Theoretical Modeling of the Itoh-Tsujii Inversion Algorithm for Enhanced Performance on k-LUT based FPGAs

Sujoy Sinha Roy, Chester Rebeiro and Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
email:{sujoyetc, chester, debdeep}@cse.iitkgp.ernet.in

***Abstract*—Maximizing the performance of the Itoh-Tsujii finite field inversion algorithm (ITA) on FPGAs requires tuning of several design parameters. This is often time consuming and difficult. This paper presents a theoretical model for the ITA for any Galois field and k-input LUT based FPGA ($k > 3$). Such a model would aid a hardware designer to select the ideal design parameters quickly. The model is experimentally validated with the NIST specified fields and with 4 and 6 LUT based FPGAs. Finally, it is demonstrated that the resultant designs of the Itoh-Tsujii Inversion algorithm is most optimized among contemporary works on LUT based FPGAs.**

I. Introduction

Hardware accelerators for public key cryptography have a huge area footprint due to the arithmetic on large numbers generally used in cypto-algorithms. It is therefore essential to design accelerators that not only compute at high-speeds, but also efficiently utilize the available FPGA resources. Of all arithmetic operations used in crypto-algorithms, finite field inversion is the most complex. The *Itoh-Tsujii algorithm* (ITA) [1] is generally used for computing the inversion due to its efficieny [3]. The ITA computes the inverse by a series of multiplications and exponentiations along an addition chain. The operations are in the finite field and the exponentiations are done by power circuits, which raises the input to the power 2^n. The performance of an ITA architecture on an FPGA depends on several configurable design parameters such as the choice of addition chain, power circuits, and the number of replicated circuits. These parameters have to be selected appropriately to achieve best performance. While the selection of the addition chain is straight forward[8], selection of the other parameters is not. In the current FPGA design scenario, the only way to obtain the best configuration of the parameters is to physically evaluate every possible design strategy and then select the best performing strategy. This results in long design cycles.

This paper presents a theoretical approach to estimate the best strategy for any $k \geq 4$ input LUT based FPGA, thus reducing the design cycle time. The proposed approach uses a theoretical model of the ITA design to analyze how various design strategies affect the area, delay, and clock cycle requirements in the FPGAs. For a given field size (m), size of an LUT in the FPGA (k), and addition chain, our model predicts the best power circuit and the ideal number of replicated circuits which would give peak performance. The theoretical results are experimentally validated on 4 and 6 input LUT based FPGAs for fields specifed in NIST's Digital Signature Standard[9].

The paper is structured as follows: Section II has a brief introduction to the ITA and related works. The ITA is then generalized to use any exponentiation circuit in Section III. An ITA architecture is described in Section IV. Section V gives an estimate of the number of clocks required for this design. A theoretical analysis of area and delay for this ITA architecture is presented in Section VI. Section VII describes how design parameters can be tuned to get maximum performance. Theoretical estimates are validated on NIST fields[9] in VIII. This section also contains performance comparisons with existing works. The final section has the conclusion.

II. Preliminaries

For an element $a \neq 0$ in the field $GF(2^m)$, the inverse of a is the element $a^{-1} \in GF(2^m)$ such that $a \cdot a^{-1} = a^{-1} \cdot a = 1$. The Itoh-Tsujii inversion algorithm was proposed in [1] for normal bases representations and in [2] for polynomial basis representations. The ITA works as follows: For $a \in GF(2^m)$, let $\beta_k(a) = a^{2^k-1}$ and $k \in N$. Then, by Fermat's little theorem, $a^{-1} = \beta_{m-1}(a)^2$. For simplicity, we denote $\beta_k(a)$ by β_k. In [5], a recursive sequence (Equation 1) is used with an addition chain for $m-1$ to compute β_{m-1}.

$$\beta_{k+j}(a) = (\beta_j)^{2^k}\beta_k = (\beta_k)^{2^j}\beta_j \tag{1}$$

In [6], further speed up was obtained by a parallel implementation of the algorithm. In [7] and [8] a quad circuit was used for ITA implementations in 4 input LUT based FPGAs. The quad circuit is just 1.5 times the size of the squarer (instead of twice), and uses an addition chain for $(m-1)/2$ instead of $m-1$. So, in quad ITA, clock cycles are saved with minimal impact in the area.

In this paper we denote the operation b^{2^n} as *powering* b *by* n, where $b \in GF(2^m)$. In [3], [5], and [6] powering is done using squarer circuits $(n = 1)$. In [7] and [8], a powering by 2 circuit was used.

