UDC 004.382

FPGA CORES FOR FAST MULTIPLICATIVE INVERSE CALCULATION IN GALOIS FIELDS

Rodrigue Elias[1], Valerii Hlukhov[2], Mohammed Rahma[2], Ivan Zholubak[2]

[1]Lebanese International University
[2]Lviv Polytechnic National University

Abstract*. There are two common methods for division in a Galois Field GF(2^m): based on multiplication an extended Euclidean algorithm for polynomial basis and exponentiation method for normal basis. The disadvantage of the first one is the dependence of division time on the value of operands. So in the study some undependable on operand values methods based on squares and square roots calculation are tested to select ones with the best hardware and time complexity for polynomial basis. All methods were implemented as field programmable integrated circuit (FPGA) cores, their work was verified by simulation.*

Keywords*: binary Galois Field, multiplicative inverse, extended Euclidean algorithm, exponentiation.*

Introduction

Currently, the mathematical basis for digital signature processing is elliptic curves [1]. The processing of the points of the elliptic curve is based on the operations over elements of Galois Field *GF*(2^m), $m \leq 1000$ is the number of bits in code [2], the field elements can be represented in polynomial and normal bases. Hardware implementation of Galois Field processors in FPGA which must perform multiplication and division over such giant codes for such tasks and fields requires high hardware and time costs. The implementation of complex computational algorithms in FPGA at the level of logic elements is a common practice [3].

The quotient α/β can be computed in Galois Field processors directly (by an algorithm with inputs α and β), or indirectly (by computing the multiplicative inverse β^{-1} and then multiplying it by α) [1]. Direct division method I (further Method 1 in this study) is the extended Euclidean algorithm. The extended Euclidean algorithm uses a polynomial basis representation for GF (2^m). Method II (further Methods 2, 3 and 4) is exponentiation. The multiplicative inverse of β can be found efficiently in any basis representation via $\beta^{-1} = \beta^k$ where k is any positive integer satisfying $k \equiv -1 \pmod m$ where m is the degree of β. There is also a specialized algorithm of Itoh, Teechai, and Tsujii [4] for exponentiation to the power $k = 2^m - 2$. The efficiency improvements are especially significant when squaring can be done quickly (e.g., in a normal basis representation).

In [5 - 7] the method for fast calculation of square roots $\beta^{1/2}$ in binary Galois fields was described, so it was idea to transform it to inverse elements β^{-1} calculation.

In [8] it is shown that the hardware multiplication in polynomial and normal bases has approximately equal hardware and time complexity, the software multiplication time in a polynomial basis is 1-2 orders of magnitude less then multiplication time in a normal basis, but the structural complexity of the multipliers for a normal basis is *m* times greater then the structural complexity of the multipliers for the polynomial basis [9]. But disadvantage of a polynomial basis and recommended in [2] based on multiplication extended Euclidean algorithm is the dependence of Galois Fields inverse elements computing time on the value of operands [8].

In [10] was shown that multiplication in binary fields has one of the highest time complexities, it is third after multiplication complexities in fields with characteristics 3 and 5. High complexity creates additional difficulties for hackers. Therefore, achieved results was checked and following study is focus on multipliers for binary fields.

In [11] multiplicative inverse calculation in base of bit-parallel multiplier-accumulator was studied and was shown that based on direct hardware division inverter has the best hardware and time costs in comparison with them.

1. Research objective

In this study some undependable on operands and based on fast squares and square roots calculation methods of inverse elements computing in polynomial basis are tested to select ones with the best hardware and time complexity. All methods must be implemented and tested in FPGA.

