Decomposition of Multi-Output Boolean Functions - PRELIMINARY VERSION

Dimitrios Voudouris, Marios Kalathas and George Papakonstantinou
National Technical University of Athens
(funded by the project Protagoras/NTUA)
email: dvoudour@cslab.ece.ntua.gr, mkalath@cslab.ece.ntua.gr, papakon@cslab.ece.ntua.gr

Abstract

In this paper the problem of decomposition, mapping and minimization of multi-output boolean functions is discussed. First the mathematical formulation for single-output boolean functions is presented and then it is extended to multi-output functions. The proposed formulations use function decompositions, ETDDs (EXOR ternary decision diagrams) and multi-valued logic.

1 Introduction

All logic circuits are governed by the rules of boolean logic. Although the recent advancements in the field of VLSI (Very Large Scale Integration) helped to create better circuits (faster, smaller and with less heat dissipation), the technical boundaries imposed by the limitations of the technology are starting to become obvious. This is the reason why the research in the field of boolean logic problems is a hot scientific topic.

One of the most important boolean problems is the effective decomposition and minimization of a boolean (switching) function. This function may have many input variables and many outputs. Every such function can be expressed-decomposed in many different ways (Ashenhurst[1], Curtis[2], Roth and Carp[3] did pioneering work on this field).

A significant kind of expressions for a boolean function are the so-called *Exclusive or Sum Of Products* (ESOP) expressions. They are so important because they can easily be mapped to standard commercial architectures (like FPGAs - Field Programmable Gate Arrays) and furthermore they can be described and analyzed using mathematical terms. Other similar expressions are the *Sum Of Products* (SOP) expressions. For an n-variable function, the upper bound in the number of product terms of ESOPs is $29 \cdot 2^{n-7}, n > 6$ [18] as opposed to 2^{n-1} for SOPs. Morevover, XOR gates have excellent testabillity properties. This is the reason why ESOP expressions are considered to be more important than SOP. Finally the problem of minimizing an ESOP expression of a boolean function, in its most general form (regarding a function with arbitrary number of input variables and number of outputs), is known to be an NP-hard problem. The problem has been solved in the past for special cases regarding the number of input variables and outputs and the number of produced terms [18].

The physical evolution in the theory of ESOP is to create expressions using different kind of terms (more complex). In this work the terms used are called complex terms (or Maitra terms). The expressions produced are called *Exclusive or Sum of Complex Terms* (ESCT). The main advantage of the ESCT expressions over ESOP is that they contain fewer number of terms (and therefore lead to smaller and better logic circuits). Another interesting property is that they can easily be mapped to architectures that are considered reversible [6], therefore they can be used to create circuits with less heat dissipation. They even can be used in quantum computing [6]. The disadvantage is that the effective decomposition and minimization of ESCT expressions seems to be even more difficult than the corresponding one using ESOP expressions.

The problem of decomposing a boolean function to the exclusive-or sum of ESCT functions and to find a near minimal ESCT expression for it has been addressed in the past [6, 7, 8, 9, 10, 11, 14, 15] for single-output functions. In Ref [16] two algorithms (regarding single-output boolean functions) were presented for minimizing the number of complex terms in an ESCT expression. The first one guaranteed minimality for functions up to 5 variables. The second algorithm applied the first one on groups of complex terms, inside the cellular array, as a complex term transformation operation. The previous algorithm has been improved in Ref [17] with the introduction of relative complex terms and equivalent ESCT expressions.

In this work a mathematical formulation is presented (regarding the extension of the algorithms and methods presented in [16] and [17]) for multi-output switching functions.

2 Theoretical background

In this section we provide some background definitions.

Definition 1. A boolean function with n-input variables and m-outputs (multi-output function) is a mapping: $f : \{0,1\}^n \rightarrow \{0,1\}^m$. Variables $x_1, \ldots, x_n$ are called the support of f. If $m = 1$ then f is a single-output boolean function. Generally a multi-output boolean function can be considered as m different single-output functions.