978-3-9810801-7-9/DATE11/©2011 EDAA

III. Generalization of the ITA for any Exponentiation Circuit

Theorems 1 and 2 show that the ITA algorithm can be extended to use any 2^n circuit and not just squarers or quads [2]. For any $a \in GF(2^m)$ and natural number k, define $\alpha_k(a) = a^{2^{nk}-1}$.

Theorem 1 If $a \in GF(2^m)$, $\alpha_{k_1}(a) = a^{2^{nk_1}-1}$ and $\alpha_{k_2}(a) = a^{2^{nk_2}-1}$ then

$$\alpha_{k_1+k_2}(a) = (\alpha_{k_1}(a))^{2^{nk_2}} \alpha_{k_2}(a)$$

where $k1,\ k2\ and\ n \in N$.

Theorem 2 The inverse of an element $a \in GF(2^m)$ is

$$a^{-1} = \begin{cases} \{\alpha_{\frac{m-1}{n}}(a)\}^2 & \text{if } n \mid (m-1); \\ \{(\alpha_q(a))^{2^r} \beta_r(a)\}^2 & \text{if } n \not\mid (m-1). \end{cases}$$

where $nq + r = m - 1\ and\ n,\ q,\ and\ r \in N$.

$\alpha_q(a)$ is calculated using an addition chain for q. A 2^n circuit based ITA is presented in Algorithm 1.

Algorithm 1 genita (2^n based ITA)

Require: An element $a \in GF(2^m)$ and addition chain $U = \{u_i\}$ for q

$$U = \{1, 2, \ldots, q\}$$

Ensure: $a^{-1} \in GF(2^m)$ such that $a^{-1} \cdot a = 1$

1: **begin**
2: $l = length\ of\ U$
3: $\alpha_{u_1} = a^{2^n-1}$
4: $\beta_r = a^{2^r-1}$
5: **for** $i = 2$ to l **do**
6: $\quad p = u_{i-1}$
7: $\quad q = u_i - u_{i-1}$
8: $\quad \alpha_{u_i} = \alpha_q * \alpha_p^{2^{nq}}$
9: **end for**
10: $\alpha_{u_l} = \alpha_{u_l}^{2^r} \cdot \beta_r$
11: $a^{-1} = \alpha_{u_l} * \alpha_{u_l}$
12: **return** a^{-1}
13: **end**

An overhead of using a 2^n circuit is the requirement to compute $\alpha_1 = a^{2^n-1} = a \cdot a^2 \cdot a^{2^2} \cdots a^{2^{n-1}}$. This can be done using an addition chain for $2^n - 1$. Such a technique requires $l' - 1$ clock cycles, where l' is the length of addition chain for $2^n - 1$.

Another overhead occurs when $n \not\mid (m-1)$. In this case $\beta_r(a) = a^{2^r-1}$, where $1 \le r \le (n-1)$, is required to be computed. Since $1 \le r \le (n-1)$, computation of $\beta_r(a)$ can be done during α_1 computation using an addition chain for $2^n - 1$ which contains r.

IV. 2^n-ITA Architecture for a Combinational Multiplier

The quad based ITA architecture in [8] was shown to have best performance compared to other architectures, therefore we design our 2^n circuit based ITA as a generalization of this architecture.

