Low Space Complexity CRT-based Bit-Parallel $GF(2^n)$ Polynomial Basis Multipliers for Irreducible Trinomials

Jiajun Zhang and Haining Fan

Abstract

By selecting the largest possible value of $k \in (n/2, 2n/3]$, we further reduce the AND and XOR gate complexities of the CRT-based hybrid parallel $GF(2^n)$ polynomial basis multipliers for the irreducible trinomial $f = u^n + u^k + 1$ over $GF(2)$: they are always less than those of the current fastest parallel multipliers – quadratic multipliers, i.e., n^2 AND gates and $n^2 - 1$ XOR gates. 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 AND and XOR gate complexities of the CRT-based hybrid multipliers are reduced by 15.3% on average. Especially, for the 124 values of such n, the two kinds of multipliers have the same time complexity, but the space complexities are reduced by 15.5% on average. As a comparison, the previous CRT-based multipliers consider the case $k \in [2, n/2]$, and the improvement rate is only 8.4% on average.

I. 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

Jiajun Zhang and Haining Fan are with the School of Software and TNLIST, Tsinghua University, Beijing, China.
E-mail: zjjzhaoyun@126.com and fhn@tsinghua.edu.cn

also obvious: the largest space and time complexities respectively. On the other hand, hybrid approaches in [1], [2] and [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] and [5] (or its preprint in [6]). 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 [7]. Thanks to the property of the ceiling function "$\lceil\cdot\rceil$", the time complexity of this multiplier matches the fastest bit-parallel multiplier – the 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 $k \in (n/2, 2n/3]$. Before explaining our motivation, we recall the multiplier 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 $\{x^i | 0 \leq i \leq n-1\}$, 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:

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

where

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

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

$$c = \sum\nolimits_{i=0}^{n-1} c_i x^i = s \bmod f. \tag{2}$$

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

$$s = f \cdot q + c.$$

In order to apply the Chinese Remainder Theorem (CRT) in the second step, multipliers in [5] adopt the following identity:

$$s = f \cdot q + c = (f + 1)q + (c + q). \tag{3}$$

We note that addition and subtraction are the same 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]:

(B.1). $a \cdot b \bmod x^k$;

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

Thus 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 $C_{\text{A}} = n^2/2$, $C_1 = k^2/2$ and $C_2 = (n - k)^2$ respectively. Fig 1 (a) and Fig 1 (b) illustrate them for the two special cases $k = n/3$ and $k = n/2$ respectively. For the case $k = n/3$, the summation of $C_{\text{A}} + C_1 + C_2$ 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 $0 < k \leq n/3$. 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

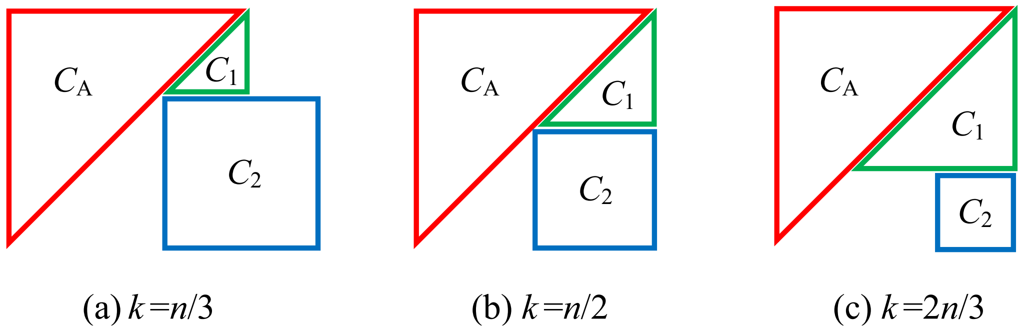

Fig. 1. Approximate AND (or XOR) gate Complexity

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.

In fact, the time complexities of other quadratic multipliers are often functions of k too. For a given value of n, it is, therefore, better to select an optimal value of k if it is possible. The above 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 quadratic multipliers. For example, the XOR gate delays of the shifted polynomial basis multiplier in [8] and the polynomial basis Montgomery multiplier in [9] are as follows:

$$\begin{cases} \lceil \log_2(n+k) \rceil, & 2k > n, \\ \lceil \log_2(2n-k-1) \rceil, & 2k < n. \end{cases}$$

Because a polynomial is irreducible iff its reciprocal polynomial is irreducible, the optimal value of k is just the one that $|k-n/2| = |2k-n|/2$ is minimal for these two multipliers.

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

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

namely, the approximate AND (or XOR) gate complexity, reaches its minimal value when $k = 2n/3$, i.e., Fig 1 (c), not $k = n/2$. Therefore, we consider multipliers based on irreducible trinomial $f = u^n + u^k + 1$ for the case $n/2 < k \leq 2n/3$ 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 122 values of n: they have the same time complexity, but the space complexities are reduced by 8.4% on average.

In the following, we first derive explicit formulae of $c = ab$ for the case $n/2 < k \leq 2n/3$ and then compare their complexities.

II. The Expression of $c = ab \in GF(2^n)$ for $n/2 < k < 2n/3$

In this section, we consider the case $n/2 < k < 2n/3$. The special case $k = 2n/3$ will be discussed later because its expression is different slightly. We first derive the expressions of the quotient and the remainder of $s = a \cdot b$ divided by $(f+1) = x^n + x^k$, and then present the expression of $c = ab$ in $GF(2^n)$.

A. *Compute the quotient q of $s = a \cdot b$ divided by $(f+1) = x^n + x^k$*

Since $degree(q) \leq n-2$ in (3), we can define q as $q := \sum_{i=0}^{n-2} q_i x^i$. Replacing f in (3) by $f = x^n + x^k + 1$, we have

$$s = (f+1)q + (c+q) = q \cdot x^n + q \cdot x^k + (c+q).$$

It is clear that

$$q_i = s_{i+n}, \text{ for } \; k-1 \leq i \leq n-2. \tag{4}$$

Moreover, we have $s_{i+n} = q_i + q_{i+n-k}$ for $2k-1-n \leq i \leq k-2$. The terms q_{i+n-k} for $2k-1-n \leq i \leq k-2$ are the same as the terms q_j for $k-1 \leq j \leq n-2$, which also appear in (4). Thus we have

$$q_i = s_{i+n} + s_{i+2n-k}, \quad \text{where } \; 2k-1-n \leq i \leq k-2. \tag{5}$$

Moreover, we have $s_{i+n} = q_i + q_{i+n-k}$ for $0 \leq i \leq 2k-2-n$. The terms q_{i+n-k} for $0 \leq i \leq 2k-2-n$ are the same as the terms q_j for $n-k \leq j \leq k-2$, which appear in (5). Thus we have

$$q_i = s_{i+n} + s_{i+2n-k} + s_{i+3n-2k}, \quad \text{where } \; 0 \leq i \leq 2k-2-n. \tag{6}$$

Using (4) , (5) and (6), we get the following expression of q:

$$\begin{aligned} q &= \sum_{i=0}^{n-2} q_i x^i \\ &= \sum_{i=0}^{2k-2-n} (s_{i+n} + s_{i+2n-k} + s_{i+3n-2k}) x^i + \sum_{i=2k-1-n}^{k-2} (s_{i+n} + s_{i+2n-k}) x^i + \sum_{i=k-1}^{n-2} s_{i+n} x^i \\ &= \sum_{i=0}^{n-2} s_{i+n} x^i + \sum_{i=0}^{k-2} s_{i+2n-k} x^i + \sum_{i=0}^{2k-2-n} s_{i+3n-2k} x^i. \end{aligned} \tag{7}$$

B. Compute the remainder $(c+q) = \langle a \cdot b \rangle_{f+1}$

Because $k < 2n/3$, i.e., $3k < 2n$, we have the Bezout identity

$$x^k \cdot x^{2n-3k} + (x^{n-k}+1) \cdot (x^{n-k}+1) = 1.$$

So we know that x^{2n-3k} is the multiplicative inverse of $x^k \bmod (x^{n-k}+1)$ and $(x^{n-k}+1)$ is the multiplicative inverse of $(x^{n-k}+1) \bmod x^k$. Therefore, the remainder $(c+q) = \langle a \cdot b \rangle_{x^n+x^k}$ can be computed using the CRT as follows:

$$\langle a \cdot b \rangle_{x^n+x^k} = \left\langle \langle a \cdot b \rangle_{x^k} \cdot (x^{n-k}+1) \cdot (x^{n-k}+1) + \langle a \cdot b \rangle_{x^{n-k}+1} \cdot x^k \cdot x^{2n-3k} \right\rangle_{x^n+x^k}. \quad (8)$$

