International Journal of Security and Its Applications
Vol. 10, No. 9 (2016), pp.109-120
http://dx.doi.org/10.14257/ijsia.2016.10.9.12

Fast Hardware Implementations of Inversions in Small Finite Fields for Special Irreducible Polynomials on FPGAs

Haibo Yi [1], Weijian Li[2] and Zhe Nie[1],

[1]*School of Computer Engineering, Shenzhen Polytechnic. 518055 Shenzhen, China*
[2]*School of Computer Science, Guangdong Polytechnic Normal University. 510000 Guangzhou, China, weijianlee@126.com*
haiboyi@126.com, weijianlee@126.com, niezhe@szpt.edu.cn

Abstract

Inversions in small finite fields are the most computationally intensive field arithmetic and have been playing a key role in areas of cryptography and engineering. The main algorithms for small finite field inversions are based on Fermat's little theorem, extended Euclidean algorithm, Itoh-Tsujii algorithm and other methods. In this brief, we present techniques to exploit special irreducible polynomials for fast inversions in small finite fields $GF(2^n)$, *where* n *is a positive integer and* $0 < n < 16$. *Then, we propose fast inversions based on Fermat's theorem for two special irreducible polynomials in small finite fields, i.e. trinomials and All-One-Polynomials (AOPs). Trinomials can be represented by polynomials* $x^n + x^m + 1$ *and AOPs can be represented by polynomials* $x^n + x^{n-1} + ... + 1$, *where* m *is a positive integer and* $0 < m < n$. *Our designs have low hardware requirements, regular structures and are therefore suitable for hardware implementation. After that, our designs are programmed in Very-High-Speed Integrated Circuit Hardware Description Language (VHDL) by using integrated environment Altera Quartus II and implemented on a low-cost Field-Programmable Gate Array (FPGA). The experimental results on FPGAs show that our designs provide significant reductions in executing time of inversions in small finite fields, e.g. the executing time of inversion in* $GF(2^7)$ *is 18.80 ns and the executing time of inversion in* $GF(2^{12})$ *is 29.57 ns.*

Keywords*: Inversion, finite field, irreducible polynomial, trinomial, All-One-Polynomial (AOP), Field-Programmable Gate Array (FPGA), Very-High-Speed Integrated Circuit Hardware Description Language (VHDL), executing time, hardware implementation*

1. Introduction

Finite fields are widely used in in many areas, *e.g.*, cryptography, signal processing, error codes and clustered file system. Finite field arithmetic has gained increasing importance due to the fact that it is one of the most fundamental operations in such areas. Among finite field arithmetic, multiplications and inversions have been received continuous attentions. Efficient implementations of such arithmetic in finite fields are essential, *e.g.*, multiplication [1-10], inversion [11-30] and division [31-38]. More and more designs and implementations of multiplications, inversions and divisions in finite fields have been proposed.

Irreducible polynomials are one of the focuses of finite fields due to the fact that they are playing an important role in finite field arithmetic, *e.g.*, multiplication and inversion. [1] proposes a parallel multiplier for a special irreducible polynomial $x^n + x + 1$ in

ISSN: 1738-9976 IJSIA

composite fields, where n is a positive integer; [2] proposes a Mastrovito multiplier for trinomials in $GF(2^n)$, where n is a positive integer, trinomials can be represented by polynomials $x^n + x^m + 1$ and $m = 1, 2, ..., n-1$; [3] proposes a parallel multiplier for pentanomials in $GF(2^n)$, where n is a positive integer, pentanomials can be represented by polynomials $x^n + x^m + x^k + x^t + 1$ and $n > m > k > t > 0$; [4] proposes a Mastrovito multiplier for All-One-Polynomials (AOPs) and Equally-Spaced-Polynomials (ESPs) in $GF(2^n)$, where n is a positive integer, AOPs can be represented by polynomials $x^n + x^{n-1} + ... + 1$ and ESPs can be represented by polynomials $x^n + x^{n-2} + x^{n-4} + ... + 1$.

Designs and implementations of multiplications and inversions for special irreducible polynomials are very efficient. However, there are few designs of inversions for special irreducible polynomials.

The main algorithms for inversions in finite fields are based on Fermat's little theorem [11-20], extended Euclidean algorithm [21-23] and other methods [24-30]. In particular, inversions in small finite fields are crucial to many cryptographic systems, *e.g.*, AES [39-42], CLEFIA [43] and Multivariate Public Key Cryptography (MPKC) [44]. AES and CLEFIA use a Substitution-Box (S-Box) [45] in their algorithms, which is generated by determining the multiplicative inverse for a given element; MPKC requires inversions in solving systems of linear equations. Thus, it is desirable to improve inversions in small finite fields.