2. Software time complexity of multipliers for polynomial basis

One of the computer system hacking methods is the brute-force method [12], in which the general-purpose computer selects all sorts of keys or passwords until one of them fits. The same operations over Galois fields elements are performed both during the execution of the hack program and in the hardware crypto processors. For general purpose computers, it is possible to estimate the time of execution of the main operation, multiplication of the Galois fields elements, for extended fields with different characteristics, but with approximately the same order. The Galois Field GF(2^{999}) can be used as a basis. The Maple 2017 package was used for calculations [13]. The relative times of execution of multiplications in different fields with respect to the multiplication time in the binary field GF(2^{999}) are shown Table 1 and in the Fig. 1, where prime $p \approx 2^{999}$ is characteristic of prime GF(p). The relative time complexity was determined by the ratio of the multiplication time in the field GF(d^m) to the multiplication time in the field GF(2^{999}).

Table 1

The relative time complexity of multiplication in extended field with different characteristics

Field Characteristic	Relative Time
2	1,00
3	1,46
5	1,18
7	0,84
11	0,59
13	0,53
17	0,45
19	0,42
23	0,38
29	0,33
p	0,03

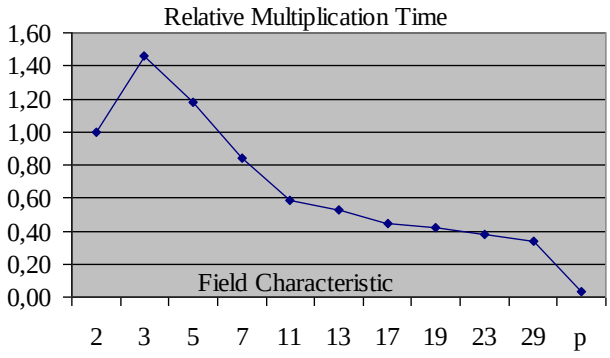

Fig. 1. Relative time complexity of multiplying in a different fields

As can be seen from the Table 1, software multiplication of triple extended field elements has the longest execution time. It provides hardware cryptoprocessors based on such fields additional protection against hacking. Software-implemented operations over prime field elements are executed in the fastest way, that indicates the inappropriateness of cryptographic processors based on such fields. Multiplication in binary fields has one of the highest time complexity, it is third after multiplication complexities in fields with characteristics 3 and 5. Therefore, the following study will focus on binary fields.

3. Binary Galois Field Processor

Currently, the mathematical basis for digital signature processing is elliptic curves [1]. The processing of the points of the elliptic curve is based on the operations over elements of Galois Field GF(2m), m ≤ *1000 is* the number of bits in code [2], the field elements can be represented in polynomial and normal bases. Hardware implementation of Galois Field processors in FPGA which must perform multiplication and division over such giant codes for such tasks and fields requires high hardware and time costs.

One of Galois Field GF(2^m) processor is shown in Fig. 2. Processor operational unit consist of multiplier and arithmetical and logical unit (ALU). ALU performs addition and square calculation. Necessary for elliptic curve cryptography division can be executed by software or firmware.

Since calculation time of realized in GF processor (Fig. 2) inverter does not depend on the Galois Field elements value, processor does not have a comparator which is necessary for element's value analyzing and which was in the previous processor [11].

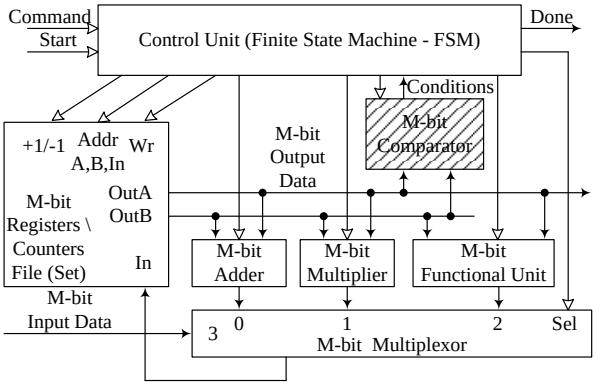

Fig. 2. Galois Field processor [5]

In this study undependable on operands and based on squares and square roots calculation

methods of inverse elements computing in polynomial basis are tested to select ones with the best hardware and time complexity in polynomial basis for their implementation into FPGA GF processor with parallel multiplier. All methods were implemented and tested in FPGA.

