A Hex Itoh-Tsujii inversion algorithm for FPGA platforms

Murugesan Kalaiaarasì, Vepadappu Raman Venkatasubramani, and Sivasubramanian Rajaram

Abstract The Itoh-Tsujii inversion algorithm forms a major contribution in finding the inverse in cryptographic applications such as Elliptic Curve Cryptography. In this paper, a new Hex Itoh-Tsujii inversion algorithm is proposed to compute the multiplicative inverse efficiently on Field-Programmable Gate-arrays (FPGA) platforms for binary fields generated by NIST recommended irreducible trinomials. The Hex Itoh Tsujii inversion Algorithm based proposed architecture is constructed with hex circuits and quad addition chain. This combination improves the resource utilization. The experimental results show that the proposed work has better Area-Time Performance compared with the existing implementation.

Keywords: field programmable gate arrays (FPGA), Itoh-Tsujii inversion algorithm (ITA), lookup table (LUT), finite fields (FF)

Classification: Integrated circuits (memory, logic, analog, RF, sensor)

1. Introduction

In the present digital environment, the Finite Field (FF) arithmetic plays an important role in secure communication. The performance of the digital secure system in the field of cryptography, depends on performance, area, and time of the underlying arithmetic operations [1]. It is crucial to carry out arithmetic operations such as addition, multiplication, and inversion rapidly for secure communication over \(GF(2^m)\), where \(m\) is the field size [2]. Among different arithmetic operation, FF inversion is one of the most important and computationally complex operations. Several attempts have been made to fasten the inverse operation [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22] using squarer and quad circuits on FPGA platforms. There are two useful approaches for finding an inverse over \(GF(2^m)\) which includes Extended Euclidean Algorithm (EEA) [14, 23] and Fermat’s Little Theorem (FLT) [24]. EEA requires more additional logic to compute the Greatest Common Divisor of the polynomial and is not much suitable for area efficient hardware implementation. FLT theorem is used to compute the inverse of an element over \(GF(2^m)\) by exponentiation of \(2^m - 2\), which reuses the multiplier and squarer that is more suitable for hardware [12]. It reduces the extension field inversion to binary field inversion which is easier for inverse operation. This express uses FLT to obtain an inverse architecture with high Area-time performance over irreducible trinomial \(GF(2^m)\). We propose a new Hex inversion algorithm to find the inverse of an element \(A \in GF(2^m)\). Also, we propose a new Hex circuit to perform the hex exponentiation \((A^{16})\) in the inverse architecture with reduced area. The inverse algorithm for this architecture follows a Quad addition sequence [21] for irreducible trinomials. It achieves higher Area-Time Performance (ATP) than the existing techniques specified in the literature. In addition, the proposed architecture is scalable for any irreducible trinomial.

2. Preliminaries

The inverse of an element \(A \in GF(2^m)\) according to FLT is given by

\[
A^{2^m-1} = 1 \quad A + A^{2^m-2} = 1 \quad A^{-1} = A^{2^m-2} \quad (1)
\]

\[
A^{-1} = (A(A^{2^{m-1}})^2')^2 \quad (2)
\]

From Eq.(2) the inverse of an element \(A\) is obtained from \(A^{2^m-2}\), that requires \(m - 1\) squaring and \(m - 2\) multiplications. In [24], the authors introduced addition chain [25] in the Itoh Tsujii inversion Algorithm (ITA) that reduces the required number of multiplications to \((2 \times \log_2^{m-1})\) with \(m-2\) number of squaring. ITA was first introduced in Gaussian normal basis [14, 24, 26, 27], later in [3] the algorithm was presented over Polynomial basis using addition chain for the field size \(GF(2^{193})\). In [3] parallel implementations were made for the same field size using squarer circuits. Later in [5], this squarer circuits was replaced by Quad circuits in-order to reduce the LUTs and improve the performance. Generally the inverse of an element \(A \in GF(2^m)\) is given by the following equation Eq.(3) [3, 5].

\[
A^{-1} = (\beta_{p-1}(A))^2 \quad (3)
\]

For any non zero element \(A \in GF(2^m)\), the inverse is computed recursively using [3] Eq.(4). Where \(\beta_p(A) = A^{2^p-1}\) where \(p \in N\) and \(p = m - 1\). For simplicity \(\beta_p(A)\) is denoted as \(\beta_p\), for any element \(\beta_{p+q}\) can be expressed as a function of \(\beta_p\) and \(\beta_q\).