We present techniques to exploit special irreducible polynomials for fast inversions in $GF(2^n)$. We propose fast inversions based on Fermat's theorem for two special irreducible polynomials in $GF(2^n)$, trinomials and AOPs, where n is a positive integer, trinomials can be represented by polynomials $x^n + x + 1$ and AOPs can be represented by polynomials $x^n + x^{n-1} + ... + 1$.

Our design is well suited for Field Programmable Logic Arrays (FPGAs). FPGA is an integrated circuit designed to be configured by a customer or a designer after manufacturing. The FPGA configuration is generally specified using a Hardware Description Language (HDL), similar to that used for an Application-Specific Integrated Circuit (ASIC).

The trend on the usage of FPGAs is hardware acceleration, where one can use the FPGA to accelerate certain parts of an algorithm and share part of the computation between the FPGA and a generic processor, *e.g.,* cryptography, arithmetic, digital signal processing.

We back up the claims with implementations of our design on a low-cost Altera FPGA, which are programmed in Very-High-Speed Integrated Circuit Hardware Description Language (VHDL) by using integrated environment Altera Quartus II. The experimental results show that our designs provide significant reductions in executing time.

The rest of this paper is organized as follows: Section 2 introduces finite field, inversion and FPGA. Section 3 proposes fast inversions for special irreducible polynomials in small finite fields. Section 4 presents implementations of our design on a low-cost Altera FPGA. Section 5 presents conclusions of this paper.

2. Preliminaries

2.1. Finite Fields

In mathematics, a finite field is a field that contains a finite number of elements. As with any field, it is a set on which the basic operations of addition, multiplication and inversion have been defined.

The prime field $GF(p)$ of order and characteristic p is constructed as the integers modulo p, where p is a prime number. Thus, the elements are represented by integers in

the range $0, 1, \ldots p-1$. Given a prime power $q = 2^n$ with $n > 1$, the field $GF(q)$ can be explicitly constructed. One chooses first an irreducible polynomial f in $GF(2)[X]$ of degree n. Then the quotient ring $GF(q) = GF(2)[X]/f$ of the polynomial ring $GF(2)[X]$ by the ideal generated by f is a field of order q.

2.2. Inversions in Finite Fields

Suppose that a is an element in a finite field, the multiplicative inverse for a can be calculated a number of different ways.

Brute-force Search:

The simplest way of inversions in finite fields is brute-force search, *i.e.*, multiply a by every number in the finite field until the product is one.

Fermat's Little Theorem:

Fermat's little theorem is the basis for the Fermat primality test and is one of the fundamental results of elementary number theory. The theorem is named after Pierre de Fermat, who stated it in 1640. It is called the little theorem to distinguish it from Fermat's last theorem.

The theorem states that if p is a prime number, then for any integer a, the number $a^p - a$ is an integer multiple of p. In the notation of modular arithmetic, this is expressed as $a^p \equiv a \pmod{p}$.

For example, if $a = 2$ and $p = 7$, $2^7 = 128$ and $128 - 2 = 7 \times 18$ is an integer multiple of 7. If a is not divisible by p, Fermat's little theorem is equivalent to the statement that $a^{p-1} - 1$ is an integer multiple of p or in symbols $a^{p-1} \equiv 1 \pmod{p}$.

For example, if $a = 2$ and $p = 7$, then $2^6 = 64$ and $64 - 1 = 63$ is thus a multiple of 7. Since the nonzero elements of $GF(2^n)$ form a finite group with respect to multiplication, $a^{2^n-1} = 1$, thus the inverse of a is a^{2^n-2}.

Extended Euclidean Algorithm:

In arithmetic and computer programming, the extended Euclidean algorithm is an extension to the Euclidean algorithm, which computes, besides the greatest common divisor of integers a and b, the coefficients of Bézout's identity, that is integers x and y such that $ax + by = \gcd(a, b)$. It allows one to compute also, with almost no extra cost, the quotients of a and b by their greatest common divisor.

