ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)
On 21st & 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Bipartite *GF* (2^m) Modular Multiplier Method

V.R.Venkatasubramani[#1], M.Arunarumugam[#2], R.Ragavendran[#3], S.Bhuvaneswaran[#4],
P.Naveen Kumar[#5], S. Rajaram[#6]

Department Of ECE, Thiagarajar College of Engineering, Madurai,Tamilnadu, India

***Abstract*—**This paper proposes a new modular multiplication method over *GF(2^m)* that uses Least Significant Digit Multiplier and Hybrid Karatsuba Multiplier (HKM), which are state of the art field for secured Elliptic Curve Cryptography (ECC), according to NIST. The work suggests the operand multiplicand to be split into two parts that can be processed separately in parallel thereby increasing the computational speed. The lower part of the split multiplicand can be processed by calculating a product modulo *p(α)*of the multiplier using Least Significant Digit (LSD).The upper part of the split multiplicand can be processed using HKM by calculating a product modulo *p(α)* of the multiplier. A HKM requires least amount of space on a FPGA. The LSD provides excellent area-time trade-off. Complexity analysis comparison shows that the proposed scheme has better calculation speed and has more flexibility in making the compromise between area and time.

***Keywords*—**Least Significant Digit (LSD);Hybrid Karatsuba Multiplier (HKM);Elliptic Curve Cryptography(ECC).

1.INTRODUCTION

Curve-based cryptography, especially Elliptic Curve cryptography (ECC) [4] [2], has become increasingly popular in the last few years. ECC scheme show good properties for software and hardware implementation because of the relatively short operand length compared to other public-key schemes, like RSA [7]. They are thus the preferred cryptosystem for the important domain of embedded applications. There are two types of finite fields standardized as underlying structure for ECC: prime fields *GF(p)* and characteristic two fields *GF(2^m)*. The latter ones are preferred for hardware realizations due to the smaller hardware circuits required for the corresponding arithmetic. ECC is used for exchanging keys over an insecure channel and for digital signatures that require the use of large module, which makes repeated modular multiplications a computationally intensive task. The overall time and area complexity of ECC implementations heavily depends on the finite field multiplier architecture[10] used. Hence, designing efficient multiplier can have major benefits. For high-performance systems, parallelism must be exploited as much as possible to achieve a high throughput. There are numerous techniques for speeding up modular multiplication that has been reported in the literature. Among them, two major techniques are notable. One is based on the LSD modular multiplication algorithm that allows implementations to do a trade-off between speed and area[14], thus resulting in fast implementations tuned to the available resources of the hardware. The other is based on the Karatsuba algorithm, which has the best area-time product on an FPGA compared to implementations in [9] [12].Techniques for improving the speed of these two approaches have been developed separately.

The work proposes a Bipartite *GF(2^m)* Modular Multiplication (BMM), which takes advantage of these two approaches. The multiplicand is split into two parts that can then be processed separately in parallel. The HKM and the LSD significant multiplier algorithm can be employed to process the upper and lower parts of the split multiplicand, respectively. Due to parallel processing, the proposed method is suitable for hardware as well as software implementation in a multiprocessor environment.

The remainder of the paper is organized as follows: Section II gives an overview of LSD Multipliers and conditions for choosing efficient reduction polynomials and HKM. Section 3 introduces our Bipartite Modular Multiplication. Section 4 shows the complexity analysis for the existing LSD Multiplier and Modified LSD multiplier. Finally, we end this conclusion.

