

Iterative Method of Minimization of Arbitrary Boolean Functions of Many Variables

Arkadij Zakrevskij

$$\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 x_5 \bar{x}_6.$$

Abstract — An iterative algorithm of minimization of Boolean functions of many variables based on usage of parallel operations above adjacent elements in Boolean space of arguments is offered. It includes the operation of fast finding of elements of characteristic set with small number of neighbors and creation of implicants defined by them. The iterative procedure of application of this operation to sequentially reduced characteristic set and operation of simplification of the obtained conjuncts lead to a correct solution.

Index Terms—Boolean function minimization, iterative algorithm, prime implicants, computer experiment

I. INTRODUCTION

The classical problem of minimization of Boolean functions in the class of disjunctive normal forms (DNF) can be formulated as follows.

Let us define an arbitrary Boolean function $f(x) \equiv f(x_1, x_2, \dots, x_n)$ in the standard way by a Boolean vector f with 2^n components numbered from left to right, starting with zero. Here the component number k sets the value of the function f on the set of values of arguments representing the generally accepted binary code b^k of number k .

For example, the following vector f represents the Boolean function f of six variables $x_1, x_2, x_3, x_4, x_5, x_6$, receiving value 1 on 27 collections of values of arguments constituting the characteristic set $M^1 = \{000000, 000011, 000101, \dots, 111110\}$ of function f . The ordinal numbers of the appropriate components of vector f are 0, 3, 5, ... 62.

-----	-----	-----	-----	x_2
-----	-----	-----	-----	x_3
-----	-----	-----	-----	x_4
-----	-----	-----	-----	x_5
-----	-----	-----	-----	x_6

10010101 00100110 00101101 10110010
 | 00010010 01010100 10001001 00111010 x_1

Note, that such a vector can be interpreted as a perfect normal form of a Boolean function, which terms are represented by single components of the vector and their codes. For example, the code $b^{10} = 001010$ of element f_{10} defines the complete elementary conjunction

The problem consists in finding a DNF, minimized as possible, for the function f presented by a given vector f . Let us remark, that the traditional methods of solving this task need run-time fast growing with growth of the number of variables n , and practically become unacceptable at $n > 20$, when the number of terms in the perfect DNF is measured in millions [1]. In this connection an original method is offered in the given paper, for obtaining DNF of Boolean functions the number of which arguments can reach 24. The method is based on application of efficient parallel operations above Boolean 2^n -vectors offered in papers [2, 3].

One of such operations is the *operation of conjunctive symmetrizing* the vector f by the variable x_i , denoted below as $Sf \wedge i$. At its execution the vector f is divided into 2^{n-1} couples of components adjacent by the variable x_i and both units of each couple gain the value equal to conjunction of the values of these units. Let us remind that adjacent such components of the vector f are named, which correspond to sets of values of arguments distinguishing exactly in one argument.

When the number of variables n exceeds 5, it is convenient to represent vector f as a Boolean matrix of the size $2^5 \times 2^{n-5}$, presenting its 32-component rows by words in computer memory (that is adequate for the majority of modern computers). Then any two units of vector f adjacent by the variable x_i will belong to the same word if $i < 6$ and to different words otherwise, that is possible to use for acceleration of calculations.

II. BUILD-UP OF THE BOOLEAN MATRIX OF NEIGHBORHOOD N

At the first stage of the offered method the *Boolean matrix of neighborhood N* of the size $n \times 2^n$ is created by means of n -fold application of the operation $Sf \wedge i$. Each row n_i of this matrix represents the result of execution of the operation $Sf \wedge i$ at a concrete parameter value i (from 1 up to n). The matrix N represents the structure of the characteristic set M^1 of function $f(x)$, where this function receives value 1. The element n_i^k of the matrix N receives value 1 if and only if the element f_k of vector f equals 1 and has a neighbor by the variable x_i which also has value 1. Thus, the row n_i of this matrix represents the subset of elements from the set M^1 , having in the same set neighbors by the variable x_i , and the column n^k displays, by which variables has neighbors the element from M^1 presented by the component f_k of vector f .

Manuscript received 23 March, 2009.

Arkadij Zakrevskij is with the United Institute of Informatics Problems of the National Academy of Sciences of Belarus, 220023, Minsk; e-mail: zakr@tut.by.

Let us illustrate obtaining the matrix N with an example of the same vector f , specifying a random Boolean function of six variables:

10010101	00100110	00101101	10110010	
00010010	01010100	10001001	00111010	f
00010000	00000100	00001001	00110010	
00010000	00000100	00001001	00110010	n_1
00000101	00100010	00000101	00100010	
00000000	00010000	00000000	00010000	n_2
00000100	00000100	00100000	00100000	
00010000	00010000	00001000	00001000	n_3
00010001	00100010	00000000	00100010	
00000000	01000100	10001000	00100010	n_4
00000101	00000000	00000101	10100000	
00000000	01010000	00000000	00001010	n_5
00000000	00000000	00001100	00110000	
00000000	00000000	00000000	00110000	n_6

III. MATRIX REPRESENTATION OF DNF

In the similar form, the Boolean vector of solution g and the Boolean matrix of solution D of the same dimension as f and N may be used to represent the required DNF, considered as some collection of intervals of the Boolean space $M = \{0, 1\}^n$. Some elements of these intervals, by one for each interval, are marked with ones in vector g , and the internal variables of these intervals are shown in columns of matrix D .

Considering vectors f and g as sets of the elements of space M marked in them, and matrix N and D as subsets of Cartesian product of sets x and M , we shall formulate obvious

A f f i r m a t i o n 1. $g \subseteq f$ and $D \subseteq N$.

In other words, vector g and matrix D can be obtained accordingly from vector f and matrix N by replacement in them some ones with zeros, as it is done in the method circumscribed in this paper.

For example, according to that method, vector f

10010101	00100110	00101101	10110010
00010010	01010100	10001001	00111010

is transformed to vector g

10010100	00000110	00101000	10010000
00000010	01000000	10000001	00001000

and matrix N – to matrix D presenting DNF of function f (it will be shown later).

In the more known form this DNF is set by a ternary matrix, which columns represent elementary conjunctions ($x_1 x_2 x_3 x_4 x_5 x_6$, $\bar{x}_2 \bar{x}_3 \bar{x}_4 x_5 x_6$, etc.)

1	0-0-0000-111-1
2	00-0-111100111
3	00011-01101001
4	0011-010010-11
5	01-0110-11-01-
6	011100-0-01010

The number p of conjunctions in the obtained DNF is equal to the weight of the vector of solution g , and the length of DNF (the total of conjunction ranks) is equal to $pn - q$, where q is the number of elements in matrix D .

Let us remark, that the introduced above Boolean matrices and vectors at program implementation of the offered method are handled in internal cycles and, therefore, should be represented in RAM, that limits their allowable size. Taking into account parameters of modern PCs, it is possible to affirm that the given method is implemented fast enough if the number of variables of the minimized Boolean function does not exceed 24.

IV. SELECTION OF ELEMENTS WITH SMALL NUMBER OF NEIGHBORS

The build-up of a DNF implementing function f is expedient to begin with the search for elements of the solution kernel – obligatory simple implicants of high rank. Let us term obligatory such a simple implicant of function f , which enters anyone shortest DNF of this function. The search is facilitated by preliminary allocation in vector f of elements of characteristic set M^1 with small number of neighbors, as such elements can determine implicants of high rank displayed by elementary conjunctions with many literals.

The numbers of neighbors for all elements of vector f with value 1 are presented by the finite-valued vector w :

10010101	00100110	00101101	10110010
00010010	01010100	10001001	00111010
0..2.3.3	..2..22.	..1.23.3	1.62..3.
...2..0.	.2.3.2..	1...3..1	..332.3.w

However, it is more convenient in the long term for program implementation of the subsequent calculations, to sort elements by the number of neighbors and to present the result by a series of Boolean vectors m_i in which ones mark elements with i neighbors.

For the considered example, at $i < 4$, we shall receive:

10000000	00000000	00000000	00000000	
00000010	00000000	00000000	00000000	m_0
00000000	00000000	00100000	10000000	
00000000	00000000	10000001	00000000	m_1
00010000	00100110	00001000	00010000	
00010000	01000100	00000000	00001000	m_2
00000101	00000000	00000101	00000010	
00000000	00010000	00001000	00110010	m_3

It is easy to receive these vectors – rows of the Boolean matrix M , by efficient component-wise operations above rows of matrix N , that essentially accelerates the search for appropriate implicants.

V. FINDING OF PRIME IMPLICANTS

Let us designate through t^k the ternary vector obtained from the Boolean vector b^k (the code of element f_i) by appropriation of value "-" to components marked with ones in column n^k of matrix N . Vector t^k can be interpreted as some interval Int_k of the Boolean space $M = \{0, 1\}^n$, and also as

corresponding elementary conjunction, which can appear a prime implicant of the function f .

A f f i r m a t i o n 2. The vector t^k represents an obligatory prime implicant of the function f if and only if $Int_k \subseteq M^1$.

A f f i r m a t i o n 3. For each obligatory prime implicant of the function f there exists in matrix N a column n^k , appropriate to which ternary vector t^k represents this implicant.

Obligatory prime implicants of rank n are easily found – they are represented by elements of set M^1 not having neighbors. They are enumerated in vector m_0 and represented in corresponding columns of matrix N , not containing 1s. Similarly, all obligatory prime implicants of rank $n-1$ are enumerated in vector m_1 and also represented in appropriate columns of matrix N – containing one 1.

The detection of obligatory prime implicants of smaller rank is a little bit more difficult. However, it is easy enough for ranks $n-2$ and $n-3$, being carried out by means of component-wise operations above some columns of the neighborhood matrix N .

