Minimization of reversible wave cascades - PRELIMINARY VERSION

Dimitrios Voudouris(dvoudour@cslab.ece.ntua.gr)
Stergios Stergiou(stergiou@cslab.ece.ntua.gr)
George Papakonstantinou(papakon@cslab.ece.ntua.gr)

Abstract

In this paper two algorithms for the synthesis and minimization of a CA *(cellular array architecture)* are proposed. Starting from a completely specified single-output switching function, our methods produce rectangularly shaped arrays of cells, interconnected in chains, with an effort to minimize the number of the produced chains (cascades). This kind of cellular topology is known throughout the bibliography as Maitra cellular arrays. The significance of those algorithms is underlined by the fact that this particular type of cellular architecture can be mapped to reversible circuits and gates *(generalized Toffoli gates)*, which are the type of logic used in quantum circuits. The proposed methodologies include use of ETDDs *(EXOR ternary decision diagrams)*, and switching function decompositions (including new types of boolean expansions).

1 Introduction

A circuit is called reversible if it has the same number of inputs and outputs and maps each input vector into a unique output vector and vice versa. One of their important properties is that they consume minimal amount of power due to the fact that they lose no information [16]. It is expected that they will become more attractive for future technologies. All quantum logic gates are also reversible [11]. Systematic logic synthesis algorithms for reversible logic are still immature, although some methods have been presented in the related literature [3], [12]. In Ref [3] the $k*k$ *Generalized family of Toffoli* gates were introduced, based on Maitra cascades [1], to form a special type of cellular arrays, called reversible wave cascades (or Maitra cellular arrays). This special architecture is one of the simplest forms of cellular arrays because it requires very simple cells, with limited interconnection between them(Fig. 1). The cells are connected in chains, usually called Maitra cascades or terms or even complex terms, (upper plane) which in turn are linked together by the collector row (lower plane), to form the wave cascades array. In this paper, the collector row implements a XOR function using as inputs the outputs of the cascades. Every Maitra cell can realize a two-variable switching function. One of the inputs is the corresponding variable and the other is the output of the previous (in physical order) cell in the chain. It will be shown in section 2 that a Maitra cellular array architecture forms a reversible logic circuit.

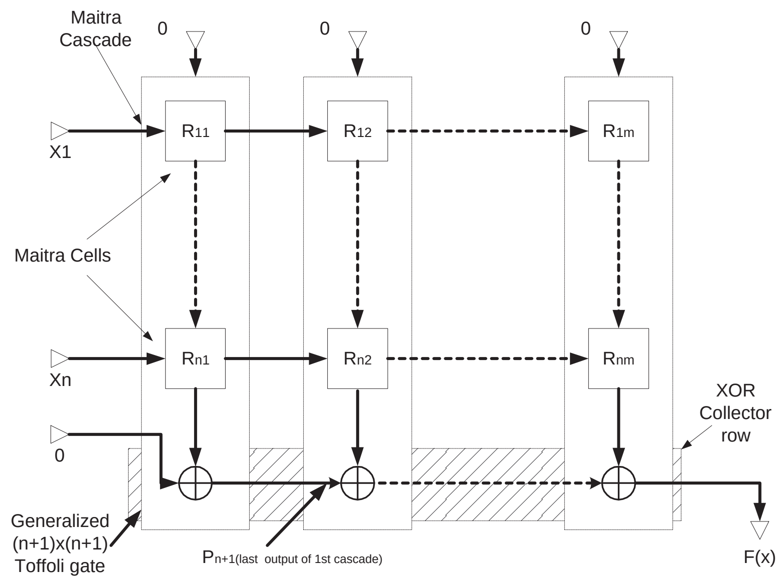


Figure 1: Reversible wave cascade CA

Synthesis of cellular arrays and cascades and especially of Maitra cellular arrays has been a main scientific interest in the sixties and seventies, although due to technical difficulties, no corresponding tools or hardware architectures have been created [5]. Recently, with the introduction of CA type architectures and more specifically LUT-based and CA-type FPGAs, a growing need for specialized tools that can map arbitrary switching functions to cellular array architectures has emerged ([4], [5]).

Although algorithms for mapping switching functions to such architectures have been developed in the past, the minimization of a wave cascade (the special form of a switching function which can be directly mapped to a Maitra cellular array) and especially those that use XOR collector rows is still an open problem.