2.PRELIMINARIES

2.1 DIGIT-SERIAL MULTIPLIERS

Finite field multiplication in *GF(2^m)* of two elements A and B to obtain a result *C=A.B mod p* ⟨*α*⟩ can be done using LSD multipliers for binary fields *GF(2^m)*. LSD multipliers provide a trade-off between speed and area [8]. Hence it is most preferred in cryptographic hardware implementations [1], [5].Certain *B*'s coefficients are processed at the same time. The number of coefficients that are processed in parallel is the digit-size *D*. The total number of digits is given by $d = \lceil m/D \rceil$.The multiplier can be rewritten as $B = \sum_{i-0}^{d-1} B_i \alpha^{D_i}$,where .

$$B_i = \sum_{j=0}^{D-1} b_{D_i+j} \alpha^j, 0 \le i \le d-1$$

Assuming *B* has been properly padded with zero coefficients for the most significant bits. Hence, the multiplication can be performed as shown in Algorithm 1.
Where *a,bε GF (2^m)* and *b ≠0*.

2.1.1 Reduction mod p(α) for Digit Multipliers

In step 7 of Algorithm 1, products of the form $A\alpha^D$ have to be reduced mod $p(\alpha)$. Optimum irreducible polynomials can be determined using the Theorem 1 and 2 to minimize the complexity of the reduction operation [8].

Theorem 1. Assume that the irreducible polynomial is of the form $p(\alpha) = \alpha^m + p^{k\alpha^k + \sum_{j=0}^{k-1} p_j \alpha^j}$, with $k<m$, for $t \leq m-1 \leq k$ $.\alpha^{m-1}$ can be reduced to a degree less than m in one step with the following equation:

$$\alpha^{m+t} mod\ p(\alpha) = p_k \alpha^{k+t} + (\sum_{j=0}^{k-1} p_j \alpha^{j+1}) \quad (2)$$

*Theorem 2***.** For digit multipliers with digit-element size *D*, when $D \leq m-k$, the intermediate results in Algorithm 1 (Step 4 and Step 6) can be reduced to a degree less than m in one step.

Theorems 1 and 2 implicitly say that, for a given irreducible polynomial $p(\alpha) = \alpha^m + p_k \alpha^k + \sum_{j=0}^{k-1} p_j \alpha^j$, the digit-element size D has to be chosen based on the value of k, the degree of the second highest coefficient in the irreducible polynomial.

2.2 Karatsuba Multiplier

The Karatsuba multiplier[6] can be used to compute multiplications in any field of the form $GF(2^m)$, The inputs A and B are split into two: A_H, A_L and B_H, B_L respectively. The A_L and B_L terms have $\lceil m/2 \rceil$bits and the remaining $\lfloor m/2 \rfloor$ bits are in A_H and B_H terms. Hence Karatsuba multiplication requires two $\lceil m/2 \rceil$ bit multiplications. Maximum number of AND gates and XOR gates required for the $2^{\lceil \log_2 m \rceil}$ bit basic recursive Karatsuba multiplier is

#AND gates: $n^{\log_2 3}$

#XOR gates: $\sum_{r=0}^{\log_2 n} 3^r (\frac{4n}{2^r} - 4)$

In addition, *m* number of two input XOR gates are required for the computation of $A_H + A_L$ and $B_H + B_L$.

The Simple Karatsuba multiplier is basic recursive Karatsuba multiplier [15] with smaller modification. If an *n* bit multiplication is needed to be done, *n* being any integer, the input is split into two polynomials as in equation (3). The polynomials A_L and B_L have $\lceil n/2 \rceil$ terms while the A_H and B_H polynomials have $\lfloor n/2 \rfloor$ terms. Karatsuba multiplication can be done with two $\lceil n/2 \rceil$bit multiplications and a single $\lfloor n/2 \rfloor$ bit multiplication. The Simple Karatsuba multiplier requires at most one bit padding (for the (A_H+A_L) (B_H+B_L) multiplication). It therefore requires lesser number of gates for implementation as compared with the Binary Karatsuba multiplier [15]. For an *n* bit multiplication, the Simple Karatsuba multiplier would be used recursively for $\lceil log2n \rceil$times. This is higher than the Binary Karatsuba multiplier which would be used recursively for $\lfloor log2n \rfloor$times. Hence, the delay in the Simple Karatsuba algorithm is expected to be higher than that of the Binary Karatsuba algorithm.

