Overlap-free Karatsuba-Ofman Polynomial Multiplication Algorithm

Haining Fan, Jiaguang Sun, Ming Gu and Kwok-Yan Lam

Abstract

We describe how a recently proposed way to split input operands allows for fast VLSI implementations of $GF(2)[x]$ Karatsuba-Ofman multipliers. The XOR gate delay of the proposed multiplier is better than that of previous Karatsuba-Ofman multipliers. For example, it is reduced by about 33% and 25% for $n = 2^i$ and $n = 3^i$ $(i > 1)$, respectively.

Index Terms

Karatsuba algorithm, Karatsuba-Ofman algorithm, polynomial multiplication, subquadratic space complexity multiplier, finite field.

I. Introduction

Published in 1962 [1], Karatsuba-Ofman algorithm (KOA) was the first integer multiplication method broke the quadratic complexity barrier in positional number systems. Due to its simplicity, its polynomial version is widely adopted to design VLSI $GF(2^n)$ parallel multipliers in $GF(2^n)$-based cryptosystems [9]-[27]. Two parameters are often used to measure the performance of a $GF(2^n)$ parallel multiplier, namely, the space and time complexities. The space complexity is often represented in terms of the total number of 2-input XOR and AND gates used. The corresponding time complexity is given in terms of the maximum delay faced by a signal due to these XOR and AND gates. Symbols "T_A" and "T_X" are often used to represent the delay of one 2-input AND and XOR gates, respectively.

E-mails: fan_haining@yahoo.com, {sunjg, guming, lamky}@tsinghua.edu.cn

The existing bit parallel $GF(2^n)$ multipliers may be simply classified into the following three categories according to the asymptotic space complexity of the multiplication algorithm : subquadratic, quadratic and hybrid multipliers. KOA has been used in many subquadratic and hybrid multipliers. These multipliers first perform a KOA-based multiplication of two input binary polynomials $A = \sum_{i=0}^{n-1} a_i x^i$ and $B = \sum_{i=0}^{n-1} b_i x^i$, and then a modulo reduction operation using the field generating irreducible polynomial. To explain the general idea of KOA easily, we will assume that $n = 2m = 2^i$ $(i > 1)$ in the following.

First, previous KOA implementations split polynomials A and B into the "most significant half" and the "least significant half" as follows:

$$A = \sum_{i=0}^{n-1} a_i x^i = x^m \sum_{i=0}^{m-1} a_{m+i} x^i + \sum_{i=0}^{m-1} a_i x^i = x^m A_H + A_L,$$

$$B = \sum_{i=0}^{n-1} b_i x^i = x^m \sum_{i=0}^{m-1} b_{m+i} x^i + \sum_{i=0}^{m-1} b_i x^i = x^m B_H + B_L,$$

where $A_H = \sum_{i=0}^{m-1} a_{m+i} x^i$, $A_L = \sum_{i=0}^{m-1} a_i x^i$, B_H and B_L are defined similarly.

Then the product AB is computed recursively using

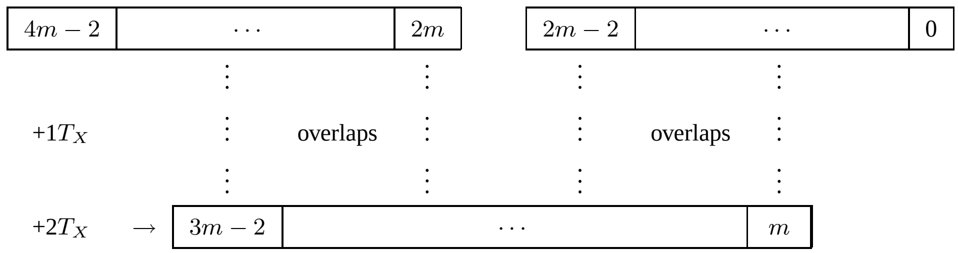
$$AB = A_H B_H x^{2m} + \{[(A_H + A_L)(B_H + B_L)] - [A_H B_H + A_L B_L]\} x^m + A_L B_L. \tag{1}$$

Please note that "$-$" is the same as "+" in $GF(2)[x]$. For the VLSI implementation of (1), terms in square brackets are calculated concurrently, and therefore two XOR gate delays, i.e., $2T_X$, are required to compute the expression in curly brackets besides the cost to compute the three half sized products.