Definition 2. A subfunction $f_i, i = 0, 1, 2$ of a boolean function $f(x_1, \ldots, x_n)$, regarding variable x_1 of f's support is defined as:

- $f_0 = f(0, x_2, \ldots, x_n)$
- $f_1 = f(1, x_2, \ldots, x_n)$
- $f_2 = f_0 \oplus f_1$

Definition 3. Let x_i be binary variable literals, y a binary value (constant input) and G_i arbitrary 2-input 1-output boolean functions ($1 \leq i \leq n$). Then $U = G_n(x_n, G_{n-1}(x_{n-1}, G_{n-2}(x_{n-2}, \ldots, G_1(x_1, y))))$ is an n-variable complex term (or Maitra term) that depends on variables $x_1, \ldots, x_n$. Functions G_i will be called the ESCT cell functions of the term.

A product term is a special case of a complex term where the $G_i(x, y)$ function may be of the form: $xy, \bar{x}y, x\bar{y}, \bar{x}\bar{y}, x, y, 0, 1$. If the last four cases are not allowed then the product term is actually a minterm.

Definition 4. An *Exclusive or Sum of Complex Terms* expression (ESCT expression or Maitra expression) for a switching function f is an exclusive-OR sum of complex terms:

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

where m is the number of complex terms. All complex terms M_i inside the expression have the same variable ordering.

In the previous definition, if instead of complex terms, we use product terms, then the produced expression is called an *Exclusive or Sum Of Products* (ESOP) expression.

Definition 5. A minimal (or exact) expression of a single-output switching function $f(x_1, \ldots, x_n)$ of n variables, is defined as the ESCT expression which has the fewest number of complex terms comparing to any other ESCT expression for this function.

The same definition applies for multi-output boolean functions but in this case different outputs may share common terms, in order to reduce the overall weight.

Definition 6. The weight $w(f)$ of a switching function $f(x_1, \ldots, x_n)$ of n variables is defined as the number of complex terms in a minimal expression of f.

It has been proved in [16] that the complement of a complex term is also a complex term, therefore for a boolean function f: $w(f) = w(\bar{f})$.

Definition 7. The minterm(MT) representation of an n variable switching function f is a 2^n bit vector where the ith bit is 1 if the ith (its binary representation equals i) minterm of f is 1.

For example the MT formulation of $f(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4$ is [8000]. The MT formulation of $f = x_1x_2x_3x_4 \oplus \bar{x_1}\bar{x_2}\bar{x_3}\bar{x_4}$ is [8001], where the numbers are hexadecimal.

Table 1: ESCT cell function 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

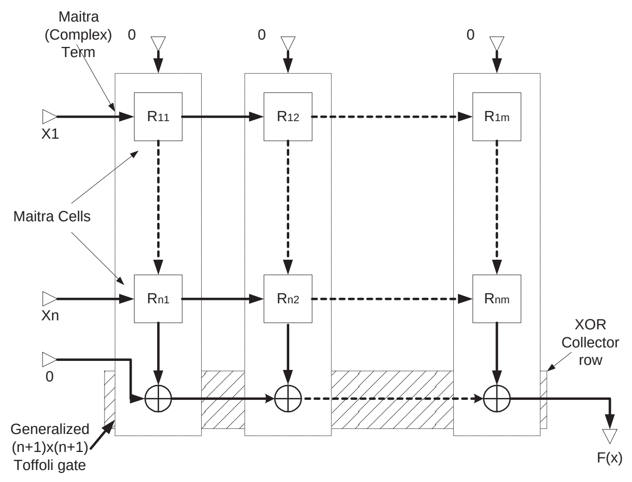

Figure 1: Reversible wave cascade CA

An ESCT expression can be directly mapped to a reversible wave cascade cellular architecture(Fig. 1).

It has been proved [5] that an ESCT cell function doesn't need to belong to the complete set of every two-variable switching function. A set of only six functions is sufficient (complete set). Of course there are many equivalent such sets[13]. We have adopted (in the rest of the paper) one of them which can be seen in the Table 1. The complex terms that use such a complete set for function G are called *restricted complex terms* [5]. From this point on, without loss of generality, when we mention complex terms, we will refer to restricted complex terms.