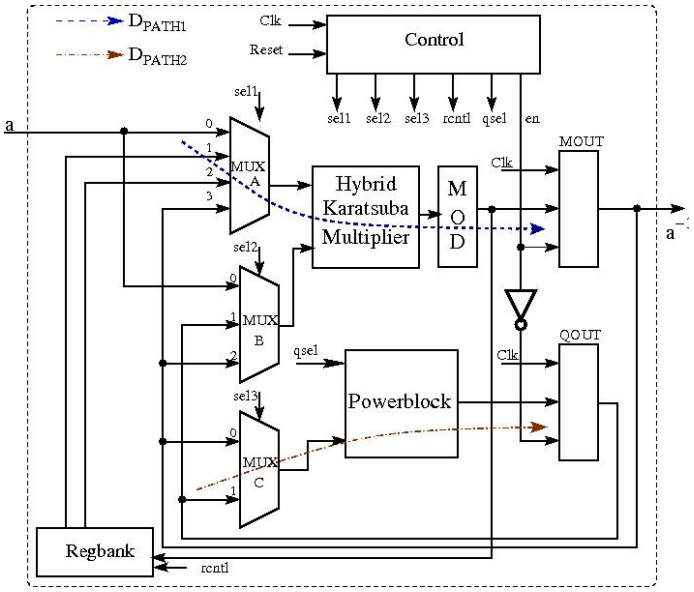

Fig. 1. FPGA Architecture for 2^n ITA

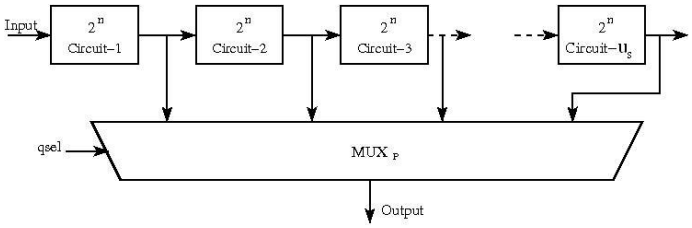

Fig. 2. Powerblock Design with cascaded 2^n circuits

The only difference between the two architectures is in the *powerblock*, which contains 2^n circuits instead of quads. We now give a brief description of the 2^n based ITA architecture (Figure 1). The finite field multiplier used is of combinational type and follows the *hybrid Karatsuba algorithm* [10]. The powerblock is used to raise the power of its input a to any power of 2 (ie. a^{2^i} for some positive integer i). The control signal for the powerblock is *qsel*. Buffers $MOUT$ and $QOUT$ are used to latch the output of the multiplier and the powerblock respectively. Control signal *en* is used in each clock cycle to latch the output of either the multiplier or the powerblock (but not both). Any intermediate result that is required in a later step is stored in a register bank. A control block is used to generate all the control signals that drive a Finite State Machine (FSM).

The performance of the architecture as a function of the area, delay, and number of clock cycles required, mainly depends on the multiplier and the powerblock. The design parameters of these blocks have to be tuned in order to achieve optimal performance. In the next sections we show how estimations can be made to achieve this.

V. Clock Cycle Requirement for 2^n Circuit based ITA

Computing α_{u_i}, $2 \le i \le l$ in line 8 of Algorithm 1 requires a powering of α_p by 2^q, followed by a multiplication by α_q.

Computing $\alpha_{u_p}^{2^q}$ is done using the powerblock in Figure 1. This computation requires $u_i - u_{i-1}$ (where u_i and u_{i-1} is an element in the addition chain for q) exponentiations. Using a single 2^n circuit will take $u_i - u_{i-1}$ clock cycles. The number of clock cycles is reduced by using a cascade of u_s number of 2^n circuits (Figure 2), where the output of one 2^n circuit is fed to the input of the next. If the number of repeated powering is less than u_s, a multiplexer is used to tap out interim outputs. Considering u_s number of cascaded 2^n circuits, $\lceil \frac{u_i - u_{i-1}}{u_s} \rceil$ number of clock cycles are required for computing $\alpha_p^{2^{nq}}$. For the multiplication with α_q, a single clock cycle is required (assuming a combinational multiplier). Additionally, $l' - 1$ clock cycles are required to compute α_1, where l' is the length of the addition chain for $2^n - 1$. The final squaring requires a clock cycle, and an extra clock cycle is used to indicate completion of the field inversion. In all, the total number of clock cycles required when $n \mid (m-1)$ is

$$\#ClockCycles = l^{'} + l + \sum_{i=2}^{l} \left\lceil \frac{u_i - u_{i-1}}{u_s} \right\rceil \quad (2)$$

When $n \not\mid (m-1)$, r additional clock cycles are required for $(\alpha_q(a))^{2^r}$ calculation and one clock cycle for $(\alpha_q(a))^{2^r}\beta_r(a)$ calculation (line 10 in Algorithm 1). Thus, the total clock cycle requirement for $n \not\mid (m-1)$ is

$$\#ClockCycles = l^{'} + l + 1 + r + \sum_{i=2}^{l} \left\lceil \frac{u_i - u_{i-1}}{u_s} \right\rceil \quad (3)$$

VI. Area and Delay Estimations for the 2^n Circuit based ITA Architecture

In this section we give a theoretical estimation of the total area and delay requirement in terms of k input LUTs.

A. LUT requirements for a function of x variables:

A k input LUT (k-LUT) can be considered a black box that can perform any functionality of a maximum of k variables. If there is a single variable, then no LUT is required. If there are more than k variables, then more than one $k-$LUTs are required to implement the functionality. The total number of k input LUTs for a function with x variables can be expressed as,

$$lut(x) = \begin{cases} 0 & \text{if } x \leq 1 \\ 1 & \text{if } 1 < x \leq k \\ \lfloor \frac{x-k}{k-1} \rfloor + 2 & \text{if } x > k \text{ and } (k-1) \not\mid (x-k) \\ \frac{x-k}{k-1} + 1 & \text{if } x > k \text{ and } (k-1) \mid (x-k) \end{cases} \quad (4)$$

B. Delay in LUTs for a function of x variables:

Delay in FPGAs comprise of LUT delays and routing delays. By experimentation we found that the total delay in the ITA design is proportional to the number of LUTs in the critical path. That is, we found that more number of LUTs imply longer delays. We therefore make the assumption that the critical path has the most number of LUTs. We denote this number of LUTs as *maxlutpath* and use this parameter to compare delays between designs.