\[
\beta_{p+q} = (\beta_p)^{2^q} \ast \beta_q \quad (4)
\]

3. Proposed design

Generally exponentiation is done by squarer circuits [3, 4, 6, 7, 12, 18, 20, 28], quad circuits [5, 21, 22] and octet circuits (field size \(GF(2^{256})\)) [18, 28]. We propose a novel HITA using Hex circuits to compute exponentiation in-order to save the area with increased ATP. This section gives a brief overview of the construction of Hex circuits in HITA and the hardware architecture for the HITA using those Hex circuits.
3.1 Hex circuit

In this paper, the Hex circuit is constructed using xor gate as shown in Table I. The squaring \(( c = a + a = a^2 )\) of two polynomials with irreducible trinomial \(P(x)\), is given by \(C(x) = A^2(x) \mod P(x)\). For example the polynomial representation of \(A(x)\) upto \(m\)-bit is given in Eq.(5) and its squaring is given in Eq.(6).

\[
A(x) = \sum_{i=1}^{m-1} A_i x^i \quad (5)
\]

\[
A^2(x) = \sum_{i=1}^{m-1} A_i^2 x^i \mod P(x) \quad (6)
\]

The squaring of \(m\) bit is performed by inserting a zero between consecutive bits of binary representation which results in \(2m - 1\) bits [29]. Then the reduction takes place as per Fig. 1 and the final result \(C\) is obtained by \(C = P @ Q @ R @ S\), which is of \(m\) bits resulting in reduced squaring. The repetition of these steps leads to the final hex circuit. The squarer, quad and hex operation over the field \(GF(2^{15})\) with irreducible trinomial \(\chi^5 + \chi + 1\) is listed in the Table I using XORs. Since, polynomial FF arithmetic are carry free, addition operations are simple XORs [1] that is more suitable for hardware.

As seen in Table I, FPGA with 6 input LUTs yields the best LUT utilization with Hex circuits without significant change in delay. This is also shown in Table II for an element \(A \in GF(2^{233})\). The synthesis results of squarer, and hex in Virtex4 (XC4VFX140), Virtex 5 (XC5VLX220) and Virtex 7 (XC7VX1140) FPGA shows that Hex circuit has less number of underutilized LUTs as compared to squarer and quad circuits. The Table III shows the comparison of squarer, Quad, and Hex block for two different fields \(GF(2^{15})\) and \(GF(2^{233})\) synthesized in Virtex 7, which also shows a lower percentage of underutilized LUTs among the available Quad and squarer device.

\[
LUT_{SR_{hex}} = \frac{\#LUT_{Hex}}{2 \times \#LUT_{Quad}} \approx 0.73
\]

\[
LUT_{SR_{hex}} = \frac{\#LUT_{Hex}}{4 \times \#LUT_{square}} \approx 0.55
\]

From the Table II & III, it is shown that the hex circuit saves about 26\% \(LUT_{SR_{hex}}\) of LUTs compared to the quad circuit and 45\% \(LUT_{SR_{hex}}\) of LUTs compared to the squarer circuit. Thus, in this work, hex circuits are used to complete the inverse of the element for the field generated by irreducible trinomials on FPGA platform.

3.2 Hex Itoh-Tsujii algorithm (HITA)

On reflection to the observations of low percentage of underutilized LUTs for hex circuits in Table III, we propose a hex-ITA algorithm for irreducible trinomials. In this work, we have used the quad addition chain [21], that is formed by decomposing the value of \(m\). Owing to its smaller addition chain length compared with brauer chain, the resulting addition chain for NIST recommended trinomials is shown in the Table IV. The following theorem and proof shows that the inverse of an element \(A \in GF(2^m)\) can be generated from HITA.

