+1}} \neq X_{i+1}^{R_{i+1}}$		
	case	tail opt	Rule	case	tail opt	Rule
$P_1^i = P_2^i$	19	$(\cdot, \cdot)$	19	30	$(\cdot, \cdot)$	30
	20	$(\oplus, \oplus)$	20	31	$(\oplus, \oplus)$	31
	21	$(\cdot, \oplus)$	21	32	$(\cdot, \oplus)$	32
$P_2^i = I$	22	$(\cdot, \cdot)$	22	33	$(\cdot, \cdot)$	33-1 and 33-2
	23	$(\oplus, \cdot)$	23	34	$(\oplus, \cdot)$	34
$P_1^i = \overline{P_1^i}$	24	$(\cdot, \cdot)$)	24	35	$(\cdot, \cdot)$	35-1 and 35-2
	25	$(\oplus, \oplus)$	25	36	$(\oplus, \oplus)$	36
	26	$(\cdot, \oplus)$	26	37	$(\cdot, \oplus)$	37
$P_1^i \neq P_2^i$	27	$(\cdot, \cdot)$	27-1 and and 27-2	38	$(\cdot, \cdot)$	38-1 and 38-2
	28	$(\oplus, \oplus)$	28	39	$(\oplus, \oplus)$	39
	29	$(\cdot, \oplus)$	29	40	$(\cdot, \oplus)$	40-1, 40-2, 40-3, and 40-4

terms. While in exorlink, the universal literals do not reduce the distance. Also, the concept of subterm is not used in exorlink, because the variables are ordered independently in product terms. While in complex terms, the concept of subterm is introduced mainly for the purpose of protecting the variable orders. There is one more difference between the complex term linking and the product term linking: in product terms, the operators are AND only, so the operators of the two corresponding literals are always equal. On the other hand, in complex term, the operators may be different.

5. Distance of Two Complex Terms

Like the case of exorlink [3], the **distance of two complex terms** is a measurement of the differences of these terms. The distance of two complex terms is related to the number of resultant complex terms to be generated by linking, which is important information for the ESCT optimization. In product term linking, the distance checking is simple: comparing each pair of literals in the two product terms. In complex terms linking, the situation is much more complicated. Counting the number of different literals or the number of different operators is not enough to determine the distance. It has to be taken into account different types of heads, tail literals and tail operators.

Definition 13. The **distance**, denoted as $\delta_X(n)$, of two complex terms is a measurement of the differences of these terms in an ESCT. Here n means that each of the complex terms has n literals. The distances are determined in Table 4.

Table 4: The Distance of Two Complex Terms

	$X_{i+1}^{S_{i+1}} = X_{i+1}^{R_{i+1}}$			$X_{i+1}^{S_{i+1}} = I^i$		
	c	tail opt	$\delta_X(n)$	c	tail opt	$\delta_X(n)$
$P_1^i = P_2^i$	1	$(\cdot, \cdot)$	0	12	$(\cdot, \cdot)$	1
	2	$(\oplus, \oplus)$	0			
	3	$(\cdot, \oplus)$	1	13	$(\cdot, \oplus)$	1
$P_2^i = I$	4	$(\cdot, \cdot)$	1	14	$(\cdot, \cdot)$	1
	5	$(\oplus, \cdot)$	1			
$P_1^i = \overline{P_1^i}$	6	$(\cdot, \cdot)$	1	15	$(\cdot, \cdot)$	1
	7	$(\oplus, \oplus)$	1			
	8	$(\cdot, \oplus)$	1	16	$(\cdot, \oplus)$	1
$P_1^i \neq P_2^i$	9	$(\cdot, \cdot)$	$\delta_X(n-1)$	17	$(\cdot, \cdot)$	$\delta_X(n-1)+1$
	10	$(\oplus, \oplus)$	$\delta_X(n-1)$			
	11	$(\cdot, \oplus)$	2	18	$(\cdot, \oplus)$	$\delta_X(n-1)$

