FPGA Implementation of Itoh-Tsujii Inversion Algorithm

Ravi Kishore Kodali
Department of Electronics and Communication Engineering,
National Institute of Technology, Warangal

Chandana N. Amanchi
Department of Electronics and Communication Engineering,
National Institute of Technology, Warangal

Shubham Kumar
Department of Electronics and Communication Engineering,
National Institute of Technology, Warangal

Lakshmi Boppana
Department of Electronics and Communication Engineering,
National Institute of Technology, Warangal

***Abstract*—Elliptic Curve Cryptography (ECC) has been gaining popularity due to its shorter key size requirements. It uses arithmetic operations including addition, subtraction, multiplication and inversion in finite fields. For an efficient implementation of ECC, it is very important to carry out these operations faster using lesser resources. The in version operation consumes most of the time and more resources. The *Itoh-Tsujii* algorithm can be used to carry out the computation of multiplicative inverse by making use of *Brauer* addition chains in less time. This work presents an FPGA implementation of the multiplicative inversion for the key lengths of 194-, 233-, and 384- bits. A resource comparison for these key lengths is also made. This work uses *Sunar-Koc* multiplier for the finite field, GF(2^m) multiplication.**

Keywords: ECC; multiplication; inversion.

I. INTRODUCTION

Finite fields are widely used while implementing various cryptographic techniques. Finite field operations, mainly inversion plays a key role in the Elliptic curve cryptography (ECC) implementation. Consider an element a GF (2m). The inverse of a is a−1 GF (2m) for which a.a−1 = 1. There are many algorithms that have been proposed to compute multiplicative inverse such as Extended Euclidean algorithm, Binary inversion algorithm, Itoh-Tsujii (IT) inversion algorithm, and so on. Among these, the Itoh-Tsujii algorithm consumes fewer clock cycles to carry out the multiplicative inversion of an element. The basis for this algorithm is Fermat's little theorem. According to this theorem, for a non-zero element a GF (2m), a−1 = a2m −2. This exponentiation can be carried out by squaring and multiplication operations.

The Itoh-Tsujii algorithm makes use of Brauer addition chains to decrease the number of multiplication operations required to compute a2m−2. The squaring operation in normal basis representation, which is achieved by using cyclic shift operations, is less compute intensive. The multiplication operation is more complex. Hence, it is required to choose an efficient algorithm to carry out the multiplication operation. The Sunar-Koc finite field multiplication algorithm is the one in which the numbers are converted into optimal normal basis of type-II, then multiplied and then converted back. It requires only 1.5 × (m2 − m) number of XOR gates for GF(2m).

We have implemented the algorithm using Field programmable gate array (FPGA) with key lengths of 194-, 233-, and 384- bits. A resource comparison of the implementation of the IT algorithm for different fields: GF(2194), GF(2233), GF(2384) based on the utilized LUTs, flip-flops, and maximum allowable clock frequency is also presented. The rest of the paper is organized as follows: section II provides literature review, section III presents mathematical preliminaries which include the optimal normal basis of type II and Brauer addition chains. The Sunar-Koc multiplier is discussed in section IV. The Itoh-Tsujii inversion algorithm and its implementation for GF (2384) are presented in section V. The results and comparison are given in section VI and section VII concludes the work.

II. LITERATURE REVIEW

The IT inversion algorithm is generalized for extension fields, GF(qm) in [1]. But the implementation is done using standard basis representation. The algorithm is generalised for standard basis and required expressions for complexity are derived. Frobenius map is explored for exponentiations and it is shown that the standard basis complexity is almost similar to that of normal basis. The inversion algorithm using standard basis with minimum number of clock cycles is implemented in [2]. The algorithm is implemented and compared using squarer ITA, quad ITA, Full ITA, etc.

It perform the inversion over GF(2233) or GF(2409) within 10 clock cycles. Wang proposed the implementation of the algorithm using normal basis, which uses (m − 2) multiplication operations and (m − 1) cyclic shifts in GF(2m). Itoh and Tsujii have proposed a fast algorithm by making use of normal bases [3]. Normal basis has a useful property that the squaring can be achieved by mere cyclic shift operation. It is proved that the proposed algorithm is capable of computing the

[978-1-4799-4040-0/14/$31.00 ©2014 IEEE]

multiplicative inverse with a maximum of 2 log2(m − 1) multiplications and (m − 1) cyclic shifts.