The modular inverse of a modulo m in finite fields can be found with the extended Euclidean algorithm. The algorithm finds solutions to Bézout's identity $ax + by = \gcd(a, b)$, where finite field elements a and b are given, and x, y and $gcd(a, b)$ are the integers that the algorithm discovers. Therefore, since the modular inverse in finite fields is the solution to $ax \equiv 1 \pmod{m}$, by the definition of congruence, $m \mid ax - 1$, which means that m is a divisor of $ax - 1$. This, in turn, means that $ax - 1 = qm$. Rearranging produces $ax - qm = 1$, with a and m given, x the inverse, and q an integer multiple that will be discarded. This is the exact form of equation that the extended Euclidean algorithm solves — the only difference being that $gcd(a, m) = 1$ is predetermined instead of discovered. Thus, a needs to be coprime to the modulus, or the inverse won't exist. This algorithm runs in time $O(log(m)2)$, assuming $|a| < m$, and is generally more efficient than exponentiation.

LUT:

Table 1. LUT in $GF(2^n)$ and LUT in $GF(p)$.

LUT in $GF(2^n)$		LUT in $GF(p)$	
Address	Inverse	Address	Inverse
$(000...010)_2$	i_2	$(000...010)_2$	i_2
$(000...011)_2$	i_3	$(000...011)_2$	i_3
...	...	...	...
$(111...110)_2$	i_{n-2}	$p-2$	i_{p-2}
$(111...111)_2$	i_{n-1}	$p-1$	i_{p-1}

LUT is a lookup-based method for inversions in small finite fields. It stores the inverses of finite field elements in a table, then it can obtain the inverse from the table for any element in a finite field.

Table 1 is LUTs in $GF(2^n)$ and $GF(p)$ respectively, where element i_j is inverse of its address (*i.e.*, $j^{-1} = i_j$) and $j = 2,3,..., p-2, p-1,..., n-2, n-1$.

The starting address is $(000...010)_2$ since we do not need to compute inversions of $(000...000)_2$ and $(000...001)_2$. The ending addresses $(111...111)_2$ and $p-1$ are the last elements in $GF(2^n)$ and $GF(p)$ respectively. The area of LUT is $n\times(2^n-2)$ bits if the inversion is in $GF(2^n)$ and $n\times(p-2)$ bits if it is in $GF(p)$. Other words, it has a space complexity of $O(n\times 2^n)$. Consequently, LUT is unsuitable for inversions in very large finite fields.

2.3. FPGA

FPGA is an integrated circuit designed to be configured by a customer or a designer after manufacturing. The FPGA configuration is generally specified using a HDL, similar to that used for an ASIC.

FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects, like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like *AND* and *XOR* logic gates. In most FPGAs, such as Altera and Xilinx FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

An FPGA can be used to solve any problem which is computable. This is trivially proven by the fact that FPGA can be used to implement a soft microprocessor, such as the Xilinx MicroBlaze or Altera Nios II. Their advantage lies in that they are sometimes significantly faster for some applications because of their parallel nature and optimality in terms of the number of gates used for a certain process.

FPGAs originally began as competitors to CPLDs and competed in a similar space, that of glue logic for PCBs. As their size, capabilities, and speed increased, they began to take over larger and larger functions to the point where some are now marketed as full systems on chips (SoC). Particularly with the introduction of dedicated multipliers into FPGA architectures in the late 1990s, applications which had traditionally been the sole reserve of DSPs began to incorporate FPGAs instead.

Another trend on the usage of FPGAs is hardware acceleration, where one can use the FPGA to accelerate certain parts of an algorithm and share part of the computation between the FPGA and a generic processor, *e.g.*, cryptography, arithmetic, digital signal processing.

Specific applications of FPGAs are the focuses in areas of engineering and cryptography, which include digital signal processing, software-defined radio, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas.

Traditionally, FPGAs have been reserved for specific vertical applications where the volume of production is small. For these low-volume applications, the premium that companies pay in hardware costs per unit for a programmable chip is more affordable than the development resources spent on creating a specific ASIC for a low-volume application. New cost and performance dynamics have broadened the range of viable applications.

3. Fast Inversions for Special Irreducible Polynomials in Small Finite Fields

3.1. Fast Inversions Based on Fermat's Theorem

First, let β be an element in small finite fields $GF(2^n)$. According to Fermat's theorem, we have

$$\beta^{-1} = \beta^{2^n-2}.$$

Since

$$2^n - 2 = \sum_{i=1}^{n-1} 2^i ,$$

we have

$$\beta^{-1} = \prod_{i=1}^{n-1} \beta^{2^i} .$$

It can be observed that the first step is to compute β^{2^i}, where p(x) is the irreducible polynomial in $GF(2^n)$.