**Theorem 1:** The inverse of any nonzero element \(A \in GF(2^m)\) using HITA is

\[
A^{-1} = \left[R_{(m-1)\bmod 4}(A)\right]^2 \quad (7)
\]

The multiplicative inverse is computed using recursive operation(hex and multiplication) with the help of addition chain

| Table I | Square, Quad and Hex circuit for \(GF(2^{15})\) |
|---------|----------------------------------|
| Output | Square | Quad | Hex |
| 0       | \(b_0\) | \(b_0 + b_{11}\) | \(b_0 + b_{12} + b_{14}\) |
| 1       | \(b_1 + b_2\) | \(b_2 + b_{11}\) | \(b_2 + b_{12} + b_{14}\) |
| 2       | \(b_3 + b_4\) | \(b_4 + b_{11}\) | \(b_4 + b_{12} + b_{14}\) |
| 3       | \(b_5 + b_6\) | \(b_6 + b_{11}\) | \(b_6 + b_{12} + b_{14}\) |
| 4       | \(b_7 + b_8\) | \(b_8 + b_{11}\) | \(b_8 + b_{12} + b_{14}\) |
| 5       | \(b_9 + b_{10}\) | \(b_{10} + b_{11}\) | \(b_{10} + b_{12} + b_{14}\) |
| 6       | \(b_11 + b_{12}\) | \(b_{12} + b_{11}\) | \(b_{12} + b_{12} + b_{14}\) |
| 7       | \(b_{13} + b_{14}\) | \(b_{13} + b_{11}\) | \(b_{13} + b_{12} + b_{14}\) |

**Fig. 1** Reduction of \(A^2(x)\)

**Table II** LUT and delay comparison of squarer, Quad and Hex circuits on Virtex 5, 6 and 7 FPGAs size ratio

| Device | Square | Quad | Hex | LUT delay | LUT delay | LUT delay |
|--------|--------|------|-----|-----------|-----------|-----------|
| Virtex6 | 153 | 3.706 | 230 | 3.881 | 316 | 4.056 | 0.53 | 0.73 |
| Virtex7 | 153 | 0.844 | 230 | 0.989 | 340 | 1.01 | 0.55 | 0.74 |
| Virtex7 | 153 | 0.813 | 230 | 0.953 | 340 | 0.98 | 0.55 | 0.74 |

| SR-SizeRatio, \(hq\)-Hex and quad-Square |
|----------------------------------------|

**Table III** Comparison of squarer, Quad and Hex circuits for field size \(GF(2^{15})\) and \(GF(2^{233})\) on Virtex 7 FPGA (XC7Vx1140 – 2fg1930)

| Field size | Square | Quad | Hex | LUT delay | LUT delay | LUT delay |
|------------|--------|------|-----|-----------|-----------|-----------|
| \(GF(2^{15})\) | 153 | 3.706 | 230 | 3.881 | 316 | 4.056 | 0.5490 | 0.7304 |
| \(GF(2^{233})\) | 153 | 0.844 | 230 | 0.989 | 340 | 1.01 | 0.555 | 0.7391 |

| Table IV | Quad addition chain for \(GF(2^m)\) ; \(m=193, 233, 409\) |
|----------|------------------|
| \(m\) | Quadr Addition Chain | Bauer addition chain |
| 193 | \(1,2,3,12,48,192\) | \(1,2,3,6,12,24,48,96,192\) |
| 233 | \(1,4,13,16,58,232\) | \(1,2,3,6,7,14,28,29,58,116,232\) |
| 409 | \(1,2,6,8,24,26,102,408\) | \(1,2,3,6,12,24,25,50,51,102,204,408\) |
in [30]. Here we define \([\beta_{(m-1)/4}(A)] = \beta_k\) where \(k\) is constant and is given as,

\[ \beta_k = A^{16^k-1} \quad (8) \]

let \(k = p + q + r + s\) with \(p, q, r, s\) as four positive integer, then \(\beta_{p+q+r+s} \in GF(2^m)\) can be expressed using lemma [30] as

\[ \beta_{p+q+r+s} = (\beta_{p+q})^{16^r+s} \beta_{p+q} \quad (9) \]

where \(\beta_{p+q} = (\beta_p)^{16^r} \beta_p\) and \(\beta_{r+s} = (\beta_r)^{16^s} \beta_r\). The values of \(p, q, r, s\) depends on the addition chain. **Proof for Eq.(3)**

\[ A^{-1} = [\beta_{(m-1)/4}(A)]^2 = (A^{16^{(m-1)/4}-1})^2 \]

\[ = (A^{2^{(m-1)/4}-1})^2 \quad (10) \]

