International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

ASIC Architectures for Implementing ECC Arithmetic over Finite Fields

Hemanth Ravindra[1], Jalaja S[2]

[1]Post Graduate Scholar, Department of ECE,
Bangalore Institute of Technology, Bangalore India

[2]Research Scholar (VTU) and Assistant Professor, Department of ECE,
Bangalore Institute of Technology, Bangalore, India

Abstract: The ever growing need for improved security for applications over internet has resulted in wide acceptance of Elliptic Curve Cryptography (ECC) in industry and academic research. This growth has started the spread of architectures for implementing ECC from FPGA towards ASIC. Computing scalar multiplication and point inversion forms the core ECC architecture. This paper discusses the ASIC based implementation of these ECC arithmetic primitives over finite fields GF(2^m). Scalar multiplication is based on a recursive variant of Karatsuba Algorithm and Inversion algorithms are based on quad-ITA. The arithmetic components are designed using Verilog and implemented using Cadence 45nm fast technology library. The proposed variation of Karatsuba Multiplier has low power considerations and better area delay product.

Keywords: ASIC based ECC, Karatsuba Algorithm variations, Combination of Algorithms, Quad-ITA, Low power design.

1. Introduction

The continued growth of applications over internet has greatly contributed to growth of Public-key cryptography. ECC offers improved security at shorter key sizes, when compared to other public-key based cryptosystems in use today. This has spread the use of ECC over a wide range cryptographic applications and generated lot of interest in academic as well as industry. The major applications are in the resource constrained environment, typically lower power, like smart cards, mobile banking, personal digital assistants and e-commerce. The use of Elliptic curves in cryptography was advocated independently by Neil Koblitz and Victor S Miller in 1980's [1-2]. Elliptic curves have been in commercial use since 2004. Today ECC is recognized and being standardized by organizations Like IEEE, ANSI, NIST and ISO. The advantages of ECC are due to inherent property of Elliptic curves to perform finite field arithmetic operations [3]. The security for ECC is defined by Discrete Logarithm Problem (ECDLP).

Traditionally ECC was implemented over FPGAs due to flexibility and lower costs. The growing needs can benefit from ASIC based design to provide faster and higher performances at costs and flexibility comparable to FPGAs. ASICs can offer dedicated hardware designs operating at lower area - increasing throughput and low power - increasing battery life. Also ASIC designs are difficult to read back further enhancing the security after implementation.

There are three major types of elliptic curves studied for implementing ECC [4].

- A pseudorandom curve over GF(p).
- A pseudorandom curve over GF(2^m).
- A special curve over GF(2^m) called a Koblitz curve or anomalous binary curve.

GF(2^m), an m-dimensional extension field of GF(2), is suitable for hardware implementation. Finite field arithmetic has no carry propagation and can be implemented with basic gates. This creates the need for optimization. The computations in ECC happen over vector and also scalars. A point P on the elliptic curve is a vector, which is converted to scalar to perform the point multiplication with a scalar k. The product kP converted back to point form by computing inversion. The effectiveness with which these two operations are performed defines the performance of the ECC.

Numerous works have been reported, which effectively performs Scalar multiply and inversion using Hardware, Firmware, Software or combination of approaches. Few of the literatures studied as a basis to this work is summarized below.

In [5], the ECC processor based on squarer's, adders and multipliers in the data path is proposed. This work also advocates the use of hybrid coordinate representation in affine, Jacobian, and López-Dahab form.

The LD coordinate form of the elliptic curve over binary finite fields is

$$Y^2 + XYZ = X^3 + aX^2Z^2 + b. \quad (1)$$

In [6], an end-to-end hardware implementation system for ECC is developed on an FPGA. The high performance is obtained with an optimized digit-serial shift-and-add multiplier for finite fields. Inversion is done with a dedicated division circuit.

Several acceleration techniques based on pre-computation are proposed for Koblitz curves in [7].

In [8], a pipelined ECC processor is developed which uses a combined algorithm to perform point doubling and point addition. This is the fastest reported in literature working with a pipelined architecture. However, the seven stage pipeline

used has huge area requirements.

In [9], a reconfigurable elliptic curve processor over GF(2^{167}) is designed and the processor consists of main controller and arithmetic units. The implementations of ECC processor over ASICs, are presented in works [10] and [11].