For $i = 1,2,...,n-1$, we compute

$$\beta^{2^i} = \sum_{j=0}^{n-1} u_{ij} x^{2^i \times j} \ mod \ p(x).$$

If p(x) is chosen, for $i = 1,2,...,n-1$, we can compute

$$x^{2^i \times j} \ mod \ p(x) = \sum_{j=0}^{n-1} v_j x^j .$$

Accordingly, for $i = 1,2,...,n-1$, we can compute

$$\beta^{2^i} = \sum_{j=0}^{n-1} k_{ij} x^j .$$

The second step of inversion in finite fields is to multiply $n-1$ elements, *i.e.*, β^2, β^4, …, $\beta^{2^{n-1}}$.

Multiplication in finite fields is performed in two steps.

The first step is to perform the polynomial multiplication. The second step is to reduce modulo the irreducible polynomial.

Let $a(\mathrm{x}) = \sum_{i=0}^{n-1} a_i x^i$ and $b(\mathrm{x}) = \sum_{i=0}^{n-1} b_i x^i$ be elements in $GF(2^n)$, and

Table 2. Special Irreducible Polynomials in Finite Fields $GF(2^n)$.

Special irreducible polynomials	Polynomials Representations
Normal	$x^n + x^k + ... + x^t + ... + x^u + ... + 1$
Trinomial	$x^n + x^k + 1$
Trinomial	$x^n + x + 1$
Pentanomial	$x^n + x^k + x^t + x^u + 1$
AOP	$x^n + x^{n-1} + ... + 1$
ESP	$x^n + x^{n-2} + x^{n-4} + ... + 1$
ESP	$x^n + x^{n-3} + x^{n-6} + ... + 1$

$$c(x) = a(x) \times b(x) (\mathrm{mod}\,(p(x))) = \sum_{i=0}^{n-1} c_i x^i$$

be the expected multiplication result.

First, we compute v_{ij} for $i = 0,1,...,2(n-1)$ and $j = 0,1,...,n-1$ according to

$$x^i \bmod p(x) = \sum_{j=0}^{n-1} v_{ij} x^j.$$

Next, we compute S_i by AND logic gates for $i = 0,1,...,2(n-1)$ by

$$S_i = \sum_{j+k=i} a_j b_k.$$

After that, we compute c_i by XOR logic gates for $i = 0,1,...,n-1$ by

$$c_i = \sum_{j=0}^{2(n-1)} v_{ji} S_j.$$

Finally, the multiplication result is $c(\mathrm{x}) = \sum_{i=0}^{n-1} c_i x^i$.

In sum, it can be observed from the above computations that efficient irreducible polynomials can provide significant reductions in executing time of inversions in finite fields.

Special irreducible polynomials are summarized in Table 2, which include trinomials, AOPs and ESPs.

3.2. Fast Inversions for Trinomials $x^n + x + 1$

We present techniques to exploit special irreducible polynomials - trinomials $x^n + x + 1$ for fast inversions in small finite fields $GF(2^n)$.

Irreducible polynomials with the form of $x^n + x + 1$ in finite fields are summarized in Table 3, where some trinomials $x^n + x + 1$ cannot be chosen as irreducible polynomials, *e.g.*, $x^5 + x + 1$, $x^8 + x + 1$, $x^{10} + x^9 + ... + 1$, $x^{11} + x^{10} + ... + 1$, $x^{12} + x^{11} + ... + 1$.

Since $x^n + x + 1$ is chosen as the irreducible polynomial in small finite fields $GF(2^n)$, for $i = 1,2,...,n-1$, $x^{2^i \times j}$ can be computed as follows,

$$x^{2^i \times j} \ mod\ (x^n + x + 1) = \sum_{j=0}^{n-1} v_j x^j.$$

Then, for $i = 0,1,..., n-1$, we compute x^i as follows,

Table 3. Irreducible Polynomials with the Form of $x^n + x + 1$ in Small Finite Fields, where $1 < n < 13$.

Finite field	Irreducible polynomials $x^n + x + 1$
$GF(2^2)$	$x^2 + x + 1$
$GF(2^3)$	$x^3 + x + 1$
$GF(2^4)$	$x^4 + x + 1$
$GF(2^5)$	-
$GF(2^6)$	$x^6 + x + 1$
$GF(2^7)$	$x^7 + x + 1$
$GF(2^8)$	-
$GF(2^9)$	$x^9 + x + 1$
$GF(2^{10})$	-
$GF(2^{11})$	-
$GF(2^{12})$	-

$x^i \bmod p(x) = x^i$.

