I. J. Computer Network and Information Security, 2023, 3, 1-14
Published Online on June 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2023.03.01

Galua Field Multipliers Core Generator

I. M. Zholubak*
Lviv Polytechnic National University, Computer Engineering Department, Stepana Bandery 12, Lviv, Ukraine
E-mail: Ivan.M.Zholubak@lpnu.ua
ORCID iD: https://orcid.org/0000-0001-8871-7222
*Corresponding Author

V. S. Hlukhov
Lviv Polytechnic National University, Computer Engineering Department, Stepana Bandery 12, Lviv, Ukraine
E-mail: valerii.s.hlukhov@lpnu.ua
ORCID iD: https://orcid.org/0000-0002-0542-7447

Received: 28 July 2022; Revised: 12 November 2022; Accepted: 01 April 2023; Published: 08 June 2023

Abstract: An important part of based on elliptical curves cryptographic data protection is multipliers of Galois fields. For based on elliptical curves digital signatures, not only prime but also extended Galois fields GF(p^m) are used. The article provides a theoretical justification for the use of extended Galois fields GF(d^m) with characteristics d > 2, and a criterion for determining the best field is presented. With the use of the proposed criterion, the best fields, which are advisable to use in data protection, are determined.

Cores (VHDL descriptions of digital units) are considered as structural part of based on FPGA devices. In the article methods for cryptoprocessors cores creating were analyzed. The article describes the generator of VHDL descriptions of extended Galois field multipliers with big characteristic (up to 2^{998}). The use of mathematical packages for calculations to improve the quality of information security is also considered.

The Galois field multipliers generator creates the VHDL description of multipliers schemes, describes connections of their parts and generates VHDL descriptions of these parts as result of Quine-McCluskey Boolean functions minimization method. However, the execution time of the algorithm increases with increasing amount of input data. Accordingly, generating field multipliers with large characteristic can take frepom a few seconds to several tens of seconds.

It's important to simplify the design and minimize logic gates number in a field programmable gate array (FPGA) because it will speed up the operation of multipliers. The generator creates multipliers according to the three variants.

The efficiency of using multipliers for fields with different characteristics was compared in article.

The expediency of using extended Galois fields GF($d^{m)}$ with characteristics d > 2 in data protection tools is analyzed, a criterion for comparing data protection tools based on such Galois fields is determined, and the best fields according to the selected criterion when implemented according to a certain algorithm are determined.

Index Terms: Modified Guild cell (MGC), Galois fields (GF), Boolean functions, Galua field multiplier generator, field programmable gate array (FPGA).

1. Introduction

Integrated circuits are designed on the basis of elements that perform simple logical operations - Boolean functions. The Quine McCluskey method is used to minimize them [1].

Characteristic features of the current stage of development of computer technology are the development and implementation of cyberphysical systems (CPS), as well as preparation for the emergence of quantum computers. The emergence and development of CPS, one of the main features of which is the use of wireless technology, raises the issue of data protection in these systems. The constant increase of computer productivity, the emergence of new technologies and algorithms, and the introduction of new digital technologies can be used by attackers to violate information security. This necessitates the search for new, more reliable methods of information security (IS). It is desirable that these methods are based on already known technologies and tools and improve their effectiveness. Today, one of the methods of IS is the use of digital signatures, which are based on algorithms for processing the points of elliptic curves (EC) and elements of extended binary GF(2^m) and prime GF(d) Galois fields. The capabilities of quantum computers make it dangerous to use existing algorithms based on the use of EC. Although powerful quantum computers have not yet emerged, IS algorithms have been introduced that remain reliable in the quantum computer

age. One of the possible methods is a method based on the use of EC isogenies in the Galois field GF(2^m). To calculate it, the same operations are used as in existing digital signature algorithms that use Galois fields GF(2^m). Also with the binary fields GF(2^m) other extended Galois fields GF(d^n) can be used, such that $2^m \approx d^n$. When processing codes of the elements of extended binary Galois fields, it is necessary to process binary codes, the length of which is m (according to modern standards, m can reach 1000). Processing of such codes is the purpose of operational units for Galois fields that are used in the IS based on EC. The article describes the tool for generating Galois field multipliers VHDL descriptions of different characteristic and different order, which are used in IS based on EC. A feature of IS algorithms is their multilevel structure, where specific mathematical operations on multi-bit codes are performed at different levels: operations over elements of prime GF(d) and extended GF(d^m) Galois fields, operations over points of elliptic curves. Depending on the conditions of use, it is necessary to provide the configuration of operating devices that implement these algorithms: to change the Galois field, the characteristic for the representation of field elements, to change the elliptic curve. One of the main elements for dedicated computers functional units design is FPGA, which has the following important features [2]:

- FPGA facilitates the next rapid transition to ASIC, that ensures mass production.
- FPGA provides hardware implementation of algorithms and storage within the chip of intermediate results in the execution of these algorithms.
- Modern FPGAs ensure the preservation of intellectual property and complicate unauthorized duplication and "reverse engineering".

At the present stage, when ISs are implemented in the CFS, it becomes important to ensure their work in real-time. This requires the use of high-speed hardware solutions - dedicated processors, which are implemented in FPGA. The multi-level dedicated processor (SP), which performs operations over elliptic curve points when processing digital signatures, is used as a basis for designing IS tools. The design of such dedicated processor requires the use of special sections of mathematics: Galois fields, elliptic curves (EC), etc. Elements of Galois fields and EC points are represented by multi-bit binary codes (which are represented by hundreds and thousands of bits). SP requires original functional units to perform operations on them. The SP operates on the basis of such theoretical provisions that allow to consider it as a dedicated computer system with architecture that is different from the known architectures of usual computers. Modern computer tools are required to adhere to the principles of open systems, focusing on the use of open standards. Creating SP multiplier is one of the most difficult tasks. In this article, we will consider the creation of a Galois field multiplier core generator for SP.

Modern information security tools use operations on extended Galois fields GF(2^n) with large degrees n, their elements are represented in the polynomial or normal basis [3]. That's why the creation of tools for operations on the elements of such Galois fields is a very promising scientific and engineering direction. The investigation of the hardware, structural and time complexity of the multipliers of the elements of such fields is a difficult task, because you have to multiply the codes, the bit size of which coincides with the degree of the Galois field and can reach 1000 bits. The object of study are fields with characteristics that are prime numbers (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, …, 997). The order of such fields is approximately 2^{998}.

2. Related Work

In recent years, the concept of elliptical curves has found its application in cryptography [4]. The reason for this is that elliptic curves over finite fields form finite groups, on which it is easy to determine arithmetic operations due to the rich structure of the groups [5]. Until now, cryptography has worked with multiplicative groups on some finite fields. Elliptic curves are similar to these groups, but their advantage is that there is more freedom to choose an elliptic curve than to choose a finite field. In addition, elliptical cryptosystems provide better data protection. Galois field arithmetic is used to perform operations over points of elliptic curves, the element codes of such fields, are presented in polynomial or normal bases ([6, [7]). Multipliers for Galois fields are characterized by high hardware [8], structural [9] and time [10] complexity.

In [11] architecture that trades a reduction in resources with an increase in the number of clock cycles is described. This architecture is also fine grain scalable in both the time and the area (or logic) dimensions thus facilitating implementations that maximize their use of finite FPGA resources while achieving fast computational speed. In [12] authors propose Galois field arithmetic using irreducible polynomial to generate the S-box for AES. In [13] authors consider a reconfigurable digital circuit which performs various Galois field arithmetic operations with the same set of hardware. Using this hardware, we can perform a few operations in parallel. In [14] authors show that the Reed-Solomon codes built on a Galois field of characteristic 3, can be used in transmission through a visible light communication channel, due to the information can be encoded using the colors red, green and blue. In [15] authors introduce the concept of multivalued neural element over Galois field relative to any system of characters of group on which we define logical functions. In [16] a method is proposed to implement Markov probability functions defined based on a stochastic matrix of a predefined size, when using a set of multivariate polynomials over Galois field and an array of evenly distributed non-correlated (pseudo) random numbers of a predefined capacity. In [17] new structure of

square operator takes advantage of Normal Basis representation of GF elements is described. In [18] Galois fields are used to improve Energy Management System. In [19] authors demonstrate that Correlation Power Analysis using only a specific bit has the superior performance because of modulo operation in Galois field for AES-128 encryption algorithm use. The paper [20] considers algorithms for multiplying field elements, while paying attention to both table multiplication and multiplication "on the fly" and algorithms for calculating multiplicative inversion [21]. The research work [22] proposes a framework for checking the correctness of Galois field arithmetic operations in digital circuits. In [23] a technique is proposed for performing distributed nonlinear-polynomial (NP) computing over Galois field.