The finite field multiplier is the critical path in the ECC Processor design and preventing it from becoming idle improves the overall performance.

Algorithms for finding the multiplicative inverse are based upon extended Euclidean algorithms (EEA) and the Itoh-Tsujii Algorithm (ITA) [12]. EEA and its variants, the binary EEA and Montgomery [13] inverse algorithms are compact in hardware, but slower when compared to ITA. The ITA is faster but requires large area mainly taken up by a dedicated multiplier unit. This multiplier can be reused from the scalar multiplier by the ITA for inverse computations. This multiplier doesn't constitute area overhead of the ITA. The resulting architecture of ITA without the multiplier is as compact as the EEA making it a suitable multiplicative inverse option in hardware [14].

In [15], the authors discuss the use of basic exponentiation matrix chains to implement Inversion using minimum delay. This method trades delay for minimum clock cycle and not power efficient.

In [16], the use of LD Coordinates along the critical path is restructured to use non critical path such that logic structures are implemented in parallel. They also discuss the squarer based architecture of point addition and doubling iteration based on key bits.

A combined architecture for FPGA and ASIC is discussed in [17]. Here Redundant basis architectures are used to improve area, delay and power product. But these can be achieved by using divide and conquer approach of Karatsuba Variations as discussed in this work.

In [18], a normal basis multiplier using Gaussian model for Koblitz curve is discussed for constrained, secure applications. The GNB representation is reconfigured to meet the hardware complexity through sharing the addition/accumulation with other field additions. But the use of Addition chains and exponentiations becomes easy and still yields an efficient implementation.

The rest of the paper is organized as follows: A typical elliptic curve crypto processor with its arithmetic hierarchy is described in section 2; section 3 presents design of finite field multiplier. In section 4, a quad-ITA design is implemented. The results are discussed in 5, with consideration to area power and time requirements. Section 6 concludes the paper.

2. A Typical Elliptic Curve Cryptoprocessor

Elliptic curve crypto systems have a layered hierarchy as shown in Figure1. The bottom layer constituting the arithmetic on the underlying finite field most prominently influences the area and critical delay of the overall implementation. The group operations on the elliptic curve and the scalar multiplication influences the number of clock cycles required for encryption.

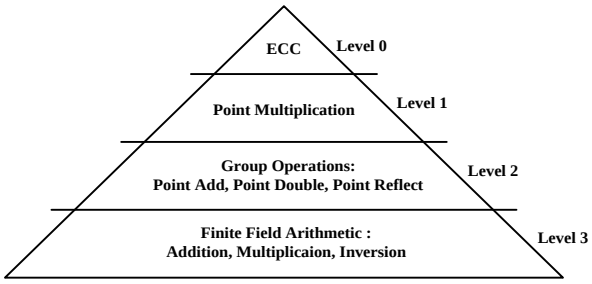

Figure 1: ECC layered hierarchy for arithmetic operations

To be usable in real world applications, implementations of the crypto system must be efficient, scalable, and reusable. Applications such as smart cards and mobile phones require implementations where the amount of resources used and the power consumed is critical. Such implementations should be compact and designed for low power. Computation speed is a secondary criterion. Also, the degree of reconfigurability of the device can be kept minimum [19]. This is because such devices have a short lifetime and are generally configured only once. On the other side of the spectrum, high performance systems such as network servers, data base systems etc. require high speed implementations of Elliptic Curve Cryptoporcessor (ECCP). The crypto algorithm should not be the bottleneck on the application's performance. These implementations must also be highly flexible. Operating parameters such as algorithm constants, etc. should be reconfigurable.