In Ref [4] cube calculus operations, variable reordering (using also the EXORLINK operation) and technology folding of input and output planes of the array are used in order to minimize the number of product terms of a given switching function (a product term is a special case of a Maitra term). The final minimized expressions are composed of product terms (not complex terms). In Ref [5] the mapping of incompletely-specified, multi-output switching functions is performed using the concept of KFDD (Kronecker function decision diagrams) and three heuristic algorithms are given. In Ref [6], TDDs (ternary decision diagrams) along with variable reordering technics are used, to produce cellular cascades. The expressions produced from the above algorithm, are not composed of complex terms (as defined above), since the output of a cell in a chain can be connected not only to its physically next, but to others as well. Moreover, the usage of SHANNON and DAVIO decompositions at each level of the TDD produces product terms. In Ref [7] a similar method is introduced that uses TDDs, but applies for incompletely specified functions as well. The resulting CA architecture may contain, except the usual XOR collector row, an OR collector row as well. In Ref [8] an extension of the theory of Maitra cascades is proposed, by extending the number of possible inputs and outputs of each cell, using multi-value logic. Folding technics are also used. All the above methods utilize architectures that are different from the wave cascade architecture which is used in this paper. In Ref [14], [15] a systematic method is proposed for minimizing expressions composed of Maitra terms, using an extension of the EXORLINK operation. The architecture which was proposed, though, is different from the one used in this paper. In Ref [3] another algorithm is presented for mapping Maitra cellular arrays to reversible gates (more specifically generalized Toffoli gates). Nevertheless their algorithm has not been implemented and they only give upper bounds on the number of stages in terms, based on methods from ESOP minimization. Finally, none of the above algorithms guarantees minimality.

2 Theoretical background

In this section we provide some background definitions. An expression of a switching function, suitable for mapping to a Maitra cascade (cell chain), is called a Maitra term. A more formal definition [3] follows:

Definition 1. A complex Maitra term (complex term for simplicity) is recursively defined as follows:

1. Constant 0 (1) Boolean function is a Maitra term.
2. A literal is a Maitra term.
3. If M_i is a Maitra term, a is a literal, and G is an arbitrary two-variable Boolean function, then $M_{i+1} = G(a, M_i)$ is a Maitra term.

Additionally, it is required that each variable appears in each Maitra term only once.

Definition 2. A reversible wave cascade expression for a single-output switching function is an exlusive-OR sum of complex terms:

$$f = \sum_{i=1}^{m} \oplus M_i,$$

where m is the number of complex terms. Moreover, the same variable ordering is used for every M_i.

Definition 3. A minimal (or exact) expression of a switching function $f(x_1, \dots, x_n)$ of n variables is defined as the wave cascade expression which has the least number of terms comparing to every other wave cascade expression for this function.

Definition 4. The weight $w(f)$ of a switching function $f(x_1, \dots, x_n)$ is defined as the number of complex terms in a minimal expression of f.

Definition 5. A switching function is called cascade realizable, if it has weight 1.

It can be proved [1] that every two-variable switching function is cascade realizable.

A wave cascade expression can be directly mapped to a reversible wave cascade cellular architecture(Fig. 1).It consists of two-input, one-output cells *(Maitra cells)* which, normally, implement every two-variable switching function, i.e. sixteen functions. Many cells are linked together to form chains *(cascades)*. The cascades of a wave cascade expression constitute the upper-input plane of the cellular architecture and are linked together through the collector row, i.e. a horizontal array of cells implementing the XOR function (lower-collecting plane). The number of cells in every cascade is equal to the number of variables of the function. The first input of every Maitra cell is supplied by one of the variables and the second one is the output of the previous (in physical order) cell in the cascade.

Definition 6. A $k * k$ generalized Toffoli gate is defined [3] as: $P_1 = A_1, P_2 = A_2, \ldots, P_{n-1} = A_{n-1}, P_n = f_{n-1}(A_1, A_2, \ldots, A_{n-1}) \oplus A_n$, where A_i are the inputs of the gate, P_i are the outputs of the gate and f_{n-1} is an arbitrary switching function of $n-1$ variables.