Thus, we need to compute the terms $\langle a \cdot b \rangle_{x^k}$ and $\langle a \cdot b \rangle_{x^{n-k}+1}$ in (8) first. The expression of $\langle a \cdot b \rangle_{x^k}$ can be derived from the expression of $a \cdot b$ given in (1). It is clear that

$$\langle a \cdot b \rangle_{x^k} = \sum_{i=0}^{k-1} s_i x^i. \quad (9)$$

In order to compute the term $\langle a \cdot b \rangle_{x^{n-k}+1} = \left\langle \langle a \rangle_{x^{n-k}+1} \cdot \langle b \rangle_{x^{n-k}+1} \right\rangle_{x^{n-k}+1}$ in (8), we first compute $\langle a \rangle_{x^{n-k}+1}$. Since it is the input of the operation $\left\langle \langle a \rangle_{x^{n-k}+1} \cdot \langle b \rangle_{x^{n-k}+1} \right\rangle_{x^{n-k}+1}$, we define it as $\sum_{i=0}^{n-k-1} g_i x^i$. Because $n/2 < k < 2n/3$, i.e., $2k-n-1 < n-k-1$, we have

$$\begin{aligned}
\sum_{i=0}^{n-k-1} g_i x^i &:= \langle a \rangle_{x^{n-k}+1} \\
&= \left\langle \sum_{i=0}^{n-k-1} a_i x^i + \sum_{i=n-k}^{n-1} a_i x^i \right\rangle_{x^{n-k}+1} \\
&= \sum_{i=0}^{n-k-1} a_i x^i + \left\langle \sum_{j=0}^{k-1} a_{j+n-k} x^{j+n-k} \right\rangle_{x^{n-k}+1} \\
&= \sum_{i=0}^{n-k-1} a_i x^i + \left\langle \sum_{i=0}^{k-1} a_{i+n-k} x^{i} \right\rangle_{x^{n-k}+1} \\
&= \sum_{i=0}^{n-k-1} a_i x^i + \left\langle \sum_{i=0}^{n-k-1} a_{i+n-k} x^{i} + \sum_{i=n-k}^{k-1} a_{i+n-k} x^{i} \right\rangle_{x^{n-k}+1} \\
&= \sum_{i=0}^{n-k-1} a_i x^i + \sum_{i=0}^{n-k-1} a_{i+n-k} x^{i} + \left\langle \sum_{j=0}^{2k-n-1} a_{j+2n-2k} x^{j+n-k} \right\rangle_{x^{n-k}+1} \\
&= \sum_{i=0}^{n-k-1} a_i x^i + \sum_{i=0}^{n-k-1} a_{i+n-k} x^{i} + \sum_{i=0}^{2k-n-1} a_{i+2n-2k} x^{i} \\
&= \sum_{i=0}^{2k-n-1} (a_i + a_{i+n-k} + a_{i+2n-2k}) x^i + \sum_{i=2k-n}^{n-k-1} (a_i + a_{i+n-k}) x^i .
\end{aligned} \tag{10}$$

Similarly, we can obtain the expression of $\sum_{i=0}^{n-k-1} h_i x^i := \langle b \rangle_{x^{n-k}+1}$.

Now the term $\langle a \cdot b \rangle_{x^{n-k}+1}$ in (8) can be calculated using the schoolbook polynomial multiplication algorithm in (1).

$$\begin{aligned}
\langle a \cdot b \rangle_{x^{n-k}+1} = \left\langle \langle a \rangle_{x^{n-k}+1} \cdot \langle b \rangle_{x^{n-k}+1} \right\rangle_{x^{n-k}+1} = \left\langle \left(\sum_{i=0}^{n-k-1} g_i x^i \right) \left(\sum_{i=0}^{n-k-1} h_i x^i \right) \right\rangle_{x^{n-k}+1} \\
= \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j} \right) x^{n-k-1} + \sum_{i=0}^{n-k-2} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j} \right) x^i .
\end{aligned} \tag{11}$$

Based on the above equations (8), (9) and (11), we can obtain the following expression of the remainder $(c+q) = \langle a \cdot b \rangle_{f+1}$.

$$
\begin{aligned}
& (c+q) = \langle a \cdot b \rangle_{x^n+x^k} \\
= \; & \left\langle \langle a \cdot b \rangle_{x^k} \cdot (x^{n-k}+1) \cdot (x^{n-k}+1) + \langle a \cdot b \rangle_{x^{n-k}+1} \cdot x^k \cdot x^{2n-3k} \right\rangle_{x^n+x^k} \\
= \; & \left\langle \left(\sum_{i=0}^{k-1} s_i x^i \right) \cdot (x^{2n-2k}+1) + \right. \\
& \left. \left[\left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j} \right) x^{n-k-1} + \sum_{i=0}^{n-k-2} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j} \right) x^i \right] x^{2n-2k} \right\rangle_{x^n+x^k} \\
= \; & \sum_{i=0}^{k-1} s_i x^i + \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j} \right) x^{2n-2k-1} \\
& + \left\langle \sum_{i=0}^{k-1} s_i x^{i+2n-2k} + \sum_{i=0}^{n-k-2} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j} \right) x^{i+2n-2k} \right\rangle_{x^n+x^k} \\
= \; & \sum_{i=0}^{k-1} s_i x^i + \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j} \right) x^{2n-2k-1} + \left\langle \sum_{i=0}^{2k-n-1} s_i x^{i+2n-2k} + \sum_{i=2k-n}^{k-1} s_i x^{i+2n-2k} \right\rangle_{x^n+x^k} + \\
& \left\langle \sum_{i=0}^{2k-n-1} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j} \right) x^{i+2n-2k} + \right. \\
& \left. \sum_{i=2k-n}^{n-k-2} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j} \right) x^{i+2n-2k} \right\rangle_{x^n+x^k}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{i=0}^{k-1} s_i x^i + \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j}\right) x^{2n-2k-1} + \\
&\quad \sum_{i=0}^{2k-n-1} s_i x^{i+2n-2k} + \sum_{i=0}^{2k-n-1} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j}\right) x^{i+2n-2k} \\
&\quad + \left\langle \sum_{i=2k-n}^{k-1} s_i x^{i+2n-2k} \right\rangle_{x^n+x^k} + \left\langle \sum_{i=2k-n}^{n-k-2} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j}\right) x^{i+2n-2k} \right\rangle_{x^n+x^k} \\
&= \sum_{i=0}^{k-1} s_i x^i + \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j}\right) x^{2n-2k-1} + \\
&\quad \sum_{i=2n-2k}^{n-1} s_{i-2n+2k} x^i + \sum_{i=2n-2k}^{n-1} \left(\sum_{j=0}^{i-2n+2k} g_j h_{i-2n+2k-j} + \sum_{j=i-2n+2k+1}^{n-k-1} g_j h_{i-n+k-j}\right) x^i \\
&\quad + \sum_{i=2k-n}^{k-1} s_i x^{i+n-k} + \sum_{i=2k-n}^{n-k-2} \left(\sum_{j=0}^{i} g_j h_{i-j} + \sum_{j=i+1}^{n-k-1} g_j h_{i+n-k-j}\right) x^{i+(n-k)} \\
&= \sum_{i=0}^{k-1} s_i x^i + \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j}\right) x^{2n-2k-1} + \\
&\quad \sum_{i=2n-2k}^{n-1} s_{i-2n+2k} x^i + \sum_{i=2n-2k}^{n-1} \left(\sum_{j=0}^{i-2n+2k} g_j h_{i-2n+2k-j} + \sum_{j=i-2n+2k+1}^{n-k-1} g_j h_{i-n+k-j}\right) x^i + \\
&\quad \sum_{i=k}^{n-1} s_{i-(n-k)} x^i + \sum_{i=k}^{2n-2k-2} \left(\sum_{j=0}^{i-(n-k)} g_j h_{i-(n-k)-j} + \sum_{j=i-(n-k)+1}^{n-k-1} g_j h_{i-j}\right) x^i \\
&= \sum_{i=0}^{k-1} s_i x^i + \sum_{i=k}^{2n-2k-2} \left(\sum_{j=0}^{i-(n-k)} g_j h_{i-(n-k)-j} + \sum_{j=i-(n-k)+1}^{n-k-1} g_j h_{i-j} + s_{i-(n-k)}\right) x^i + \\
&\quad \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j} + s_{n-k-1}\right) x^{2n-2k-1} + \\
&\quad \sum_{i=2n-2k}^{n-1} \left(s_{i-2n+2k} + s_{i-(n-k)} + \sum_{j=0}^{i-2n+2k} g_j h_{i-2n+2k-j} + \sum_{j=i-2n+2k+1}^{n-k-1} g_j h_{i-n+k-j}\right) x^i. \qquad (12)
\end{aligned}
$$