Many devices for processing Galois field GF(d^m) elements are known, which are used in various cryptographic transformations [24].

A matrix multiplier for binary numbers [25] which consists of Guild cells [26] is known. Also multiplier which is based on modified Guild cells to perform multiplication over Galois field elements GF(d^m) is known. In article [27] we are considered the hardware costs of matrix Galois field GF(d^m) multipliers when $d < 4$, in work [8] we are considered the hardware costs of multipliers when $d \leq 998$. The process of creating core generators is described in [28].

3. Problem Description

For use extended Galois fields GF(d^m) with characteristics $d > 2$ in data protection systems, it is necessary to choose a criterion for determining the best field. Also, using the proposed criterion, it is necessary to determine the best fields that should be used for data protection.

The internal structure of the modified Guild cells and its influence on the assessment of Galois field GF(d^m) multipliers hardware complexity is the subject of this work. The use of characteristic $d > 2$ is insufficiently researched and creates many more options for coding and, accordingly, can increase the resistance of the cipher to hacking [29].

Although there are many tools for creating Galois field multipliers on FPGA (field programming gate array) on hardware description languages (VHDL, Verilog, HLS), however, this approach is extremely time consuming. Therefore, there was a need to create a VHDL description generator of such multipliers. Described in this article generator allows the user to generate multipliers for FPGA, so user can explore their advantages and disadvantages.

4. The Goal of the Work

The purpose of the work is to theoretically justify the feasibility of using extended Galois fields GF(d^m) with characteristics $d > 2$ in data protection tools, to define a criterion for comparing data protection tools based on such Galois fields, to determine the best fields according to the chosen criterion. Also, the purpose of the article is to create a tool for generating VHDL descriptions of multipliers of elements of such fields for further use of these descriptions in the implementation of data protection tools on FPGAs.

The generator has to create multipliers according to the three most common variants:

- in base of Modified Guild Cell (MGC) which is presented as one whole element (a black box) – first variant.
- in base of MGC which is presented as multiplier and adder - second variant.
- in base of MGC which is a circuit that consists of simple logic gates and implements the result of Boolean functions - third variant.

5. Theoretical Background of Calculating the Hardware Costs of Multipliers

It is proposed to take the hardware complexity of the multipliers as a basis for comparing different extended Galois fields. It is also suggested to use multipliers based on Modified Guild Cells (MGC).

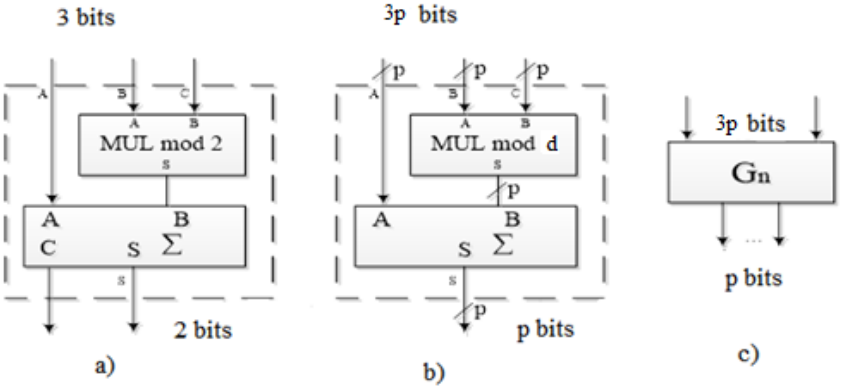

Fig.1. a) Guild Cell, b) scheme of the Modified Guild Cell for processing elements of Galois fields GF(d^m), c) symbol of the Modified Guild Cell GF(d^m)

The multiplier for Galois fields GF(d^m) can be implemented on the basis of Modified Guild Cells (MGC). MGCs for Galois fields GF(d^m) should have 3p inputs and p outputs, each $p = \lceil \log_2 m \rceil$ bit (Fig. 1). When using modern FPGAs, the logic cells of which are built on the basis of software 6-input combinational circuits (LUTs), the implementation of such Guild cells on the FPGA in the most general case, when the structure of the MCG is not specified, but only the number of its inputs and outputs is taken into account, requires $q_1 = (2^{3p-h-1} - 1) \cdot p$ LUTs, h is the number of inputs of the LUT used. We use 6-input LUTs.

It is convenient to estimate the hardware complexity in comparison with the costs of the multiplier for the binary Galois Field GF(2^n).

For the variant, when the MGC is a whole element, the coefficient of hardware complexity $k_{mul} = k_g * k_k$, де $k_g = {k_{gd}}/{k_{g2}}$, $k_k = \frac{k_{kd}}{k_{k2}}$ are coefficients of complexity of MGC, k_{gd} and k_{g2}, k_{kd} and k_{k2} are number of LUTs in MGC and number of MGC for Galois Fields GF(d^m) and GF(2^n), respectively.

For binary Galois fields GF(2^n) $k_{g2} = 1$, for others $k_{gd} = (2^{p-h-1} - 1) * k$, where $p = 3 * \lceil \log_2 d \rceil$, and k $= \lceil \log_2 d \rceil$. It follows that $k_{gd} = (2^{3*\lceil \log_2 d \rceil - h - 1} - 1) * \lceil \log_2 d \rceil$. So:

$$k_{gd} = \left(2^{3*\lceil \log_2 d \rceil - h - 1} - 1\right) * \lceil \log_2 d \rceil, d \geq 3 \quad (1)$$

In binary fields GF(2^n) it is required to implement the multiplier with $k_{k2}=2n^2 - 2n + 1$ MGCs. In the Galois Field with a characteristic d GF(d^m) - $k_{kd} = 2m^2 - 2m + 1$ and additionally $(m-1) * (2^{3*\lceil \log_2 d \rceil - h - 1} - 1) * \lceil \log_2 d \rceil$ LUTs needed to find the coefficient by which an irreducible polynomial must be multiplied. These hardware costs for the implementation of the element f, which forms this coefficient, can be neglected in this case, since they are small compared to the costs for the implementation of the MGCs themselves. The element f, used to find the coefficient by which to multiply the polynomial to sum the intermediate result, calculates a function that depends on two parameters: $f = (d - G_m) \, mod \, d = (-G_m) \, mod \, d$, where d is the basis of the field, G_m is the output of the Modified Guild Cell in the direct course of calculations. The coefficient at the highest degree of an irreducible polynomial is always equal to 1. So:

$$k_k \approx \frac{2m^2 - 2m + 1}{2n^2 - 2n + 1} \quad (2)$$

k_k- the ratio of the number of MGC of the GF(d^m) field to GF(2^n).

$$k_{mul} \approx \frac{(2^{3*\lceil \log_2 d \rceil - h - 1} - 1) * \lceil \log_2 d \rceil (2m^2 - 2m + 1)}{2n^2 - 2n + 1} \approx 2^{3*\lceil \log_2 d \rceil - h - 1}, d \geq 3 \quad (3)$$

k_{mul}- the ratio of the number of LUTs of the GF(d^m) field to GF(2^n).

In this case $d^m \approx 2^n$. Then $m \approx log_d 2^n = \frac{n}{log_2 d}$, $k_k \approx \frac{\frac{2n^2}{(log_2 d)^2} - \frac{2n}{log_2 d} + 1}{2n^2 - 2n + 1} \approx log_2^{-1} d$, $k_{mul} \approx \frac{(2^{3*\lceil \log_2 d \rceil - h - 1} - 1)(\log_2 d)}{log_2 d} \approx 2^{3*\lceil \log_2 d \rceil - h - 1}$.

For small n k_{mul} is necessary to calculate using more accurate formulas (1, 2, 3).

