T-CP/38/9//27548

AT^2-Optimal Galois Field Multiplier for VLSI

Martin Furer
Kurt Mehlhorn

Reprinted from
IEEE TRANSACTIONS ON COMPUTERS
Vol. 38, No. 9, September 1989

AT^2-Optimal Galois Field Multiplier for VLSI

MARTIN FÜRER AND KURT MEHLHORN

***Abstract*—Finite or Galois field arithmetic is central in many encoding and decoding procedures for error detecting and error correcting codes. In this paper, VLSI designs for Galois field multipliers are presented. For every prime p, these designs (for $GF(p^n)$ multiplication in standard representation) are asymptotically optimal with respect to area A and time T. In fact, the lower bound $AT^2 = \Omega(n^2)$ is matched for every computation time T in the range $[\Omega(\log n), O(\sqrt{n})]$. Analogous results hold for variable primes p too. The designs are based on the DFT on a structure similar to Fermat rings. For $p = 2$ the DFT uses 3^lth instead of 2^lth roots of unity.**

***Index Terms*—AT^2 optimal, DFT, Fermat ring, finite field multiplier, integer multiplier, standard representation of Galois fields, VLSI design.**

I. Introduction

For every prime p and $n \geq 1$, there is exactly one finite or Galois field $GF(p^n)$ with p^n elements. For $n = 1$ the finite field $GF(p)$ is just $Z_p = Z/pZ$ the field of integers modulo p. For $n > 1$ the standard representation of Galois fields is that by the polynomials of degree $\leq n - 1$ over Z_p modulo a polynomial of degree n which is irreducible over Z_p. Equivalently the elements of the vector space $GF(p^n)$ over Z_p are given by coordinates relative to a basis of the form $\{1, \alpha, \alpha^2, \cdots, \alpha^{n-1}\}$. Naturally, there are other representations (e.g., by powers of a primitive element) where multiplication is easy, but then addition is hard [19].

Galois field arithmetic is central to most error detecting and error correcting codes [4], [8], [9]. The most important practical case is $p = 2$. Circuits and chips have been proposed to do Galois field arithmetic (e.g., [18]). These designs are reasonable for the case $GF(2^4)$, but for n not much greater than 4, the corresponding generalizations of these designs get quite big, because the asymptotic complexity is far from being optimal. More efficient $GF(2^n)$ multipliers are possible for many exponents n (including $n = 4$) [6], when the irreducible polynomial is allowed to be nonprimitive.

In this paper, we exhibit an AT^2-optimal Galois field multiplier based on the AT^2-optimal integer multipliers [11], [13] for the synchronous model of VLSI [5], [17]. Although our design is asymptotically optimal, suboptimal variants of it are preferable for small n. We will not explore these variants in the present paper but rather refer the reader to [11] where practical (but suboptimal) integer multipliers are discussed. Their ideas carry over to Galois field multiplication.

Galois field multiplication is done in two steps. At first two polynomials (of degree $n-1$) over Z_p are multiplied, then the resulting polynomial is reduced modulo a fixed irreducible polynomial (of degree n). These two steps are described in Sections II and III. Multiplication of polynomials is done by discrete Fourier transform (DFT). Especially for $p = 2$, it is more involved for $Z_p[x]$ than for $Z[x]$. We conclude with an extension to the case of variable p, and state some open problems.

Manuscript received February 16, 1987; revised August 10, 1988. A preliminary version of this paper appeared in *VLSI Algorithms and Architecture*, Proceedings of the Aegean Workshop on Computing, Loutraki, Greece, July 1986, *Lecture Notes in Computer Science Vol. 227*, Berlin, Germany: Springer, pp. 217-225.

M. Fürer is with the Department of Computer Science, The Pennsylvania State University, University Park, PA 16802.

K. Mehlhorn is with the Department of Applied Mathematics and Computer Science, University of the Saarland, D-6600 Saarbrücken, West Germany.

IEEE Log Number 8927548.