**Proof for Eq.(8) by substituting Eq.(9)**

\[ \beta_k = \beta_{p+q+r+s} = (\beta_{p+q})^{16^r+s} \beta_{p+q} \]

\[ = ((\beta_{p+q})^{16^r} \beta_{p+q})^{16^s} \beta_{p+q} \]

\[ = ((A^{16^r-1})^{16^s} \beta_{p+q})^{16^r+s} \beta_{p+q} \]

\[ = (A^{16^r+s-1})^{16^s} (A^{16^r-1})^{16^r+s} A^{16^r+s-1} \]

\[ = (A^{16^r+s-1})^{16^r+s-1} \] (11)

The procedure to obtain the inverse using hex-ITA is given in Algorithm 1. For the input \(A\), the algorithm (1) computes \(A^{-1} = [\alpha_{(m-1)/4}]^2\) for \(GF(2^m)\). The benefit of this algorithm is the use of Hex circuits instead of Quad and it also uses an addition chain of length \(m - 1\) instead of addition chain length \(m - 1\), which speeds up the process. This saves the clock cycles.

The inverse for an element \(A \in GF(2^{233})\) \((A^{-1} = [\alpha_{58}(A)]^2)\) using HITA is shown in Table V. This computation requires the quad addition chain of length \(\frac{m-1}{2}\) with maximum value of its element as 58 instead of 232. Table V shows the steps to compute the inverse of an element \(A \in GF(2^{233})\). Using the quad add chain, this algorithm calculates \(A^{-1}\) that requires 63 hex operations. The number of clock cycle for Hex ITA is given by the Eq.(11). It includes 5 clock cycle for pre-computation and 1 clock cycle for final squaring and 2L (L-length of addition chain) clock cycles for multiplications and remaining for hex operation.

### Algorithm 1 Algorithm for Proposed HITA

**input** : \(A, \alpha_c, \text{ addition chain for the given field} \)  
**output** : \(A^{-1}\)  
**begin**

\(L = length(\alpha_c)\)

\(A^3 = A \times A\)

\(A^3 = A^3 \times A; A^3 \times A = A^3; \alpha_{ac} = A^{15}\)

**foreach** \((\alpha_{ci} \in \alpha_c (2 \leq i \leq L))\) do

If \((\alpha_{ci} \times 0 = 0)\) then

\(\alpha_{p*} = \alpha (\alpha_{p})^{16^s} \alpha_q = \alpha_{r+s}\)

\(\alpha_{ac_1} = (\alpha_{p*})^{16^s} \alpha_{r+s}\)

end

If \((\alpha_{ci} \times 0 = 1)\) then

\(\alpha_{p*} = \alpha (\alpha_{p})^{16^s} \alpha_q = \alpha_{r+s}\)

\(\alpha_{ac_1} = (\alpha_{p*})^{16^s} \alpha_{r+s}\)

end

end

**end**

\(A^{-1} = \alpha_{L} \times \alpha_{L} \) (Finalsquaring)

**End**

\[ Clockcycle = 5 + 1 + (2L) + \sum_{i=2}^{L} (\frac{\alpha_{ci} - \alpha_{ci-1}}{\alpha_{cs}}) + 1 \] (11)

Overall, the computation of inverse of an element \(A \in GF(2^{233})\) takes 33 clock cycles as indicated in Table V.

### 3.3 Architecture

The proposed HITA architecture for computing inverse of the element \(A\) using the Table V over the field \(GF(2^{233})\) is shown in Fig. 2. The two key blocks in the proposed architecture are Multiplier block and Hex block. The multiplier used here is Hybrid Karatsuba multiplication algorithm [24], which is a combinational multiplier.

The Hex block in Fig. 2 is constructed by cascaded hex circuits as shown in Fig. 3. The number of cascaded hex circuits affects the area-time performance of HITA. For \(GF(2^{233})\), the best performance is obtained with four cascaded hex circuits as shown in Fig. 4. Hence, in the proposed architecture, the hex block is designed with four cascaded hex circuits. A multiplexer in hex block controlled by \(H_{sel}\) determines which one of the four power has to be selected out to the input of \(H_{out}\) register. In Fig. 3, if \(x \in GF(2^{233})\)
is the input to hex block \( (H_{in}) \) and \( H_{sel} \) be the hex powers to be generated, then the output from the hex block is of the form \( x^{16}, x^{16^2}, x^{16^4} \) (since four cascaded blocks are selected). Two buffers \( M_{out} \) and \( H_{out} \) stores the values calculated from Multiplier and hex block, respectively. Either the output of the multiplier or the output of the hex block is considered at each clock phase, depending on the \( en \) signal. The register bank is used to store the value for further use. The control block is an FSM that generates control signal based on the number of cascaded hex circuits in the hex block and the quad addition chain. At each clock cycle, the control block will generate the control signal for \( sel1, sel2, sel3, en \) and \( H_{sel} \) in the architecture, where \( sel1, sel2 \) and \( sel3 \) are the selection line for \( Mux \ A, B \) (multiplication) and \( mux \ C \) (hex operation). Table V shows the inverse computation of an element \( A \in GF(2^{233}) \).