For the variant, when the MGC consists of a multiplier and an adder, each of which has 2p inputs and p outputs, evaluated separately, for the implementation of one MGC it will be necessary $q_2 = 2 \cdot (2^{2p-h-1} - 1) \cdot p$ LUTs. Then: $\frac{q_1}{q_2} = \frac{(2^{3p-h-1} - 1) \cdot p}{2 \cdot (2^{2p-h-1} - 1) \cdot p} \approx \frac{2^{3p-h-1}}{2 \cdot 2^{2p-h-1}} = 2^{p-1}$ - the cost ratio for the implementation of one MGC according to the first and second variant. From the formula, we can see that the internal structure of the MGC significantly affects the estimation of hardware costs. Additional consideration of the internal structure of the MGC reduces the estimated value of hardware complexity compared to the first variant of estimating hardware costs, when only the number of inputs and outputs is considered.

Let's estimate hardware costs according to the second variant. For binary Galois fields $k_{g2} = 1/2$, for other fields $k_{gd} = (2^{2*\lceil \log_2 d \rceil - h - 1} - 1) * \lceil \log_2 d \rceil * 2$. So:

$$k_{gd} = |2^{2*\lceil \log_2 d \rceil - h - 1} - 1| * \lceil \log_2 d \rceil * 2 \quad (4)$$

In binary fields GF(2^n) it is required $2n^2 - 2n + 1$ MGC to implement the multiplier, and in the Galois field GF(d^m) - $2m^2 - 2m + 1$ MGC. So:

$$k_k \approx \frac{2m^2 - 2m + 1}{2n^2 - 2n + 1} \quad (5)$$

At the same time $d^m \approx 2^n$. Then $m \approx log_d 2^n = \frac{n}{log_2 d}$, $k_k \approx \frac{\left(\frac{2n^2}{(log_2^2 d)} - \frac{2n}{log_2 d} + 1\right)}{2n^2 - 2n + 1} \approx log_2^{-1} d$,

$$k_{mul} \approx \frac{|2^{2*\lceil \log_2 d \rceil - h - 1} - 1| * \lceil \log_2 d \rceil * 2}{\log_2 d} \approx 2^{2*\lceil \log_2 d \rceil} - 4 \quad (6)$$

Comparing the formulas for finding k_{mul} for the first and second cases, we see that in the second variant of the analysis of the hardware complexity of the MGC, at large values of p, the value of the hardware complexity generally decreases in 2^{p-1} times.

Now let's calculate the hardware costs for multipliers MGC that consist of small logical elements. SMn is a multiplier element that performs modular multiplication and addition and has result and carry outputs, i.e. it is a MGC. SMch is an element that performs the operation of adding or subtracting a number in the complementary code during division without restoring the remainders. Sn is a node that defines the type of operation, subtraction or addition when dividing without restoring remainders. Rn is an element that determines whether one more addition operation is required to find the result. Look Fig. 2.

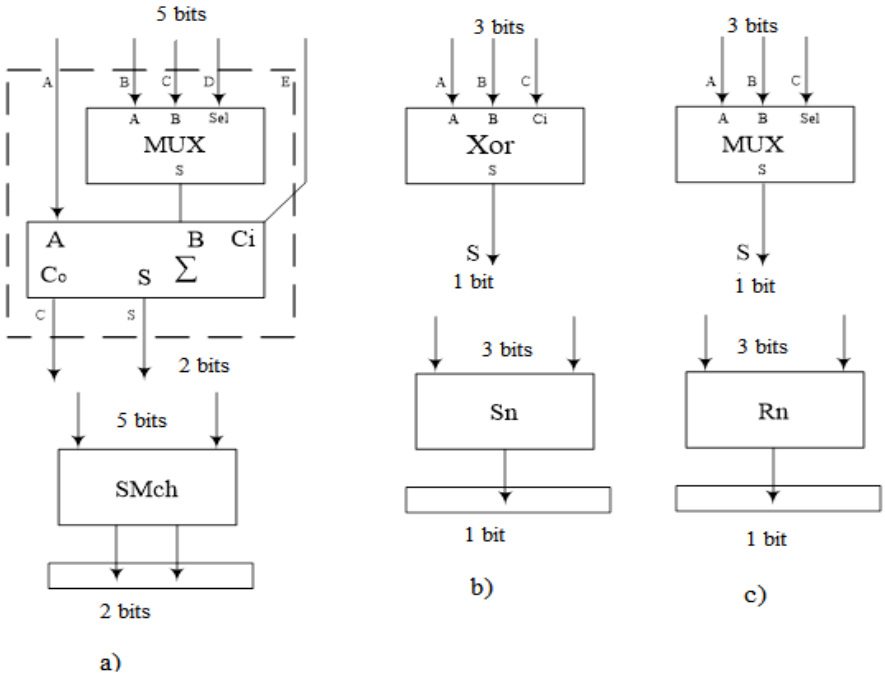

Fig.2. a) scheme and symbol of element SMch, b) scheme and symbol of element Sn, c) scheme and symbol of element Rn

The coefficient of hardware costs of the multiplier for the elements of the field GF(d^m) relative to the similar costs of the multiplier for the elements of the field GF(2^n) $k_{mul} = k_g * k_k$, where $k_g = {k_{gd}}/{k_{g2}}$, $k_k = \frac{k_{kd}}{k_{k2}}$ coefficients of complexity and number of MGC, k_{gd} and k_{g2}, k_{kd} and k_{k2} are the number of LUTs in the MGC and the number of MGCs for the Galois fields GF(d^m) and GF(2^n), respectively.

For binary Galuis Fields GF(2^n) $k_{g2} = 1$, for another fields:

$$k_{gd} = KSMn * KLUT1 + KSMch * KLUT2 + KSn * KLUT3 + KRn * KLUT4 + KSUMn * KLUT5,$$

KSMn, KSMch, KSn, KRn - number of elements, respectively, SMn, SMch, Sn, Rn in the Galois field multiplier, KSUMn – number of sumator elements, KLUTn - number of LUTs that are used for according element.

$$KLUT1 = KLUT2 = KLUT3 = KLUT4 = 1, KLUT5 = 2.$$

$$k_{gd} = (\lceil \log_2 d \rceil)^2 * 2 + (\lceil \log_2 d \rceil)^2 * 2 + \lceil \log_2 d \rceil * 1 + \lceil \log_2 d \rceil * 1 + \lceil \log_2 d \rceil * 2$$

$$k_{gd} = 2(\lceil \log_2 d \rceil)^2 + 2(\lceil \log_2 d \rceil)^2 + 2\lceil \log_2 d \rceil + 2\lceil \log_2 d \rceil = 4(\lceil \log_2 d \rceil)^2 + 4\lceil \log_2 d \rceil.$$

Then:

$$k_g = 2(\lceil \log_2 d \rceil)^2 + 2(\lceil \log_2 d \rceil)^2 + \frac{1}{2}\lceil \log_2 d \rceil + 2\lceil \log_2 d \rceil = 4(\lceil \log_2 d \rceil)^2 + 4\lceil \log_2 d \rceil \quad (7)$$

In binary fields GF(2^n) to implement the multiplier it is required $k_{k2}=2n^2 - 2n + 1$ Guild Cells, and in Galuis Fields GF(d^m) with the characteristic d $k_{kd}=2m^2 - 2m + 1$. Also, it is additionally necessary to have

$$(m - 1) * (2^{3*\lceil \log_2 d \rceil - h - 1} - 1) * \lceil \log_2 d \rceil$$

LUTs to find the coefficient by which the irreducible polynomial must be multiplied (these hardware factors can be neglected in this case, since they are small compared to the costs for the implementation of the Guild cells themselves). So:

$$k_k \approx \frac{2m^2 - 2m + 1}{2n^2 - 2n + 1} \approx \frac{m^2}{n^2} \approx \left(\frac{m}{n}\right)^2 \text{ for big n} \quad (8)$$

$$k_{mul} \approx \frac{(4(\lceil \log_2 d \rceil)^2 + 4\lceil \log_2 d \rceil)m^2}{n^2}, d \geq 2 \quad (9)$$

and $d^m \approx 2^n$. Then $m \approx log_d\, 2^n = \frac{n}{log_2 d}$, $k_k \approx log_2^{-2} d$, $k_{mul} \approx \frac{4(\lceil log_2 d \rceil)^2 + 4\lceil log_2 d \rceil}{(log_2 d)^2}$. $lim_{d \to \infty} k_{mul} = 4$.