In Ref [3] it was proved that a $k * k$ generalized Toffoli gate is reversible. A Maitra cascade, composed of n cells, plus the corresponding XOR collector cell, is a $(n+1) * (n+1)$ Toffoli gate (Fig. 1) where: $P_1 = A_1 = X_1, \ldots, P_n = A_n = X_n, A_{n+1} = 0$ (or the output of another Toffoli gate) $, P_{n+1} = f_n(A_1, A_2, \ldots, A_n) \oplus A_{n+1}$ and $f_n(A_1, A_2, \ldots, A_n) = G_n(G_{n-1}(\ldots G_1(0, X_1), X_{n-1}), X_n)$ ($G_i, i \leq n$ are arbitrary boolean functions defined in Def. 1). It follows that a Maitra cellular array is a reversible logic circuit because it is composed of reversible gates.

It has been proved [2] that a Maitra cell doesn't need to implement every two-variable switching function. A set of only six functions is sufficient(complete set). A function which is cascade realizable using all 16 two-variable functions, is also cascade realizable using a complete set of 6 two-variable functions. Of course there are many equivalent such sets [13]. In the rest of this paper, we will use one of them (Table 1). The cascades that use cells which implement any switching function from the complete set are called *Restricted Maitra Cascades* and lead to smaller implementations, since only three bits per Maitra cell are required instead of four. From this point on, without loss of generality, when we mention Maitra cascades, we will be referring to restricted Maitra cascades.

Table 1: Cell index set

Cell index(r)	$F_r(x, y)$
1	$x + y$
2	$\overline{x} + y$
3	$\overline{x}y$
4	xy
5	$x \oplus y$
6	y

It is assumed that the first cell in every cascade has one of its inputs connected to 0 (for symmetry reasons). In the following lemma we prove that the first cell of a chain needs to be of index 1, 2 and 6 only.

Lemma 1. *The set {1, 2, 6} of indices for the first cell of a complex term is complete.*
Proof. One of the inputs of the first cell of every complex term is the constant 0. So the possible values for the output(F_r, r the index of the cell) of the first cell are:

- $F_1 = x_1$
- $F_2 = \bar{x_1}$
- $F_3 = 0$
- $F_4 = 0$
- $F_5 = x_1$
- $F_6 = 0$
  So the three possible values are $x, \bar{x}, 0$ *and in order to produce them, we may use indices 1, 2, 6.*

In the following section we will define new switching function decompositions.

3 Switching Function Decompositions

Definition 7. Let $f(\mathbf{x})$ be a switching function and $\mathbf{x}$ the vector of its variables. Let x_1 be one of the variables in the vector $\mathbf{x}$. Then $f(x_1 = 0, x2, \ldots)$, $f(x_1 = 1, x2, \ldots)$ and $\{f(x_1 = 0, x2, \ldots) \oplus f(x_1 = 1, x2, \ldots)\}$ are subfunctions of f, regarding variable x_1. For simplicity, in the rest of this paper, we will refer to $f(x_1 = 1, x2, \ldots)$ as f_1, to $f(x_1 = 0, x2, \ldots)$ as f_0, to $\{f(x_1 = 0, x2, \ldots) \oplus f(x_1 = 1, x2, \ldots)\}$ as f_2 and to x_1 as x.

Given a Boolean function $f(\mathbf{x})$, where $\mathbf{x}$ is the vector of the function's variables, and a variable x in this vector, we can express f as:

- $f(\mathbf{x}) = \bar{x} f_0 \oplus x f_1$
- $f(\mathbf{x}) = x f_2 \oplus f_0$
- $f(\mathbf{x}) = \bar{x} f_2 \oplus f_1$

The first of those rules is known as the *SHANNON expansion* and the rest as the *DAVIO expansions*(positive Davio and negative Davio respectively). Those previous expressions (also known as switching function expansions), decompose a function to a XOR-sum of its subfunctions' expressions and have been extensively used in ESOP minimization.

Before the introduction of the new switching decompositions, two lemmas, will be presented, for merging Maitra cells.

Lemma 2 (Two cell merging)**.** *The relation*
$F_{r_1}(x, y_1) \oplus F_{r_2}(x, y_2) = F_r(x, y_1 \oplus y_2)$*,* $y_1 \neq y_2$ *and* $y_1 \neq \bar{y_2}$
is true iff:
$(r_1, r_2, r) = (1, 1, 3), (1, 3, 1), (2, 2, 4), (2, 4, 2), (3, 3, 3),$
$(4, 4, 4), (5, 5, 6), (5, 6, 5), (6, 6, 6)$
Proof. The above lemma can easily be proved exhaustively.

The next lemma can easily be derived from the previous.

