Fast Montgomery-like Square Root Computation for All Trinomials

Yin Li and Yu Zhang

We introduce a new type of Montgomery-like square root formulae in $GF(2^m)$ defined by an arbitrary irreducible trinomial, which is more efficient compared with classic square root operation. By choosing proper Montgomery factor for different kind of trinomials, the space and time complexities of such square root computations match or outperform the best results. A practical application of the Montgomery-like square root in inversion computation is also presented.

Introduction: Field square root computation is an important building block in the design of some elliptic curve primitives [1, 2] and a parallel multiplicative inversion algorithm [3]. Consider a binary extension field generated with an irreducible polynomial $GF(2^m) \cong \mathbb{F}_2[x]/(f(x))$ where $f(x)$ is irreducible over $\mathbb{F}_2$. Let A be an arbitrary element in the field $GF(2^m)$. The field square root operation of A, denoted as $\sqrt{A}$ or $A^{1/2}$, is to find $D \in GF(2^m)$ such that $D^2 = A$. When using normal base representation, square root computation is simply a circular operation. Hence, most previous works for square root computation focused on other representations.

Until now, the square root computation can be obtained using Fermat Little Theorem or pre-computation of $x^{1/2}$ [4]. In [5], Rodríguez-Henríquez et al. proposed an alternative method using the inversion of the multiplicative matrix, which is constructed for squaring operation. Their approach theoretically can be applied to any type of generating polynomials. When the generating polynomial $f(x)$ is a trinomial, the worst case of their algorithm costs about $\frac{5m}{2}$ XOR gates with the delay of $3T_X$. Compared with squaring operation for trinomials [6], the squaring root computation is a little more complicated.

In this contribution, we consider a new type of square root by adding a specific Montgomery-like factor. Explicit formulae of such operations are presented in $GF(2^m)$ defined by an irreducible trinomial $f(x) = x^m + x^k + 1$. We show these formulae are very simple and have smaller space and time complexities in parallel implementation. Then, we describe the application of this new type of square root in the domain of field inversion computation.

Montgomery-like Square Root: Let $A = \sum_{i=0}^{m-1} a_i x^i$ be an arbitrary element of $GF(2^m)$ using polynomial basis representation. To compute the square root of A, we first partition A into two parts according the degree parity of its intermediate, i.e.,

$$A = A_{even}^2 + xA_{odd}^2,$$

where $A_{even} = \sum_{i=0}^{(m-1)/2} a_{2i}x^i$ and $A_{odd} = \sum_{i=0}^{(m-3)/2} a_{2i+1}x^i$, if m is odd, or $A_{even} = \sum_{i=0}^{m/2-1} a_{2i}x^i$ and $A_{odd} = \sum_{i=0}^{m/2-1} a_{2i+1}x^i$, if m is even. Therefore, the square root of A is given by

$$A^{\frac{1}{2}} = A_{even} + x^{\frac{1}{2}} A_{odd}. \tag{1}$$

It is clear that the computation of $x^{1/2}$ is crucial to the square root computation. Actually, the element $x^{1/2}$ is a constant that can be pre-computed. Thus, the square root computation can be implemented as performing a field multiplication between A_{odd} and $x^{1/2}$ and then adding with A_{even}. However, this method requires a constant multiplication, which is still an expensive operation. Therefore, we introduce a Montgomery-like square root to simplify the calculation of $x^{1/2}$ and avoid previous complex operation by choosing proper a factor. We first give the definition of this new square root.

Definition 1: Let A be an arbitrary element and ω be a fixed element of $GF(2^m)$, respectively. Then the Montgomery-like square root of A is defined as $A^{1/2} \cdot \omega$, while ω is named as the Montgomery-like factor.