Finally, the three polynomials $A_H B_H x^{2m}$, $[(A_H + A_L)(B_H + B_L) - A_H B_H - A_L B_L] x^m$ and $A_L B_L$ in (1) are XORed in an overlap module by adding coefficients of common exponents of x together [26]. In order to explain overlaps of common exponents of x clearly, we present the following table, which shows ranges of x's exponents in these three polynomials, and make two remarks about overlaps.

TABLE I

RANGES OF x'S EXPONENTS IN THREE POLYNOMIALS OF (1)



Remark 1: Overlaps occur only when $n \geq 4$ (or $m \geq 2$), and there is no overlap when $n = 2$ (or $m = 1$). Because of these overlaps, one XOR gate delay T_X is required in the overlap module. Therefore, a total of 3 XOR gate delays, i.e., $3T_X$, are required in (1) besides the cost of the recursive computation of three half sized products.

Remark 2: Let $n = kd$ ($k > 1$ and $d > 0$), previous generalizations of KOA split the two input operands into k successive block each with d coefficients. Since the product of two degree-($d-1$) polynomials is a polynomial of degree-($2d-2$), overlaps always exist if $d > 1$.

We now compute exact complexities of the above binary polynomial KOA (1). First, we introduce some symbols of [5]. Let $\mathcal{S}$ and $\mathcal{D}$ stand for "Space" and "Delay", respectively. We use $\mathcal{S}^{\otimes}(n)$, $\mathcal{S}^{\oplus}(n), \mathcal{D}^{\otimes}(n)$ and $\mathcal{D}^{\oplus}(n)$ to denote the number of multiplication (AND) and addition (XOR) operations, the time delays introduced by multiplication and addition operations, respectively. Then the following recurrence relations, which describe the complexities of KOA, can be established.

$$\begin{cases} \mathcal{S}^{\otimes}(2) = 3, \\ \mathcal{S}^{\otimes}(n) = 3\mathcal{S}^{\otimes}(n/2); \end{cases} \qquad \begin{cases} \mathcal{D}^{\otimes}(2) = 1, \\ \mathcal{D}^{\otimes}(n) = \mathcal{D}^{\otimes}(n/2); \end{cases}$$

$$\begin{cases} \mathcal{S}^{\oplus}(2) = 4, \\ \mathcal{S}^{\oplus}(n) = 3\mathcal{S}^{\oplus}(n/2) + 4n - 4; \end{cases} \quad \text{and} \quad \begin{cases} \mathcal{D}^{\oplus}(2) = 2, \\ \mathcal{D}^{\oplus}(n) = \mathcal{D}^{\oplus}(n/2) + 3. \end{cases}$$

After solving the above recurrence relations using formulae derived in [5], we have the following complexity results for the binary polynomial KOA [9], [26].

$$\begin{cases} \mathcal{S}^{\otimes}(n) = n^{\log_2 3}, \\ \mathcal{S}^{\oplus}(n) = 6n^{\log_2 3} - 8n + 2, \\ \mathcal{D}^{\otimes}(n) = 1, \\ \mathcal{D}^{\oplus}(n) = 3\log_2 n - 1. \end{cases}$$

Besides KOA, a Toeplitz matrix-vector product approach was presented recently to construct subquadratic $GF(2^n)$ multipliers [5]. It takes advantage of a shifted polynomial basis [6] and applies the coordinate transformation technique of [7] and [8]. Both the space and time complexities of the resulting multiplier are better than those of the best KOA-based subquadratic multipliers. For example, with $n = 2^i$ ($i > 0$), the space complexity is about 8% better, while the time complexity is about 33% better, respectively.