We now present a theoretical estimation of *maxlutpath* of a function with x variables considering k-LUT based FPGAs. The *maxlutpath* of an x variable function depends on the number of inputs to the LUT and is given by,

$$maxlutpath(x) = \lceil log_k(x) \rceil. \quad (5)$$

C. LUT and Delay estimates for the Multiplier:

ITA implementations in binary fields require a polynomial multiplier. For hardware implementations, the *Karatsuba* multiplier is the fastest[3]. The hybrid Karatsuba multiplier [10] is shown to have the best performance for FPGA based implementations. An area estimation for the hybrid Karatsuba multiplier is presented in [10]. Here we show much tighter area estimations for the hybrid Karatsuba multiplier.

An m bit hybrid Karatsuba multiplier consists of two $\lceil \frac{m}{2} \rceil$ bit and one $\lfloor \frac{m}{2} \rfloor$ bit multipliers. Similarly each such lower order multiplier is split into three smaller multiplications. If the number of bits of the multiplicands is less than a threshold (τ), the School-Book multiplication is invoked.

We have experimentally found that a threshold of 16 is best suited for both 4 and 6 and input LUTs. For multiplications with multiplicands having greater than 16 bits, the simple Karatsuba multiplier is used. For m bit multiplicands, the simple Karatsuba multiplier consumes $(2m-1)$ LUTs (assuming $k \geq 4$) to combine the output of the smaller multiplications. The total LUT requirement of the multiplier is given by the recursive formula:

$$\begin{aligned} \#LUT_{hkmul}(m) &= 2 \times LUT_{hkmul}(\lceil \frac{m}{2} \rceil) + \\ & \quad LUT_{hkmul}(\lfloor \frac{m}{2} \rfloor) + 2m - 1 \end{aligned} \quad (6)$$

The total number of LUTs for a τ bit School-Book multiplier is given by,

$$\#LUT_{sbmul} = 2 \sum_{i=1}^{\tau-1} lut(2i) + lut(2\tau) \quad (7)$$

and delay in terms of LUTs is given by,

$$\begin{aligned} \#D_{sbmul} &= maxlutpath(2\tau) \\ &= \lceil log_k(2\tau) \rceil \end{aligned} \quad (8)$$

The height of the hybrid Karatsuba multiplier is $\lceil log_2(\frac{m}{\tau}) \rceil$. Delay of the entire hybrid Karatsuba multiplier in terms of LUTs is given by,

$$\begin{aligned} \#D_{hkmul} &= height\ of\ tree + D_{sbmul} \\ &= \lceil log_2(\frac{m}{\tau}) \rceil + \lceil log_k(\tau) \rceil \end{aligned} \quad (9)$$

D. LUT and Delay estimates for the Reduction Circuit:

The output of an m bit multiplier has $2m-1$ bits and is fed to a modular reduction circuit to convert it into an m bit output. We denote the total number of LUTs in the modular reduction circuit by LUT_{Mod} and delay in terms of LUTs by D_{Mod}. For fields generated by irreducible trinomials, total number of $k-$LUTs for this reduction circuit is almost equal to the field size m (considering $k \geq 4$) and the *maxlutpath* is 1. For fields generated by irreducible pentanomials, total number of $k-$LUTs of the reduction circuit is almost twice the field size m for $4 \leq k < 6$ and is almost equal to m for $k \geq 6$, while the *maxlutpath* is 2.

E. LUT and Delay estimates of a 2^n Circuit:

The output of a squarer circuit in $GF(2^m)$ is given by $d = A \cdot a$, where A is an $m \times m$ binary square matrix and a is the input column matrix $[a_0\ a_1\ \cdots\ a_{m-1}]^T$. Output d is the column matrix $[d_0\ d_1\ \cdots\ d_{m-1}]^T$. The matrix A depends on the irreducible polynomial and m. Similarly, the output of a 2^n circuit which raises an input $a \in GF(2^m)$ to a^{2^n}, can be expressed as $d = A^n \cdot a$ [2]. A particular output bit d_i is the XOR of some of the elements in the column matrix a. We use a program to get equations of all the m output bits for the 2^n circuit and then using Equation (4), we obtain the total LUT requirement per output bit d_i for $0 \leq i \leq m-1$. The total LUT requirement of the entire 2^n circuit is given by,

$$\#LUT_{2^n} = \sum_{i=0}^{m-1} lut(d_i).$$

Similarly, Equation (5) gives the delay per output bit d_i in terms of LUTs. Since all output bits are computed in parallel, the delay of the 2^n circuit is the maximum LUT delays of all the d_i output bits and is given by,