Next, for $i = n$, we compute x^i as follows,

$x^n \bmod p(x) = x + 1$.

After that, for $n < i < 2n-1$, we compute x^i as follows,

$x^i \bmod p(x) = x^{i-n+1} + x^{i-n}$.

It can be observed from the above computations that, compared with normal irreducible polynomials, the inversions in $GF(2^n)$ are more efficient when the irreducible polynomials are trinomials $x^n + x + 1$.

3.3. Fast Inversions for AOPs

We present techniques to exploit special irreducible polynomials - AOPs $x^n + x^{n-1} + ... + 1$ for fast inversions in $GF(2^n)$.

Irreducible polynomials with the form of $x^n + x^{n-1} + ... + 1$ in finite fields are summarized in Table 4, where some AOPs $x^n + x + 1$ cannot be chosen as irreducible polynomials, *e.g.*, $x^3 + x^2 + x + 1$, $x^5 + x^4 + ... + 1$, $x^7 + x^6 + ... + 1$, $x^8 + x^7 + ... + 1$, $x^9 + x^8 + ... + 1$, $x^{11} + x^{10} + ... + 1$.

Since $x^n + x^{n-1} + ... + 1$ is chosen as the irreducible polynomial in small finite fields $GF(2^n)$, for $i = 1,2,..., n-1$, $x^{2^i \times j}$ can be computed as follows,

$$x^{2^i \times j} \ mod \ (x^n + x^{n-1} + ... + 1) = \sum_{j=0}^{n-1} v_j x^j.$$

For $i = 0,1,..., n-1$, we compute x^i as follows,

$x^i \bmod p(x) = x^i$.

For $i = n$, we compute x^i as follows,

$x^n \bmod p(x) = x^{n-1} + x^{n-2} + ... + 1$.

For $n < i < 2n-1$, we compute x^i as follows,

$x^i \bmod p(x) = x^{i-n-1}$.

It can be observed from the above computations that, compared with normal irreducible polynomials, the inversions in $GF(2^n)$ are more efficient when the irreducible polynomials are AOPs $x^n + x^{n-1} + ... + 1$.

Table 4. Irreducible Polynomials with the Form of $x^n + x^{n-1} + ... + 1$ in Small Finite Fields, where $1 < n < 13$.

Finite field	Irreducible polynomials $x^n + x^{n-1} + ... + 1$
$GF(2^2)$	$x^2 + x + 1$
$GF(2^3)$	-
$GF(2^4)$	$x^4 + x^3 + ... + 1$
$GF(2^5)$	-
$GF(2^6)$	$x^6 + x^5 + ... + 1$
$GF(2^7)$	-
$GF(2^8)$	-
$GF(2^9)$	-
$GF(2^{10})$	$x^{10} + x^9 + ... + 1$
$GF(2^{11})$	-
$GF(2^{12})$	$x^{12} + x^{11} + ... + 1$

4. Implementation

In order to prove that our designs have low latency for inversions in small finite fields $GF(2^n)$, the designs are modeled in VHDL by using Altera Quartus II version 8.0 and implemented on EP2S130F1020I4 FPGA device, which is a member of Altera Stratix family.

Altera Quartus II is a programmable logic device design software produced by Altera. Quartus II enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Quartus includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation.

We implement our designs and Table 5 gives insight in the performance of the experimental results, which show that our designs provide significant reductions in executing time, *e.g.,* the executing time of inversion in $GF(2^7)$ is 18.80 ns and the executing time of inversion in $GF(2^{12})$ is 29.57 ns.

5. Conclusion

Among field operations, inversions in finite fields have been playing a key role in many areas, *e.g.,* cryptography, signal processing and clustered file system. In particular, inversions in small finite fields are crucial to many cryptographic systems, *e.g.,* AES, CLEFIA and MPKC. AES and CLEFIA use a S-Box in their algorithms, which is generated by determining the multiplicative inverse for a given element; MPKC requires inversions in solving systems of linear equations. Thus, it is desirable to improve inversions in small finite fields.

The main algorithms for inversions in small finite fields are based on Fermat's little theorem, extended Euclidean algorithm and other methods.

Designs and implementations of multiplications and inversions for special irreducible polynomials are very efficient. However, there are few designs of inversions for special irreducible polynomials.