C. The expression of $c = ab$ *in* $GF(2^n)$

Based on the expressions of q, i.e., (7), and $c + q$, i.e., (12), we can obtain the expression of $c = q + (c + q)$ in $GF(2^n)$.

$$\begin{aligned}
c \;=\; & \sum_{i=0}^{2k-2-n}(s_{i+n}+s_{i+2n-k}+s_{i+3n-2k})x^i+\sum_{i=2k-1-n}^{k-2}(s_{i+n}+s_{i+2n-k})x^i+\\
& \sum_{i=k-1}^{n-2}s_{i+n}x^i+\sum_{i=0}^{k-1}s_ix^i+\\
& \sum_{i=k}^{2n-2k-2}\left(\sum_{j=0}^{i-(n-k)}g_jh_{i-(n-k)-j}+\sum_{j=i-(n-k)+1}^{n-k-1}g_jh_{i-j}+s_{i-(n-k)}\right)x^i+\\
& \left(\sum_{j=0}^{n-k-1}g_jh_{n-k-1-j}+s_{n-k-1}\right)x^{2n-2k-1}+\\
& \sum_{i=2n-2k}^{n-1}\left(s_{i-2n+2k}+s_{i-(n-k)}+\sum_{j=0}^{i-2n+2k}g_jh_{i-2n+2k-j}+\sum_{j=i-2n+2k+1}^{n-k-1}g_jh_{i-n+k-j}\right)x^i\\
=\; & \sum_{i=0}^{2k-2-n}\{((s_i))+s_{i+n}+(s_{i+2n-k}]+[s_{i+3n-2k})\}x^i+\\
& \{[[s_{2k-1-n}]]+[[[s_{n+k-1}]]]+s_{2k-1}\}x^{2k-1-n}+\\
& \sum_{i=2k-n}^{k-2}\{[s_i+[s_{i+2n-k})]+(s_{i+n}]\}x^i+\{[[[s_{n+k-1}]]]+(s_{k-1})\}x^{k-1}\\
& +\sum_{i=k}^{2n-2k-2}\left\{\sum_{j=0}^{i-(n-k)}g_jh_{i-(n-k)-j}+\sum_{j=i-(n-k)+1}^{n-k-1}g_jh_{i-j}+[s_{i-(n-k)}+s_{i+n}]\right\}x^i\\
& +\left\{\sum_{j=0}^{n-k-1}g_jh_{n-k-1-j}+[s_{n-k-1}+s_{3n-2k-1}]\right\}x^{2n-2k-1}\\
& +\sum_{i=2n-2k}^{n-2}\left\{[s_{i+n}+s_{i-(n-k)}]+((s_{i-2n+2k}))+\sum_{j=0}^{i-2n+2k}g_jh_{i-2n+2k-j}+\sum_{j=i-2n+2k+1}^{n-k-1}g_jh_{i-n+k-j}\right\}x^i\\
& +\left\{[[s_{2k-n-1}]]+(s_{k-1})+\sum_{j=0}^{2k-n-1}g_jh_{2k-n-1-j}+\sum_{j=2k-n}^{n-k-1}g_jh_{k-1-j}\right\}x^{n-1}. \qquad (13)
\end{aligned}$$

Here, we use different combinations of "(", ")", "[" and "]" to represent the terms that can be reused.

III. Two types of the CRT-based multipliers

In this section, we analyze the complexities of the Type-A and Type-B multipliers presented in [5]. 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. For a detailed description of these two types of multipliers, please refer to [5].

A. *Complexities of Type-A multipliers*

We need to determine the complexities of all coefficients c_i's ($0 \le i \le n-1$) in equation (13). This equation includes only terms s_i, h_i and g_i.

The expressions of s_i ($0 \le i \le 2n-2$) are given in (1). For $0 \le i \le n-1$, the term $s_i = \sum_{j=0}^{i} a_j b_{i-j}$ is the summation of $i+1$ product terms $a_j b_{i-j}$. For $n \le i \le 2n-2$, the term $s_i = \sum_{j=i+1-n}^{n-1} a_j b_{i-j}$ is the summation of $2n-1-i$ product terms $a_j b_{i-j}$.

The expressions of g_i ($0 \le i \le n-k-1$) are given in (10). The expression of h_i is similar to that of g_i and they have the same complexity. Clearly, the complexities to compute all g_i and h_i for $0 \le i \le n-k-1$ are $2((2k-n)2+(2n-3k)) = 2k$ XOR gates and 2 T_X gate delays due to the parallelism.

TABLE I

THE NUMBER OF THE PRODUCT TERMS IN THE COEFFICIENT c_i OF x^i.

x^i	The number of the product terms
$0 \le i \le 2k-2-n$	$(\underline{i+1}) + (2n-1-(n+i)) + \underline{(2n-1-(i+2n-k))} + \overline{(2n-1-(i+3n-2k))} = 3k-2-2i$
$i = 2k-1-n$	$(\underline{2k-1-n+1}) + \underline{(2n-1-(n+k-1))} + (2n-1-(n+2k-1-n)) = 2n-k$
$2k-n \le i \le n-k-1$	$[\underline{(i+1)+(2n-1-(i+2n-k))}] + \underline{(2n-1-(n+i))} = n+k-1-i$
$n-k \le i \le k-2$	$[\underline{(i+1)+\overline{(2n-1-(i+2n-k))}}] + \underline{(2n-1-(n+i))} = n+k-1-i$
$i = k-1$	$\underline{k} + \underline{(2n-1-(n+k-1))} = n$
$k \le i \le 2n-2k-2$	$[\underline{(2n-1-(i+n))+(i-n+k+1))}] + (n-k) = n$
$i = 2n-2k-1$	$[\underline{(n-k+1)+(2n-1-(3n-2k-1))}] + (n-k) = n$
$2n-2k \le i \le n-2$	$(n-k) + [\underline{(2n-1-(n+i))+(i-n+k+1)}] + (\underline{i-2n+2k+1}) = 2k+1-n+i$
$i = n-1$	$(n-k) + \underline{k} + (\underline{2k-1-n+1}) = 2k$

For the simplicity of description, we also call $g_i h_j$ in (13) a product term. Therefore, the number of the AND gates used in the proposed multipliers is equal to the of the product terms excluding the reusable terms. Table I lists the number of the product terms in each coefficient of x^i. The numbers of the reusable product terms are overlined and underlined in the table.

Thus, the number of the AND product terms that can be saved is:

$$\begin{aligned}\phi_{\text{AND}} &= (n-k)k + k + \sum_{i=0}^{2k-2-n}(i+1) + \\ &\quad \sum_{i=0}^{2k-2-n}(2k-n-1-i) + \sum_{i=0}^{2k-2-n}(k-1-i) \\ &= 3k^2 - 2nk - k + n/2 + n^2/2.\end{aligned}$$

Therefore, the total number of the AND gates used in the Type-A multiplier is

$$\begin{aligned}\Delta &= (2n-3k+1)n + \sum_{i=0}^{2k-2-n}(3k-2-2i) + \\ &\quad \sum_{i=2k-1-n}^{k-2}(n+k-1-i) + \sum_{i=2n-2k}^{n-2}(2k+1+i-n) + 2k \\ &\quad -3k^2 - 2nk - k + n/2 + n^2/2 \\ &= (3n^2 + 3k^2 - 4kn - n + k)/2 \\ &= n^2 + \frac{(n-k)(n-1-3k)}{2}, \end{aligned} \tag{14}$$

which coincides with the value of δ obtained in [5].

The number of the XOR gates that can be saved in the Type-A multiplier is:

$$\begin{aligned}\phi_{\text{XOR\_A}} &= (n-k-1)(k-1) + (k-1) + (k-2) + \sum_{i=0}^{2k-2-n}(i) + \\ &\quad \sum_{i=0}^{2k-2-n}(2k-n-2-i) + \sum_{i=0}^{2k-2-n}(k-2-i) \\ &= 3k^2 - 2nk - 6k + 5n/2 + n^2/2 + 1.\end{aligned}$$

Therefore, the total number of the XOR gates is the summation of $2k$ (computing g_i and h_i) and the number of "+" in equation (13) excluding the reusable terms:

$$2k + [(\Delta + \phi_{\text{AND}}) - n] - \phi_{\text{XOR\_A}} = \Delta + 7k - 3n - 1.$$

We now derive the time complexity of the type-A multiplier using Table I, which is the maximum AND and XOR gate delays of the coefficient c_i for $0 \le i \le n-1$.