	$X_{i+1}^{S_{i+1}} = \overline{X_{i+1}^{R_{i+1}}}$			$X_{i+1}^{S_{i+1}} \neq X_{i+1}^{R_{i+1}}$		
	c	tail opt	$\delta_X(n)$	c	tail opt	$\delta_X(n)$
$P_1^i = P_2^i$	19	$(\cdot, \cdot)$	1	30	$(\cdot, \cdot)$	1
	20	$(\oplus, \oplus)$	1	31	$(\oplus, \oplus)$	1
	21	$(\cdot, \oplus)$	1	32	$(\cdot, \oplus)$	2
$P_2^i = I$	22	$(\cdot, \cdot)$	1	33	$(\cdot, \cdot)$	2
	23	$(\oplus, \cdot)$	1	34	$(\oplus, \cdot)$	1
$P_1^i = \overline{P_1^i}$	24	$(\cdot, \cdot)$	1	35	$(\cdot, \cdot)$	2
	25	$(\oplus, \oplus)$	0	36	$(\oplus, \oplus)$	1
	26	$(\cdot, \oplus)$	1	37	$(\cdot, \oplus)$	2
$P_1^i \neq P_2^i$	27	$(\cdot, \cdot)$	$\delta_X(n-1)+1$	38	$(\cdot, \cdot)$	$\delta_X(n-1)+1$
	28	$(\oplus, \oplus)$	2	39	$(\oplus, \oplus)$	$\delta_X(n-1)$
	29	$(\cdot, \oplus)$	2	40	$(\cdot, \oplus)$	2

Table 4 is similar to Table 3, except the column "Rule" is replaced by "$\delta_X(n)$" (distance). Column's caption c denotes a case. On the column of distance, there are either a constant value 0, 1 or 2, or a formula $\delta_X(n-1)$, or $\delta_X(n-1)+1$. There are three cases in Table 4, case 1, case 2 and case 25, in which the distance are 0. The conditions to apply cases 1, 2 and 25 have been discussed in the previous section. As shown in [4], in these three cases, the number of resultant complex terms is 0. There are 22 cases in the table which have distance value 1. For instance, in case 3, the heads and tail literals are identical, the tail operators are AND and XOR respectively, the two complex terms can be merged into one complex term, as shown in [4]. The distance in this case is 1. There are 8 cases, in which the distance is 2. Among these cases, some of the cases generate one group of two resultant complex terms. These cases are case 11, case 28, case 29, case 32 and case 37. In all these cases, at least one of the tail operators is XOR. Some other cases are similar to the distance 2 exorlink. They generate two groups of resultant complex terms. Each group has two complex terms. These cases include case 33 and case 35. There is a special case, case 40, which generates 4 groups of resultant complex terms. The smallest group is the group 4, which has two complex terms. In Table 4, the distance value for case 40 is 2, which is the smallest value for this case.

Definition 14. The **terminated** cases for checking the distance of complex terms are those cases in Table 4, in which the distance values are constant. The **non-terminated** cases are those in which the distance values are not constant.

Those cases discussed above, which have distance values of 0, 1 or 2, are terminated cases. During the process of checking the distance of two complex terms, if any of the terminated cases is encountered, the distance of the two complex terms is determined. And the checking process stops. For the non-terminated cases, the distance cannot be determined at the current step, the process continues to check the subterms, until a terminated case is encountered. Among the non-terminated cases, the distance values for case 9, case 10, case 18 and case 39 are $\delta_X(n-1)$, which means at current step, the distance value remains the same. For the remaining cases, case 17, case 27 and case 38, the distance values are $\delta_X(n-1)+1$, which means the distance value at the current step is increased by 1.

Example 15. Given are two complex terms; $P_1 = x_1 \cdot \bar{x}_2 + x_3 \cdot x_4 \oplus x_5$, $P_2 = \bar{x}_1 + x_2 \cdot \bar{x}_3 \cdot \bar{x}_4 \cdot x_6$. The distance checking procedure is shown below.

(1) The heads are: $P_1^5 = x_1 \cdot \bar{x}_2 + x_3 \cdot x_4 \oplus x_5$, $P_2^5 = \bar{x}_1 + x_2 \cdot \bar{x}_3 \cdot \bar{x}_4$. $P_1^5 \neq P_2^5$, the heads are different. The tail literals are I and x_6. Both the tail operators are AND. This is case 17 (see Table 4). The distance value is $\delta_X(5)+1$. The distance at this step is increased by 1 (the initial value of distance is 0). This is a non-terminated case. $\delta_X(5)$ is checked at next step.