$$\#D_{2^n} = Max(LUT\ delay\ of\ d_i) \quad for\ 0 \leq i \leq m-1.$$

F. LUT and Delay estimates of a Multiplexer:

For a $2^s : 1$ MUX, there are s selection lines, thus the output of a 2^s input MUX is a function of $2^s + s$ variables. So the total LUT requirements to implement the output functionality is $lut(2^s+s)$. For $GF(2^m)$, each input line to the MUX has m bits and the output has m bits. Thus the total LUT requirement for a 2^s input MUX, is given by

$$\#LUT_{MUX} = m \times lut(2^s + s). \tag{10}$$

Delay of the MUX in terms of LUTs is equal to the *maxlutpath* of $2^s + s$ variables and is given by,

$$\#D_{MUX} = maxlutpath(2^s + s). \tag{11}$$

If $2^{s-1} < number\ of\ inputs < 2^s$, then estimations in (10) and (11) for 2^s inputs give an upper bound. Practically, the values in this case are slightly lesser than the values for 2^s inputs in (10) and (11), and therefore the difference can be neglected.

G. LUT and Delay estimates for the Powerblock:

Let the powerblock contain u_s number of cascaded 2^n circuits and the multiplexer MUX_P. The multiplexer is controlled by r number of selection lines where $2^{r-1} < u_s \leq 2^r$. Thus number of k-LUTs in MUX_P is given by,

$$\#LUT_{MUX_P} = m \times lut(2^r + r)$$

Total number of LUTs in the powerblock is,

$$\#LUT_{Powblk} = u_s \times LUT_{2^n} + LUT_{MUX_P}$$

Using Equations (5) and (11), delay of MUX_P in terms of LUTs is,

$$\#D_{MUX_P} = \lceil log_k(u_s + log_2 u_s) \rceil \tag{12}$$

Since, there are u_s number of cascaded 2^n circuits, the delay of the entire cascade is u_s times the delay of a single 2^n circuit. The delay of the multiplexer in the powerblock is added to this delay to get the total delay of the powerblock.

$$\begin{aligned} \#D_{Powblk} &= u_s \times D_{2^n} + D_{MUX_P} \\ &= u_s \times D_{2^n} + \lceil log_k(u_s + log_2 u_s) \rceil. \end{aligned} \tag{13}$$

H. Total LUT estimate of the entire ITA architecture:

From Figure (1), it can be seen that the total number of LUTs in the ITA architecture is the sum of the LUTs in the multiplier, reduction block, powerblock, MUX_A, MUX_B, and MUX_C. The state machine (control block in Figure 1) consumes very few LUTs and is not considered in the overall LUT count. Thus the total number of $k-$LUTs in the entire ITA is,

$$\begin{aligned} \#LUT_{ITA} &= LUT_{Multiplier} + LUT_{Mod} + LUT_{Powblk} + \\ &\quad LUT_{MUX_A} + LUT_{MUX_B} + LUT_{MUX_C} \end{aligned}$$

I. Total Delay estimate of the entire ITA architecture:

In Figure (1), there are two parallel paths. The delay of the first path (D_{PATH1}) is through MUX_A, multiplier, and reduction block. The delay of second path (D_{PATH2}) is through MUX_C and the $Powerblock$. The delay of the entire ITA architecture is equal to the maximum of D_{PATH1} and D_{PATH2}. Using Equations (9) and (11), D_{PATH1} can be approximated in terms of LUTs as

$$D_{PATH1} \approx \lceil \frac{6}{k} \rceil + \lceil log_2(\frac{m}{\tau}) \rceil + \lceil log_k(\tau) \rceil + D_{Mod}$$

,where D_{Mod} has values of 1 and 2 for trinomials and pentanomials respectively. Similarly, D_{PATH2} can be approximated using Equation (11) as

$$\begin{aligned} D_{PATH2} &= D_{MUX_C} + D_{Powblk} \\ &\approx 1 + u_s \times D_{2^n} + \lceil log_k(u_s + log_2(u_s)) \rceil \end{aligned}$$