Lemma 3 (N cell merging)**.** *The relation*
$\sum \oplus F_{r_i}(x, y_i) = F_r(x, \sum \oplus y_i)$*,* $y_i \neq y_k, y_i \neq \bar{y_k}, \forall k \neq i$
is true according to Table 2:

Table 2: Cell merging

Merging N cells with index r_i	r
Odd number with index 1 and any with index 3	1
Odd number with index 2 and any with index 4	2
Even number with index 1 and any with index 3	3
Even number with index 2 and any with index 4	4
Odd number with index 5 and any with index 6	5
Even number with index 5 and any with index 6	6

Proof. If we consider the first case of Table 2, then according to lemma 2, in order to create a cell of index 1, we must merge one cell of index 1 and one of index 3. A cell of index 3 can be created by merging any number of index 3 cells or even number of index 1 cells. Therefore, we need odd number of index 1 cells and any number of index 3 cells to produce a cell of index 1. Similar arguments can be used for the other cases of the table.

Theorem 1 (New Decompositions)**.** *Given a Boolean function* $f(\boldsymbol{x})$*, where* $\boldsymbol{x}$ *is the vector of the function's variables, and a variable* x *in this vector, we can express* f *as:*

$$f(\boldsymbol{x}) = (x + f_2) \oplus (x \oplus f_1) \tag{1}$$

$$f(\boldsymbol{x}) = (x + f_0) \oplus (x\bar{f}_1) \tag{2}$$

$$f(\boldsymbol{x}) = (x\bar{f}_2) \oplus (x \oplus f_0) \tag{3}$$

$$f(\boldsymbol{x}) = (x + \bar{f}_0) \oplus (\bar{x} + \bar{f}_1) \tag{4}$$

$$f(\boldsymbol{x}) = (\bar{x} + \bar{f}_2) \oplus \bar{f}_0 \tag{5}$$

$$f(\boldsymbol{x}) = (x + \bar{f}_2) \oplus \bar{f}_1 \tag{6}$$

$$f(\boldsymbol{x}) = (\bar{x}\bar{f}_2) \oplus (x \oplus \bar{f}_1) \tag{7}$$

$$f(\boldsymbol{x}) = (\bar{x}\bar{f}_0) \oplus (\bar{x} + f_1) \tag{8}$$

$$f(\boldsymbol{x}) = (\bar{x} + f_2) \oplus (x \oplus \bar{f}_0) \tag{9}$$

Proof. *We will prove that the above rules are equivalent to the SHANNON expansion.*

For (1) it holds: $f = [x + f_2] \oplus [x \oplus f_1] = \overline{[\bar{x}\bar{f}_2]} \oplus [x \oplus f_1] = [\bar{x}(f_1 \oplus \bar{f}_0)] \oplus \overline{[x \oplus f_1]} = \bar{x}\bar{f}_0 \oplus \bar{x}f_1 \oplus \bar{x} \oplus f_1 = \bar{x}f_0 \oplus xf_1$

For (2) it holds: $f = (x + f_0) \oplus (x\bar{f_1}) = \bar{x}\bar{f_0} \oplus x\bar{f_1} \oplus 1 = \bar{x}\bar{f_0} \oplus x\bar{f_1} \oplus (x \oplus \bar{x}) = \bar{x}f_0 \oplus xf_1$

For (3) it holds: $f = \{x\bar{f_2}\} \oplus [x \oplus f_0] = x(f_0 \oplus \bar{f_1}) \oplus x \oplus f_0 = \bar{x}f_0 \oplus xf_1$

Cases 4, 5 and 6 are the Shannon, and Davio expansions when we complement each term. Cases 7, 8, 9 are derived from cases 1, 2 and 3, respectively, if we complement each term.

Theorem 2. *Let f be a switching function and f_0, f_1, f_2 its subfunctions, expressed in the form of XOR-sum of complex terms. The application of the above rules ((1), (2), (3), (4), (5), (6), (7), (8), (9)) creates expressions for f which are also in the form of XOR-sum of complex terms.*
Proof. In the above decompositions every term of the XOR-sum is of the form:

$G(x, f_i)$, *where* $G(x, f_i) = \begin{cases} x + f_i \\ \bar{x} + f_i \\ xf_i \\ \bar{x}f_i \\ x \oplus f_i \\ f_i \end{cases}$, *and* f_i *is* f_0, f_1, f_2.

