Space Optimized Multiplier Architecture for Embedded Cryptoprocessor

Sunil Devidas Bobade
Research Scholar
S.G.B.Amravati University
Amravati, India

Vijay R. Mankar, Ph.D
Deputy Secretary
RBTE
Pune, India

ABSTRACT
The finite field modular multiplier is the most critical component in the elliptic curve crypto processor (ECCP) consuming the maximum chip area and contributing the most to the device latency. Modular multiplication, point multiplication, point doubling are few of the critical activities to be carried out by multiplier in ECC algorithm, and should be managed without compromising on security and without burdening space and time complexities. Since the area complexity of the Crypto processor is mainly based on the Modular Multiplier incorporated within the ECC processor, the major contribution of this work includes the replacement of traditional Karatsuba multiplier with the proposed space optimized multiplier inside the processor The complete modular multiplier and the cryptoprocessor module is synthesized and simulated using Xilinx ISE Design suite 14.4 software. Experimental investigation show an improvement in area efficiency of cryptoprocessor, since proposed scheme occupies relatively reduced percentage area of FPGA as compared to the one using traditional Karatsuba multiplier.

Keywords
ECC: Double point multiplication: Karatsuba Multiplier: Systolic Multipliers: Area Complexity.

1. INTRODUCTION
The Area occupancy of most popular public key algorithm is based on modular arithmetic, where the most critical operation is modular multiplication [1]. Hence by targeting area occupancy of modular multiplier the overall area complexity of cryptoprocessor can be drastically reduced. The main advantage of ECC proposed by Victor Miller and Neal Koblitz is the usage of shorter key with no compromise on security which is at par with RSA. Also, ECC is ideal for hardware implementation as the software counterpart is dead slow. Hence ECC is the best choice for cryptographic hardware implementation and is widely used in embedded system for memory data protection. For ECC a number of hardware implementations have been proposed, and included in many standards such as IEEE 1363and NIST.

Double point multiplication and point doubling are the two critical operations in the ECC algorithm. Figure 1 and figure 2 shows double point multiplication and doubling operation to be carried out in ECC. Modular multiplication is the most primitive and critical operation in ECC. Of the various algorithms, Karatsuba algorithm is known to be the most optimum algorithm available to perform both the multiplication task. So most of the implementations of ECC scheme employ Karatsuba multiplier for the task.

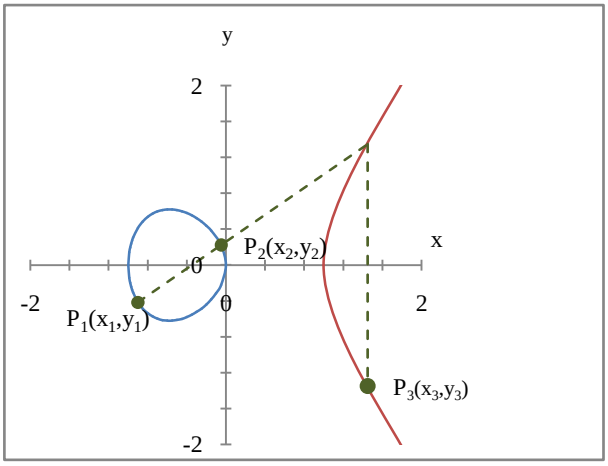

Figure 1. ECC Double point multiplication

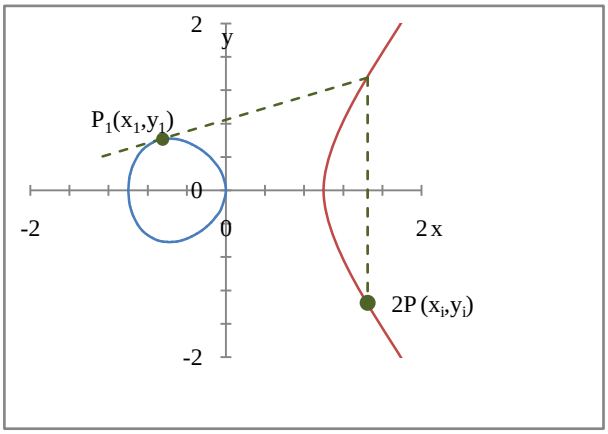

Figure 2. ECC point doubling

The design of Finite field multiplier in cryptoprocessor is the complex design issue in the implementation of the ECC processor. A number of multipliers with different area and time complexity are reported in the available literatures. The Karatsuba algorithm is agreed upon as a most efficient multiplication algorithm and is widely adopted in VLSI implementation of ECC processor. We have investigated the area impact of replacing traditional Karatsuba multipler. Instead of traditional Karatsuba Multiplier, we have replaced it with finite field multiplier which adopts a systolic approach. The space complexity of the resulting multiplier is found to be much better than those of traditional multipliers. This is a significant achievement if we intend to use this cryptoprocessor in embedded systems.