For $0 \le i \le 2k-n-2$, the maximum gate delay is from the coefficient

$$c_0 = s_0 + s_n + s_{2n-k} + s_{3n-2k},$$

which includes $1 + (n-1) + [\underline{k-1}] + [\underline{2k-1-n}] = 3k-2$ product terms $a_i b_j$. Because the terms s_{2n-k} and s_{3n-2k}, which are the summations of $[\underline{k-1}]$ and $[\underline{2k-1-n}]$ product terms $a_i b_j$ respectively, will be reused, and $k-1$ is lager than $2k-1-n$, we compute s_{2n-k} first. In order to obtain the explicit gate delay formula of c_0, we assume that $2^{v-1} < k-1 \leq 2^v$ for some positive integer v. These $k-1$ product terms can be XORed using a single binary XOR subtree of height v. Therefore, the total gate delay to compute the subtree of s_{2n-k} is $1T_A + vT_X$. During the period to compute s_{2n-k}, the $[\underline{2k-1-n}]$ product terms in s_{3n-2k} can be XORed simultaneously.

The left $(n-1)+1+1 = n+1$ product terms $a_i b_j$ can then be processed as follows. Let $n+1 = y \cdot 2^v + z$ by the division algorithm, where $0 \leq z < 2^v$. We split these $(n+1)$ product terms into $\lceil \frac{n+1}{2^v} \rceil$ groups, where the last group may have z product terms if $z > 0$ and the other groups have 2^v product terms each. During the period to compute s_{2n-k}, i.e., $1T_A + vT_X$, the product terms in these $\lceil \frac{n+1}{2^v} \rceil$ groups can be XORed in parallel using $\lceil \frac{n+1}{2^v} \rceil$ binary XOR subtrees of the same height v. Finally, these $\lceil \frac{n+1}{2^v} \rceil$ summations and the result of s_{2n-k} are XORed using a binary XOR tree at the cost of $\lceil \log_2(\lceil \frac{n+1}{2^v} \rceil + 1) \rceil$ XOR gate delays. Therefore, the total XOR gate delay to compute the coefficient c_0 is

$$v + \left\lceil \log_2\left(\left\lceil \frac{n+1}{2^v} \right\rceil + 1 \right) \right\rceil = \lceil \log_2(n+1+2^v) \rceil \,, \tag{15}$$

where $2^{v-1} < k-1 \leq 2^v$, by the following lemma 1 [5].

Lemma 1: Let v and i be positive integers. If $i \geq 2^v$ then $v + \lceil \log_2 \lceil \frac{i}{2^v} \rceil \rceil = \lceil \log_2 i \rceil$.

For $i = 2k-n-1$, the coefficient

$$c_{2k-n-1} = s_{2k-n-1} + s_{n+k-1} + s_{2k-1}$$

includes $[\underline{2k-n}] + [\underline{n-k}] + [2n-2k] = 2n-k$ product terms $a_i b_j$. Because the terms s_{2k-n-1} and s_{n+k-1}, which are the summation of $[\underline{2k-n}]$ and $[\underline{n-k}]$ product terms $a_i b_j$ respectively, will be reused, and $n-k$ is lager than $2k-n$, we compute the s_{n+k-1} first. We assume that $2^{v-1} < n-k \leq 2^v$ for some positive integer v. So these $n-k$ product terms can be XORed using a single binary XOR subtree of height v. Therefore, the total gate delay to compute the subtree of s_{n+k-1} is $1T_A + vT_X$. Besides, during the period to compute s_{n+k-1}, the $[\underline{2k-n}]$ product terms in s_{2k-n-1} can be XORed simultaneously. The left $2n-2k+1$ product terms $a_i b_j$ can then be processed as follows. Let $2n-2k+1 = y \cdot 2^v + z$ by the division algorithm, where $0 \leq z < 2^v$. We split these $2n-2k+1$ product terms into $\lceil \frac{2n-2k+1}{2^v} \rceil$ groups, where the last group may have z product terms if $z > 0$ and the other groups have 2^v product terms each. During the period to compute s_{n+k-1}, i.e., $1T_A + vT_X$, the product terms in these $\lceil \frac{2n-2k+1}{2^v} \rceil$ groups can be XORed in parallel using $\lceil \frac{2n-2k+1}{2^v} \rceil$ binary XOR subtrees of the same height v. Finally, these $\lceil \frac{2n-2k+1}{2^v} \rceil$ summations and the result of s_{n+k-1} are XORed using a binary XOR tree at the cost of $\lceil \log_2(\lceil \frac{2n-2k+1}{2^v} \rceil + 1) \rceil$ XOR gate delays. Therefore, the total XOR gate delay to compute the coefficient c_{2k-n-1} is

$$v + \left\lceil \log_2\left(\left\lceil \frac{2n-2k+1}{2^v} \right\rceil + 1\right) \right\rceil = \lceil \log_2(2n - 2k + 1 + 2^v) \rceil, \tag{16}$$

where $2^{v-1} < n - k \leq 2^v$, by lemma 1.

For $2k - n \leq i \leq k - 2$, the maximum gate delay is from the coefficient

$$c_{2k-n} = [s_{2k-n} + s_{n+k}] + s_{2k}$$

which includes $\underline{[(2k - n + 1) + (n - k - 1)]} + [2n - 2k - 1] = [\underline{k}] + [2n - 2k - 1] = 2n - k - 1$ product terms $a_i b_j$. Because the terms $[s_{2k-n} + s_{n+k}]$ and s_{2k}, which are the summation of $[\underline{k}]$ and $2n - 2k - 1$ product terms $a_i b_j$ respectively, will be reused, and $2n - 2k - 1$ is larger than $[\underline{k}]$, we compute the terms of s_{2k} first. We assume that $2^{v-1} < 2n - 2k - 1 \leq 2^v$ for some positive integer v. So these $2n-2k-1$ product terms can be XORed using a single binary XOR subtree of height v. Therefore, the total gate delay to compute the subtree of s_{2k} is $1T_A + vT_X$. Besides, during the period to compute s_{2k}, the $[\underline{k}]$ product terms in $[s_{2k-n}+s_{n+k}]$ can be XORed simultaneously. Therefore, the total XOR gate delay to compute the coefficient c_{2k-n} is

$$v + 1 = \lceil \log_2(2^{v+1}) \rceil \leq \lceil \log_2(4n - 4k - 2 + 2^v) \rceil \tag{17}$$

by lemma 1 and $2^{v-1} < 2n - 2k - 1 \leq 2^v$.

For c_{k-1}, the coefficient

$$c_{k-1} = s_{n+k-1} + s_{k-1}$$

which includes $[\underline{n-k}] + [\underline{k}] = n$ product terms $a_i b_j$. We can get the delay of c_{k-1} using the same way we compute the c_i of $2k - n \leq i \leq k - 2$. Therefore, the total XOR gate delay to compute the coefficient c_{k-1} is

$$v + 1 = \lceil \log_2(2^{v+1}) \rceil \leq \lceil \log_2(2k + 2^v) \rceil \tag{18}$$

by lemma 1 and $2^{v-1} < k \leq 2^v$.

For $k \le i \le 2n-2k-2$, the coefficient is

$$c_i = \sum_{j=0}^{i-(n-k)} g_j h_{i-(n-k)-j} + \sum_{j=i-(n-k)+1}^{n-k-1} g_j h_{i-j} + [s_{i-(n-k)} + s_{i+n}].$$

Each coefficient c_i includes a reusable term $[s_{i-(n-k)} + s_{i+n}]$ and $n-k$ product terms $g_i h_j$. The reusable term, which is also part of the terms in c_i for $2k-n \le i \le k-2$, includes $(k-n+1+i)+(n-1-i)=k$ product terms $a_i b_j$. Therefore, these coefficients c_i's have the same distribution pattern of the product terms $a_i b_j$ and $g_i h_j$, and their gate delays are equal. In order to compute c_i, the terms g_i and h_i should be generated first according to equation (10), and then they are ANDed in the schoolbook multiplication (11). The product terms $g_i h_j$ are just the intermediate result of this multiplication operation, and they can be generated at the cost of $(1T_A + 2T_X)$ gate delays. Then these coefficients c_i's can be computed similar to the way we computed c_0. The only difference is that all product terms in c_0 are of the form $a_i b_j$, but the terms g_i and h_i here should be generated first using $2T_X$ delay. Therefore, the size of the groups should be halved twice, and the total XOR gate delay to compute these coefficients is

$$v + \left\lceil \log_2\left(\left\lceil \frac{n-k}{2^{v-2}} \right\rceil + 1\right) \right\rceil = \lceil \log_2(4n-4k+2^v) \rceil, \tag{19}$$

where $2^{v-1} < k \le 2^v$.