The *Itoh-Tsujii* algorithm is modified for the binary fields which are generated by irreducible trinomials for efficient implementation on FPGA platforms in [4]. It makes better utilization of FPGA resources and also requires shorter addition chains. The LUT utilization is maximized in this algorithm. Also, this has flexibility of scaling with respect to finite-field sizes. A theoretical model for the ITA for any Galois field and k-input (k > 3) LUT based FPGA is given in [5]. A fast hardware implementation of multiplicative inversion in GF(2^m) based on *Sunar-Koc* multiplier and *Itoh-Tsujii* inversion is presented in [6] using the optimal normal basis of type II. The IT algorithm implementation for GF(2^{233}) is discussed. The required area and clock frequency for this algorithm is compared with the previous works [7], [8], [9]. A detailed discussion on *Sunar-Koc* is given in [10], [11]. It is shown that this algorithm requires only 1.5(m^2 − m) *XOR* gates for the implementation.

III. MATHEMATICAL BACKGROUND

A. Optimal normal basis of type II

Consider an element β GF(2^m).

Then the set, M, where M ={β, β2, β22 , β23 .....β2m−1}, is called the normal basis with β as the normal element. The optimal normal basis of type II for GF(2^m) can be calculated by making use of β if there exists γ such that $\beta = \gamma + \gamma^{-1}$, and γ is the primitive (2m + 1)th root of unity.

i.e. $\gamma^{2m+1} = 1$ and $\gamma^i = 6\ 1$ for any $1 \le i < (2m + 1)$. There-fore,an optimal normal basis of type II can be constructed only if p is prime, where (p = 2m + 1) and also if either of the following conditions hold [10]:

- 2 is a primitive root modulo p
- p = (7 mod 8) and the multiplicative order of (2 mod p) is m.

It is found that there are 319 values of m in the range m ∈ [2, 2001]. Therefore the optimal normal basis of type II

$$N = \left\{\gamma + \gamma^{-1}, \gamma^2 + \gamma^{-2}, \gamma^{2^2} + \gamma^{-2^2}, \ldots \gamma^{2^{m-1}} + \gamma^{-2^{m-1}}\right\} \quad (1)$$

B. Brauer chains

Fermat's little theorem says that the multiplicative inverse of an element a GF(2^m) is a^{2m-2}. The inverse can be given by equation (2).

$$a^{-1} = [\beta_{m-1}(a)]^2, \quad (2)$$

Where $\beta_k = a^{2^k-1} \in GF(2^m)$ [4].

Let $\beta_k(a)$ be represented by β_k. The inverse is computed using addition chain recursively as given in equation (3).

$$\beta_{k+j}(a) = \beta_j^{2^k}\beta_k = \beta_k^{2^j}\beta_j \quad (3)$$

Consider an example of finding the inverse in GF (2^{384}).

$$a^{-1} = a^{2^{384}-2} = a^{2(2^{383}-1)} = \beta_{383}^2 \quad (4)$$

It requires the calculation of β_{383}. Therefore,the Brauer addi-tion chain for 383 is obtained from Table -I and is given by equation (5).

$$U = \{1, 2, 4, 5, 10, 11, 22, 23, 46, 47, 94, 95, 190, 191, 382, 383\} \quad (5)$$

The square of β_{383} provides the inverse of a.