2. RELATED WORK

Literature is a significant treasure house of various VLSI architectures for point multiplication in ECC. Numerous space optimized modular multipliers have been proposed in [2,3]. Of them all, Karatsuba- Ofman algorithm [4] is considered to be highly speed efficient. In [2,4] a variant of Karatsuba multiplier of the type GF((2n)^8) is presented. Ashkan Hosseinzadeh Namin, Huapeng Wu,and Majid Ahmadi proposed a word-level finite field multiplier using normal basis [5]. In [6] Hossein Mahdizadeh and Massoud Masoumi built elliptic curve cryptographic processor by parallelization of the multipliers. Bit-parallel [7], bit-serial [8], digit-serial/parallel [9], multipliers or systolic architectures [10] are the existing solutions that can be classified. Each type of these solutions has different properties: bit-serial are slow but small, bit-parallel solutions are fast but larger. Digit as well as systolic architectures provides some tradeoff between speed and area of the solution

Theoretical modeling of elliptic curve scalar multiplier on LUT-based FPGAs for area and speed was designed by Sujoy Sinha Roy et.al [11]. In this method two primitives used in elliptic curve scalar multiplier architecture (ECSMA) implemented on k input lookup table (LUT)-based field-programmable gate arrays to approximate the delay of different characteristic. It was used to determine the optimal number of pipeline stages and the ideal placement of each stage in the ECSMA. In order to perform point addition and doubling in a pipelined data path suitable scheduling was created. The three stage pipelined architecture for double and add based scalar multiplication is performed on Xilinx Virtex V platforms over GF (2^{163}). The implementation uses a novel pipelined bit-parallel Karatsuba multiplier that has subquadratic complexity. In this design efficient choice of scalar multiplication algorithm, optimized field primitives, balanced pipeline stages, and enhanced scheduling of point arithmetic resulted in a high-speed architecture with a significantly small area.

Hossein Mahdizadeh and Massoud Masoumi [12] designed a novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over GF (2^{163}). In architecture the critical path of the Lopez–Dahab scalar point multiplication architecture was organized and reordered by the maximum architectural and timing improvements, such that logic structures were implemented in parallel and operations in the critical path were diverted to noncritical paths. In the implemented design the execution delay of the LD algorithm has been reduced by parallelization of the multipliers in the implementation of the calculations of projective coordinates

Kazuo Sakiyama et.al [13] implemented a tripartite modular multiplication. In multiplication for maximizing a level of parallelism, systematic approach was implemented for modular multiplication. The algorithm which is used in this method effectively integrates three different existing algorithms they are a classical modular multiplication based on Barrett reduction, the modular multiplication with Montgomery reduction and the Karatsuba multiplication algorithms in order to reduce the computational complexity and increase the potential of parallel processing. In multiprocessor environment for hardware and software implementations this algorithm is very effectively used. In this design a modular multiplier using these algorithms achieves a higher speed comparing to the other algorithms for modular multiplication.

The Massey-Omura multiplier operates in normal basis representations of the field elements. With this representation, the structure of the multiplication becomes highly uniform resulting in efficient hardware architecture. The architecture takes a parallel input but the result is produced serially [14]. Another multiplier based on normal basis is the Sunar-Koç [15] multiplier. The multiplier requires lesser hardware compared to the Massey-Omura multiplier but has similar timing requirements.

