Multiplication in GF(2^m): area and time dependency/efficiency/complexity analysis.

Danuta Pamuła*/. Edward Hrynkiewicz*. Arnaud Tisserand****

** Silesian University of Technology, Gliwice, Poland:*
(e-mail edward.hrynkiewicz, danuta.pamula @polsl.pl)
*** IRISA, CNRS, INRIA Centre Rennes – Bretagne, University of Rennes*
(e-mail: arnaud.tisserand, danuta.pamula@irisa.fr)

Abstract: Efficient arithmetic units are crucial for cryptographic hardware design. The cryptographic systems are based on mathematical theories thus they strongly depend on the performance of the arithmetic units comprising them. If the arithmetic operator does not take a considerable amount of resources or is time non efficient it negatively impacts the performance of the whole cryptosystem. This work is intended to analyse the hardware possibilities of the algorithms performing multiplication in finite field extensions *GF(2^m)*. Such multipliers are used in Elliptic Curve Cryptography (ECC) applications. There are only two operations defined in the field: addition and multiplication. Addition is considered as a trivial operation – it is a simple bitwise XOR. On the other hand multiplication in the field is a very complex operation. To conform to the requirements of ECC systems it should be fast, area efficient and what is the most important perform multiplication of large numbers (100 - 600 bits). The paper presents analysis of *GF(2^m)* two-step modular multiplication algorithms. It considers classical (standard, school-book) multiplication, matrix-vector approach algorithm and Karatsuba-Ofman algorithm, exploring thoroughly their advantages and disadvantages.

Keywords: ECC, finite fields arithmetic, GF(2^m), computer arithmetic, Karatsuba-Ofman

1. INTRODUCTION

Cryptographic systems are getting more and more important and demanding nowadays. Various mathematical theories are exploited to make the cryptographic applications and devices suitable for data protection in today's computer systems, which are now spanning almost all domains of our lives, facilitating most of them. Not always consciously we are using many cryptographic protocols to secure or just to provide integrity of our digital data. Thus in order to conform to the continuously changing speed and security demands of computer system, the hardware designers have to provide efficient cryptosystem devices. To create such cryptosystems the designer needs to employ efficient arithmetic units.

Recently the cryptography based on elliptic curves defined over finite fields gained much popularity, especially due to the fact that in public key cryptosystems it allows to use smaller keys than RSA algorithms (algorithms based on classical mathematical theories) to provide the same or even higher security level. The longer are the keys the more difficult to handle them. It is not only harder to create units performing mathematical operation on long keys but also to store and transfer them.

The most popular finite fields (Lidl et al. (1994)) used for computations in Elliptic Curve Cryptography (ECC) are *GF(p)*, where *p* is a prime, and *GF(2^m)*, so called binary fields extensions. This work concerns arithmetic in *GF(2^m)* field. The intention of the authors is to help the hardware designers to choose the right algorithm for efficient implementation of *GF(2^m)* operations.

There are two main operations defined in the field: addition and multiplication. The addition operation is said to be trivial because in binary field it can be substituted with bitwise XOR operation. The case is different for multiplication in the field. It is considered as a complex operation. There exist also division operation in the field, which is even more complex than multiplication, but it is usually substituted by a set of multiplications and additions. The field requires multiplication modulo irreducible polynomial *f (x)* generating the field (Roman (2006)). The input operands are first multiplied and then the result of the multiplication needs to be reduced by an irreducible polynomial.

The article briefly presents some binary finite field theory elements necessary to understand the construction of the derived algorithms. Then it analyses a group of the multiplication algorithms investigating their advantages and disadvantages regarding cryptographic hardware design. The analysis targets FPGA devices as an implementation platform. In the last section the results of implementations are summarised and some conclusions are drawn.

2. BINARY FINITE FIELD EXTENSIONS ARITHMETICS

2.1 Binary finite field extensions arithmetic

A field comprises a set ***F*** and two operations: addition and multiplication. If set ***F*** is finite then the field is also finite.

The finite fields are also called Galois fields and denoted as $GF(q = p^m)$ or $F_{q=p^m}$, p is the so-called characteristic of the field. The two most studied cases in cryptography are prime fields $GF(p)$ (p is a prime number) and binary fields $GF(2^m)$ (where $p = 2$) (Hankerson et al. (2004)). To construct the binary finite field extension $F = GF(2^m)$ we use an irreducible polynomial $f(x)$ of degree m:

$$f(x) = x^m + f_{m-1}x^{m-1} + \ldots + f_2x^2 + f_1x + 1, \quad (1)$$

where $f_i \in GF(2)$. The field can be viewed as a vector space which elements are represented with a use of a specific basis – here polynomial basis: $\{1, x, x^2, \ldots, x^{m-2}, x^{m-1}\}$. Thus each element in the field can be represented as follows:

$$A = \sum_{i=0}^{m-1} a_i x^i = a_0 + a_1x + a_2x^2 + \ldots + a_{m-2}x^{m-2} + a_{m-1}x^{m-1},$$

where $a_i \in GF(2)$.