The gate delay of $c_{2n-2k-1}$ is the same as that of the above case $k \le i \le 2n-2k-2$.

We now count the gate delay of the coefficient c_i for $2n-2k \le i \le n-2$. The maximum gate delay is from the coefficient

$$c_{n-2} = [s_{2n-2} + s_{k-2}] + [s_{2k-n-2}] + \sum_{j=0}^{2k-n-2} g_j h_{2k-n-2-j} + \sum_{j=2k-n-1}^{n-k-1} g_j h_{n-2-j}$$

which includes a reusable term $[s_{2n-2} + s_{k-2}]$, a reusable term $[s_{2k-n-2}]$ and $n-k$ product terms $g_i h_j$. In order to compute c_i, the terms g_i and h_i should be generated first at the cost of $(1T_A + 2T_X)$ gate delays according to equation(10). Because the terms $[s_{2n-2} + s_{k-2}]$ and $[s_{2k-n-2}]$, which are the summation of $[\underline{k}]$ and $[\underline{2k-n-1}]$ product terms $a_i b_j$ respectively, will be reused, and $[\underline{k}]$ is larger than $[\underline{2k-n-1}]$, we compute the terms $[s_{2n-2} + s_{k-2}]$ first. Then coefficient c_{n-2} can be computed similar to the way we computed c_i for $k \le i \le 2n-2k-2$. The total XOR gate delay to compute c_{n-2} is

$$v + \left\lceil \log_2\left(\left\lceil \frac{n-k+1}{2^{v-2}} \right\rceil + 1\right) \right\rceil = \lceil \log_2(4n-4k+4+2^v) \rceil, \tag{20}$$

where $2^{v-1} < k \le 2^v$.

The gate delay of the coefficient c_{n-1} is as same as that of the coefficient c_{n-2}.

Because the values of v are different in (15), (16), (17), (18), (19) and (20), we compare all gate delays of c_i, and find that the maximum delay is from (17) and (20). Therefore, the time complexity of the Type-A multiplier is

$$1T_A + \left\lceil \log_2(max(4n-4k+4+2^v, 4n-4k-2+2^t)) \right\rceil T_X,$$

where $2^{t-1} < 2n-2k-1 \leq 2^t$ and $2^{v-1} < k \leq 2^v$.

B. Complexities of Type-B multipliers

The method to compute the coefficient c_0 presented before (15) is not optimal for some values of n and k. The Type-B multiplier is designed to overcome this disadvantage at the cost of some more XOR gates. In order to generalize this idea, we define $w(k)$ as the Hamming weight of the integer k. We also define

$$p(k) = \sum_{i=0}^{k} w(k).$$

We first consider the coefficient c_{k-1} and c_{n-1} which include the reusable term s_{k-1}. They each include k product terms $a_i b_j$, and we arrange them in the way discussed in [5]. Thus we have $w(k)$ subtrees: the j-th nonzero bit in the binary expansion of k corresponds to the subtree of height j. After constructing the binary XOR tree of c_{k-1}, we obtain the summation of leaf nodes in each subtree. These $w(k)$ subtree summations can then be XORed into the binary XOR tree of c_{n-1} at the cost of $w(k)$ XOR gates. So the number of XOR gates that can be saved in the reusable term s_{k-1} is $k - w(k)$, while this value in the Type-A multiplier is $k-1$.

Therefore, the number of the AND gates used in this type of multiplier is equal to that of the Type-A multiplier, but the number of the XOR gates that can be saved is different. This number is

$$\begin{aligned}
\phi_{\text{XOR\_B}} &= k-w(k)+(n-k-1)(k-w(k))+ \\
&\quad (n-k-w(n-k))+(2k-n-w(2k-n))+ \\
&\quad \sum_{i=1}^{2k-1-n}(i-w(i))+\sum_{0}^{2k-2-n}(k-1-i-w(k-1-i))+ \\
&\quad \sum_{0}^{2k-2-n}(2k-1-n-i-w(2k-1-n-i)) \\
&= 3k^2-2nk-k+n/2+n^2/2-w(k)(n-k)-w(2k-n)-w(n-k) \\
&\quad -(p(k-1)-p(n-k))-2p(2k-1-n).
\end{aligned}$$

Thus, the total number of the XOR gates of the Type-B multiplier is

$$\begin{aligned}
&2k+\Delta+(3k^2-2nk-k+n/2+n^2/2)-n-\phi_{\text{XOR\_B}} \\
=\;& \Delta+2k-n+ \\
& w(k)(n-k)+w(2k-n)+w(n-k)+2p(2k-1-n)+p(k-1)-p(n-k).
\end{aligned}$$

The time complexity of the Type-B multiplier can be obtained similar to the way we processed c_i in the Type-A multiplier before.

For $0 \le i \le 2k-2-n$, the maximum gate delay is from the coefficient c_0, which is $1T_A + \lceil \log_2(3k-2) \rceil T_X$. For c_{n-1}, the maximum XOR gate delay is $1T_A + \lceil \log_2(3n-k) \rceil T_X$. The gate delays of the other coefficients are not greater than those of c_{n-1} and c_0. Because

$$3n-k > 3k-2 \quad \text{when} \quad n/2 < k < 2n/3,$$

the time complexity of the Type-B multiplier is equal to $1T_A + \lceil \log_2(3n-k) \rceil T_X$.

IV. Type-A and Type-B multipliers for $k = 2n/3$

In this case, the expression of $\langle a \rangle_{x^{n-k}+1}$ is

$$\begin{aligned}
&\sum_{i=0}^{n-k-1} g_i x^i := \langle a \rangle_{x^{n-k}+1} \\
=\;& \sum_{i=0}^{n-k-1} a_i x^i + \sum_{i=0}^{n-k-1} a_{i+n-k} x^i + \sum_{i=0}^{2k-n-1} a_{i+2n-2k} x^i \\
=\;& \sum_{i=0}^{2k-n-1} (a_i + a_{i+n-k} + a_{i+2n-2k}) x^i.
\end{aligned}$$

The expression of $c+q$ is

$$\begin{aligned}
c+q &= \sum_{i=0}^{k-1} s_i x^i + \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j}\right) x^{n-1} + \sum_{i=2n-2k}^{n-1} s_{i-2n+2k} x^i + \sum_{i=k}^{n-1} s_{i-(n-k)} x^i + \\
&\quad \sum_{i=2n-2k}^{n-2} \left(\sum_{j=0}^{i-2n+2k} g_j h_{i-2n+2k-j} + \sum_{j=i-2n+2k+1}^{n-k-1} g_j h_{i-n+k-j}\right) x^i \\
&= \sum_{i=0}^{k-1} s_i x^i + \\
&\quad \sum_{i=k}^{n-2} \left(\sum_{j=0}^{i-2n+2k} g_j h_{i-2n+2k-j} + \sum_{j=i-2n+2k+1}^{n-k-1} g_j h_{i-n+k-j} + s_{i-2n+2k} + s_{i-(n-k)}\right) x^i + \\
&\quad \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j} + s_{2k-n-1} + s_{k-1}\right) x^{n-1}.
\end{aligned}$$

So we can get the result

$$\begin{aligned}
c &= c + (c+q) \\
&= \sum_{i=0}^{k-1} s_i x^i + \\
&\quad \sum_{i=2n-2k}^{n-2} \left(\sum_{j=0}^{i-2n+2k} g_j h_{i-2n+2k-j} + \sum_{j=i-2n+2k+1}^{n-k-1} g_j h_{i-n+k-j} + s_{i-2n+2k} + s_{i-(n-k)}\right) x^i + \\
&\quad \left(\sum_{j=0}^{n-k-1} g_j h_{n-k-1-j} + s_{2k-n-1} + s_{k-1}\right) x^{n-1} + \\
&\quad \sum_{i=0}^{2k-2-n} (s_{i+n} + s_{i+2n-k} + s_{i+3n-2k}) x^i + \sum_{i=2k-1-n}^{k-2} (s_{i+n} + s_{i+2n-k}) x^i + \sum_{i=k-1}^{n-2} s_{i+n} x^i \\
&= \sum_{i=0}^{2k-2-n} \{(s_i) + s_{i+n} + (s_{i+2n-k}] + [s_{i+3n-2k})\} x^i + \\
&\quad \{[[s_{2k-1-n}]] + (((s_{n+k-1}))) + s_{2k-1}\} x^{2k-1-n} + \\
&\quad \sum_{i=2k-n}^{k-2} \{[s_i + [s_{i+2n-k})] + (s_{i+n}]\} x^i + \{((s_{k-1})) + (((s_{n+k-1})))\} x^{k-1} + \\
&\quad \sum_{i=k}^{n-2} \left\{\sum_{j=0}^{i-2n+2k} g_j h_{i-2n+2k-j} + \sum_{j=i-2n+2k+1}^{n-k-1} g_j h_{i-n+k-j} + [s_{i+n} + s_{i-(n-k)}] + (s_{i-2n+2k})\right\} x^i + \\
&\quad \left\{[[s_{2k-n-1}]] + ((s_{k-1})) + \sum_{j=0}^{n-k-1} g_j h_{n-k-1-j}\right\} x^{n-1}.
\end{aligned}$$

A. Type-A and Type-B multipliers

The two types of multipliers are similar to those presented in the previous section.