$$C(x)=(A_Hx^{n/2}+A_L)(B_Hx^{n/2}+B_L)$$
$$=A_HB_Hx^n+(A_HB_L+A_LB_H)x^{n/2}+A_LB_L$$
$$=A_HB_Hx^n+((A_H+A_L)(B_H+B_L)+A_HB_H+A_LB_L)x^{n/2}$$
$$+A_LB_L \quad (3)$$

The basic Karatsuba multiplier explains a method to multiply two *n* bit polynomials using three $\lfloor n/2 \rfloor$ bit multipliers. This can be achieved by splitting the *n* bit polynomial into a 2-term polynomial with $\lfloor n/2 \rfloor$ bits in each term. It was shown that if *A* and *B* are two $n=3k$bit polynomials, the Karatsuba multiplier for 3-term polynomials can be used as shown in equation (4). This results in six multiplications and 13 additions

$$C=AB$$
$$=(A_2x^{2n/3} + A_1x^{n/3} + A_0)(B_2x^{2n/3} + B_1x^{n/3} + B_0)$$
$$=A_2B_2x^{4n/3} + (A_2B_1 + A_1B_2)x^n + (A_2B_0 + A_0B_2 + A_1B_1)x^{2n/3} +$$
$$(A_1B_0 + A_0B_1)x + A_0B_0$$
$$=A_2B_2x^{4n/3} + ((A_2 + A_1)(B_2 + B_1) + A_2B_2 + A_1B_1)x^n +$$
$$(((A_2 + A_0)(B_2 + B_0) + A_2B_2 + A_1B_1 + A_0B_0)x^{2n/3} +$$
$$((A_1 + A_0)(B_1 + B_0) + A_1B_1 + A_0B_0)x^{n/3} + A_0B_0 \quad (4)$$

To apply the Binary Karatsuba multiplier recursively: Let *n* be a composite number (if *n* is prime we pad it by one bit) with the prime factors in increasing order being $\{p_1, p_2, p_{3\ldots}\}$. To multiply two n bit numbers, we should first do the $p_1$term Karatsuba. Each term is having n/p_1bits. The n/p_1term multiplication can be done using $p_2$term Karatsuba. The p_1/p_2term multiplication can be done using the $p_3$term Karatsuba and so on.

3.BIPARTITE MODULAR MULTIPLICATION METHOD

In this section, a novel Bipartite Modular Multiplication (BMM) is proposed. The block diagram of the proposed BMM is shown in Fig 1. The multiplier A to be split into two parts A_H and A_L so that $A=A_H+A_L$, The term $A_L.Bmodp(\alpha)$ [13], can be calculated using the proposed LSD multiplier shown in Algorithm1 that processes the lower part of the split multiplier A_L. The other term, $A_H.Bmodp(\alpha)$ [13], can be calculated using the Hybrid Karatsuba Multiplier algorithm that processes the upper part of the split multiplier A_H. These two calculations are performed in parallel. Since the split operator A_H and A_L are half the length of A , the calculations $A_L.B \bmod p(\alpha)$and $A_H.Bmod\ p(\alpha)$ are performed faster than the individual execution of the LSD multiplication algorithm [11] and has better area-time trade-off than that of the Hybrid Karatsuba Multiplication algorithm [15] with unsplit operator.