(2) $P_1^4 = x_1 \cdot \bar{x}_2 + x_3 \cdot x_4$, $P_2^4 = \bar{x}_1 + x_2 \cdot \bar{x}_3 \cdot \bar{x}_4$. $P_1^4 \neq P_2^4$. The tails are XOR x_5 and AND I. This is case 18. The distance value is $\delta_X(4)$. The distance at this step remains the same. This is a non-terminated case. $\delta_X(4)$ is checked at next step.

(3) $P_1^3 = x_1 \cdot \bar{x}_2 + x_3$; $P_2^3 = \bar{x}_1 + x_2 \cdot \bar{x}_3$. Please note that $P_1^3 = \overline{P_2^3}$. The tails are $\cdot x_4$ and $\cdot \bar{x}_4$. This is case 24. The distance value is 1. This is a terminated case. The distance value at the previous steps is 1 and at this step is also 1. The final distance value is 2.

The above example shows that the distance of the two given complex terms is 2. Thus two resultant complex terms will be generated by linking them. Below, examples of distance 0, 1, 2 and 3 m-links are presented.

A. Distance 0 M-Link

If the distance of two complex terms is 0, no resultant term is generated. The two complex terms are removed. For both the exorlink and m-link, if the two product terms or the two complex terms are equal, their distance is 0. There is one more case of distance 0 for m-link, the case 25 in Table 4.

Example 16. Given are two complex terms; $P_1 = a \cdot \bar{b} + c \oplus d$, $P_2 = \bar{a} + b \cdot \bar{c} \oplus \bar{d}$. In this example, head $P_1^3 = a \cdot \bar{b} + c$ and head $P_2^3 = \bar{a} + b \cdot \bar{c}$. Since $\overline{P_2^3} = \overline{\bar{a} + b \cdot \bar{c}} = a \cdot \bar{b} + c$, $P_1^3 = \overline{P_2^3}$. According to linking rule 25, [4], $P_1 \uplus P_2 = (a \cdot \bar{b} + c \oplus d) \uplus (\bar{a} + b \cdot \bar{c} \oplus \bar{d}) = 0$. No resultant complex term is generated in this example.

B. Distance 1 M-Link

Distance 1 m-link of two complex terms generates one resultant complex term. In other words, the two complex terms can be merged to one complex term. There are 22 such cases in Table 4. One of these cases is shown in the following example.

Example 17. Given are two complex terms: $P_1 = a \cdot \bar{b} + c \cdot d$, $P_2 = \bar{a} + b \cdot \bar{c} \cdot \bar{d}$. This example is similar to the previous example. The only difference from the previous one is that the tail operators are AND instead of XOR. By checking Table 4, this example is in case 24, the distance is 1. By performing linking rule 24, $(P_1^{n-1} \cdot X_n^{S_n}) \uplus (P_2^{n-1} \cdot X_n^{R_n}) = P_2^{n-1} \oplus X_n^{S_n}$, the result is: $P_1 \uplus P_2 = \bar{a} + b \cdot \bar{c} \oplus d$.

C. Distance 2 M-Link

There are two situations in which the distance of two complex terms is 2: a distance 2 terminated case, or a non-terminated case followed by a terminated case.

Example 18. Given are two complex terms: $P_1 = x_1 \cdot x_2 \oplus x_3 \cdot x_5 \cdot x_6$, $P_2 = \bar{x}_1 \oplus \bar{x}_2 \cdot \bar{x}_4 \oplus x_6$. From the product term linking point of view, these two complex terms are different by nearly all the literals and operators. Actually, only the tail literals are the same in these two complex terms. However, according to Table 4, this is case 11, which is a terminated case and the distance of the two complex terms is 2. According to Rule 11: $(P_1^{n-1} \cdot X_n^{S_n}) \uplus (P_2^{n-1} \oplus X_n^{R_n}) = \overline{P_1^{n-1}} \cdot X_n^{S_n} \uplus P_2^{n-1}$, the result is: $P_1 \uplus P_2 = (x_1 \cdot x_2 \oplus \bar{x}_3 + \bar{x}_5 \cdot x_6) \uplus (\bar{x}_1 \oplus \bar{x}_2 \cdot \bar{x}_4)$.