The above operations form the Maitra cell set of Table 1, therefore every such form of G can be implemented using one Maitra cell. Hence, if the expressions of two subfunctions f_i to be used, in one of the rules, have only one complex term, then the expression of f would consist of the two complex terms of its subfunctions, each one with one more cell of index 1, 2, 3, 4, 5 or 6 respectively. If the expressions of the subfunctions have more complex terms, then Lemma 3 will be used to determine what kind of cells to be added.

Theorem 2 indicates that the expansions, which were proposed in Theorem 1, produce expressions in the form of a XOR-sum of complex terms.

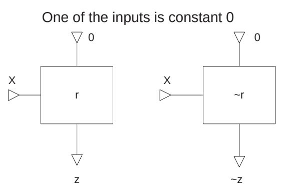
Theorem 3 (Complement function)**.** *The complement function of a complex term is also a complex term.*
Proof. We will use induction. The theorem obviously holds for cascades of one cell. If it holds for cascades of n cells, then it will be shown that it holds for n+1 cells. If the input (output of n cells) is the cascaded constant 0, then, for n+1 cells, our rule gives:

Original function	***Applying the rule***
$F_1 = x$	$F_2 = \bar{x}$
$F_2 = \bar{x}$	$F_1 = F_5 = x$
$F_5 = x$	$F_2 = \bar{x}$

In any other case, the input of the (n+1)th cell of the cascade will be complemented (when compared to the original cascade - see also the respective figure). By applying the rule we get the following results:

Original function	***Applying the rule***
$F_1 = x + y$	$F_3 = \bar{x}\bar{y} = \overline{x + y}$
$F_2 = \bar{x} + y$	$F_4 = x\bar{y} = \overline{\bar{x} + y}$
$F_3 = \bar{x}y$	$F_1 = \overline{x + \bar{y}} = \overline{\bar{x}y}$
$F_4 = xy$	$F_2 = \overline{\bar{x} + \bar{y}} = \overline{xy}$
$F_5 = x \oplus y$	$F_5 = x \oplus \bar{y} = \overline{x \oplus y}$
$F_6 = y$	$F_6 = \bar{y}$

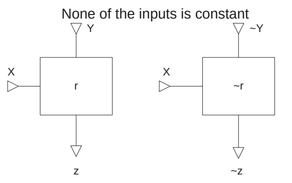
Corollary 1. *A switching function f and it's complement function $\bar{f}$ have the same weight.*



One of the inputs is constant 0

Original cell	Complement cell
1	2
2	1 or 5
5	2

Index 6 applies the rules to the next cell.



None of the inputs is constant

Original cell	Complement cell
1	3
2	4
3	1
4	2
5	5
6	6

Figure 2: Complement form

Theorem 3 can be used to compute the complemented forms presented in Theorem 1.

Theorem 4. *If $F_r(x, y^1)$, $F_q(x, y^2)$, and $F_g(x, y^3)$ are three cells, with $y^1, y^2, y^3 = y, \bar{y}$, then the equation:*

$$F_r(x, y^1) \oplus F_q(x, y^2) = F_g(x, y^3)$$

holds according to Table 3.
Proof. It can easily be proved exhaustively.

The following theorems introduce the concept of normalized forms and reversible wave cascade minimization.

Theorem 5. *Each minimal expression of a switching function f can always be written in one of the following normalized forms:*

$$f = F_p(x_1, y) \tag{10}$$

Table 3: Term Merging

$y^1{=}y$	$y^2{=}y$	$y^3{=}y$
r	q	g
1	4	5
1	5	4
3	4	6
3	6	4
4	5	1
4	6	3

$y^1{=}y$	$y^2{=}y$	$y^3{=}\bar{y}$
r	q	g
1	2	6
1	6	4
2	3	5
2	5	1
2	6	3
3	5	4

$y^1{=}y$	$y^2{=}\bar{y}$	$y^3{=}y$
r	q	g
1	4	6
1	6	2
2	1	5
2	3	6
2	5	3
2	6	1

$y^1{=}y$	$y^2{=}\bar{y}$	$y^3{=}y$
r	q	g
3	4	5
3	5	2
5	1	2
5	4	3
6	3	2
6	4	1

with $(p, y) = (1, f_0), (2, f_1), (3, f_0), (4, f_1), (5, f_0), (6, f_0)$
OR

$$f = F_p(x_1, y) \oplus F_q(x_1, z) \tag{11}$$

with $(p, q, y, z) = (3, 4, f_0, f_1), (3, 6, f_2, f_1), (4, 6, f_2, f_0),$
$(1, 5, f_2, f_1), (1, 4, f_0, \overline{f_1}), (4, 5, \overline{f_2}, f_0)$*, considering also that forms* $F_p(x, y) \oplus F_q(x, y)$ *and* $\overline{F_p(x, y)} \oplus \overline{F_q(x, y)}$ *are equivalent.*
OR