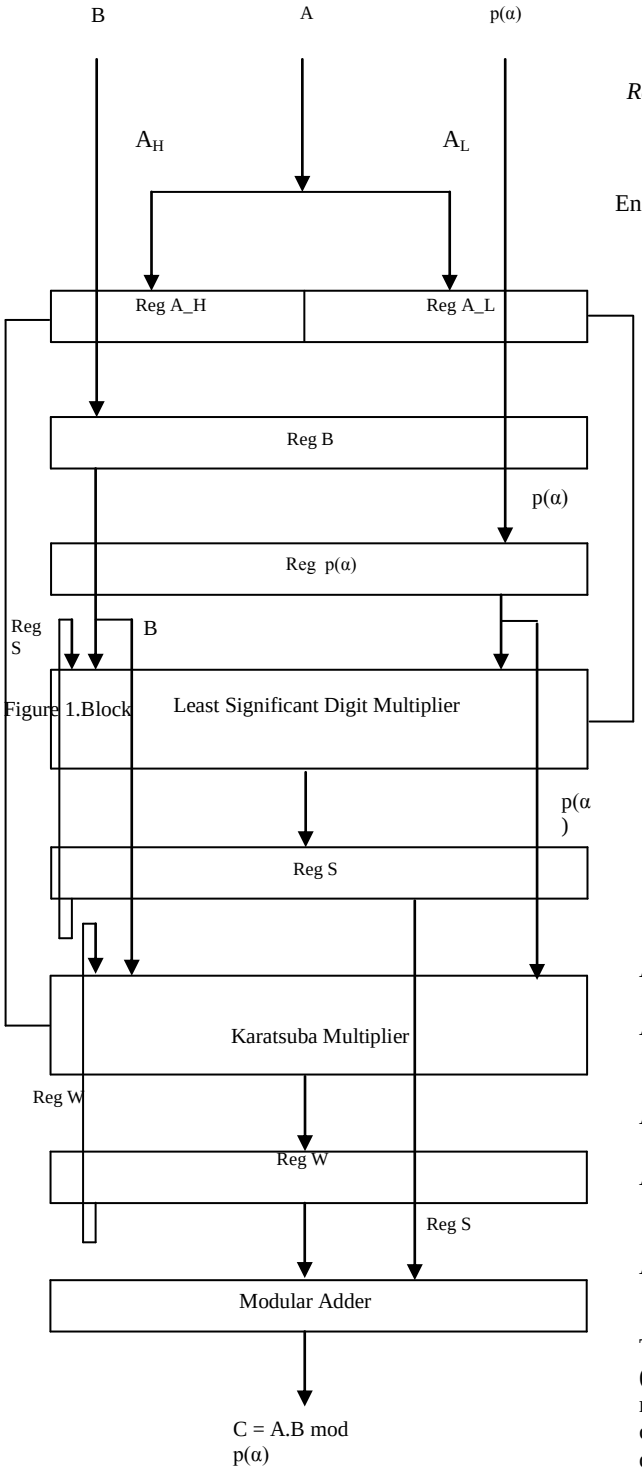

Figure 1.Block

3.1 LSD Multiplier

The Digit serial multiplier architecture in [8][11]is used as a basis for developing the proposed BMM. It consists of three main components - LSD multiplier, main reduction circuit, and final reduction circuit. The LSD multiplier computes the intermediate *C* and stores it in the accumulator. The main reduction circuit is a shifter cum reduction unit that shifts *A* left by *D* positions and reduce the result *mod p(α)*.The final reduction circuit is used to reduce the contents in the accumulator to get the final result *C*.

Algorithm 1 Proposed LSD Multiplier

Require: $A_L = \sum_{i=0}^{m-1} a_i \alpha^i$, where,$a_i \in GF(2)$,

$B = \sum_{i=0}^{d-1} B_i \alpha^{D_i}$,where B_i is as in (1)

Ensure: $C \equiv A_L.B \ mod\ p(\alpha) = \sum_{i=0}^{m-1} c_i \varepsilon^i$, where $c_i \in GF(2)$

$1: C \leftarrow 0$

$2: for\ i = 0\ \ to \left\lceil \frac{m}{2D} \right\rceil - 1\ do$

$3: \quad C \leftarrow B_i A_L[\frac{m}{2} - 1 : 0] \times 2^{Di} + C$

$4: end\ \ for$

$5: A_L \leftarrow A_L \times 2^{\left[\frac{m}{2}\right]}$

$6: for\ i = \left\lceil \frac{m}{2D} \right\rceil to \left\lceil \frac{m}{D} - 1 \right\rceil do$

