A Technique to Speed up the Modular Multiplicative Inversion over GF(P) Applicable to Elliptic Curve Cryptography

V. Sridhar,PhD.
Professor
Department of Electronics & Communication
PET Research foundation, PESCE, Mandya

Anil Kumar M .N
Research Scholar
PET Research foundation
PESCE, Mandya

ABSTRACT
This paper presents a technique to speed up the computation of inversion of NIST recommended elliptic curve with modulus p^{521}-1. The property of multiplicative inverse between pair of numbers over Meresenne's prime is used to reduce the number of iterations in the Binary Inversion Algorithm in GF(p). This increases the speed requirement for point operations applicable to Elliptic Curve Cryptography. This paper proposes an model of the architecture to achieve the above objective which uses parallelism in multiplicative inversion arithmetic block to speed up the computation.

Key words
Elliptic curve cryptography, Binary Inversion Algorithm, GF(p) arithmetic operators.

1. INTRODUCTION
The data security has become an important and urgent need for modern applications such as health care information, confidential communication, storage and financial services. The public key cryptosystem is the most effective for the secure data transaction and messaging. The challenge to implement the most popular public key cryptosystem, RSA is the rapidly growing key size. Elliptic Curve cryptography has been considered an alternative to RSA.

The application of Elliptic curves in public-key cryptography was proposed by Koblitz[1] and Miller[2] in 1985. Since then, enormous amount of work has been done on elliptic curve cryptography(ECC).The attractiveness of using elliptic curves is that similar level of security can be achieved with considerably shorter keys than in methods based on the difficulties of solving discrete logarithms over integers or integer factorizations. Therefore ECC has become final choice in smart cards, credit cards and mobile phones due to its strength to provide equivalent security compared to RSA. It is estimated that security level of 160 and 224 bits ECC cryptosystem is equivalent to the 1024 and 2048 bits RSA respectively[3,4]

The research on efficient algorithms and hardware accelerations have concentrated on efficient implementation of elliptic curve point multiplication which is the fundamental operation of all elliptic curve cryptosystems. The elliptic curve point multiplication is computed with point operations which, further are computed using finite field arithmetic. Although the point multiplication itself is hard to parallelize, it is possible to efficiently use parallelism [5],[6],[7], [8] in field arithmetic specifically to some of the NIST recommended elliptic curves.

To date, little research has been done on ECC hardware architectures over GF (p). [10] – [13] described some important contributions. Satoh and Takano [10], described an elliptic curve cryptographic processor which is able to operate in GF (p) and GF (2^m). Their work does not fully deal with modular inversion, which is the costliest operation even with projective coordinates. They have computed this using Fermat's Little Theorem which requires modular exponentiation. Orlando and Paar [11], proposed an architecture using high-radix Montgomery multiplier with adders and register used to compute field additions/subtractions and to perform comparisons. Again Fermat's Little Theorem is used to perform modular inversion. In [12] bitwise computations have been described.

Kendel Anayi, Hamad Alrimeigh, Daler Rakhmatov [9], described a flexible ECC processor for performing additions, subtractions, multiplications and inversions over prime finite fields GF(p). They have used Binary Inversion Algorithm to perform modular inversion. We have developed a technique to speed up the computation of this modular multiplicative inverse computation.

The main contributions of this work include the following.

i) A technique to speed up the computation of modular multiplicative inversion which uses Binary Inversion Algorithm. This technique can be used specifically to NIST recommended elliptic curve with modulus p^{521}-1.
ii) A model of the architecture for inversion is proposed which uses parallelism in multiplicative inversion arithmetic block to speed up the computation

The outline of this paper is as follows. In section 2, the back ground of Elliptic Curve Cryptography (ECC) is discussed. In section 3, how the property of multiplicative inverse between pair of numbers over a Mersenne's prime speed up the computation which uses Binary Inversion Algorithm is discussed. Section 4 deals with the proposed model of the architecture for inversion, section 5 deals with the results and finally section 6 concludes the work.