Since these Toeplitz matrix-vector product formulae are obtained by transposing [3, Th6, p.17] corresponding polynomial KOA-like formulae, the following question arises naturally: is it possible to reduce the time or space complexity of KOA further? We answer this question positively in the next section. We will propose a fast VLSI implementations of the polynomial Karatsuba-Ofman algorithm. It applies a recently proposed method to split input operands [2]. The XOR gate delay of the proposed $GF(2)[x]$ multiplier is better than that of previous Karatsuba-Ofman multipliers. For example, it is reduced by about 33% and 25% for $n = 2^i$ and $n = 3^i$ $(i > 1)$, respectively.

II. Fast Polynomial KOA Implementation

We first introduce the splitting method in [2], where it is used to compute the short product of two power series. Instead of splitting input operands into the "most significant half" and the "least significant half", the method split operands according to the parity of x's exponent. That is to say, we may rewrite A and B as follows

$$A = \sum_{i=0}^{n-1} a_i x^i = \sum_{i=0}^{m-1} a_{2i} x^{2i} + \sum_{i=0}^{m-1} a_{2i+1} x^{2i+1} = \sum_{i=0}^{m-1} a_{2i} x^{2i} + x \sum_{i=0}^{m-1} a_{2i+1} x^{2i},$$

$$B=\sum_{i=0}^{n-1} b_i x^i=\sum_{i=0}^{m-1} b_{2i}x^{2i}+\sum_{i=0}^{m-1} b_{2i+1}x^{2i+1}=\sum_{i=0}^{m-1} b_{2i}x^{2i}+x\sum_{i=0}^{m-1} b_{2i+1}x^{2i}.$$

Now let $y=x^2$, $A_e(y)=\sum_{i=0}^{m-1} a_{2i}y^i$ and $A_o(y)=\sum_{i=0}^{m-1} a_{2i+1}y^i$. $B_e(y)$ and $B_o(y)$ are defined similarly. Operands A and B can be rewritten as $A=A_e(y)+xA_o(y)$ and $B=B_e(y)+xB_o(y)$. Please note that terms $A_e(y)$, $A_o(y)$, $B_e(y)$ and $B_o(y)$ are polynomials in y of degrees less than m. Therefore multiplication operations between them may also be computed recursively. Using the above splitting method of A and B, we have the following KOA-like formula

$$\begin{aligned} AB &= (A_e(y)+xA_o(y))(B_e(y)+xB_o(y)) \\ &= \{A_e(y)B_e(y)+x^2A_o(y)B_o(y)\}+x\{A_e(y)B_o(y)+A_o(y)B_e(y))\} \\ &= \{A_e(y)B_e(y)+yA_o(y)B_o(y)\}+ \\ &\quad x\{[(A_e(y)+A_o(y))(B_e(y)+B_o(y))]+[A_e(y)B_e(y)+A_o(y)B_o(y)]\}. \end{aligned} \tag{2}$$

For the VLSI implementation of (2), multiplying a polynomial by x or $y=x^2$ is equivalent to shifting its coefficients, and no gate is required. It is easy to see that the expansion of $\{A_e(y)B_e(y)+yA_o(y)B_o(y)\}$ in (2) contains only terms with even exponents of x since $y=x^2$, and the expansion of $x\{[(A_e(y)+A_o(y))(B_e(y)+B_o(y))]+[A_e(y)B_e(y)+A_o(y)B_o(y)]\}$ contains only terms with odd exponents of x. Thus, there is no overlap exists when computing their summation, and no gate is required either. Moreover, terms in square brackets can be computed concurrently, and the addition operation requires 1 XOR gate delay T_X. Therefore, we know that computing AB via (2) needs $only$ a total of $2T_X$ besides the cost of the recursive computation of three half sized products. Please recall that the corresponding XOR gate delay is $3T_X$ in (1). Consequently, the following recurrence relations, which describe the algorithm complexities, can be established.