In Table VI, during clock cycle 1, the element \( A \) is given as input to the architecture with control signal enabling \( sel1, sel2 \) and \( en = 1 \) for multiplication, that generates \( A^2 \) using the combinational multiplier after one clock cycle. Likewise, \( a_1(A) = A^{15} \) is generated after 5 clock cycle. The signal from \( RB1_{sel}, RB2_{sel} \) is the select line for register bank 1 and 2, which determines when to store the data in the register bank from the multiplier output. In the similar manner the architecture computes \( A^{-1} \) after 33 clock cycles as per Table VI.

4. Experimental results

In order to compare the proposed architecture with existing works, the results were taken on Virtex-5 and Virtex 7 FPGA platforms without pre-computation using the synthesis tool of Xilinx ISE 11.1 design software. Table VII compares the performance of proposed HITA inversion of an element \( A \in GF(2^m) \) with existing works [5, 8]. This table shows that the ATP has significant increase in ATP as compared to the existing works. As a consequence, the computation time \( T = (delay \times clockcycles) \) gets reduced substantially with an increase in performance and is given by the Eq.(12).

\[
P = \frac{1}{(No.of\ LUTs \times T)}
\]  

For \( GF(2^{233}) \) our proposed HITA requires 28 clock cycles, this in turn gives better performance when compared to [5] for the same device Virtex 5. Both the fields, \( GF(2^{233}) \) and \( GF(2^{409}) \) is also implemented in Virtex 5 and Virtex
5. Conclusion

In this express, a HITA is proposed for FF generated by NIST recommended irreducible trinomials. Subsequently, a HITA architecture has been developed for efficient implementation on FPGA platforms. The implementation results shows better Area-Time Performance, comparatively over existing techniques. This design is scalable and also can be traded-off between area and time making it suitable for resource constraint cryptographic applications.