2. ECC BACKGROUND
2.1 Elliptic Curves over GF (P)
The elliptic curve arithmetic is defined over Galois field GF(p) where p is a prime number greater than 3 . All arithmetic operations are modulo p. The elliptic curve equation E over GF(p) is given by:$y^2 = x^3 + ax + b$; where p > 3, $4a^3 + 27b^2 \neq 0$, and x, y, a,b$\in$ GF(p). There is also a single element named the point at infinity or the zero point denoted

O, which serves as the additive identity. For any point P(x, y)∈ E , we have: P + O = P .

2.2 Point addition and Point Doubling

Additions in GF(p) are controlled by the following rules:

$O = -O$

$P(x, y) + O = P(x, y)$

$P(x, y) + P(x, -y) = O$

The addition of two different points on the elliptic curve is computed as shown below.

$P(x_1 , y_1) + P(x_2 , y_2) = P(x_3 , y_3)$; where $x_1 \neq x_2$

$\lambda = (y_2 – y_1)/(x_2 – x_1)$

$x_3 = \lambda^2 – x_1 – x_2$

$y_3 = \lambda(x_1 – x_3) – y_1$

The addition of a point to itself (point doubling) on the elliptic curve is computed as shown below

$P(x_1 , y_1) + P(x_1 , y_1) = P(x_3 , y_3)$;

$\lambda = (3(x_1)^2 + a) /(2y_1)$

$x_3 = \lambda^2 – 2x_1$

$y_3 = \lambda(x_1 – x_3) – y_1$

2.3 Point Multiplication

Scalar multiplication Q=k.P is the result of adding point P to itself (k-1) times

$$Q = k.P = P + P + ……. + P.$$
(k-1 Times)

The binary method is the simplest and oldest efficient method for point multiplication. It is based on the binary expansion of k. The corresponding algorithm shown in Fig.1.

INPUT: A point P and an integer k
OUTPUT: Q = k.P

1. Q←P
2. **For** j = L− 2… 1, 0
   2.1 Q ← 2 Q
   2.2 **IF** k_j = 1 **THEN** Q←Q + P
3. **RETURN** Q

Fig.1: Binary scalar multiplication algorithm adopted from [7]

If we assume that on average ‘n’ is the number of ones in k which is equal to n = L / 2, the binary method requires (L −1) point doublings and n point-additions where L denotes the number of bits of the scalar k. The point doubling and point addition require inversion. Hence the average number of inversion is (L-1) + n.

3. SPEEDING UP MULTIPLICATIVE INVERSE COMPUTATION

In this section, Binary Inversion Algorithm for inversion over GF(p), property of multiplicative inverse between pair of two numbers over Mersenne’s prime and how this property speed up the multiplicative inverse computation is discussed.

3.1. Binary Inversion Algorithm

The extended Euclidean algorithm uses the division operations to compute the inversion. The binary inversion algorithm replaces the divisions with cheaper shifts (divisions by 2) and subtractions.

The modular multiplicative inverse a^{-1} mod p of an integer a exists if and only if a and p are relatively prime, that is gcd (a,p) =1. One of the efficient modular inversion algorithms is Binary Inversion Algorithm shown below.

INPUT: Prime p and a ∈ [1, p-1]
OUTPUT: a^{-1} mod p

1. u=a, v=p, x1=1, x2=0
2. while (u !=1 and v!=1) **do**

while u is even do
   2.2.1 u = u/2
   2.2.2 if x1 is even then x1= x1/2 else
      x1= (x1+p)/2
   2.3 end while
   2.4 while v is even do
   2.5.1 v= v/2
   2.5.2 if x2 is even then x2= x2/2 else
      x2 = (x2+p)/2
   2.6 end while
   2.7 if u≥ v then u=u-v, x1=x1-x2
   2.8 else v=v-u, x2=x2-x1
3. end while
4.1 if u=1 then return x1 mod p
4.2 else return x2 mod p

Fig.2: Binary Inversion Algorithm in GF(p)

The step 2 of the algorithm runs iteratively. The steps from 2.1 to 2.3 and steps from 2.4 to 2.6 perform concurrently. A speed improvement can be obtained if the number of iterations (subtractions) of step2 is reduced. The binary modular inversion algorithm can be easily be modified to perform modular division b/a mod p by initializing x1 variable in step 1 by b instead of 1.In the subsequent sections iteration denote subtraction.