A f f i r m a t i o n 4. Assume, that element f_k of vector f has two neighbors. Then vector t^k represents an obligatory prime implicant of the function f if and only if the component f_j of vector f is equal to 1, where $b^j = b^k \oplus n^k$.

For example, $b^{27} \oplus n^{27} = 011011 \oplus 100001 = 111010$, $j=58$ and $f_{58} = 1$; therefore, ternary vector $t^{27} = -1101-$ represents an obligatory prime implicant.

A f f i r m a t i o n 5. Let us assume, that element f_k of vector f has three neighbors. Then vector t^k represents an obligatory prime implicant of the function f if and only if $n^k \leq n^j$ (the vector n^k is covered with vector n^j), where $b^j = b^k \oplus n^k$.

The proof of this assertion is based on the fact, that the vectors b^k and b^j are opposite elements of the interval Int_k of rank 3 and together with their neighbors they cover all eight elements of this interval (see fig. 1).

There is a problem of practical expediency of checking the components of vector f for satisfying the conditions formulated in the assertions 4 and 5. Let us assume that a Boolean function f of n variables is preset with probability $1/2$ of having value 1 in its components, independently of one another. Consider now some component with value 1 which has k neighbors. How probable it is, that this component determines an appropriate prime implicant of rank $n-k$?

A f f i r m a t i o n 6. The probability of such an event is equal to $1/2^t$, where $t = 2^k - (k+1)$.

This probability fast tends to zero. For example, at $k=2$, 3, 4 and 5 it equals $1/2$, $1/16$, $1/2048$ and $1/67\,208\,864$, accordingly, being independent on n .

Therefore, in the offered heuristic algorithm only such single components of vector f are exposed to analysis, the number of the neighbors for which does not exceed three.

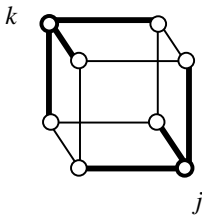


Fig. 1

VI. OBTAINING A PARTIAL SOLUTION

In the offered algorithm implicants of high ranks are sequentially found. The result is fixed by introduction of ones into the vector of solution g (at first $g = 0$), definition of corresponding to them columns of the solution matrix D , and correction of vector f^* , representing the set of yet uncovered elements of the characteristic set M^1 .

1. $g := m_0$, $f^* := f \setminus m_0$. So the implicants of rank n are found (in the given example there exist two such implicants presented by vectors 000000 and 100110 and defined by elements of vector f with numbers 0 and 38).

2. Here are considered sequentially the elements of vector f , marked both in vectors m_1 and f^* . For a current element f_k the corresponding to it vector b^k is taken, which component marked by one in column n^k of the neighborhood matrix N , is changed for symbol «-». The result represents a prime implicant of rank $n-1$. The vectors g and f^* are accordingly corrected: in the first vector one 1 is added, and from the second two 1s are deleted.

So in the given example elements with numbers $k=18, 24, 48$ and 55 are sequentially considered, to which sets 010010, 011000, 110000 and 110111 correspond and which determine prime implicants of rank $n-1$: 01-010, 0110-0, 110-00, -10111. The vector of solution g receives the value

```
10000000 00000000 00100000 10000000
00000010 00000000 10000001 00000000  g
```

and accordingly (as a result of deleting covered elements – they are marked by underline) the value of the vector f^* varies:

```
00010101 00100110 00001100 00010010
00010000 01010100 00000000 00111010  f*
```

3. At this stage the elements f_k with two neighbors marked simultaneously in vectors m_2 and f^* are considered sequentially and implicants of rank $n-2$ are created.

First elements f_k satisfying the condition of the assertion 4 are found. Corresponding components of vector g receive value 1, the columns d^k of matrix D remain equal to columns n^k of matrix N , and elements covered with intervals Int_k are deleted from vector f^* .

If the condition of the assertion 4 is not satisfied, one of two neighbors of element f_k is selected. (desirably yet not covered). In the column d^k only one corresponding 1 remains and the operation foreseen for an element with one neighbor is fulfilled.

In the given example (it appears rather simple) that leads to the final solution – covering all elements of the characteristic set M^1 , obtaining of the couple of vectors

```
10010100 00000110 00101000 10010000
00000010 01000000 10000001 00001000  g
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000  f*
```

build-up of the matrix of solution D , obtained from N by deleting of one unity in every column marked in vector

```

00010000 00100110 00001000 00010000
00010000 01000100 00000000 00001000  m2

```

and deleting of all unities in the columns which have not been marked in vector g :

```

00010000 00000100 00000000 00010000
00000000 00000000 00000001 00000000  d1
00000100 00000010 00000000 00000000
00000000 00000000 00000000 00000000  d2
00000000 00000000 00100000 00000000
00000000 00000000 00000000 00000000  d3
00000000 00000010 00000000 00000000
00000000 00000000 10000000 00000000  d4
00000100 00000000 00000000 10000000
00000000 01000000 00000000 00001000  d5
00000000 00000000 00001000 00010000
00000000 00000000 00000000 00000000  d6

```

4. If the vector f^* remains nonzero, the elements with three neighbors marked simultaneously in vectors m_3 and f^* are considered. The elements satisfying the condition of the assertions 5 are found, and the corresponding implicants of rank $n - 3$ are entered into solution. At omission of the given condition the implicants of higher rank ($n - 1$ and $n - 2$) are found, defined by these elements.

As a result of the circumscribed procedure of processing of elements of vector f , having no more than three neighbors, we obtain a set of implicants constituting a *partial solution* S , and vector f^* , representing *residual* – the collection of uncovered by this solution elements of set M^1 .

VII. ITERATIVE ALGORITHM

The idea of this heuristic algorithm consists in the following. At first it discovers a partial solution for the vector f , then fulfils the same operation for the residual f^* , supplementing the set of obtained implicants and accordingly simplifying vector f^* (deleting some ones from it). If after simplification vector f^* will contain some ones, the following iterations are fulfilled up to that moment, when f^* becomes equal to zero.

The implicants obtained at that can be not obligatory and even not prime. Therefore in conclusion they are reduced to prime ones (by lowering the rank), and also the obtained doubles are eliminated.

Let us demonstrate the algorithm on a concrete example, Let $n = 19$ and each element of vector f receives value 1 with probability $p = 1/4$. At this supposition a random vector f with 147 232 elements was generated and the neighborhood matrix N with $2^{19} = 534 288$ columns was constructed with following distribution of columns on number of ones in them ($N(i)$ columns contain i ones each):

```

i = 0  1  2  3  4  5  6
N(i) = 296 2087 7180 16352 25357 29868 26913
i = 7  8  9  10 11 12 13 14 15 16
N(i) = 19406 11379 5401 2087 690 172 35 8 1 0

```

At the first iteration of algorithm the elements of the set M^1 with number of neighbors $i = 0, 1, 2$ and 3 are handled, that is $296 + 2087 + 7180 + 16352 = 25915$ elements. The obligatory prime implicants defined by them are found, altogether $296 + 2082 + 1974 + 80 = 4432$. The total number of the prime implicants retrieved at the first iteration (together with non-obligatory ones) is equal to 24 379. They are marked in the vector of solution g and the elements covered with them are deleted from the variable residual vector f^* (in the beginning equal to f). The number of ones in vector f^* becomes equal to 81 910.

At the second iteration a new matrix N , representing the relation of neighborhood on elements of the set M^1 marked in the residual vector f^* , is considered. In this vector the elements having no more than three neighbors in the same vector are discovered, and determined by them implicants are found. Vectors g and f^* gain new values. If after that $f^* \neq 0$, the following iteration is fulfilled.

So for the considered example four iterations are fulfilled, before vector f^* becomes equal to 0 and all elements of the set M^1 appear covered with 61 477 implicants with the following distribution on ranks (for example, there exist 2 458 implicants of rank 19):

19 – 2 458, 18 – 33 668, 17 – 25 237, 16 – 114.

As was already said, not all these implicants appear prime. Therefore two procedures of bringing the obtained solution to correct appearance are in conclusion fulfilled. First of them simplifies implicants, appealing to the initial neighborhood matrix N and deleting from implicants some literals if possible. It brings to a new distribution of implicants on ranks:

19 – 296, 18 – 20 933, 17 – 38 617, 16 – 1 631.

The second procedure eliminates doubles among the obtained implicants, therefore the number of latter's is reduced down to 60 972, and their distribution on ranks receives the following appearance:

19 – 296, 18 – 20 933, 17 – 38 353, 16 – 1 390.

VIII. COMPUTER EXPERIMENTS

The explained above iterated algorithm was software implemented in C++ and tested on the computer (Pentium IV, 2.8 GHz). In order to make more convenient dealing with Boolean vectors, in which the neighbors for considered elements of vector f are enumerated, the neighborhood matrix N was transposed beforehand.

A series of experiments were carried out on a set of pseudo-random Boolean functions with two parameters: the number of variables n and the density of ones r , defining the expected number of ones q in vector f representing Boolean function f ($r = 32q / 2^n$). For example, at $r = 16$ an absolutely random Boolean function is considered, receiving value 1 on each element of Boolean space with probability $1/2$.

First of all, the boundaries of practical applicability of the offered algorithm in the space of parameters n and r were defined. The point is, that at a large enough value of parameter r the algorithm can stop the operation after some

number of iterations, because the values $N(0)$, $N(1)$, $N(2)$ and $N(3)$ can appear equal to zero, while vector f^* will remain distinct from $\mathbf{0}$.