II. Multiplying Two Polynomials Over Z_p.

Let $A(x)$, $B(x) \in Z_p[x]$ be two polynomials of degree $n-1$. We want to compute $C(x) = A(x)B(x)$ by making use of the DFT. Schönhage [15] suggested the following strategy.

We treat the case $p \neq 2$ first. Let $T \in [\log n, \sqrt{n}]$ be an even power of two and let k be a multiple of $T/2$ with $p \nmid K$ and $kT \geq 2n$. We will describe a design with execution time $O(T)$ and area $O(n^2/T^2)$. It is based on DFT of order T.

Let

$$A(x) = \sum_{i=0}^{T/2-1} A_i(x) x^{ki}$$

where the A_i's are $(k-1)$-degree polynomials in x.

Define

$$\bar{A}(y) = \sum_{i=0}^{T/2-1} A_i(x) y^i \in Z_p[x][y].$$

The polynomial $\bar{B}(y)$ is defined similarly. Let $\bar{C}(y) = \bar{A}(y)\bar{B}(y)$. Then $C(x) = \bar{C}(x^k)$ and the coefficients of $\bar{C}$ are polynomials of degree $2k-2$. Without loss of information, they can be interpreted as elements of

$$Z_p[x]/(x^{2k}+1) =: R.$$

We compute $\bar{C}(y)$ from $\bar{A}(y)$ and $\bar{B}(y)$ by DFT, pointwise multiplication and DFT^{-1} over the ring R.

Schönhage and Strassen [16] have introduced the DFT based on Fermat rings instead of fields. We follow the presentation in [2] replacing 2 by x.

Definition: An element ω of a commutative ring with unity is a *principal Tth root of unity* if

$$1)\ \omega \neq 1,\ 2)\ \omega^T = 1,\ \text{and}\ 3)\ \sum_{j=0}^{T-1} \omega^{jq} = 0 \qquad \text{for } 1 \leq q < T.$$

Hence, principal roots of unity in rings have the main properties of primitive roots of unity in fields.

Proposition: Let T be a positive power of 2, ω be a positive power of x, and p be an odd prime. Then T and ω have multiplicative inverses and ω is a principal Tth root of unity in $Z_p[x]/(\omega^{T/2}+1)$.

Proof: T has an inverse even in Z_p, because $p \nmid T$. Modulo $\omega^{T/2} + 1$, we have $\omega^T = (\omega^{T/2})^2 = (-1)^2 = 1$ implying $\omega\omega^{T-1} = 1$.

Let $T = 2^l$. Then

$$\sum_{j=0}^{T-1} \omega^{jq} = \prod_{i=0}^{l-1} (1+\omega^{2^i q}) \qquad \text{for all } q.$$

For $1 \leq q < T$ there is an $i \in \{0, \cdots, l-1\}$ such that $T | 2^{i+1}q$, and $2^{i+1}q/T$ is odd. This implies

$$1+\omega^{2^i q} = 1 + (-1)^{2^{i+1}q/T} = 0,$$

and

$$\sum_{j=0}^{T-1} \omega^{jq} = 0. \qquad \blacksquare$$

As a consequence of the proposition, the T-point discrete Fourier transform (with respect to ω) and its inverse exist in the ring $Z_p[x]/(\omega^{T/2}+1)$.

We choose $\omega^{T/2} = x^{2k}$, i.e., $\omega = x^{4k/T}$. The DFT requires addition of ring elements and multiplication by powers of ω. Since $R = Z_p[x]/(x^{2k}+1)$, additions are simple and take time $O(\log p)$. Also multiplications by powers of ω are almost cyclic shifts (note that $x^{2k} \equiv -1$) of the coefficients of the polynomials. From now on we proceed exactly as in [11], i.e., we compute the DFT on a $\sqrt{T} \times \sqrt{T}$ mesh of modules—each module holding an element of R (initially a coefficient of $\bar{A}$ or $\bar{B}$)—with $O(\sqrt{T})$ exchange steps of elements between adjacent modules and $O(\log T)$ arithmetic steps. Each exchange step takes time $O(\sqrt{T})$, and each arithmetic step can be done in time $O(T/\log T)$. For the details we refer the reader to [11]. The pointwise multiplications are also in complete analogy to [11].