For single-output boolean functions, we adopt the assumption that the first input (y in Definition 3) of every complex term is the constant 0. Therefore a complex term is characterized by its ESCT cell functions, so we can represent it by a series of indexes (using the corresponding index shown in table 1), utilizing three bits per each. For example function $f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2)x_3 + x_4$ has weight 1. Using functions from the previously defined cell set, it can be represented as: (1541) or in bits: (001101100001).

In the rest of this paper, we will declare a complex term representation of a function by enclosing it in parenthesis.

Definition 8. Let X be variable that takes a value from $V = \{0, \dots, v-1\}$ and $S \subseteq V$. Then X^S is a literal of X such as $X^S = 1$ when $X \in S$ and $X^S = 0$ when $X \in V \backslash S$.

Definition 9. Let X^{S_1}, X^{S_2} be two literals of variable X. Then $X^{S_1} \oplus X^{S_2} \equiv X^{(S_1 \cup S_2) \backslash (S_1 \cap S_2)}$.

Definition 10. A multi-valued input, binary output function f is a mapping $f : V_1 \times V_2 \times \cdots \times V_n \to \{0, 1\}$, where $V_i = \{0, \dots, v_i - 1\}$.

In this paper we examine mappings of the form $f : \{0,1\} \times \cdots \times \{0,1\} \times \{0, \dots, v-1\} \to \{0, 1\}$.

The weight of a multi-valued switching function is defined in accordance to Definition 6. The subfunctions of a multi-valued boolean function f, regarding a binary variable of f's support, are also identical to the ones in Definition 2.

Definition 11. The multiple value minterm (MVMT) formulation m of an $(n+1)$–variable function $f : \{0,1\} \times \cdots \times \{0,1\} \times \{0,\ldots,v-1\} \rightarrow \{0,1\}$ is a $2^{n+\lceil \lg(v) \rceil}$ bit vector. Let $x_1^{\{a_1\}} \cdots x_n^{\{a_n\}} X^S$ be a cube of f, where $a_i \in \{0,1\}$ and $S \subseteq \{0,\ldots,v-1\}$. Let p_1 be equal to the binary number $< a_1,\ldots,a_n >$ and $p_2 = 2^{\lceil \lg(v) \rceil}$. O is a bit vector of size p_2 whose i–th bit is 1 if $i \in S$. Then, bits $[(p_1 \cdot p_2 + p_2 - 1)..(p_1 \cdot p_2)]$ of the MVMT m are equal to O.

For example let $f : \{0,1\}^4 \times \{0,\ldots,v-1\} \rightarrow \{0,1\}$. Intuitively, the MVMT of f can be seen as the interleaving of v MTs of 4-variable functions f_i, where $f_i = f(x_1,x_2,x_3,x_4,i)$. Let $f_i = [d3d4], [c1d5], [d1cf],], [d5cb], [c13b]$ for $i = 0,1,2,3,4$ respectively. Then the MVMT of f is $[1f1f000d0008011f0f0f10131c071c1e]$. More specifically (in terms of bits): $f_0 = [d3d4] = [1101-0011-1101-0100]$, $f_1 = [c1d5] = [1100-0001-1101-0101]$, $f_2 = [d1cf] = [1101-0001-1100-1111]$, $f_3 = [d5cb] = [1101-0101-1100-1011]$, $f_4 = [c13b] = [1100-0001-0011-1011]$

The last two digits of the MVMT (1e) are produced by using the last digits of the MT formulations of f_4, f_3, f_2, f_1, f_0 and padding with zeros until size: $2^{\lceil lg(5) \rceil} = 8$. So we use 3 zeros and then the last digits of f_4, f_3, f_2, f_1, f_0. Thus: (000)(11110)=[1e]. The rest of the digits of MVMT are produced using similar arguments.

In the rest of this paper we will declare a MT or MVMT representation of a function by enclosing it in brackets.

Definition 12. Let x_i be binary literals, y takes a value from $\{0,\ldots,v-1\}$, G_i $(2 \leq i \leq n)$ is an arbitrary boolean function $\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$ and G_1 is an arbitrary mapping $\{0,1\} \times \{0,\ldots,v-1\} \rightarrow \{0,1\}$. Then $U = G_n(x_n, G_{n-1}(x_{n-1}, G_{n-2}(x_{n-2},\ldots,G_1(x_1,y))))$ is a mv-term.

