IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 8, Issue 1, Ver. I (Jan.-Feb. 2018), PP 01-15
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design of RSA Processor and Field Arithmetic of ECC with Vedic Multipliers for Nodes in Wireless Sensor Networks

Leelavathi G, Shaila K, Venugopal K R
Electronics and Communication EngineeringVTU-Research Centre, Vivekananda Institute of Technology, Bengaluru, India
PrincipalUniversity Visvesvaraya College of Engineering,IEEE Fellow, Bengaluru, India
Corresponding Author: Leelavathi G

Abstract:*In Wireless Sensor Nodes due to the resource constraintsthe fast multipliers are preferred for data processing. In this paper, the RSA processor using Vedic multiplication technique is proposed which is capable of achieving considerable speed and with minimum area utilization. The multiplication of two prime numbers is implemented using Nikhilam and Urdva-tiryagbhyam multipliers.The results show that there is good improvement in delay and device utilization usingUrdvaTriyagbagam method. Urdva-tiryagbhyamis utilized in Point addition and Point doubling, which are finite field arithmetic of ECC in both prime and binary field. Multipliers are implemented on RSA and ECC over NIST/SECG GF(p) and* GF (2^M) *curves and estimates the algorithms with respect to performance in speed and memory usage.*

Keywords: *Elliptic Curve Cryptography, FPGA,NikhilamMultiplier, RSA algorithm, Urdva-tiryagbhyam Multiplier, Wireless Sensor Networks.*

Date of Submission: 05-01-2018 Date of acceptance: 03-02-2018

I. Introduction

The increase in application of Wireless Sensor Networks (WSNs) in environments like home, military and commercialrequiresspeeding up of the data processing in the network [1].Cryptography is a practice for building the message secure with encryption and decryption processes.ECC and RSA are the public-key algorithms that have been investigated by the researchcommunal for several years. The RSA was developed by Rivest, Shamir and Adleman in 1976.Koblitz and Miller available work on ECC in 1985. Modular exponentiationis the major operation underlying in RSA and its security isin question due to its difficulty of factoring large integers. ECC operates on groups of points over elliptic curves and originates its security from the hardness of the elliptic curve discrete logarithm problem (ECDLP). The Public Key Cryptography (PKC)involves cryptographic processes which are computationally intensive and sensor nodes are strictly resource constrained. In encryption systems multipliers play important role. High speed systems with low power consumption and time delay mainly depends on multiplier execution time. Compared to conventional multiplication, Vedic technique of multiplication involves very less number of operations resulting in fasterand high performance multiplier[2-4]. Compared to RSA, ECCbetter suited for WSNs, since ECC offers smaller key sizes, performs faster computation, as well as memory, energy and bandwidth are saved.

The construction of Vedic multipliers is based on Vedic Sutras. The Vedic mathematics has been distributed into sixteen different Sutras and their methods are comparable to the working of human mind which is capable of condensing the complex calculations into simpler ones, consuming less power and attains lower chip area[4].In this work, the multiplication is computed using NikhilamandUrdva-Tiryagbhayam multiplication methodologies for RSAalgorithm that improves the performance in terms of area and speed.Nikhilamincludes minimum number of steps, space, time saving and cerebral calculation.*Urdhva – Tiryagbhayam*formula is appropriate to all cases of multiplication and division of a large number by another large number.This is based on "Vertically and Crosswise" technique andcreates almost all the numeric computations faster and easier. The benefit of multiplier developed on this sutra is that with the rise in the number of bits, area and delay increase at a smaller rate.So this Sutra is used for design of field arithmetic'sof ECC, Point addition and Point doubling.

***Motivation*:**Security is generally expensive and thespeed is more obviousin WSNs dueto limited resources of the sensor nodes. In orderto increase the speed while appropriately utilizingthe available resources, it is important to use multipliers which have better performance with respect to security algorithms. Vedic multipliers allow the encryption and decryption of messages to speed up RSA algorithm and finite field arithmetic of ECC.

***Organization*:**The security techniques, public key cryptography and RSA cryptography, different multiplication algorithms are described in Section II. In Section III,Problem definition, Processor Model,Implementationof Vedic multiplier techniques,RSA Cryptography together with Algorithm and Performance Evaluation are discussed. Section IV comprisesConclusions.

II. Related Work

Shaila et al., [2-3]designs a key predistribution scheme for WSNs. Tiwari et al., [5]proposed Vedic multiplier constructed on algorithm of ancient Indian Vedic Mathematics, for low power and high speed applications and is implemented on ALTERA Cyclone –II FPGA.Kunchugi et al.,[6]discussed the efficiency of UrdhvaTriyagbhyam Vedic method for multiplication that takes11 logic cells for nibble multiplier and propagation time of 4.585ns. Based on the formulas of ancient Indian Vedic Mathematics, a novel complex number multiplier ASIC design that are highly suitable for high speed complex arithmetic circuits is discussed in [7][8].

Hudder et al.,[9]presented a novel architecture to accomplish high speed multiplication using compressors with ancient Vedic mathematical techniques. Kumaravel et al.,[10] examines modular multiplication using Vedic mathematics that increases the speed of performance. The multipliers are implemented with Karastuba-ofman and Booth algorithm considering delay as a parameter. In [11-13], authors implemented Vedic multiplier and compared their work with Booth and array multipliers.

Pritam et al., [14] proposed algorithms based on 1's complement subtraction which will remarkably improve the computational efficiency of scalar multiplication. Soramet al., [15-17]experimentallyevaluated the performance of RSA and ECC. Shaila et al., [18] discussed about constraints of WSNs and proposed a scheme called Modified Blooms Scheme (MBS).Thapiyal et al., [19] utilizes Vedic Mathematics to implement Elliptic Curve Encryption that speeds up the task of multiplication process.Houssain et al., [14] elaborated on the study of hardware implementations of ECC in Wireless Sensor Networks. Both Binary and Primary field are used for the implementation of point addition and point doubling for ECC.Our proposed work provides moderate level of security for resource constrained devices.

III. Problem Definition And Model

A. *Problem Definition*

In a given WSNs, encryption of data is performed before it is transmitted. This is achieved by using multiplication techniques.Multipliers are the main source in high speed arithmetic logic units. A particular multiplication technique in encryption algorithm is not able to deliver all the desired performance properties for nodes in WSNs, so Public-key cryptography is adopted.Public-key cryptography iscomputationally expensive for WSNs if not accelerated by cryptographic hardware.

B. *Objective*

The main objective of this work is to design and develop efficient multipliers that increase the performance of RSA and ECC processor with Vedic multiplication methods.

C. *Algorithm for RSA Processor*

The complete process involves three phases.

Phase 1. Key generation

a. Selection of prime numbers p, q
b. Computation of$n = p * q$ …………………*(1)*
c. Computation of$phi = (p - 1) * (q - 1)$……*(2)*
d. Choose public key e,satisfying the condition
$1 < e < phi,$ gcd$(e, phi) = 1$…………… *(3)*
e. Finding the private key d,
$d * e = 1 mod(phi)$…………………….*(4)*
f. $Private\ key = (d, n)$
g. $Public\ key = (e, n)$

Phase 2. Encryption

$C = M^{e} mod\ n$…………………………………….*(5)*

Where C is cipher text and M is plaintext

Phase 3. Decryption

$M = C^{d} mod\ n$……………………………….…*(6)*

*D. **ECC Arithmetic***

ECC is computationallyintensive since it includes arithmetic operations in finite fields. An elliptic curve cryptosystem functions in a group of points on an elliptic curve defined over a finite field. To improve performance most practical ECC implementations are binary extension fields GF(2^m) and prime fields GF(p). The prime field types permit efficient software implementation, particularly on processors furnished with a fast integer multiplier. For hardware implementation, binary extension fields GF(2^m) are commonly the superior choice.

