Microprocessors and Microsystems 40 (2016) 45–52

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Low space-complexity and low power semi-systolic multiplier architectures over GF(2^m) based on irreducible trinomial

Fayez Gebali [a], Atef Ibrahim [a,b,c,*]

[a] University of Victoria, Victoria, BC, Canada
[b] Prince Sattam Bin AbdulAziz University, Alkharj, Saudi Arabia
[c] Electronics Research Institute, Cairo, Egypt

ARTICLE INFO

Keywords:
Trinomial multiplier
Finite field multiplication
Systolic arrays
Parallel architectures
Pipeline processing
VLSI

ABSTRACT

This paper proposes a three bit-serial and digit-serial semi-systolic $GF(2^m)$ multipliers using Progressive Product Reduction (PPR) technique. These architectures are obtained by converting the $GF(2^m)$ multiplication algorithm into an iterative algorithm using systematic techniques for scheduling the computational tasks and mapping them to Processing Elements (PE). Three different semi systolic arrays were obtained. ASIC implementation of the proposed designs and previously published schemes were used to verify the performance of the proposed designs. One proposed design has at least 29% lower area compared to previously published bit/digit serial multipliers. This design has also at least 70% lower power compared to previously published bit/digit serial multipliers. Another proposed design has at least 12% lower power-delay product (PDP) compared to previously published bit/digit serial multipliers. This makes the proposed designs more suited to resource-constrained embedded applications.

1. Introduction

Efficient arithmetic operations in finite fields are important in many applications, including coding theory, computer algebra systems, information theory, number theory, and public-key cryptosystems e.g., elliptic curve cryptosystems (ECC) [1]. Multiplication over $GF(2^m)$ is the basic field operation which is frequently encountered in most of these applications. In the literature, there are many multipliers with different bases, e.g., polynomial basis, normal basis, and dual basis. Numerous hardware architectures have been proposed for polynomial-basis finite-field multiplication over $GF(2^m)$ [2–28]. In terms of design style, all of these architectures can be classified into two basic forms. The first form is systolic or semi-systolic architecture and the second form is nonsystolic architecture. Nonsystolic designs aim to reduce the number of partial products to realize the multipliers with the least hardware and shortest latency [2–9]. On the other hand, the systolic architecutres [10–29] posses a lot of advantages over the nonsystolic ones due to the following reasons: processor element (PE) modularity, interprocessor communication regularity and locality, simple PE control, and high-throughput rates due to piplelining [28–30].

There are five main contributions in this work. (1) We propose a new multiplication technique in $GF(2^m)$ that starts by performing the reduction operation on the most significant partial product progressing to the lower-order partial products. (2) We provide formal linear and nonlinear techniques for scheduling the operations of the multiplication algorithm. (3) We provide formal linear and nonlinear techniques for assigning the operations of the multiplication algorithm to processors. (4) we proposed several semi-systolic architectures over GF(2^m) based on irreducible trinomial using Progressive Product Reduction (PPR) technique. (5) We verified the performance of the proposed designs and the other published designs by ASIC implementations of all designs. The ASIC implementations' results show that the proposed designs have lower area and power that makes them more suited to resource-constrained embedded applications.

The rest of the paper is organized as follows: Section 2 discusses finite field multiplication over $GF(2^m)$ based on irreducible trinomials. Section 3 discusses converting the trinomial based field multiplier algorithm to an iterative multiplication algorithm using Progressive Product Reduction (PPR) technique. Section 4 shows how to parallelize the PPR iterative multiplication algorithm using linear data scheduling and using linear and nonlinear assignment of computation task to PEs. Section 5 discusses design space exploration for the PPR iterative multiplication algorithm. Section 6 discusses the complexity of the proposed designs as well as that of previously published designs. Finally, Section 7 provides the conclusions of this work.

* Corresponding author Prince Sattam Bin Abdulaziz University, Alkharj, Saudi Arabia. Tel.: +96 6546825905.
E-mail addresses: fayez@ece.uvic.ca (F. Gebali), atef@ece.uvic.ca, attif_ali2002@yahoo.com (A. Ibrahim).

http://dx.doi.org/10.1016/j.micpro.2015.11.016
0141-9331/

2. Problem formulation

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

$$Q(x) = x^m + q_{m-1}x^{m-1} + \cdots + q_2x^2 + q_1x + 1 \tag{1}$$

where $q_i \in GF(2)$ for $0 \le i \le m$. In this paper we consider trinomial irreducible polynomials of the form

$$Q(x) = x^m + x^k + 1 \tag{2}$$

This form represents two of the five $GF(2^m)$ field polynomials recommended by the National Institute of Standards and Technology (NIST) for ECC [31]. The two values of the pair m and k are $m = 233$, $k = 73$ or $m = 409$, $k = 87$. This motivated several systolic implementations using these polynomials [6–9,28].