Very traditional version of modular multiplier is a Karatsuba Multiplier where the operands 'A' and 'B' are divided into two equal-size parts AL and AH, BL and BH respectively, which represent the m/2 higher and lower order bits of A and B.

$$A(x)=\sum_{i=0}^{m-1}a_i x^i=\sum_{i=\frac{m}{2}}^{m-1}a_i x^i+\sum_{i=0}^{\frac{m}{2}-1}a_i x^i$$

$$B(x)=\sum_{i=0}^{m-1}b_i x^i=\sum_{i=\frac{m}{2}}^{m-1}b_i x^i+\sum_{i=0}^{\frac{m}{2}-1}b_i x^i$$

$$A(x)=x^{\frac{m}{2}}A_H+A_L$$

Similarly, $B(x)=x^{\frac{m}{2}}B_H+B_L$

The product 'C' can be computed as

$$C(x)=A(x)B(x)$$

By using above equations product can be represented as,

$$C(x)=A_L B_L+A_H B_H x^m+(A_H B_L+A_L B_H)x^{\frac{m}{2}}$$

To further improve the computation of the product 'C' equation can be modified as,

$$C(x)=A_L B_L+A_H B_H x^m+$$

$$(A_L B_L+A_H B_H+(A_H+A_L)(B_H+B_L)x^{\frac{m}{2}}$$

From the above equation, the architecture for Karatsuba multiplier is designed as shown in figure 3.

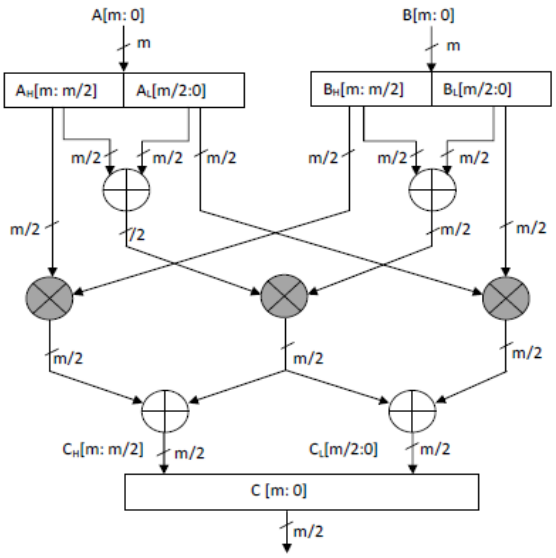

Figure 3. Traditional Karatsuba Multiplier

3. PROPOSED MULTIPLIER DESIGN

The complication of finite field multiplier architecture mostly depends on parameters of the field. These parameters are: illustration of the elements of the field (basis), field, size l and irreducible polynomial $f(x)$ generating the field. The elements are demonstrated by polynomial (standard, canonical) basis, in which they are characterized as binary polynomials. For representing ECC standards, the two parameters (field size l and irreducible polynomial $f(x)$) along with the multiplier incorporated in the processor are chosen, since the proposed ECC processor should occupy less resource.

Digit as well as systolic architectures provides some tradeoff between speed and area of the solution. It is observed that it may be inefficient, especially time inefficient, when we apply it to large polynomials and perform operation bit by bit (bit-serial). But alternatively, it may be area inefficient, if operation is performed on all bits parallelly. Instead of processing vector (polynomial) bit by bit or parallelly, process it in 16-bit words. This 16-bit seems to be the appropriate size according to the research conducted [16]. After executing vast research [17] on the best word size and although for field sizes: 409 or 571 a 32-bit word seems more suitable, research shows that in most cases 16-bit word gives enhanced results. Thus for such partitioning, proposed multiplier perform $\left\lceil \frac{l}{16} \right\rceil - 1$ operations.

Let $\alpha(x), \beta(x)$ be two polynomials of degree l for which we have to compute the finite field multiplication product.

$$\alpha(x) = \sum_{k=0}^{l-1} \alpha_k x^k = \alpha_0 x^0 + \alpha_1 x^1 + ... + \alpha_{l-1} x^{l-1}$$

$$\beta(x) = \sum_{k=0}^{l-1} \beta_k x^k = \beta_0 x^0 + \beta_1 x^1 + ...\beta_{l-1} x^{l-1}$$

$f(x)$ let be an irreducible polynomial of degree l, where $f(x) = x^l + \sum_{k=0}^{l-1} f_k x^k$

Then the product of the two polynomials $\gamma(x)$ is given by

$$\gamma(x) = \alpha(x).\beta(x) \bmod f(x)$$

$$\chi(x) = \alpha(x).\beta(x) = \alpha(x)(\sum_{k=0}^{15} \beta_k x^k) + \alpha(x)(\sum_{k=16}^{31} \beta_k x^k) + ...+ \alpha(x)(\sum_{k=16(\lceil \frac{l}{16} \rceil - 1)}^{l-1} \beta_k x^k)$$