References

[1] J.P. Deschamps, et al.: "Hardware Implementation of Finite-Field Arithmetic (McGraw-Hill Education, 2009).
[2] J. Li, et al.: "Speed-oriented architecture for binary field point multiplication on elliptic curves," IEEE Access 7 (2019) 32048 (DOI: 10.1109/ACCESS.2019.2903170).
[3] F. Rodriguez-Henriquez, et al.: "Parallel Itoh-Tsujii multiplicative inversion algorithm for a special class of trinomials," Designs, Codes and Cryptography 45 (2007) 19 (DOI: 10.1007/s10623-007-9073-6).
[4] Q. Deng, et al.: "A fast hardware implementation of multiplicative inversion in GF(2^m)," 2009 Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PrimeAsia) (2009) 472 (DOI: 10.1109/PRIMEASIA.2009.5397344).
[5] C. Rebeiro, et al.: "Revisiting the Itoh-Tsujii inversion algorithm for FPGA platforms," IEICE Trans. Very Large Scale Integration (VLSI) Syst. 19 (2011) 1508 (DOI: 10.1109/TVLSI.2010.2051343).
[6] S.S. Roy, et al.: "Generalized high speed Itoh-Tsujii multiplicative inversion architecture for FPGAs," Integration (2012) 307 (DOI: 10.1016/j.vlsi.2011.11.007).
[7] L. Parrilla, et al.: "Minimum-clock-cycle Itoh-Tsujii algorithm hardware implementation for cryptography applications over GF(2^m) fields," Electronics Letters 48 (2012) 1162 (DOI: 10.1049/el.2012.1427).
[8] V.R. Venkatasubramani, et al.: "An improved quad Itoh-Tsujii algorithm for FPGAs," IEICE Electron. Express 10 (2013) 20130612 (DOI: 10.1587/elex.10.20130612).
[9] J. Hu, et al.: "Fast and generic inversion architectures over GF(2^m) using modified Itoh-Tsujii algorithms," IEEE Trans. Circuits Syst. II, Exp. Briefs 62 (2015), 367 (DOI: 10.1109/TCSII.2014.2387612).
[10] B. Rashidi, et al.: "High-speed hardware architecture of scalar multiplication for binary elliptic curve cryptosystems," Microelectronics Journal 52 (2016) 49 (DOI: 10.1016/j.mejo.2016.03.006).
[11] H. Yi, et al.: "Fast hardware implementations of inversions in small finite fields for special irreducible polynomials on FPGAs," International Journal of Security and Its Applications 10 (2016) 109 (DOI: 10.14257/ijsia.2016.10.9).
[12] B. Rashidi, et al.: "High-performance and high-speed implementation of polynomial basis Itoh-Tsujii inversion algorithm over GF(2^m)," IET Information Security 11 (2017) 66 (DOI: 10.1049/iet-ifs.2015.0461).
[13] J. Maitin-Shepard: "Optimal software-implemented Itoh–Tsujii inversion for $\bar{F}_m$, Design Codes Cryptography 82 (2017) 301 (DOI: 10.1007/s10623-016-0260-1).
[14] B. Rashidi, et al.: "High-speed hardware implementation of Gaussian normal basis inversion algorithm over $\bar{F}_m$, Microelectronics Journal 63 (2017) 138 (DOI: 10.1016/j.mejo.2017.04.001).
[15] A. Ibrahim, et al.: "New systolic array architecture for finite field division," IEICE Electron. Express 15 (2018) 20180255 (DOI: 10.1587/elex.15.20180255).
[16] J. Guajardo, et al.: "Fast inversion in composite Galois fields $G(2^{n}\downarrow sup n)$sup $m/2$ m/2/2, Proceedings in IEEE International Symposium on Information Theory (1998), 295 (DOI: 10.1109/ISIT.1998.708900).
[17] L. Li, et al.: "Fast inversion in GF(2^m) with polynomial basis using optimal addition chains," 2017 IEEE International Symposium on Circuits and Systems (ISCAS) (2017) 1 (DOI: 10.1109/ISCAS.

Table VI Control words for GF(2^{233}) in Hex ITA from Table V

| $a_1(A)$ | clock | sel1 | sel2 | sel3 | $B_{rel}$ | $e$ |
|---------|-------|------|------|------|----------|---|
| 1       | 0     | 1    | 0    | 1    | 0        | 1 |
| 2       | 1     | 0    | 1    | 0    | 0        | 1 |
| 3       | 3     | 1    | 0    | 1    | 1        | 1 |
| 4       | 1     | 1    | 0    | 1    | 0        | 1 |
| 5       | 3     | 3    | 1    | 0    | 1        | 1 |

$\alpha_2(A)$: $\{6, x, x, 0, 0, 0\}$

| $\alpha_3(A)$ | $\{7, x, 1, 2, x, 1\}$ |
| $\alpha_4(A)$ | $\{8, x, x, 0, 0, 2\}$ |
| $\alpha_5(A)$ | $\{9, 1, 2, x, x, 1\}$ |

| $\alpha_6(A)$ | $\{10, x, x, 1, 1, x\}$ |
| $\alpha_7(A)$ | $\{11, 3, 2, 0, x, 1\}$ |
| $\alpha_8(A)$ | $\{12, x, x, 1, 0, 3\}$ |
| $\alpha_9(A)$ | $\{13, 3, 2, x, x, 1\}$ |
| $\alpha_{10}(A)$ | $\{14, x, x, 0, 4, 0\}$ |
| $\alpha_{11}(A)$ | $\{15, x, x, 1, 1, 0\}$ |
| $\alpha_{12}(A)$ | $\{16, 3, 3, x, 2, 1\}$ |

Table VII Comparison of the proposed architecture and other works

| Implementaton | Resources utilized (Slices, BRAM, GCLK) | C | $T$ (MHz) | $P$ |
|---------------|----------------------------------------|---|----------|---|
| Sequential [90] | 1000S:12.3 | 21.2 | 28 | 1.32 | 78.2 |
| Parallel [3] | 11081:12.2 | 21.2 | 20 | 0.94 | 95.7 |
| Quad-ITA [5] | 10420.02 | 35 | 21 | 0.60 | 160 |
| Modified QITA [8] | 10190.0 | 37 | 20 | 0.54 | 181 |
| Proposed HITA | 10195.01 | 38.7 | 18 | 0.465 | 214.6 |

The data was tabulated in Table VII with better performance. The increase in performance is obtained due to decrease in computation time and significant decrease in LUTs.