$7: \quad C \leftarrow B_i A_L[m-1:0] + C$

$8: \quad A_L \leftarrow A_L \alpha^D \bmod p(\alpha)$

$9: end\ \ for$

$10: return\ (C \bmod p(\alpha))$

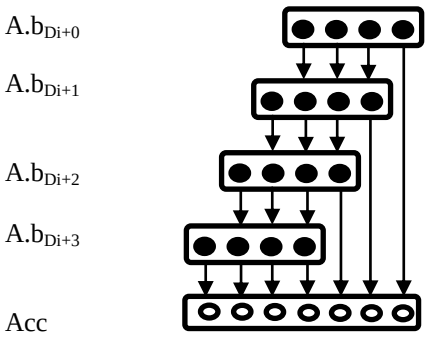

Figure.3. LSD multiplier for D = 4

The LSD multiplier performs the operation$C \equiv A_L B mod\ p(\alpha)$ (Step 3 and 6in Algorithm 1). The implementation of the LSD multiplier is as shown in Fig. 3 for a digit size *D= 4*[11]. It consists of *AND*ing the multiplicand A with each element of the digit of the multiplier B and *XOR*ing the result with the accumulator *Acc* and storing it back in *Acc*. The LSD multiplier in Step 3 of Algorithm 1 requires $mD/2$*AND* operations for$\lceil m/2D \rceil - 1$iterations.(denoted by the black dots), $\lceil mD/2 \rceil$*XOR* operations for$\lceil m/2D \rceil - 1$ iterations.(for *XOR*ing the columns denoted by the vertical line plus the *XOR* operations for the accumulator),In this LSD multiplier always most significant$\lceil m/2 \rceil$bits of the result is zero for $\lceil m/2D \rceil - 1$ iterations. The number of Flip-Flops (FF) required for accumulating the result *C is* $m/2 + D - 1$. Thus there is a reduction of$\lceil mD/2 \rceil$ number of AND operations,$\lceil mD/2 \rceil$ number of XOR operations and $m/2$ number of FFs activity compared to that in [11]. However the operations in Step 6 of

Algorithm 1 are similar to that in [11] except for the decrease in iteration by half. The complexity analysis for Algorithm 1 is shown in Table 1.The proposed LSD multiplier method requires$\lceil m/D \rceil$ multiplications, $\lceil m/D \rceil$ additions,$3m/D + 3$ reads, and$m/2D + 2$writes, The critical path delay of LSD multiplier is$\Delta_{AND} + \lceil \log_2 (D+1) \rceil \Delta_{XOR}$, same as that in [11].