The above computation consists of two steps

1) Multiplication of the polynomial coefficients $((\chi(x) = \alpha(x).\beta(x))$

2) Reducing the result obtained using irreducible polynomial $(\bmod f(x)).\ \chi(x)$ is given as,

The modification of above equation is given as,

$$\chi(x) = (\sum_{k=0}^{15} \alpha_k x^k)(\sum_{k=0}^{15} \beta_k x^k) + (\sum_{k=16}^{31} \alpha_k x^k)(\sum_{k=16}^{31} \beta_k x^k) + ...+ (\sum_{k=16(\lceil \frac{l}{16} \rceil - 1)}^{l-1} \alpha_k x^k (\sum_{k=16(\lceil \frac{l}{16} \rceil - 1)}^{l-1} \beta_k x^k)$$

Let

$$\alpha^{[k]} \Rightarrow \sum_{k=16(\lceil \frac{l}{16} \rceil - 1)}^{l-1} \alpha_k x^k \text{ and } \beta^{[k]} \Rightarrow \sum_{k=16(\lceil \frac{l}{16} \rceil - 1)}^{l-1} \beta_k x^k$$

Then,

$$\chi(x) = \alpha^{[0]} \beta^{[0]} + \alpha^{[1]} \beta^{[1]} + \alpha^{[2]} \beta^{[2]} + ...+ \alpha^{\lceil \frac{l}{16} \rceil - 1} \beta^{\lceil \frac{l}{16} \rceil - 1}$$

Proposed Multiplier processes the two words and generates the product using modular arithmetic.

$Input \quad : A(\alpha_0\ \alpha_1\ \alpha_2 .....\alpha_{l-1}),$
$B(\beta_0\ \beta_1\ \beta_2 .....\beta_{l-1}) \in G(2^m),$
$A and\ B \neq 0,$
$F(1\ f_1\ f_2 ..... f_l)$
$Output \ : \Gamma(\gamma_0\ \gamma_1\ \gamma_2 .....\gamma_{l-1})$
1. $Initialize\ \mathrm{X}\ and\ \Gamma$
2. $For\ k = 0\ to\ l-1$
3. $For\ s = 0\ to \lceil l/16 \rceil - 1$
4. $A_s \leftarrow \alpha_0 - \alpha_{\lceil l/16 \rceil - 1}$
   $B_s \leftarrow \beta_0 - \beta_{\lceil l/16 \rceil - 1}$
5. $\mathrm{X}_s \leftarrow A_s \times B_s$
   $\mathrm{X}_{s+1} \leftarrow \mathrm{X}_s + \mathrm{X}_{s-1}$
6. $end\ for$
7. $end\ for.$
8. $\Gamma \leftarrow \mathrm{X}_s \bmod F$
9. $return\ \Gamma$

Figure 4, shows the finite field multiplier architecture. Initially the inputs α and β are partitioned into 16-bits using the shift registers. Then the 16-bit word is directly fed to the T block and the other input of the T block is connected to the input α through a single bit bus. T block incorporates an array of tri-state buffers . Each bit of input β is fed as input to the tri-state buffer and the single bit from input α controls the buffer. For instance if the single bit of α is 1 then the 16-bits from the input β is allowed to the k bits

left shifter block or else denied. The k bits left shifter shifts the output bits from the T block with respect to the k value. The shifted outputs are then XORed using the XORing logic block. The XORed output for the first 16-bits is stored in the χ register which is then updated when the next set of values is processed. Once the process as discussed above is completed for the l bit inputs the final value in the χ register is fed to the $\text{mod } f(x)$ computation block and the final finite field product is γ is obtained.