3.2 The property and the technique to reduce the number of iterations

If x is a number [1 ≤x ≤p-1] and p is a Mersenne’s prime then the multiplicative inverse of p-x(this is obtained by p XOR x) is the complement of multiplicative inverse of x. The table 1 illustrates this property.

The number of iterations of step2 to compute the inverse in the Binary Inverse Algorithm is different for x and p-x. The table 2 shows the number of iterations of step 2 of Binary Inversion Algorithm over Mersenne’s prime p31.

Table 1. The relation between multiplicative inverse of x and p-x over p_{31}

Number x in decimal	Binary of x	x^{-1} in decimal	x^{-1} in binary	Number p-x in decimal	Binary of p-x	$(p-x)^{-1}$ in decimal	$(p-x)^{-1}$ in binary
1	00001	1	00001	30	11110	30	11110
2	00010	16	10000	29	11101	15	01111
3	00011	21	10101	28	11100	10	01010
4	00100	8	01000	27	11011	23	10111
5	00101	25	11001	26	11010	6	00110
6	00110	26	11010	25	11001	5	00101
7	00111	9	01001	24	11000	22	10110
8	01000	4	00100	23	10111	27	11011
9	01001	7	00111	22	10110	24	11000
10	01010	28	11100	21	10101	3	00011
11	01011	17	10001	20	10100	14	01110
12	01100	13	01101	19	10011	18	10010
13	01101	12	01100	18	10010	19	10011
14	01110	20	10100	17	10001	11	01011
15	01111	29	11101	16	10000	2	00010

Table 2. Number of iterations (subtractions) of step2 to compute the inverse in the Binary Inverse Algorithm for number x and p-x over p_{31}.

Number x	Number of Iterations	Number p-x	Number of iterations	Difference in the number of iterations
1	0	30	7	7
2	2	29	3	1
3	7	28	10	3
4	3	27	4	1
5	7	26	7	0
6	8	25	9	1
7	8	24	10	2
8	4	23	5	1
9	5	22	9	4
10	8	21	8	0
11	8	20	9	1
12	9	19	9	0
13	6	18	6	0
14	9	17	7	2
15	6	16	5	1
Average difference in the number of iterations				24/15=1.6

If each of all possible x, p-x inputs are processed concurrently in the Binary inversion algorithm, it is found that 11 out of 15 combinations produces different number of iterations in step 2. Only 4 out of 15 possible combinations have same number of iterations. The difference in the number of iterations for x and p-x inputs ranges from 0 to 7.

In order to speed up the computation of multiplicative inversion, we have proposed parallel architectures namely, process1 and process2 which are discussed in the section 4. Both the processes are Binary Inversion Algorithm. The process 1 computes inversion of x while process 2 computes inversion of p-x concurrently. The computation time is different for process 1 and process 2 because of the difference in the number of iterations of step2. Hence either of the processes computes inversion faster than the other for 74% (with respect to the table 2) of all the possible concurrent inputs. If the output is taken from process 2, the result of the process is inverted else it is taken directly from the process 1. Hence a speed improvement is obtained by reducing the number of iterations with concurrent processes with additional two inversions (one inversion to obtain the input p-x from x and another to obtain inverted output from process2). The schematic of this technique is shown in figure 3.

4. PROPOSED MODEL OF ARCHITECTURE

The Binary Inversion Algorithm in GF(p) can be implemented by hardware. Our proposed model consists of parallel arithmetic units for the computation of inversion. One arithmetic unit computes the inversion of input x while other arithmetic unit computes the inversion of p-x input concurrently. The results shown in the section 5 indicates that this parallel architecture can be implemented to speed up the computation of inversion. The following section describes architecture of process1. For the process2 we instantiate the same architecture by replacing u,v,p,x1 and x2 by p-u,v,p,x1 and x2.

After loading u,v,p,x1 and x2 values into registers(step1 of Algorithm), the two comparators compare the values of u and v registers with 1(step2 of algorithm).If any one of

comparator output becomes true then the final result is made available in x1 or x2. The comparators output can be used to select either x1 or x2 through multiplexers.

There are two blocks, block1 and block2 which perform concurrently. Block1 performs steps 2.1, 2.2.1, 2.2.2, 2.3 and block2 performs steps 2.4, 2.5.1, 2.5.2, 2.6 respectively. The outputs of block1 and block2 are the updated values of u, x1 and v, x2 respectively.

