Computers and Electrical Engineering 61 (2017) 104–115

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Unified systolic array architecture for finite field multiplication and inversion☆

Atef Ibrahim[a,b,d,*], Turki Alsomani[c], Fayez Gebali[d]

[a] Prince Sattam Bin Abdulaziz University, Al-Kharj 11924, Riyadh, Saudi Arabia.
[b] Electronics Research Institute, El-Tahrir Street, Dokki 12622, Cairo, Egypt.
[c] Computer Engineering Department, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[d] ECE Department, University of Victoria, Victoria, BC V8W 2Y2, Canada

ARTICLE INFO

Article history:
Received 27 August 2016
Revised 11 June 2017
Accepted 12 June 2017

Keywords:
Finite field inversion
Cryptosystems
Systolic arrays
Hardware security
Resource-constrained applications
ASIC

ABSTRACT

This paper proposes a new unified systolic array architecture to perform multiplication and inversion operations in $GF(2^m)$ based on the bit serial multiplication algorithm and the previously modified extended Euclidean algorithm. This architecture is explored by applying a regular technique to the multiplication and inversion algorithms. It has lower area and power complexities as well as it achieves a moderate speed. Also, it has a simple structure with processing elements have local communication with each other. The implementation results of the proposed design and the comparable published designs show that the proposed design saves more area (ranging from 18.8% to 23.0%) and saves more energy (ranging from 18.2% to 47.0%) over the compared efficient designs. This makes it more suitable for applications that impose more constraints on area and power consumption.

1. Introduction

In resource-constrained devices, the application of public key cryptography (PKC) is limited due to the limitations of Power consumption. Since both Elliptic-curve cryptography (ECC) and Hyper Elliptic Curve Cryptography (HECC) algorithms have the merit of giving the same level of security using smaller key sizes comparing to other PKC algorithms, the ECC and HECC algorithms can be used in applications that have more constraints on power and timing [1]. There are recently more ECC hardware implementations reported in the literature that meets the power and timing constraints of these applications [2–4]. The essential operation used in both ECC and HECC based protocols is the scalar multiplication operation. This operation is performed as a sequence of point addition and point doubling in the case of ECC-based protocols and performed as a sequence of divisor addition and divisor doubling in the case of HECC-based protocols. Both point/divisor operations are carried out as a sequence of field addition, field multiplication, and field inversion. Compared to the field addition, the implementations of field multiplication and inversion are most expensive in both ECC and HECC cryptography. Accelerating these operations using software implementations is based on using a pre-computed look-up tables that lead to increasing the memory overhead. Also, the insufficient amount of available memory in resource-constrained applications limits the field size and leads to different performance across the architectures. On the other hand, accelerating these computations

☆ Reviews processed and approved for publication by Editor-in-Chief.

* Corresponding author.
E-mail addresses: attif_ali2002@yahoo.com, atef@ece.uvic.ca (A. Ibrahim), tfsomani@uqu.edu.sa (T. Alsomani), fayez@uvic.ca (F. Gebali).

http://dx.doi.org/10.1016/j.compeleceng.2017.06.014
0045-7906/

using hardware accelerators does not put any restrictions on the memory or field sizes and achieves a better performance than software implementations. Moreover, the significant reduction in computation time in hardware-based solutions leads to a significant saving in energy over the software-based solutions. Thus, in this work, we concentrate on the hardware-based solutions by presenting a unified hardware structure to perform multiplication and inversion operations. Sharing the data-path between the two operations saves more area resources and power than the case of using separate data-path for each operation and thus makes the design more suitable for constrained implementations of cryptographic primitives in resource-constrained embedded applications.

There are various algorithms and systolic architectures proposed for modular multiplication in $GF(2^m)$ [5,6]. The multiplier bit-serial algorithms can be classified into two categories, the most significant bit (MSB) first algorithms and the least significant bit (LSB) first algorithms. It is important to indicate that LSB-first bit-serial multiplier has shorter critical path than MSB-first bit-serial multipliers [5]. Also, these algorithms are iterative algorithms that make it easy to be implemented using systolic array approaches. Systolic arrays are also used to implement inversion in GF(2^m) [7–10].

A unified bit-serial multiplication/division structure was proposed in [11] based on the Stein's algorithm [12]. This unified bit-serial structure reduces the area by 27% compared to the structures consisting of one separated divider and multiplier. Nonetheless, the integration of multiplier and divider increases the critical path delay and this leads to slowing down both operations. Also, a unified bit-serial multiplication/inversion structure was proposed in [7] based on the modified extended Euclidean-based algorithm [8].

In this paper, we propose a unified systolic architecture that is suitable for resource-constrained devices to perform multiplication and inversion operations in $GF(2^m)$ based on the LSB-first bit-serial multiplication algorithm and the previously modified extended Euclidean algorithm [7,8]. This unified architecture is composed of a one-dimensional array of logic cells and has area complexity of $O(m)$. Since the inversion and multiplication operations share the same data path, they save much more area and power than architectures with separate inverters and multipliers as well as they have the same critical path delay as the stand alone inverter. The unified architecture is explored by applying a regular technique proposed by the third author [13–17] to the multiplication and inversion algorithms.

The paper is organized as follows: Section 2 gives a brief discussion about the LSB-First bit serial multiplication algorithm and the previously modified extended Euclidean-based inversion algorithm over GF(2^m) and the conversion of these algorithms into the bit-level form. Section 3 gives a brief discussion of the proposed methodology employed to explore the unified systolic array architecture. Section 4 explains the proposed design complexity and compares it to the previous work. At the end, Section 5 provides the conclusions of this work.

2. Finite field multiplication and inversion

This section provides a brief discussion about the LSB-first bit-serial multiplication and the previously modified extended Euclidean-based inversion algorithms over GF(2^m) and the conversion of these algorithms into the bit-level form.

2.1. Finite field multiplication

A finite field over GF(2^m) could be defined using the irreducible polynomial:

$$Q(x) = x^m + q_{m-1}x^{m-1} + \cdots + q_i x^i + \cdots q_2 x^2 + q_1 x + 1 \tag{1}$$

where $q_i \in GF(2)$ for $0 < i < m$.

Field elements A and B in $GF(2^m)$ can be represented by the polynomial:

$$A(x) = \sum_{i=0}^{m-1} a_i\ x^i \tag{2}$$

$$B(x) = \sum_{i=0}^{m-1} b_i\ x^i \tag{3}$$

where a_i and $b_i \in GF(2)$ for $0 \le i < m$.

Multiplication in $GF(2^m)$ is defined as polynomial multiplication of A(x) and B(x) modulo Q(x). The product T can be computed using the LSB-first multiplication algorithm shown in Algorithm 1 [5]. This algorithm computes two intermediate polynomials, $T(x)$ and $S(x)$ that are stored in m-bit registers. Here we use T^i and S^i to denote the values of T and S after ith iteration. b_i represents the ith coefficient of the multiplier B. In the initial step of the algorithm, the coefficients of the irreducible polynomial $Q(x)$ and the coefficients of polynomial $A(x)$ are assigned to the variables R^0 and S^0, respectively. Also, the variable T^0 is assigned zero values.

Algorithm 2 shows the conversion of Algorithm 1 into bit-level form. In this algorithm, the terms s_j^i and t_j^i represent the jth bit of S and T at iteration i, respectively. In each iteration of the outer for loop, control bit $c1^i$ is set to the value of the MSB of S^{i-1}, s_{m-1}^{i-1}.