The selection of the factor ω depends on the computation of $x^{1/2}$. Note that the generating polynomial here is $f(x) = x^m + x^k + 1$. We have $x = x^{m+1} + x^{k+1}$. Clearly, the formula of $x^{1/2}$ varies according to the parities of m and k. Four cases are considered:

Case 1: m even, k odd. In this case, we know that $2 \mid m$, but $2 \nmid k$. Let $n = m/2$, then we have

$$\begin{aligned} & x^{\frac{1}{2}} = x^n \cdot x^{\frac{1}{2}} + x^{\frac{k+1}{2}} \\ \Rightarrow \quad & x^{\frac{k+1}{2}} = (1 + x^n) \cdot x^{\frac{1}{2}}, \\ \Rightarrow \quad & x^{-\frac{1}{2}} = (1 + x^n) \cdot x^{-\frac{k+1}{2}}, \\ \Rightarrow \quad & x^{\frac{1}{2}} = (1 + x^n) \cdot x^{-\frac{k-1}{2}}. \end{aligned}$$

Plug the above expression to (1), one can check that

$$A^{\frac{1}{2}} = A_{even} + x^{\frac{1}{2}} A_{odd} = A_{even} + (A_{odd} + A_{odd}x^n) \cdot x^{-\frac{k-1}{2}}.$$

Note that A_{odd} consists of n terms, thus, there is no overlap between A_{odd} and $A_{odd}x^n$. Let ω be $x^{(k-1)/2}$, the Montgomery-like square root of this case is

$$A^{\frac{1}{2}} \cdot \omega = A^{\frac{1}{2}} \cdot x^{\frac{k-1}{2}} = A_{even}x^{\frac{k-1}{2}} + (A_{odd} + A_{odd}x^n).$$

Since the degree of A_{even} and A_{odd} are $n-1$, $\deg(A_{even}x^{(k-1)/2}) = n - 1 + (k-1)/2 \le n + (m-1)/2 = m - 1$ and $\deg(A_{odd} + A_{odd}x^n) = n + n - 1 = m - 1$. Therefore, no further reduction is needed in above expression. At this time, let $D = \sum_{i=0}^{m-1} d_i x^i = A^{1/2} \cdot x^{(k-1)/2}$. Then,

$$d_i = \begin{cases} a_{2i+1}, & 0 \le i \le \frac{k-3}{2}, \\ a_{2i-k+1} + a_{2i+1}, & \frac{k-1}{2} \le i \le n-1, \\ a_{2i-k+1} + a_{2i+1-m}, & n \le i \le n-1+\frac{k-1}{2}, \\ a_{2i+1-m}, & n + \frac{k-1}{2} \le i \le m-1, \end{cases} \tag{2}$$

where $0 < k \le m - 1$.

Case 2: m even, $k = m/2$ odd. This case is actually a special case of Case 1. It is easy to check that $x^{1/2} = (1 + x^k) \cdot x^{-(k-1)/2}$. Also notice that $m = 2k$ and $x^m = x^k + 1$. Hence, the previous expression can be rewritten as $x^{1/2} = x^{2k-(k-1)/2}$. So,

$$A^{\frac{1}{2}} = A_{even} + A_{odd}x^{k+\frac{k+1}{2}}.$$

It is easy to check that there are at most $(k+1)/2$ terms of which degrees are out of the range $[0, m-1]$ and the reduction is very easy. Thus, let $\omega = 1$ and $D = \sum_{i=0}^{m-1} d_i x^i = A^{1/2}$, we have

$$d_i = \begin{cases} a_{2i} + a_{2i+k}, & 0 \le i \le \frac{k-1}{2}, \\ a_{2i}, & \frac{k+1}{2} \le i \le k-1 \\ a_{(2i+k) \bmod m}, & k \le i \le m-1. \end{cases} \tag{3}$$

This formula simply coincide with the result present in [5].