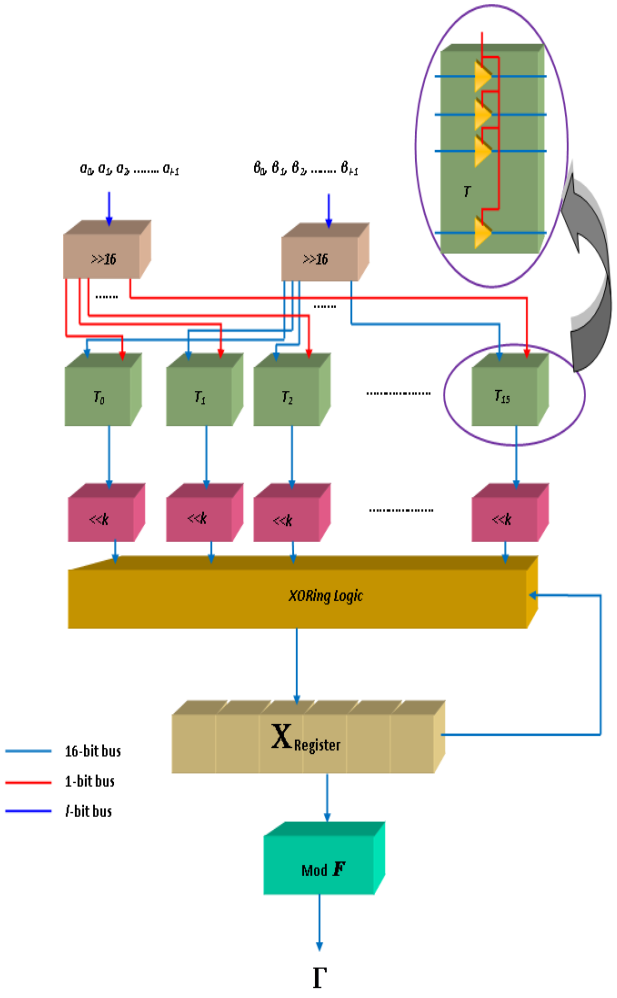

Figure 4. Proposed Finite Field Multiplier

4. RESULTS AND DISCUSSIONS

In this section, focus is on the FPGA implementations of the cryptoprocessor using proposed and Karatsuba multiplier and then comparing area footprints of two cryptoprocessors. The architecture is coded in verilog HDL and is synthesized using Xilinx ISE version 14.4 design software and is implemented on Xilinx Virtex-4 xc4vlx200ff1513 FPGA. The RTL schematic for the implemented Finite Field multiplier is shown in figure 5.

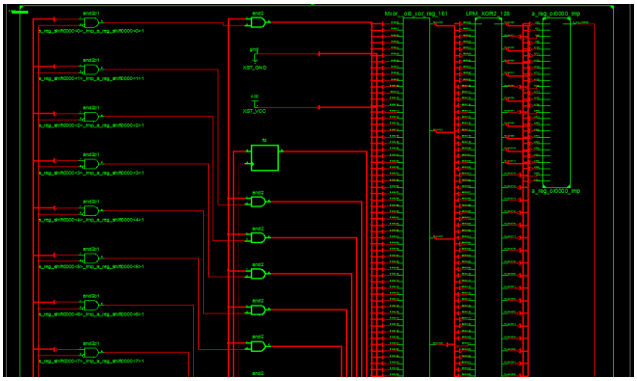

Figure 5. RTL Schematic for Proposed modular Multiplier

The design is simulated for using ISE Simulator tool present in the Xilinx software and the simulation output is as shown in figure 6.

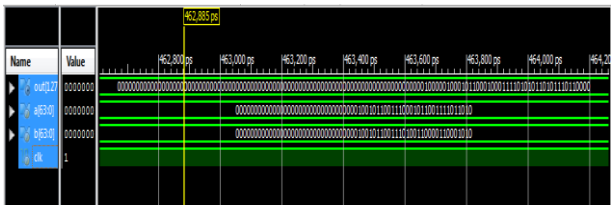

Figure 6. Simulation Results for proposed modular multiplier

4.1 Power Requirement

Power consumed by proposed Modular Multiplier is reported in Table 1 below. The clock frequency is varied between 100 MHz to 1800 MHz and the corresponding power consumed is plotted in figure 7.