The block3 performs the computation of step 2.7 and step 2.8. The updated values of u, x1, v, x2 are made available from block1 and block2 to block 3 if both the u and v values are odd. The updated values can be made available to block 3 by performing AND operation of the least significant bits of u and v and selecting through a multiplexer. The Figure 4 shows the proposed model of the architecture .

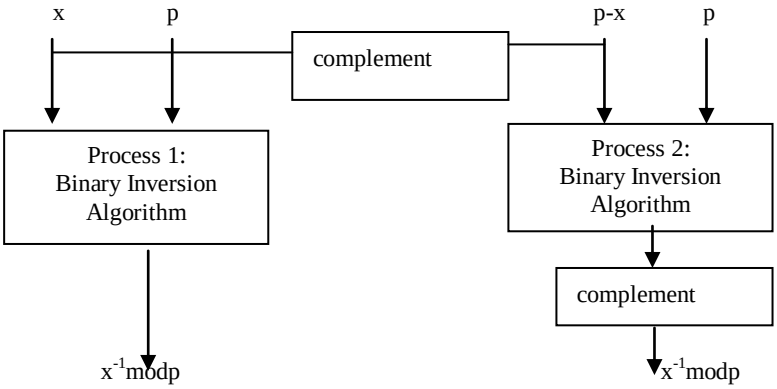

Fig.3: Schematic of the technique

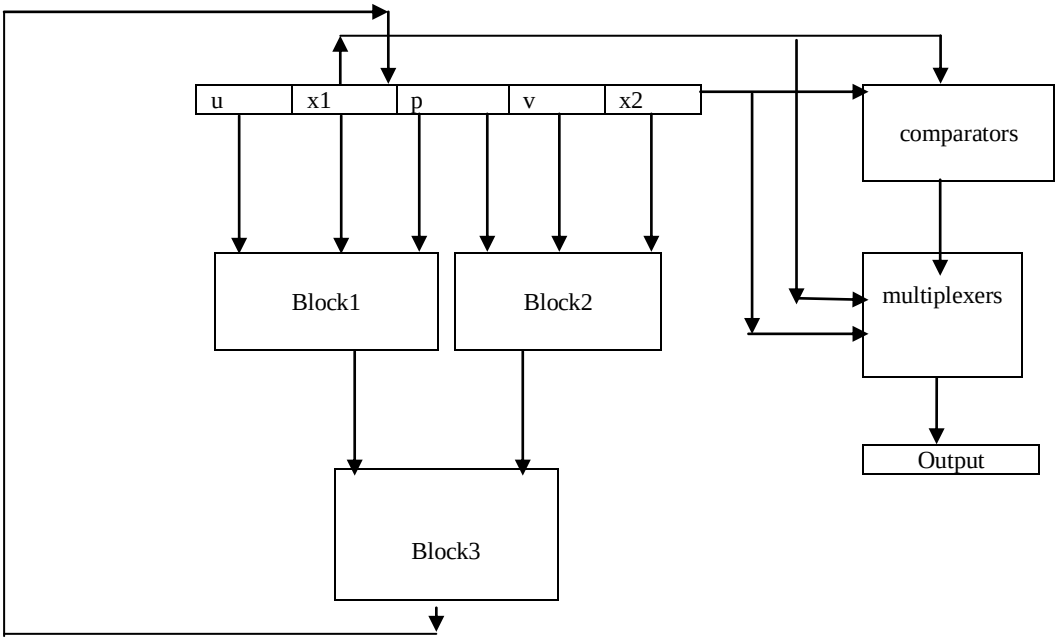

Fig.4: Proposed Model of the Architecture

5. RESULTS AND DISCUSSION

In order to compute the efficiency of this technique, we computed the number of concurrent inputs x and p-x that produce difference in the number of iterations and average difference in the number of iterations over different Mersenne's Primes. The results of these are tabulated in table 3.

Because of the unfeasibility to compute average difference in the number of iterations with all possible concurrent inputs x and p-x over Mersenne's Primes 2^{61} -1, 2^{89} -1, 2^{107} -1, 2^{127} -1, 2^{521} -1, we computed average difference in the number of iterations over randomly selected areas of number space of each prime. The result of this is tabulated in Table 4. It is found that the Binary Scalar Multiplication algorithm to compute Q=k.P over GF(p_{521}) reduce the number of iterations in the step2 of Binary Inversion Algorithm on average by (521-1)1.62+(521/2)1.62=1264.41.