We now turn to the more difficult case $p = 2$, using again an idea of Schönhage [15].

Unfortunately, there is no inverse of the usual 2^l-point DFT, because $T = 2^l$ has no inverse in $Z_p[x]/(x^{2k}+1)$. Therefore, a $T = 3^l$-point DFT is employed. Hereby the "Fermat polynomial" $x^{2k}+1$ is replaced by $x^{2k}+x^k+1$ with k a multiple of $T/3$. Choosing $\omega = x^{3k/T}$ implies.

$$\omega^{2T/3} + \omega^{T/3} + 1 = 0 \ \text{in}\ Z_2[x]/(x^{2k}+x^k+1).$$

We know that DFT's work in fields, and we argue why they work also in the ring

$$Z_2[x]/(x^{2k}+x^k+1).$$

In $Z_2[x]$, the gcd of $x^{3k}-1$ and its derivative $3kx^{3k-1}$ is 1 for k odd. Therefore, $x^{3k}-1$ and its divisor $x^{2k}+x^k+1$ have no multiple irreducible factors. By the Chinese Remainder Theorem $Z_2[x]/(x^{2k}+x^k+1)$ is isomorphic to the direct product of the fields $Z_2[x]/p_j(x)$, where the $p_j(x)$ are the irreducible factors of $x^{2k}+x^k+1$. All we have to make sure is that the isomorphism maps ω into a primitive Tth root of unity in each direct factor. But this is the case because

$$\omega^{2T/3} + \omega^{T/3} + 1 = 0 \ \text{modulo all}\ p_j(x)$$

(ω is a root of the Tth cyclotomic polynomial) implies

$$\omega^T = 1 \ \text{and}\ \omega^{T/3} \neq 1 \ \text{modulo all}\ p_j(x).$$

Hence, the DFT and its inverse work fine and can be computed fast by the following algorithm (for T a power of 3, k the smallest odd multiple of $T/3$ not less than $2n/T$, $\omega = x^{3k/T}$ and arithmetic modulo $x^{2k}+x^k+1$).

```
DFT($\omega$; $a_0, \cdots, a_{T-1}$):
if $T = 1$ then DFT($\omega$; $a_0$) := $a_0$ else
$t := T/3$
for $j := 0$ to $t-1$ do
    $b_j := a_j + a_{j+t} + a_{j+2t}$
    $c_j := (a_j + a_{j+t}\omega^t + a_{j+2t}\omega^{2t})\omega^j$
    $d_j := (a_j + a_{j+t}\omega^{2t} + a_{j+2t}\omega^t)\omega^{2j}$
DFT($\omega$; $a_0, \cdots, a_{T-1}$) := (DFT($\omega^3$; $b_0, \cdots, b_{t-1}$), DFT($\omega^3$; $c_0, \cdots, c_{t-1}$), DFT($\omega^3$; $d_0, \cdots, d_{t-1}$)).
```

The arithmetic modulo $x^{2k}+x^k+1$ can be replaced by arithmetic modulo

$$x^{3k}-1 = (x^{2k}+x^k+1)(x^k-1)$$

with only the final results reduced further. Then multiplication by powers of ω is just a cyclic shift.

We now proceed exactly as the case $p \neq 2$ except for one difference. The computation of the DFT of order T on the $\sqrt{T} \times \sqrt{T}$ mesh consists of three phases. (cf. [11, Appendix 2])

1) a DFT of order $\sqrt{T}$ on all columns,
2) local multiplications by powers of ω,
3) DFT of order $\sqrt{T}$ on all rows.

Steps 1 and 3 are realized by computing the DFT on a linear array of processors of length $O(\sqrt{T})$ in time $O(\sqrt{T})$. Note, however, that $\sqrt{T}$ was a power of two in the case $p \neq 2$ and is a *power of three* now. We need the following lemma with $m = \sqrt{T}$.

Lemma: Let $m = 3^l$. Then the mth order DFT of an m-vector can be computed on a linear array of m processing units in $O(m)$ exchange steps and $O(\log m)$ arithmetic steps.