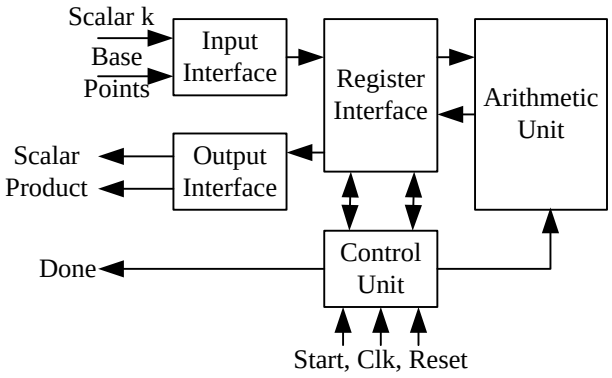

Figure 2: A typical elliptic curve cryptoprocessor

The Arithmetic primitives from the ECCP hierarchy can be grouped to constitute a dedicated hardware. A typical ECCP is given in Figure2. This crypto processor should implement the double and add scalar multiplication algorithm. Point doubling is performed for every iteration loop of the algorithm. Point addition is performed only when the bit is set in the binary expansion of scalar input k. This constitutes the Arithmetic unit (AU) of ECCP shown in Figure3. The output is the scalar multiplication product kP. Here P is the base point on the curve.

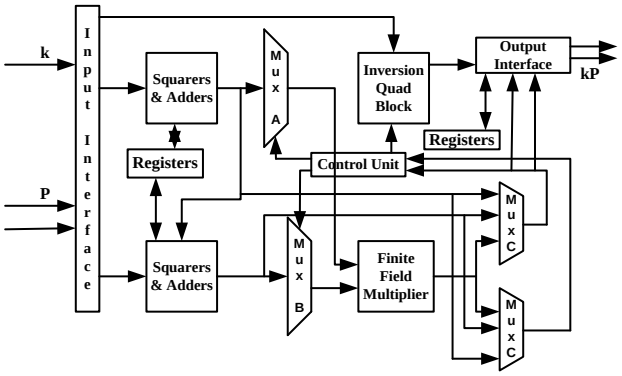

Figure 3: Arithmetic unit- Scalar multiplier and Inversion

The major components are the Quad block and Finite Field Multiplier block. The multiplier is used in Scalar multiplication as well as while computing inverse to Affine Co-ordinate. The Quad block is a cascaded quad circuit block, used only for inversion as a final computation step. The AU also has adders and squarer's which do not add to area or latency of the crypto processor significantly when compared with multiplier.

At every clock cycle the control unit produces a control word. Control words are produced in a sequence depending on the type of elliptic curve operation being done. The control word signals control the flow of data and also decide the operations performed on the data. This also helps to reuse the finite field multiplier by an enable signal.

3. Design of Finite Field Multiplier

The finite field multiplier is the integral operation of the ECCP and occupies highest level in the ECC design hierarchy. It's the most computationally intensive component, occupies most area and also has the longest latency. So it can be said, the performance of multiplier defines the performance of the ECCP. Finite field multiplication of two elements in the field GF(2^m) is given by equation

$$C(x)= A(x)*B(x) \bmod P(x) \qquad (2)$$

Where C(x), A(x), B(x) are in the field GF(2^m) and P(x) is the irreducible polynomial of the generator field in GF(2m). The product is essentially computed in two steps.

1. Compute C'(x) = A(x) * B(x)
2. Compute modulo on C'(x).

The Karatsuba multiplier [20] is based on divide and conquer approach and recursion to multiply A(x) and B(x). With each recursion the size of the multiplication required reduces by half. This leads to a reduction in the number of AND gates required at the cost of an increase in XOR gates. This algorithm has a complexity of O($m^{\log_2 3}$) for polynomial representations of finite fields. This is the only multiplier to have the sub quadratic complexity. It has also been shown in [21] that the Karatsuba multiplier if designed properly is also the fastest. This work uses a variation of Karatsuba Multiplier to perform finite field multiplication.

ECC uses binary extension fields that have a prime degree [4]. So the basic recursive multiplier has to be adapted for ECC. The design approaches in adapting are

1. Sequential circuit approach - Less hardware and latency. Requires several clock cycles to produce the result. At every clock cycle the outputs are fedback into the circuit, resulting in hardware reusability. Can be pipelined [22].
2. Combinational circuit approach – Large area and delay, but output generated in single clock cycle. Literatures [23,24] are examples of this approach.

Karatsuba Algorithm has a very rich design space and versatile making many variations possible. We look at few variations of Karatsuba Algorithm in combinational design approach. This work proposes and uses a combination of general Karatsuba multiplier with a threshold detection to achieve a beneficial recursion. Since this variation uses the best advantages of Simple Karatsuba and General Karatsuba implementations, we refer the proposed algorithm as Optimized Karatsuba multiplier which is a hybrid variation. The optimized Karatsuba Algorithm is given in Algorithm 1.