The two field elements A and B to be multiplied are represented by the polynomials:

$$A = \sum_{i=0}^{m-1} a_i\, \alpha^i \tag{3}$$

$$B = \sum_{j=0}^{m-1} b_j\, \alpha^j \tag{4}$$

where α is a root of $Q(x)$, a_i and $b_j \in GF(2)$ for $0 \le i, j < m$. Since α is a root of $Q(x)$, we can write $Q(\alpha)$ using (2) in the form:

$$Q(\alpha) = \alpha^m + \alpha^k + 1 = 0 \tag{5}$$

or:

$$\alpha^m = \alpha^k + 1 \tag{6}$$

After the modulo operation, the product C will be m-bits long and is given by:

$$\begin{aligned} C &= A \times B \bmod Q(\alpha) \\ &= \left[\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i\, b_j\, \alpha^{i+j} \right] \bmod Q(\alpha) \end{aligned} \tag{7}$$

$$= \sum_{k=0}^{m-1} c_k\, \alpha^k \tag{8}$$

where c_k represents bit k of the product C. It is not practical to perform the modulo operation on the polynomial in (7) whose degree is $2m-2$. Since the modulo operation is distributive, we can write (7) in the form:

$$C = \sum_{i=0}^{m-1} b_i[\, \alpha^i A \mod Q(\alpha)] \tag{9}$$

$$= \sum_{i=0}^{m-1} [C_i \mod Q(\alpha)] \tag{10}$$

We note from (9) or (10) that each partial product C_i is a polynomial of degree $m + i - 1$. More specifically, partial product C_i is given by

$$C_i = b_i \alpha^i A \mod Q(\alpha) \tag{11}$$

Each partial product C_i is a polynomial of degree $m + i - 1$ that must be reduced before the addition operation is performed, which is not too practical. Since the addition operation is associative, we iteratively perform the reduction operation on the different powers of the partial product C_i (11). We call this method Progressive Product Reduction (PPR) as will be explained below.

3. Progressive Product Reduction (PPR) technique

We convert (9) or (10) into an iteration using decreasing powers of the summation index i. Our strategy is to use Theorem 1 in the sequel to reduce the degree term C_i by one so we can add it to the adjacent lower degree term C_{i-1}. We can write Eq. (11) in the following form:

$$C = \langle C_0 + \alpha \langle C_1 + \alpha \langle C_2 + \cdots + \alpha \langle C_{m-2} + \alpha C_{m-1} \rangle \cdots \rangle \rangle \tag{12}$$

where

$$\langle C_i + \alpha C_{i+1} \rangle \equiv C_i + [\alpha C_{i+1} \mod Q(\alpha)] \tag{13}$$

The expression in (12) can be expressed iteratively as

$$\text{Ç}^m = 0 \tag{14}$$

$$\text{Ç}^i = \left\langle C_i + \alpha \text{Ç}^{i+1} \right\rangle \tag{15}$$

$$0 \le i < m$$

$$C = \text{Ç}^0 \tag{16}$$

where Ç^i is the intermediate value of the iteration at step i. Notice from (15) that we reduce a polynomial Ç^{i+1} of degree $m + i$ to another one of degree $m + i - 1$.

The theorem below shows how a polynomial of degree $i + 1 \ge m$ can be reduced to a polynomial of degree i using the irreducible polynomial $Q(\alpha)$ in Eq. (6).

Theorem 1. *Assume an m-term polynomial with degree $l + m - 1$ of a form similar to one of the partial products in (7) or (8):*

$$P(\alpha) = \alpha^l \sum_{i=0}^{m-1} q_i \alpha^i \tag{17}$$

We can reduce the order of this polynomial by one using Eq. (6).

Proof. We can write $P(\alpha)$ in (17) in the form:

$$P(\alpha) = \alpha^{l-1} \sum_{i=0}^{m-1} q_i \alpha^{i+1} \tag{18}$$

We make use of the expression for α^m in (6) in (18) to get:

$$P(\alpha) = \alpha^{l-1} \left[q_{m-1} \left(\alpha^k + 1 \right) + \sum_{i=1}^{m-1} q_{i-1} \alpha^i \right] \tag{19}$$

Thus the input polynomial $P(\alpha)$ has now been reduced by one as required.

□

From (14)–(16) and (19) we can arrive at the bit-level iterations for our PPR algorithm as:

$$c_j^m = 0 \tag{20}$$

$$c_j^i = c_{j-1}^{i+1} + b_i a_j, \quad j \ne 0, k \tag{21}$$

$$c_j^i = c_{j-1}^{i+1} + b_i a_j + c_{m-1}^{i+1}, \quad \text{for } j = 0,\ k \tag{22}$$

$$c_j = c_j^0 \tag{23}$$

In the above equations it was assumed that

$$c_j^{-1} = 0, \quad \text{for } 0 \le j < m \tag{24}$$

where the term c_j^{-1} represents the jth bit of the partial product at the start of iterations.