For example, if $n=17$ and $r=15$, the algorithm stops after eight iterations, finding only 636 implicants. But at $n=17$ and $r=14$ it discovers after 90 iterations 20 077 implicants covering the whole set M^1 (after consequent simplification of these implicants and eliminating doubles their number is reduced to 19 811). Therefore, the couple $(n, r) = (17, 14)$ is a unit of the upper bound of the algorithm usage.

In table I the basic characteristic of this boundary obtained experimentally is represented. The table strings corresponds to the numbers of variables n (from 4 up to 23) and appropriate to them maximum values of the parameter r , at which the program works correctly.

TABLE I

n	r	N	C	S	It	T
4	16	5	3	10	1	0.00
5	16	16	8	27	1	0.00
6	16	31	14	62	1	0.00
7	16	62	31	169	1	0.00
8	16	138	58	360	2	0.00
9	16	274	107	744	3	0.00
10	16	556	194	1513	3	0.00
11	16	1083	379	3355	4	0.01
12	16	2203	735	7127	5	0.03
13	16	4337	1414	15057	7	0.09
14	16	8734	2780	32266	12	0.35
15	15	16404	5313	67324	13	1.01
16	14	31021	10181	139827	13	3.12
17	14	61150	19811	291507	90	24.37
18	13	114347	38007	500357	21	42.26
19	12	212620	72343	1220742	12	148
20	11	392995	137215	2462996	10	610
21	11	786345	270642	5121875	18	2499
22	10	1442274	512529	10255122	10	8858
23	9	2620069	966357	20386490	8	31064

The following values are shown in strings of this and other tables: N – the number of ones in vector f (the number of implicants in the perfect DNF of function f), C – the number of implicants in obtained DNF, S – the length of obtained DNF (the total of implicant ranks), Q_k – the number of implicants of the rank k in obtained DNF (at $k = n, n-1, n-2, n-3, n-4$), It – the number of iterations, T – the run-time in seconds.

The table II shows the behavior of the algorithm at the number of variables $n = 17$. It is evident, that the number of iterations It and the run-time T grow at increase of the density of ones r .

TABLE II

n	r	N	C	S	It	T
17	2	12285	7856	127689	2	0.27
17	4	20427	11117	177205	2	0.53
17	6	28594	13761	215604	3	1.90
17	8	36507	15621	240722	3	4.04
17	10	44756	17348	263178	4	6.49
17	12	52999	18691	279341	7	8.43
17	14	61150	19811	291507	90	24.52

The behavior of the algorithm at the greatest possible for it number of variables ($n=24$) is shown in table III. At increase of the density of ones r by one the run-time T grows more than twice, reaching 6.8 hours at $r = 4$.

TABLE III

n	r	N	C	S	It	T
24	1	1047350	685881	15982597	2	1693
24	2	1571532	919682	21231157	2	4068
24	3	2095590	1124293	25759214	3	10695
24	4	2619724	1297946	29532155	3	24348

Table IV shows some additional results of computer experiments at the number of arguments $n = 17$. Apparently, at increase of the density of ones r the distribution of the obtained implicants on ranks fast varies in favor of implicants of smaller ranks.

TABLE IV

r	Q_{17}	Q_{16}	Q_{15}	Q_{14}	Q_{13}	It	T
2	2283	5283	290	0	0	2	0.27
4	1147	8162	1802	6	0	2	0.53
6	417	8406	4887	51	0	3	1.90
8	135	6370	8883	233	0	3	4.04
10	41	3864	12456	986	1	4	6.49
12	6	1880	13899	2896	10	7	8.43
14	1	684	12842	6224	60	9	24.52
							0

IX. CONCLUSION

In the offered algorithm of minimization of Boolean functions of many variables (up to 24) the following ideas are implemented:

1) The role of elementary operands is played by Boolean vectors with 2^n components representing arbitrary Boolean functions of n variables.

2) The Boolean matrix of neighborhood N by the size $n \times 2^n$ is created, mapping the structure of the characteristic set M^1 .

3) With its help prime implicants of four higher ranks are quickly found, coating a part of the set M^1 .

4) The structure of the residual is represented by a new matrix N , the new implicants of high rank are found, etc. The iterations will stop, when the set M^1 becomes empty.

5) Finally, the obtained implicants are transformed to prime ones and any doubles are eliminated.

The algorithm is implemented by the efficient program, developed by Nikolaj Toropov from the UIIP of NAS of Belarus. It can appear useful at solution of systems of the Boolean equations, verification of logic circuits and solution of other labor-consuming tasks of the theory of Boolean functions.

REFERENCES

- [1] Zakrevskij A. D. Logical design of cascade circuits. – oscow, 1981 (in Russian).
- [2] Zakrevskij A. D. Parallel operations over neighbors in Boolean space. – Proceedings of the Sixth International Conference CAD DD-07, Minsk, 2007. Vol. 2, pp. 6–13.
- [3] Arkadij Zakrevskij. Programming Calculations in Many-Dimensional Boolean Space // Radioelectronics & Informatics, No 1(40), January-March 2008, pp. 19-25.

Reduction of Hardware Amount for Control Unit with Address Transformer

Alexandr A. Barkalov, Larisa A. Titarenko, Alexandr S. Lavrik

Abstract — The method of hardware reduction is proposed oriented on control units and CPLD chips. The method is based on a wide fan-in of PAL macrocells allowing using more than one source of microinstruction address. The method of logical condition replacement is used for optimization of microinstruction addressing block. An example of proposed method application is given.

Index Terms—Address transformer, CMCU, CPLD.

I. INTRODUCTION

Complex programmable logic devices (CPLD) are widely used for implementation of logic circuits of control units [1]. As a rule, CPLD include macrocells of programmable array logic (PAL) [2], [3]. To design a logic circuit with optimal characteristics, some peculiarities of logic elements in use and a control algorithm to be interpreted should be taken into account. If a control algorithm is represented by a linear graph-scheme of algorithm (GSA), then it can be interpreted using a model of compositional microprogram control unit (CMCU) [4]. One of the distinctive features of CPLD is the wide fan-in of macrocells [5], [6]. It can be used for increasing of the number of sources for classes of pseudoequivalent operational linear chains [7], [8]. The method is proposed in this article based on the abovementioned feature of CPLD, as well as on the replacement of logical conditions [1].

The aim of this research is reduction of the hardware amount in logic circuit of CMCU due to simultaneous use of more than one code source and the replacement of logical conditions. The task of research is the development of design method resulted in the hardware amount decrease for blocks of microinstruction addressing and microinstruction address transformer.

II. FEATURES OF CMCU WITH MICROINSTRUCTION ADDRESS TRANSFORMER

Let GSA Γ be represented by sets of vertices B and arcs E . Let $B = \{b_0, b_E\} \cup E_1 \cup E_2$, where b_0 is an initial vertex, b_E is a final vertex, E_1 is a set of operator vertices, where $|E_1| = M$, and E_2 is a set of conditional vertices. A vertex $b_q \in E_1$ contains a microinstruction $Y(b_q) \subseteq Y$, where $Y = \{y_1, \dots, y_N\}$ is a set of data-path microoperations [1]. Each vertex $b_q \in E_2$ contains a single element of the set of logical conditions $X = \{x_1, \dots, x_L\}$. Let GSA Γ be a linear GSA, that is a GSA with more than 75% of operator vertices.

Let us form a set of operational linear chains (OLC) $C = \{\alpha_1, \dots, \alpha_G\}$ for GSA Γ , where each OLC $\alpha_g \in C$ is a sequence of operator vertices and each pair of its adjacent components corresponds to some arc of the GSA. Each OLC $\alpha_g \in C$ has only one output O_g and the arbitrary number of inputs. Formal definitions of OLC, its input and output can be found in [4]. Each vertex $b_q \in E_1$ corresponds to microinstruction MI_q kept in a control memory (CM) of CMCU and it has an address $A(b_q)$. The microinstructions can be addressed using

$$R = \lceil \log_2 M \rceil \quad (1)$$

bits, represented by variables $T_r \in T = \{T_1, \dots, T_R\}$. Let OLC $\alpha_g \in C$ include F_g components and the following condition takes place:

$$A(b_{g_{i+1}}) = A(b_{g_i}) + 1, \quad (2)$$

In equation (2) b_{g_i} is the i -th component of OLC $\alpha_g \in C$, where $i = 1, \dots, F_g - 1$.

If outputs O_i, O_j are connected with an input of the same vertex, then OLC $\alpha_i, \alpha_j \in C$ are pseudoequivalent OLC (POLC) [2]. Let us construct the partition $\Pi_C = \{B_1, \dots, B_I\}$ of the set $C_1 \subseteq C$ on the classes of POLC. Let us point out that $\alpha_g \in C_1$ if $\langle O_g, B_E \rangle \notin E$. Let us encode the classes $B_i \in \Pi_C$ by binary codes $K(B_i)$ with

Manuscript received March 7, 2009.

A. A. Barkalov is with University of Zielona Gora, Poland.

e-mail: A.Barkalov@iie.uz.zgora.pl

L. A. Titarenko is with University of Zielona Gora, Poland.

e-mail: L.Titarenko@iie.uz.zgora.pl

A. S. Lavrik is with Donetsk National Technical University, Donetsk, Ukraine. e-mail: alexandrnordlavrik@gmail.com

$$R_1 = \lceil \log_2 I \rceil \quad (3)$$

bits and use the variables $\tau_r \in \tau = \{\tau_1, \dots, \tau_{R_1}\}$ for the encoding. In this case a GSA Γ can be interpreted using the model of CMCU U1 with address transformer (Fig. 1).