Example 19. Given are two complex terms: $P_1 = x_1 \cdot x_2 \oplus x_3 \oplus \bar{x}_4 \cdot x_5$, $P_2 = \bar{x}_1 + \bar{x}_2 \oplus \bar{x}_3 \cdot \bar{x}_4 \oplus x_6$. In this example, the two heads are different, one of the tail literals is universal, the other tail operator is XOR. By checking Table 4, this is case 18. The corresponding linking rule is: $(P_1^{n-1} \cdot X_n^{S_n}) \uplus (P_2^{n-1} \oplus X_n^{R_n}) = (P_1^{n-1} \uplus P_2^{n-1}) \uplus X_n^{R_n}$. At this step, the distance does not increase. We continue to check the distance of the two subterms: $P_1 = x_1 \cdot x_2 \oplus x_3 \oplus \bar{x}_4 \cdot x_5$, $P_2 = \bar{x}_1 + \bar{x}_2 \oplus \bar{x}_3 \cdot \bar{x}_4$. At this step, case 17 is encountered. From Table 4, case 17 increases the distance by 1, and we continue to check the subterms $P_1 = x_1 \cdot x_2 \oplus x_3 \oplus \bar{x}_4$. $P_2 = \bar{x}_1 + \bar{x}_2 \oplus \bar{x}_3 \cdot \bar{x}_4$. Note that at this step, $P_1^3 = \overline{P_2^3}$. Now a terminated case is encountered, case 8, of which the distance is 1. The distance of the two complex terms P_1 and P_2 is 2.

D. Distance 3 M-Link

Example 20. Given are two complex terms: $P_1 = x_1 \cdot x_2 \oplus x_3 \oplus \bar{x}_4 \cdot x_5$, $P_2 = \bar{x}_1 \oplus \bar{x}_2 \cdot \bar{x}_4 \oplus x_6$. In this example, the first two steps are the same as the previous example. The accumulated value of distance is 1, and we continue to check the subterms: $P_1 = x_1 \cdot x_2 \oplus x_3 \oplus \bar{x}_4$, $P_2 = \bar{x}_1 \oplus \bar{x}_2 \cdot \bar{x}_4$. At this step, case 11 is encountered. This is a terminated case and its value of distance is 2. The distance of the two complex terms P_1 and P_2 is 3.

6. Experimental Results

Look ahead strategy is applied in ESCT optimization program MINICT [6]. It starts from an ESOP minimized by EXORCISM-MV, and its operation is followed by output folding. The time of folding is negligible and is included in the total times given in the table. MINICT it-

	In	Out	C_{T0}	C_{in}	C_T	C_{CO}	C	C/C_{in}	Time
xor5	5	1	5	30	1	1	6	0.2	0.01
con1	7	2	9	81	8	1	64	0.79	0.02
5xp1	7	10	32	544	8	7	112	0.21	0.10
adr4	8	5	31	403	16	1	144	0.36	0.06
rd53	5	3	14	112	9	1	54	0.48	0.03
rd73	7	4	35	385	25	1	200	0.52	0.28
wgt8	8	4	56	672	36	2	360	0.54	0.34
rd84	8	4	59	708	46	1	414	0.58	0.48
squar5	5	8	19	247	17	5	170	0.69	0.05
bw	5	28	22	726	22	19	528	0.73	4.06
mlp4	8	8	60	960	57	5	741	0.77	2.82