Table VIII Comparison for inversion in GF(2^{233}) on XC3V2000 FPGA.

| Implementation | Resources utilized (Slices, BRAM, GCLK) | C | $T$ (MHz) | $P$ |
|---------------|----------------------------------------|---|----------|---|
| Sequential [90] | 1000S:12.3 | 21.2 | 28 | 1.32 | 78.2 |
| Parallel [3] | 11081:12.2 | 21.2 | 20 | 0.94 | 95.7 |
| Quad-ITA [5] | 10420.02 | 35 | 21 | 0.60 | 160 |
| Modified QITA [8] | 10190.0 | 37 | 20 | 0.54 | 181 |
| Proposed HITA | 10195.01 | 38.7 | 18 | 0.465 | 214.6 |

The results were tabulated in Table VII with better performance. The increase in performance is obtained due to decrease in computation time and significant decrease in LUTs. Table VIII compares the proposed work in the same field and FPGA platform with [3, 5, 8, 30]. The table shows that proposed implementation is superior in the computation time utilizing less resources with no BRAM required. There is 34.125% and 18.5% increase in performance as compared to [5] and [8].
[18] P. Zode, et al.: “Fast architecture of modular inversion using Itoh-Tsujii algorithm,” International Symposium on VLSI Design and Test (2017) (DOI: 10.1007/978-981-10-7470-7_5).

[19] J. Li, et al.: “A fast modular inversion FPGA implementation over GF(2^n) using modified x^{2n} unit,” IEEE International Symposium on Circuits and Systems (ISCAS) (2018) 1 (DOI: 10.1109/ISCAS.2018.8351614).

[20] M. Imran, et al.: “Lopez Dahab based elliptic crypto processor (ECP) over GF(2^{165}) for low-area applications on FPGA,” International Conference on Engineering and Emerging Technologies (ICEET), (2018) 1 (DOI: 10.1109/ICEET1.2018.8338645).

[21] M. Kalaiarasi, et al.: “A parallel quad Itoh-Tsujii multiplicative inversion algorithm for FPGA platforms,” Third ISEA Conference on Security and Privacy (ISEA-ISAP) (2020) 31 (DOI: 10.1109/ISEA-ISAP49340.2020.234996).

[22] M. Kalaiarasi, et al.: “High-speed inversion using x^{4n} units,” Advances in Communication Systems and Networks, Lecture Notes in Electrical Engineering 656 (Springer, Singapore, 2020) (DOI: 10.1007/978-981-15-3992-3_50).

[23] B. Rashidi, et al.: “Efficient hardware structure for extended Euclidean-based inversion over GF(2^n),” IET Computers & Digital Techniques 13 (2019) 282 (DOI: 10.1049/iet-cdt.2018.5077).

[24] T. Itoh and S. Tsujii: “A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases.” Information and Computation 78 (1988) 171 (DOI: 10.1016/0890-5401(88)90024-7).

[25] E. Knuth: The Art of Computer Programming, Addison-Wesley, Boston, MA, 1998.

[26] A. Reyhani-Masoleh, et al.: “New multiplicative inverse architectures using Gaussian normal basis,” IEEE Trans. Comput. 68 (2019) 991 (DOI: 10.1109/TC.2018.2859941).

[27] A. Monfared, et al.: “A new multiplicative inverse architecture in normal basis using novel concurrent serial squaring and multiplication,” 2017 IEEE 24th Symposium on Computer Arithmetic (ARITH) (2017) 164 (DOI: 10.1109/ARITH.2017.37).

[28] J. Guajardo, et al.: “Itoh-Tsujii inversion IN STANDARD BASIS AND ITS APPLICATION IN CRYPTOGRAPHY AND CODES,” DESIGN Codes and CRYPTOGRAPHY, 25, (2002) 207 (DOI: 10.1023/A:1013860532636).

[29] D. Hankerson, et al.: guide to elliptic curve cryptography (Springer Science & Business Media, 2006).

[30] F. Rodriguez-Henriquez, et al.: “A fast implementation of multiplicative inversion over GF(2^m),” International Conference on Information Technology: Coding and Computing (ITCC’05) (2005) (DOI: 10.1109/itcc.2005.8).