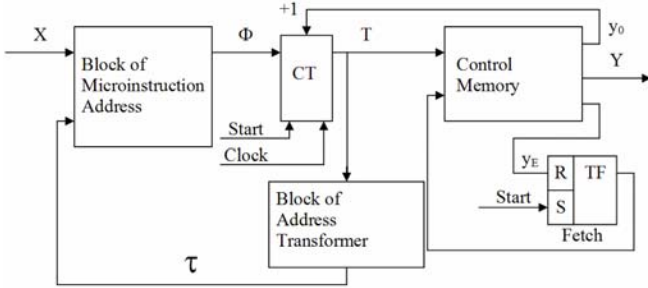


Fig. 1. Structural diagram of CMCU U1

The pulse *Start* causes loading of the first microinstruction address into a counter CT and set up of a fetch flip-flop TF. If *Fetch* = 1, then microinstructions can be read out the control memory CM. If a current microinstruction does not correspond to an OLC output, then a special variable y_0 is formed together with microoperations $Y_q \subseteq Y$. If $y_0 = 1$, then content of the CT is incremented according to the addressing mode (2). Otherwise, a block of microinstruction address BMA generates functions

$$\Phi = \Phi(\tau, X) \quad (4)$$

to load the next microinstruction address into the CT. In the same time, a block of address transformer BAT generates functions

$$\tau = \tau(T). \quad (5)$$

If the output of OLC $\alpha_g \notin C_1$ is reached, then $y_E = 1$. It causes reset of TF and operation of CMCU U1 is terminated.

Such an organization of CMCU permits decrease of the number of terms in functions Φ from H_1 till H_0 , where H_1 , H_0 is the number of terms for equivalent Moore and Mealy finite state machines (FSM) respectively. But the block BAT consumes some macrocells or cells of PROM used for implementation of CM. In this article we propose some CMCU U_2 , where $H_2 = H_0$ and the block BAT consumes less hardware than its counterpart in U_1 . Here H_2 means the number of terms in functions Φ for CMCU U_2 .

III. MAIN IDEA OF PROPOSED METHOD

Let us point out that logic circuits for BMA, CT, TF and BAT are implemented as the parts of CPLD. To implement the CM one should use PROM chips with t outputs, where $t \in \{1, 2, 4, 8, 16\}$. Let us address the components of OLC

$\alpha_g \in C_1$ in such a manner that condition (2) takes place and the maximal possible amount of classes $B_i \in \Pi_C$ is represented by a single generalized interval of R-dimensional Boolean space. Such an addressing needs a special algorithm which should be developed.

Let $\Pi_C = \Pi_A \cup \Pi_B$, where $B_i \in \Pi_A$ if this class is represented by one interval, and $B_i \in \Pi_B$ otherwise. The counter CT is a source of the codes for $B_i \in \Pi_A$. If condition

$$\Pi_B = \emptyset \quad (6)$$

takes place, then the block BAT is absent. Otherwise, only output addresses for OLC from classes $B_i \in \Pi_B$ should be transformed. It is enough

$$R_2 = \lceil \log_2 (I_B + 1) \rceil \quad (7)$$

bits for such an encoding, where $I_B = |\Pi_B|$ and 1 is added to take into account the case when $B_i \in \Pi_A$. Let us point out that some part of these codes can be implemented using free outputs of PROM. Let us use the hot-one encoding of microoperations [2] when CM word has $N+2$ bits. In this case the CM can be implemented using

$$R_0 = \left\lceil \frac{N+2}{t} \right\rceil \quad (8)$$

chips with enough amount of cells (not less than M). Obviously, that R_3 outputs of PROM are free, where

$$R_3 = R_0 * t - N - 2. \quad (9)$$

If condition

$$R_3 \geq R_2 \quad (10)$$

takes place, then the CM is a source of the codes for $B_i \in \Pi_B$ and the block BAT is absent. This approach permits to decrease the number of PAL macrocells in the logic circuit of block BMA, as well as the number of PROM chips used for the address transformation.

Further optimization of the block BMA logic circuit is possible due to the logical condition replacement [1]. In this case the set X is replaced by some set $P = \{P_1, \dots, P_Q\}$, where $Q \ll L$. The structural diagram of CMCU U_2 based on this principle is shown in Fig. 2.

In CMCU U_2 , codes $K_A(B_i)$ of the classes $B_i \in \Pi_A$ are represented by variables $T_r \in T$, whereas codes $K_B(B_i)$ of the classes $B_i \in \Pi_B$ by variables $v_r \in V$, where $|V| = R_2$. In contrast to CMCU U_1 , there is no block BAT, and the block BMA implements functions

$$\Phi = \Phi(T, V, P). \quad (11)$$

Variables $p_q \in P$ are generated by a block of logical conditions (BLC) as the following system

$$P = P(T, V, X). \quad (12)$$

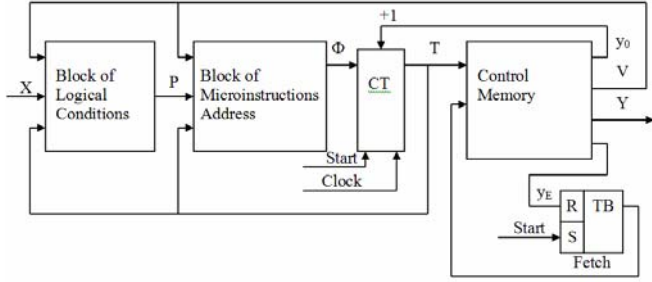


Fig. 2. Structural diagram of CMCU U_2

Let the symbol $U_i(\Gamma_j)$ mean that CMCU U_1 interprets Γ_j , and symbol $Q_i(\Gamma_j)$ determine the number of macrocells in the logic circuit of BMA for CMCU $U_i(\Gamma_j)$, where $i=1,2$. Let each macrocell have S inputs, where G_q variables $p_q \in P$ are connected with inputs of the macrocell number q . The proposed method can be applied if the following condition

$$Q_q + R + R_2 \leq S \quad (13)$$

takes place, where $q=1, \dots, Q_1(\Gamma_j)$. If condition (13) is violated, the number $Q_2(\Gamma_j)$ exceeds tremendously the number $Q_1(\Gamma_j)$.

The following method is proposed in our article for synthesis of CMCU U_2 :

1. Constructions of sets C , C_1 , and Π_C for GSA Γ .
2. Microinstruction addressing.
3. Constructions of sets Π_A and Π_B .
4. Encoding of classes $B_i \in \Pi_B$.
5. Construction of control memory content.
6. Replacement of logical conditions.
7. Construction of CMCU transition table.
8. Specification of block BLC.
7. Implementation of CMCU logic circuit.

IV. EXAMPLE OF APPLICATION OF PROPOSED METHOD

Let the sets $C = \{\alpha_1, \dots, \alpha_9\}$, $C_1 = \{\alpha_1, \dots, \alpha_8\}$ and $\Pi_C = \{B_1, \dots, B_5\}$ be formed for a GSA Γ_1 , where $\alpha_1 = \langle b_1, b_2 \rangle$, $\alpha_2 = \langle b_3, \dots, b_6 \rangle$, $\alpha_3 = \langle b_7, b_8 \rangle$, $\alpha_4 = \langle b_5, \dots, b_{13} \rangle$, $\alpha_5 = \langle b_4, \dots, b_{17} \rangle$, $\alpha_6 = \langle b_{18}, \dots, b_{21} \rangle$, $\alpha_7 = \langle b_{22}, \dots, b_{25} \rangle$, $\alpha_8 = \langle b_{26}, \dots, b_{28} \rangle$, $\alpha_9 = \langle b_{29}, \dots, b_{31} \rangle$, $B_1 = \{\alpha_1\}$, $B_2 = \{\alpha_2, \alpha_3\}$, $B_3 = \{\alpha_4, \alpha_5\}$, $B_4 = \{\alpha_6, \alpha_7\}$, $B_5 = \{\alpha_8\}$. Thus, $I = 5$, $R_1 = 3$, $\tau = \{\tau_1, \tau_2, \tau_3\}$, $M = 31$, $R = 5$.

Let us address the microinstructions using some modification of the algorithm from [4]. Now we have $A(b_1) = 00000, \dots, A(b_{25}) = 110000, A(b_{26}) = 11100, \dots, A(b_{28}) = 11110, A(b_{29}) = 11001, \dots, A(b_{31}) = 11011$. Let us construct the Karnaugh map marked by the variables

$T_r \in T = \{T_1, \dots, T_5\}$ (Fig. 3). This map contains outputs of OLC $\alpha_g \in C$ and code space intervals corresponding to the classes $B_i \in \Pi_C$.

The sign * in this map stands for the case when a vertex $b_q \in E_1$ with address $A(b_q)$ is not the output of OLC $\alpha_g \in C_1$. The following code intervals can be derived from Fig. 3: the class B_1 corresponds to interval 0000*, the class B_2 to 001**, the class B_3 to 01*** and 10000, the class B_4 to 101** and 11000, the class B_5 to 111**. Let us point out that $\alpha_9 \notin C_1$ and the class $B_6 = \{\alpha_9\}$ is not considered here.

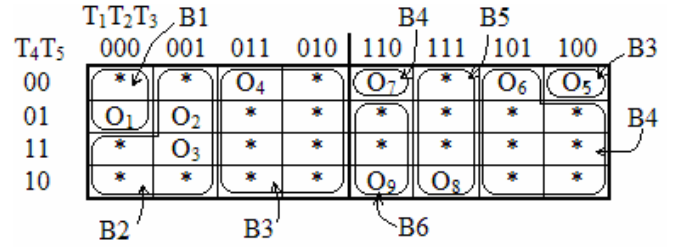


Fig. 3. Karnaugh map for outputs of OLC

The obtained intervals determine the sets $\Pi_A = \{B_1, B_2, B_5\}$ and $\Pi_B = \{B_3, B_4\}$. Let $N=12$ for the GSA Γ_1 and $t=4$. In this case we have $R_3=2$, condition (10) takes place, because $R_2 = \lceil \log_2(2+1) \rceil = R_3$. Therefore, the model of CMCU $U_2(\Gamma_1)$ can be applied and the block BAT is absent.