Table 5: *Some results of MINICT*

erates M-link operations of distance 3 and next distance 2, executing distance 1 and distance 0 operations whenever possible. It is very similar to ESOP minimization strategy from [5]. Therefore, because of this general similarity, our method can be easily adapted to **any** rule-based approach to ESOP minimization, for instance the simulated annealing approach from [8]. The goal of LCA synthesis is to *minimize the total area*, which is measured by the number of rows multiplied by the number of columns. The number of rows corresponding to the number of complex terms. The number of columns consists of two parts. The input part corresponding to the number of input variables, which is fixed. The output part corresponding to the output columns, which can be minimized. The goal of ESCT optimization is to *minimize the number of complex terms* in the ESCT expressions. Some results of MINICT starting from minimized ESOPs [5] are shown in Table 5. The inputs of the program are minimized ESOPs. The outputs are ESCTs with minimized number of complex terms and minimized number of output columns. In Table 5, the columns In and Out are number of input variables and number of output variables, respectively. C_{T0} is the number of product terms in the ESOPs. C_{in} is the initial cost. According to the cost function defined in Section 5 the initial cost; $C_{in} = C_{T0} \times (In + Out)$. In Table 5, the column C_T is the number of complex terms after complex terms minimization. The column C_{CO} is the number of output columns after column folding [1,2,4]. Column C is the cost of minimized ESCTs. $C = C_T \times (In + C_{CO})$. Column C/C_{in} gives the ratio of the costs after the minimization and before the minimization. *Time* is cpu seconds on a Pentium 133 MHz PC. For instance, the first test case, 5xp1, has 7 inputs, 10 outputs, and 32 product terms. The initial cost is $32 \times (7 + 10) = 544$. It has been minimized to 8 complex terms and 7 output columns. The cost is $8 \times (7 + 7) = 112$. The cost ratio is 112 / 544 = 0.21.

7. Conclusions

We proposed a new representation and based on it factorization algorithms for AND/OR/EXOR circuits. Our theory can find applications for both ASICs and FPGAs. Based on complex term theorems [4, 6], we developed here minimization algorithms for logic cell array optimization. The advantages of our approach are the following: **(1)** Multi-level factorization method was given that applies uniformly to AND, OR and EXOR gates. **(2)** The factorization method addresses concurrently the issues of space, delay times and regularity. **(3)** The approach can be used not only to the kind of regularity as addressed by MINICT, but also for a more general tree factorization. **(4)** It can be also extended to SCT factorization that starts from SOP rather than ESOP form. **(5)** All presented ESCT methods have their counterparts in corresponding new SCT methods. **(6)** A mixed array is possible in which some output functions can be realized by SCT and some by ESCT, which will further improve the results. **(7)** If both SCT minimizations and ESCT minimizations are performed and the better results between the two are selected, the overall results could be improved. **(8)** Mixed OR and EXOR term operators can be applied in the collecting plane of LCA, thus creating MSCT, a mixed sum of complex terms [2]. MSCT minimization problem combines those of ESCT and SCT minimization.

References

[1] A. Sarabi, N. Song, M. Chrzanowska-Jeske and M. Perkowski, "A Comprehensive Approach to Logic synthesis and Physical Design for two-dimensional Logic Arrays," *Proc. 31th DAC*, pp. 321-326, June, 1994.

[2] N. Song, M. Perkowski, M. Chrzanowska-Jeske and A. Sarabi, "A New Design Methodology for Two-Dimensional Logic Arrays," *VLSI Design*, 1995, Vol. 3, Nos. 3-4, pp. 315-332.

[3] N. Song and M. Perkowski, "Minimization of Exclusive Sum of Products Expressions for Multiple-Valued Input Incompletely Specified Functions," *IEEE Trans. on CAD*, Vol. 15, no. 4, pp. 385-395, 1996.

[4] N. Song, "A New Design Methodology for Two-Dimensional Logic Cell Arrays," *Ph.D. Thesis*, Nov. 14, 1997, PSU.

[5] N. Song, M. Perkowski, "A New Fast Approach to Approximate ESOP Minimization for Incompletely Specified Multi-Output Functions," *Proc. RM'97*, pp. 61 - 72.

[6] N. Song, M. Perkowski, "Minimization of Exclusive Sums of Multi-Valued Complex Terms for Logic Cell Arrays," *Proc. ISMVL'98*, Fukuoka, Japan, May 1998.

[7] G. Lee, "Logic Synthesis for Cellular Architecture FPGAs using BDDs," *In ASP-DAC'97*, pp. 253 - 258, January 1997.

[8] L. Parilla, J. Ortega, and A. Lloris, "Using Simulated Annealing in the Minimisation of AND-EXOR Functions," *Electronic Letters*, Vol. 30, No. 22, pp. 1838 - 1839, October 1994.