Table II lists the number of the product terms in each coefficient c_i of x^i. The number of the AND gates in the Type-A and Type-B multipliers are all equal to the number of the product terms excluding the reusable terms. In this case, the number of AND gates that can be saved is equal to that of the case $2/n < k < 2n/3$.

$$\phi_{\text{AND}} = 3k^2 - 2nk - k + n/2 + n^2/2 = n^2/2 - n/6.$$

Therefore, the total number of the AND gates is

$$\begin{aligned}\Delta &= n + 2k + \sum_{i=0}^{2k-2-n} (3k-2-2i) + \\ &\quad \sum_{i=2k-1-n}^{k-2} (n+k-1-i) + \sum_{i=k}^{n-2} (2k+1+i-n) - \phi_{\text{AND}} \\ &= 4nk + n/2 - k - 3k^2 - n^2/2 \\ &= (5n^2 - n)/6,\end{aligned}$$

which coincides with the value of Δ in (14).

The total number of the XOR gates in the Type-A multiplier is the summation of
$2(2(2k-n)) = 2k$ (computing g_i and h_i) and the number of "+" in equation (II) excluding the reusable terms. The number of the XOR gates that can be saved in the Type-A multiplier is equal to that of the case $2/n < k < 2n/3$:

$$\phi_{\text{XOR\_A}} = 3k^2 - 2nk - 6k + 5n/2 + n^2/2 + 1 = n^2/2 - 3n/2 + 1.$$

So the total number of the XOR gates of the Type-A multiplier is

$$2k + [(\Delta + \phi_{\text{AND}}) - n] - \phi_{\text{XOR\_A}} = \Delta + 7k - 3n - 1 = (5n^2 + 9n)/6 - 1.$$

Similarly, the number of the XOR gates that can be saved in the Type-B multiplier is also the same as that of the case $2/n < k < 2n/3$:

$$\begin{aligned}\phi_{\text{XOR\_B}} &= 3k^2 - 2nk - k + n/2 + n^2/2 - w(k)(n-k) - w(2k-n) - w(n-k) \\ &\quad -(p(k-1) - p(n-k)) - 2p(2k-1-n) \\ &= n^2/2 - n/6 - w(k)(n-k) - w(2k-n) - w(n-k) \\ &\quad -(p(k-1) - p(n-k)) - 2p(2k-1-n). \qquad (21)\end{aligned}$$

So the total number of the XOR gates of the Type-B multiplier is:

$$\begin{aligned} & 2k + \Delta + \phi_{\text{AND}} - n - \phi_{\text{XOR\_B}} \\ = \; & (5n^2 + n)/6 + \\ & w(k)(n-k) + w(2k-n) + w(n-k) + 2p(2k-1-n) + \\ & p(k-1) - p(n-k). \end{aligned} \tag{22}$$

TABLE II

THE NUMBER OF THE PRODUCT TERMS IN THE COEFFICIENT c_i OF x^i WHILE $k = 2n/3$.

x^i	The number of the product terms
$0 \le i \le 2k-2-n$	$(\underline{i+1}) + (2n-1-(n+i)) + \underline{(2n-1-(i+2n-k))} + \overline{(2n-1-(i+3n-2k))} = 3k-2-2i$
$i = 2k-1-n$	$(\underline{2k-1-n+1}) + \underline{(2n-1-(n+k-1))} + (2n-1-(n+2k-1-n)) = 2n-k$
$2k-n \le i \le k-2$	$[\underline{(i+1)+\overline{(2n-1-(i+2n-k))}}] + \underline{(2n-1-(n+i))} = n+k-1-i$
$i = k-1$	$\underline{k} + \underline{(2n-1-(n+k-1))} = n$
$k \le i \le n-2$	$(n-k) + [\underline{(2n-1-(n+i)) + (i-n+k+1)}] + (\underline{i-2n+2k+1}) = 2k+1-n+i$
$i = n-1$	$(n-k) + \underline{k} + (\underline{2k-1-n+1}) = 2k$

Now we discuss the gate delay of the case $k = 2n/3$.

For $0 \le i \le 2k-n-2$, the maximum gate delay is from the coefficient

$$c_0 = s_0 + s_n + s_{2n-k} + s_{3n-2k},$$

which includes $1 + (n-1) + [\underline{k-1}] + [\underline{2k-1-n}] = 3k-2$ product terms $a_i b_j$. We can get the explicit gate delay in the Type-A multiplier using the way we discussed above, and it is

$$v + \left\lceil \log_2\left(\left\lceil \frac{n+1}{2^v} \right\rceil + 1\right) \right\rceil = \lceil \log_2(3k/2 + 1 + 2^v) \rceil$$

by lemma 1 and $2^{v-1} < k-1 \le 2^v$.

As for the Type-B multiplier, the gate delay is $1T_A + \lceil \log_2(3k-2) \rceil T_X$.

For $i = 2k-n-1$, the coefficient

$$c_{2k-n-1} = s_{2k-n-1} + s_{n+k-1} + s_{2k-1}$$

includes $[\underline{2k-n}] + [\underline{n-k}] + [2n-2k] = 2n-k = 2k$ product terms $a_i b_j$. For the Type-A multiplier, the gate delay is

$$v + \left\lceil \log_2\left(\left\lceil \frac{2n-2k+1}{2^v} \right\rceil + 1\right) \right\rceil = \lceil \log_2(k + 1 + 2^v) \rceil$$

by lemma 1 and $2^{v-1} < n-k \leq 2^v$. For the Type-B multiplier, the gate delay is $1T_A + \lceil \log_2(2k) \rceil \, T_X$.

For c_{k-1}, the coefficient

$$c_{k-1} = s_{n+k-1} + s_{k-1}$$

includes $\underline{[n-k]} + \underline{[k]} = n = 3k/2$ product terms $a_i b_j$. For the Type-A multiplier, the total XOR gate delay to compute the coefficient c_{k-1} is

$$v + \lceil \log_2(1+1) \rceil = \lceil \log_2(2^{v+1}) \rceil \leq \lceil \log_2(2k + 2^v) \rceil$$

by lemma 1 and $2^{v-1} < k \leq 2^v$. For the Type-B multiplier, the total XOR gate delay is $1T_A + \lceil \log_2(3k/2) \rceil \, T_X$.

For $k \leq i \leq n-2$, the maximum gate delay is from the coefficient

$$c_{n-2} = [s_{2n-2} + s_{k-2}] + [s_{2k-n-2}] + \sum_{j=0}^{2k-n-2} g_j h_{2k-n-2-j} + \sum_{j=2k-n-1}^{n-k-1} g_j h_{n-2-j}.$$

It includes a reusable term $[s_{2n-2} + s_{k-2}]$, a reusable term $[s_{2k-n-2}]$ and $n-k$ product terms $g_i h_j$. For the Type-A multiplier, the total XOR gate delay is

$$v + \left\lceil \log_2(\left\lceil \frac{n-k+1}{2^{v-2}} \right\rceil + 1) \right\rceil = \lceil \log_2(2k + 4 + 2^v) \rceil \leq \lceil \log_2(4k+4) \rceil$$

by lemma 1 and $2^{v-1} < k \leq 2^v$. For the Type-B multiplier, the total gate delay is $1T_A + \lceil 2 + \log_2(\lceil \frac{3k-n-2}{4} \rceil + (n-k)) \rceil \; T_X = 1T_A + \lceil \log_2(7k/2 - 2) \rceil \; T_X$.

For c_{n-1}, the gate delay of Type-A multiplier is as same as that of the coefficient c_{n-2}. For the Type-B multiplier, the total XOR gate delay is $1T_A + \lceil \log_2(7k/2) \rceil \, T_X$.