6. CONCLUSION

We have presented a technique to speed up the computation of inversion by reducing the number of iterations. This is achieved by applying the concepts of parallel processes which use Binary Inversion Algorithm. This technique has been applied to NIST prime p_{521} which reduce the number of

iterations on average by 1264.41.Our future effort will target speeding up computation of individual computational blocks, integration of the proposed architecture with these arithmetic modules to perform scalar multiplication and its FPGA implementation.

Table3: Number of concurrent inputs x and p-x that produce difference in the number of iterations over different Mersenne's Primes

Mersenne's Prime	Number of concurrent inputs x and p-x that produce difference in number of iterations	Percentage of concurrent inputs x and p-x that produce difference in number of iterations	Average difference in the number of iterations
2^7-1	36	56.69	1.6
2^{13}-1	2276	55.57	1.58
2^{17}-1	36912	56.32	1.61
2^{19}-1	147481	56.25	1.629
2^{31}-1	605483014	56.39	1.613

Table 4: Average difference in the number of iterations over Mersenne's Primes 2^{61} -1, 2^{89} -1, 2^{107} -1, 2^{127} -1, 2^{521} -1

Mersenne's Prime	Average difference in the number of iterations
2^{61} -1	1.65
2^{89} -1	1.73
2^{107} -1	1.54
2^{127} -1	1.56
2^{521} -1	1.62

7. REFERENCES

[1] N. Koblitz, "Elliptic curve cryptosystems", Math.comput., vol.48,pp.203-209,1987.

[2] V.Miller, "Use of elliptic curves in cryptography", in Advances in Cryptology (CRYPTO), Newyork:Springer, 1986,vol.218,pp.417-426.

[3] A.Daly, W.Marnane, T.Kerins, E.Popovici, An FPGA implementation of a GF(p) ALU for encryption processors, Microprocessors and Microsystems, vol.28,2004,pages 253-260.

[4] Santhosh Ghosh, Monjur Alam,Indranil Sen Gupta , Dipanwita Roy Chowdhury, IIT Kharagpur,10th Euromicro Conference on Digital System Design Architectures, Methods and Tools(DSD 2007).

[5] G.B. Agnew, R.C. Mullin and S.A Vanstone", An implementation of elliptic curve cryptosystems over F2155", IEEE J. Selected Areas of Communication, vol.11,n05,pp.804-813, Jun.1993.

[6] J. Goodman and A. Chandrakasan, "An energy efficient reconfigurable public-key cryptography processor architecture", in Cryptographic Hardware and Embedded Systems (CHES). New York: Springer, 2000, vol.1965,pp.175-190.

[7] A.Satoh and K.Takano, "A Scalable dual-field elliptic curve cryptography processor", IEEE Transaction on Computers, vol.52,no.4,pp.449-460,Apr.2003

[8] Kimmo Jarvinen and Jorma Skytta, "On Parallellization of High-Speed Processors for Elliptic Curve Cryptography", IEEE transactions on VLSI Systems vol.16, no.9,Sept.2008.

[9] Kendall Anayi,Hamad Alrimeih and Daler Rakhmatov, "Flexible Hardware Processor for Elliptic Curve Cryptography", IEEE Transactions on VLSI Systems, vol.17,No.8,August 2009.

[10] A.Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor", IEEE Trans. Comput., vol.52,no.4,pp.449-460, April 2003.

[11] G. Orlando and C. Paar, " A scalable GF(p) Elliptic Curve Processor Architecture for Programmable Hardware", pp.356-371, CHES 2001, LNCS 2162.

[12] S.B. Ors, L. Batina and B. Preneel, " Hardware implementation of elliptic curve processor over GF(p)", in Proc. 14th IEEE International Conference on Application-Specific Systems, Architectures and Processors, Jun 2003, pp.433-443.

[13] Ciaran J.Mcivor, Marie McLoone, "Hardware Elliptic Curve Cryptographic Processor Over GF(p)", IEEE Trans. On Circuits and Systems-1, vol.53,No.9, September 2006.