Let us encode the classes $B_i \in \Pi_B$ in the following way: $K_B(B_3) = 01$, $K_B(B_4) = 10$. Now the code 00 corresponds to the case, when $B_i \in \Pi_A$. The code 11 can be used for optimization of other codes. Finally we get $K_B(B_3) = *1$ and $K_B(B_4) = 1*$. Besides, the following codes can be derived from the Karnaugh map shown in Fig. 3: $K_A(B_1) = 0000*$, $K_A(B_2) = 001**$, $K_A(B_5) = 111**$.

The following procedure is proposed for construction of the control memory content, which can be viewed as some modification of the known method [4]:

1. $q=1$.
2. If $b_q \in E_1$, then the memory cell with address $A(b_q)$ contains $Y(b_q)$. Otherwise, go to point 6.
3. If b_q is not the output of OLC $\alpha_g \in C$, then the memory cell with address $A(b_q)$ contains y_0 .
4. If b_q is the output of OLC $\alpha_g \notin C_1$, then the memory cell with address $A(b_q)$ contains y_E .
5. If b_q is the output of OLC $\alpha_g \in C_1$, where $a_g \in B_i$, then the memory cell $A(b_q)$ contains code $K_B(B_i)$.
6. If all vertices of GSA Γ are analyzed, then go to step

7. Otherwise, $q := q + 1$, go to point 2.

7. End.

There are no problems in this procedure application. So, this step in our article is omitted.

Let the transitions for classes $B_i \in \Pi_C$ be specified by the following system of generalized formulae of transitions (GFT) [4]:

$$\begin{aligned} B_1 &\rightarrow x_1 b_3 \vee \overline{x_1 x_4 b_5} \vee \overline{x_1 x_4 b_7}; \\ B_2 &\rightarrow x_3 b_9 \vee \overline{x_3 b_{26}}; \\ B_3 &\rightarrow x_1 b_{18} \vee \overline{x_1 x_2 b_{20}} \vee \overline{x_1 x_2 b_{26}}; \\ B_4 &\rightarrow x_5 b_{27} \vee \overline{x_5 b_5}; \\ B_5 &\rightarrow x_6 b_{24} \vee \overline{x_6 x_7 b_{29}} \vee \overline{x_6 x_7 b_{19}}. \end{aligned} \quad (14)$$

A GFT is a some modification of transition formulae using for finite-state-machines (FSM) [1]. The fact that OLCs replace FSM states is taken into account. Because all transitions are the same for OLC $\alpha_g \in B_i$, then the classes

$B_i \in \Pi_C$ are written into GFT. The expression ‘‘GFT’’ underlines the fact of OLC outputs replacement by corresponding classes. Obviously, the transition for class $B_i \notin \Pi_C$ are not considered, because CMCU operation is terminated after generation of the variable y_E .

Let $X(B_i)$ be a set of logical conditions determining transitions from the class $B_i \in \Pi_C$, where $|X(B_i)| = Q_i$. Let $Q = \max(Q_i | B_i \in \Pi_C)$, then logical conditions $x_i \in X$ can be replaced by Q elements of the set P .

Using system (14), the following sets can be obtained: $X(B_1) = \{x_1, x_4\}$, $X(B_2) = \{x_3\}$, $X(B_3) = \{x_1, x_2\}$, $X(B_4) = \{x_5\}$, $X(B_5) = \{x_6, x_7\}$, $Q_1 = Q_3 = Q_5 = 2$, $Q_2 = Q_4 = 1$, $Q = 2$. Now we have the set $P = \{p_1, p_2\}$. Let us form a table for logical condition replacement having columns $B_i \in \Pi_C$ and rows $p_q \in P$. If logic condition $x_i \in X$ is replaced by variable $p_q \in P$ for class $B_i \in \Pi_C$, then the symbol x_i is written on intersection of the row p_q and column B_i . The replacement is executed in a way minimizing appearance of the same variable x_i in the different rows of the table (Table I).

TABLE I

LOGICAL CONDITION REPLACEMENT FOR CMCU $U_2(\Gamma_1)$

B_i	B_1	B_2	B_3	B_4	B_5
p_1	x_1	x_3	x_1	–	x_6
p_2	x_4	–	x_2	x_5	x_7

Let us transform the system of GFT by replacement of conditions $x_i \in X$ by variables $p_q \in P$ (using Table I in our case):

$$\begin{aligned} B_1 &\rightarrow p_1 b_3 \vee \overline{p_1 p_2 b_5} \vee \overline{p_1 p_2 b_7}; \\ B_2 &\rightarrow p_1 b_9 \vee \overline{p_1 b_{26}}; \\ B_3 &\rightarrow p_1 b_{18} \vee \overline{p_1 p_2 b_{20}} \vee \overline{p_1 p_2 b_{26}}; \\ B_4 &\rightarrow p_2 b_{27} \vee \overline{p_2 b_5}; \\ B_5 &\rightarrow p_1 b_{24} \vee \overline{p_1 p_2 b_{29}} \vee \overline{p_1 p_2 b_{19}}. \end{aligned} \quad (15)$$

Such a system is used to construct the CMCU U_2 transition table (Table II) having columns $B_i, K(B_i), b_q, A(b_q), P_h, \Phi_h, h$. The system (15) is used to construct such a table for the CMCU $U_2(\Gamma_1)$. The table includes $H = 13$ lines, it is determined by the number of terms in system (15). In this table there are two columns, $K_A(B_i)$ and $K_B(B_i)$, to represent the codes corresponding to the classes Π_A and Π_B .

TABLE II
TABLE OF TRANSITIONS FOR CMCU $U_2(\Gamma_1)$

B_i	$K_A(B_i)$	$K_B(B_i)$	b_q	$A(b_q)$	P_h	Φ_h	h
			b_3	00010	p_1	D_4	1
B_1	0000*	00	b_5	00100	$\overline{p_1 p_2}$	D_3	2
			b_7	00110	$\overline{p_1 p_2}$	$D_3 D_4$	3
			b_9	01000	p_1	D_2	4
B_2	001**	00	b_{26}	11100	$\overline{p_1}$	$D_1 D_2$ D_3	5
			b_{18}	10001	p_1	$D_1 D_5$	6
B_3	*****	*0	b_{20}	10011	$\overline{p_1 p_2}$	$D_1 D_4$ D_5	7
			b_{26}	11100	$\overline{p_1 p_2}$	$D_1 D_2$ D_3	8
B_4	*****	1*	b_{27}	11101	p_2	$D_1 D_2$ $D_3 D_5$	9
			b_5	00100	$\overline{p_2}$	D_3	10
			b_{24}	10111	p_1	$D_1 D_3$ $D_4 D_5$	11
B_5	111**	00	b_{29}	11001	$\overline{p_1 p_2}$	$D_1 D_2$ D_5	12
			b_{19}	10010	$\overline{p_1 p_2}$	$D_1 D_4$	13

The table of transitions is used to derive equations (11), having the following terms:

$$F_h = \left(\bigwedge_{r=1}^R T_r^{l_{rh}} \right) * \left(\bigwedge_{r=1}^{R_2} V_r^{E_{rh}} \right) * P_h. \quad (16)$$

In (16), the symbol $l_{rh} \in \{0, 1, *\}$ stands for the value of bit r of the code $K_A(B_i)$ from the line h of the table, where $T_r^0 = \overline{T_r}, T_r^1 = T_r, T_r^* = 1$ ($r = 1, \dots, R$). The symbol $E_{rh} \in \{0, 1, *\}$ stands for the value of bit r of the code $K_B(B_i)$ from the line h of the table, where $V_r^0 = \overline{V_r}, V_r^1 = V_r, V_r^* = 1$ ($r = 1, \dots, R_2$). In both cases we have $h = 1, \dots, H$. For example, the following sum-of-the-products (SOP) can be derived from Table II:

$$D_1 = F_5 \vee \dots \vee F_9 \vee F_{11} \vee F_{12} \vee F_{13} = \\ = \overline{T_1 T_2 T_3 V_1 V_2 P_1} \vee V_2 \vee V_1 P_2 \vee T_1 T_2 T_3 \overline{V_1 V_2} \quad (\text{after minimizing of initial SOP}).$$

The block BLC is specified by its table having the following columns: $B_i, K_A(B_i), K_B(B_i), P_1, \dots, P_Q, i$ (Table III). The table is constructed using the initial table of logical replacement (see Table I).

TABLE III
SPECIFICATION OF BLOCK BLC FOR CMCU $U_2(\Gamma_1)$

B_i	$K_A(B_i)$	$K_B(B_i)$	p_1	p_2	i
B_1	0000*	00	x_1	x_4	1
B_2	001**	00	x_3	–	2
B_3	*****	*1	x_1	x_2	3
B_4	*****	1*	–	x_5	4
B_5	111**	00	x_6	x_7	5

This table is used to derive system (12). For example, the following equation can be derived from Table III:

$$P_1 = \overline{T_1 T_2 T_3 T_4 V_1 V_2 x_1} \vee \overline{T_1 T_2 T_3 V_1 V_2 x_3} \vee V_2 x_1 \vee T_1 T_2 T_3 V_1 V_2 x_6.$$

Implementation of CMCU U_2 logic circuit is reduced to the implementation of systems (12)-(13) with PAL macrocells and control memory with PROM chips. There are many effective methods for solution of these tasks [3], because of it we do not discuss this step in our article.