In summary, the max gate delays of Type-A and Type-B multipliers are $1T_A + \lceil \log_2(2k + 4 + 2^v) T_X \rceil$ and $1T_A + \lceil \log_2(7k/2) \rceil \, T_X$ respectively for the case $k = 2n/3$.

A simple calculation show that the above expressions of space and time complexities for the case $k = 2n/3$ are all coincides with those for the case $k \in (n/2, 2n/3)$ obtained in the previous section. Therefore, we combine them together and make a comparison in the next section.

V. Comparisons

We list the complexities of the proposed multipliers and other trinomial-based parallel multipliers in Table III. Because their AND gate delays are all $1T_A$, we ignore them and list only XOR gate delays in the last column.

TABLE III

COMPLEXITIES OF $f(u) = u^n + u^k + 1$-BASED PARALLEL MULTIPLIERS

Multipliers	# AND Gate	# XOR Gate	XOR Gate Delay
Quadratic, $2k > n$ [8] [9]	n^2	$n^2 - 1$	$\lceil \log_2(n+k) \rceil$
Quadratic, $2k < n$ [8] [9]	n^2	$n^2 - 1$	$\lceil \log_2(2n-k-1) \rceil$
Karatsuba hybrid n even [7]	$\frac{3n^2}{4}$	$\frac{3n^2}{4} + 2.5n + k - 4$	$\lceil \log_2(8n-8) \rceil$
Karatsuba hybrid n odd [7]	$\frac{3n^2+2n-1}{4}$	$\frac{3n^2}{4} + 4n + k - 5.75$	$\lceil \log_2(8n-8) \rceil$
PB Type-A, $k \in (1, n/2)$ [5]	Δ	$\Delta + 3k - n$	$\lceil \log_2(max(3n-3k-1, 2n-2k+2^v)) \rceil$
PB Type-B, $k \in (1, n/2)$ [5]	Δ	$\Delta + 2k - n + k \cdot w(k)$	$\lceil \log_2(3n-3k-1) \rceil$
PB Type-A, $k \in (n/2, 2n/3]$	Δ	$\Delta + 7k - 3n - 1$	$\lceil \log_2(max(4n-4k+4+2^v, 4n-4k-2+2^t)) \rceil$
PB Type-B, $k \in (n/2, 2n/3]$	Δ	$\Delta + 2k - n + \sigma$	$\lceil \log_2(3n-k) \rceil$
where $\Delta = n^2 + (n-k)(n-1-3k)/2$, $2^{v-1} < k \leq 2^v$, $2^{t-1} < 2n-2k-1 \leq 2^t$ and $\sigma = w(k)(n-k) + w(n-k) + w(2k-n) + 2p(2k-1-n) + p(k-1) - p(n-k)$.			

The space complexity of the fastest quadratic parallel multipliers for irreducible trinomials are n^2 AND gates and $n^2 - 1$ XOR gates. It is easy to check that the following two inequalities are *always* valid for $n > 4$ and $k \in (n/2, 2n/3]$:

$$\begin{cases} \text{\# AND gates:} & \Delta = n^2 + (n-k)(n-1-3k)/2 < n^2, \\ \text{\# XOR gates:} & \Delta + 7k - 3n - 1 < n^2 - 1. \end{cases}$$

Therefore, the space complexity of the proposed multiplier is always less than that of the fastest quadratic parallel multipliers. As for the time complexity, thanks to the property of the ceiling function "$\lceil \cdot \rceil$", these multipliers may have the *same* XOR gate delay, depending on the values of n and k.

For the purpose of comparison, we examine the 317 values of n such that $n \in [5, 999]$ and $u^n + u^k + 1$ is irreducible over $GF(2)$ for some $k \in (n/2, 2n/3]$. Compared to the current fastest parallel multipliers – quadratic multipliers, the space complexities are reduced, for all these 317 values of n, by 15.3% on average. Especially, for the 124 values of such n, the two kinds of multipliers have the same time complexity, but the space complexities are reduced by 15.5% on average. On the other hand, the CRT-based multipliers in [5] consider the case $k \in [2, n/2]$, and reduce the space complexities by only 8.4% on average.

VI. An example: $f = u^{21} + u^{14} + 1$

From the expression of the quotient q of $a \cdot b$ divided by $(f+1) = x^{21} + x^{14}$, i.e., (7), we have

$$\begin{aligned} q &= s_{40}x^{19} + s_{39}x^{18} + s_{38}x^{17} + s_{37}x^{16} + s_{36}x^{15} + s_{35}x^{14} + s_{34}x^{13} \\ &\quad +(s_{33}+s_{40})x^{12} + (s_{32}+s_{39})x^{11} + (s_{31}+s_{38})x^{10} + (s_{30}+s_{37})x^{9} \\ &\quad +(s_{29}+s_{36})x^{8} + (s_{28}+s_{35})x^{7} + (s_{27}+s_{34})x^{6} \\ &\quad +(s_{26}+s_{33}+s_{40})x^{5} + (s_{25}+s_{32}+s_{39})x^{4} + (s_{24}+s_{31}+s_{38})x^{3} + (s_{23}+s_{30}+s_{37})x^{2} \\ &\quad +(s_{22}+s_{29}+s_{36})x + (s_{21}+s_{28}+s_{35}). \end{aligned}$$

In order to obtain $(c+q) = \langle a \cdot b \rangle_{x^{14}(x^7+1)}$ using the CRT, we compute $\langle a \rangle_{x^7+1}$ and $\langle b \rangle_{x^7+1}$ first:

$$\begin{aligned} \langle a \rangle_{x^7+1} &= \sum_{i=0}^{6} g_i x^i \\ &= (a_0+a_7+a_{14}) + (a_1+a_8+a_{15})x + (a_2+a_9+a_{16})x^2 + (a_3+a_{10}+a_{17})x^3 \\ &\quad +(a_4+a_{11}+a_{18})x^4 + (a_5+a_{12}+a_{19})x^5 + (a_6+a_{13}+a_{20})x^6; \end{aligned}$$

$$\begin{aligned} \langle b \rangle_{x^7+1} &= \sum_{i=0}^{6} h_i x^i \\ &= (b_0+b_7+b_{14}) + (b_1+b_8+b_{15})x + (b_2+b_9+b_{16})x^2 + (b_3+b_{10}+b_{17})x^3 \\ &\quad +(b_4+b_{11}+b_{18})x^4 + (b_5+b_{12}+b_{19})x^5 + (b_6+b_{13}+b_{20})x^6. \end{aligned}$$

Next, we compute the two intermediate values in the CRT formula, i.e., (9) and (11), using the schoolbook polynomial multiplication algorithm:

$$\langle a \cdot b \rangle_{x^{14}} = \sum_{i=0}^{13} s_i x^i$$

and

$$\begin{aligned}
\langle a\cdot b\rangle_{x^{n-k}+1} &= \left\langle \langle a\rangle_{x^7+1}\cdot\langle b\rangle_{x^7+1}\right\rangle_{x^7+1}\\
&= (g_5h_1+g_0h_6+g_1h_5+g_2h_4+g_3h_3+g_4h_2+g_6h_0)x^6+\\
&\quad (g_2h_3+g_5h_0+g_0h_5+g_3h_2+g_4h_1+g_1h_4+g_6h_6)x^5+\\
&\quad (g_1h_3+g_2h_2+g_3h_1+g_0h_4+g_4h_0+g_5h_6+g_6h_5)x^4+\\
&\quad (g_1h_2+g_2h_1+g_0h_3+g_3h_0+g_6h_4+g_5h_5+g_4h_6)x^3+\\
&\quad (g_1h_1+g_0h_2+g_2h_0+g_6h_3+g_5h_4+g_3h_6+g_4h_5)x^2+\\
&\quad (g_1h_0+g_0h_1+g_6h_2+g_5h_3+g_2h_6+g_3h_5+g_4h_4)x+\\
&\quad (g_0h_0+g_4h_3+g_1h_6+g_2h_5+g_5h_2+g_6h_1+g_3h_4).
\end{aligned}$$