$$\begin{cases} \mathcal{S}^{\otimes}(2) = 3, \\ \mathcal{S}^{\otimes}(n) = 3\mathcal{S}^{\otimes}(n/2); \end{cases} \qquad \begin{cases} \mathcal{D}^{\otimes}(2) = 1, \\ \mathcal{D}^{\otimes}(n) = \mathcal{D}^{\otimes}(n/2); \end{cases}$$

$$\begin{cases} \mathcal{S}^{\oplus}(2) = 4, \\ \mathcal{S}^{\oplus}(n) = 3\mathcal{S}^{\oplus}(n/2) + 4n - 4; \end{cases} \quad \text{and} \quad \begin{cases} \mathcal{D}^{\oplus}(2) = 2, \\ \mathcal{D}^{\oplus}(n) = \mathcal{D}^{\oplus}(n/2) + 2. \end{cases}$$

Their solutions are as follows:

$$\begin{cases} \mathcal{S}^{\otimes}(n) = n^{\log_2 3}, \\ \mathcal{S}^{\oplus}(n) = 6n^{\log_2 3} - 8n + 2, \\ \mathcal{D}^{\otimes}(n) = 1, \\ \mathcal{D}^{\oplus}(n) = 2\log_2 n. \end{cases}$$

Compared to previous implementations of polynomial KOA , the XOR gate delay of (2), i.e., $\mathcal{D}^{\oplus}(n)$, is about 33% better for $n = 2^i$ ($i > 0$).

Similar to the generalizations of KOA, we may also derive some KOA-like formulae for polynomials of higher degrees. As an example, we show the formula for $n = 3k = 3^i$ $(i > 1)$. Let $y = x^3$ and split A as follows

$$\begin{aligned} A &= \sum_{i=0}^{n-1} a_i x^i = \sum_{i=0}^{k-1} a_{3i} x^{3i} + x \sum_{i=0}^{k-1} a_{3i+1} x^{3i} + x^2 \sum_{i=0}^{k-1} a_{3i+2} x^{3i} \\ &= A_0(y) + xA_1(y) + x^2 A_2(y), \end{aligned}$$

where $A_0(y) = \sum_{i=0}^{k-1} a_{3i} y^i$, $A_1(y) = \sum_{i=0}^{k-1} a_{3i+1} y^i$ and $A_2(y) = \sum_{i=0}^{k-1} a_{3i+2} y^i$.

Then we have

$$\begin{aligned} AB &= \{A_0B_0 + y[(A_1 + A_2)(B_1 + B_2) + A_1B_1 + A_2B_2]\} + \\ &\quad x\{[(A_0 + A_1)(B_0 + B_1) + A_0B_0 + A_1B_1] + yA_2B_2\} + \\ &\quad x^2\{(A_0 + A_2)(B_0 + B_2) + A_0B_0 + A_2B_2 + A_1B_1\}, \end{aligned}$$

where "(y)"s in expressions $A_i(y)$ and $B_i(y)$ are omitted.

The following recurrence relations describe the complexities of this formula.

$$\begin{cases} \mathcal{S}^{\otimes}(3) = 6, \\ \mathcal{S}^{\otimes}(n) = 6\mathcal{S}^{\otimes}(n/3); \end{cases} \qquad \begin{cases} \mathcal{D}^{\otimes}(3) = 1, \\ \mathcal{D}^{\otimes}(n) = \mathcal{D}^{\otimes}(n/3); \end{cases}$$

$$\begin{cases} \mathcal{S}^{\oplus}(3) = 12, \\ \mathcal{S}^{\oplus}(n) = 6\mathcal{S}^{\oplus}(n/3) + \frac{22}{3}n - 10; \end{cases} \quad \text{and} \quad \begin{cases} \mathcal{D}^{\oplus}(3) = 3, \\ \mathcal{D}^{\oplus}(n) = \mathcal{D}^{\oplus}(n/3) + 3. \end{cases}$$

III. Comparisons

Table II compares asymptotic complexities of proposed formulae with the previous KOA and Toeplitz matrix-vector product (TMVP) formulae over the ground field $GF(2)$, where $\#$AND and $\#$XOR denote the total number of AND and XOR gates, respectively. The size of operands is assumed to be $n = 2^t$ or 3^t ($t > 0$). We list complexities of the TMVP in the table because both KOA and TMVP can be used to design $GF(2^n)$ parallel multipliers, which is an important application field of these two algorithms. But we must emphasize that these two algorithms are $distinct$, and each of them have their own application fields [4].