A finite state machine drives the control signals. Each control signal depends only on the value of the state variable and nothing else. Therefore it can be assumed that the delay in generating the control signals depends on the number of bits in the $state$ register. The $state$ variable increments up to the value of $\#ClockCycles$. Assuming a binary counter, the number of bits in the $state$ variable is $log_2(\#ClockCycles)$. The delay in the control signals is

$$D_{Cntrl} = maxlutpath(log_2(\#ClockCycles))$$

This delay is added to the path delay. The total delay in the ITA in terms of LUTs is

$$\#D_{ITA} = Max(D_{PATH1}, D_{PATH2}) + D_{Cntrl} \tag{14}$$

VII. Obtaining the Optimal performing ITA architecture

We measure the performance of the ITA hardware by the metric

$$Performance = \frac{1}{(LUT_{ITA} \times D_{ITA} \times ClockCycles)}$$

For a given field $GF(2^m)$ and k-LUT based FPGA, the powerblock can be configured with different power circuits (2^n) and number of cascades (u_s). An increase in u_s, decreases the clock cycles required at the cost of an increase in area and delay through the powerblock. To get maximum performance, u_s should be chosen in such a way that it minimizes clock cycles without increasing the delay (D_{ITA}) and area (LUT_{ITA}) significantly. D_{PATH1} is not dependent on n or u_s and is constant. From Equation (14), it follows that D_{ITA} is minimum when

$$D_{PATH1} \approx D_{PATH2}. \tag{15}$$

When Equation (15) is satisfied, we can assume that D_{ITA} in Equation (14) is

$$\#D_{ITA} = D_{PATH1} + D_{Cntrl}.$$

D_{Cntrl} has a small value and is less dependent on n and u_s. Since D_{PATH1} is constant, D_{ITA} can be considered to be a constant when Equation(15) is satisfied.

In Equation of LUT_{ITA}, the multiplier consumes a significant portion of total area. The area of a 2^n circuit is small. So, variation of n and u_s under Equation (15) causes negligible variation in total area. Thus, we can assume that LUT_{ITA} remains almost unaffected when Equation (15) is satisfied.

The $\#ClockCycles$ changes significantly with n and u_s. Clock cycle requirement in Equations (2) and (3) can be approximated as ,

$$Clock\tilde{C}ycles \approx (2n-1) + l + r + \lceil \frac{m-1}{nu_s} \rceil \tag{16}$$

where $2n-1$ is the clock cycle requirement for α_1 computation and l is the length of addition chain for $\lfloor \frac{m-1}{n} \rfloor$. From Equation (16), it can be seen that with increase in n, α_1 computation increases linearly with n, but the term $\lceil \frac{m-1}{nu_s} \rceil$ reduces. When n is small, $\lceil \frac{m-1}{nu_s} \rceil$ is significant compared to $2n-1$. So, $Clock\tilde{C}ycles$ reduces significantly with increase in n. For large values of n, the term $(2n-1)$ dominates the term $\lceil \frac{m-1}{nu_s} \rceil$. So, $Clock\tilde{C}ycles$ increases monotonically with n. This implies that we could restrict n to those values for which we get savings in $\#ClockCycles$. This effect of increase in n on clock cycles for α_1 computation and on $\#ClockCycles$ is shown in Figure (3) for the field $GF(2^{233})$.

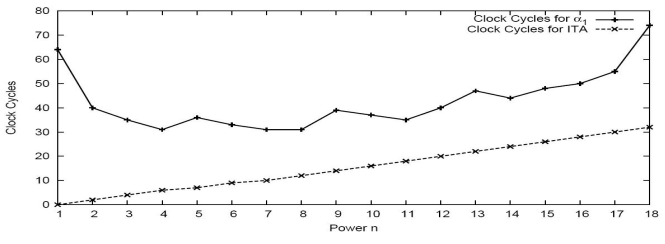

Fig. 3. Number of Clock Cycles with power n

Minimization of $\#ClockCycles$ without increasing the D_{ITA} leads to maximum performance. The following algorithm is used to obtain best design parameters.

1) Starting from the squarer ($n = 1$) circuits, estimate u_s so that Equation (15) is achieved. Plug in the values of n and u_s in Equation (16) and get $Clock\tilde{C}ycles$. For $n = 1$, we call it $Clock\tilde{C}ycles_1$.
2) Now, for a higher value of n say $n_i = i$, where $i > 1$, the number of cascades u_{s_i} is determined as in Step 1 and the corresponding value of $Clock\tilde{C}ycles$ (say $Clock\tilde{C}ycles_i$ is obtained.
3) The second step is repeated for incremental values of n_i until $Clock\tilde{C}ycles_i$ increases monotonically (as per the previous discussion) as n_i increases.

The best design strategy for the powerblock is the configuration $(\hat{n}, \hat{u_s})$ which gives the minimum $Clock\tilde{C}ycles$. This is given by

$$(\hat{n}, \hat{u_s}) = \underset{(n_i, u_{s_i})}{\operatorname{argmin}} (Clock\tilde{C}ycles_i) \qquad \text{For } i \geq 1.$$

For example, when $m = 233$ and $k = 4$, $(2, 8)$ and $(4, 5)$ give minimum value of $Clock\tilde{C}ycles$ and satisfy Equation (15) closely. In the next section we support our theoretical estimates with experimental results.