Case 3: both m and k are odd. Let $n = \frac{m+1}{2}$, then

$$x^{\frac{1}{2}} = x^n + x^{\frac{k+1}{2}}.$$

One can check that the above formula is very easy and the corresponding square root computation needs no ω to simplify its computation. Analogous with Case 2, let $\omega = 1$ and $D = \sum_{i=0}^{m-1} d_i x^i = A^{\frac{1}{2}}$, then

$$d_i = \begin{cases} a_{2i}, & 0 \le i \le \frac{k-1}{2}, \\ a_{2i} + a_{2i-k}, & \frac{k+1}{2} \le i \le \frac{m-1}{2}, \\ a_{2i-k}, & k \le i \le m-1. \end{cases} \tag{4}$$

Case 4: m odd, k even. Let $n = \frac{m-1}{2}$, then

$$\begin{aligned} & x^{\frac{1}{2}} = x^{n+1} + x^{\frac{k}{2}} \cdot x^{\frac{1}{2}} \\ \Rightarrow \quad & x^{n+1} = (1 + x^{\frac{k}{2}}) \cdot x^{\frac{1}{2}}, \\ \Rightarrow \quad & x^{-\frac{1}{2}} = (1 + x^{\frac{k}{2}}) \cdot x^{-(n+1)}, \\ \Rightarrow \quad & x^{\frac{1}{2}} = (1 + x^{\frac{k}{2}}) \cdot x^{-n}. \end{aligned}$$

Plug the above expression to (1), then

$$\begin{aligned} A^{\frac{1}{2}} &= A_{even} + x^{\frac{1}{2}} A_{odd} \\ &= A_{even} + (A_{odd} + A_{odd}x^{\frac{k}{2}}) \cdot x^{-n} \end{aligned}$$

Let ω be x^n, then the Montgomery-like square root is

$$A^{\frac{1}{2}} \cdot \omega = A^{\frac{1}{2}} x^n = A_{even}x^n + A_{odd} + A_{odd}x^{\frac{k}{2}}.$$

Please note that A_{odd} consists of n terms and A_{even} consists of $n+1$ terms. Thus, $A_{even}x^n$ and A_{odd} are non-overlapping with each other. Also notice that $\deg A_{odd} = n - 1$, $\deg A_{even} = n$ and $k < m$, so $\deg A_{odd}x^{\frac{k}{2}} = n - 1 + k/2 \le n + (m-1)/2 = m - 1$,

$\deg A_{even}x^n = 2n = m-1$. Therefore, no further reduction is required for $A^{1/2}x^n$. Let $D = \sum_{i=0}^{m-1} d_i x^i = A^{\frac{1}{2}} \cdot x^n$, then

$$d_i = \begin{cases} a_{2i+1}, & 0 \le i \le \frac{k}{2} - 1, \\ a_{2i+1} + a_{2i+1-k}, & \frac{k}{2} \le i \le n-1, \\ a_{2i+1-k} + a_{2i-m+1}, & n \le i \le n + \frac{k}{2} - 1, \\ a_{2i-m+1}, & n + \frac{k}{2} \le i \le m-1. \end{cases} \quad (5)$$

where $0 < k < m-1$. If $k = m-1$, then

$$d_i = \begin{cases} a_{2i+1}, & 0 \le i \le n-1, \\ a_{2i-m+1} + a_{2i-m+2}, & n \le i \le m-2, \\ a_{2i-m+1}, & i = m-1. \end{cases} \quad (6)$$

According to expression (2) to (6), we summarize the space and time complexity of the Montgomery-like square root computation in Table 1. It is clear that, by choosing proper factor ω, the Montgomery-like square root costs are at most $m/2$ (or $(m-1)/2$) XOR gates with only 1 T_X gate delay, which outperforms or matches the best square root computation algorithms [5].