Algorithm 1: okmul (Optimized Karatsuba Multiplier)

Input: The multiplicands A, B and their length m

Output: C of length 2m − 1 bits

```
1. begin
2.     if m < 29 then
3.         return gkmul*(A,B,m)
4.     else
5.         l = [m/2]
6.         A′ = A[m−1 ・ ・ ・ l] + A[l−1 ・ ・ ・ 0]
7.         B′ = B[m−1 ・ ・ ・ l] + B[l−1 ・ ・ ・ 0]
8.         P1 = okmul(A[l−1 ・ ・ ・ 0],B[l−1 ・ ・ ・ 0], l)
9.         P2 = okmul(A′ ,B′ , l)
10.            P3 = okmul
           (A[m−1 ・ ・ ・ l],B[m−1 ・ ・ ・ l],m − l)
11.        return (P3 << 2l) + (P1 + P2 + P3) << l + P1;
12.    end
13. end
```

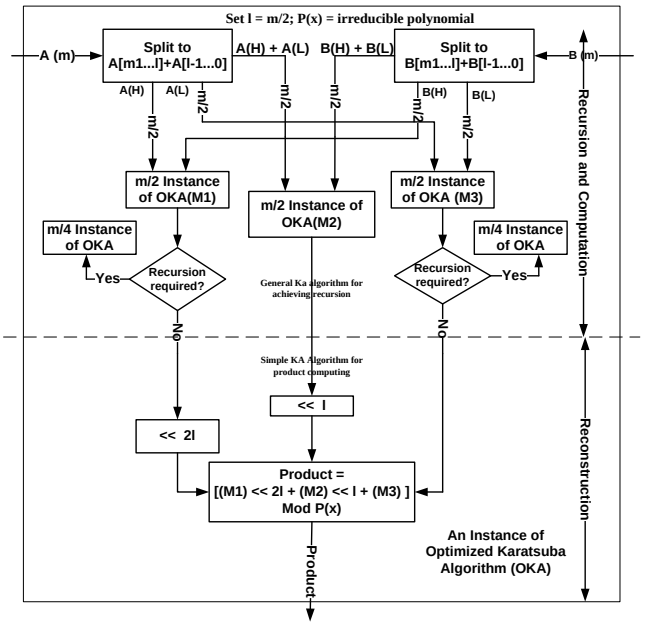

Figure 4: Architecture diagram for OKA multiplier Instance

The following advantages are obtained from using the Hybrid approach

1. The simple Karatsuba Algorithm ensures recursive application of Karatsuba Algorithm when multiplier complexity dominates register overhead.
2. The general Karatsuba Algorithm ensures normal multiplication when register overhead dominates multiplier complexity.

This approach results in low power and low area parameters by effectively managing recursion and thus also costs less cycle to operate hence better speed.

The number of recursion levels in hybrid Karatsuba multiplier is given by

$$r = [\log2\ (m/29)] + 1 \quad (3)$$

The levels of recursion for a GF(233^3) multiplier using OKA is given in Table 1.

Table 1: Levels of Recursion for GF (233)

m	r
233	4
116	3
58	2
29	1
14	0

The number of AND gates required:

$$\frac{3^{r-1}}{2} * \left[\frac{m}{2^{r-1}}\right] * \left[\left(\frac{m}{2^{r-1}}\right) + 1\right]$$

The number of XOR gates required:

$$3^{r-1}\left(10\left[\frac{m}{2^r}\right]^2 - 7\left[\frac{m}{2^r}\right] + 1\right) + \sum_{i=1}^{r-2} 3^i \left(4\left[\frac{m}{2^i}\right] - 4\right)$$

$Delay_{(h)}[m] =$
$Delay_{(s)}[m] – Delay_{(s)}[m/2^{r-1}] + Delay_{(g)}[m/2^{r-1}].$ (4)

4. Quad- Itoh Tsujii Algorithm

The Itoh- Tsujii algorithm was introduced in 1988, and is based on Fermat's Little theorem. For an element a ϵ GF (2^m) the inverse computation is done using equation

$$a^{-1} = a^{2m-2} \quad (5)$$

This technique requires (m-1) squaring and (m-2) multiplications. The ITA reduced the cost of multiplications by using Addition chains.