Due to the computational expenditure of inversion associated to multiplication, projective coordinate methods have been projected which circumvents the inversion operation. Point arithmetic based on the Lopez - Dahab coordinate(LD) is effective for hardware implementation.In this work, Scalar multiplication operations are to be carried out with point adding in mixed coordinates *i.e.,* one point is in affine and another point is in L-D projective coordinate and point doubling in L-D projective coordinates in the binary field.Both point addition and doubling operations are foundation of scalar multiplication.The representation considered in this paper is a normal base that is most suitable for hardware implementation. An elliptic curve over GF (2^M) is defined as the cubic equation E:

$$y^2 + xy = x^3 + ax^2 + b \qquad (7)$$

Where a, b, x, y $\in$ GF (2^M) and b $\neq$ 0.
The Weiestrass equation defining an elliptic curve over GF(p):

$$y2 \text{ (mod p)} = (x3+ax+b) \text{ mod p} \qquad (8)$$

Where x, y are defines elements of GF(p), and a, b are integer modulo p,

IV. RSA Processor

*E. **RSA Processor- Design Unit***

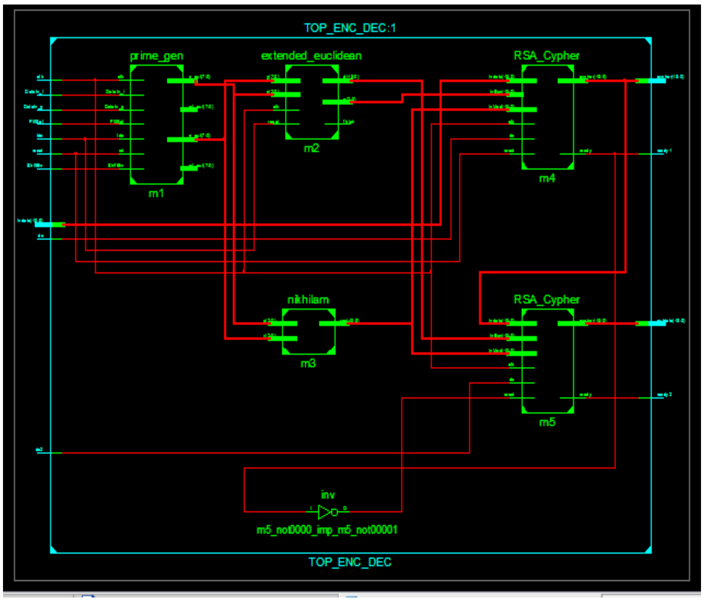


Fig.1 Schematic Diagram of theComplete Processor

Figure 1 shows the complete model in which both private key and public keys are generated. The public key of the sender is distributed over the network that is utilized for decryption process. The encryption process uses the private key of the sender, provides more security for data through digital signature and authentication.

The RSA processor shown in Figure 1 is modeled using Verilog and synthesized on FPGA device Spartan 3 XC3S400-5pq208 for software and hardware implementation.The encryption block (RSA_Cipher m4) computes cipher $C=M^e$ *mod n*. The message (*M*)in binary form is taken as input along with public key (*e*) and modulus *(n*). The decryption block (RSA_Cipher m5) decodes user message (*M*) from cipher (*C*) by computing$M=C^d$ *mod n*, where,dis the private key. Hence, the original message which is in the binary format is recovered safely.

F. Simulation and Hardware Results

Simulation results of Complete RSA processor is illustrated in this section. The primality tester outputs for *p,q*are shown in Figure 2.

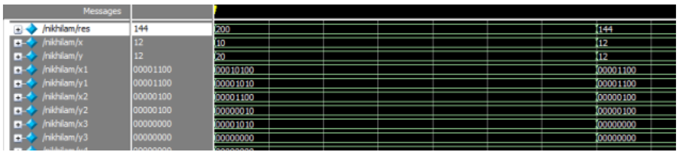


Fig 3. Result of Nikhilam Multiplier

Nikhilam and Urdhva – Tiryagbhayam multiplier results with 8 bit data is shown in Figure 3 and Figure 4.

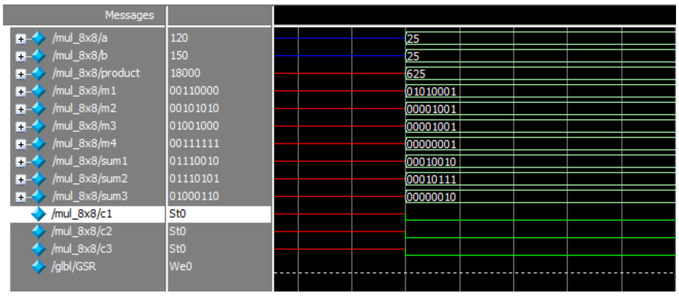


Fig 4. Results of Urdhva – Tiryagbhayam

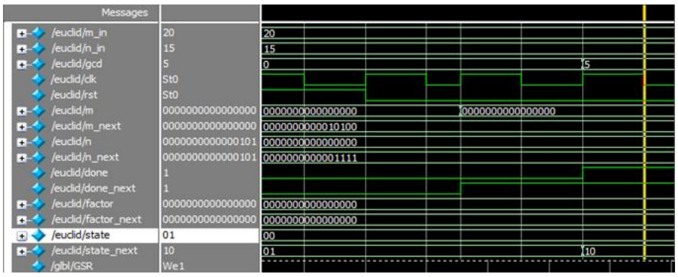


Fig 5. Extended Euclidean algorithm output for GCD

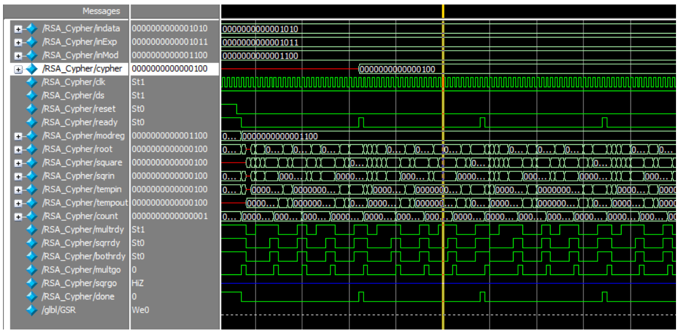
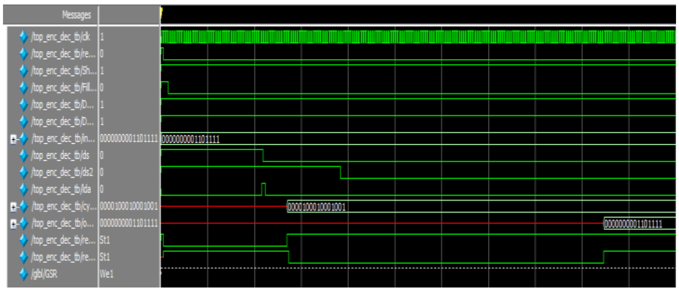