N	Theoretical estimation					Experimental values			
	Clock Cycle	LUT	LUT Delay	Total Delay (ns)	P	Clock Cycle	LUT	Total Delay (ns)	P
7	35	19523	13	13.3	110	35	21049	13.5	100
8	32	19753	14	13.5	117	32	21291	13.47	109
10	30	20912	16	13.9	114	30	23926	13.72	102
12	29	21372	18	14.3	113	29	24220	14.28	100

TABLE I
THEORETICAL AND EXPERIMENTAL COMPARISON FOR DIFFERENT NUMBER OF CASCADES OF A 2^2 BASED ITA IN THE FIELD $GF(2^{233})$

VIII. VALIDATION OF THEORETICAL ESTIMATIONS

For validation of the theoretical estimates, experiments were done on Xilinx Virtex IV and Virtex V FPGAs using ISE version 10.1. Our estimation model used *maxlutpath* to estimate delays. In FPGAs, the delays are difficult to characterize. The delay of an LUT and routing delay depends on the device technology and CAD tool. To estimate FPGA delay of any ITA architecture for a field $GF(2^m)$, we do the following. We first synthesize a squarer based ITA architecture (which we call '*reference architecture*') and obtain its delay. We denote this delay as '*reference delay*'. Using our estimation model, we estimate the *maxlutpath* of the reference architecture (which we call *reference maxlutpath*). It was found experimentally that the FPGA delay of the ITA architecture varies proportionally with the *maxlutpath*. Therefore, for ITA designs in the same field as the reference architecture, we can consider the variation in FPGA delay to be proportional to *maxlutpath* variation. This is given by

$$reference\ delay + \mathcal{C} \times (maxlutpath - reference\ maxlutpath)$$

Here $\mathcal{C}$ is a constant which depends on the device technology. In our experiments we used *4vlx80ff1148-11* and *xc5vlx220-2-ff1760* FPGAs, which have 4-LUT and 6-LUT respectively. For the 4−LUT FPGA, we experimentally found $\mathcal{C} = 0.2$, while the 6-LUT had $\mathcal{C} = 0.1$.

Table (I) shows comparisons between the theoretical and experimental results for quad based ITA in $GF(2^{233})$ with different number of cascaded quad blocks for 4-LUT based

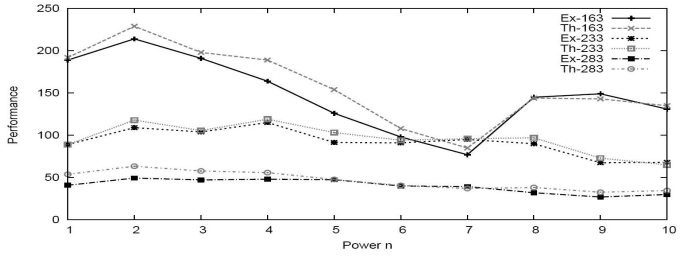

Fig. 4. Performance variation with power n in 4-LUT based FPGA

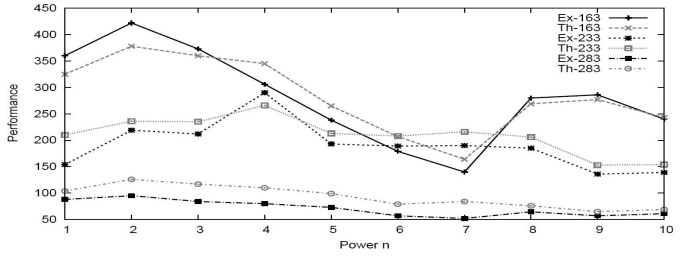

Fig. 5. Performance variation with power n in 6-LUT based FPGA

FPGAs. In all Tables, performance is denoted by P. The table (I) shows that our estimation of $\hat{u_s} = 8$ gives best performance for $\hat{n} = 2$. Graphs in Figures (4) and (5) show performance variations with powering in 4 and 6−LUT based FPGAs for the trinomial field $GF(2^{233})$ and pentanomial field $GF(2^{163})$ and $GF(2^{283})$ [9]. In the graphs 'Ex P' implies performance obtained by experiments and 'Th P' implies theoretical estimates. From the graphs, it can be seen that experimental and theoretical performances follow the same trend. The small differences occur due to unpredictablity of FPGA routing.

Table (II) shows the best performing configurations for some of the NIST recommended binary fields [9] and $GF(2^{193})$. The power circuit used (ITA type) and the number of cascades in the table result in best performance for the respective field. It can be seen that in most cases powers higher than quad give best results.