From Fig. 3 we can see that the better fields for first variant are with characteristic d = 3, for second variant – it's are with characteristics d = 3, 5 and 7. For third variant fields with large characteristics have 4-time bigger hardware complexity than binary (Fig. 4).

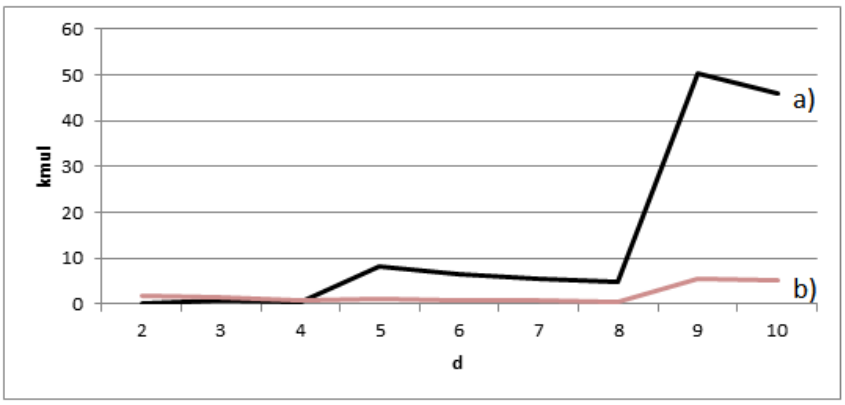

Fig.3. The ratio of hardware costs of multipliers of elements of Galois fields GF(d^m) and GF(2^n) according to the first – a) and second – b) variant

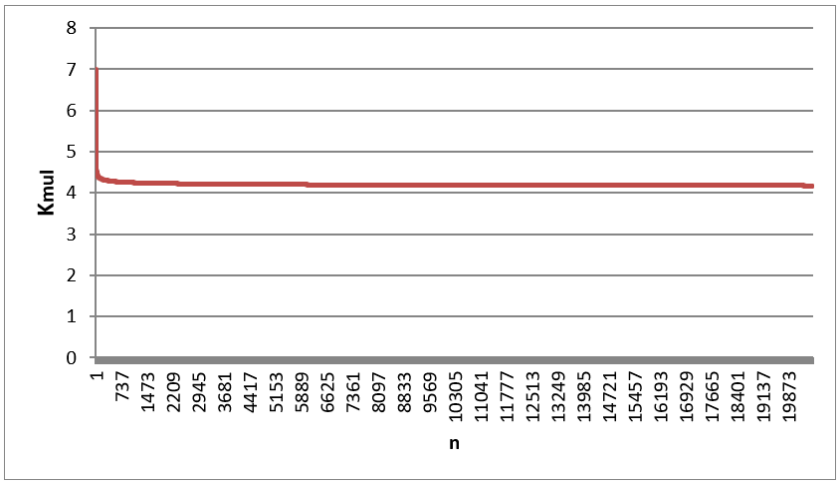

Fig.4. The ratio of hardware costs of multipliers of elements of Galois fields GF(d^m) and GF(2^n) according to the third variant

6. General Approach to the Design of Core Generators

VHDL [30] was chosen to develop kernel software models. Currently, there are many tools on the market for the development of integrated circuits on FPGA. However, they all have a significant drawback - design time. Therefore, project generation systems according to certain configuration parameters are becoming more and more relevant.

The multiplier is created from basic model device that can be configured by different parameters. Another approach to reducing design time is based on the use of high-level programming languages to describe hardware. Using this approach simplifies the design process and elevates it to the level of software design. This approach allows you to get both hardware and software to implement the task.

Significant disadvantages of modern design approaches are the use of specific high-level programming languages. This approach greatly complicates the design time and is based on the use of ready-made units, which greatly increases the number of resources used on the chip and increases the execution time of the program.

For many years, the efforts of VLSI developers [31] have been aimed at solving the problems of functional design at the level of register transfers [32]. Accordingly, the main investments were directed to the development of synthesis, modeling and verification tools at the RTL level. However, in recent years, VLSI and CAD developers have begun to pay attention to the system level of design. It's necessary due to the increase in the size of projects and the desire to reduce design costs. The main purpose of creating design tools at the system level is to simplify the design process, which assumes that the developer does not need to have perfect circuit design skills, but only be able to program in a high-level language.

Design tools must include the following requirements:

- ease of use. That is the friendliness of the interface and extensive functionality.
- ability to test the developed device.
- ability to estimate hardware costs, power consumption and performance of the synthesizied in FPGA device.
- ability to choose the best option. That is, the best one in terms of hardware costs, speed and power consumption.

Modern development systems have the following problems:

- the complexity of the design process. The developer is required to have a deep knowledge of the hardware on which the device will be implemented.
- design time.
- performance of the developed device.

To simplify and reduce design costs, ready-made circuit solutions are used. They are commonly called cores. In this approach, the device is designed on the basis of a ready-made and tested core. The core can be either the simple element of the processor or the entire processor. To ensure high performance in this approach, it is necessary that the structure of the core corresponds as accurately as possible to the task.

The core design process can be divided into two areas: creating a reconfigured unit and creating a configured one.

A reconfigured unit is an approach based on the use of a ready-made solution, with the ability to configure for a specific task, changing the values of certain parameters. Configured units are units that are created by a set of specific rules and algorithms, but they are created for a specific task.

7. Features of the Galois Fields Elements Multipliers Core Generators Design

There is a serious problem with creating Galois field elements multipliers - they are very large, but contain many similar units.

Next, you can consider comparing fields with the approximately equal orders but with different characteristics (Table 1). It is extremely difficult or almost impossible to manually create such multipliers. Therefore, it was decided to develop a Galois field multiplier generator for Galois fields with order approximately up to 2^{998} and with different characteristics.

The generator was implemented in C++. A block diagram of the generator is shown in Fig. 5. The generator implements multipliers by three variants:

- in base of Modified Guild Cell (MGC) which is presented as one whole element (a black box),
- in base of MGC which is presented as multiplier and adder,
- in base of MGC which is a circuit that consists of simple logic gates and implements the result of Boolean functions minimization.

Table 1. The number of modified guild cells in Galois field multipliers with approximately equal order and different characteristics

The Galois field GF(d^m) for which the FPGA multiplier is built	Galois field order (approximately, O_d)	Relative Galois field order, $RO_d=O_d/O_2$	Number of MGC (approximately, $N_C= m^2 + (m-1)^2$	Number of MGC inputs and outputs (approximately, $N_{IO} = N_C \times 4 \times \lceil \log_2 d \rceil$)
GF(2^{998})	2,68e+300	1,00	1990013	7960052
GF(3^{630})	3,86e+300	1,44	792541	6340328
GF(5^{430})	3,60e+300	1,34	368941	4427292
GF(7^{355})	1,02e+300	0,38	251341	3016092
GF(11^{289})	9,17e+300	3,42	166465	2663440
GF(13^{270})	5,82e+300	2,17	145261	2324176
GF(17^{244})	1.69e+300	0,63	118585	2371700
GF(19^{235})	3,21e+300	1,19	109981	2199620
GF(23^{221})	8,74e+300	3,27	97241	1944820
GF(31^{201})	5,80e+299	0,22	80401	1608020
GF(37^{191})	3,36e+299	0,13	72581	1741944
GF(41^{186})	9,50e+299	0,35	68821	1651704
GF(43^{184})	3,61e+300	1,35	67345	1616280
GF(47^{180})	9,49e+300	3,54	64441	1546584
GF(53^{174})	1,06e+300	0,40	60205	1444920