Messages
/RSA_Cypher/indata 0000000000001010
/RSA_Cypher/inExp 0000000000001011
/RSA_Cypher/inMod 0000000000001100
/RSA_Cypher/cypher 0000000000000100
/RSA_Cypher/clk St1
/RSA_Cypher/ds St1
/RSA_Cypher/reset St0
/RSA_Cypher/ready St0
/RSA_Cypher/modreg 0000000000001100
/RSA_Cypher/root 0000000000000100
/RSA_Cypher/square 0000000000000100
/RSA_Cypher/sqrin 0000000000000100
/RSA_Cypher/tempin 0000000000000100
/RSA_Cypher/tempout 0000000000000100
/RSA_Cypher/count 0000000000000001
/RSA_Cypher/multrdy St1
/RSA_Cypher/sqrrdy St0
/RSA_Cypher/bothrdy St0
/RSA_Cypher/multgo 0
/RSA_Cypher/sqrgo HiZ
/RSA_Cypher/done 0
/glbl/GSR We0
0000000000001010
0000000000001011
0000000000001100
0000000000000100
0000000000001100

Fig 6. Simulation Results of RSA Processor

The Euclid algorithm gives the proof that the GCD of encryption key and *phi* in the RSA algorithm is one *i.e.*,Euclid algorithm takes the two input values and calculates its gcd, if the gcd is one it gives the output '*1*' else it gives the gcd of two numbers but it discards the output. The results are provided in Figure 5.



```
Messages
/top_enc_dec_tb/clk        1
/top_enc_dec_tb/re...      0
/top_enc_dec_tb/Sh...      1
/top_enc_dec_tb/Fill...    0
/top_enc_dec_tb/D...       1
/top_enc_dec_tb/D...       1
/top_enc_dec_tb/in...      0000000001101111   0000000001101111
/top_enc_dec_tb/ds         0
/top_enc_dec_tb/ds2        0
/top_enc_dec_tb/lda        0
/top_enc_dec_tb/cy...      0000100010001001   0000100010001001
/top_enc_dec_tb/o...       0000000001101111   0000000001101111
/top_enc_dec_tb/re...      St1
/top_enc_dec_tb/re...      St1
/glbl/GSR                  We1
```

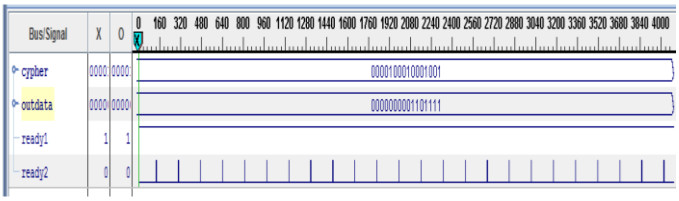
Fig 7. Simulation Result with Input, Output and Recovered Input

Figure 6 and Figure 7displays the complete RSA processor results with encryption and decryption of data.The message(*M*), encryption key(*e*) andmodulas value(*n*) are observed as Indata, InExp and Inmod respectively. From the timing diagrams the time required for conversion to cipher text and convert back to original message can be observed.

The Hardware implementation of RSA processor on Xilinx Spartan III FPGA development board is shown in Figure 8. When the *rst=0* and *fillsel=0*, the ready signals should be '*1*'.

The input 8-bit data that is generated by the LFSR internally for the encryption along with the product result of the Vedic multiplier *i.e.*, modulus value and a message **(16'd0000000010111101)** isprocessed. The output of the encrypted data is obtained as **(0000011111111100).** For decryption of data a decryption key, the encrypted data and modulus is required. The output is found as **(16'd0000000010111101)** hence the original data is obtained and verified.

Fig 8. Hardware Implementation of the RSA Processor



```
Bus/Signal   X      O      0 160 320 480 640 800 960 1120 1280 1440 1600 1760 1920 2080 2240 2400 2560 2720 2880 3040 3200 3360 3520 3680 3840 4000
cypher       0000   0000   0000100010001001
outdata      0000   0000   0000000001101111
ready1       1      1
ready2       0      0
```

Fig 9. Ciphered Output ThroughChipscope

Figure 9 shows the ciphered output, through chipscopefor the input data 16'd0000000001101111, the encrypted output obtained is 16'd0000100010001001. The original data is retraced using the decryption key with output 16'd0000100010001001.

G. *Performance Analysis*

In [12] author reports delay from different implementations that varies from 23.18η seconds to 32.01 η seconds. From Table 1 and Figure 10, it is observed that Nikhilam occupies more area than Urdhva, but the delay is 50% less and power consumption remains the same for both the multipliers.Considering less delay, Nikhilam multiplier can be chosen for implementation, whereas with area the Urdhva performs better. The device utilization on Spartan 3 FPGA, compared with [11] is given in Table 2, shows less device utilization in our implementation.

The multiplier can be selecteddepending on the type of application.For nibble multiplication, 11 logic cells are utilized and propagation delay is 4.585 η seconds in [6]. The Urdva multiplier occupies 9% of area [9] which is 75% more than our implementation.

Table 1. Performance Comparison of Multipliers

Parameters/Technique	**Nikhilam**	**Urdhva – Tiryagbhyam**
Area	315 slices	94 slices
Delay	12.832nanosec	26.391nanosec
Power(Static)	0.060watts	0.060watts

Table 2. Device Utilization Comparison of Multipliers

Bonded IOBs	[11]	%	Our Work	%
Urdhva – Tiryagbhyam	33/108	30	32/141	23
Nikhilam	33/108	30	32/141	23

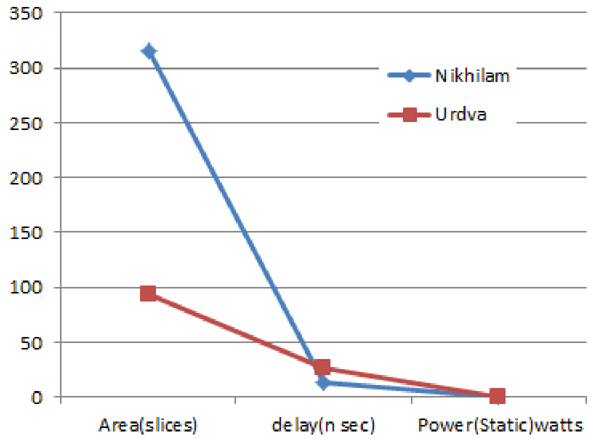


Fig 10. Comparison of Constraints

Observing the results obtained from Table 1-4, it can be concluded that Urdhva gives the best device utilization, with higher frequency and no delay is found. So the Urdhva is selected for the implementation of ECC arithmetic operations, point addition and point doubling.