TABLE I BRAUER ADDITION CHAIN FOR GF(2^{384}) EXTRACTION TABLE

	$\beta_j(a)$	$\beta_{j+k}(a)$	Exponentiation
1.	$\beta_1(a)$		a
2.	$\beta_2(a)$	$\beta_{1+1}(a)$	$(\beta_1)^{2^1}\beta_1 = a^{2^2-1}$
3.	$\beta_4(a)$	$\beta_{2+2}(a)$	$(\beta_2)^{2^2}\beta_2 = a^{2^4-1}$
4.	$\beta_5(a)$	$\beta_{4+1}(a)$	$(\beta_4)^{2^1}\beta_1 = a^{2^5-1}$
5.	$\beta_{10}(a)$	$\beta_{5+5}(a)$	$(\beta_5)^{2^5}\beta_5 = a^{2^{10}-1}$
6.	$\beta_{11}(a)$	$\beta_{10+1}(a)$	$(\beta_{10})^{2^1}\beta_1 = a^{2^{11}-1}$
7.	$\beta_{22}(a)$	$\beta_{11+11}(a)$	$(\beta_{11})^{2^{11}}\beta_{11} = a^{2^{22}-1}$
8.	$\beta_{23}(a)$	$\beta_{22+1}(a)$	$(\beta_{22})^{2^1}\beta_1 = a^{2^{23}-1}$
9.	$\beta_{46}(a)$	$\beta_{23+23}(a)$	$(\beta_{23})^{2^{23}}\beta_{23} = a^{2^{46}-1}$
10.	$\beta_{47}(a)$	$\beta_{46+1}(a)$	$(\beta_{46})^{2^1}\beta_1 = a^{2^{47}-1}$
11.	$\beta_{94}(a)$	$\beta_{47+47}(a)$	$(\beta_{47})^{2^{47}}\beta_{47} = a^{2^{94}-1}$
12.	$\beta_{95}(a)$	$\beta_{94+1}(a)$	$(\beta_{94})^{2^1}\beta_1 = a^{2^{95}-1}$
13.	$\beta_{190}(a)$	$\beta_{95+95}(a)$	$(\beta_{95})^{2^{95}}\beta_{95} = a^{2^{190}-1}$
14.	$\beta_{191}(a)$	$\beta_{190+1}(a)$	$(\beta_{190})^{2^1}\beta_1 = a^{2^{191}-1}$
15.	$\beta_{382}(a)$	$\beta_{191+191}(a)$	$(\beta_{191})^{2^{191}}\beta_{191} = a^{2^{382}-1}$
16.	$\beta_{383}(a)$	$\beta_{382+1}(a)$	$(\beta_{382})^{2^1}\beta_1 = a^{2^{383}-1}$

IV. SUNAR-KOC MULTIPLICATION

The IT algorithm requires squaring and multiplication operations, while carrying out an inversion operation. The squaring operation is a simple cyclic shift in normal basis. The multiplication can be performed efficiently using *Sunar-Koc*. In this algorithm, the numbers are first converted into shifted form of canonical basis (optimal normal basis of type II) and then the product is computed. The numbers are then converted back into normal basis. This involves three steps:

1. Conversion of the normal basis into its equivalent shifted form of canonical basis

2. Computation of the product of the numbers
3. Conversion of the resultant product into its normal basis

The conversion technique of the normal basis into its equivalent shifted form of canonical basis is as follows:

- If 2 is primitive modulo p, the set of powers of (2 modulo p) $\{2, 2^2, 2^3, \ldots 2^{2m-1}, 2^{2m}\}$ (modp) is equivalent to {1, 2, 3, ...2m− 1, 2m}. Therefore the elements in the normal basis $\gamma^{2i} + \gamma^{-2i}$ can take the form $\gamma^{j} + \gamma^{-j}$ for j ∈ [1, 2m]. The powers can be brought into the range [1,m] by writing the elements $\gamma^{j} + \gamma^{-j}$ as $\gamma^{(2m+1)-j} + \gamma^{-(2m+1)+j}$ for j ≥ m + 1.
- If the multiplicative order of (2 modulo p) is equal to m, then the set of powers of (2 modulo p) $\{2, 2^2, 2^3, \ldots 2^{2m-1}, 2^{2m}\}$ are m distinct integers in [1, 2m]. Then the powers are brought into the range [1, m] as given in equation (6).

$$j = 2^i (mod\ p) \quad if \quad 2^i (mod\ p) \in [1, m] \tag{6a}$$

$$j = (2m+1) - 2^i (mod\ p) \quad if \quad 2^i (mod\ p) \in [m+1, 2m] \tag{6b}$$

Second step involves the multiplication of numbers in the shifted form of canonical basis. Let us consider two numbers X,Y □ GF (2^m). They can be expressed in basis N as given in equation (7).