Table 1: Comparison between Montgomery-like and ordinary square root

	Montgomery-like			ordinary	
Case	ω	$\#XOR$	Delay	$\#XOR$	Delay
1. m even, k odd	$x^{\frac{k-1}{2}}$	$\frac{m}{2}$	1 T_X	$\frac{m+k-1}{2}$	2 T_X
2. m even, $k=\frac{m}{2}$ odd	1	$\frac{m+2}{4}$	1 T_X	$\frac{m+2}{4}$	1 T_X
3. m, k odd	1	$\frac{m-1}{2}$	1 T_X	$\frac{m-1}{2}$	1 T_X
4. m odd, k even	$x^{\frac{m-1}{2}}$	$\frac{m-1}{2}$	1 T_X	$\frac{m+k-1}{2}$	3 T_X

Example: Consider a finite field $GF(2^9)$ defined by $x^9 + x^4 + 1$. According to previous description, since $x^9 + x^4 + 1$ satisfies case 4, we have $\omega = x^{(9-1)/2} = x^4$. Given an arbitrary element $A = \sum_{i=0}^{8} a_i x^i$, the Montgomery-like square root of A is $A^{1/2} \cdot x^4 = \sum_{i=0}^{8} d_i x^i$, where

$$\begin{array}{lll} d_0 = a_1, & d_3 = a_7 + a_3, & d_6 = a_4, \\ d_1 = a_3, & d_4 = a_0 + a_5, & d_7 = a_6, \\ d_2 = a_5 + a_1, & d_5 = a_2 + a_7, & d_8 = a_8. \end{array}$$

The computation of above coefficients d_i requires 4 XOR gates and 1 T_X in parallel.

Application: In [5], the authors proposed an efficient algorithm for exponentiation using square root over $GF(2^m)$. Their algorithm mainly utilized the equation

$$A^{2^{m-i}} = A^{2^{-i}}, i = 1, 2, \cdots, m-1,$$

where $A \in GF(2^m)$ is an arbitrary nonzero element. The above equation can be easily proved using Fermat Little Theorem. Hence, the squaring operation based exponentiation can also be implemented using square root operation. Let $e = (e_{m-1}, \cdots, e_1, e_0)_2$ be a m-bit nonzero integer. Then, the exponentiation A^e can be written as

$$A^e = \prod_{i=0}^{m-1} A^{2^i e_i} = \prod_{i=0}^{m-1} A^{2^{-(m-i)} e_i} = \prod_{i=1}^{m-1} A^{2^{-(m-i)} e_i} \cdot A^{e_0}.$$

When $e = 2^m - 2 = (1, 1, \cdots, 1, 0)_2$, the exponentiation A^e compute the inversion of A. By substituting the ordinary square root with Montgomery-like square root, we give a new inversion algorithm.

Algorithm 1 Inversion based on Montgomery-like Square root

Input: $A \in GF(2^m)$, $f(x)$, $e = (1, 1, \cdots, 1, 0)_2$
Output: $B = A^{-1} = A^e \bmod f(x)$

1: $B = A$;
2: **for** $i = 1$ to $m-2$ **do**
3: $\quad B = B^{\frac{1}{2}} \cdot \omega$;
4: $\quad B = B \cdot A \bmod f(x)$;
5: **end for**
6: $B = B^{\frac{1}{2}} \cdot \omega$;
7: $B = B \cdot \omega^*$;
8: **return** B;

Since we use the Montgomery-like square root computation to substitute the original one, ω^* in step 8 is a compensatory parameter that used to correct the final exponentiation (for case 1 and 4 of Table 1). According to step 2 and 3 of Algorithm 1, we totally perform $m-1$ Montgomery-like square root computation. Therefore, to make the final result correct, we have

$$\begin{aligned} (\omega^*)^{-1} &= \omega \cdot \omega^{\frac{1}{2}} \cdot \omega^{\frac{1}{4}} \cdots \omega^{\frac{1}{2^{m-2}}} \\ &= \omega^{1+2^{-1}+2^{-2}+\cdots+2^{-(m-2)}} = \omega^{2-2^{-(m-2)}}. \end{aligned}$$