3.1 Brauer Addition Chains

An addition chain [25] is a sequence of integers of the form U = (u_0, u_1, u_2, …., u_r) for n ϵ N satisfying the following properties

1. $u_0 = 1$
2. $u_r = n$
3. $u_i = u_j + u_k$ for some $k \leq j < i$.
4. If the addition chain satisfy the property j = i- 1, it constitute a special class of addition chains called Braeur chains.

An optimal addition chain for n is the smallest addition chain for n.

The inverse equation can be rephrased as

$$a^{-1} = (a^{2^{m-1}-1})^2 \quad (6)$$

Reusing notations from [22], for k ϵ N, let

$$\beta_k(a) = a^{2^{k-1}} \epsilon\ GF\ (2^m) \quad (7)$$

Then,

$$a^{-1} = [\beta_{m-1}(a)]^2 \quad (8)$$

In [26] a recursive sequence is used with an addition chain to compute the multiplicative inverse. $\beta_{k+j}(a) \in GF(2^m)$ can be expressed as shown in Equation 9. For simplicity of notation we shall represent $\beta_k(a)$ by β_k.

$$\beta_{k=j} = (\beta_j)^{2^k}\beta_k = (\beta_k)^{2^j}\beta_j \quad (9)$$

In general if l is the length of the addition chain, finding the inverse of an element in GF (2^m) requires (l – 1) multiplications and (m – 1) squaring. The length of the addition chain is related to m by the equation $l \leq [\log_2 m]$ [25], therefore the number of multiplications required by the ITA is much lesser than that of the conventional method.

The procedure for obtaining the inverse for an odd m using the quad-ITA is shown in Algorithm2. The algorithm assumes a Brauer addition chain.

Algorithm 2: Quad -ITA

Input : The element a ∈ GF(2^m) and the Brauer chain U = {1, 2, · · · , m−1/2 , m − 1}
/*(1, 2, 3, 6, 7, 14, 28, 29, 58, 116, 232)*/
Output : The multiplicative inverse a^{-1}

1 begin
2 l = length(U)
3 a^2 = finitemul(a, a); /* finitemul: OKA of Algorithm1 */
4 $\alpha_{u1} = a^3 = a^2 \cdot a$
5 foreach $u_i \in U$ $(2 \le i \le l-1)$ **do**
6 p = u_i−1
7 q = $u_i - u_{i-1}$
8 α_{ui} = finitemul $\{(\alpha_p)^{4q}, \alpha_q\}$
9 end
10 a^{-1} = finitemul $(\alpha_{u(l-1)}, \alpha_{u(l-1)})$
11 **end**

Table 2: Exponentiation for GF (2 233) using Brauer Chains

	‘ui (a)	‘[uj + uk](a)	Exponentiation
1	1		‘a^3
2	2	1+1	α^{4^2-1}
3	3	2+1	α^{4^3-1}
4	6	3+3	α^{4^6-1}
5	7	6+1	α^{4^7-1}
6	14	7+7	$\alpha^{4^{14}-1}$
7	28	14+14	$\alpha^{4^{28}-1}$
8	29	28+1	$\alpha^{4^{29}-1}$
9	58	29+29	$\alpha^{4^{58}-1}$
10	116	58+58	$\alpha^{4^{116}-1}$

The architecture for implementing the quad-ITA [27, 28] is given in Figure 5.

Summary of Quad-ITA

1. The quadblock consists of 14 cascaded circuits, each circuit generating the fourth power of its input. Qsel line is asserted by the control unit to dictate which power gets passed on to the output. The output of the quadblock can be represented as qin^{4qsel}.
2. Two buffers MOUT and QOUT store the output of the multiplier and the quadblock respectively. En signal controls which block is active during a clock cycle. Quadblock and Multiplier block are mutually exclusive. A register bank may be used to store results of each step (α_{ui}) of Algorithm. A result is stored only if it is required for later computations.