Proof: We refer to the recursive definition of the DFT given above; replace T by m. We start with element a_i stored in the ith unit. We can clearly transport in time $O(m)$ the elements a_i, $a_{i+m/3}$, $a_{i+2m/3}$ to processing units i, $i + m/3$, $i + 2m/3$ in parallel for all i, $0 \leq i < m/3$. We then compute b_j, c_j, d_j with $O(1)$ arithmetic steps and complete the DFT by computing in parallel three DFT's of one third the size. The time bound follows. ■

III. Reduction Modulo an Irreducible Polynomial

The straightforward method of reducing a $2n-1$ degree polynomial modulo an extremely simple polynomial like $x^n - 1$ causes $\Omega(n)$ wires to cross $\Omega(n)$ others and has thus an area $\Omega(n^2)$. We need to perform much better for more complicated polynomials.

In order to reduce

$$p(x) = \sum_{j=0}^{2n-1} a_j x^j \text{ modulo } q(x) = \sum_{j=0}^{n} b_j x^j,$$

we let

$$(q(x))^{-1} = \sum_{j=-\infty}^{-n} c_j x_j$$

$$p(x)(q(x))^{-1} = \sum_{j=-\infty}^{n-1} d_j x_j,$$

and we compute the first n terms

$$c_{-n}, c_{-n-1}, \cdots, c_{-2n+1} \text{ and } d_{n-1}, \cdots, d_0$$

of these formal Laurent series. We "round" $p(x)(q(x))^{-1}$ to

$$s(x) = \sum_{j=0}^{n-1} d_j x^j$$

and obtain the remainder

$$r(x) = p(x) - q(x)s(x)$$

which is a polynomial of degree $n-1$ congruent to $p(x)$ modulo $q(x)$.

Actually, the first n coefficients $c_{-n}, \cdots, c_{-2n+1}$ of $(q(x))^{-1}$ can be preprocessed. And together with the coefficients $b_0, \cdots, b_n$ of $q(x)$, they are built into the hardware of the chip. In fact, we can do even better by storing directly their Fourier transforms. In any case, we have shown the following.

Theorem 1: For every fixed prime p there is a class of VLSI designs doing Galois field multiplication. The design with parameters $n \in N$ and $T \in [\log n, \sqrt{n}]$ does $GF(p^n)$ multiplication in standard notation (modulo a built-in irreducible polynomial of degree n over Z_p) in time $O(T)$ and has area A with $AT^2 = O(n^2)$.

This result is optimal, because the lower bound argument for integer multiplication [1], [5] extends to Galois fields.

IV. Extensions and Open Problems

We have so far investigated the complexity of $GF(p^n)$ multiplication for fixed p and variable n. We now deal with the case of variable n and p, i.e., we investigate the complexity of $GF(p^n)$ multipliers as a function of n and p. Now assume that elements of $GF(p^n)$ are given in standard form with their coefficients (elements of Z_p) given as binary integers.

Theorem 2: Every class of $GF(p^n)$ multipliers needs $AT^2 = \Omega((n \log p)^2)$, and for every $T \in [\log n + \log\log p, \sqrt{n} \log p]$ there are $GF(p^n)$ multipliers with area A, time $O(T)$, and $AT^2 = O((n \log p)^2)$.

The lower bound argument extends to this case, because in addition to the shifting of coefficients, shifts within the (binary) coefficients can easily be produced.

To obtain the upper bound, we use AT^2-optimal integer multiplication and reduction modulo p employing a precomputed binary approximation of p^{-1}.

The similarity between the polynomial arithmetic and the integer arithmetic used in the VLSI design of Theorem 2 suggests that actually the same pieces of hardware could serve either purpose. Therefore, the same chip can be used for multiplication in any Galois field $GF(p^n)$ with $p^n \leq 2^m$. Then a prime p and an irreducible polynomial $q(x)$ over Z_p are additional inputs. Divisions by p and $q(x)$ have to be calculated on the chip; hence, the range of T is a little bit restricted [3], [10], [12], [14].