4. Core Generator for Galois Field Processor Units Generation

Dedicated Core Generator [10] which produces VHDL-descriptions of inverter cores has been modified to generate VHDL-descriptions on base of fast parallel multipliers. The Core Generator parameters which user can set are:

multiplicative inverse method type (based on bit-parallel or full-parallel multipliers);

Galois Field GF(2^m) degree m ($3 \le m \le 1000$);

Galois Field GF(2^m) irreducible binary polynomial F.

The inverter cores are generated as set of standard logical blocks:

SqrR – square root calculator;

Sqr – square calculator;

Mul – bit-parallel multiplier-accumulator [10] or parallel multiplier for this study;

Rg – register;

MX – multiplexor;

Cmp – comparator;

Cntr – control unit as finite state machine (FSM).

In structure of GF processor some of these blocks are already present. Extra block will be collected in additional functional unit (FU). And only hardware resources of FU were calculated as hardware complexity for multiplicative inverse implementation.

For investigation all parts of every inverter are collected in one core. Schematic symbols of generated cores are shown in Fig. 3, where inverter U1 uses only multiplier, U2 uses multiplier and squarer, U3 uses multiplier, squarer and square rooter (method 2 in [10]).

For study cores with m=64 were generated:

M: integer := 64; logM: integer := 8;

F: std_logic_vector(M:0) := '1'& x"010100000101001B".

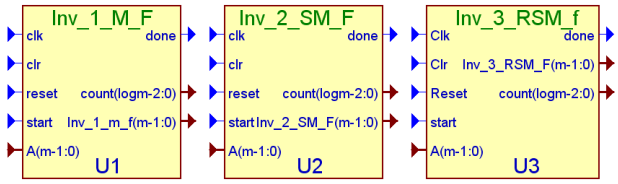

Fig. 3. Symbols of generated inverters cores

5. Exponentiation Based Firmware Division Methods

5.1. The Method Based on the Use of Squaring and Square roots

Since the multiplicative group of the Galois field GF(2^n) is cyclic of order 2^n-1, for any nonzero element $a \in GF(2^n)$, the inverse a^{-1} over $GF(2^n)$ can be compute using the Itoh-Tsujii method as follows: by Lagrange's theorem, $a^{2^m-1} = 1$, which implies that $a^{2^m-2} = a^{-1}$.

The proposed in [10] algorithm inverses element a using Square Root and Square (SRS) calculation in polynomial basis with calculation time undependable from element a code as follows:

$$a^{-1} = \left(a^{2^{\pm\left(\frac{m}{2}\right)}} \right)^{E\_O} \otimes \prod_{j=1}^{\frac{m-1-E\_O}{2}} a^{2^j} \otimes a^{2^{-j}} \quad (1)$$

$$a^{-1} = \left(a^{2^{\pm\left(\frac{m}{2}\right)}} \right)^{E\_O} \otimes \prod_{j=1}^{\frac{m-1-E\_O}{2}} a^{2^j} \otimes \prod_{j=1}^{\frac{m-1-E\_O}{2}} a^{2^{-j}} \quad (2)$$

Where E_O = Even_odd = (m-1)mod 2.

Method 2 algorithm:

Input: $a \in GF(2^m)$ *for* $m \ge 3$;

E_O =(m-1)mod 2.

Output: A^{-1}

1. *Beta* ← *(m-1-E_O)/2; P*← *1;*
   $R = \sqrt{A}$; $S = A^2$.
   Compute $P_{RS} = R \times S$.
2. *Compute* $P = P \times P_{RS}$; *Beta=Beta-1.*
3. *If Beta = 0 then go to last step 5.*
   Else $R = \sqrt{R}$; $S = S^2$.
   Go to step 2.
4. *If (E_O = 0) then* $A^{-1} = P$
   Else $A^{-1} = P \times \sqrt{R}$ *or* $A^{-1} = P \times S^2$.
5. *Return* A^{-1}.