Table 1: Power Report for the proposed modular multiplier

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.842	0.037	0.805
Vccaux	2.500	0.150	0.000	0.150
Vcco25	2.500	0.003	0.000	0.003
		Total	Dynamic	Quiescent
Supply Power (W)		1.392	0.044	1.347

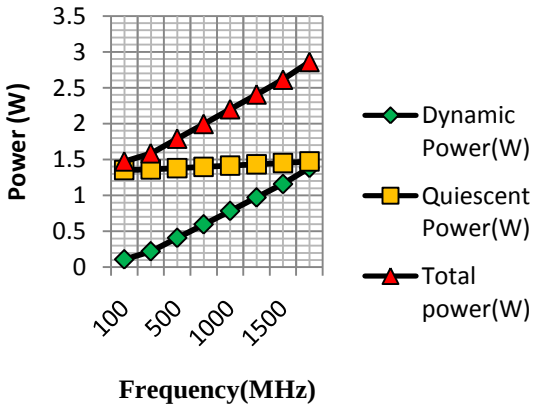

Figure 7. Power consumption of Proposed Multiplier

4.2 Multiplier Area Report

Since the area of the complete processor mainly depends on the incorporated GF multiplier, most of the slices in the target device are utilized by it. From table 2, for 256 bit multiplicands, proposed modular Multiplier needs 374 slices out of 89,088 available slices in the target device. Among the 178,176 available four input LUTs only 583 are used. The multiplier also needs only 467 out of 178,176 Flip Flops.

Table 2: Device utilization Summary for the two multipliers

	Karatsuba Multiplier			Proposed Multiplier		
M size	Flip flops	LUTs	Slices	Flip flops	LUTs	Slices
2	227	98	124	240	118	127
4	236	171	168	265	153	146
8	242	316	237	307	248	191
16	245	426	292	387	482	307
32	525	739	473	467	580	374
64	534	723	469	467	597	381
128	620	851	576	467	599	382
256	579	1535	916	467	583	374

Thus proposed Modular multiplier as compared in figure 8, exhibits a savior of 23.88% in terms of Flip flop slices. Proposed Multiplier involves 62 % fewer LUTs and utilizes 59% fewer slices as compared to traditional Karatsuba multiplier for 256 bit multiplication.

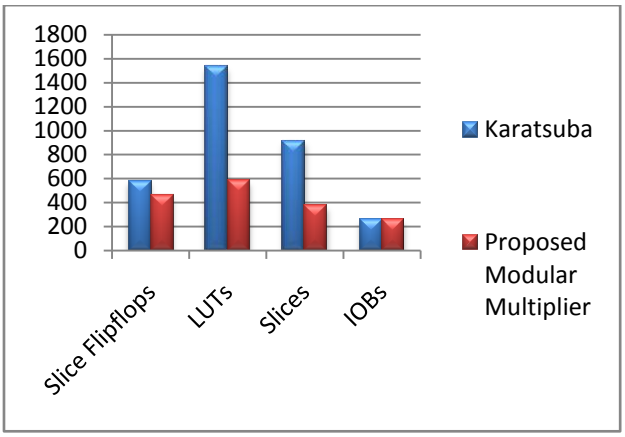

Figure 8. Resource utilization summary for two multipliers for m=256

4.3 Cryptoprocessor Area Report

The crypto processor implementation uses a double point multiplication algorithm proposed in [18] and area comparision is carried out for cryptoprocessors using proposed multiplier and Karatsuba multiplier and the one proposed in [19].

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Total Number Slice Registers	383	178,176	1%
Number used as Flip Flops	381		
Number used as Latches	2		
Number of 4 input LUTs	412	178,176	1%
Number of occupied Slices	308	89,088	1%
Number of Slices containing only related logic	308	308	100%
Number of Slices containing unrelated logic	0	308	0%
Total Number of 4 input LUTs	456	178,176	1%
Number used as logic	378		
Number used as a route-thru	44		
Number used as Shift registers	34		
Number of bonded IOBs	255	960	26%
Number of BUFG/BUFGCTRLs	1	32	3%
Number used as BUFGs	1		
Number of DSP48s	7	96	7%
Average Fanout of Non-Clock Nets	1.79		

Figure 7. Device Utilization Summary for Cryptoprocessor using Proposed Modular Multiplier

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Total Number Slice Registers	485	178,176	1%
Number used as Flip Flops	483		
Number used as Latches	2		
Number of 4 input LUTs	386	178,176	1%
Number of occupied Slices	406	89,088	1%
Number of Slices containing only related logic	406	406	100%
Number of Slices containing unrelated logic	0	406	0%
Total Number of 4 input LUTs	488	178,176	1%
Number used as logic	352		
Number used as a route-thru	102		
Number used as Shift registers	34		
Number of bonded IOBs	247	960	25%
Number of BUFG/BUFGCTRLs	1	32	3%
Number used as BUFGs	1		
Number of DSP48s	22	96	22%
Average Fanout of Non-Clock Nets	1.52		