$$X = \sum_{(i=1)}^{m} x_i \beta_i = \sum_{(i=1)}^{m} x_i (\gamma^i + \gamma^{-i}) \tag{7a}$$

$$Y = \sum_{(j=1)}^{m} y_j \beta_j = \sum_{(j=1)}^{m} y_j (\gamma^j + \gamma^{-j}) \tag{7b}$$

$$\begin{aligned} Z = X.Y &= (\sum_{(i=1)}^{m} x_i \beta_i).(\sum_{(j=1)}^{m} y_j \beta_j) \\ &= (\sum_{(i=1)}^{m} x_i (\gamma^i + \gamma^{-i})).(\sum_{(j=1)}^{m} y_j (\gamma^j + \gamma^{-j})) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{m} x_i y_j (\gamma^{(i-j)} + \gamma^{-(i-j)}) \\ &+ \sum_{i=1}^{m} \sum_{j=1}^{m} x_i y_j (\gamma^{(i+j)} + \gamma^{-(i+j)}) \\ &= Z1 + Z2 \end{aligned} \tag{8}$$

Z1 and Z2 can be further expressed by

$$\begin{aligned} Z1 &= \sum_{i=1}^{m} \sum_{j=1}^{m} x_i y_j (\gamma^{(i-j)} + \gamma^{-(i-j)}) \\ &= \sum_{1 \le i,j \le m}^{i \ne j} x_i y_j (\gamma^{(i-j)} + \gamma^{-(i-j)}) \end{aligned} \tag{9a}$$

$$\begin{aligned} Z2 &= \sum_{i=1}^{m} \sum_{j=1}^{m} x_i y_j (\gamma^{(i+j)} + \gamma^{-(i+j)}) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{m-i} x_i y_j (\gamma^{(i+j)} + \gamma^{-(i+j)}) \\ &+ \sum_{i=1}^{m} \sum_{j=m-i+1}^{m} x_i y_j (\gamma^{(i+j)} + \gamma^{-(i+j)}) \\ &= W1 + W2 \end{aligned} \tag{9b}$$

Hence, the product, Z, can be calculated as given [6] by equation (10).

$$Z = Z1 + W1 + W2 \tag{10}$$

The third step involves the conversion of the product back into its normal basis. The Sunar-Koc multiplication uses 1.5 × (m2 − m) XOR gates, whereas Massey-Omura algorithm uses 2 × (m2 − m) XOR gates. And the time complexities of both these algorithms are almost same. Hence Sunar-Koc algorithm is preferred for multiplication.

V. ITOH-TSUJII INVERSION ALGORITHM

Let us consider an element a ∈ GF(2m). The inverse of the element can be calculated using Itoh-Tsujii inversion algorithm as given in equation (11).

$$a^{-1} = a^{2^m - 2} \tag{11}$$

$$a^{2^m - 2} = a^{2(2^{m-1} - 1)}$$

when (m-1) is even,

$$2^{m-1} - 1 = (2^{\frac{m-1}{2}} - 1)(2^{\frac{m-1}{2}} + 1) \tag{12a}$$

When (m-1) is odd,

$$2^{m-1} - 1 = 2(2^{\frac{m-2}{2}} - 1)(2^{\frac{m-2}{2}} + 1) + 1 \tag{12b}$$

The exponents are further simplified in a similar manner. An example is worked out with m = 384.
Let us consider an element a ∈ GF(2^{384}).

$$a^{-1} = a^{2^{384}-2} = a^{2(2^{383}-1)}$$
$$a^{2^{383}-1} = a^{2(2^{382}-1)+1} = a^{2(2^{191}+1)(2^{191}-1)+1}$$
$$a^{2^{191}-1} = a^{2(2^{190}-1)+1} = a^{2(2^{95}+1)(2^{95}-1)+1}$$
$$a^{2^{95}-1} = a^{2(2^{94}-1)+1} = a^{2(2^{47}+1)(2^{47}-1)+1}$$
$$a^{2^{47}-1} = a^{2(2^{46}-1)+1} = a^{2(2^{23}+1)(2^{23}-1)+1}$$
$$a^{2^{23}-1} = a^{2(2^{22}-1)+1} = a^{2(2^{11}+1)(2^{11}-1)+1}$$
$$a^{2^{11}-1} = a^{2(2^{10}-1)+1} = a^{2(2^{5}+1)(2^{5}-1)+1}$$
$$a^{2^{5}-1} = a^{2(2^{4}-1)+1} = a^{2(2^{3}+1)(2^{2}-1)+1}$$
$$= a^{2(2^{3}+1)(2^{1}-1)(2^{1}+1)+1} = a^{2(2^{2}+1)(2^{1}+1)+1}$$

The inverse in GF(2^{384}) is computed as given in Algorithm -1.