$$f = F_p(x_1, y) \oplus F_q(x_1, z) \oplus F_r(x_1, g) \tag{12}$$

with $p = 3, q = 4, r = 6$ *and* $y \oplus z = f_2$*,* $y \oplus g = f_0$*,* $z \oplus g = f_1$*, considering that forms* $F_p(x, y) \oplus F_q(x, y) \oplus F_r(x, y)$ *and* $\overline{F_p(x, y)} \oplus \overline{F_q(x, y)} \oplus F_r(x, y)$*,* $\overline{F_p(x, y)} \oplus F_q(x, y) \oplus \overline{F_r(x, y)}$*,* $F_p(x, y) \oplus \overline{F_q(x, y)} \oplus \overline{F_r(x, y)}$ *are equivalent. Forms* $F_3(x, y) \oplus F_4(x, y) \oplus F_6(x, y)$ *and* $F_3(x, y) \oplus F_4(x, \bar{y}) \oplus F_5(x, y)$ *are also equivalent.*

For the proof refer to Ref [10].

Theorem 6. *At least one minimal expression of a switching function* $f(x_1, \ldots, x_n)$ *with less than 6 variables* ($n < 6$) *or any number of variables and weight* < 6 *can be obtained from the minimal expressions of* f_0, f_1, f_2*.*
For the proof refer to Ref [10].

After proposing the concept of normalized forms, every decomposition, presented before, can be expressed as:

$$f = F_{cell^1}(x, f^1) \oplus F_{cell^2}(x, f^2) \tag{13}$$

Cell indexes cell^1 and cell^2 will be called *normalized cell indexes* and $F_{cell^1}(x, f^1)$, $F_{cell^1}(x, f^1)$ *normalized terms.* Table 4 presents the normalized form of all switching decompositions proposed in this paper.

Table 4: Decompositions and Normalized cell indexes

Decomposition	f^1	Cell^1	f^2	Cell^2
Shannon	f_0	3	f_1	4
P Davio	f_2	4	f_0	6
N Davio	f_2	3	f_1	6
New (1)	f_2	1	f_1	5
New (2)	f_0	1	$\bar{f}_1$	4
New (3)	$\bar{f}_2$	4	f_0	5
New (4)	$\bar{f}_0$	1	$\bar{f}_1$	2
New (5)	$\bar{f}_2$	2	$\bar{f}_0$	6
New (6)	$\bar{f}_2$	1	$\bar{f}_1$	6
New (7)	$\bar{f}_2$	3	$\bar{f}_1$	5
New (8)	$\bar{f}_0$	3	f_1	2
New (9)	f_2	2	$\bar{f}_0$	5

4 The heuristic algorithms

Based on the previous theorems and lemmas, we present two heuristic algorithms that produce reversible wave cascade expressions, for single-output switching functions, and furthermore they minimize the number of complex terms in them.

4.1 Approach 1

The first algorithm receives as input a single-output switching function in minterm formulation and decomposes it using ETDDs (EXOR Ternary Decision Diagrams, a DD where the third branch is the XOR-sum of the other two). Every function in the ETDD is decomposed, using the standard Shannon and Davio, along with the new expansions, presented in Theorem 1. During the composition (the reverse procedure of decomposition) of an expression by its subfunctions' forms, equal or complemented complex terms are merged. The final minimal expression of f will be produced by the minimal expressions of its subfunctions. The pseudocode of the algorithm (*Min1*) is presented in Alg. 1.

4.2 Approach 2

This second algorithm uses Min1 as a cube transformation and minimization technique. It creates groups of complex terms, in an iterative way, and then applies Min1 to each of them.

The pseudocode of the algorithm (*Min2*) is presented in Alg. 2.

References

[1] K.K. Maitra "Cascaded switching networks of two-input flexible cells" IRE Trans. Electron. Comput., pp, 136-143, 1962.

[2] R. C. Minnick, "Cutpoint cellular logic" IEEE Trans. Electron. Comput., vol. EC-13. pp. 685-698, Dec, 1964.