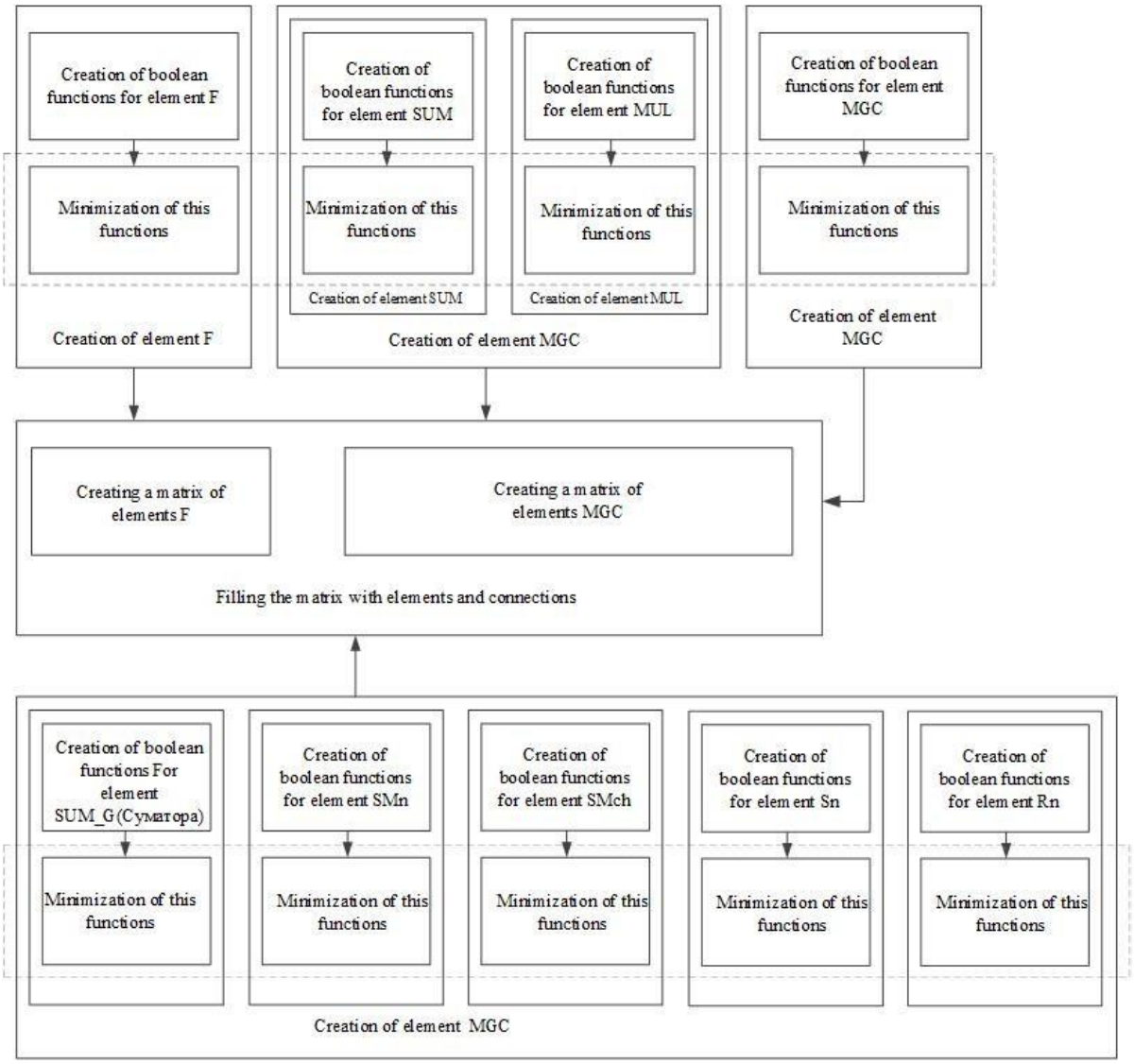

Fig.5. Block diagram of the Galois field multiplier generator

Structure of multiplier where MGC is presented as one whole element and as multiplier and adder is described in [9], where MGC consist of simple logic gates in [8].

The process of generating multipliers is divided into the following stages:

- Formation of Boolean functions for units;
- Minimization of these functions;
- Formation of units:

SUM (an adder);
MUL (a multilpier);
$F = (-G_m) \bmod d$, d – field characteristic, G_m – output of MGC or SUM or MUL;
SMn (an element that performs modular multiplication and addition and has result and carry outputs);
SMch (an element that performs the operation of adding or subtracting a number in the complementary code during division without restoring the remainder;
Sn (an element that defines the type of operation, subtraction or addition, during dividing without restoring remainders;
Rn (an element that determines whether one more addition operation is required to find the result (only for variants 2 and 3);

- Formation of MGC;
- Establishing connections between units (multiplier formation).

When forming Boolean functions, the generator automatically generates them according to the truth tables. Next step is Boolean functions minimization by Quine–McCluskey method. Then, on the basis of Boolean functions, units F, SUM, MUL, SMn, SMch, Sn, Rn [23] are formed in the VHDL description language. When forming a modified Guild cell, its VHDL description is generated. Then VHDL description of the multiplier is formed, which describes connections between a large number of MGC and F elements.

The Tab. 2 describes the software implementation of the generator. Table show classes for each variant separately. All data exchange is through files.

Table 2. Classes in GF generator for 3 structurs of multiplier.

MGC is a whole element	MGC consists of a multiplier and an adder	MGC consists of simple logic gates
transformation_and_minimization	transformation_and_minimization	transformation_and_minimization
generateFunctionsForF	generateFunctionsForF	generateFunctionsForF
create_f	create_f	create_f
MGC	MGC	MGC
create_MGC	create_MGC	create_MGC
generateFunctionsForMGC	-	-
MGC	MGC	MGC
create_MGC	create_MGC	create_MGC
creating_matrix	creating_matrix	creating_matrix
	generateFunctionsForMUL	generateFunctionsForSMch
	create_MUL	create_SMch
	generateFunctionsForSUM	generateFunctionsForSMn
	create_SUM	create_SMn
		generateFunctionsForSn
		create_Sn
		generateFunctionsForRn
		create_Rn
		generateFunctionsForAdder
		create_Adder

For 1st variant, Boolean functions of F and MGC elements are formed in the generateFunctionsForF and generateFunctionsForMGC classes, respectively. Generated functions are written to files. The transformation_and_minimization class then reads these functions from the file and minimizes them using the Quine-McCluskey method [33] and writes them back to these files. The create f and create MGC classes form VHDL templates for descriptions of F and MGC units based on minimized Boolean functions and write them to files. The felement and MGCclasses are the corresponding units F and MGC, from which the create_matrix class forms a matrix.

Classes MGC and F form connections between units.

For variant 2-unit F is formed similarly to variant 1. The MGC block is formed on the basis of MUL and SUM blocks, for which Boolean functions are formed in the generateFunctionForMUL and generateFunctionForSUM classes, respectively. These functions are written to a file. Then they are minimized by the transformation_and_minimization class. The Create_MUL and Create_SUM classes form a VHDL description of the MUL and SUM units. Next, the create_matrix class forms a multiplier.

For variant 3 the structure of the MGC is completely different, but the approach to creating the multiplication matrix is the same. This variant uses generateFunctionForSMch, generateFunctionForSMn, generateFunctionForSn, generateFunctionForRn, generateFunctionForSUM classes. The transformation_and_minimization class minimizes these functions. The classes Create_SMch, Create_SMn, Create_Sn, CreateRn and Create_SUM form VHDL descriptions of these units. The SMch and SMn units are complex because they form the matrix for large MGCs. Then, on the basis of the created MGC, a multiplier is formed.

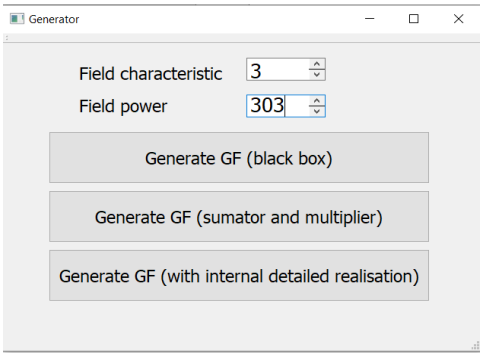

Fig.6. Galois field multiplier generator interface

In the Fig. 6 we can see the interface of the multiplier generator. It is possible to set the field characteristic as prime number from 2 to 2^{998} and the polinomial degree from 2 to 998, but with an order near 2^{998} (see Table. 1) or less. There are three buttons for generating according to three variants. The process of generating Galois field (see Table. 1) multipliers are quite time consuming and can take from a few minutes to several hours.

In Fig. 7 the scheme of synthesized in Xilinx Vivado multiplier GF(7^3) of the MGC which consists of a multiplier and an adder is shown. Fig. 8 shows the multiplier diagram.

Some multipliers were generated by the generator and the synthesized circuits in Xilinx Vivado environment were analyzed. We will analyze the generated multipliers in the implementation of the MGC as one element, the MGC consisting of a multiplier and an adder and MGC which is a circuit that consists of simple logic gates. A fairly large number of MGCs makes it very difficult to create these multipliers by hand. Millions of MGCs are practically used on multipliers (Table 1).