Algorithm 1 Algorithm for the calculation of Multiplicative Inverse

Input: a, Output: a^{-1}
Calculate $\alpha = a^2$
Store α and calculate $\beta = \alpha^{2(2^{191}+1)}$
Store β and calculate $\gamma = \beta^{2(2^{95}+1)}$
Store γ and calculate $\delta = \gamma^{2(2^{95}+1)}$
Store δ and calculate $\eta = \delta^{2(2^{47}+1)}$
Store and calculate $\vartheta = \eta^{2(2^{23}+1)}$
Store ϑ and calculate $\lambda = \vartheta^{2(2^{11}+1)}$
Store λ and calculate $\mu = \lambda^{2(2^{5}+1)}$
Store μ and calculate $\sigma = \mu^{2(2^{2}+1)(2^{1}+1)}$
Compute the product of $\alpha, \beta, \gamma, \delta, \eta, \vartheta, \lambda, \mu,$ and σ

The algorithm for the calculation of multiplicative inverse in GF(2^{384}) is presented below:
It can be implemented using three registers:(T_1, T_2 and invout), a multiplier and a squarer. Multiplicative inverse algorithm for GF(2^{384}):

Input : a
Output : a^{-1}
Registers : T_1, T_2, invout
1. $T_1 \leftarrow a$, $T_2 \leftarrow a$
   $T_1 = a$, $T_2 = a$
2. invout$\leftarrow T_1T_2$, $T_1 \leftarrow T_1T_2$
   $T_1 = a^2$, invout $= a^2$
3. $T_1 \leftarrow T1^2$, $T_2 \leftarrow T1^2$
   $T_1 = a^{2(2)}$, $T_2 = a^{2(2)}$
4. cyclic shift T_2 191 times:$T_2 \leftarrow T_2^{2^{191}}$
   $T_2 = a^{2(2(2^{191}))}$
5. $T_1 \leftarrow T_1T_2$
   Let $t = 2(2^{191} + 1)$
   $T_1 = a^{2(t)}$
6. invout$\leftarrow$ invout.T_1
   invout $= a^{2[t+1]}$
7. $T_1 \leftarrow T_1^2$, $T_2 \leftarrow T_1^2$
   $T_1 = a^{2t(2)}$, $T_2 = a^{2t(2)}$
8. cyclic shift T_2 95 times: $T_2 \leftarrow T_2^{2^{95}}$
   $T_2 = a^{2t(2)(2^{95})}$
9. $T_1 \leftarrow T_1T_2$
   Let $u = 2(2^{95} + 1)$
   $T_1 = a^{2tu}$
10. invout$\leftarrow$ invout.T_1
    invout $= a^{2[t[u+1]+1]}$
11. $T_1 \leftarrow T_1^2$, $T_2 \leftarrow T_1^2$
    $T_1 = a^{2tu(2)}$, $T_2 = a^{2tu(2)}$
12. cyclic shift T_2 47 times:$T_2 \leftarrow T_2^{2^{47}}$
    $T_2 = a^{2tu(2)(2^{47})}$
13. $T_1 \leftarrow T_1T_2$
    Let $v = (2)(2^{47} + 1)$
    $T_1 = a^{2tuv}$
14. invout$\leftarrow$ invout.T_1
    invout $= a^{2[t[u[v+1]+1]+1]}$
15. $T_1 \leftarrow T_1^2$, $T_2 \leftarrow T_1^2$
    $T_1 = a^{2tuv(2)}$,
    $T_2 = a^{2tuv(2)}$
16. cyclic shift T_2 23 times:$T_2 \leftarrow T_2^{2^{23}}$
    $T_2 = a^{2tuv(2)(2^{23})}$
17. $T_1 \leftarrow T_1T_2$
    Let $w = 2(2^{23} + 1)$
    $T_1 = a^{2tuvw}$
18. invout$\leftarrow$ invout.T_1
    invout $= a^{2[t[u[v[w+1]+1]+1]+1]}$
19. $T_1 \leftarrow T_1^2$, $T_2 \leftarrow T_1^2$
    $T_1 = a^{2tuvw(2)}$,
    $T_2 = a^{2tuvw(2)}$
20. cyclic shift T_2 11 times:$T_2 \leftarrow T_2^{2^{11}}$
    $T_2 = a^{2tuvw(2)(2^{11})}$
21. $T_1 \leftarrow T_1T_2$
    Let $x = (2)(2^{11} + 1)$
    $T_1 = a^{2tuvwx}$
22. invout$\leftarrow$ invout.T_1
    invout $= a^{2[t[u[v[w[x+1]+1]+1]+1]+1]}$
23. $T_1 \leftarrow T_1^2$, $T_2 \leftarrow T_1^2$
    $T_1 = a^{2tuvwx(2)}$,
    $T_2 = a^{2tuvwx(2)}$
24. cyclic shift T_2 5 times:$T_2 \leftarrow T_2^{2^5}$
    $T_2 = a^{2tuvwx(2)(2^5)}$
25. $T_1 \leftarrow T_1T_2$
    Let $y = 2(2^5 + 1)$
    $T_1 = a^{2tuvwxy}$
26. invout$\leftarrow$ invout.T_1
    invout $= a^{2[t[u[v[w[x[y+1]+1]+1]+1]+1]+1]+1]}$
27. $T_1 \leftarrow T_1^2$, $T_2 \leftarrow T_1^2$
    $T_1 = a^{2tuvwxy(2)}$,
    $T_2 = a^{2tuvwxy(2)}$
28. cyclic shift T_2 2 times:$T_2 \leftarrow T_2^{2^2}$