We present techniques to exploit special irreducible polynomials for fast inversions in $GF(2^n)$. We propose fast inversions based on Fermat's theorem for two special irreducible polynomials in $GF(2^n)$, trinomials and AOPs, where n is a positive integer, trinomials can be represented by polynomials $x^n + x + 1$ and AOPs can be represented by polynomials $x^n + x^{n-1} + ... + 1$.

Our design is well suited for FPGA implementations. The trend on the usage of FPGAs is hardware acceleration, where one can use the FPGA to accelerate certain parts of an algorithm and share part of the computation between the FPGA and a generic processor, *e.g.*, cryptography, arithmetic, digital signal processing.

We back up the claims with implementations of our design for inversions in small finite fields on a low-cost Altera FPGA EP2S130F1020I4, which are programmed in VHDL by using integrated environment Altera Quartus II version 8.0.

The experimental results show that our designs provide significant reductions in executing time, *e.g.*, the executing time of inversion in $GF(2^7)$ is 18.80 ns and the executing time of inversion in $GF(2^{12})$ is 29.57 ns.

Table 5. Implementations of Inversions in Finite Fields for Special Irreducible Polynomials.

Finite Field	$x^n + x + 1$	Time (ns)	AOPs	Time (ns)	Normal	Time (ns)
$GF(2^2)$	$x^2 + x + 1$	8.39	$x^2 + x + 1$	8.39	$x^2 + x + 1$	8.39
$GF(2^3)$	$x^3 + x + 1$	8.64	-	-	$x^3 + x^2 + 1$	8.65
$GF(2^4)$	$x^4 + x + 1$	8.61	$x^4 + x^3 ... + 1$	8.61	$x^4 + x + 1$	8.61
$GF(2^5)$	-	-	-	-	-	-
$GF(2^6)$	$x^6 + x + 1$	8.87	$x^6 + x^5 + ... + 1$	8.87	$x^6 + x^4 + x^2 + x + 1$	8.88
$GF(2^7)$	$x^7 + x + 1$	18.80	-	-	$x^7 + x^6 + ... + x^2 + 1$	22.60
$GF(2^8)$	-	-	-	-	-	-
$GF(2^9)$	$x^9 + x + 1$	22.81	-	-	$x^9 + x^8 + x^6 + ... + 1$	25.06
$GF(2^{10})$	-	-	$x^{10} + x^9 + ... + 1$	25.40	$x^{10} + x^3 + x^2 + ... + 1$	27.61
$GF(2^{11})$	-	-	-	-	-	-
$GF(2^{12})$	-	-	$x^{12} + x^{11} + ... + 1$	29.57	$x^{12} + x^3 + x^2 + ... + 1$	31.19

Acknowledgments

This work is supported by Major Project of Educational Scientific Planning of Shenzhen (No. ybfz15141), Ideological and Political Education Project of Shenzhen Polytechnic (No. 801522z20018), Shenzhen Science and Technology Program under Grant (No. JCYJ20150617155357681), Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (No. 2014KQNCX177), 2016 Guangdong Province Public Welfare Research and Ability Construction Project (No. 2016A010101030) and Quality Engineering Project of Department of Education of Guangdong Province (2014).