Table 3. Comparison of Multipliers for Device Utilization

Device	**Parameters**	**Nikhilam**	**Urdhva**
Artix 7	Slice Registers	0%	0%
	LUTs	2%	1%
	Frequency	38.806Mhz	194.76Mhz
	Path delay	No	No
Kintex	Slice Registers	0%	0%
	LUTs	0%	0%
	Frequency	50.698Mhz	228.760Mhz
	Path delay	No	No
Virtex	Slice Registers	28%	23%
	LUTs	9%	10%
	Frequency	15.110Mhz	62.888Mhz
	Path delay	No	No
Spartan 3	Slice Registers	0%	0%
	LUTs	0%	0%
	Frequency	43.679 Mhz	202.360 Mhz
	Path delay	No	No

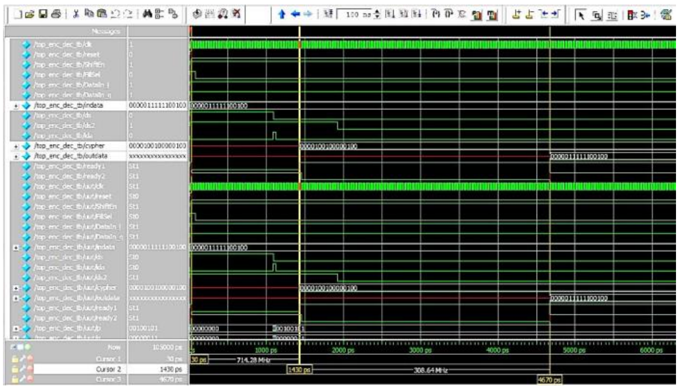


Fig.11 Analysis of Encryption and Decryption Timing

From the Figure 11 the time required for encryption and decryption with 16 bit data is calculated as below.

Encryption Timing = (time position of cursor2) - (time position of cursor1)
= (1430 - 30) ps
= 1430 ps

Decryption Timing = (time position of cursor3) - (time position of cursor2)
= (4670 - 1430) ps
= 3240 ps

From the simulation results obtained with 16 bit data Encryption decryption timings remains same for Nikhilam and Urdhava. The time required is interms of picoseconds and very less compare to the previous results from literature survey.

Table 4. Comparison with RSA encryption and decryption

Device	Parameters	RSA Encryption and Decryption
Artix 7	Slice Registers	6%
	LUTs	30%
	Frequency	38.806
	Path delay	No
	Static Power	0.107Watts
Kintex	Slice Registers	0%
	LUTs	1%
	Frequency	50.698
	Path delay	No
	Static Power	0.148 Watts
Virtex	Slice Registers	0%
	LUTs	0%
	Frequency	50.698
	Path delay	No
	Static Power	0.671 Watts
Spartan 3	Slice Registers	28%
	LUTs	9%
	Frequency	15.110
	Path delay	No
	Static Power	0.060 Watts

V. ECC Finite Arithmetic Model

The ECC finite arithmetic model architecture mainly consists of the main control unit which contains a control signal which is responsible for selecting the Binary or Primary field. Then it is followed by the elliptic curve arithmetic unit which is used to perform the point addition and point doubling with the aid of Vedic multiplier (*Urdva-tiryagbhyam*). The architecture is first implemented with 8 bits of input data with dual field. One field is selected at a time then both point addition and point doubling are performed. Further, the elliptic curves defined over the binary field and primary field GF(2^{163}) and GF(2^{192}) respectively. *Urdva-tiryagbhyam*technique is used for point addition and point doubling operations.The block diagram for point addition and point doubling in binary field and prime field (dual field) using *Urdva-tiryagbhyam*is given in Figure 12.The proposed design has been coded in Verilog HDL, simulated by ModelSim and implemented on Virtex XC5VLX50Tdevice. Figure 13 shows Dual field addition and doubling simulation results for 8bits.

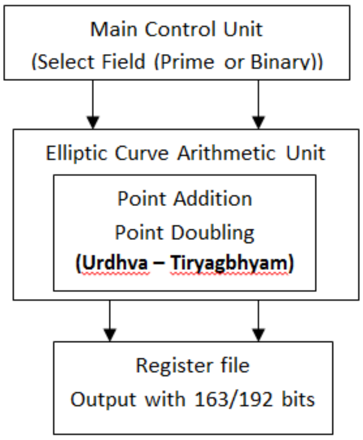


Fig 12. Block Diagram of Elliptic Curve Arithmetic Implementation

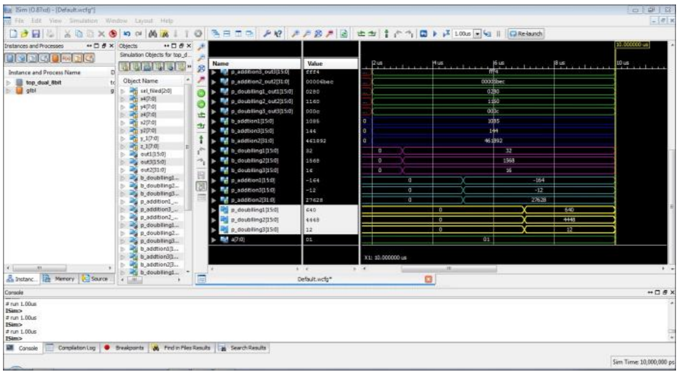


Fig 13. Dual field addition and doubling simulation results(8 bits)

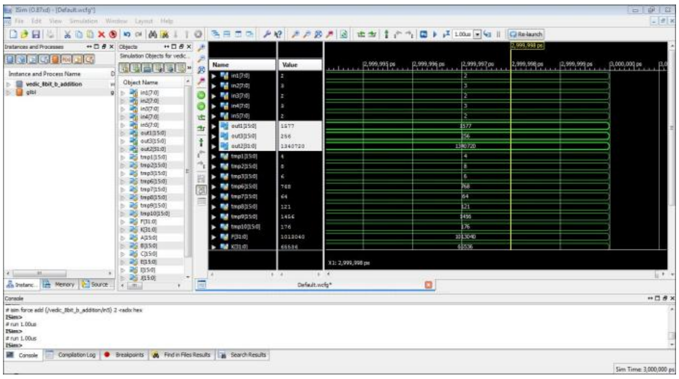
H. Point Addition and Point Doubling Over Binary Field

This point addition requires mainly two points one is in projective coordinate andanother point affine coordinate, the outcome will be in the projective directions.

The algorithm shown in Figure 14, is used to perform the point addition over the binary filed using Lopez Dahab mixed coordinates, since it uses two points one is of affine, second point is of projective coordinates.

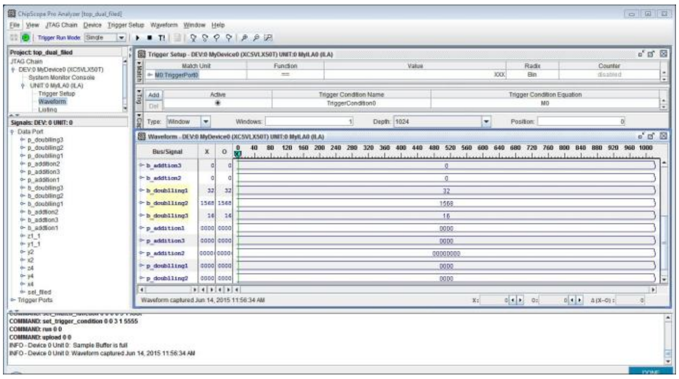
```
Inputs: $A(x_2, y2)$, $Q(X_4,Y_4,Z_4)$.

Outputs: $R(X_3,Y_3,Z_3)$.

$A=Y_4+y_2*Z_4^2$;
$B=X_4+x_2*Z_4$;
$C=B*Z_4$;
$Z_3=C*C$;
$D=x_2*Z_3$;
$E=A+B*B+aC$;
$X_3=A*A+C*E$;
$I=D+X_3$;
$J=A*C+Z_3$;
$F=I*J$;
$K=Z_3*Z_3$;
$Y_3=F+x_2*K+y_2*K$.
```

Fig 14. Algorithm for Point addition in Binary Field



```
# isim force add {/vedic_8bit_b_addition/in5} 2 -radix hex
ISim>
# run 1.00us
ISim>
# run 1.00us
ISim>
```

Fig 15. Binary field addition (8 bits)



```
COMMAND: set_trigger_condition 0 0 3 1 5555
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0: Sample Buffer is full
INFO - Device 0 Unit 0: Waveform captured Jun 14, 2015 11:56:34 AM
```

Fig 16. Binary field addition (8 bits) on Chipscope

Figure 15 shows the simulation output for 8 bits in binary field and the Hardware output with Chipscope inFigure 16.