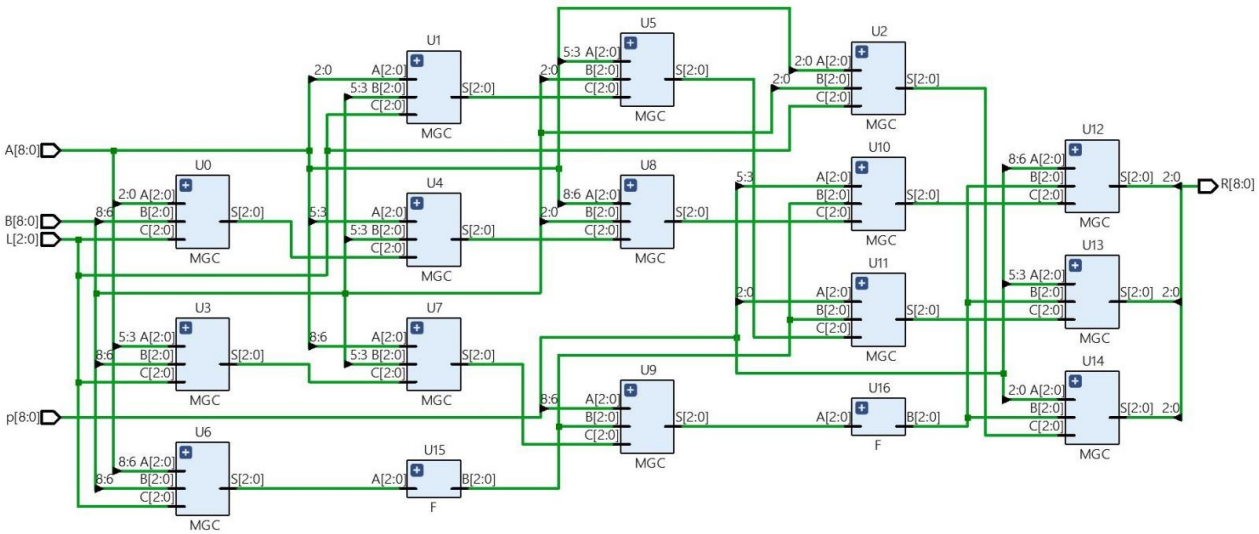

Fig.7. The scheme of the multiplier GF(7^3) in the implementation of the MGC as one element

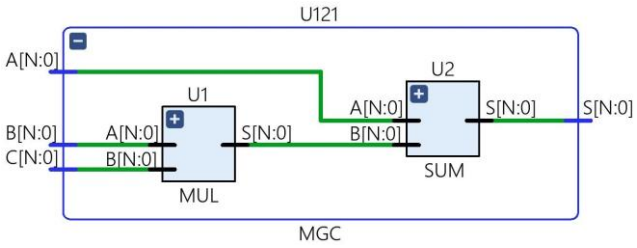

Fig.8. Scheme of the MGC in the implementation

8. Comparison of Hardware Complexity of Generated Multipliers

In the table 3 and 3a, the real results and theoretical results of generation of multiplier VHDL-descriptions for FPGA Virtex UltraScale+ XCVU9P [34] are presented. This FPGA has 2069000 LUTs. In the implementation of multipliers with MGC, which is a whole element, of their hardware costs increase rapidly with increasing field order. These costs are the lowest for Galois binary and ternary fields. In the implementation of the MGC, which consists of a multiplier and an adder, the hardware costs also increase rapidly with increasing field order. The lowest hardware costs for this architecture are for fields with characteristics d = 2, 3, 5, 7. In the implementation of MGC multipliers by architecture, when MGC consists of simple logic gates, the hardware costs increase with the increase of the field order. Therefore, according to three mentioned architectures it is possible to build any multiplier. Table shows multiplier generation time too. Time increases with raising field characteristic. All measurements were taken in computer with such characteristics:

CPU: Intel(R) Core(TM) i5-3570 CPU;
frequency: 3.40 GHz;
memory: 16 GB;
Windows 10, 64 bits.

Table 3. Multipliers hardware costs as LUT amount NR_d, NT_d and generation time of Galois field multipliers on for FPGA Virtex UltraScale+ XCVU9P that has 2069000 LUTs (for fields with relatively small order)

The field for which FPGA multiplier is built	d	Galois field order (approximately, O_d)	MGC is a whole element			MGC consists of a multiplier and an adder			MGC consists of simple logic gates		
			LUT amount, NR_d	Generation time, sec.	LUT amount, NT_d	LUT amount, NR_d	Generation time, sec.	LUT amount, NT_d	LUT amount, NR_d	Generation time, sec.	LUT amount, NT_d
GF(2^{50})	2	1,12E+15	2504	1,0	4901	2190	0,5	2450	18784	0,5	29208
GF(3^{32})	3	1,85E+15	4034	1,4	3970	4032	0,7	1984	17950	0,5	36510
GF(5^{22})	5	2,38E+15	19936	1,6	41625	5615	0,8	2768	17867	0,5	35049
GF(7^{18})	7	1,62E+15	16851	3,0	27585	3522	1,2	1837	16689	0,5	23366
GF(13^{14})	13	3,93E+15	32134	8,0	185420	10211	2,0	10216	15369	0,5	23658

Table 4 and Table 4a show a comparison of theoretical KT_{mul} and real KR_{mul} hardware costs of multiplier as their relationship with KT2 and KR2 costs of multiplier for binary fields when MGC are implementing according to 3 variants. $KT_{mul} = NT_d/NT_2$, $KR_{mul} = NR_d/NR_2$. It can be seen from the Table 4 and Table 4a and graphs Fig. 9, that when the MGC is implemented as a whole element, ternary Galois fields are 3 % better than binary ones. When implementing a multiplier with MGC that consists of a multiplier and an adder, then compared to a binary field, the field with characteristic 3 has a 11 % bigger cost index, the field with characteristic 5 has a 20 % bigger cost index, the field with characteristic 7 has a 18 % bigger cost index.

Table 4. Multipliers theoretical and real hardware costs comparison on for FPGA Virtex UltraScale+ XCVU9P that has 2069000 LUTs (for fields with relatively small order)

Field	d	Galois field order (O_d)	$C_o = \frac{O_d}{O_2}$	MGC is a whole element				MGC consists of a multiplier and an adder				MGC consists of simple logic gates			
				KT_{mul}	$\frac{KT_{mul}}{C_o}$	KR_{mul}	$\frac{KR_{mul}}{C_o}$	KT_{mul}	$\frac{KT_{mul}}{C_o}$	KR_{mul}	$\frac{KR_{mul}}{C_o}$	KT_{mul}	$\frac{KT_{mul}}{C_o}$	KR_{mul}	$\frac{KR_{mul}}{C_o}$
GF(2^{50})	2	1,12E+15	1	1	1	1	1	1	1	1	1	1	1	1	1
GF(3^{32})	3	1,85E+15	1,65	0,81	0,43	1,61	0,97	0,81	0,49	1,84	1,11	1,25	0.75	0,95	0,57
GF(5^{22})	5	2,38E+15	2,12	8,49	4	7,96	3,75	1,13	0,53	2,56	1,2	1,2	0,56	0.95	0,44
GF(7^{18})	7	1,62E+15	1,35	5,63	4,17	6,72	4,97	0,75	0,55	1,6	1,18	0,8	0,59	0,88	0,65
GF(13^{14})	13	3,93E+15	3,5	37,8	10,8	12,83	3,66	4,17	1,19	4,65	1,32	0,81	0,23	0.81	0,23

Table 3a. Multipliers hardware costs as LUT amount NR_d, NT_d and generation time of Galois field multipliers on for FPGA Virtex UltraScale+ XCVU9P that has 2069000 LUTs (for fields with relatively high order)

The field for which FPGA multiplier is built	d	Galois field order (approximately, O_d)	MGC is a whole element			MGC consists of a multiplier and an adder			MGC consists of simple logic gates		
			LUT amount, N_d	Generation time, sec.	LUT amount, NT_d	LUT amount, N_d	Generation time, sec.	LUT amount, NT_d	LUT amount, N_d	Generation time, sec.	LUT amount, NT_d
GF(2^{998})	2	2,67E+300	1013715	98	1990013	888165	52	995006	not fit	57	15920104
GF(13^{270})	13	5,81E+300	not fit	889	75222491	not fit	423	4149177	not fit	61	12895284