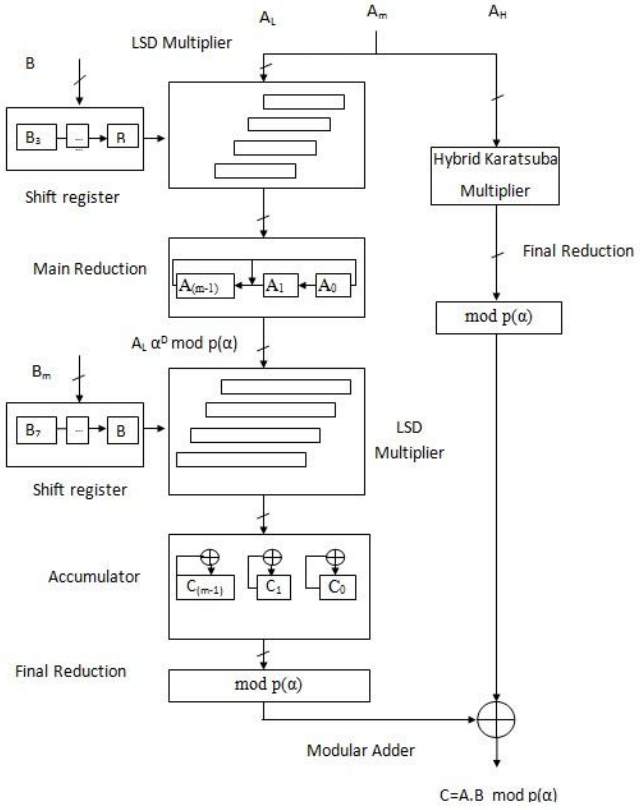

Figure.2. Proposed LSD Multiplier Architecture

3.2 Hybrid Karatsuba Multiplier

The Hybrid Karatsuba multiplier performs the operation for $C \equiv A_H B mod\ p(\alpha)$.In Hybrid Karatsuba multiplier, except the final recursion all recursions are done using the Simple Karatsuba multiplier. The final recursion is done using the General Karatsuba multiplier when the multiplicands have a size lesser than 29 bits. The initial recursions using the Simple Karatsuba multiplier result in low gate count, while the final recursion using the General Karatsuba multiplier results in low LUT requirements.

For a 233-bit Hybrid Karatsuba the initial four recursions are done using the Simple Karatsuba multiplier, while the final recursion is done with 14-bit and15-bit General Karatsuba multipliers. The General Karatsuba multiplier however, the smallest multiplication is a 13-term13-bit multiplication. This has several operations which can be grouped in terms of their inputs. Therefore, the General Karatsuba multiplier obtains maximum utilization of the slices of the FPGA. In Hybrid Karatsuba Multiplier design we implement the initial recursions using the Simple Karatsuba multiplier and the final recursion is using the General Karatsuba multiplier. For a 233 bit Hybrid Karatsuba Multiplier, we do all the larger multiplications using the Simple Karatsuba algorithm. The smallest multiplications, i.e. 14-bit and 15-bit, are done using the General Karatsuba algorithm. We now determine the upper bound for the number of gates required for an *n* bit Hybrid Karatsuba multiplier. Let $n' = 2^{\lceil \log_2 n \rceil}$ and let *k* be the number of recursions needed for Simple Karatsuba multiplication. The final recursion using the General Karatsuba algorithm is done with $m(m \le m' = n'/2^k)$bit multiplicands. The number of *AND* gates required for an *m*' bit General Karatsuba multiplication is*m'(m'+1) / 2*. Similarly, k recursions of Simple Karatsuba multiplication require $\sum_{r=0}^{k} 3^r \left(\frac{4n}{2^r} - 4\right)$*XOR* gates, and *m*' bit General Karatsuba multiplication require *(5/2)m'²-(7/2)m'+1 XOR* gates. The upper bound for the total number of AND gates and *XOR* gates required for the *n* bit Hybrid Karatsuba multiplication is given.

Method	Multiplications	Adds	Reads	Writes
LSD Multiplier	$\frac{2m}{D}$	$\frac{m}{D}$	$\frac{4m}{D}+2$	$\frac{m}{D}+1$
Proposed LSD Multiplier	$\frac{m}{D}$	$\frac{m}{D}$	$\frac{3m}{D}+3$	$\frac{m}{2D}+2$

#AND gates: $3^k m'(m'+1)/2$

#XOR gates: $\left(\frac{5m'^2}{2} - \frac{7m'}{2} + 1\right)3^{k+1} + \sum_{r=0}^{k} 3^r \left(\frac{4n}{2^r} - 4\right)$