References

[1] C. Paar, "A new architecture for a parallel finite field multiplier with low complexity based on composite fields", IEEE Transactions on Computers, vol. 45, no. 7, **(1996)**, pp. 856-861.
[2] B. Sunar and C. K. Koc, "Mastrovito multiplier for all trinomials", Computers, IEEE Transactions on, vol. 48, no. 5, **(1999)**, pp. 522-527.
[3] F. Rodriguez-Henrquez and C. K. Koc, "Parallel multipliers based on special irreducible pentanomials", IEEE Transactions on Computers, **(2003)**, pp. 1535-1542.
[4] A. Halbutogullari and C.K. Koc, "Mastrovito multiplier for general irreducible polynomials", Computers, IEEE Transactions on, vol. 49, no. 5, **(2000)**, pp. 503-518.
[5] M. A. Hasan, "Look-up table-based large finite field multiplication in memory constrained cryptosystems", IEEE Transactions on Computers, vol. 49, no. 7, **(2000)** July, pp. 749-758.
[6] R. Azarderakhsh and M. Mozaffari-Kermani, "High-Performance Two-Dimensional Finite Field Multiplication and Exponentiation for Cryptographic Applications", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 10, **(2015)** Oct., pp. 1569-1576.
[7] A. H. Namin, H. Wu and M. Ahmadi, "Comb Architectures for Finite Field Multiplication in $F(2^m)$", in IEEE Transactions on Computers, vol. 56, no. 7, **(2007)** July, pp. 909-916.
[8] I. Cascudo, R. Cramer, C. Xing and A. Yang, "Asymptotic Bound for Multiplication Complexity in the Extensions of Small Finite Fields", in IEEE Transactions on Information Theory, vol. 58, no. 7, **(2012)** July, pp. 4930-4935.
[9] R. Katti and J. Brennan, "Low complexity multiplication in a finite field using ring representation", in IEEE Transactions on Computers, vol. 52, no. 4, **(2003)** April, pp. 418-427.
[10] M. C. Mekhallalati, M. K. Ibrahim and A. S. Ashur, "New low complexity bidirectional systolic structures for serial multiplication over the finite field $GF(q^m)$", in IEE Proceedings - Circuits, Devices and Systems, vol. 145, no. 1, **(1998)**, Feb., pp. 55-60.
[11] S. T. J. Fenn, M. Benaissa and D. Taylor, "Fast normal basis inversion in $GF(2^n)$", Electronics Letters, vol. 32, **(1996)** August, pp. 1566-1567.
[12] C. Rebeiro, S. S. Roy, D. S. Reddy and D. Mukhopadhyay, "Revisiting the Itoh-Tsujii inversion algorithm for FPGA platforms", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 8, **(2011)**, pp. 1508-1512.
[13] R. Azarderakhsh, K. Jarvinen and V. Dimitrov, "Fast inversion in $GF(2^m)$ with normal basis using hybrid-double multipliers", IEEE Transactions on Computers, vol. 63, no. 99, **(2012)**, pp. 1041-1047.
[14] L. Parrilla, A. Lloris, E. Castillo and A. Garcia, "Minimum-clock-cycle Itoh-Tsujii algorithm hardware implementation for cryptography applications over $GF(2^m)$ fields", Electronics Letters, vol. 48, no. 18, **(2012)**, pp. 1126-1128.
[15] S. T. J. Fenn, M. Benaissa and D. Taylor, "Finite field inversion over the dual basis", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 4, no. 1, **(1996)** Mar., pp. 134-137.
[16] M.-H. Jing, J.-H. Chen, Z.-H. Chen and Y.-H. Chen, "Low complexity architecture for multiplicative inversion in $GF(2^m)$", In IEEE Asia Pacifc Conference on Circuits and Systems, APCCAS 2006, Singapore, IEEE, Washington, DC, USA, **(2006)** December 4-7, pp. 1492-1495.
[17] AV Dinh, RJ Bolton and R Mason, "A low latency architecture for computing multiplicative inverses and divisions in $GF(2^m)$", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 48, no. 8, **(2001)**, pp. 789-793.
[18] C. C. Wang, TK Troung, H. M. Shao, L. J. Deutsch, J. K. Omura and I. S. Reed, "VLSI architectures for computing multiplications and inverses in $GF(2^m)$", IEEE Transactions on Computers, C-34, no. 8, **(1985)**, pp. 709-717.
[19] A V. Dinh, R. J. Palmer, R. J. Bolton and R. Mason, "A low latency architecture for computing multiplicative inverses and divisions in $GF(2^m)$", In 2000 Canadian Conference on Electrical and Computer Engineering, Halifax, NS, Canada, volume 1, IEEE, Washington, DC, USA, **(2000)** March 07-10, vol. 1, pp. 43-47.
[20] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases", Information and computation, vol. 78, no. 3, **(1988)**, pp. 171-177.
[21] A. K. Daneshbeh and M. A. Hasan, "A class of unidirectional bit serial systolic architectures for multiplicative inversion and division over $GF(2^m)$", IEEE Transactions on Computers, vol. 54, no. 3, **(2005)** Mar., pp. 370- 380.
[22] Z. Yan and D.V. Sarwate, "New systolic architectures for inversion and division in $GF(2^m)$, IEEE Transactions on Computers, vol. 52, no. 11, **(2003)** Nov., pp. 1514-1519.