Figure 8. Device Utilization Summary for Cryptoprocessor using Karatsuba Multiplier

The device utilization area report as in Figure 9 depicts that hardware implementation for the ECC processor employing proposed multiplier utilizes 308 out of 89,088 available slices in the target device and 456 out of 178,176 available LUTs. Among the 383 of 178,176 Slice registers present, 381 act as Flip-flops and two of them are used as latch. 7 out of 96 DSP-48 blocks are utilized by implemented architecture of complete ECC processor.

The device utilization area report as in Figure 10 depicts that hardware implementation for the ECC processor employing traditional Karatsuba multiplier utilizes 406 out of 89,088 available slices in the target device and 488 out of 178,176 available LUTs. Among the 485 of 178,176 Slice registers present, 483 act as Flip-flops and two of them are used as latch. 22 out of 96 DSP-48 blocks are utilized by ECC processor.

The comparison of implemented ECC processor employing proposed modular multiplier and the cryptoprocessor using traditional Karatsuba Multiplier for 16 bit operation with respect to the area occupied (Slice registers,Slices, LUTs and IOBs) is tabulated in Table 3 and compared in figure 11. Proposed implementation utilize about 21.03 % reduced register slices, 24.13% reduced slices and 6.56 % LUTs. This is a significant achievement as ECC cryptoprocessors are widely used for security services in embedded system which is known to be highly resources thirsty.

Table 3: Device Utilization Summary for the two implemented Cryptoprocessors

Cryptoprocessor type	Size	Slices	LUTs	IOBs
Using Karatsuba Multiplier	233	406	488	247
Using proposed Multiplier	233	308	456	255
A.Kaleel Rahuman et.al [19]	193	466	932	210

Comparison of built cryptoprocessor with similar reported ECC cryptoprocessor [19] shows improvement in terms of area occupancy. The experimental results when compared with the device utilization reported in similar work showed that architecture for ECC processor using proposed multiplier needs about 19.31% reduced slices ,42.48% reduced LUTs and 72.21% IOBs. Hence proposed hardware implementation show an efficiency in terms of the area utilization and a tradeoff between area-performance, an added advantage.

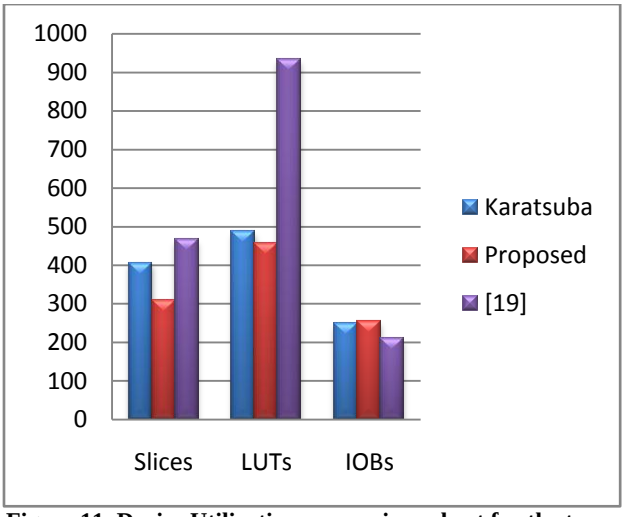

Figure 11. Device Utilization comparison chart for the two implemented Cryptoprocessors with [19]

5. CONCLUSION

We have proposed a novel method to implement the modular multiplier for ECC cryptography. Analysis suggests that using a proposed multiplier in ECC cryptoprocessor instead of traditional Karatsuba Multiplier eventually helps in reducing space complexity of cryptoprocessor. This is a significant achievement if this multiplier is to be employed in VLSI implementation of elliptical curve cryptography for the protection of memory in embedded systems. Hardware implementation of cryptoprocessor shows an efficiency in terms of the area utilization.

6. REFERENCES

[1] Blake, I. F., Seroussi, G., and N. P. *Elliptic curves in cryptography*. Cambridge University Press, New York, NY, USA, 1999.

[2] C. Paar. *Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields*. PhD thesis, University at GH Essen, VDI Verlag, 1994.