Table 4a. Multipliers theoretical and real hardware costs comparison on for FPGA Virtex UltraScale+ XCVU9P that has 2069000 LUTs (for fields with relatively high order)

Field	d	Galois field order (O_d)	$C_o = \frac{O_d}{O_2}$	MGC is a whole element				MGC consists of a multiplier and an adder				MGC consists of simple logic gates			
				KT_{mul}	$\frac{KT_{mul}}{C_o}$	KR_{mul}	$\frac{KR_{mul}}{C_o}$	KT_{mul}	$\frac{KT_{mul}}{C_o}$	KR_{mul}	$\frac{KR_{mul}}{C_o}$	KT_{mul}	$\frac{KT_{mul}}{C_o}$	KR_{mul}	$\frac{KR_{mul}}{C_o}$
GF(2^{998})	2	2,67E+300	1	1	1	1	1	1	1	1	1	1	1	1	1
GF(13^{270})	13	5,81E+300	3,5	37,8	10,8	-	-	4,17	1,19	-	-	0,81	0,23	-	-

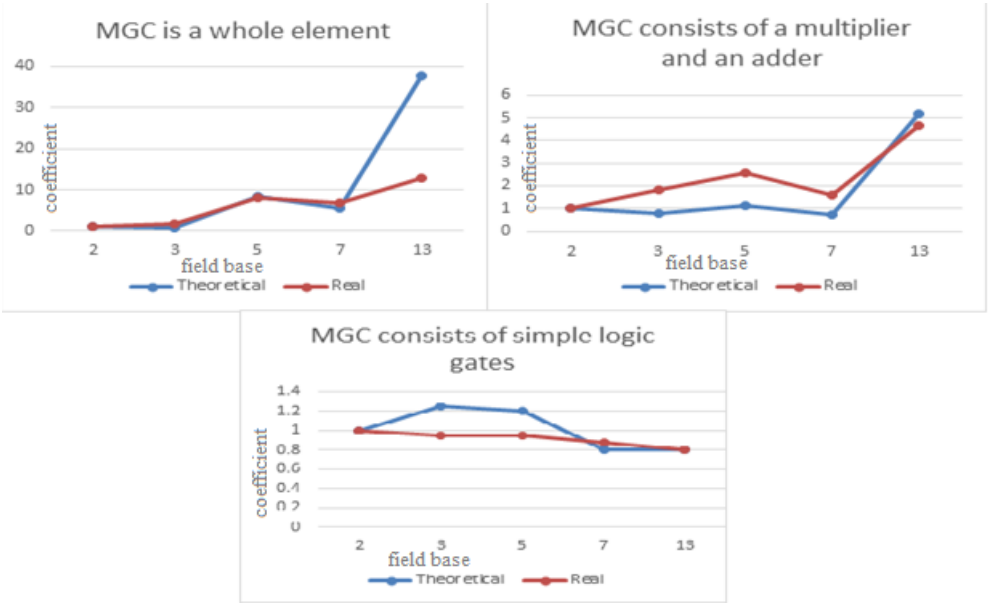

Fig.9. Comparison of theoretical and real hardware complexity coefficient in 3 variants of GF Multiplier realization

9. Conclusions and Future Scope

The paper summarized the theoretical foundations of creating Galois field multipliers on FPGAs. 3 variants for constructing Galois field multipliers were given, and a comparison of these variants was given. A tool was developed for generating VHDL-descriptions of multipliers of elements of such fields for further use of these descriptions in the implementation of data protection devices on FPGAs.

Hardware complexity was chosen as the criterion for comparing Galois field multipliers. The paper shows that extended Galois fields GF(d^m) with characteristics d > 2 are advisable to use in data protection tools, since the particular complexity of fields with characteristic d = 3 Galois fields is 3 % better than that of fields with characteristic d = 2 when implementing MGC as a whole element.

When the MGC consists of a multiplier and an adder, fields with characteristic d = 3 have 11 %, fields with characteristic d = 5 have 20 %, and fields with characteristic d = 7 have 18 % higher indicators of hardware complexity then binary fields.

The article shows the architecture and results of the Galois field multiplier core generator with different order up to 9,49e+300 elements. This generator allows you to generate multipliers VHDL-descriptions that would be almost impossible to create manually. The generator forms descriptions of multipliers that consist of MGC. 3 MGC construction variants are also proposed. The implementation of MGC as a whole element and multiplier and adder has advantages in fields with characteristics d = 2, 3, 5, 7.

The implementation of MGC as a matrix multiplier and adder has advantages in the implementation of a multiplier with large field characteristic. The generation time of vhdl-descriptions can reach 889 seconds. It is also planned to create pipeline cores.

References

[1] Quine, Willard Van Orman, "The Problem of Simplifying Truth Functions", The American Mathematical Monthly, Vol.59, No.8, pp. 521–531, 1952. DOI:10.2307/2308219. JSTOR 2308219

[2] Sudha Ellison, Mathe Lakshmi Boppana, “Bit-parallel systolic multiplier over GF(2^m) for irreducible trinomials with ASIC and FPGA implementations”, Department of Electronics and Communicating Engineering, National Institute of Technology, Varangal, Telangana 506004, India, IET Journal IET Circuits Devices Syst., Vol. 12, Iss. 4, pp. 315-325, 2018. DOI:10.1049/iet-cds.2017.0426

[3] H. El-Razouk, "Input-Latency Free Versatile Bit-Serial GF(2^m) Polynomial Basis Multiplication," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 5, pp. 589-602, May 2022, doi: 10.1109/TVLSI.2022.3155611.

[4] R. Bulat and M. R. Ogiela, "Personalized Cryptography Algorithms – A Comparison Between Classic and Cognitive Methods," 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S), 2022, pp. 43-44, doi: 10.1109/DSN-S54099.2022.00026.

[5] David Kohel, “Arithmetic statistics of Galois groups”, The open book series, 2019, V.1, pp. 353-374. https://msp.org/obs/2019/2-1/obs-v2-n1-p22-p.pdf

[6] M. A. Mehrabi, C. Doche and A. Jolfaei, "Elliptic Curve Cryptography Point Multiplication Core for Hardware Security Module," in IEEE Transactions on Computers, vol. 69, no. 11, pp. 1707-1718, 1 Nov. 2020, doi: 10.1109/TC.2020.3013266.

[7] X. Heng, J. Shen, N. Fan and W. Gao, "Fast Continuous Scalar Multiplication Algorithms on Twisted Edwards Elliptic Curve," 2022 International Conference on Networks, Communications and Information Technology (CNCIT), 2022, pp. 89-95, doi: 10.1109/CNCIT56797.2022.00022.

[8] Zholubak I., Hlukhov V., “Hardware costs of the Galois field GF(d^m) with large base”, “Computer Systems and Networks”, № 881, Publishing House of Lviv Polytechnic National University, Lviv, pp. 41 – 47, 2017. DOI: https://doi.org/10.23939/csn2017. 881.041

[9] V. Hlukhov, A. Kostyk, I. Zholubak, M. Rahma. “Galois Fields Elements Processing Units for Cryptographic Data Protection in Cyber-Physical Systems”, Advances in Cyber-Physical Systems, Volume 2, Number 2, Lviv Polytechnic National University, pp. 47 – 53, 2017. https://doi.org/10.23939/acps2017.02.047

[10] Rodrigue Elias, Valerii Hlukhov, Mohammed Rahma, Ivan Zholubak, “FPGA cores for fast multiplicative inverse calculation in Galois Fields”, Electrotechnic and computer systems, Odessa, pp. 227-233, 2018. DOI: https://doi.org/10.15276/eltecs.27.103.2018.26

[11] J. L. Imana, "LFSR-Based Bit-Serial GF(2^m) GF (2^m) Multipliers Using Irreducible Trinomials," in IEEE Transactions on Computers, vol. 70, no. 1, pp. 156-162, 1 Jan. 2021, doi: 10.1109/TC.2020.2980259.