Algorithm 1: Min1(f: Switching function in minterm formulation) returns (Min Expr of f)

```
begin
    ETDD=Generate_ETDD($f$);
    Expr($f$)=Minimize(ETDD);
    // If $f$ is a 2-variable function then it is cascade realizable and the
    corresponding complex term is obtained from a Look-up table.
    // From the ternary tree, find the subfunctions of $f$.
    // If the expressions of the subfunctions of $f$ are not already found,
    apply this algorithm to each subfunction recursively.
    // If a subfunction is constant then produce the expressions of $f$ from
    a non-constant subfunction. The number of terms in a minimized ex-
    pression of $f$ equals the number of terms in any minimized expression
    of the non-constant subfunction, but the terms of $f$ have one more
    cell (because the support of f contains one more variable), which can
    be calculated from Table 4 and Lemma 3.
    // If no subfunction is constant, then use the decompositions of The-
    orem 1 and the already obtained expressions of the subfunctions of
    f to find the expressions for $f$ (according to Theorem 5). If same
    or complemented terms exist inside an expression then merge them
    according to Theorem 4. At the end keep the expressions with the
    least number of terms.
    // Produce all the equivalent forms for $f$ according to Theorem 5.
    // All the above expressions are the resulting minimized expressions
    of $f$.
end
```

[3] A. Mishchenko, M. Perkowski, ”Logic Synthesis of Reversible Wave Cascades”,International Workshop on Logic And Synthesis 2002, New Orleans, Louisiana, June 4-7, 2002.

[4] A. Sarabi, N. Song, ”A comprehensive approach to logic synthesis and physical design for two-dimensional logic arrays”, DAC 1994, 321-326.

[5] I. Schaefer, M. Perkowski, H. Wu ”Multilevel logic synthesis for cellular FPGAs based on orthogonal expansions”, Proc, IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design, Sept. 1993, Hanburg, Germany, pp. 42-51.

[6] G. Lee ”Logic synthesis for celullar architecture FPGA using BDD”, ASP-DAC 97, pp 253-258 Jan 1997.

[7] G. Lee, R. Drechsler ”ETDD-based Synthesis of term-based FPGAs for incompletely specified boolean functions”, ASP-DAC 1998.

[8] P. Lindgren, R. Drechsler, B. Becker ”Look-up table FPGA synthesis from minimized multi-valued pseudo kronecker expressions”,ISMVL 98.

[9] G. Papakonstantinou, F. Gritzali ”Modulo-2 expressions of switching functions”, Electronic Letters, 13(1977).

Algorithm 2: Min2(input: Switching function f expressed as a set of complex terms, Number of Iterations) returns (Min Expr of f)

```
output, 5Terms, NewTerms, Expr(f): Expression of f in minterm formulation;
loop: integer;
ETDD: Ternary tree;
begin
    output=input;
    loop=0;
    while loop < Number of Iterations do
        5Terms=Pick 5 random terms from output;
        output=output-5Terms;
        Expr($f$) = Min1(5Terms);
        NewTerms=Pick randomly an expression from Expr($f$);
        output=output+NewTerms;
        loop++;
    endw
    return output;
end
```

[10] G. Papakonstantinou "Synthesis of cutpoing cellular arrays with exclusive-OR collector row", Electronic Letters, 13(1977).

[11] J. Preskill "Lecture notes in quantum computing", http://www.Theory.caltech.edu/ preskill/ph229

[12] M. H. A. Khan, M, Perkowski "Logic synthesis with cascades of new reversible gate families", Reed Muller 2003.

[13] G. Papakonstantinou "Cascade Transformation", IEEE Transactions on computers, Jan 1976.

[14] N. Song, M. Perkowski "Minimization of exclusive sums of multi-valued complex terms for logic cell arrays", ISMVL 98, p 32.

[15] N. Song, M. Perkowski "A new approach to and/or/exor factorization for regular arrays", Proc. 1998 Euromicro, pp. 269-276, Vasteras, Sweden, August 25-27, 1998.

[16] C. Bennet "Logical Reversibility of Computation", IBM Journal of Research and Development, 17, 1973, pp. 525-532.

[17] A. Mishchenko, M. Perkowski "Fast Heuristic Minimization of Exclusive-Sums-of-Products", 5th International Reed-Muller Workshop, Starkville, Mississippi, Aug 2001.