INTEGRATION the VLSI journal 58 (2017) 55–63

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

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

Low space complexity CRT-based bit-parallel $GF(2^n)$ polynomial basis multipliers for irreducible trinomials

Jiajun Zhang[a,*], Haining Fan[b]

[a] School of Software and TNLIST, Tsinghua University, Beijing, China
[b] Department of Computer Science and Technology and TNLIST, Tsinghua University, Beijing, China

ARTICLE INFO

Keywords:
Finite field
Multiplication
Polynomial basis
The Chinese Remainder Theorem
Irreducible polynomial

ABSTRACT

This paper presents new space complexity records for the fastest parallel $GF(2^n)$ multipliers for about 22% values of n such that a degree-n irreducible trinomial $f = u^n + u^k + 1$ exists over $GF(2)$. By selecting the largest possible value of $k \in (n/2, 2n/3]$, we further reduce the space complexities of the Chinese remainder theorem (CRT)-based hybrid polynomial basis multipliers. Our experimental results show that among the 539 values of $n \in [5, 999]$ such that f is irreducible for some $k \in [2, n-2]$, there are 317 values of n such that $k \in (n/2, 2n/3]$. For these irreducible trinomials, the space complexities of the CRT-based hybrid multipliers are reduced by 14.3% on average. As a comparison, the previous CRT-based multipliers considered the case $k \in [2, n/2]$, and the improvement rate is 8.4% on average for only 290 values of n among these 539 values of n.

1. Introduction

Low complexity multipliers are key modules of $GF(2^n)$-based cryptographic chips. Their theoretical space and time complexities, namely, the total number of 2-input AND/XOR gates (the $GF(2)$ multiplication/addition) and their corresponding gate delays (denoted by "T_A" and "T_X"), depend on various factors, for example, the method to represent field elements: polynomial, normal and dual bases; the underlying multiplication algorithms: quadratic and subquadratic algorithms etc. Pure quadratic and subquadratic multipliers are of great theoretical importance. They have the lowest time and space complexities respectively, but their disadvantages are also obvious: the largest space and time complexities respectively. On the other hand, hybrid approaches in [1–3] provide a trade-off between the time and space complexities. These multipliers first perform a few subquadratic iterations to reduce the whole space complexities, and then one quadratic step on small input operands to achieve lower time complexity. Therefore, this hybrid approach is often adopted in practical applications.

Recently, two different hybrid multipliers are presented in [4,5]. The former uses the matrix representation, and adopts the "1-quadratic-and-then-subquadratic" computational mode. This method reduces the total space complexity for current ASIC implementations. The latter uses the polynomial representation, and follows the "1-subquadratic-and-then-quadratic" computational mode first proposed in [6]. Thanks to the property of the ceiling function "$\lceil\cdot\rceil$", the time complexity of this multiplier matches the fastest bit-parallel multiplier – the pure quadratic multiplier – when $u^n + u^k + 1$ is irreducible for some $k \in [(n-1)/3, n/2]$. Its highlight is the AND and XOR space complexities: the following bounds of the fastest bit-parallel multipliers were broken for the first time:

$$\begin{cases} \#\text{AND gates:} & n^2 \\ \#\text{XOR gates:} & n^2 - 1. \end{cases}$$

In this work, we report a further reduction of the space complexities obtained in [5] for some irreducible trinomials $u^n + u^k + 1$ where $k \in (n/2, 2n/3]$, and therefore present new space complexity records for these irreducible trinomials. Before explaining our motivation, we recall the multipliers in [5] first.

Let $f(u) = u^n + u^k + 1$ $(n > 2)$ be an irreducible trinomial of degree n over $GF(2)$. All elements of the finite field $GF(2^n) := GF(2)[u]/(f(u))$ can be represented using a polynomial basis $\{x^i | 0 \le i \le n-1\}$, where x is a root of f. Given two field elements $a(x) = \sum_{i=0}^{n-1} a_i x^i$ and $b(x) = \sum_{i=0}^{n-1} b_i x^i$, where $a_i, b_i \in GF(2)$, the classical polynomial basis multiplication algorithm computes the $GF(2^n)$ product $c(x) = \sum_{i=0}^{n-1} c_i x^i$ of $a(x)$ and $b(x)$ using the following two steps. For the sake of simplicity, we omit "(x)" in polynomial "$a(x)$" and denote $a(x)$ by a.