V. CONCLUSION

The proposed method is oriented on hardware amount decrease in the logic circuit of microinstruction address transformer, which is the part of CMCU. This task solution is based on use of more than one source of codes for classes of pseudoequivalent OLC. As a limit, three sources can be used. Optimization of the logic circuit of block of microinstruction addressing is reached due to usage of the know method of logical condition replacement. It allows decrease for the number of required inputs of PAL macrocells. It gives an additional possibility in use of these free inputs for receiving of variables used for OLC classes encoding.

Unfortunately, the gain in hardware is accompanied by decrease of CMCU performance because of the propagation time of additional block BLC. Besides, this block consumes some recourses of the chip. Therefore, the proposed method can be applied if total hardware amount for BMA and BLC is less than hardware amount for BMA of equivalent CMCU U_1 .

The scientific novelty of proposed method is determined by simultaneous use the PLA macrocells wide fan-in and logical condition replacement. It leads to hardware amount decrease for blocks BMA and BAT. If condition (10) takes place, then the block BAT is absent. The practical significance of this method is determined in decrease for the number of macrocells in CMCU logic circuit. It allows getting logic circuits with less hardware than for control units known from literature. Our investigation shows that

the number of macrocells is decreased up to 10% for CMCU $U_2(\Gamma_j)$ in comparison with equivalent CMCU $U_1(\Gamma_j)$.

There are two directions of our further research. The first, we should develop an algorithm permitting decrease for the number of OLC with outputs addresses to be transformed. The second, we should try this approach for CPLD based on programmable logic arrays [9], as well as on FPGA [10].

REFERENCES

- [1] *Baranov S.* Logic Synthesis for Control Automata. Kluwer Academic Publishers, 1994. 312 pp.
- [2] *Grushvitskiy R.I., Mursaev A.H., Ugrumov E.P.* System's Design by Using Programmable Logic Microchips – St. Petersburg: BHV, 2002. 608 p. (in Russian).
- [3] *Solov'ev V.V.* Digital System Design based on Programmable Logic Integral Schemas – oscow: Hot Line-TELEKOM, 2001. 636 p. (in Russian).
- [4] *Barkalov A., Titarenko L.* Logic Synthesis for Compositional Microprogram Control Units. – Berlin: Springer, 2008. 272 p.
- [5] Altera devices overview. http://www.altera.com/products/devices/common/dev-family_overview.html.
- [6] Xilinx CPLDs http://www.xilinx.com/products/silicon_solutions/cplds/index.htm.
- [7] *Barkalov A.A., Zeleneva I.Y., Lavrik A.S.* Using PLIS peculiarities for control device schema optimization / Scinse Trans. of Donetsk Nationality Technical University. Series «Informatics, Cybernetics and Calculation Techniques». Issue 9 (132) – Donetsk: DonNTU. – 2008. P. 178-182. (in Russian).
- [8] *Barkalov A.A., v lev S. ., Krasichkov . ., Lavrik .S.* Control Devise Optimization with transformation of microcommands address / Proc. of IX International Semnar.– Taganrog. Book. 3. 2008. P. 12-20.
- [9] CoolRunner CPLD Datasheet. <http://www.xilinx.com/support/documentation/coolrunner-ii.htm>
- [10] *Maxfield C.* The Design Warrior's Guide to FPGAs. – Amsterdam: Elsevier, 2004. – 541 pp.



Alexandr A. Barkalov – Doctor of Science, Professor of the Donetsk National Technical University (Ukraine), Professor of the Uniwersytet Zielonogórski (Poland).
Scientific interests: digital control units, SoPC
Address: Campus A, Budynek Dydaktyczny / A-2
prof. Z. Szafrana str. 2, 65-516 Zielona Góra
E-mail: A.Barkalov@iie.uz.zgora.pl



Larisa A. Titarenko – Doctor of Science, Professor of the Kharkov National University of Radioelectronics (Ukraine), Professor of the Uniwersytet Zielonogórski (Poland). Scientific interests: digital control units, telecommunication systems. Address: Campus A, Budynek Dydaktyczny / A-2 prof. Z. Szafrana str. 2, 65-516 Zielona Góra
E-mail: L.Titarenko@iie.uz.zgora.pl



Alexandr S. Lavrik – Post graduate Student of the Donetsk National Technical University.

Numerical Simulation of Charge Diffusion on the Surface of a Dendrimer Molecule

Artyom V. Andreyev, Oleksiy V. Klymenko

Abstract — This work presents the results of numerical simulation of charge transfer within the spherical outer shell of a dendrimer macromolecule induced by electrode polarisation on which the molecule is adsorbed. It is shown that under linear variation of the electrode potential the electric current peak contains information about the regimes of charge diffusion in the molecular shell. Thus the obtained results can be used for the determination of kinetic parameters from experimental data.

Index Terms — numerical simulation, surface diffusion, dendrimer, electrochemistry

I. INTRODUCTION

THE study of intramolecular processes is of extremely high interest both from the fundamental and practical points of view bearing in mind potential future applications in microelectronics and data storage devices. This area of research has been developing for several decades and yielded important experimental and theoretical results [1-3]. However, experimental investigation of such processes is exceptionally complicated and expensive owing to very high rates of such processes that require application of equipment with sufficiently high time resolution (of the order of nanoseconds and below). As was recently shown [4-7], intramolecular electron transfer within adsorbed large molecules can be successfully investigated by electrochemical methods. The latter works report experimental and theoretical studies of the process of electron transfer between active groups containing ruthenium atoms and located within the outer shell of a spherical dendrimer molecule adsorbed on the surface of a platinum working electrode. The application of negative voltage to the platinum working electrode with adsorbed dendrimers leads to the reduction of ruthenium atoms within active centres in the immediate vicinity of the platinum surface. It has been proposed that, following this initial reaction, the active centres within the dendrimer

outer shell begin to exchange charge between Ru(II) and Ru(III) atoms via electron tunnelling ('hopping'). The overall charge transfer in this case is reminiscent of diffusion of electrons within the outer dendrimer shell.

In order to understand the kinetics of this process we perform here the numerical simulation of electric current responses under the conditions of linear sweep voltammetry for different values of kinetic parameters which facilitates the analysis of experimental data.

II. MODEL

An adsorbed on the electrode surface dendrimer molecule is schematically shown in Fig. 1, where θ_0 is the characteristic angle reflecting the degree of dendrimer molecule deformation upon its adsorption and r_0 is the apparent radius of the adsorbed molecule. Electrode polarization enforces electron transfer (ET) between the electrode and active centres groups of the dendrimer shell located in the immediate vicinity of the electrode surface. Following the reduction of these sites the electron transfer occurs between neighbouring sites within the surface layer of the dendrimer molecule. i.e. via "hopping" diffusion of electrons. On average, due to the random character of electron transfer reactions between neighbouring sites (and, in general, a significant number of adsorbed molecules performing this in parallel), this process may be described as continuous diffusion of electrons with a diffusion coefficient D within the thin molecular shell.

The diffusion flux of electrons is then given by the first Fick's law:

$$\vec{j} = -D \text{grad } c, \quad (1)$$

where c is the surface concentration of transferred electrons in the dendrimer outer shell (in mol cm^{-2}) or, equivalently, surface concentration of reduced active sites. For the sake of simplicity we neglect here the thickness of the dendrimer shell containing active centres.

Due to the axial symmetry of the system the diffusion laws may be formulated in one of the axial cross-sections of the molecule using the spherical coordinate system centred at the dendrimer centre. Thus the flux along the meridian of the dendrimer is represented as:

Manuscript received July 15, 2009.

A. V. Andreyev is a 4th year undergraduate student at Kharkov National University of Radioelectronics, 14 Lenin Ave., Kharkov, 61166, Ukraine.

O. V. Klymenko is with the Mathematical and Computer Modelling Laboratory, Kharkov National University of Radioelectronics, 14 Lenin Ave., Kharkov, 61166, Ukraine (phone: +38 057 702 09 69; fax: +38 057 702 10 13; e-mail: klymenko@kture.kharkov.ua).

$$j = -D \frac{\partial c}{\partial l} = -\frac{D}{r_0} \frac{\partial c}{\partial \theta}. \quad (2)$$

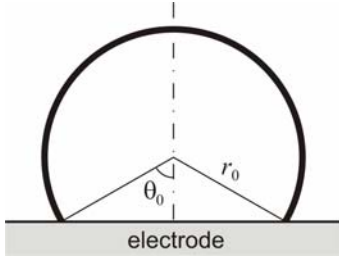


Fig.1. Scheme of a dendrimer molecule adsorbed on electrode surface

On the other hand, the charge conservation law must be obeyed, which in this case is given by the continuity relation leading to the second Fick's law:

$$\frac{\partial c}{\partial t} = -\text{div} \vec{j} = \frac{D}{r_0^2} \left(\text{ctg} \theta \frac{\partial c}{\partial \theta} + \frac{\partial^2 c}{\partial \theta^2} \right). \quad (3)$$

The initial condition ($t = 0$) to the diffusion equation (3) is given by

$$c(0, \theta) = 0. \quad (4)$$

The boundary conditions reflect the symmetry of the system, i.e. when $\theta = \pi$

$$\left. \frac{\partial c}{\partial \theta} \right|_{\theta=\pi} = 0, \quad (5)$$

and kinetics of electron transfer from the electrode into the dendrimer molecule:

$$-D \left. \frac{\partial c}{\partial l} \right|_{l=\theta_0} = -k_b c + k_f (c^0 - c), \quad (6)$$

where k_b and k_f are the rates of forward and reverse reactions (viz. reduction of active centres in the immediate vicinity of the electrode surface and their oxidation, respectively), and c^0 is the surface concentration of active sites within the dendrimer outer shell. ET rates are given by the Butler-Volmer theory as [8]:

$$k_f = k_0 \exp\left(-\frac{\alpha F}{RT} (E - E_0)\right); \quad (7)$$

$$k_b = k_0 \exp\left(\frac{(1-\alpha)F}{RT} (E - E_0)\right), \quad (7b)$$

where k_0 is the standard rate constant; E is the electrode potential; E_0 is the formal redox potential; α is the transfer coefficient; F is the Faraday constant; R is the gas constant; T is the absolute temperature. The physical meaning of k_0 in this case is somewhat different from the classical heterogeneous rate constant in electrochemistry (see a thorough discussion of this point in [5]) but this is not important for the model considered here.