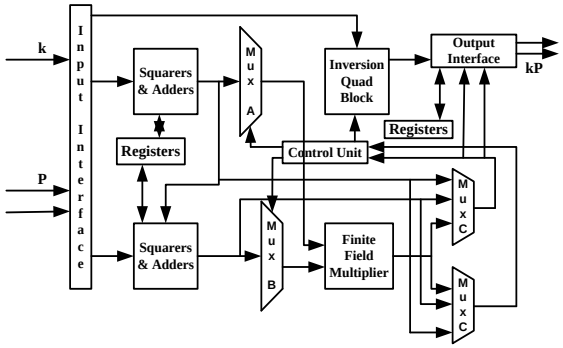

Figure 5: Architecture diagram for Quad-ITA

3. The controller is a state machine designed based on the adder chain and the number of cascaded quad circuits in the quadblock. At every clock cycle, control signals are generated for the multiplexer selection lines enables to the buffers and access signals to the register bank.
4. The length of the addition chain influences the number of clock cycles required to compute the inverse, hence proper selection of the addition chain is critical to the design. For a given m, there could be several optimal addition chains. It is required to select one chain from available optimal chains. The amount of memory required by the addition chain can be used as a secondary selection criterion. The memory utilized by an addition chain is the register’s required for storage of the results from intermediate steps.
5. Using Brauer chains has the advantage that for every step (except the first) at least one input is read from the output of the previous step. The output of the previous step is stored in MOUT therefore need not be read from any register and no storage is required. The second input to the step would ideally be a doubling. For example, computing α116 (a) requires only α58 (a). Since α58 (a) is the result from the previous step, it is stored in MOUT. Therefore computing α116 (a) does not require any stored values.
6. The number of quad circuits cascaded (u_s) has an influence on the clock cycles, frequency, and area requirements of the quad-ITA. Increasing the number of cascaded blocks would reduce the number of clock cycles required at the cost of an increase in area and delay.
7. In general for addition chains used in ECC, the value of $(u_l - u_{l-1})$ is as large as (m−1)/2 and much greater than u_s, therefore the clock cycles saved is significant.

Usually point doubling takes 4 states and point double and add will take 4 + 8 = 12 states. The implementation of this Algorithm is optimized to perform both point doubling and point double and add operations using all the 12 states. A dummy adder logic is used where point add is not required. This is to ensure the implementation does not leave out any power traces for different state computation.

5. Results

The design followed the classical Design, Simulate and Synthesis cycle implemented using Cadence Tool Suites - Simvision for Simulation and RTL Compiler for Synthesis. The synthesis made use of 45 nm Fast Technology library Version xx, under the operating conditions of a fast balanced

tree with segmented wire load mode. The synthesis used Incremental step optimization flow for all the designs.

The proposed multiplier is implemented over bit lengths of 163, 193 and 233 bit in accordance with the prime curves used in ECC. The performance parameters are derived in terms of Area, Time (operating clock cycles) and Power and compared between different variations of Karatsuba algorithm namely, Simple Karatsuba Algorithm (SKA)-parent algorithm in recursive form, binary implementation (BKA), and proposed implementation (OKA). Table2 gives the synthesis results of this implementation.

A brief comparison between various algorithms shows that, the proposed algorithm is found to be better in terms of Area, power and time required to operate over the implemented range. These results are tabulated in Table 3.

Table 2: Area, Time and Power of SKA, BKA and OKA

Simple Implementation (SKA)			
Bits	**Area**	**Timing(pS)**	**Power(nW)**
163	544286	3430	234268579.000
193	671683	3486	305770390.000
233	867156	3438	419994621.000
Binary Implementation (BKA)			
Bits	**Area**	**Timing(pS)**	**Power(nW)**
163	707838	3980	314557905.000
193	847869	5022	385854841.000
233	935434	4064	458291477.000
Optimized Implementation (OKA)			
Bits	**Area**	**Timing(pS)**	**Power(nW)**
163	467313	3194	174594221.000
193	633530	3282	244037581.000
233	797428	3315	352673702.000

Table 3: Area, Time and Power Compare

SKA Vs OKA			
Bits	**Area**	**Timing**	**Power**
163	14.14	6.88	25.47
193	5.68	5.85	20.19
233	8.04	3.58	16.03
BKA vs OKA			
Bits	**Area**	**Timing**	**Power**
163	33.98	19.75	44.50
193	25.28	34.65	36.75
233	14.75	18.43	23.05
All parameters in percentage			

Since the Quad-ITA uses multipliers exponentiation to perform Inversion, the multipliers were tested on the Quad-ITA block. Since the Quad-ITA is optimized to operate in equal number of states for all values of scalar, there was no much timing difference, but the proposed algorithm did better in area(reduction by 11.83%) and power(reduction by 11.7%) parameters.