(i) Conventional polynomial multiplication:

$$s = a \cdot b = \sum_{t=0}^{2n-2} s_t x^t, \tag{1}$$

* Corresponding author.
E-mail addresses: zjjzhaoyun@126.com (J. Zhang), fhn@tsinghua.edu.cn (H. Fan).

http://dx.doi.org/10.1016/j.vlsi.2017.02.008
Received 26 June 2016; Received in revised form 23 January 2017; Accepted 20 February 2017
Available online 21 February 2017

where

$$s_t = \sum_{\substack{i+j=t \\ 0\le i,j<n}} a_i b_j = \begin{cases} \sum_{i=0}^{t} a_i b_{t-i}, & 0 \le t \le n-1, \\ \sum_{i=t+1-n}^{n-1} a_i b_{t-i}, & n \le t \le 2n-2. \end{cases} \quad (2)$$

(ii) Reduction $\bmod f = x^n + x^k + 1$:

$$c = \sum_{i=0}^{n-1} c_i x^i = s \bmod f. \quad (3)$$

The product c obtained in the second step is the remainder of s divided by f, i.e.,

$$s = f \cdot q + c. \quad (4)$$

In order to apply the Chinese remainder theorem in the second step, multipliers in [5] adopt the following identity:

$$s = f \cdot q + c = (f+1)q + (c+q). \quad (5)$$

We note that addition is the same with subtraction in fields of characteristic 2.

This identity converts the seemingly unbreakable step (ii) "modulo the degree-n irreducible trinomial f" into the following two problems:

(A) Compute the quotient q of s divided by $f+1$;
(B) Compute the remainder $(c+q)$ of s divided by $f+1$.

Then the product c of elements a and b can be constructed by $c = q + (c+q)$.

Because the degree-n polynomial $f+1 = x^n + x^k$ is clearly reducible, the CRT can be used to divide problem (B), i.e., $a \cdot b \bmod (x^n + x^k)$, into the following two smaller subproblems [5]:

(1) $a \cdot b \bmod x^k$;
(2) $a \cdot b \bmod (x^{n-k} + 1)$.

Therefore the product c of elements a and b is now divided into three problems: (A), (B.1) and (B.2). In order to understand the advantage of this method, we make a rough estimate of the AND (or XOR) gate complexities of these three problems. We consider only the quadratic part and ignore the linear part. From the analysis in [5], these three complexities are $C_A = n^2/2$, $C_1 = k^2/2$ and $C_2 = (n-k)^2$ respectively. The summation of $C_A + C_1 + C_2$ is a function of k. Fig. 1 depicts the decreasing trend of this summation while the value of k increases. For the case $k = n/3$, this summation is n^2, which is approximately equal to the AND (or XOR) gate complexity of the fastest quadratic multiplier. In fact, multipliers in [5] are not better than the best quadratic multipliers for the case $0 < k \le n/3$. However, for the case $k = n/2$, this summation is only $7n^2/8$, which is the best case discussed in [5].

Now we explain our motivation. The CRT divides the size-n problem (B) into two smaller subproblems (B.1) and (B.2) of sizes k and $n-k$ respectively. According to the balancing principle of the algorithm design, it is better to select k, if possible, such that the difference between sizes of the two subproblems (B.1) and (B.2), i.e., $|k-(n-k)| = |2k-n|$, is as small as possible. Multipliers in [5] just follow this principle, and select the largest possible k in the range of $[1, n/2]$ as the best candidate.