The three most commonly used bases for cryptographic purposes are *standard* (polynomial) basis, *normal* basis and *dual* basis. Each basis has its advantages and disadvantages and it is hard to say which one is the most suitable for the best operator solution. The polynomial basis yields the most regular structures and is the most popular, normal basis requires only a trivial shift operation to perform squaring and dual basis seems to be perfect for error-correcting codes. In many solutions authors tend to combine bases in order to exploit their advantages to create the most optimal designs.

In this work the authors consider only polynomial representation of the field elements.

Addition of two polynomials is carried out under modulo 2 arithmetic; thus it may be said that it is performed as the bitwise exclusive *OR*. This operation is regarded as a very simple one due to the fact that we do not need to bother for example about carry propagation. However if we XOR large numbers the operation, although simple, takes a lot of hardware resources.

Multiplication in a field is a more complex. Most problems creates the fact that it is a "modular" multiplication - the result of multiplication of two field elements is another element of the field. Multiplication is defined as polynomial product of two operands performed modulo irreducible generating polynomial $f(x)$. Let $A, B \in GF(2^m)$, be the *(m-1)* degree polynomials where

$$A = \sum_{i=0}^{m-1} a_i x^i = a_0 + a_1x + a_2x^2 + \ldots + a_{m-2}x^{m-2} + a_{m-1}x^{m-1}$$

and

$$B = \sum_{i=0}^{m-1} b_i x^i = b_0 + b_1x + b_2x^2 + \ldots + b_{m-2}x^{m-2} + b_{m-1}x^{m-1},$$

and let

$$F(x) = f(x) = x^m + f_{m-1}x^{m-1} + \ldots + f_2x^2 + f_1x + 1,$$

be an irreducible polynomial generating the field. Then

$$C = A \cdot B \bmod F(x), \quad (2)$$

where $C = c_0 + c_1x + c_2x^2 + \ldots + c_{m-2}x^{m-2} + c_{m-1}x^{m-1}$ is also an element of the field.

There exist many algorithms for performing multiplication in the binary finite field extensions (Erdem et al. (2006)). Generally they can be divided into two types: two-step algorithms and interleaved multiplication algorithms. Two-step algorithms perform modular multiplication in two steps, first the multiplication is done and then the result is reduced. Whereas interleaved multiplication algorithms perform simultaneously multiplication and reduction. This article analyses and compares only two-step algorithms. Interleaved algorithms are out of the scope of this work (Rodriguez-Henriquez et al. (2006)).

2.2 Two-step multiplication algorithms

Let *A, B, C* be polynomials of degree *(m-1)* belonging to a field $GF(2^m)$, and let $f(x)$ be an irreducible polynomial generating the field. We need also to define *D* the polynomial of degree *2m-2*. Thus in two-step modular multiplication we perform:

1) Multiplication $D = A \cdot B$;
2) Reduction $C = D \bmod F(x)$.

The most popular methods for performing two-step multiplications are: classical approach (school-book method), matrix-vector approach and Karatsuba-Ofman divide (Karatsuba, et al. (1963)) and conquer method. The reduction operation can be implemented by means of multiplication that is why its details are not considered here.

The hardware for all algorithms presented here was designed in VHDL. It was synthesised and implemented using Xilinx ISE 9.2 and the newest 12.1 environment, and targeted for Xilinx FPGA Spartan3E 1200. Such a small chip is suitable for our tests. The arithmetic operators used in cryptosystems should be not only time but also area efficient so if the solution for multiplication exceeds the size of Spartan device it means that it is not an area efficient design.

2.3 Standard multiplication algorithms

There are few approaches to standard multiplication most of them are based on shift-and-add method (Deschamps et al. (2009)). The method is based on the idea:

$$D = A \cdot B = \left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{i=0}^{m-1} b_i x^i\right) =$$

$$b_0\left(\sum_{i=0}^{m-1} a_i x^i\right) + b_1x\left(\sum_{i=0}^{m-1} a_i x^i\right) + b_2x\left(\sum_{i=0}^{m-1} a_i x^i\right) \quad (3)$$

$$+ \ \ldots \ +$$

$$b_{m-2}x^{m-2}\left(\sum_{i=0}^{m-1} a_i x^i\right) + b_{m-1}x^{m-1}\left(\sum_{i=0}^{m-1} a_i x^i\right)$$

which for $m = 4$ could be illustrated as follows: