5th International Conference on Emerging Trends in Engineering, Technology, Science and Management
Institution of Electronics and Telecommunication Engineers, Ganganagar, Bengaluru, Karnataka, India (ICETETSM-17)
20th August 2017, www.conferenceworld.in ISBN: 978-93-86171-60-3

A Light Weight Implementation of ECC Cryptosystem on FPGA for nodes in Wireless Sensor Networks

Leelavathi G[1], shaila K[1], Venugopal K R[2]

[1] *Vivekananda Institute of Technology, Bengaluru,Karnataka,India*

[2] *University College of Engineering, Bengaluru,Karnataka,India*

ABSTRACT

The ECC Cryptosystem is implemented on FPGA for sensor nodes that are used in Wireless Sensor Networks in which nodes are FPGA based. Scalar or Point multiplication, the most important operation of ECC is carried out with double and add algorithm which is implemented using Karatsuba Multiplier. The Karatsuba Multiplier designed for basic finite field multiplication operations at lower levels of design increased the overall performance of the cryptosystem in terms of area, speed, operating frequency and power consumption. The processor is optimized for scalar multiplications with Lopez Dahab and Mixed coordinate systems. This work concentrates on lightweight implementation of ECC that is suitable for Wireless Sensor Nodes. The previous implementations mainly concentrates only on Point multiplication where as in our work we have implemented complete ECC cryptosystem with 40% less device utilization on FPGA Artix-7xc7a100t-2csg324 and the speed achieved is 432MHz.

Keywords: Elliptic curve cryptography, FPGA, Karatsuba Multiplier, Point Multiplication, Wireless Sensor Networks

I. INTRODUCTION

The special needs of Wireless Sensor Networks (WSNs) such as security issues, communication protocols and hardware platforms, require an intense research activity. Different applications employed on the same WSNs environment will have diverse security requirements, inferring the necessity of using dissimilar security algorithms. Encryption is a sensible countermeasure to protect data, although it increases new processing load to the nodes. Conventional microcontrollers found sensor nodes do not provide adequate computation power to process public key cryptographic operations [1][2]. The practice and potentials of FPGAs in sensor node architectures and their applications are explored in [2].

Neal Koblitz and Victor Miller proposed Elliptic Curve Cryptography (ECC) in the year 1985. At present, ECC is the most efficient and preferred PKC system with shorter keys. The security is to concentrate the difficulty of solving Elliptic Curve Discrete Logarithmic Problem (ECDLP)[3]. ECC is attaining popularity since it provides similar security with significantly smaller key lengths. This feature makes it suitable for resource constrained devices like wireless sensor nodes.

Motivation: Now-a-days Field Programmable Gate Arrays are crossing the 28 nanometer CMOS threshold that offer enhanced platforms which adds additional processing competences to a typical sensor node. Due to inadequate resources of the sensor nodes security is considered to be expensive, the speed and area are more notable in WSNs. Thus, in order to increase the speed and minimize area, while utilizing the available resources, it is important to use multiplier which has better performance with respect to security algorithms. Karatsuba multiplier has been employed to allow the encryption and decryption of messages to speed up ECC algorithm with minimum device utilization [2-4].

Organization: In Section II, research works related to security techniques, public key cryptography and ECC cryptography, different multiplication algorithms are explained. In Section III, implementation is described with Karatsuba multiplier techniques, ECC Cryptography along with Algorithm and Performance Evaluation. Section IV contains Conclusions.

II. EXPERIMENTAL AND COMPUTATIONAL DETAILS

The modules discussed in this section are designed and implemented in VHDL code and tested on ISE 13.2 software using XILINX FPGA Artix-7 xc7a100t-2csg324 device and simulated with Modelsim Simulator. This FPGA is advised for implementation of Public Key Cryptography as it supports different degrees of security and high-speed execution for GF computations. The static power consumption is very less compared to other FPGAs [2].

2.1 Elliptic Curve Cryptography

In the latter decade, the method of hardware implementation of ECC algorithm distinguished a much focused competition, due to essentiality of the constraints *i.e.*, security, speed and area. The performance of an ECC algorithm depends competently on the arithmetic in the fundamental Galois Field (GF). The Galois Fields are prime order fields GF (p) or Characteristic two fields or Binary fields GF (2^m). Both provides equivalent level of security, GF (2^m) is simple and can be implemented in hardware. In the GF, addition and subtraction are equivalently achieved by modulo-2 arithmetic.

An Elliptic Curve over the Binary field GF (2^m) considered is

$$y^2+x*y=x^2+ax^2+b \qquad (1)$$

Where, a, b $\in$ GF (2^m), $b\neq0$. Set of points on E (GF (2^m)) also comprises point O, which is point at infinity.

Point on the elliptic curve is a couple of elements x, y $\in$ GF (2^m) that fulfill P=(x, y) $\in$ E (GF (2^m)) of equation (1).

The input data considered in this work is 8 bits. It is encoded to 192 bit point of the elliptic curve *p(x, y)*. It is projected as *p(x, y, z)* before performing encryption. Encryption is carried out with 192 bit point in projective *p(x, y, z)* to produce cipher text, which double secures the data. Decoding is performed at decryption end to get back the original text of 8 bits. The point addition and point doubling are performed with different coordinate systems and are discussed in the following section.

2.2 Field Arithmetic and Coordinate Systems

In affine coordinate system a point on the elliptic curve is represented as *p(x, y)*. To perform field arithmetic operations i.e., point doubling and point addition, field division is required in affine coordinate system. To avoid this time consuming operation, projective coordinate systems are adopted for point doubling and point addition operation which significantly affects the performance of scalar multiplication in turn the whole ECC cryptographic operation [3].

The equation (1) in projective coordinates is given by,

$$y^2z+xyz=x^3+ax^2z+bz^3 \qquad (2)$$

Projective coordinates encompass representing a curve point as triplet x,y,z $\in$ E (GF (2^m)), *i.e., p(x, y, z)*. When comparing with different popular projective coordinate systems, the computational cost of Lopez and Dahab (LD) projective coordinate is found to be lesser and most logical for hardware implementations. In this work, Scalar multiplication operations are performed with point doubling in LD and point addition in mixed coordinates.

Point Doubling: It is the addition of a point *p(x, y)* on elliptic curve to itself to attain a new point *q(x, y)* on the same curve. LD projective coordinates are employed for point doubling in which *p(x, y)* is projected to $p(x_1, y_1, z_1)$. The output is obtained as *q(x2, y2, z2)* is performed with following equations:

$$x_2=(3x_1^2+az_1^4)-8y_1^2x_1 \qquad (3)$$

$$y_2=(3x_1^2+ az_1^4)(4x_1^2y_1^2-x_3)-8y_1^4 \qquad (4)$$

$$z_2=2y_1z_1 \qquad (5)$$

Point Addition: It is addition of two points' *p(x, y)* and *q (x, y)* on an elliptic curve to get another point *r(x, y)* on the same elliptic curve. Mixed coordinates is utilized, where one point (*x*), is in affine and one more point (*y*) in LD projective coordinate. The normal elliptic point *(x, y)* is projected to *p(x1, y1, z1)* and the other point is in affine *i.e., p(x2, y2)*. Point addition to obtain *p(x3, y3, z3)* is performed with following equations:

$$x_3=(y_2z_13-y_2)^2-(x_2z_1^2-x_1)^2(x_1+x_3z_1^2) \qquad (6)$$

$$y_3=(y_2z_1^3-y_2)-x_1(x_1z_1^2-x_1)2-y_1(x_2z_1^2-x_1)^3 \qquad (7)$$

$$z_3=(x_2z_1^2-x_1)z_1 \qquad (8)$$

The multiplications involved in Point doubling and Point addition are performed using Karatsuba multiplier which comprises three types of computation divide, conquer and combine [4]. Number of finite field operations required for Group operations are given in Table 1.

Table 1. Finite Field Operations.

Group Operations	Squaring	Addition	Multiplication
Point Addition	3	7	7
Point Doubling	5	5	8

2.3 Karatsuba Multiplier

The effective implementations of finite field operations can be established by performing additions, multiplications and inversion that greatly impacts the performance of ECC Cryptosystem. There are several methods to implement the finite field multiplication, out of which Karatsuba algorithm is termed as a fast multiplication algorithm published in the year 1962. In conventional multiplication method, multiplying two integers of *n*-bits takes $O(n^2)$ bit operations. Numerous algorithms are available to improve $O(n^2)$ multiplication. Karatsuba multiplication algorithm uses divide and conquer technique and requires $O(n^{\log 3})$ bit operations, to multiply two integers of *n*-bits. Karatsuba algorithm accomplishes multiplication operation by substituting some multiplications with subtraction and addition operations which are less expensive. Due to recursion overhead Karatsuba algorithm is slower than classical multiplication for small inputs and it is more proficient for large numbers. Karatsuba multiplier is used as the finite field multiplier in point addition and doubling modules [4].

2.4 Scalar Point Multiplication

Scalar point multiplication is the process of computing $Q = k.P$, where $Q, P \in E(GF(2^m))$ and *k* is a scalar value. It is performed by repetitive point additions and doubling which is the basic cryptographic operations.

There are numerous algorithms for performing Elliptic Curve Scalar Point Multiplication. Scalar Point Multiplication performed in the design is given in Algorithm 1. The multiplication module used [5] in our design is shown in the Figure 1. Basically, it is double and add method, built on the binary expansion of the integer *k*. In this algorithm point doubling is performed in each iteration of *k*. In any iteration, if the particular bit of *k* is 1, then a point addition is also performed [4].

Input to the module is point $P(x, y, z)$ and scalar *k*, where, *k* has a size of *t* bits. The output obtained after multiplication is *K.P*. Multiplication of a scalar value *K* with a point *P* and *2P* output from point doubling block is used for further cascading of modules. Multiplication module used in our implementation is shown in the Fig.1. The module consists of both cascaded point doubling (PD) and point addition (PA) blocks. A point *P* is multiplied in every stage to produce *2P, 3P, 4P…., 2t-1, 2tP* [5].

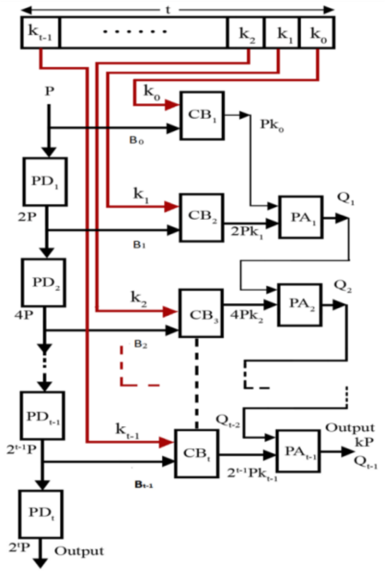


Figure 1: Scalar Point Multiplication Module[5].

```
Algorithm 1: Scalar Point Multiplication
Input: Point on the curve P(x, y, z),
t-bit integer $K = \sum_{i=0}^{t-1} K_i 2^i, K_i \in \{0,1\}$
Output: Point on the Curve $Q(x_o, y_o, z_o)$= K*P(x, y, z)
$Q(x_o, y_o, z_o)$= P(x, y, z) // Q=P
For i=t-1 to 0 do
P(x,y,z)=point double(x,y,z) //P=2P
If $K_i$=1 then //P=P+Q
P(x,y,z)=pointadd($x_o$, $y_o$, $z_o$, x, y, z)
End if
End for
Return $Q(x_o, y_o, z_o)$
```

The Scalar Point Multiplication is represented as *Q* = *K.P*. It is given by,

$$K*P(x, y, z) = (2^{t-1} K_{t-1}+2^{t-2} K_{t-2}+\ldots+2^2 K_2+2 K_1+ K_0 \qquad (9)$$

$$K*P(x, y, z) = (2^{t-1} P(x, y, z)K_{t-1}+2^{t-2} P(x, y, z)K_{t-2}+\ldots+2^2 P(x, y, z)K_2+2 P(x, y, z)K_1+ K_0 \quad P(x, y, z) \qquad (10)$$

Where, *P*(*x*, *y*, *z*) is a Point on the Elliptic Curve represented as projective point in design. *K.P* with respect to the Figure 1 is given as:

$$K * P(x,y,z) = \sum_{i=0}^{t-1} 2^i P(x,y,z)\, K_i = \sum_{i=0}^{t-1} B_i K_i \qquad (11)$$

Where, $B_i = 2^i P(x, y, z)$ for *i* = 0, 1, 2 …*t*-1.

The Controlled Buffers (CB) generate $2^i Pk_i$ for i=0 to (t–1).The bit k_i of K and 2^i P are the inputs to the respective CB. The output of each CB is one of the inputs to the corresponding Point Adder block. Point addition in every stage depends on the value of k bit. If k=1, present value of CB and previous value of CB is added. If k=0, then only previous value of the controlled block is added with next stage of CB. The output $t2^tP$ from the last PD block is shown, where 2^tP provides cascading with other modules.

2.5 Encryption and Decryption

2.5.1 Key Generation. In Elliptic Curve Diffie-Hellman Protocol, the elliptic curve equation and the base point *P(x,y)* and are made public. Private key of User A is k_A which is a scalar value, multiplied to point *P(x,y)* to generate public key. User A's public key is equal to k_A.* *P(x,y)*. Similarly, private key of User B, is k_B and k_B*P(x,y)* is user B's public key. The shared secret key (SSK) between two parties A and B is easily calculated by

$$SSK(x,y)= k_A(k_B*P(x,y))= k_B (k_A * P(x,y)) \qquad (12)$$

The Processes involved in Scalar Point Multiplication is represented in Fig 2.

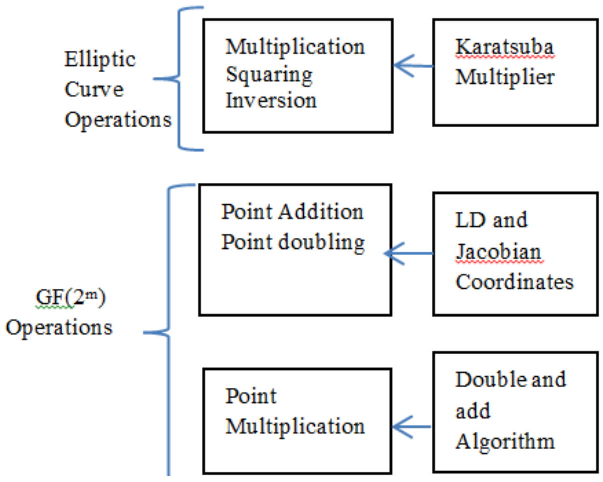

Elliptic Curve Operations
Multiplication Squaring Inversion
Karatsuba Multiplier
GF(2m) Operations
Point Addition Point doubling
LD and Jacobian Coordinates
Point Multiplication
Double and add Algorithm

Figure 2: Processes Involved in Point Multiplication

2.5.2 Encryption. The message to be encoded is embedded into the x-coordinate of a point on the elliptic curve *(Pm (xm, ym)).* While user A or User B is sending the message to the other, shared secret key has to be added to the message *M(x,y)* to produce the Cipher text *C(x,y).*

C(x,y)=M(x,y)+SSK(x,y) (13)

2.5.3 Decryption. In decryption the shared secret key is subtracted from the Cipher text *C(x,y)* to obtain plaintext message *M(x,y).*

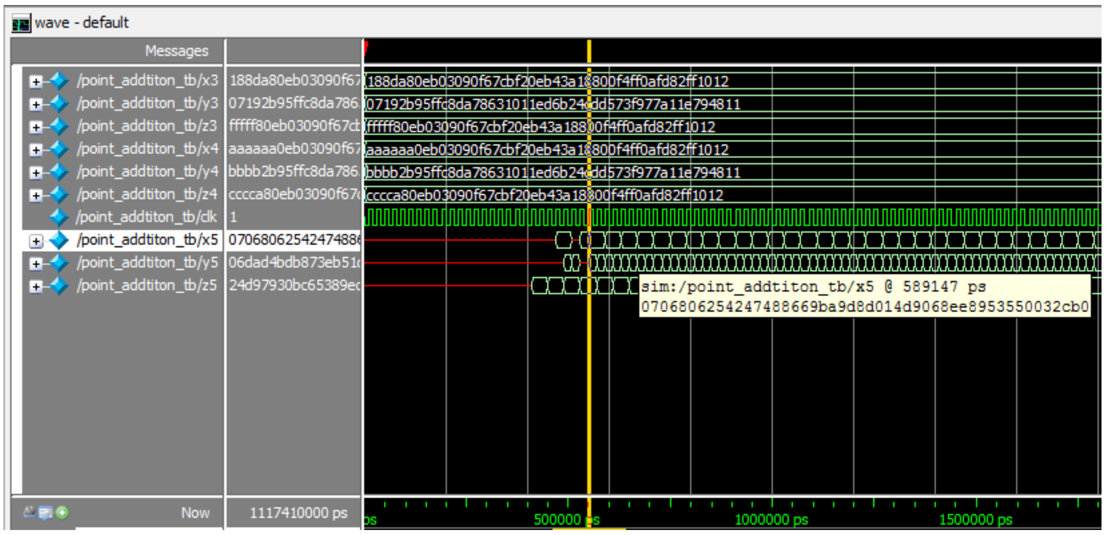
M(x,y)= C(x,y)- SSK(x,y)

(14)

III. RESULTS AND DISCUSSION

3.1 Point Addition and Point Doubling

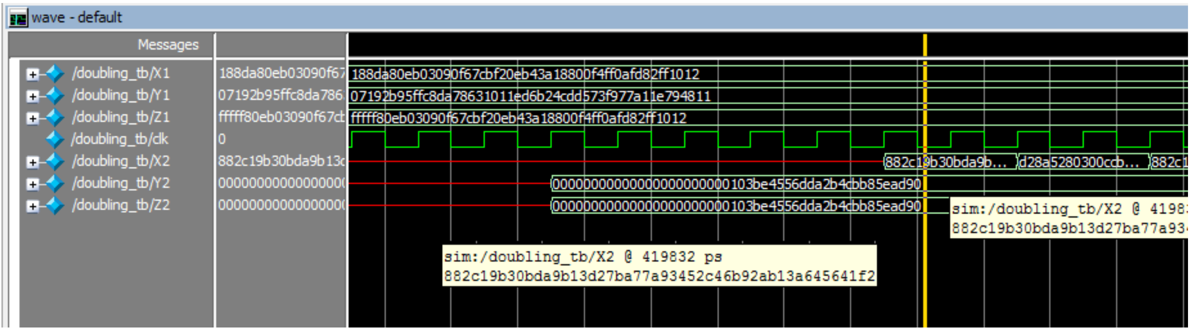
The Point addition operation output is shown in Fig.3, displays a point *p(x3, y3, z3)* added to *p(x4, y4, z4)* to produce *p(x5, y5, z5).*



```
wave - default
Messages
/point_addtiton_tb/x3   188da80eb03090f67   188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
/point_addtiton_tb/y3   07192b95ffc8da786   07192b95ffc8da78631011ed6b24cdd573f977a11e794811
/point_addtiton_tb/z3   fffff80eb03090f67cb fffff80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
/point_addtiton_tb/x4   aaaaaa0eb03090f67   aaaaaa0eb03090f67cbf20eb43a18800f4ff0afd82ff1012
/point_addtiton_tb/y4   bbbb2b95ffc8da786   bbbb2b95ffc8da78631011ed6b24cdd573f977a11e794811
/point_addtiton_tb/z4   cccca80eb03090f67c  cccca80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
/point_addtiton_tb/clk  1
/point_addtiton_tb/x5   0706806254247488
/point_addtiton_tb/y5   06dad4bdb873eb51
/point_addtiton_tb/z5   24d97930bc65389e
sim:/point_addtiton_tb/x5 @ 589147 ps
0706806254247488669ba9d8d014d9068ee8953550032cb0
Now  1117410000 ps
ps  500000 ps  1000000 ps  1500000 ps
```

Figure 3: Timing Diagram of Point Addition Operation

Point doubling operation output with timing diagram is shown in Fig 4. The point considered for doubling is *p*(*x1*, *y1*, *z1*) and the output point obtained is *p*(*x2*, *y2*, *z2*).



```
wave - default
Messages
/doubling_tb/X1   188da80eb03090f67   188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
/doubling_tb/Y1   07192b95ffc8da786   07192b95ffc8da78631011ed6b24cdd573f977a11e794811
/doubling_tb/Z1   fffff80eb03090f67ct fffff80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
/doubling_tb/clk  0
/doubling_tb/X2   882c19b30bda9b13c   882c19b30bda9b...  d28a5280300ccb...  882c1
/doubling_tb/Y2   0000000000000000(   000000000000000000000000103be4556dda2b4cbb85ead90
/doubling_tb/Z2   0000000000000000(   000000000000000000000000103be4556dda2b4cbb85ead90

sim:/doubling_tb/X2 @ 419832 ps
882c19b30bda9b13d27ba77a93452c46b92ab13a645641f2

sim:/doubling_tb/X2 @ 4198
882c19b30bda9b13d27ba77a93
```

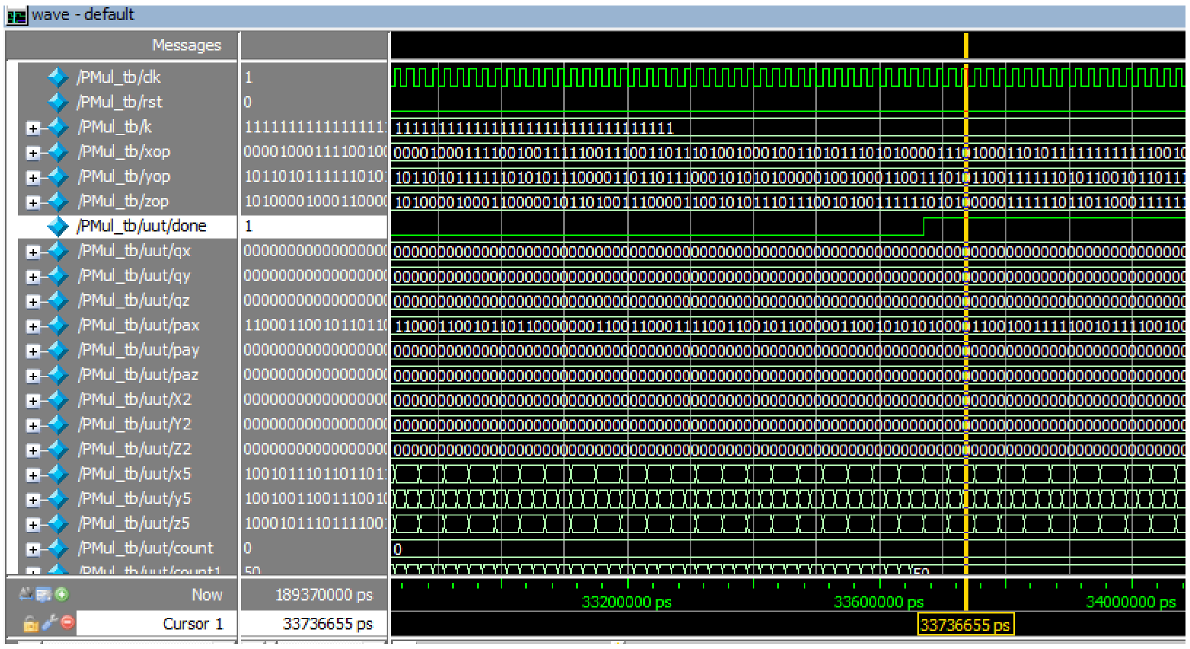
Figure 4: Timing Diagram of Point Doubling Operation

3.2 Point Multiplication (Scalar Multiplication)

The point multiplication module performs the major operation in ECC. This module includes point addition and doubling operations. Key generation for sender and receiver is performed using the point multiplication. Fig 5 shows synthesized result for Point Multiplication.

3.3 Karatsuba Multiplier

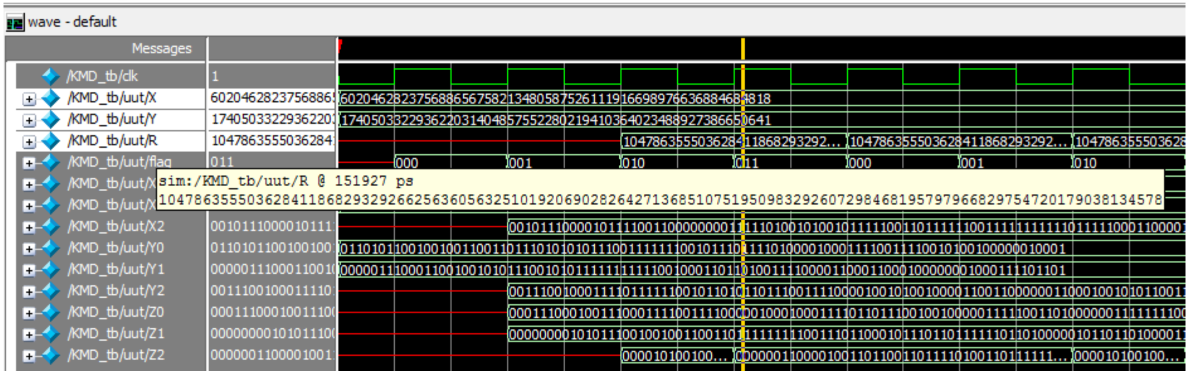
The Karatsuba Multiplier plays an important role in multiplying two 192 bit data in point addition and point doubling operations. The timing diagram with results is shown in Fig 6. The two input numbers are *X*, *Y* and *R* is the result.



```
wave - default
Messages
/PMul_tb/clk          1
/PMul_tb/rst          0
/PMul_tb/k            1111111111111111
/PMul_tb/xop          0000100011110010(
/PMul_tb/yop          1011010111111010
/PMul_tb/zop          1010000100011000(
/PMul_tb/uut/done     1
/PMul_tb/uut/qx       0000000000000000(
/PMul_tb/uut/qy       0000000000000000(
/PMul_tb/uut/qz       0000000000000000(
/PMul_tb/uut/pax      1100011001011011(
/PMul_tb/uut/pay      0000000000000000(
/PMul_tb/uut/paz      0000000000000000(
/PMul_tb/uut/X2       0000000000000000(
/PMul_tb/uut/Y2       0000000000000000(
/PMul_tb/uut/Z2       0000000000000000(
/PMul_tb/uut/x5       1001011101101101
/PMul_tb/uut/y5       1001001100111001(
/PMul_tb/uut/z5       1000101110111100
/PMul_tb/uut/count    0
/PMul_tb/uut/count1   50

Now        189370000 ps     33200000 ps     33600000 ps     34000000 ps
Cursor 1   33736655 ps      33736655 ps
```

Figure 5: Timing Diagram of Point Multiplication Operation



```
wave - default
Messages
/KMD_tb/clk        1
/KMD_tb/uut/X      6020462823756886
/KMD_tb/uut/Y      1740503322936220
/KMD_tb/uut/R      1047863555036284
/KMD_tb/uut/flag   011
/KMD_tb/uut/X
/KMD_tb/uut/X
/KMD_tb/uut/X2     001011100001011
/KMD_tb/uut/Y0     011010110010010
/KMD_tb/uut/Y1     000001110001100
/KMD_tb/uut/Y2     001110010001111
/KMD_tb/uut/Z0     000111000100111
/KMD_tb/uut/Z1     000000001010111
/KMD_tb/uut/Z2     000000110000100

sim:/KMD_tb/uut/R @ 151927 ps
10478635550362841186829329266256360563251019206902826427136851075195098329260729846819579796682975472017903813457 8
```

Figure 6: Results of Karatsuba Multiplier

3.3. Encryption and Decryption

The Encryption module encrypts the plain text *M*(*x*, *y*) to Cipher text *C*(*x*, *y*). The synthesized result is shown in the Fig 7 and Figure 8 shows RTL schematic and simulation result respectively. The original text *i.e.*, 8bit data (00010101) and cipher text *C*(*x*, *y*) 192 bit are shown in the Fig 8.

The decryption module decrypts the cipher text *C*(*x*, *y*) back to plain text of 8 bit value. The synthesized result is shown in Fig 9. From the cipher text *C*(*x*, *y*) the original text of 8 bit (00010101) is recovered.

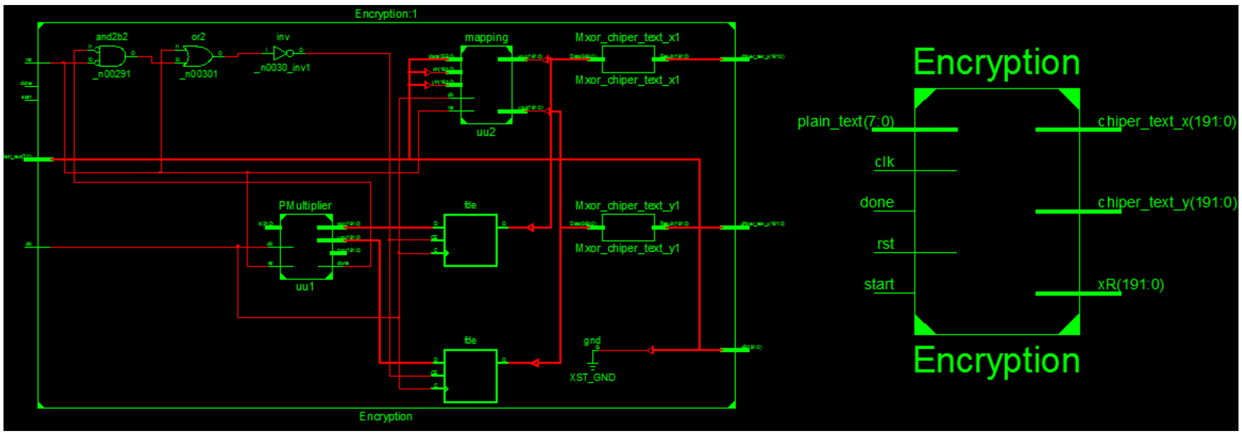
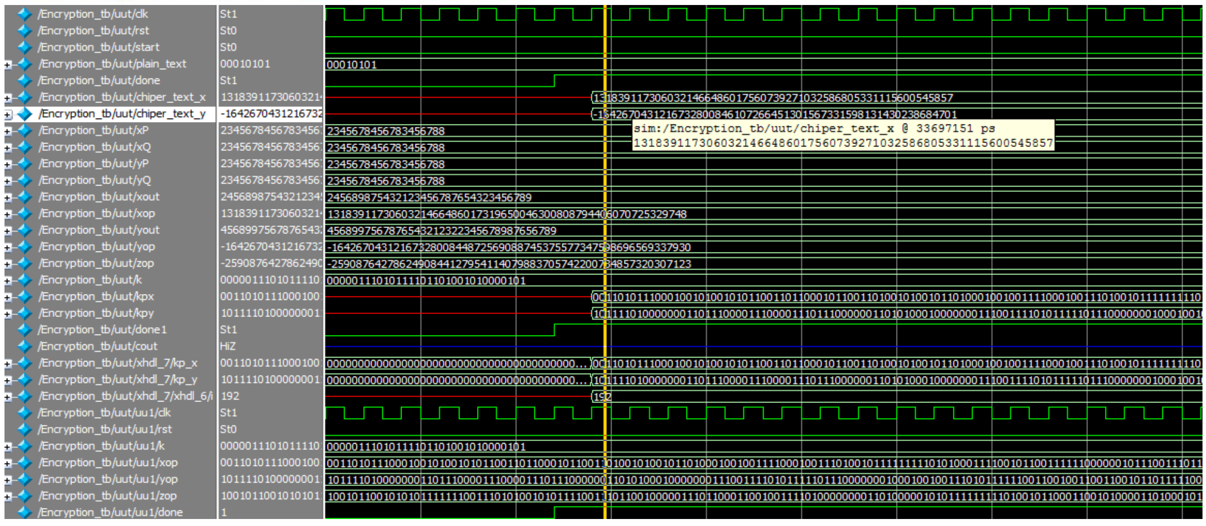


Figure 7: Schematic of Encryption Module

3.3 Analysis of the ECC Cryptosystem

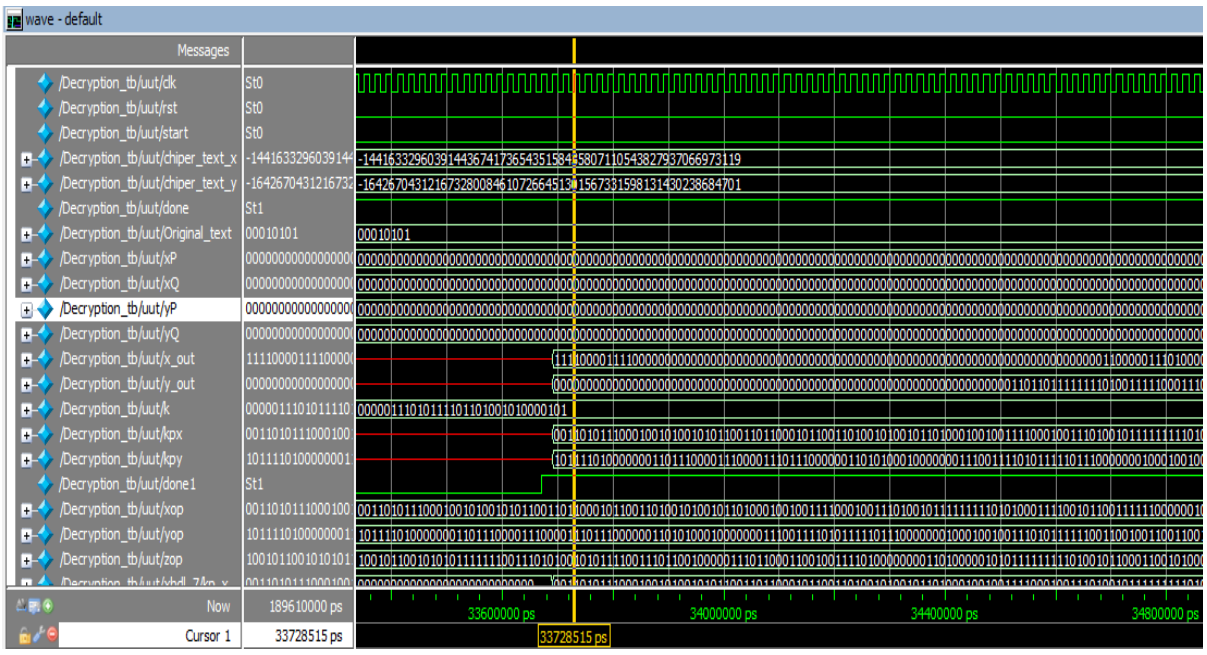
William et al., [6] designs a pipelined ASIP for ECC using FPGA, XCV2600E-FG1158-8 the hardware (area) consumed by point multiplier and ASIP are compared with our work. In [7] Khaleel et al., presents Folded Modular Multiplier with Virtex-4 as target device. Comparing the time delay obtained by our work with [7] is Point doubling (2.610µsec), Point Addition (4.720 µsec) and Point Multiplication (1.078 msec) shows that the computation is performed faster and there is no combinational path delay.



```
/Encryption_tb/uut/clk                St1
/Encryption_tb/uut/rst                St0
/Encryption_tb/uut/start              St0
/Encryption_tb/uut/plain_text         00010101
/Encryption_tb/uut/done               St1
/Encryption_tb/uut/chiper_text_x      1318391173060321
/Encryption_tb/uut/chiper_text_y      -1642670431216732
/Encryption_tb/uut/xP                 2345678456783456
/Encryption_tb/uut/xQ                 2345678456783456
/Encryption_tb/uut/yP                 2345678456783456
/Encryption_tb/uut/yQ                 2345678456783456
/Encryption_tb/uut/xout               2456898754321234
/Encryption_tb/uut/xop                1318391173060321
/Encryption_tb/uut/yout               4568997567876543
/Encryption_tb/uut/yop                -1642670431216732
/Encryption_tb/uut/zop                -2590876427862490
/Encryption_tb/uut/k                  0000011101011110
/Encryption_tb/uut/kpx                0011010111000100
/Encryption_tb/uut/kpy                1011110100000001
/Encryption_tb/uut/done1              St1
/Encryption_tb/uut/cout               HiZ
/Encryption_tb/uut/xhdl_7/kp_x        0011010111000100
/Encryption_tb/uut/xhdl_7/kp_y        1011110100000001
/Encryption_tb/uut/xhdl_7/xhdl_6/i    192
/Encryption_tb/uut/uu1/clk            St1
/Encryption_tb/uut/uu1/rst            St0
/Encryption_tb/uut/uu1/k              0000011101011110
/Encryption_tb/uut/uu1/xop            0011010111000100
/Encryption_tb/uut/uu1/yop            1011110100000001
/Encryption_tb/uut/uu1/zop            1001011001010101
/Encryption_tb/uut/uu1/done           1

sim:/Encryption_tb/uut/chiper_text_x @ 33697151 ps
13183911730603214664860175607392710325868053311115600545857
```

Figure 8: Simulation Results of Encryption and Decryption



```
wave - default
Messages
/Decryption_tb/uut/clk                St0
/Decryption_tb/uut/rst                St0
/Decryption_tb/uut/start              St0
/Decryption_tb/uut/chiper_text_x      -144163329603914
/Decryption_tb/uut/chiper_text_y      -1642670431216732
/Decryption_tb/uut/done               St1
/Decryption_tb/uut/Original_text      00010101
/Decryption_tb/uut/xP                 0000000000000000
/Decryption_tb/uut/xQ                 0000000000000000
/Decryption_tb/uut/yP                 0000000000000000
/Decryption_tb/uut/yQ                 0000000000000000
/Decryption_tb/uut/x_out              1111000011110000
/Decryption_tb/uut/y_out              0000000000000000
/Decryption_tb/uut/k                  0000011101011110
/Decryption_tb/uut/kpx                0011010111000100
/Decryption_tb/uut/kpy                1011110100000001
/Decryption_tb/uut/done1              St1
/Decryption_tb/uut/xop                0011010111000100
/Decryption_tb/uut/yop                1011110100000001
/Decryption_tb/uut/zop                1001011001010101
Now        189610000 ps     33600000 ps   34000000 ps   34400000 ps   34800000 ps
Cursor 1   33728515 ps      33728515 ps
```

Figure 9: Simulation Results of Decryption module

Kaleel et al., [8] implements ECC Cryptosystem using Xilinx XC4VLX200 FPGA device. According to Portilla et al., [9] the FPGA based ECC implementation requires less energy compared to a low power microcontroller implementation. Benaissa et al., [10] extended GPP based implementation to ASIPs by developing novel word level algorithms for multiplication and squaring. Azarderaksh et al., [11] use a number of digit serial field multipliers to attain fast scalar multiplication.

The hardware required for implementation for Point multiplication in our work is 40% less than previous implementations. Our experimental results of device utilization are compared with [6-13] in Table 2 and Table 3. There is no combinational path delay for Group and Elliptic Curve Operations which increases the speed of Encryption and Decryption Process. Most of the previous works concentrates on the Point Multiplication rather than complete ECC system. In this work complete ECC Cryptosystem with 40% less device utilization is achieved with a speed of 432MHz.

Table 2: Comparison of Device Utilization for Point Multiplier

Author	Field(m)	FFs	LUTs	Slices
William[6]	163	4686	26930	15368
Chelton[6]	163	7962	26364	16209
Rebeiro[12]	233	-	37073	19209
Azarderaksh	163	-	22815	122834
Sujoy[13]	163	-	10195	3513
Hossein[14]	163	-	30895	16544
Our Work	192	504	9788	690

Table 3: Comparison of Device Utilization for ECC

Author	Field(m)	Device	LUTs	Frequency
Xining	128	XC4VLX60	15168	125 MHz
Kaleel [8]	256	XC4VLX20	16209	143 MHz
Our Work	192	XC7A100T	9915	432 MHz

IV. CONCLUSIONS

We have performed both Point Multiplication and ECC on Artix7 FPGA that is advised platform for Public Key Cryptography. The experimental results show that the area utilized by both Point Multiplier and ECC is around 40% less than the previous implementations. Hence, this work suitable for resource constrained devices like Wireless Sensor Nodes.

REFERENCES

1. Akyildiz, I., F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: A Survey. In: ACM Communications, 38(4), pp. 393-422. (2002)
2. Antonio de la Piedra, An Braeken, AbdellahTouhafi.: Sensor Systems Based on FPGAs and their Applications: A Survey. In: Journal of Sensors (2012), 12, 12235-12264; DOI:10.3390/s 120912235. (2012)
3. Gustavo, D., Sutter, Jean-Pierre Deschamps, Jose Luis Imana.: Efficient Elliptic Curve Point Multiplication Using Difit-Serial Binary Field Operations. In: IEEE Transactions on Industrial Electronics. Vol.60, no.1, pp.217-225; DOI: 10.1109/TIE.2012.2186104. (2013)
4. Can Eyupoglu.: Performance Analysis of Karatsuba Multiplication algorithm for Different Bitlengths. In: Proceedings of the World Conference on Technology, Innovation and Entrepreneurship. ELSEVIER, Procedia-Social Sciences 195(2015), pp. 1860-1864.(2015)
5. Shylashree, N., Sridhar, V.,: Hardwar Realization of High Speed Elliptic Curve Point Multiplication. In:

Journal of Computer Science. Vol.10, no.7, pp.1094-1106; DOI: 10.3844/jcssp.2014.1094.1106. (2014)

6. William, N., Chelton, Mohammed Benaissa,: Fast Elliptic Curve Cryptography on FPGA. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 16(2):198-205; DOI: 10.1109/TVLSI.2007.912228. (2008)
7. Osama AL-Khaleel, Chris Papachristou, Francis Wolf and Kiamal Pekmestzi. 2007. An Elliptic Curve Cryptosystem Design Based on FPGA Pipeline Folding. in 13th IEEE International Conference on Online Testing Symposium. DOI: 10.1109/IOLTS.2007.15.
8. Kaleel Rahuman, A., Athisha G.: A Reconfigurable Architecture for Elliptic Curve Cryptography Using FPGA. vol. 2013, Article ID 675161, 8 pages. DOI:http://dx.doi.org/10.1155/2013/675161. (2013)
9. Portilla, J., Otero, A., de la Torre, E., Riesgo, T., Steckina, O., Peter, S., Langendorfer, P.: Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors. In: International Journal of Distributed Sensor Networks. Vol. 2010, Article ID 740823, 12 pages. DOI:10.1155/2010/740823. (2010)
10. Mohammed Benaissa, Wei Ming Lim,.: Design of Flexible GF(2m) Elliptic Curve Cryptography Processors. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Vol.14, n0.6, pp.659-662; DOI: 10.1109/TVLSI.2006. 878235.(2006)
11. Azarderakhsh, R., Reyhani-Masoleh, A.: Efficient FPGA Implementations of Point Multiplication on Binary Edwards and Generalized Hessian Curves using Gaussian Normal Basis. In: IEEE Transactions on VLSI systems. Vol.16, no.99. (2011)
12. Rebeiro, C., Mukhopadhyay, D.: High Speed Compact Elliptic Curve Cryptoprocessor for FPGA Platforms. In: Proceedings of 9th International Conference on Cryptology. pp. 376-388. (2008)
13. Sujoy Sinha Roy, Chester Rebeiro, Debdeep Mukhopadhyay,. : Theoretical Modeling of Elliptic Curve Scalar Multiplier on LUT-Based FPGAs for Area and Speed. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Vol.21, no.5, pp.901-909. DOI: 10.1109/TVLSI.2012.2198 502. (2013)
14. Hossein Mahdizadeh, Massoud Masoumi,.: Novel Architecture for Efficient FPGA Implementation of Elliptic Curve Cryptographic Processor Over GF(2^{163}). In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Vol.21, no.12, pp.2330-2333; DOI: 10.1109/TVLSI.2012.2230410. (2013)
15. Xining Cui, Jingwei Yang,.: An FPGA based Processor for Elliptic Curve Cryptography. In: Proceeding of International Conference on Computer Science and Information Processing (CSIP), pp. 343-349. (2012)