Inputs (x1,y1,z1)

Yields (x4,y4,z4)

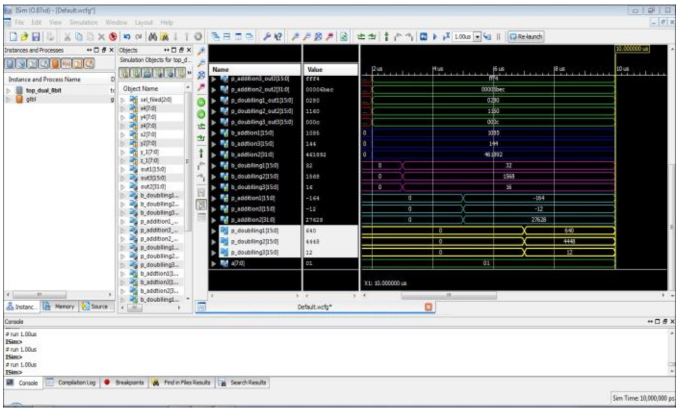
$Z_4=Z_1^2*X_1^2$,

$X_4=X_1^4+bZ_1^4$,

$Y_4=(Y_1^2+aZ_4+bZ_1^4)*X_4+Z_4*bZ_1^4$.

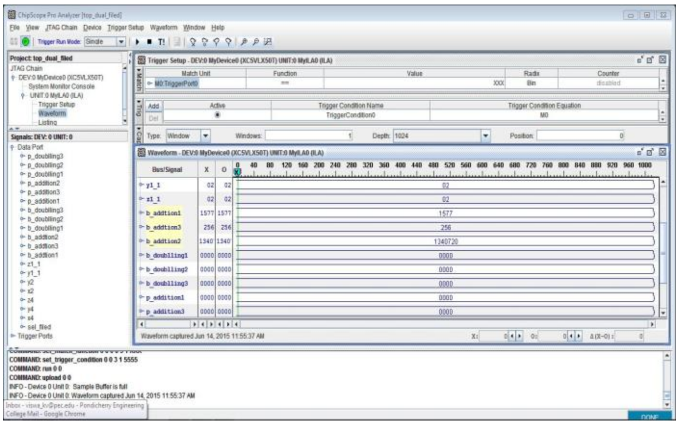
Fig 17. Algorithm for Point doubling in Binary Field

The point doubling algorithm with projective coordinates is shown in Figure 17. The corresponding simulation output and hardware output with chip scope is given in Figure 18 and Figure 19.



```
# run 1.00us
ISim>
# run 1.00us
ISim>
# run 1.00us
ISim>
```

Fig 18. Binary field doubling using mixed coordinates (8 bits)



Trigger Setup - DEV:0 MyDevice0 (XC5VLX50T) UNIT:0 MyILA0 (ILA)

Match Unit	Function	Value	Radix	Counter
M0:TriggerPort0	==	XXX	Bin	disabled

Active	Trigger Condition Name	Trigger Condition Equation
●	TriggerCondition0	M0

Type: Window Windows: 1 Depth: 1024 Position: 0

Waveform - DEV:0 MyDevice0 (XC5VLX50T) UNIT:0 MyILA0 (ILA)

Bus/Signal	X	O	0 … 1000
y1_1	02	02	02
z1_1	02	02	02
b_addtion1	1577	1577	1577
b_addtion3	256	256	256
b_addtion2	1340	1340	1340720
b_doublling1	0000	0000	0000
b_doublling2	0000	0000	0000
b_doublling3	0000	0000	0000
p_addition1	0000	0000	0000
p_addition3	0000	0000	0000

Waveform captured Jun 14, 2015 11:55:37 AM X: 0 O: 0 Δ(X-O): 0

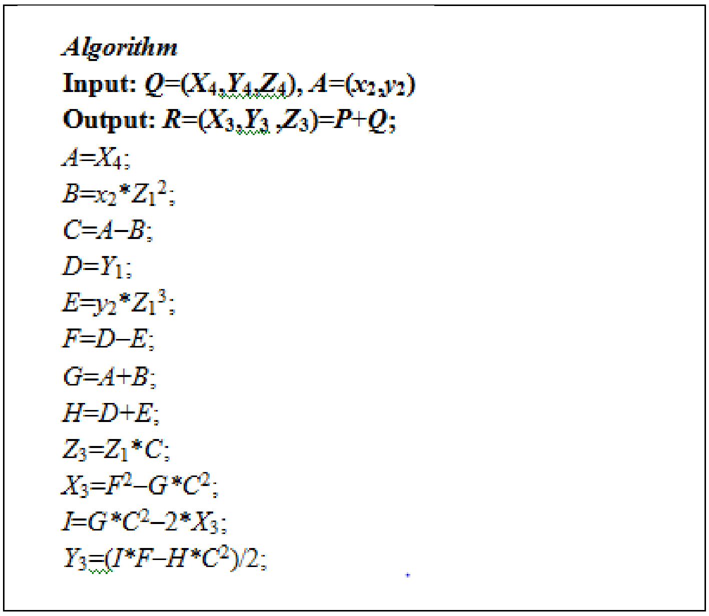
```
COMMAND: set_trigger_condition 0 0 3 1 5555
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0: Sample Buffer is full
INFO - Device 0 Unit 0: Waveform captured Jun 14, 2015 11:55:37 AM
```

Fig 19. Binary field doubling (8 bits) on Chipscope

I. *Point Addition and Point Doubling Over Prime Field*

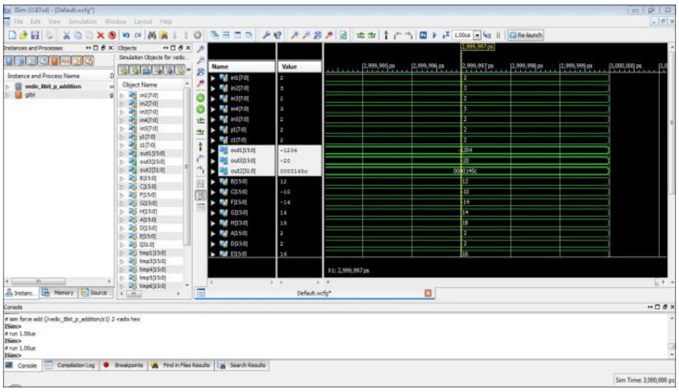
For an elliptic curve, in case of point addition over the prime filed one point is the normal point which is represented as (x,y) is projected into projective coordinates (X1,Y1,Z1),where x=X/Z^2 and y=Y/Z^3 and another point is of affine point represented as (x2,y2). Point addition algorithm and results are given in Figures 20-22.