[23] C.-T. Huang and C.-W. Wu, "High-speed easily testable Galois field inverter", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 47, no. 9, **(2000)** Sep., pp. 909-918.
[24] Jr. Kaliski, "B.S: The Montgomery inverse and its applications", IEEE Transactions on Computers, vol. 44, no. 8, **(1995)** Aug., pp. 1064-1065.
[25] E. Savas, "A carry-free architecture for Montgomery inversion", IEEE Transactions on Computers, vol. 54, no. 12, **(2005)** Dec., pp. 1508- 1519.
[26] J. Bajard, L. Imbert and C. Negre, "Arithmetic operations in finite fields of medium prime characteristic using the lagrange representation", IEEE Transactions on Computers, vol. 55, no. 9, **(2006)** Sept., pp. 1167-1177.
[27] C. McIvor, M. McLoone and J. V. McCanny, "Improved Montgomery modular inverse algorithm", Electronics Letters, vol. 40, 18, **(2004)** Sept., pp. 1110-1112.
[28] S.-W. Wei, "VLSI architectures for computing exponentiations, multiplicative inverses and divisions in $GF(2^m)$ ", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 44, no. 10, **(1997)** Oct., pp. 847-855.
[29] C.-L. Wang and J.-L. Lin, "A systolic architecture for computing inverses and divisions in finite fields $GF(2^m)$ ", IEEE Transactions on Computers, vol. 42, no. 9, **(1993)** Sep., pp. 1141-1146.
[30] M. A. Hasan and V. K. Bhargava, "Bit-serial systolic divider and multiplier for finite fields $GF(2^m)$ ", IEEE Transactions on Computers, vol. 41, no. 8, **(1992)** Aug., pp. 972-980.
[31] Z. Yu, "Low power finite field multiplication and division in re-configurable Reed-Solomon codec", Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on, **(2002)**, vol. 5, pp. V-785-V-788.
[32] Z. Yan and D. V. Sarwate, "Systolic architectures for finite field inversion and division", Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on, **(2002)**, vol. 5, pp. V-789-V-792.
[33] R. Furness, M. Benaissa and S. T. J. Fenn, "Finite field division based on recursive division algorithm and composite fields", in Electronics Letters, vol. 34, no. 18, **(1998)**, Sep. 3, pp. 1730-1731.
[34] J. P. Deschamps and G. Sutler, "Finite field division implementation", International Conference on Field Programmable Logic and Applications, 2005., **(2005)**, pp. 670-674.
[35] Y.-J. Jeong and W. Burleson, "VLSI array synthesis for polynomial GCD computation and application to finite field division", in IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 41, no. 12, **(1994)** Dec., pp. 891-897.
[36] L. Song and K. K. Parhi, "Low-energy software Reed-Solomon codecs using specialized finite field datapath and division-free Berlekamp-Massey algorithm", Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on, Orlando, FL, **(1999)**, vol. 1, pp. 84-89.
[37] K. D. Rao, C. Gangadhar and P. V. M. Krishna, "VLSI realization of a secure filter bank based transmultiplexer for images using MCKBA and finite field wavelet packet division multiplexing", 2012 Annual IEEE India Conference (INDICON), Kochi, **(2012)**, pp. 053-058.
[38] S. T. J. Fenn, M. Benaissa and D. Taylor, "$GF(2^m)$ multiplication and division over the dual basis", in IEEE Transactions on Computers, vol. 45, no. 3, **(1996)** Mar., pp. 319-327.
[39] Canright D., "A Very Compact S-Box for AES [J]", Lecture Notes in Computer Science, no. 3659, **(2005)**, pp. 441-455.
[40] Ahmad N. and Hasan S. M. R., "Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate [J]", Integration the VLSI Journal, vol. 46, no. 4, **(2013)**, pp. 333–344.
[41] Batina L., Jakobovic D. and Mentens N., "S-box Pipelining Using Genetic Algorithms for High-Throughput AES Implementations: How Fast Can We Go?[M]",// Progress in Cryptology -- INDOCRYPT 2014. Springer International Publishing, **(2014)**, pp. 322-337.
[42] Satoh A., Morioka S. and Takano K., "A Compact Rijndael Hardware Architecture with S-Box Optimization [M]", // Advances in Cryptology — ASIACRYPT 2001. **(2000)**, pp. 239-254.
[43] Shirai T., Shibutani K. and Akishita T., "The 128-bit blockcipher CLEFIA [C]", // Proceedings of the 14th international conference on Fast Software Encryption. Springer-Verlag, **(2007)**, pp. 181-195.
[44] Ding J. and Yang B. Y., "Multivariate Public Key Cryptography [M]", // Post-Quantum Cryptography, **(2009)**, pp. 193-241.
[45] Zeng J., Wang Y. and Xu C., "Improvement on masked S-box hardware implementation [C]",// International Conference on Innovations in Information Technology, **(2012)**, pp. 113-116.