An mv-term can be represented using its MVMT formulation. Alternatively it can be represented as a normal complex term (a series of ESCT cell functions - we will call this the 2v-term) along with a multi-valued variable (this variable will be the y input in Definition 3 - we will call it the mv-var). For example mv-term $(\{3\}434) = yx_1\overline{x_2}x_3$ (y multi-valued, x_1, x_2, x_3 binary variables) has 2v-term (434) and mv-var: 3. Its MVMT representation is: [00300000].

The complement of a mv-term (like the complement of a complex term) is also one mv-term and has complemented 2v-term and complemented mv-var.

2.1 Decompositions

Every boolean function can be expressed with the help of its subfunctions through relations known as boolean decompositions (or expansions).

Definition 13. Let $f(X)$ be a switching function and X the vector of its variables. Let x_1 be one of the variables in the vector X. Then, $f(x_1 = 0, x_2, \ldots)$, $f(x_1 = 1, x_2, \ldots)$ and $\{f(x_1 = 0, x_2, \ldots) \oplus f(x_1 = 1, x_2, \ldots)\}$ are subfunctions of f, regarding variable x_1. For simplicity, in the rest of this paper, we will refer to $f(x_1 = 1, x_2, \ldots)$ as f_1, to $f(x_1 = 0, x_2, \ldots)$ as f_0, to $\{f(x_1 = 0, x_2, \ldots) \oplus f(x_1 = 1, x_2, \ldots)\}$ as f_2 and to x_1 as x.

A boolean function f can be expressed as:

$f(X) = \bar{x}f_0 \oplus xf_1$, $f(X) = xf_2 \oplus f_0$, $f(X) = \bar{x}f_2 \oplus f_1$ (Shannon, positive Davio, negative Davio). Those expressions are valid even if boolean function f has a multi-valued variable, as long as the variable used for the decomposition is binary.

A switching function f can be decomposed using expansions, different than Shannon and Davio. Those are presented in Theorem 1.

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

$$f(X) = (x + f_2) \oplus (x \oplus f_1) \tag{1}$$
$$f(X) = (x + f_0) \oplus (x\bar{f}_1) \tag{2}$$
$$f(X) = (x\bar{f}_2) \oplus (x \oplus f_0) \tag{3}$$
$$f(X) = (x + \bar{f}_0) \oplus (\bar{x} + \bar{f}_1) \tag{4}$$
$$f(X) = (\bar{x} + \bar{f}_2) \oplus \bar{f}_0 \tag{5}$$
$$f(X) = (x + \bar{f}_2) \oplus \bar{f}_1 \tag{6}$$
$$f(X) = (\bar{x}\bar{f}_2) \oplus (x \oplus \bar{f}_1) \tag{7}$$
$$f(X) = (\bar{x}\bar{f}_0) \oplus (\bar{x} + f_1) \tag{8}$$
$$f(X) = (\bar{x} + f_2) \oplus (x \oplus \bar{f}_0) \tag{9}$$

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. Q.E.D.

Based on the previously described decompositions two algorithms (Min1 and EMin1) for decomposing and minimizing single-output switching functions have been presented in [16] and [17] respectively.

Lemma 1. *The relation*
$G_{r_1}(x, y_1) \oplus G_{r_2}(x, y_2) = G_r(x, y_1 \oplus y_2)$, *where G_i are ESCT cell functions, x is a binary variable and $y_1 \neq y_2$, $y_1 \neq \overline{y_2}$*
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. Q.E.D.

It is important to note that the above lemma holds even if y_1, y_2 are mv-terms (it can easily be proved).

If $y_1 = y_2$ or $y_1 = \overline{y_2}$ then $y_1 \oplus y_2$ and consequently $F_r(x, y_1 \oplus y_2)$ are reduced to one complex term[16]. The above lemma shows that we can merge any number of index 1 or 3 ESCT cell functions to one ESCT cell function of index 1 or 3 and the same principle applies to ESCT cell functions of index 2,4 and 5,6.