Total hardware resources (number of flip-flops, Look-up-Tables (LUTs) and slices) for SRS method are shown in Table 2, where also data are about clock period and their number required for multiplicative inverse calculation. Also there is complex indicator, the best method has the lowest value of this indicator.

Detailed hardware resources list (number of flip-flops, LUTs and slices) for SRS method are shown in Table 3. Only part of SRS hardware resources is placed in FU.

Table 2

Hardware Resources of Functional Unit, m=64

Method	SRS	SM	M
Total # Slices	919	792	699
Total # FFs	277	82	78
Total # LUTs	2934	2523	2334
# FFs in FU	0	0	0
# LUTs in FU (L)	514	476	0
Min clock period (P, ns)	9.8	10.0	9.4
# clocks (C)	126	126	250
complex indicator (CI =L*P*C/106)	1,13	1,00 1,13	1,64 1,00

Table 3

Detailed Hardware Resources List for Square Root and Square Used Method, m=64

By Hierarchy	# FFs	# LUTs	Placed in
SRS Method	277	2934	
Sqr	0	127	FU
SqrR	0	414	FU
Mul	0	2168	
MX_1	0	65	
MX_2	0	65	
MX_3	0	68	
MX_4	0	70	
Rg	256	0	
Cntr	7	9	

Tested version of inverter which realized SRS method is shown in Fig. 4.

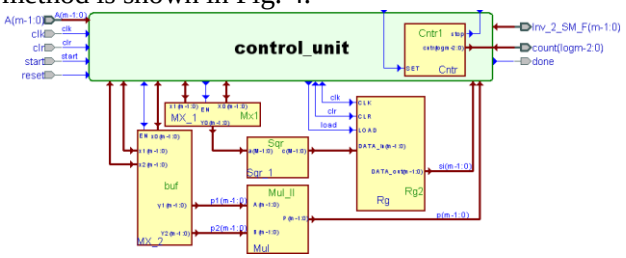

Fig. 4. An inverter that implements SRS method

5.2. The Method Based on the Use of Squaring and Multiplication

In this method the square is used for inverse element calculation in polynomial basis. The algorithm is true Itoh, Teechai, and Tsujii [4] method.

Total hardware resources (number of flip-flops, LUTs and slices) for based on the use of the Square and Multiplication (SM) method are shown in Table 2, where also data are about clock period and their number required for multiplicative inverse calculation.

Detailed hardware resources list (number of flip-flops, LUTs and slices) for SM method are shown in Table 4. Only part of SM method hardware resources is placed in FU.

Table 4

Detailed Hardware Resources List for SM method, m=64

By Hierarchy	# FFs	# LUTs	Placed in
SM Method	82	2523	
Sqr	0	476	FU
Mul	0	1167	
MX_1	0	782	
1,64 MX_2	0	83	
Rg	64	0	
Cntr	7	8	

Tested version of inverter which realized SM method is shown in Fig. 5.

5.3. The Method Based Only on the Use of Multiplication

Here the square is also used for inverse element calculation in polynomial basis but no additional elements (FU) are in GF processor (Fig. 2).

Total hardware resources (number of flip-flops, LUTs and slices) for Only Multiplication (M) used method are shown in Table 2, where also data are about clock period and their number required for multiplicative inverse calculation.

Detailed hardware resources list (number of flip-flops, LUTs and slices) for M method are shown in Table 5. No part of M method hardware resources is placed in FU.

Table 5

Detailed Hardware Resources List for M method, m=64

By Hierarchy	# FFs	# LUTs	Placed in
M Method	78	2334	
Mul	0	2178	
MX	0	138	
Rg	64	0	
Cntr	7	10	

Tested version of inverter which realized M method is shown in Fig. 6.