TABLE II

Comparisons of asymptotic complexities for $n = b^t$

b	Algorithm	#AND	#XOR	Gate delay
2	KOA [26]	$n^{\log_2 3}$	$6n^{\log_2 3} - 8n + 2$	$(3\log_2 n - 1)T_X + T_A$
	Proposed	$n^{\log_2 3}$	$6n^{\log_2 3} - 8n + 2$	$(2\log_2 n)T_X + T_A$
	TMVP [5]	$n^{\log_2 3}$	$5.5n^{\log_2 3} - 6n + 0.5$	$(2\log_2 n)T_X + T_A$
3	KOA [26]	$n^{\log_3 6}$	$\frac{16}{3}n^{\log_3 6} - \frac{22}{3}n + 2$	$(4\log_3 n - 1)T_X + T_A$
	Proposed	$n^{\log_3 6}$	$\frac{16}{3}n^{\log_3 6} - \frac{22}{3}n + 2$	$(3\log_3 n)T_X + T_A$
	TMVP [5]	$n^{\log_3 6}$	$\frac{24}{5}n^{\log_3 6} - 5n + \frac{1}{5}$	$(3\log_3 n)T_X + T_A$

IV. CONCLUSIONS

We have proposed a fast VLSI implementation of the polynomial KOA in the ring $GF(2)[x]$. It eliminates overlaps in previous designs of KOA multipliers. The XOR gate delay of the proposed $GF(2)[x]$ multiplier is better than that of previous Karatsuba-Ofman multipliers. For example, it is reduced by about 33% and 25% for $n = 2^i$ and $n = 3^i$ $(i > 1)$, respectively.

REFERENCES

[1] A. Karatsuba and Y. Ofman, “Multiplication of Multidigit Numbers on Automata,” *Soviet Physics-Doklady (English translation)*, vol. 7, no. 7, pp. 595-596, 1963.

[2] G. Hanrot, P. Zimmermann “A long note on Mulders’ short product” *Journal of Symbolic Computation* vol.37 , pp.391-401, 2004

[3] S. Winograd, “Arithmetic Complexity of Computations”, SIAM, 1980.

[4] J. von zur Gathen and J. Gerhard, “Modern Computer Algebra,” Second ed., Cambridge Univ. Press, 2003.

[5] H. Fan and M. A. Hasan, “A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields,” *IEEE Transactions on Computers*, vol. 56, no. 2, pp. 224-233, Feb. 2007.

[6] H. Fan and Y. Dai, “ Fast bit parallel $GF(2^n)$ Multiplier for All Trinomials,” *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 485-490, Apr. 2005.

[7] M. A. Hasan and V. K. Bhargava, “Division and Bit-serial Multiplication over $GF(q^m)$,” *IEE Proceedings-E*, vol. 139, no. 3, pp. 230-236, May 1992.

[8] M. A. Hasan and V. K. Bhargava, “Architecture for Low Complexity Rate-Adaptive Reed-Solomon Encoder,” *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 938-942, July 1995.

[9] 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, pp. 856-861, July 1996.

[10] C. Paar, P. Fleischmann, and P. Roelse, “ Efficient Multiplier schemes for Galois Fields $GF(2^{4n})$, ” *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 162-170, Feb. 1998.

[11] 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.

[12] M. Jung, F. Madlener, M. Ernst, and S. Huss, “A Reconfigurable Coprocessor for Finite Field Multiplication in $GF(2^n)$,” *Proc. IEEE Workshop Heterogeneous reconfigurable Systems on Chip*, 2002.