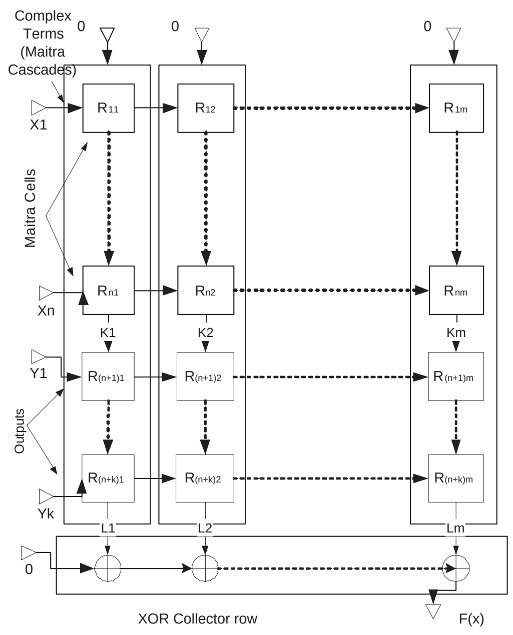

Figure 2: 1st algorithm multi-output architecture

3 Multi-output formulation

The previous theoretical results have been generalized for multi-output switching functions. Two different formulations will be presented. In all these the multi-output function is transformed into a new switching function (we will call it the characteristic function), which is the function that eventually will be minimized. From the minimal (or near minimal) expressions of the characteristic function, we can, easily, obtain the corresponding minimal (or near minimal) expressions for the starting multi-output switching function. In the following sections we will represent the starting multi-output function as **f** and its corresponding characteristic function as g (For convenience we will use bold fonts for multi-output functions and normal for single-output functions). We will analyze the process used to create the characteristic function (it is different in every approach) and present the proposed architecture (where necessary).

3.1 First Formulation

Let $\mathbf{f}(x_1, \ldots, x_n) = \{f^0, f^1, ..., f^{m-1}\}$ be the n-input, m-output switching function to be minimized ($f^i, i = 0 \ldots (m-1)$ are the corresponding outputs of f). In this approach the characteristic function (g) will be a $(n + \lceil logm \rceil)$-input, 1-output switching function described as:

$$g(x_1, \ldots, x_n, y_1, \ldots, y_{\lceil logm \rceil}) = f^i, i = \sum_{j=0}^{j \leq \lceil logm-1 \rceil} y_j 2^j \tag{10}$$

A circuit representation of this architecture is shown in Figure 2.

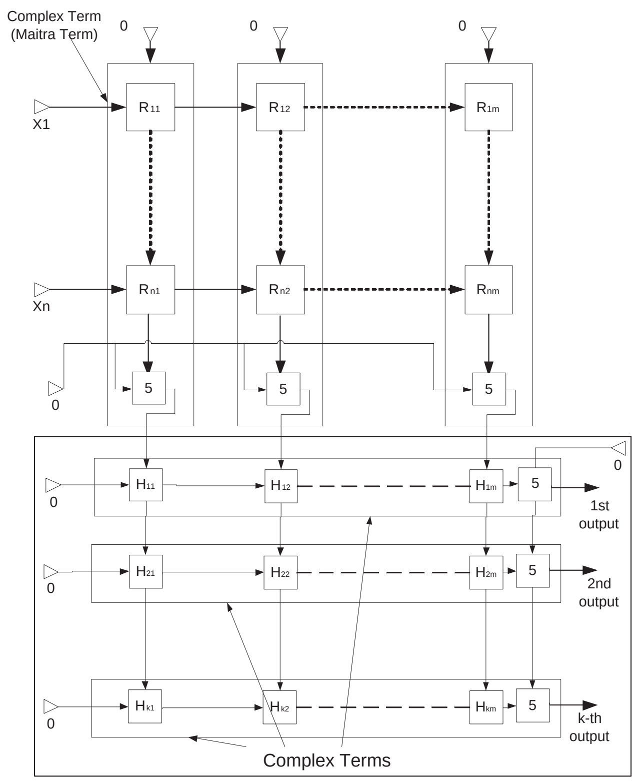

Figure 3: 1st algorithm multi-output architecture-revised

In order to decompose and minimize the characteristic function g, we use any single-output function minimization algorithm, since g is really a single-output function although it has more input variables, when compared to the original multi-output function. From the expressions of g we can acquire the corresponding expressions for the outputs f^i by setting the input variables $y_1, \ldots, y_{\lceil logm \rceil}$ to the corresponding value, according to equation 10.