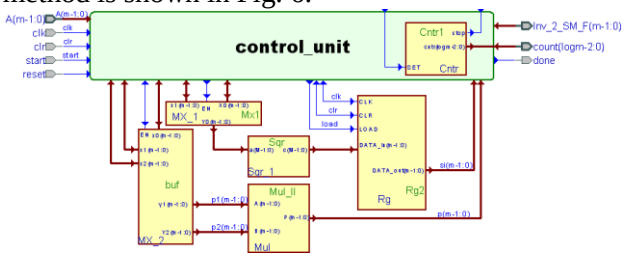

Fig. 5. An inverter that implements SM method

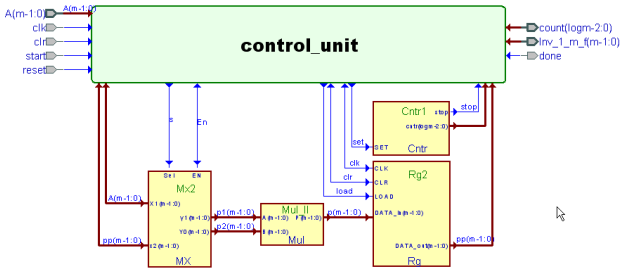

Fig. 6. An inverter that implements M method

Conclusion

Studies have shown that all considered methods based on use of squares and square roots calculations and use of fast parallel multipliers in dedicated Galois Field processor provide calculation of inverse element in binary Galois fields in a polynomial basis for time independent of the value of the operands. Based on Itoh, Teechai, and Tsujii Squaring and Multiplication methods can be recommended for use with polynomial basis too. It provides a minimum clocks for finding the inverse element and it requires less amount of inventory costs then methods based in use of square roots and squares.

References

1. Hankerson, D., Menezes, A., Vanstone, S. (2004), Guide to Elliptic Curve Cryptography, Springer-Verlag New York: ISBN 0-387-95273-4.

2. IEEE 1363-2000 (2000). Standard Specifications for Public-Key Cryptography. Copyright © 2000 IEEE. All rights reserved.

3. Palagin A. The structure of FPGA-based cyclic-code converters / Alexander Palagin, Vladimir Opanasenko, Sergey Krivoi // Optical Memory & Neural Networks (Information Optics). Springer. – 2013, Vol. 22, N.4. – PP. 207–216.

4. Itoh, T., Teechai, O., and Tsujii, S. (1986), "A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^t)$ Using Normal Bases," J. Society for Electronic Communications (Japan) 44, pp. 31–36.

5. M. Repka, "Computing pth roots in extended finite fields of prime characteristic p ≥ 2," *The Institution of Engineering and Technology*, vol. 52, no. 9, p. 718–719, 2016.

6. Gora Adj and Francisco Rodrıguez-Henrıquez, "Square root computation over even extension fields," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2829–2841, Nov 2014.

7. S. J. Aboud, "An Efficient Method for Finding Square Root" *International Journal of Statistics, ISSN:2051-8285*, vol. 37, no. 1, pp. 1103–1106, 2013.

8. Hlukhov, V. S. (2007), Comparison of polynomial and normal bases of Galois fields elements presentation [Porivniannia polinomialnoho ta normalnoho bazysiv predstavlennia elementiv poliv Halua]. Scientific Bulletin of Lviv Polytechnic National University. Computer-aided design systems. Theory and practice. vol. 591, Lviv, Ukraine, pp. 22–27 (In Ukrainian).

9. Hlukhov, V. S., Elias, R., Rahma, M. (2017), Structural Complexity of Multipliers for Galois Fields Elements in Normal and Polynomial Bases [Strukturna skladnist pomnozhuvachiv elementiv poliv Halua u normalnomu ta polinomialnomu bazysakh]. Electrotechnic and Computer Systems. – Odessa. Astroprint. – № 25(101). – pp. 324–331 (In Ukrainian).