The electric current flowing in the system is given by:

$$i = \frac{dN}{dt} F, \quad (8)$$

where N is the number of electrons (in moles) transferred onto the dendrimer molecule(s). In the case of a single molecule one has:

$$\frac{dN}{dt} = j 2\pi r_0 \sin \theta_0. \quad (9)$$

Substitution of (2) and (9) into (8) yields the following expression for the electric current:

$$i = F j 2\pi r_0 \sin \theta_0 = -2\pi D F \sin \theta_0 \left. \frac{\partial c}{\partial \theta} \right|_{\theta=\theta_0} = -2\pi D F c^0 \sin \theta_0 \frac{\partial u}{\partial \theta} \quad (10)$$

A. Dimensionless model

The introduction of dimensionless time and concentration as

$$\tau = \frac{Dt}{r_0^2}; \quad u = \frac{c}{c^0}, \quad (11)$$

allows obtaining the following dimensionless diffusion equation

$$\frac{\partial u}{\partial \tau} = \text{ctg} \theta \frac{\partial u}{\partial \theta} + \frac{\partial^2 u}{\partial \theta^2}. \quad (12)$$

The initial condition (4) becomes:

$$u(0, \theta) = 0. \quad (13)$$

The boundary conditions (5) and (6) take the following forms, respectively:

$$\left. \frac{\partial u}{\partial \theta} \right|_{\theta=\pi} = 0; \quad (14)$$

$$\left. \frac{\partial u}{\partial \theta} \right|_{\theta=\theta_0} = (K_f + K_b) u - K_f, \quad (15)$$

where

$$K_f = \exp(-\alpha \varepsilon); \quad (16a)$$

$$K_b = \exp((1-\alpha)\varepsilon), \quad (16b)$$

are the dimensionless forward and reverse rate constants with $K_0 = \frac{r_0 k_0}{D}$ being the normalized standard rate

constant and $\varepsilon = \frac{F}{RT} (E - E^0)$ the dimensionless electric potential.

In the case of linear sweep (or cyclic) voltammetry the electrode potential varies according to the equation:

$$E = E_{st} + vt, \quad (17)$$

where E_{st} is the initial potential and v is the voltage scan rate. Introducing the corresponding dimensionless parameters as

$$\varepsilon_{st} = \frac{F}{RT} (E_{st} - E_0); \quad \sigma = \frac{F}{RT} \frac{r_0^2}{D} v \quad (18)$$

results in the following dimensionless expression for the

potential variation:

$$\varepsilon = \varepsilon_{st} + \sigma\tau. \quad (19)$$

Finally, the dimensionless current for a single adsorbed dendrimer molecule is defined as

$$f = \frac{i}{2\pi D F c^0} = -\sin\theta_0 \left. \frac{\partial u}{\partial \theta} \right|_{\theta=\theta_0}. \quad (20)$$

III. NUMERICAL SIMULATIONS

The diffusion problem (14)-(17) was solved using the fully implicit finite difference scheme [9] with second order approximation of first and second derivatives in θ and first order approximation in time. The resulting linear tridiagonal systems were solved by the Thomas algorithm [10]. It should be noted that the cotangent function being a factor at the first derivative in θ tends to $-\infty$ as $\theta \rightarrow \pi$. However, this fact does not spoil the convergence of the linear solver due to off-diagonal dominance. Indeed, consider the employed finite difference scheme:

$$-\frac{\Delta\theta^2}{\Delta\tau} u_j^n = \left(1 - \frac{\Delta\theta}{2} \cot\theta_j\right) u_{j-1}^{n+1} + \left(2 + \frac{\Delta\theta^2}{\Delta\tau}\right) u_j^{n+1} + \left(1 + \frac{\Delta\theta}{2} \cot\theta_j\right) u_{j+1}^{n+1}, \quad (21)$$

where u_i^k is the value of the dimensionless concentration at grid node with the coordinates $\theta_i = \theta_0 + i\Delta\theta$ and $\tau_k = k\Delta\tau$ where $\Delta\theta$ and $\Delta\tau$ are the grid step sizes in angle and time. Note that in the limit $\theta \rightarrow \pi$ the cotangent function asymptotically tends to $-1/(\pi - \theta)$, so that in this limit (to be more precise, when the finite difference scheme corresponds to the penultimate grid point at $\theta = \pi - \Delta\theta$) the latter equation reduces to

$$-\frac{\Delta\theta^2}{\Delta\tau} u_j^n = 1.5u_{j-1}^{n+1} + \left(2 + \frac{\Delta\theta^2}{\Delta\tau}\right) u_j^{n+1} + 0.5u_{j+1}^{n+1}, \quad (22)$$

and that diagonal dominance is strictly preserved since $2 + \frac{\Delta\theta^2}{\Delta\tau} > 1.5 + 0.5$ despite apparently infinite coefficient in the original differential equation.

Numerical convergence tests have shown that the relative error in the numerically computed electric current (20) remains under 0.1% for the considered ranges of parameter values for the grid size $N_\theta \times N_\tau = 1000 \times 4000$. The program for the numerical simulation was written in Borland C++ Builder 6 and executed on a PC based on Intel Pentium 4 processor at 3GHz with 512MB of RAM.

IV. RESULTS

The analysis of the dimensionless model reveals that the current response f of the dendrimer to electrode polarisation depends on three parameters being the

characteristic adsorptive deformation angle θ_0 , the dimensionless heterogeneous rate constant K_0 and the dimensionless voltage scan rate σ . It is clear that the above parameters determine the diffusion regime in the dendrimer shell. Most notably, the finite number of active centres borne by one dendrimer macromolecule implies that it also has finite capacity of electron uptake. This is reflected, under sufficiently slow voltage scan rates, in a bell-shaped current response with the current decaying to zero after all the active centres of the dendrimer have been reduced (see Fig. 2a). Such a voltammetric shape is typical for adsorbed species or thin layer cells [8] where the time constant of reaction or mass transport is negligible versus the time constant of the excitation potential variation. On the other hand, when the latter time constant is small compared to the characteristic diffusion time (i.e. the voltage scan rate is high) the charge transfer occurs as though in an infinite space (i.e. the diffusion layer thickness is significantly smaller than the length of the dendrimer meridian $l = r_0(\pi - \theta_0)$) with the typical voltammetric waveshape shown in Fig. 2b. Accordingly, there are two main limiting regimes which are clearly observed in the dependence of the dimensionless peak current on the scan rate σ and the heterogeneous rate constant K_0 presented in Fig. 3 for a fixed value of the angle $\theta_0 = 0.1$.

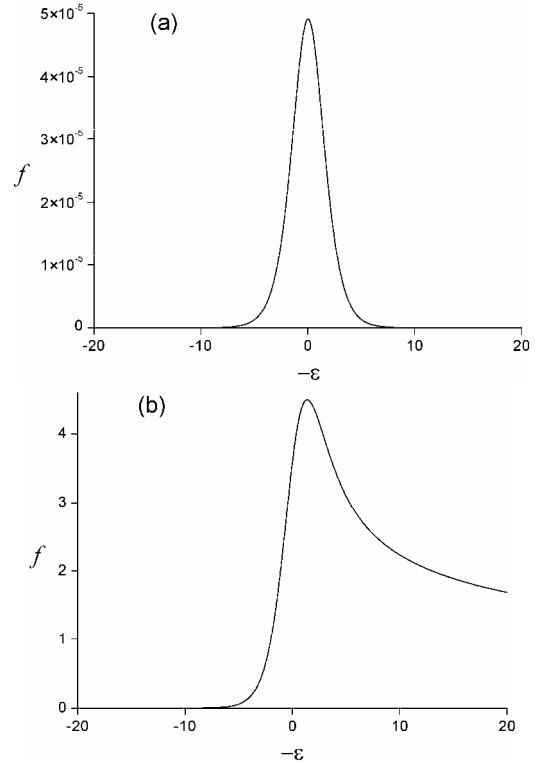


Fig. 2. Simulated linear sweep voltammograms for $K_0 = 10^6$, $\theta_0 = 0.1$ and scan rates (a) $\sigma = 10^{-4}$ and (b) $\sigma = 10^4$

The first regime, characteristic of adsorbed species or thin layer cells, corresponds to low values of σ when the peak current is proportional to the dimensionless scan rate [8]. The peak current under the second regime of unconstrained diffusion encountered under high scan rates is proportional to the square root of the scan rate [8]. This is exemplified in Fig. 4a which shows the cross-section of the surface in Fig. 3 along the $\log \sigma$ axis. Thus the slope of this curve in the range $\log \sigma < -2.5$ is equal to unity and it is $1/2$ for $\log \sigma > 2$ with a smooth transition from one limiting regime to the other.