$T_2 = a^{2tuvwxy(2)(2^2)}$

29. $T_1 \leftarrow T_1T_2$

$T_1 = a^{2tuvwxy(2)(2^2+1)}$

30. cyclic shift T_2 1 times: $T_2 \leftarrow T_2^{2^1}$

$T_2 = a^{2tuvwxy(2)(2^2+1)(2^1)}$

31. $T_1 \leftarrow T_1T_2$

Let $z = (2)(2^2 + 1)(2^1 + 1)$

$T_1 = a^{2tuvwxyz}$

32. invout$\leftarrow$ invout.T_1

invout = $a^{2[t[u[v[w[x[y[z+1]+1]+1]+1]+1]+1]+1]}$

The inverse of a, a^{-1} is available at invout.

$$\begin{aligned}
\text{invout} &= a^{2[t[u[v[w[x[y[z+1]+1]+1]+1]+1]+1]+1]} \\
&= a^{2[t[u[v[w[x[y[2(2^5+1)(2^1+1)(2^1-1)+1]+1]+1]+1]+1]+1]+1]} \\
&= a^{2[t[u[v[w[x[2(2^5+1)[2^5-1]+1]+1]+1]+1]+1]+1]} \\
&= a^{2[t[u[v[w[2(2^{11}+1)[2^{11}-1]+1]+1]+1]+1]+1]} \\
&= a^{2[t[u[v[2(2^{23}+1)[2^{23}-1]+1]+1]+1]+1]} \\
&= a^{2[t[u[2(2^{47}+1)[2^{47}-1]+1]+1]+1]} \\
&= a^{2[t[2(2^{95}+1)[2^{95}-1]+1]+1]} \\
&= a^{2[2(2^{195}+1)[2(2^{191}-1)]+1]} \\
&= a^{2[2^{383}-1]} = a^{2^{384}-2} = a^{-1}
\end{aligned}$$

VI. RESULTS

The *Itoh-Tsujii* inversion algorithm has been implemented for key lengths of 194-, 233-, and 384- bits on the Virtex 7 XC7V2000T-1FLG1925 FPGA device. The multiplier is implemented using *Sunar-Koc* technique. The numbers are first converted into basis N. They are multiplied in N basis and the result is converted back into basis M. The squarer is nothing but a cyclic shifter which is implemented using a register. The synthesis results are included in Table -II. The number of LUTs and flip-flops used are compared for different key lengths. Also, the maximum frequency of operation is also compared for these key lengths. The simulation results are also provided in Figure 1.

The implementation for 194-, 233-, and 384- bit key lengths consumed 20, 21 and 32 clock cycles respectively.