Table 4: Area, Time and Power Quad-ITA using

Quad-ITA 233 bit			
	Area	**Timing(pS)**	**Power(nW)**
OKA	1020114	7195	36686801.000
BKA	1157002	7085	41549538.000

6. Conclusion

The performance of an ECCP can be greatly improved if the underlying arithmetic primitives are carefully designed. The proposed multiplier combines the advantages of two variants of Karatsuba, namely the general and the simple Karatsuba multipliers.

The general Karatsuba multiplier has a large gate count. Use of this becomes multiplier beneficial when register overhead due to recursion dominates the multiplier complexity, Simple Karatsuba algorithm helps in recursively breaking the larger multiplications into smaller ones and also accounts for reusability in modules.

The proposed multiplier optimizes the register overhead and multiplier complexity by evaluating a threshold level up to which recursion is beneficial and uses the algorithms effectively. The initial recursion instances are computed using the simple algorithm while final small sized multiplications are done using the general algorithm. The proposed algorithm is efficient with respect to low power, area and speed when compared to other variations of Karatsuba Algorithm implementation.

The inverse architectures reuses the multiplier module and hence area efficient. This overcomes the critical drawbacks of Itoh-Tsujii Inversion methods. Also the always double and add implemented makes it resistant to simple power attacks.

References

[1] N. Koblitz, “Elliptic curve cryptosystems”, Mathematics of Computation, number 48, pages 203-209, 1987.
[2] V.S. Miller, “Use of elliptic curve in cryptography”, Advances in Cryptology– Proceedings of CRYPTO’85, Springer Verlag Lecture Notes in Computer Science 218, pages 417-426, 1986.
[3] Certicom research, “The Elliptic Curve Cryptosystem”, Certicom, April 1997.
[4] U.S. Department of Commerce,National Institute of Standards and Technology, “Digital signature standard (DSS),” 2000.
[5] M. Bednara, M. Daldrup, J. von zur Gathen, J. Shokrollahi, and J. Teich, “Reconfigurable Implementation of Elliptic Curve Crypto Algorithms,” in Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM, 2002, pp. 157–164.
[6] Nils Gura, Sheueling Chang Shantz, Hans Eberle, Sumit Gupta, Vipul Gupta, Daniel Finchelstein, Edouard Goupy, and Douglas Stebila, “An End-to-End Systems Approach to Elliptic Curve Cryptography,” in CHES ’02: Revised

Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems, London, UK, 2003, pp. 349–365, Springer-Verlag.
[7] Jerome A. Solinas, "Efficient Arithmetic on Koblitz Curves," Des. Codes Cryptography, vol. 19, no. 2-3, pp. 195–249, 2000.
[8] W. N. Chelton and M. Benaissa, "Fast Elliptic Curve Cryptography on FPGA," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 2, pp. 198–205, Feb. 2008.
[9] G.Orlando, C.Paar, A high performance reconfigurable elliptic curve processor for GF(2m), Second International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), pp 41-56.
[10] Sakyama K, Batina L, Preneel B, et al, Multicore curve-based cryptoprocessor with reconfigurable modular arithmetic logic units over GF(2^n), IEEE Transactions on Computers, vol 56 (9), pp 1269-1282, 2007.
[11] Sozzana F, Bertoni G, S Turcato, et al, A parallelized design for an elliptic curve cryptosystem coprocessor, Proceeding of the International Conference on Information Technology, IEEE Computer Society, pp 626-630, 2005.
[12] Toshiya Itoh and Shigeo Tsujii, "A Fast Algorithm For Computing Multiplicative Inverses in GF(2m) Using Normal Bases," Inf. Comput., vol. 78, no. 3, pp. 171– 177, 1988.
[13] Burton S. Kaliski, "The Montgomery Inverse and its Applications," IEEE Transactions on Computers, vol. 44, no. 8, pp. 1064–1065, 1995.
[14] Francisco Rodríguez-Henríquez, N. A. Saqib, A. Díaz-Pèrez, and Çetin Kaya K¸ oc, Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[15] Parrilla, L.; Lloris, A.; Castillo, E.; García, A., "Minimum-clock-cycle Itoh-Tsujii algorithm hardware implementation for cryptography applications over GF(2m) fields," Electronics Letters , vol.48, no.18, pp.1126,1128, August 30 2012, doi: 10.1049/el.2012.1427
[16] Mahdizadeh, H.; Masoumi, M., "Novel Architecture for Efficient FPGA Implementation of Elliptic Curve Cryptographic Processor Over {\rm GF}(2^{163}) ," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.21, no.12, pp.2330,2333, Dec. 2013, doi: 10.1109/TVLSI.2012.2230410
[17] Jiafeng Xie; Meher, P.K.; Zhi-hong Mao, "High-Throughput Finite Field Multipliers Using Redundant Basis for FPGA and ASIC Implementations," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.62, no.1, pp.110,119, Jan. 2015
[18] Azarderakhsh, R.; Jarvinen, K.U.; Mozaffari-Kermani, M., "Efficient Algorithm and Architecture for Elliptic Curve Cryptography for Extremely Constrained Secure Applications," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.61, no.4, April 2014
[19] JohannesWolkerstorfer, Hardware Aspects of Elliptic Curve Cryptography, Ph.D. thesis, Institute for Applied Information Processing and Communications, Graz University of Technology, 2004.
[20] Anatoly A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata," Soviet Physics Doklady, vol. 7, pp. 595–596, 1963.
[21] Francisco Rodríguez-Henríquez, N. A. Saqib, A. Díaz-Pèrez, and Çetin Kaya K¸ oc, Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[22] Steffen Peter and Peter Langendörfer, "An efficient polynomial multiplier in GF(2m) and its application to ECC designs," in DATE '07: Proceedings of the conference on Design, automation and test in Europe, San Jose, CA, USA, 2007, pp. 1253–1258, EDA Consortium.
[23] Peter L. Montgomery, "Five, Six, and Seven-Term Karatsuba-Like Formulae," IEEE Transactions on Computers, vol. 54, no. 3, pp. 362–369, 2005.
[24] André Weimerskirch and Christof Paar, "Generalizations of the Karatsuba Algorithm for Efficient Implementations," Cryptology ePrint Archive, Report 2006/224, 2006.
[25] Donald E. Knuth, The Art of Computer Programming Volumes 1-3 Boxed Set, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.
[26] Francisco Rodríguez-Henríquez, Guillermo Morales-Luna, Nazar A. Saqib, and Nareli 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.
[27] Rebeiro, C., Roy, S.S., Reddy, D.S., and Mukhopadhyayy, D.: 'Revisiting the Itho-Tsujii inversion algorithm for FPGA platforms', IEEE Trans. Verg Large Scale Integr. (VLSI) Syst., 2011, 19, (8), pp. 1508–1512
[28] Roy, S.S., Rebeiro, C., and Mukhopadhyay, D.: 'Theoretical modeling of the Itoh-Tsujii inversion algorithm for enhanced performance on k-LUT based FPGAs'. Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE), Grenoble, France, March 2011, Vol. 1, pp. 1–6

Author Profile

Hemanth R received his B.E in Electronics and communication from Visvesvaraya Technological University (VTU) in 2010. He worked at HCL Technologies Ltd, Bangalore for 3 years from October 2010 - October 2013 in the field of verification and validation. Currently, He is pursing M.Tech in VLSI Design and Embedded systems from Bangalore Institute of Technology, VTU.

Jalaja S holds M.Tech degree in VLSI Design and Embedded System from Visvesvaraya Technological University (VTU). She is a E-Member of IEEE and presently serving as an Assistant Professor in Electronics and Communication Engineering department at Bangalore Institute of Technology, Bangalore (Karnataka). She is currently pursuing the Ph.D. degree in Electronic and Communication engineering from the VTU, Belgaum. Her research interest includes Analysis of various algorithms to design different VLSI architectures and ASIC implementation for digital signal processing applications.