Theorem 3: For every $T \in [(\log m)^{1+\epsilon}, \sqrt{m}]$, there is a VLSI design doing $GF(p^n)$ multiplication in time $O(T)$ and area $A = O(m^2/T^2)$ on a single chip for any p and n with $p^n \leq 2^m$.

Open Problems:

1) What are the parallel and VLSI complexities (T and AT^2 measures) for computing

a) inverses in $GF(p^n)$?

b) inverses in Z_p?

c) the gcd of integers?

2) Which sophistications of our theoretical designs should be avoided in order to obtain the most efficient practical chips doing $GF(p^n)$ multiplication for reasonable values of p and n?

Remark: $\alpha \in GF(p^n)^\times$ implies $\alpha^{p^n-1} = 1$, and therefore $\alpha^{p^n-2} = \alpha^{-1}$. This yields only an $AT^2 = O(n^4(\log p)^4)$ algorithm for 1a), while the lower bound is $\Omega(n^2(\log p)^2)$. Already 1b) seems to be difficult, because inverses in Z_p are computed by Euclid's algorithm, for which no efficient parallel version is known (see [7]). But we cannot exclude that 1b) might be easier than 1c).

References

[1] H. Abelson and P. Andreae, "Information transfer and area-time tradeoffs for VLSI multiplication," *Commun. ACM*, vol. 23, pp. 20–22, Jan. 1980.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.

[3] P. W. Beame, S. A. Cook, and H. J. Hoover, "Log depth circuits for division and related problems," *SIAM J. Comput.*, vol. 15, pp. 994–1003, Nov. 1986.

[4] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.

[5] R. P. Brent and H. T. Kung, "The chip complexity of binary arithmetic," *J. ACM*, vol. 28, pp. 521–534, 1981.

[6] M. Fürer, in preparation, 1985.

[7] R. Kannan, G. Miller, and L. Rudolph, "Sublinear parallel algorithm for computing the greatest common divisor of two integers," *SIAM J. Comput.*, vol. 16, pp. 7–16, 1987.

[8] J. H. van Lint, *Coding Theory, Lecture Notes in Math., Vol. 201*, Berlin, Germany: Springer-Verlag, 1971.

[9] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1978.

[10] K. Mehlhorn, "AT^2-optimal VLSI integer division, and integer square rooting," *Integration*, vol. 2, pp. 163–167, 1984.

[11] K. Mehlhorn and F. P. Preparata, "Area-time optimal VLSI integer multiplier with minimum computation time," *Inform. Contr.*, vol. 58, pp. 137–156, 1983.

[12] ——, "Area-time optimal division for $T = \Omega((\log n)^{1+\epsilon})$," *Inform. Computat.*, vol. 72, pp. 270–282, 1987.

[13] F. P. Preparata and J. Vuillemin, "Area-time optimal VLSI networks for computing integer multiplication and discrete Fourier transform," in *Automata Languages and Programming, Lecture Notes in Computer Science, Vol. 115*. Berlin, Germany: Springer-Verlag, 1981, pp. 29–40.

[14] J. Reif, "Logarithmic depth circuits for algebraic functions," in *Proc. 24th IEEE Symp. Foundations Comput. Sci.*, Nov. 1983, pp. 138–143.

[15] A. Schönhage, "Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2," *Acta Informatica*, vol. 7, pp. 395–398, 1976.

[16] A. Schönhage and V. Strassen, "Schnelle Multiplikation grosser Zahlen," *Computing*, vol. 7, pp. 281–292, 1971.

[17] C. D. Thompson, "Area-time complexity for VLSI," in *Proc. 11th ACM Symp. Theory Comput.*, Apr./May 1979, pp. 81–88.

[18] C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed, "VLSI architectures for computing multiplications and inverses in $GF(2^m)$," *IEEE Trans. Comput.*, vol. C-34, pp. 709–717, Aug. 1985.

[19] A. A. Yao, "The entropic limitations on VLSI computations," in *Proc. 13th ACM Symp. Theory Comput.*, May 1981, pp. 308–311.