The first architecture presented in Figure 2 has a small disadvantage. We can acquire only one of the m possible outputs at any given time (we can't have all the possible outputs at the same time). To overcome this limitation we can transform that architecture to the one shown in Figure 3. This transformation does not add any additional columns to the architecture and it is very simple. It was shown before that in order to produce a specific output (suppose output z) from the architecture in Figure 2, we must set inputs $y_1, \ldots, y_{\lceil logm \rceil}$ to the appropriate constant value. It is easy to prove that if inputs $y_1, \ldots, y_{\lceil logm \rceil}$ are set to a specific value then the output value of a cascade L_i will be 0,1,K_i or $\overline{K_i}$ (refer to Figure 2). We have the following cases:

- If $L_i = 0, 1$, then this cascade is not used for output i. Therefore cell H_{iz} will be of index 6(Figure 3). If $L_i = 1$ then the output must be complemented, thus we change the initial input of the horizontal cascade z from 0 to 1 and vice versa (in order to complement the output).

- If $L_i = K_i, \overline{K_i}$, then cell H_{iz} will be of index 5 (in order to combine the output L_i of this cascade with the ones of the other cascades to produce the output). If $L_i = \overline{K_i}$, we also complement the initial input of the horizontal cascade (just like in the previous case).

3.2 Second formulation

In this case the characteristic function is produced as the interleaving of the MT representations of **f**'s outputs, according to definition 11. In this case g has the same number of input variables with **f** but the first variable is considered multi-valued.

The expressions that will be produced from the decomposition and minimization of g will be composed of mv-terms. All 2v-terms (of their corresponding mv-terms) in the final expression are used to produce the minimal expressions for each output (in the previous case some of them were used) but their initial constant input is different (for some inputs it may be one and for others it may be zero). This is declared by the multi-valued variable.

In this architecture we can't have every output produced at the same time.

For the decomposition and minimization of the characteristic function we can easily extend (in a heuristic manner) the single-output minimization algorithm Min1[16]. Equal mv-terms are considered those that have equal mv-vars and 2v-terms and complemented are considered those that have complemented mv-vars and 2v-terms.

4 Conclusions

The main contribution of this paper can be summarized as follows:

1. Two mathematical formulations for the decomposition and minimization of multi-output boolean functions are presented, as extensions of the methods presented in [16] and [17] (they concerned single-output functions).

2. An extension of the complex term to the mv-term has been presented, which is a complex term having its initial constant input multi-valued. This can be used to create expressions for boolean functions with their first variable multi-valued.

References

[1] R.L. Ashenhurst: The decomposition of switching functions, Ann. Computation Lab. Harvard University, vol. 29, pp. 74-116, 1959.

[2] H.A. Curtis: A new approach to the design of switching circuits, Princeton, N.J. Van Nostrand, 1962.

[3] J.P. Roth, R.M. Karp: Minimization over Boolean graphs, IBM Journal, April 1962, pp. 227-238.

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

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

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

[7] A. Sarabi, N. Song, M. Chrzanowska-Jeske, M. Perkowski: A comprehensive approach to logic synthesis and physical design for two-dimensional logic arrays, DAC 1994, 321-326.

[8] 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, Hanburg, Germany, pp. 42-51, Sept. 1993.

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

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

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

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

[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] D. Voudouris, S. Stergiou, G. Papakonstantinou: Minimization of reversible wave cascades, IEICE Trans. on Fundamentals, accepted for publication.

[17] D. Voudouris, G. Papakonstantinou: Maitra Cascade Minimization, 6th IWSBP, p 209, 2004.

[18] S. Stergiou, D. Voudouris, G. Papakonstantinou: Multiple-Value Exlusive-Or Sum-Of-Products Minimization Algorithms, IEICE Trans. on Fundamentals, vol.E87-A,No 1,Jan. 2004.