The point doubling over prime field is carried out by using the pure projective coordinates. It requires only one point which is in projective coordinates and the result point doubling (adding a point witch itself) is also in projective coordinates. In these algorithm *a* and *b* are called as the parameters of an elliptic curves. Here point P is the input and the output of point doubling is represented by 2P. The algorithm to perform point doubling over the prime field is represented as follows in
Figure 23, 24 and Figure 25.



```
Algorithm
Input: $Q=(X_4,Y_4,Z_4)$, $A=(x_2,y_2)$
Output: $R=(X_3,Y_3,Z_3)=P+Q$;
$A=X_4$;
$B=x_2*Z_1^2$;
$C=A-B$;
$D=Y_1$;
$E=y_2*Z_1^3$;
$F=D-E$;
$G=A+B$;
$H=D+E$;
$Z_3=Z_1*C$;
$X_3=F^2-G*C^2$;
$I=G*C^2-2*X_3$;
$Y_3=(I*F-H*C^2)/2$;
```

Fig 20. Algorithm for Point addition in Primary Field

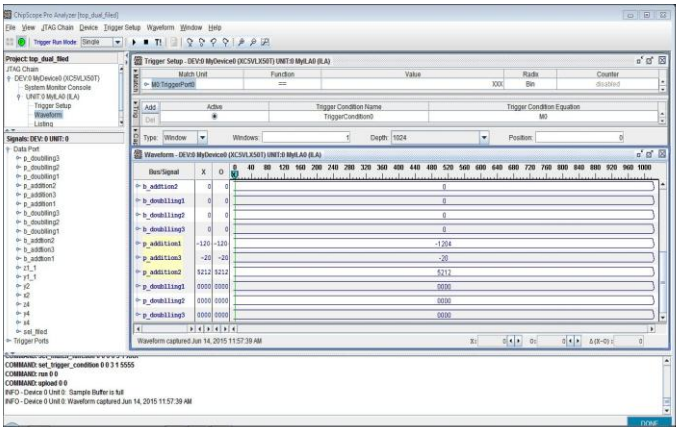


```
Name          Value
in1[7:0]      2
in2[7:0]      3
in3[7:0]      2
in4[7:0]      3
in5[7:0]      2
y1[7:0]       2
z1[7:0]       2
out1[15:0]    -1204
out3[15:0]    -20
out2[31:0]    0000145c
B[15:0]       12
C[15:0]       -10
F[15:0]       -14
G[15:0]       14
H[15:0]       18
A[15:0]       2
D[15:0]       2
E[15:0]       16

X1: 2,999,997 ps
```

```
# isim force add {/vedic_8bit_p_addition/z1} 2 -radix hex
ISim>
# run 1.00us
ISim>
# run 1.00us
ISim>
```

Sim Time: 3,000,000 ps

Fig 21.Prime field addition simulation results



```
Bus/Signal      X     O
b_addtion2      0     0
b_doublling1    0     0
b_doublling2    0     0
b_doublling3    0     0
p_addition1     -120  -120
p_addition3     -20   -20
p_addition2     5212  5212
p_doublling1    0000  0000
p_doublling2    0000  0000
p_doublling3    0000  0000
Waveform captured Jun 14, 2015 11:57:39 AM
```

```
COMMAND: set_trigger_condition 0 0 3 1 5555
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0: Sample Buffer is full
INFO - Device 0 Unit 0: Waveform captured Jun 14, 2015 11:57:39 AM
```

Fig 22. Primary field addition (8 bits) on Chipscope

Input: $\boldsymbol{P=(X_1, Y1, Z1)}$

Output: $\boldsymbol{Q=(X_4, Y4, Z4)=2P}$**;**

$A=3*X_1{}^2+a*Z_1{}^4$;

$B=4*X_1*Y_1{}^2$;

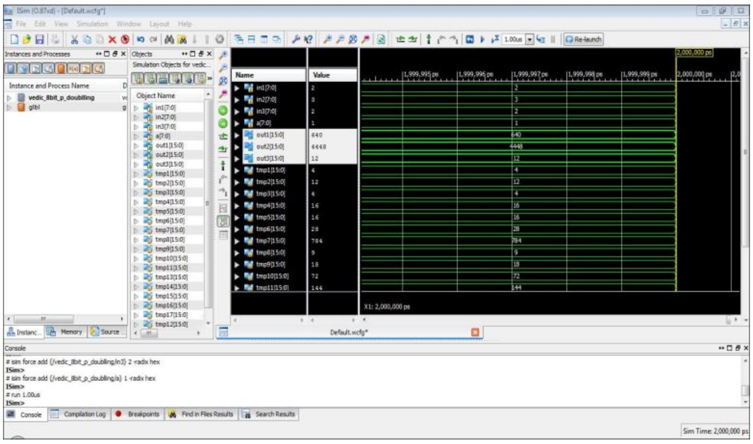
$X_4=A^2-2*B$;

$Z_4=2*Y_1*Z_1$;

$C=8*Y_1{}^4$;

$Y_4=A*(B-X_4)-C$;

Fig 23. Algorithm for Point doubling in Primary Field

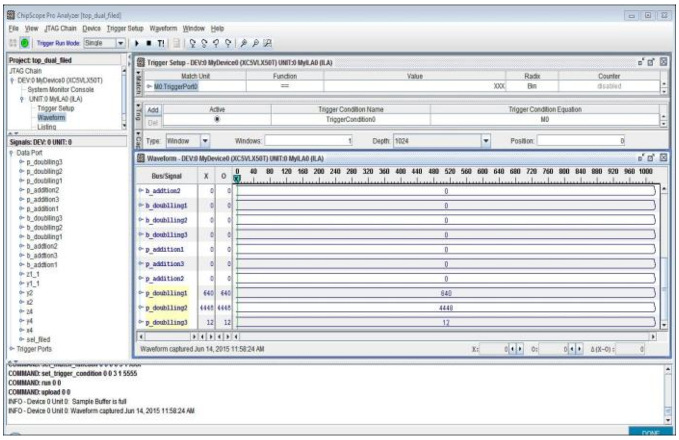


```
Name          Value
in1[7:0]      2
in2[7:0]      3
in3[7:0]      2
a[7:0]        1
out1[15:0]    640
out2[15:0]    4448
out3[15:0]    12
tmp1[15:0]    4
tmp2[15:0]    12
tmp3[15:0]    4
tmp4[15:0]    16
tmp5[15:0]    16
tmp6[15:0]    28
tmp7[15:0]    784
tmp8[15:0]    9
tmp9[15:0]    18
tmp10[15:0]   72
tmp11[15:0]   144

X1: 2,000,000 ps
```

```
# isim force add {/vedic_8bit_p_doubling/in3} 2 -radix hex
ISim>
# isim force add {/vedic_8bit_p_doubling/a} 1 -radix hex
ISim>
# run 1.00us
ISim>
```

Sim Time: 2,000,000 ps

Fig 24. Primary field doubling_simulation


ChipScope Pro Analyzer [top_dual_filed]
Trigger Setup - DEV:0 MyDevice0 (XC5VLX50T) UNIT:0 MyILA0 (ILA)
Waveform - DEV:0 MyDevice0 (XC5VLX50T) UNIT:0 MyILA0 (ILA)
p_doubling1
p_doubling2
p_doubling3
640
4448
12