Then, from the CRT expression of the remainder $(c+q)=\langle a\cdot b\rangle_{f+1}$, i.e., (8), we have

$$\begin{aligned}
c+q &= \left\langle \langle a\cdot b\rangle_{x^k}\cdot(x^{n-k}+1)\cdot(x^{n-k}+1)+\langle a\cdot b\rangle_{x^{n-k}+1}\cdot x^k\cdot x^{2n-3k}\right\rangle_{x^n+x^k}\\
&= s_0+s_1x^1+s_2x^2+s_3x^3+s_4x^4+s_5x^5+s_6x^6+s_7x^7+s_8x^8+s_9x^9+\\
&\quad +s_{10}x^{10}+s_{11}x^{11}+s_{12}x^{12}+s_{13}x^{13}+\\
&\quad (s_0+s_7+g_0h_0+g_4h_3+g_1h_6+g_2h_5+g_5h_2+g_6h_1+g_3h_4)x^{14}+\\
&\quad (s_1+s_8+g_1h_0+g_0h_1+g_6h_2+g_5h_3+g_2h_6+g_3h_5+g_4h_4)x^{15}+\\
&\quad (s_2+s_9+g_1h_1+g_0h_2+g_2h_0+g_6h_3+g_5h_4+g_3h_6+g_4h_5)x^{16}+\\
&\quad (s_3+s_{10}+g_1h_2+g_2h_1+g_0h_3+g_3h_0+g_6h_4+g_5h_5+g_4h_6)x^{17}+\\
&\quad (s_4+s_{11}+g_1h_3+g_2h_2+g_3h_1+g_0h_4+g_4h_0+g_5h_6+g_6h_5)x^{18}+\\
&\quad (s_5+s_{12}+g_2h_3+g_5h_0+g_0h_5+g_3h_2+g_4h_1+g_1h_4+g_6h_6)x^{19}+\\
&\quad (s_6+s_{13}+g_5h_1+g_0h_6+g_1h_5+g_2h_4+g_3h_3+g_4h_2+g_6h_0)x^{20}.
\end{aligned}$$

Finally, by (13), the expression of $c = ab = q + (c + q)$ is

$$\begin{aligned}
c \;=\; & ((s_0) + s_{21} + \underline{s_{28}} + \overline{s_{35}}) + ((s_1) + s_{22} + \underline{s_{29}} + \overline{s_{36}})x + \\
& ((s_2) + s_{23} + \underline{s_{30}} + \overline{s_{37}})x^2 + ((s_3) + s_{24} + \underline{s_{31}} + \overline{s_{38}})x^3 + \\
& ((s_4) + s_{25} + \underline{s_{32}} + \overline{s_{39}})x^4 + ((s_5) + s_{26} + \underline{s_{33}} + \overline{s_{40}})x^5 + \\
& \{[[s_6]] + s_{27} + (((s_{34})))\}x^6 + \\
& +([s_7 + \overline{s_{35}}] + \underline{s_{28}})x^7 + ([s_8 + \overline{s_{36}}] + \underline{s_{29}})x^8 + ([s_9 + \overline{s_{37}}] + \underline{s_{30}})x^9 + \\
& ([s_{10} + \overline{s_{38}}] + \underline{s_{31}})x^{10} + ([s_{11} + \overline{s_{39}}] + \underline{s_{32}})x^{11} + ([s_{12} + \overline{s_{40}}] + \underline{s_{33}})x^{12} + \\
& \{((s_{13})) + (((s_{34})))\}x^{13} + \\
& ((s_0) + [s_7 + s_{35}] + g_0h_0 + g_4h_3 + g_1h_6 + g_2h_5 + g_5h_2 + g_6h_1 + g_3h_4)x^{14} + \\
& ((s_1) + [s_8 + s_{36}] + g_1h_0 + g_0h_1 + g_6h_2 + g_5h_3 + g_2h_6 + g_3h_5 + g_4h_4)x^{15} + \\
& ((s_2) + [s_9 + s_{37}] + g_1h_1 + g_0h_2 + g_2h_0 + g_6h_3 + g_5h_4 + g_3h_6 + g_4h_5)x^{16} + \\
& ((s_3) + [s_{10} + s_{38}] + g_1h_2 + g_2h_1 + g_0h_3 + g_3h_0 + g_6h_4 + g_5h_5 + g_4h_6)x^{17} + \\
& ((s_4) + [s_{11} + s_{39}] + g_1h_3 + g_2h_2 + g_3h_1 + g_0h_4 + g_4h_0 + g_5h_6 + g_6h_5)x^{18} + \\
& ((s_5) + [s_{12} + s_{40}] + g_2h_3 + g_5h_0 + g_0h_5 + g_3h_2 + g_4h_1 + g_1h_4 + g_6h_6)x^{19} + \\
& ([[s_6]] + ((s_{13})) + g_5h_1 + g_0h_6 + g_1h_5 + g_2h_4 + g_3h_3 + g_4h_2 + g_6h_0)x^{20}.
\end{aligned}$$

The following table compares the space and time complexities of the proposed multipliers and the multipliers in [5]. We note that there are only four degree-21 irreducible trinomials over $GF(2)$: $x^{21} + x^2 + 1$, $x^{21} + x^7 + 1$ and their reciprocal polynomials. The best choice for the multiplier in [5] is $x^{21} + x^7 + 1$.

TABLE IV
COMPLEXITIES OF CRT-BASED $GF(2^{21})$ PARALLEL MULTIPLIERS.

Trinomial	Multiplier	# AND Gate	# XOR Gate	XOR Gate Delay
$x^{21} + x^7 + 1$, [5]	Type-A	434	434	6
	Type-B	434	448	6
$x^{21} + x^{14} + 1$, proposed	Type-A	364	398	6
	Type-B	364	429	6

VII. Conclusions

In this work, we consider the irreducible trinomial $u^n + u^k + 1$ over $GF(2)$ for some $k \in (n/2, 2n/3]$. By selecting the largest possible value of $k \in (n/2, 2n/3]$, we further reduce the space complexity of the CRT-based hybrid parallel $GF(2^n)$ polynomial basis multipliers. Compared to the current fastest parallel multipliers – quadratic multipliers, the AND and XOR gate complexities of the proposed multipliers are always less than those of the quadratic multipliers. Thanks to the property of the ceiling function "$\lceil \cdot \rceil$", they have the same time complexity for some irreducible trinomials.

For the purpose of comparison, we examine the 317 values of n such that $n \in [5, 999]$ and $u^n + u^k + 1$ is irreducible over $GF(2)$ for some $k \in (n/2, 2n/3]$. The space complexities are reduced, for all these 317 values of n, by 15.3% on average. Especially, for the 124 values of such n, the two kinds of multipliers have the same time complexity, but the space complexities are reduced by 15.5% on average. On the other hand, the CRT-based multipliers in [5] consider the case $k \in [2, n/2]$, and reduce the space complexities by only 8.4% on average.

VIII. Acknowledgment

The work was supported by NSFC under grant No.s 61373141 and 91218302.

References

[1] C. Grabbe, M. Bednara, J. Shokrollahi, J. Teich, and J. von zur Gathen, "FPGA designs of parallel high performance $GF(2^{233})$ multipliers," in *Proc. Int. Symposium on Circuits and Systems (ISCAS 2003), vol. II*, 2003, pp. 268–271.

[2] F. Rodríguez-Henríquez and Ç. K. Koç, "On fully parallel Karatsuba multipliers for $GF(2^m)$," in *Proc. Int. Conf. Computer Science and Technology (CST 2003).* ACTA Press, 2003, pp. 405–410.

[3] J. von zur Gathen and J. Shokrollahi, "Efficient FPGA-based Karatsuba multipliers for polynomials over F_2," in *Proc. 12th Workshop on Selected Areas in Cryptography (SAC 2005).* Springer, 2006, pp. 359–369.

[4] M. A. Hasan, N. Méloni, A. H. Namin, and C. Negre, "Block recombination approach for subquadratic space complexity binary field multiplication based on Toeplitz matrix-vector product," *IEEE Transactions on Computers*, vol. 61, no. 2, pp. 151–163, 2012.

[5] H. Fan, "A chinese remainder theorem approach to bit-parallel $GF(2^n)$ polynomial basis multipliers for irreducible trinomials," *IEEE Transactions on Computers, accepted. DOI: 10.1109TC.2015.2428704*, 2015.

[6] ——, "A chinese remainder theorem approach to bit-parallel $GF(2^n)$ polynomial basis multipliers for irreducible trinomials," *IACR Cryptology ePrint Archive, Report 2014/972*, 2014.

[7] M. Elia, M. Leone, and C. Visentin, "Low complexity bit-parallel multipliers for $GF(2^m)$ with generator polynomial $x^m + x^k + 1$," *IEE Electronics Letters*, vol. 35, no. 7, pp. 551–552, 1999.

[8] H. Fan and M. A. Hasan, "Fast bit parallel-shifted polynomial basis multipliers in $GF(2^n)$," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 12, pp. 2606–2615, 2006.

[9] A. Hariri and A. Reyhani-Masoleh, “Bit-serial and bit-parallel Montgomery multiplication and squaring over $GF(2^m)$,” *IEEE Transactions on Computers*, vol. 58, no. 10, pp. 1332–1345, 2009.