TABLE II FPGA SYNTHESIS RESULTS FOR DIFFERENT KEY LENGTHS

Parameter		Key length in bits		
		194-	233-	384-
Logic Utilization	Avail-able	Used	Used	Used
No. of LUT's	1221600	62431	87742	51227
No. of FF's	2443200	587	704	966
No. of IO's	1200	390	468	770
Max. freq.(MHz)		103.27	92.32	99.5

VII. CONCLUSIONS

The implementation of *Itoh-Tsujii* inversion algorithm using *Sunar-Koc* multiplier for optimal normal basis of type II is presented. The inversion operation for GF (2^{384}) has been illustrated with an example. A resource comparison of the FPGA implementations of the algorithm for GF(2^{194}),GF(2^{233}), and GF(2^{384}) is given.

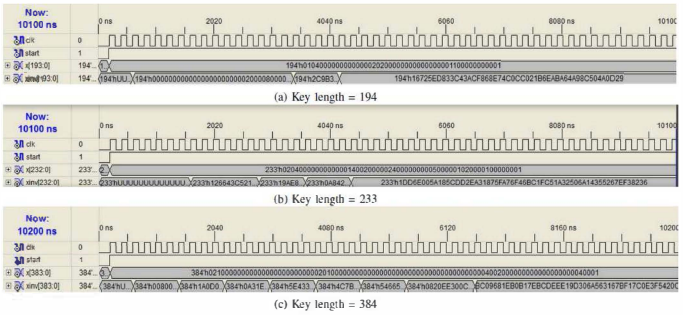

Fig. 1. Simulation results for three different key lengths

REFERENCES

[1] J. Guajardo and C. Paar, "Itoh-tsujii inversion in standard basis and its application in cryptography and codes," *Designs, Codes and Cryptography*, vol. 25, no. 2, pp. 207–216, 2002. [Online]. Available: http://dx.doi.org/10.1023/A%3A1013860532636

[2] L. Parrilla, A. Lloris, E. Castillo *et al.*, "Minimum-clock-cycle itoh-tsujii algorithm hardware implementation for cryptography applications over gf (2¡ sup¿ m¡/sup¿) fields," *Electronics letters*, vol. 48, no. 18, pp. 1126–1128, 2012.

[3] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in gf(2m) using normal bases," *Information and Computation*, vol. 78, no. 3, pp. 171 – 177, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0890540188900247

[4] C. Rebeiro, S. S. Roy, D. Reddy, and D. Mukhopadhyay, "Revisiting the itoh-tsujii inversion algorithm for fpga platforms," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 8, pp. 1508–1512, 2011.

[5] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, "Theoretical modeling of the itoh-tsujii inversion algorithm for enhanced performance on k-lut based fpgas," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 2011, pp. 1–6.

[6] Q. Deng, X. Bai, L. Guo, and Y. Wang, "A fast hardware implementation of multiplicative inversion in gf(2m)," in *Microelectronics Electronics, 2009. PrimeAsia 2009. Asia Pacific Conference on Postgraduate Research in*, 2009, pp. 472–475.

[7] A. A.-A. Gutub, A. F. Tenca, E. Savas¸, and R. K. Koc¸, "Scalable and unified hardware to compute montgomery inverse in gf (p) and gf (2n)," in *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, 2003, pp. 484–499.

[8] J. Goodman and A. P. Chandrakasan, "An energy-efficient reconfigurable public-key cryptography processor," *Solid-State Circuits, IEEE Journal of*, vol. 36, no. 11, pp. 1808–1820, 2001.

[9] F. Rodr´ıguez-Henr´ıquez, G. Morales-Luna, N. A. Saqib, and N. Cruz-Cort´es, "A parallel version of the itoh-tsujii multiplicative inversion algorithm," in *Reconfigurable Computing: Architectures, Tools and Applications*. Springer, 2007, pp. 226–237.

[10] B. Sunar and C. K. Koc¸, "An efficient optimal normal basis type ii multiplier," *Computers, IEEE Transactions on*, vol. 50, no. 1, pp. 83– 87, 2001.

[11] R. K. Kodali, P. Gomatam, and L. Boppana, "Implementations of sunar-koc multiplier using fpga platform and wsn node," in *TENCON 2013 - 2013 IEEE Region 10 Conference (31194)*, Oct 2013, pp. 1–4.