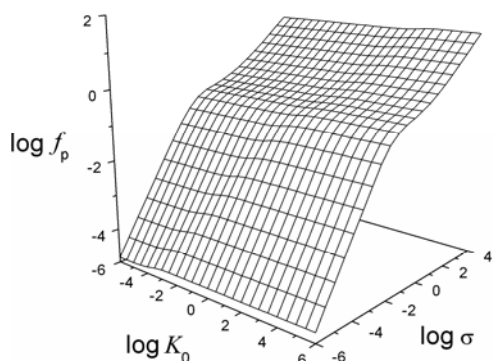
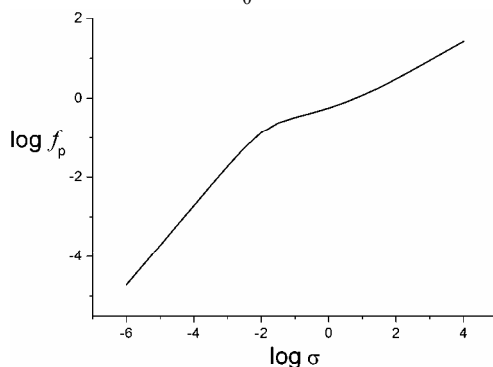
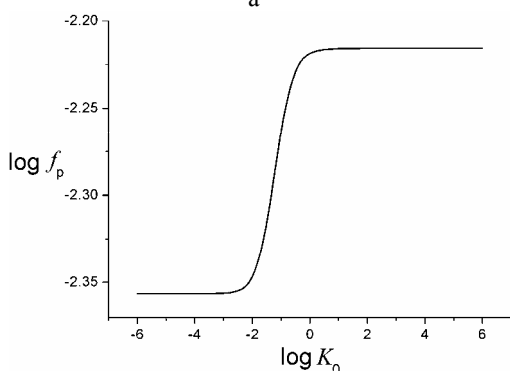


Fig. 3. Dependence of peak current on heterogeneous kinetics and scan rate for $\theta_0 = 0.1$



a



b

Fig. 4. Cross-section of surface in Fig. 3 along the line $\log K_0 = 6$ showing two different limiting regimes: "thin layer" regime for low values of σ (slope=1) and unconstrained diffusion for high values of σ (slope=1/2) – (a); cross-section of surface in (a) along the line $\log \sigma = -3.5$ showing variation between reversible and irreversible behaviour – (b)

The variation of the peak current f_p with the normalised heterogeneous rate constant is less pronounced on the scale of Fig. 3, but nevertheless it reveals the reversible ($K_0 \rightarrow \infty$) and irreversible ($K_0 \rightarrow 0$) kinetic limits for high and low values of K_0 , respectively. This is illustrated in Fig. 4b which depicts the cross-section of the $\log f_p$ surface from Fig. 3 along the line $\log \sigma = -3.5$ revealing the sigmoidal transition between the two limiting regimes. It may be observed in Fig. 3 that the transition between the irreversible and reversible limits shifts towards higher values of K_0 with increasing scan rate. This is again conditioned by the relative magnitudes of the kinetic time constant determined by K_0 and the time allocated to mass transport through the dimensionless voltage scan rate σ .

In a real experimental situation the only readily measurable parameter is the apparent size of the dendrimer molecule r_0 . However, even the determination of this parameter requires the usage of sophisticated instrumentation such as scanning tunnelling microscope. On the other hand, the actual shape of the adsorbed molecule is unknown, which in terms of the assumptions made here implies that the angle θ_0 characterising the extent of dendrimer deformation is unknown. Nonetheless this information may be accessed through electrochemical measurements over a range of voltammetric scan rates followed by a theoretical analysis of the shapes and magnitudes of recorded electric currents. To this end the working surface presented in Fig. 5 provides a means to analyse the peak current data as function of θ_0 and σ . Note here that the values of θ_0 close to π are included for completeness of the results but they correspond to a physically unrealistic situation. In fact, high values of θ_0 represent a virtually flattened molecule which means a complete loss of shape and structure. Hence the current model is not applicable under such extreme conditions. However, the values of θ_0 below ca. $2\pi/3$ seem reasonable and thus the respective part of the $\log f_p$ surface in Fig. 5 may be used for the analysis of experimental data.

As in the results presented in Fig. 3 the peak current dependence on σ has two distinct limits the first of which corresponds to the reduction of all active centres within the dendrimer outer shell while the second limit represents unconstrained diffusion. In the first limit, the variation of $\log f_p$ with θ_0 for a fixed value of σ is a decaying function which is explained simply by the reduction of the exposed surface area of the dendrimer and hence the fewer active centres, whose number is proportional to $(1 + \cos \theta_0)\sigma$. In the second limiting situation (i.e. under unconstrained diffusion conditions) the variation of $\log f_p$

with θ_0 for a fixed value of σ is not monotonic. Indeed, since in this case the dimensionless diffusion layer thickness (relative to the dendrimer radius) is much smaller than $\sin\theta_0$ (see above) the overall number of active sites that have been reduced during the voltammetric scan (and the peak current f_p) for very high values of σ is proportional to $\sin\theta_0\sigma^{-1/2}$. This is clearly observable from the computed results presented in Fig. 5.

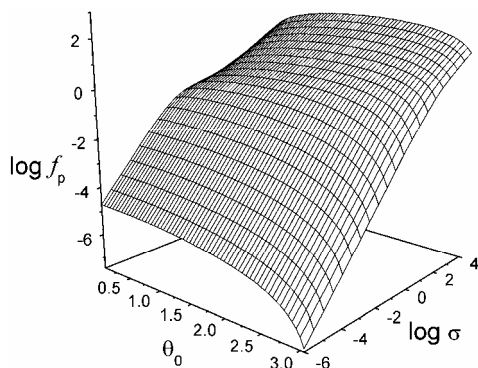


Fig. 5. Dependence of dimensionless peak current on scan rate σ and angle θ_0 characterizing dendrimer deformation upon adsorption on electrode surface in the limit $K_0 \rightarrow \infty$

V. CONCLUSION

The numerical simulation of diffusion within the thin spherical outer shell of a dendrimer molecule has yielded general results describing the behaviour of the peak current under linear sweep voltammetry. The limiting kinetic regimes identified in this work may be readily used for determining the predominant diffusional regime within the molecular shell from experimental data. Nevertheless, a more accurate determination of the values of main kinetic parameters may require a full curve fitting procedure employing the current model.

The physicochemical model considered in this work represents, in fact, a more general paradigm of two-dimensional diffusion processes occurring on curved surfaces. Subsequently this paradigm has been extended for a more complicated system involving cluster formation and growth on a spherical surface leading to moving boundary of the diffusion domain and its application to a real biological experiment [11, 12]. This confirms the importance of the present work for different areas of modern science.

REFERENCES

- [1] L. T. Calcaterra, G. L. Gloss, J. R. Miller, "Fast intramolecular electron transfer in radical ions over long distances across rigid saturated hydrocarbon spacers," *J. Am. Chem. Soc.*, vol. 105, pp. 670-671, 1983.
- [2] V. Chechik, R. M. Crooks, C. J. M. Stirling, "Reactions and reactivity in self-assembled monolayers," *Advanced Materials*, vol. 12, pp. 1161-1171, 1983.
- [3] E.J. Land, D. Lexa, R.V. Bensasson, D. Gust, T.A. Moore, A.L. Moore, P.A. Liddell, G.A. Nemeth, "Pulse radiolytic and electrochemical investigations of intramolecular electron transfer in carotenoporphyrins and carotenoporphyrin-quinone triads," *J. Phys. Chem.*, vol. 91, pp. 4831-4835, 1987.
- [4] C. Amatore, Y. Bouret, E. Maisonhaute, J.I. Goldsmith, H.D. Abruña, "Ultrafast voltammetry of adsorbed redox active dendrimers with nanometric resolution: an electrochemical microtome," *ChemPhysChem*, vol. 2, pp. 130-134, 2001.
- [5] C. Amatore, Y. Bouret, E. Maisonhaute, J.I. Goldsmith, H.D. Abruña, "Precise adjustment of nanometric-scale diffusion layers within a redox dendrimer molecule by ultrafast cyclic voltammetry: an electrochemical nanometric microtome," *Chem. Eur. J.*, vol. 7, pp. 2206-2226, 2001.
- [6] C. Amatore, Y. Bouret, E. Maisonhaute, H.D. Abruña, J.I. Goldsmith, "Electrochemistry within molecules using ultrafast cyclic voltammetry," *C. R. Chim.*, vol. 6, pp. 99-115, 2003.
- [7] C. Amatore, F. Grün, E. Maisonhaute, "Electrochemistry within a limited number of molecules: delineating the fringe between stochastic and statistical behavior," *Angew. Chem.*, vol. 42, pp. 4944-4947, 2003.
- [8] A. J. Bard, L.R. Faulkner, *Electrochemical Methods: Fundamentals and Applications*. 2nd edition, John Wiley & Sons, 2001.
- [9] R.D. Richtmyer, K.W. Morton, *Difference Methods for Initial-Value Problems*. 2nd edition, Wiley-Interscience, New York, 1967.
- [10] L.H. Thomas, *Elliptic problems in linear difference equations over a network*. Watson Sci. Comput. Lab. Rept., Columbia University, New York, 1949.
- [11] C. Amatore, A.I. Oleinick, O.V. Klymenko, I. Svir. "Theory of long-range diffusion of proteins on a spherical biological membrane. Application to protein clusters formation and actin-comet tail growth," *ChemPhysChem*, vol.10, pp. 1586-1592, 2009.
- [12] C. Amatore, O.V. Klymenko, A.I. Oleinick, I. Svir. "Diffusion with moving boundary on spherical surfaces," *ChemPhysChem*, vol.10, pp. 1593-1602, 2009.