[3] B. Sunar and C¸. K. Koc¸. Mastrovito multiplier for all trinomials. *IEEE Transactions on Computers*, 48(5):522–527, May 1999.

[4] C. Paar. A new architecture for a parallel finite field multiplier with low complexity based on composite fields. *IEEE Transactions on Computers*, 45(7):856–861, July 1996.

[5] Ashkan Hosseinzadeh Namin, Huapeng Wu, and Majid Ahmadi, "A word-level finite field multiplier using normal basis", IEEE Transactions on computers, Vol. 60, No. 6, pp: 890- 895, Jun. 2011

[6] Hossein Mahdizadeh and Massoud Masoumi, "Novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over GF(2^{163})", IEEE Transactions on very large scale integration (vlsi) systems, Vol. 21, NO. 12, pp: 2330- 2333, Dec.2013

[7] Y. I. Cho, N. S. Chang, C. H. Kim, Y.-H. Park, and S. Hong, "New Bit Parallel Multiplier With Low Space Complexity for All Irreducible Trinomials Over," IEEE Trans. VLSI Syst., vol. 20, no. 10, pp. 1903–1908, Oct. 2012.

[8] M. Morales-Sandoval, C. Feregrino-Uribe, and P. Kitsos, "Bit-serial and digit-serial GF (2^m) Montgomery

multipliers using linear feedbackshift registers," Computers Digital Techniques, IET, vol. 5, no. 2, pp.86–94, Mar. 2011.

[9] A. Hariri and A. Reyhani-Masoleh, "Digit-Serial Structures for the Shifted Polynomial Basis Multiplication over Binary Extension Fields," WAIFI 2008, LNCS 5130. Springer, pp. 103–116, Jul. 2008.

[10] J. Lin, "Low-latency Digit-serial Systolic Double Basis Multiplier over GF (2^m) using Subquadratic Toeplitz Matrix-vector Product Approach," IEEE Trans. Comput., vol. PP, no. 99, p. 1, 2012

[11] Roy. S.S, Rebeiro,C and Mukhopadhyay. D, "Theoretical modeling of elliptic curve scalar multiplier on LUT-based FPGAs for area and speed", IEEE Transactions on Very Large Scale Integration (VLSI) systems, Vol. 21, No. 5, May 2013.

[12] Hossein Mahdizadeh and Massoud Masoumi, "A novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over GF(2^{163})",IEEE Transactions on very large scale integration systems, Vol. 21, No. 12, Dec 2013

[13] Kazuo Sakiyama, Miroslav Knezevica, Junfeng Fana, , Bart Preneela, and Ingrid Verbauwhedea, "Tripartite modular multiplication", Integration, the VLSI Journal, Vol. 44, No.4, pp: 259–269, September 2011.

[14] Gregory C. Ahlquist, Brent E. Nelson, and Michael Rice, "Optimal Finite Field Multipliers for FPGAs," in FPL '99: Proceedings of the 9th International Workshop on Field-Programmable Logic and Applications, London, UK, 1999, pp. 51–60, Springer-Verlag.

[15] Ç. K. Koç and B. Sunar, "An Efficient Optimal Normal Basis Type II Multiplier," *IEEE Trans. Comput.*, vol. 50, no. 1, pp. 83–87, 2001.

[16] Y. I. Cho, N. S. Chang, C. H. Kim, Y.-H. Park, and S. Hong, "New Bit Parallel Multiplier With Low Space Complexity for All Irreducible Trinomials Over," IEEE Trans. VLSI Syst., vol. 20, no. 10, pp. 1903–1908, Oct. 2012.

[17] M. Morales-Sandoval, C. Feregrino-Uribe, and P. Kitsos, "Bit-serial and digit-serial GF (2^m) Montgomery multipliers using linear feedbackshift registers," Computers Digital Techniques, IET, vol. 5, no. 2, pp.86–94, Mar. 2011

[18] Reza Azarderakhsh and Koray Karabina,"A new double point multiplication algorithm and its application to binary elliptic curves with endomorphisms", IEEE Transactions on Computers, No.99, May 2013.

[19] A.Kaleel Rahuman and G.Athisha, "Reconfigurable Architecture for Elliptic Curve Cryptography Using FPGA", Hindawi Publishing Corporation Mathematical Problems in Engineering, 2013