[12] Hari Krishna Balupala, Kumar Rahul, Santosh Yachareni, “Galois Field Arithmetic Operations using Xilinx FPGAs in Cryptography”, IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp. 1-6, 2021. DOI: 10.1109/IEMTRONICS52119.2021.9422551

[13] Hariveer Inumarty, Mohamed Asan Basiri M., “Reconfigurable Hardware Design for Polynomial Galois Field Arithmetic Operations”, 24th International Symposium on VLSI Design and Test (VDAT), pp. 1-5, 2020. DOI: 10.1109/VDAT50263.2020.9190485

[14] Iván Jirón, Ismael Soto, Sebastián Gutiérrez, Raúl Carrasco, “Reed-Solomon codes over Galois fields of characteristic 3 for a VLC channel”, South American Colloquium on Visible Light Communications (SACVC), pp. 1-5, 2020. DOI: 10.1109/SACVLC50805.2020.9129896

[15] Fedir Geche, Oksana Mulesa, Veronika Voloshchuk, Anatoliy Batyuk, “Generalized Logical Neural Functions Over The Galois Field And Their Properties”, IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), pp. 21-24, 2019. DOI: 10.1109/STC-CSIT.2019.8929867

[16] Sergei Shalagin, Vjachtslav Zakharov, “Implementing the Markov Probability Functions Based on a Set of Polynomials over Galois Field”, International Conference on Information Technology and Nanotechnology (ITNT), pp. 1-3, 2021. DOI: 10.1109/ITNT52450.2021.9649171

[17] J. Xie, C. -Y. Lee, P. K. Meher and Z. -H. Mao, "Novel Bit-Parallel and Digit-Serial Systolic Finite Field Multipliers Over GF(2^m) Based on Reordered Normal Basis," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 9, pp. 2119-2130, Sept. 2019, doi: 10.1109/TVLSI.2019.2918836.

[18] Kashi Nath Datta, Sujoy Saha, “Binomial Galois Field Based Asynchronous Non-Adaptive Mode Energy Management System”, 11th International Conference on Communication Systems & Networks (COMSNETS), pp. 510-512, 2019. DOI: 10.1109/COMSNETS.2019.8711217

[19] Ju-Hwan Kim, Bo-Yeon Sim; Dong-Guk Han, “Optimized Power Consumption Model for Multiplication in Galois Field of AES”, International Conference on Platform Technology and Service (PlatCon), pp. 1-3, 2019. DOI: 10.1109/PlatCon.2019.8669430

[20] Ilya V. Chugunkov, Lilia D. Gatilova, Michael A. Ivanov, Bogdana V. Kliuchnikova, Alexander A. Kozlov, Evgeniy A. Salikov, “Computing in Finite Fields”, Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), pp. 273-276, 2022. DOI: 10.1109/ElConRus54750.2022.9755751

[21] Rahma, M., Zholubak, I., Hlukhov, V., “Devices for multiplicative inverse calculation in the binary Galois fields”, Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018, pp. 261–264, 2018. DOI: 10.1109/DESERT.2018.8409141

[22] Krishn Kumar Gupt, Meghana Kshirsagar, Joseph P. Sullivan, Conor Ryan, “Automatic Test Case Generation for Vulnerability Analysis of Galois Field Arithmetic Circuits”, IEEE 5th International Conference on Cryptography, Security and Privacy (CSP), pp. 32-37, 2021. DOI: 10.1109/CSP51677.2021.9357567

[23] Sergei Shalagin, Vjacheslav Zakharov, “Distributed Nonlinear-Polynomial Computing Based on a Group of Polynomials over a Galois Field in the FPGA Architecture”, IEEE 15th International Conference on Application of Information and Communication Technologies (AICT), pp. 1-4, 2021. DOI: 10.1109/AICT52784.2021.9620296
[24] H. K. Balupala, K. Rahul and S. Yachareni, "Galois Field Arithmetic Operations using Xilinx FPGAs in Cryptography," 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2021, pp. 1-6, doi: 10.1109/IEMTRONICS52119.2021.9422551.
[25] Rodriges Elias, Hlukhov V., Rahma M., Zholubak I., “Concurrent error detaction of devices for extended galois fields elements processing”, “Computer Systems and Networks”, № 905, Publishing House of Lviv Polytechnic National University, Lviv, pp. 64 – 72, 2018. DOI: https://doi.org/10.23939/csn2018.905.064
[26] DSTU 4145-2002 “Cryptographic protection of information. A digital signature based on elliptic curves”. https://itender-online.ru/wp-content/uploads/2017/09/dstu-4145-2002-1.pdf
[27] Rodrigue Elias, Valerii Hlukhov, Mohammed Rahma, Ivan Zholubak, “FPGA cores for fast multiplicative inverse calculation in Galois Fields”, Electrotechnic and computer systems, Odessa, pp. 227-233, 2018. DOI: https://doi.org/10.15276/eltecs.27.103.2018.26
[28] Melnyk A.O., Melnyk V.A., “Personal supercomputers”, Publishing House of the National University "Lviv Polytechnic", Lviv, 2012. - 600 pp.
[29] Sharma, A. Singh and A. Kumar, "Encryption and Decryption of Marker Based 3-Dimensional Augmented Reality Image Using Modified Hill Cipher Technique for Secure Transfer," 2022 IEEE 2nd International Conference on Computer Communication and Artificial Intelligence (CCAI), 2022, pp. 155-159, doi: 10.1109/CCAI55564.2022.9807727.
[30] Jiri Gaisler, "A structured VHDL Design Method", Retrieved 15 November 2017. https://gaisler.com/doc/structdesign.pdf
[31] L. Conway, "Reminiscences of the VLSI Revolution". Accessed 14, November 2019. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.693.5803&rep=rep1&type=pdf
[32] H. S. Choo et al., "Machine-Learning-Based Multiple Abstraction-Level Detection of Hardware Trojan Inserted at Register-Transfer Level," 2019 IEEE 28th Asian Test Symposium (ATS), 2019, pp. 98-980, doi: 10.1109/ATS47505.2019.00018.
[33] H. -G. Vu, N. -D. Bui, A. -T. Nguyen and ThanhBangLe, "Performance Evaluation of Quine-McCluskey Method on Multi-core CPU," 2021 8th NAFOSTED Conference on Information and Computer Science (NICS), 2021, pp. 60-64, doi: 10.1109/NICS54270.2021.9701506.
[34] P. Bhowmik, J. Hossain Pantho, J. Mandebi Mbongue and C. Bobda, "ESCA: Event-Based Split-CNN Architecture with Data-Level Parallelism on UltraScale+ FPGA," 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2021, pp. 176-180, doi: 10.1109/FCCM51124.2021.00028.

Authors’ Profiles

Ivan Zholubak is a Senior Lecturer of the Computer Engineering Department at Lviv Politechnic National University, Ukraine. He graduated from Lviv Politechnic National University with the engineer degree in Computer Engineering in 2013. In 2016 he graduated Postgraduate courses at Lviv Politechnic National University, department of Computer Engineering. He has scientific, academic and hands-on experience in the field of algorithms for hardware data protection in cryptography, robotic systems and AI. He is an author of 7 scientific papers.

Valerii Hlukhov is a professor of the Computer Engineering Departmentat at Lviv Polytechnic National University, Ukraine. He graduated from Lviv Polytechnic Institute with the engineer degree in Computer Engineering in 1977. In 1991 he obtained his Ph.D. at the Institute of Modeling Problems in Power Engineering of the National Academy of Science of Ukraine. He was recognized for his outstanding contributions into dedicated-purpose computer systems design as a Senior Scientific Researcher in 1995. He was awarded the academic degrees of Doctor of Technical Sciences in 2013 at Lviv Polytechnic National University. He became a Professor of Computer Engineering in 2014. He has scientific, academic and hands-on experience in the field of computer systems research and design proven contribution into IP Cores design methodology and high-performance reconfigurable computer systems design methodology. He is an experienced reseracher in computer data protection, including cryptographic algorithms, cryptographic processors design and implementation. Prof. Hlukhov is an author of more than 100 scientific papers, patents and monographs.

How to cite this paper: I. M. Zholubak, V. S. Hlukhov, "Galua Field Multipliers Core Generator", International Journal of Computer Network and Information Security(IJCNIS), Vol.15, No.3, pp.1-14, 2023. DOI:10.5815/ijcnis.2023.03.01