[13] M. Ernst, M. Jung, F. Madlener, S. Huss, and R. Blumel, “A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over $GF(2^n)$,” *Proc. Cryptographic Hardware and Embedded Systems (CHES 2002)*, LNCS 2523, pp. 381-399, 2003.

[14] C. Grabbe, M. Bednara, J. Shokrollahi, J. Teich and J. von zur Gathen, “ FPGA Designs of parallel hign performance $GF(2^{233})$ Multipliers,” *Proc. Int’l Symposium on Circuits and Systems (ISCAS 2003)*, vol. II, pp. 268-271, 2003.

[15] A. Weimerskirch and C. Paar, “Generalizations of the Karatsuba Algorithm for Efficient Implementations,” 2003, $http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/kaweb.pdf$.

[16] F. Rodríguez-Henríquez and Ç. K. Koç, “On fully parallel Karatsuba multipliers for $GF(2^m)$,” *Proc. Int’l Conf. Computer Science and Technology (CST 2003)*, pp. 405-410, 2003.

[17] S. S. Erdem and Ç. K. Koç, “A Less Recursive Variant of Karatsuba-Ofman Algorithm for Multiplying Operands of Size a Power of Two,” *Proc. 16th IEEE Symposium on Computer Arithmetic (Arith-16 2003)*, pp. 28-35, 2003.

[18] A. E. Cohen and K. K. Parhi, “Implementation of scalable elliptic curve cryptosystem crypto-accelerators for $GF(2^m)$,” *Proc. 13th Asilomar Conf. on Signals, Systems and Computers*, vol. 1, pp. 471 - 477, Nov. 2004.

[19] B. Sunar, “ A Generalized Method for Constructing Subquadratic Complexity $GF(2^k)$ Multipliers,” *IEEE Transactions on Computers*, vol. 53, no. 9, pp. 1097-1105, Sept. 2004.

[20] P. L. Montgomery, “ Five, Six, and Seven-Term Karatsuba-Like Formulae,” *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 362-369, Mar. 2005.

[21] H. Fan and M. A. Hasan, “Comments on “five, Six, and Seven-Term Karatsuba-Like Formulae”,” *IEEE Transactions on Computers*, vol. 56, no. 5, pp. 716-717, May 2007.

[22] N. S. Chang, C. H. Kim, Y. H. Park, and J. Lim, “A Non-Redundant and Efficient Architecture for Karatsuba-Ofman Algorithm,” *Proc. 8th International Conf. on Information Security (ISC 2005)*, LNCS 3650, pp. 288-299, 2005.

[23] Z. Dyka and P. Langendoerfer, “Area efficient hardware implementation of elliptic curve cryptography by iteratively applying karatsuba’s method,” *Proc. Conf. on Design, Automation and Test in Europe 2005*, pp. 70-75, 2005.

[24] K. Y. Chang, D. Hong, and H. S. Cho, “Low Complexity Bit-Parallel Multiplier for $GF(2^m)$ Defined by All-One Polynomials Using Redundant Representation” *IEEE Transactions on Computers*, vol. 54, no. 12, pp. 1628-1630, Dec. 2005.

[25] L. S. Cheng, A. Miri and T. H. Yeap, “Improved FPGA Implementations of Parallel Karatsuba Multiplication over $GF(2^n)$,” *Proc. 23rd Biennial Symposium on Communications*, 2006.

[26] J. von zur Gathen and J. Shokrollahi, “ Efficient FPGA-based Karatsuba Multipliers for Polynomials over F_2,” *Proc. 12th Workshop on Selected Areas in Cryptography (SAC 2005)*, LNCS 3897 pp.359-369, 2006.

[27] S. Peter and P. Langendorfer, “An Efficient Polynomial Multiplier in $GF(2^m)$ and its Application to ECC Designs,” *Proc. Conf. on Design, Automation and Test in Europe 2007*, pp. 1253-1258, 2007.