10. Hlukhov, V., Rahma, M., Zholubak, I (2018), Devices for multiplicative inverse calculation in binary Galois Fields. DESSERT'2018. 9th International IEEE Conference Dependable Systems, Services and Technologies. UKRAINE, KYIV, MAY 24–27, 2018, unpublished.

11. Hlukhov, V., Zholubak, I., Kostyk, A., Rahma M. (2017), Galois Fields Elements Processing Units for Cryptographic Data Protection in Cyber-Physical Systems. Advances in Cyber-Physical Systems. Volume II, Number 2, 2017. © Lviv Polytechnic National University, pp. 9–18, in press.

12. Password cracking. https://en.wikipedia.org/wiki/Password_cracking.

13. Maple User Manual. Copyright © Maplesoft, a division of Waterloo Maple Inc. 2017.

ЯДРА ПЛІС ДЛЯ ШВИДКОГО ЗНАХОДЖЕННЯ ЗВОРОТНИХ ЕЛЕМЕНТІВ ПОЛІВ ГАЛУА

Родріг Еліас[1], Валерій Глухов[2], Мохаммед Рахма[2], Іван Жолубак[2]

[1]Ліванський міжнародний університет
[2]Національний університет «Львівська політехніка»

Анотація. *У даний час математичною основою опрацювання цифрового підпису є еліптичні криві. При цьому опрацювання точок еліптичної кривої базується на виконанні операцій у полях*

Галуа GF(p^m)*, порядок поля може сягати значень* 2^{1000}*, елементи поля може бути представлено у поліноміальному та нормальному базисах. Великий час програмного виконання операцій у полях Галуа ускладнює злом комп'ютерних систем, для захисту яких використовується цифровий підпис. З цієї точки зору двійкові поля мають перевагу перед полями з іншими характеристиками, особливо – перед простими полями, для яких m=1. За цією ознакою двійкові поля поступаються тільки полям з характеристиками p=3 та p=5. Оскільки сучасні комп'ютери працюють на основі двійкової системи числення, використання двійкових полів Галуа для створення реалізованих на ПЛІС вузлів систем захисту інформації виправдане. Існує два поширені методи ділення в полі Галуа GF(2^m): заснований на використанні множення розширений алгоритм Евкліда для поліноміального базису і для нормального базису - метод, заснований на знаходженні зворотного елемента шляхом піднесенні до степеня. Недоліком першого є залежність часу ділення від значення операндів, що може сприяти злому систем захисту інформації. Тому в даній роботі методи знаходження зворотного елемента в поліноміальному базисі, що забезпечують незалежність часу знаходження від значення операндів, аналізувалися для вибору методу з кращими показниками апаратної і часової складності при їх реалізації на ПЛІС. У роботі аналізувалися методи, на основі використання яких можна створити VHDL-опис функціонально закінченого вузла ПЛІС (ядра ПЛІС з функцією знаходження зворотного елемента - інвертора), який можна вбудувати в спеціалізований процесор для опрацювання елементів полів Галуа, який будується у ПЛІС. У роботі наведено функціональну схему процесора та показано, які додаткові елементи в його складі потрібні для реалізації кожного з розглянутих методів. Було прийнято, що до складу процесора входить суматор та паралельний матричний помножувач. Було розглянуто три методи знаходження зворотних елементів: метод, що вимагає додаткових вузлів піднесення до квадрату (квадратора) та знаходження квадратного кореня, метод, що вимагає тільки додаткового квадратора, та метод, що не вимагає додаткових операційних вузлів. Для реалізації кожного з методів у ПЛІС було створено спеціалізований генератор ядер інверторів, який забезпечував створення VHDL-описів за характеристиками, які задавалися під час проведення досліджень: характеристику поля, утворюючий поліном, метод знаходження зворотного елемента. Усі розроблені ядра інверторів було перевірено (для m=64) шляхом моделювання та імплементовано до складу ПЛІС. За результатами моделювання визначалися часова складність, а за результатами імплементації – апаратна складність кожного з розроблених ядер. Результати оцінювання складностей наведено у роботі. Комплексне оцінювання часової та апаратної складності показало перевагу методу, заснованому на використанні додаткового квадратора.*