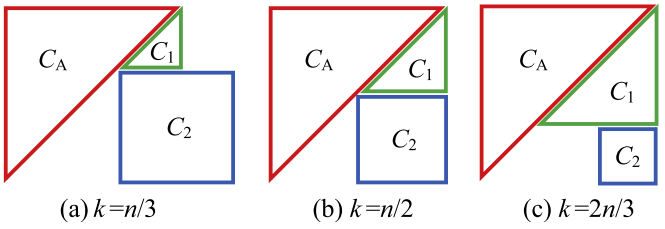

Fig. 1. Approximate AND (or XOR) gate Complexity.

In fact, this choice of the value of k, i.e., the largest possible k in the range of $[1, n/2]$, is also the best candidate in some other quadratic multipliers. For example, the shifted polynomial basis multipliers in [7] and the polynomial basis Montgomery multipliers in [8].

As for the CRT-based multiplier, a more careful observation shows that the function

$$C_A + C_1 + C_2 = n^2/2 + k^2/2 + (n-k)^2, \quad (6)$$

namely, the approximate AND (or XOR) gate complexity, reaches its minimal value when $k = 2n/3$, i.e., Fig. 1 (c). Therefore, we consider the case $n/2 < k \le 2n/3$ in this work. Our experimental results show that among the 539 values of n such that $4 < n < 1000$ and $u^n + u^k + 1$ is irreducible over $GF(2)$ for some $k > 1$, the proposed multipliers beat those in [5] for 117 values of n: they have the same time complexity, but the space complexities are reduced.

2. Two types of the CRT-based multipliers

The two types of multipliers presented in this work, i.e., the Type-A and Type-B multipliers, follow the same design approach as those in [5]: compute each coefficient c_i using a binary XOR tree. The difference of these two types of multipliers is that they adopt different computational procedures to organize the reusable terms, and therefore, provide a space-time trade-off. Type-A multipliers achieve the minimal number of the XOR gates, but their time complexities are not optimal for some irreducible trinomials. Type-B multipliers overcome this disadvantage at the cost of some more XOR gates, which are saved in the reusable terms of the Type-A multipliers. They try to construct a binary XOR tree with the smallest height. For example, $u^{68} + u^{33} + 1$ is irreducible on $GF(2)$ [5]. The coefficients c_0 and c_{35} share a common subexpression of 33 product terms. While the Type-A multiplier computes this common subexpression using a single binary XOR subtree of height 6, the Type-B multiplier splits this 33-term common subexpression into two parts: a binary XOR subtree of height 5 which computes the summation of the first 32 product terms and a binary XOR tree of height 0 which consists of the only 33rd product term. The subtree of height 5 is then shared in the coefficients c_0 and c_{35}, and the 33rd product term is combined with other terms in c_0 and c_{35} at the cost of 2 extra XOR gates. However, the total XOR gate delays of this method is less than that of the Type-A multiplier by $1T_X$. For a detailed description of these two types of multipliers, please refer to [5]. The following Fig. 2 illustrates the architecture of these multipliers.

Because the expression of each coefficient c_i is a bit complex, we derive the explicit formulae of $c=ab$ for the case $n/2 < k \le 2n/3$ in Appendix. To understand the key idea of the proposed multipliers easily, the reader may jump to the example given in Section 4 first. In the following, we analyze the complexities of the Type-A and Type-B multipliers.

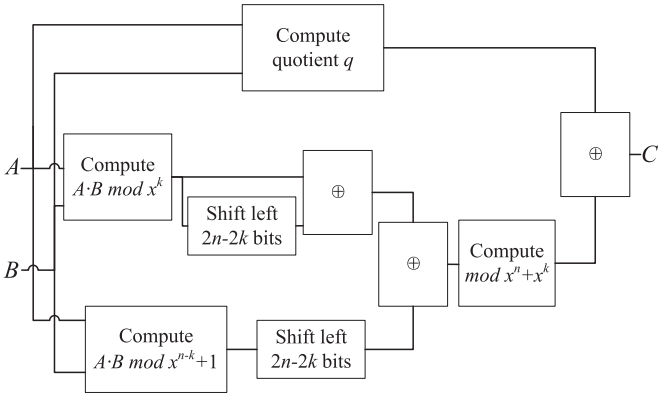

Fig. 2. The architecture of the multiplier.