Experimental and theoretical results show that performance of the ITA architecture increases significantly in 6-LUT based FPGAs compared to 4-LUT based FPGAs. This happens due to lesser LUT requirement and better routing in 6-LUT based FPGAs.

Table (III) shows a comparison of performances with existing ITA architectures for the field $GF(2^{193})$. All results are taken on the same Xilinx Virtex E platform. Our design is a 2^3 (octate) based ITA with 8 power circuits, since for $GF(2^{193})$ on Virtex E, our estimation model found $(\hat{n} = 3, \hat{u_s} = 8)$. It is clear from the table that 2^3 based ITA gives the best performance.

IX. Conclusion

This paper develops a theoretical model of the Itoh-Tsujii algorithm in order to obtain the best performance of the algorithm on FPGA platforms. The model predicts that a proper selection of power circuits (n) and number of cascades (u_s) in the power block leads to maximum performance.

Experimental Results in Virtex IV						
Field	ITA Type	Number of Cascades	Total LUT	Total Delay (ns)	Clock Cycle	P
163	2^2	7	12260	13.58	28	214
193	2^3	6	15043	12.9	26	198
233	2^4	5	21464	13.57	30	114
283	2^3	6	31802	14.82	43	49
Experimental Results in Virtex V						
Field	ITA Type	Number of Cascades	Total LUT	Total Delay (ns)	Clock Cycle	P
163	2^2	5	8692	8.79	31	422
193	2^3	8	11254	8.91	22	453
233	2^4	7	13962	9.16	27	290
283	2^2	8	27509	10.09	38	95

TABLE II
Experimental Performance of ITA for Different Fields

Implementation	Resources Utilized (Slices, Brams)	Freq (MHz) (f)	Clock Cycle (c)	Time μsec (c/f)	P
Sequential [5]	10065, 12	21.2	28	1.32	75.2
Parallel [6]	11081, 12	21.2	20	0.94	95.7
Quad-ITA [8]	10420, 0	35	21	0.60	160
Octate-ITA (our design)	8401, 0	31.9	22	0.69	172

TABLE III
Comparison for inversion in $GF(2^{193})$ on $XCV3200efg1156$

Further an algorithm is presented by which the ideal choice of (n, u_s) can be theoretically estimated. The proposed algorithm is experimentally validated on NIST specified fields and results show a close match. A comparison with related works show significant improvements in performance when our approach is followed.

References

[1] T. Itoh and S. Tsujii, "A Fast Algorithm For Computing Multiplicative Inverses in GF(2^m) Using Normal Bases," Inf. Comput., vol. 78, no. 3, pp. 171-177, 1988.

[2] J. Guajardo and C. Paar, "Itoh-Tsujii Inversion in Standard Basis and Its Application in Cryptography and Codes" Des. Codes Cryptography, vol. 25, no. 2, pp. 207-216, 2002.

[3] F. Rodríguez-Henríquez, N.A. Saqib, A. Diaz-Perez, Ç.K. Koç, "Cryptographic Algorithms on Reconfigurable Hardware", Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] A. Weinerskirch and C. Paar,"Generalizations of the Karatsuba Algorithms for Efficient Implementations", Cryptology ePrint Archive, Report 2006/224, 2006.

[5] F. Rodríguez-Henríquez, N. A. Saqib, and N. Cruz-Cortés, "A Fast Implementation of Multiplicative Inversion Over $GF(2^m)$," in ITCC 05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC05) - Volume I, IEEE Computer Society, 2005, pp. 574-579.

[6] F. Rodríguez-Henríquez, G. Morales-Luna, N. A. Saqib, and N. Cruz-Cortés, "Parallel Itoh-Tsujii Multiplicative Inversion Algorithm for a Special Class of Trinomials," Des. Codes Cryptography, vol. 45, no. 1, pp. 19-37, 2007.

[7] C. Rebeiro and D. Mukhopadhyay, "High Speed Compact Elliptic Curve Cryptoprocessor for FPGA Platforms", 9^{th} International Conference on Cryptology in India, INDOCRYPT 2008, 376–388, LNCS.

[8] C. Rebeiro, S.S. Roy, D.S. Reddy and D. Mukhopadhyay, "Revisiting the Itoh-Tsujii Inversion Algorithm for FPGA Platforms", IEEE Transactions on VLSI Systems, vol. PP Issue:99 (pre-print).

[9] U.S. Department of Commerce,National Institute of Standards and Technology, "Digital signature standard (DSS)," FIPS 186–2.

[10] C. Rebeiro and D. Mukhopadhyay, "Power Attack Resistant Efficient FPGA Architecture for Karatsuba Multiplier," in VLSID 08: Proceedings of the 21st International Conference on VLSI Design, IEEE Computer Society, 2008, pp. 706–711.