Please notice that $\omega^{2^{-(m-2)}}$ actually equals $\omega^{2^{m-(m-2)}} = \omega^{2^2}$. So

$$\omega^* = (\omega^{2-2^2})^{-1} = \omega^{2^2-2} = \omega^2.$$

The constant multiplication $B \cdot \omega^*$ can be performed using Mastrovito approach. According to Table 1, it is known that $\omega = x^{(k-1)/2}$ (or $x^{(m-1)/2}$, then $\omega^* = x^{k-1}$ (or x^{m-1}). Therefore, the Mastrovito matrix constructed from ω^* is actually sparse. Only a few XOR gates are required by corresponding matrix-vector multiplication [1], which leads to no more than $\lceil \log_2 m \rceil$ XOR gate delay. If m, k satisfy case 1 and case 4, our algorithm can save at least 1 T_X for each square root computation and finally has at least $m - \lceil \log_2 m \rceil - 1 T_X$ gain. If m, k satisfy case 2 and 3, our algorithm is the same as the classic one.

In practical application, given a fixed irreducible trinomial, the space and complexities of $B \cdot \omega^*$ can be even lower. For example, consider irreducible trinomial $x^{233} + x^{74} + 1$ proposed by NIST [7], we have $\omega^* = x^{232}$. The constant multiplication $B \cdot \omega^*$ can be obtained using 205 XOR gates with $2T_X$ delay in parallel. Therefore, the inversion algorithm based on Montgomery-like square root can save $m - 2 = 231$ T_X compared with the classic inversion algorithm that use square root computation.

Conclusion: In this paper, we have proposed a new type Montgomery-like square root computation algorithm. By choosing a proper factor, the proposed scheme has only one T_X delay and its space complexity at least as good as the best results. We also show an important application of this type of square root, which improves the computation efficiency compared with classic algorithm using ordinary square root.

Acknowledgment: This work has been supported by the National Natural Science Foundation of China (Grant No. 61402393, 61601396)

Yin Li and Yu Zhang (*Department of Computer Science and Technology, Xinyang Normal University, Henan, P.R.China*)

E-mail: yunfeiyangli@gmail.com

References

1 Hankerson, D., Menezes, A. and Vanstone, S.: 'Guide to Elliptic Cryptography', (Springer-Verlag, 2004)
2 Schroeppel, R., Beaver, C., Gonzales, R., Miller, R. and Draelos, T.: 'A Low-Power Design for an Elliptic Curve Digital Signature Chip', *Proc. Fourth Int'l Workshop Cryptographic Hardware and Embedded Systems*, 2002, pp. 366-380
3 Rodríguez-Henríquez, F., Morales-Luna, G., Saqib, N. and CruzCortés, N.: 'Parallel Itoh-Tsujii Multiplicative Inversion Algorithm for a Special Class of Trinomials', *Reconfigurable Computing: Architectures, Tools and Applications, LNCS 4419*, 2007, pp. 226-237
4 K. Fong, D. Hankerson, J. López, and A. Menezes, 'Field Inversion and Point Halving Revisited', *IEEE Trans. Computers*, 2004, **53**, (8), pp. 1047-1059
5 Rodríguez-Henríquez, F., Morales-Luna, G. and López, J.: 'Low-Complexity Bit-Parallel Square Root Computation over $GF(2^m)$ for All Trinomials', *IEEE Transactions on Computers*, 2008, **57**, (4), pp. 1-9
6 Wu, H.: 'Montgomery multiplier and squarer for a class of finite fields', *IEEE Transactions on Computers*, 2002, **51**, (5), pp. 521-529
7 'Recommended Elliptic Curves for Federal Government Use', special publication, Nat'l Inst. Standards and Technology, http:// csrc.nist.gov/csrc/fedstandards.html, July 1999.

[1] Since the Mastrovito multiplicative matrix is fixed, no AND gate is needed.