3.1.2Main Reduction Circuit

The main reduction circuit performs the$A_L \leftarrow A_L \times 2^{\lceil m/2 \rceil}$(Step 5in Algorithm 1). In this main reduction circuits can be simple left shift by *m/2*bits for $\lceil m/2D \rceil - 1$.The critical path delay is zero for $\lceil m/2D \rceil - 1$iterations. Hence we save *(k+1)D* AND operations and *kD* XOR operations reduced for $\lceil m/2D \rceil - 1$ iteration. *m* number Flip flop to store A bits and k+1 number of flip flops to store p(α) bits for this $\lceil m/2D \rceil - 1$iterations. So, these iterations save for k+1 number of flip flops. For$\lceil m/2D \rceil$to$\lceil m/D - 1 \rceil$ iteration main reduction circuit performs the $A_L \leftarrow A_L \alpha^D \bmod p(\alpha)$(Step 7in Algorithm 1). In this main reduction circuits can be replaced by simple left shift by D bits [11]. Here, the multiplicand A is shifted left by the digit-size D which is equivalent to multiplying byα^D. The result is then reduced with the reduction polynomial by *AND*ing the higher D elements of the shifted multiplicand with the reduction polynomial p(α) (shown in the figure as pointed arrows) and *XOR*ing the result[11]. We assume that the reduction polynomial is chosen according to Theorem 2 in [11].

3.1.3 Final Reduction Circuit

The final reduction circuit performs the operation *Cmodp(α)*(Step 9in Algorithm 1)., where C is of size *m+D-1*. It is implemented in [11], which is similar to the main reduction circuit (Step 7 in Algorithm 1) without any shifting [11]. Here, the most significant *(D-1)* elements are reduced using the reduction polynomial *p(α)* in [11]. The area requirement for this circuit is *(k +1) (D-1) AND* operations and *(k+1)(D-1) XOR*operationsis less than that of the main reduction circuit in [11]

IV. COMPLEXITY ANALYSIS

The complexity analysis for the existing LSD Multiplier methodis given in [1]. The complexity analysis for the Modified LSD Multiplier (MLSD Multiplier) is presented as follows

Table 4.1 Complexity analysis of proposed LSD multiplier

Table 4.2 Comparing Latency of the various methods

HKM algorithm requires $m^{\log_2 3}$ (is equal to $m^{1.585}$) multiplications. HKM is usually faster when the operand size is more than 100.

V. CONCLUSION

The paper presents a fast method for modular multiplication over GF. The multiplier is split into two parts that can then be Processed in parallel, increasing the speed of calculation. The upper part of the split multiplier is processed by calculating a product modulo M by HKM of the multiplicand and this part of the split multiplier. The lower part of the split multiplier is processed by LSD multiplier calculating a product modulo *p(α)* of the multiplier and this part of the split multiplier. Speeding up techniques can be applied to LSD multipliers by there isflexibility in choice of digit size position is left as a parameter. This allows for theinvestigation of different design trade-offs.

Operation statement	Multiplications	Adds	Reads	Writes	Iterations
$C \leftarrow 0$	0	0	0	0	0
$for\ i = 0\ to \left\lceil \frac{m}{2D} \right\rceil - 1\ do$	-	-	-	-	-
$C \leftarrow B_i A_L[\frac{m}{2} - 1 : 0] \times 2^{Di} + C$	1	1	2	0	$\left\lceil \frac{m}{2D} \right\rceil$
end for	-	-	-	-	-
$A_L \leftarrow A_L \times 2^{\left\lceil \frac{m}{2} \right\rceil}$	0	0	1	1	0
$for\ i = \left\lceil \frac{m}{2D} \right\rceil\ to\ \left\lceil \frac{m}{D} - 1 \right\rceil\ do$	-	-	-	-	-
$C \leftarrow B_i A_L[m-1:0] + C$	1	1	2	0	$\left\lceil \frac{m}{2D} \right\rceil$
$A_L \leftarrow A_L \alpha^D \bmod p(\alpha)$	0	0	2	1	$\left\lceil \frac{m}{2D} \right\rceil$
end for	-	-	-	-	-
return C mod p(α)	0	0	2	1	0
Total	$\frac{m}{D}$	$\frac{m}{D}$	$\frac{3m}{D} + 3$	$\frac{m}{2D} + 2$	-

That considerable speedup is possibleusing this method.Complexity analysis shows that. In this paper we proposed a novel Hybrid Karatsuba multiplier which uses the best of the Simple and the General Karatsuba algorithms .this result is lesser space requirements on FPGA.