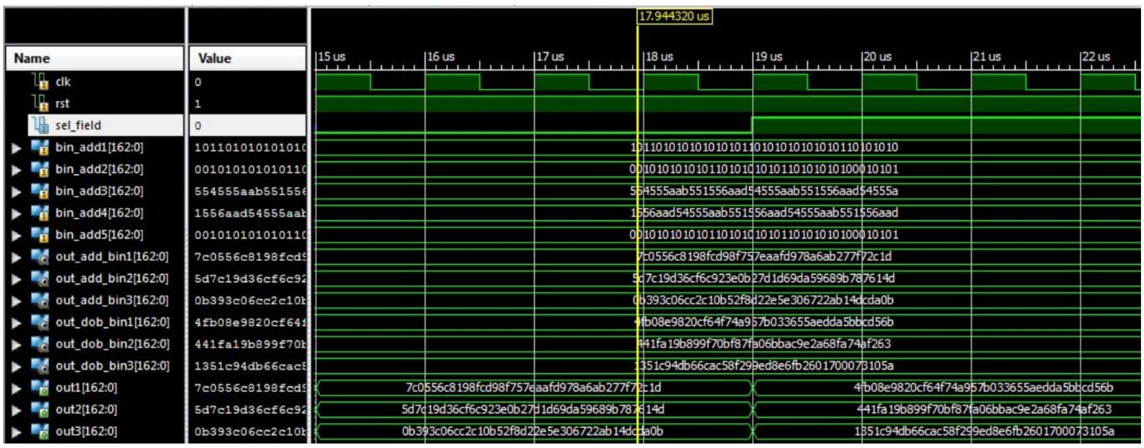
```
COMMAND: set_trigger_condition 0 0 3 1 5555
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0: Sample Buffer is full
INFO - Device 0 Unit 0: Waveform captured Jun 14, 2015 11:58:24 AM
```

Fig 25. Prime field doubling on Chipscope

A. *Performance Analysis*

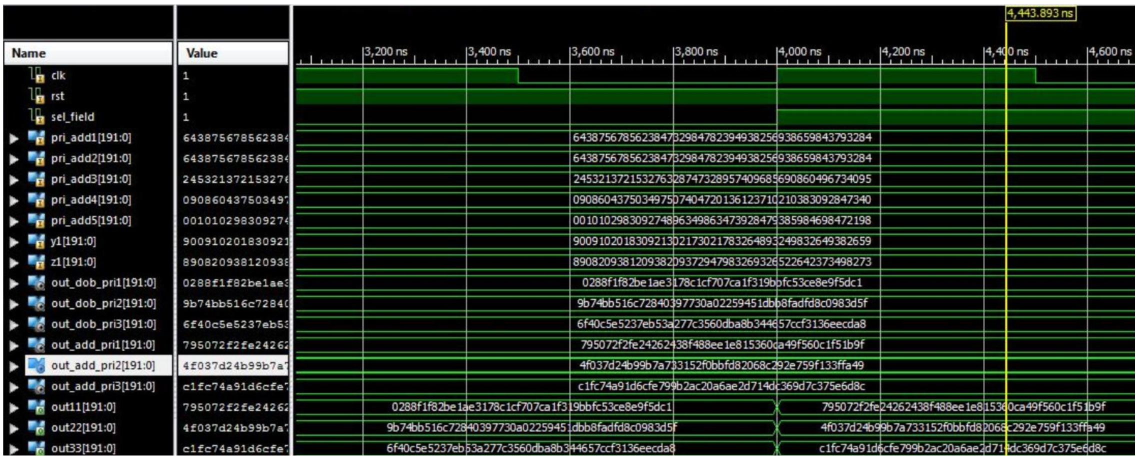
The input message data considered is of 8 bits and the time required for point addition is in terms of pico seconds. The device utilization is also very less i.e 0% for the FPGA used for the implementation. The input size is increased to 163 and 192 bits for both Binary and Primary field. The time required for Point addition and Point doubling is in terms of micro and nano seconds. Simulation waveforms are shown in Figure 26 and Figure 27.

Figure 28 shows the RTL schematic of the implemented dual field processor which performs point addition and point doubling in Binary and Primary fields.



Name	Value
clk	0
rst	1
sel_field	0
bin_add1[162:0]	10110101010101010
bin_add2[162:0]	00101010101011 0
bin_add3[162:0]	554555aab551556
bin_add4[162:0]	1556aad54555aab
bin_add5[162:0]	00101010101011 0
out_add_bin1[162:0]	7c0556c8198fcd9
out_add_bin2[162:0]	5d7c19d36cf6c92
out_add_bin3[162:0]	0b393c06cc2c10b
out_dob_bin1[162:0]	4fb08e9820cf64f
out_dob_bin2[162:0]	441fa19b899f70b
out_dob_bin3[162:0]	1351c94db66cac5
out1[162:0]	7c0556c8198fcd9
out2[162:0]	5d7c19d36cf6c92
out3[162:0]	0b393c06cc2c10b

Fig 26. Simulatiom waveform of Point Addition and Point Doubling for 163 bits



Name	Value
clk	1
rst	1
sel_field	1
pri_add1[191:0]	64387567856238
pri_add2[191:0]	64387567856238
pri_add3[191:0]	24532137215327
pri_add4[191:0]	09086043750349
pri_add5[191:0]	00101029830927
y1[191:0]	900910201830921
z1[191:0]	89082093812093
out_dob_pri1[191:0]	0288f1f82be1ae
out_dob_pri2[191:0]	9b74bb516c7284
out_dob_pri3[191:0]	6f40c5e5237eb5
out_add_pri1[191:0]	795072f2fe2426
out_add_pri2[191:0]	4f037d24b99b7a
out_add_pri3[191:0]	c1fc74a91d6cfe
out11[191:0]	795072f2fe2426
out22[191:0]	4f037d24b99b7a
out33[191:0]	c1fc74a91d6cfe

Fig 27. Simulatiom waveform of Point Addition and Point Doubling for 192 bits

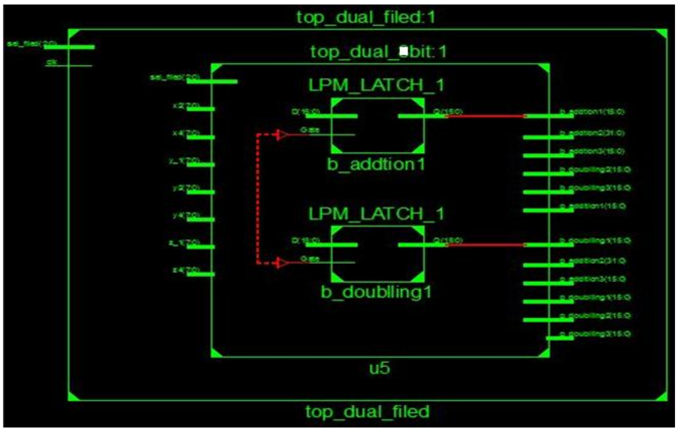
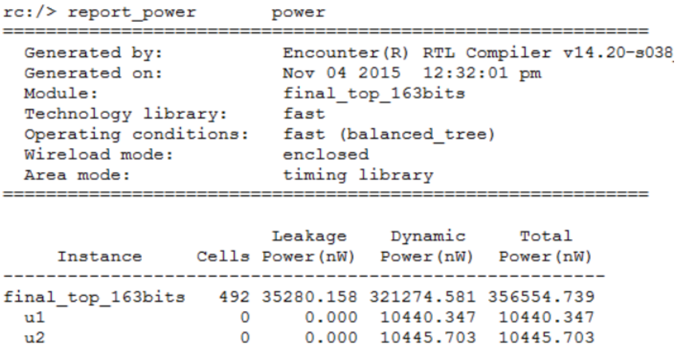
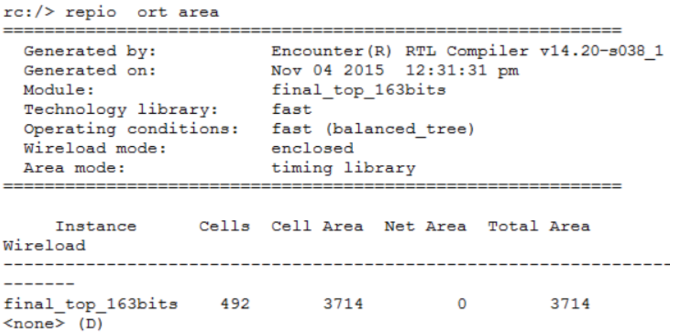