Ключові слова: *двійкове поле Галуа, мультиплікативний зворотний елемент, розширений алгоритм Евкліда, зведення в ступінь.*

ЯДРА ПЛИС ДЛЯ БЫСТРОГО НАХОЖДЕНИЯ ОБРАТНЫХ ЭЛЕМЕНТОВ ПОЛЕЙ ГАЛУА

Родриг Элиас[1], Валерий Глухов[2], Мохаммед Рахма[2], Иван Жолубак[2]

[1]Ливанский международный университет
[2]Национальный университет «Львовская политехника»

Аннотация*. Существует два распространенных метода деления в поле Галуа GF(2^m): основанный на применении умножения расширенный алгоритм Евклида для полиномиального базиса и метод возведения в степень для нормального базиса. Недостатком первого является зависимость времени деления от значения операндов. Поэтому в данной работе основанные на использовании операций возведения в квадрат и извлечения квадратного корня и независящие от значений операндов методы деления анализировались для выбора метода с лучшими показателями аппаратной и временной сложности. Методы были ориентированы для использования в полиномиальном базисе представления элементов полей Галуа GF(2^m). Все методы были реализованы в виде ядер ПЛИС, их работа была проверена моделированием.*

Ключевые слова*: двоичное поле Галуа, мультипликативный обратный элемент, расширенный алгоритм Евклида, возведение в степень.*

Received 24.03.2018

Элиас Родриг Митри, кандидат технических наук, инструктор кафедры электротехники и электронной инженерии Ливанского международного университета, Школа инженерии, Блок 1G, Ливанский международный университет, P.O. Box: 146404, Бейрут, Ливан, E-mail: rodrigue.elias@liu.edu.lb, м/т.: 961.3.492949

Elias Rodrigue Metri, PhD, an instructor at the School of Engineering at the Lebanese International University, School of Engineering, Block G, Lebanese International University, P.O. Box: 146404 Beirut, Lebanon
E-mail: rodrigue.elias@liu.edu.lb, Tel.: 961.3.492949
ORCID ID: 0000-0003-4506-0368

Глухов Валерий Сергеевич, доктор технических наук, профессор, профессор кафедры электронных вычислительных машин Национального университета «Львовская политехника», ул. С. Бандеры, 12, Львов, Украина, E-mail: glukhov@polynet.lviv.ua , м/т.: +38-063-75-72-330.

Valeriy Hlukhov, Dr. of Science, Professor, Professor of the Department of Computer Engineering, Lviv Polytechnic National University, S. Bandera Str., 12, Lviv, Ukraine, E-mail: glukhov@polynet.lviv.ua, м/т.: +38-063-75-72-330.
ORCID ID:0000-0002-0542-7447

Рахма Мохаммед Кадим, аспирант кафедры электронных вычислительных машин Национального университета «Львовская политехника», ул. С. Бандеры, 12, Львов, Украина, E-mail: muhamed_kadhem@yahoo.com , +38-093-19-62-350

Rahma Mohammed Kadhim, PhD student of the Department of Computer Engineering, Lviv Polytechnic National University, S. Bandera Str., 12, Lviv, Ukraine, E-mail: muhamed_kadhem@yahoo.com, +38-093-19-62-350
ORCID ID: 0000-0002-8377-1833

Жолубак Иван Михайлович, ассистент кафедры электронных вычислительных машин Национального университета «Львовская политехника», ул. С. Бандеры, 12, Львов, Украина, E-mail: ivanzholubak7@ukr.net, +380 63 204 3548

Ivan Zholubak, assistant of the Department of Computer Engineering, Lviv Polytechnic National University, S. Bandera Str., 12, Lviv, Ukraine, E-mail: ivanzholubak7@ukr.net, +380 63 204 3548
ORCID ID: 0000-0001-8871-7222