REFERENCES

[1] N. Gura, S. Chang, H. Eberle, G. Sumit, V. Gupta, D. Finchelstein, E. Goupy, and D. Stebila, "An End-to-End Systems Approach to Elliptic Curve Cryptography," Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES 2001), C¸ K. Koc¸ and C. Paar, eds., pp. 351-366, 2001.

[2] N.Koblitz, "Elliptic Curve Cryptosystems," Math. Computation, vol. 48, pp. 203-209, 1987.

[3] N. Koblitz, "A Family of Jacobians Suitable for Discrete Log Cryptosystems," Advances in Cryptology, Proc. Crypto '88, S. Goldwasser, ed., pp. 94-99, 1988.

[4] F. R. Henriquez, et. al., On Fully Parallel Karatsuba Multipliers for GF(2m), Proceedings of the International Conference on Computer Science and Technology, CST 2003, 2003.

[5] G. Orlando and C. Paar, "A High-Performance Reconfigurable Elliptic Curve Processor for GF(2^m)," Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES 2000), C¸ K. Koc¸ and C. Paar, eds., 2000.

[6] G. Orlando and C. Paar, "A Scalable GF(p) Elliptic Curve Processor Architecture for Programmable Hardware," Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES 2001), C¸ K. Koc, D. Naccache, and C. Paar, eds., pp. 348-363, May 2001.

[7] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[8] L. Song and K.K. Parhi, "Low Energy Digit-Serial/Parallel Finite Field Multipliers," J. VLSI Signal Processing, vol. 19, no. 2, pp. 149-166, June 1998.

[9] VLSI Computer Architecture, Arithmetic, and CAD Research Group, Dept. of Electrical Eng., Illinois Inst. of Technology, Chicago, IIT Standard Cells for AMI 0.5_m and TSMC 0.25_m/0.18_m (Version 1.6.0),2003,http://www.ece.iit.edu/vlsi/scells/home.html.

[10] Reza Azarderakhs, and ArashReyhani-Masoleh, ," Efficient FPGA Implementations of Point Multiplication on Binary Edwards and Generalized Hessian Curves Using Gaussian Normal Basis," IEEE Transactions on very large scale integration (vlsi) systems, vol. 20, no. 8, pp.1453-1466 August 2012.

[11] Sandeep Kumar,Thomas Wollinger, and ChristofPaar,"Optimum Digit Serial GF(2^m) Multipliers for Curve-Based Cryptography ", IEEE Transactions on Computer,Vol.55,no.10,pp.1306-1311,October 2006.

[12] Chester Reberio,SujoySinhaRoy,D.Sankara Reddy, and DebdeepMukhopadhyay,"Revisting the Ihho-Tsuji Inversion Algorithm for FPGA Platforms", IEEE Transactions on very large scale integration (vlsi) systems, vol. 19, no. 8, pp.1508-1512 August 2011.

[13] Marcelo E. Kaihara and Naofumi Takagi," Bipartite Modular Multiplication Method" IEEE Transactions on Computers, vol. 57, no. 2, pp.157-164 February 2008.

[14] V.R.Venkatasubramani, M.Surendar, and S.Rajaram,"Novel Methods for Montgomery Modular Multiplicationfor Public Key Cryptosystems"CNSA 2011,CCIS196,pp.320-330,2011.

[15] Chester Rebeiro and DebdeepMukhopadhyay, "Hybrid Masked Karatsuba Multiplier forGF(2^{233})"11th IEEE VLSI Design and Test Symposium, Kolkata, Au*gust 2007*.