Fig.28 RTL schematic of dual field processor



```
rc:/> report_power       power
============================================================
  Generated by:            Encounter(R) RTL Compiler v14.20-s038
  Generated on:            Nov 04 2015  12:32:01 pm
  Module:                  final_top_163bits
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

                         Leakage    Dynamic     Total
     Instance      Cells Power(nW)  Power(nW)   Power(nW)
--------------------------------------------------------
final_top_163bits    492 35280.158 321274.581 356554.739
  u1                   0     0.000  10440.347  10440.347
  u2                   0     0.000  10445.703  10445.703
```

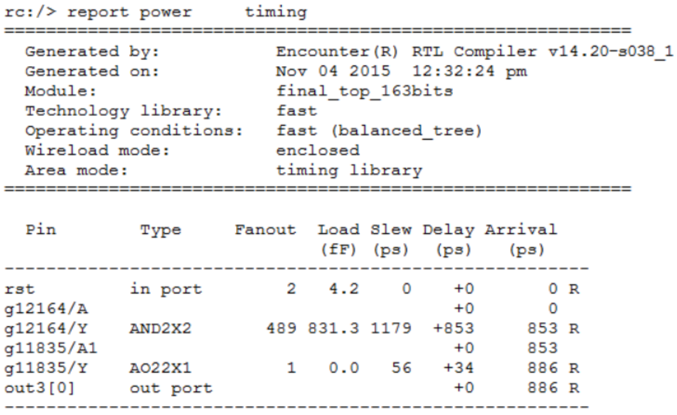
Fig 29. Power consumption of dual processor



```
rc:/> repio  ort area
============================================================
  Generated by:            Encounter(R) RTL Compiler v14.20-s038_1
  Generated on:            Nov 04 2015  12:31:31 pm
  Module:                  final_top_163bits
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

     Instance      Cells  Cell Area  Net Area  Total Area
Wireload
-----------------------------------------------------------------
-------
final_top_163bits    492       3714         0        3714
<none> (D)
```

Fig 30. Area utilized for the dual processor

The Verilog HDL code is executed with Cadence RTL Compiler for area, power and delay details. Figure 29, 30 and 31 gives the area utilized, path delay and power consumption. Figure 32 denotes physical layout of the processor implemented.



```
rc:/> report power     timing
============================================================
  Generated by:            Encounter(R) RTL Compiler v14.20-s038_1
  Generated on:            Nov 04 2015  12:32:24 pm
  Module:                  final_top_163bits
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin         Type      Fanout  Load Slew Delay Arrival
                                (fF)  (ps)  (ps)   (ps)
--------------------------------------------------------
rst          in port       2    4.2    0    +0       0 R
g12164/A                                    +0       0
g12164/Y     AND2X2      489 831.3 1179   +853     853 R
g11835/A1                                   +0     853
g11835/Y     AO22X1        1    0.0   56   +34     886 R
out3[0]      out port                       +0     886 R
--------------------------------------------------------
```

Fig 31. Delay found with dual field processor

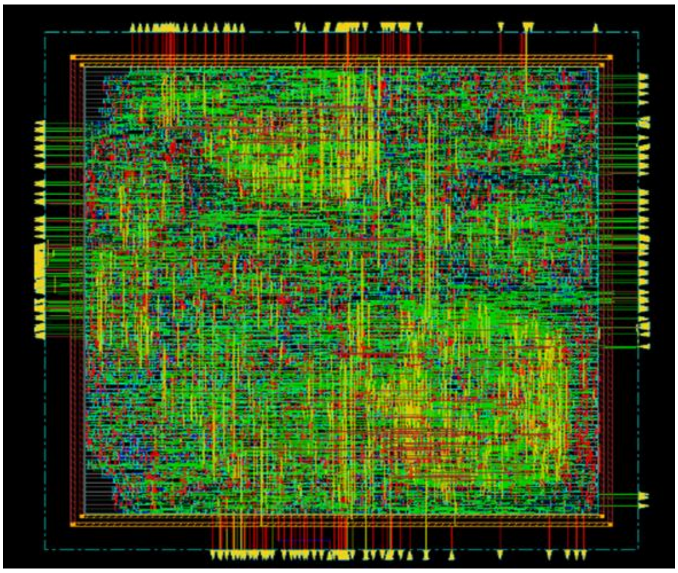


Fig 32.Physical Layout of Dual Filed Processor

VI. Conclusions

RSA processor with *multiplication algorithm*centered on the formula of ancient Indian Vedic mathematics has been projectedandimplemented with both multipliers. Both the Vedic multiplication methods*Urdva-tiryagbhyam* and*Nikhilam*, have been exploredwith respect to area, delay and static power consumption. Due to its construction, Urdhva suffers from high carry propagation delay in instance of multiplication, whereas with Nikhilam Sutra the multiplication is reduced to 50% of it.We have developed and implemented the multiplier architecture based on these Sutrasfor RSA processor. Further, it is planned to implement this work for higher bits to speed up the process of encryption and decryption in Wireless Sensor nodes.

Architectureof dual processor has been proposed with a data path capable of doing operations such as point addition and point doubling of the elliptic curves either on prime filed GF (P) or binary extended fields GF (2^m).The design of dual field processor is presented; this proposed method requires less computational time as compared with the previous implementations. Hence, it results in high performance. This is an initialization work to take up the complete Scalar multiplication in ECC for Encryption and Decryption of information.

References

[1]. Antonio de la Piedra, An Braeken, and AbdellahTouhafi. 2012. *Sensor Systems Based on FPGAs and Their Applications: A Survey.* Sensors (2012), 12, 12235-12264; DOI:10.3390/s 120912235.

[2]. Shaila K, S H Manjula, Thriveni J, Venugopal K R and L MPatnaik,"Resilience Against Node Capture Attack using Asymmetric Matrices in Key Predistribution Scheme in Wireless Sensor Networks ," *in International Journal on Computer Science and Engineering,* ISSN:0975-3397, vol. 11, no. 3, pp. 31-41, 2011.

[3]. Lata B T, VidyaRao, Sivasankari H, Tejaswi V, Shaila K, Venugopal K R, L M Patnaik, "SEAD: Source Encrypted Authentic Data for WirelessSensor Networks," *in International Journal of Engineering Research andDevelopment,* e-ISSN: 2278-067X, p-ISSN: 2278-800X, vol. 11, no. 3,pp. 01-16, 2015.

[4]. Can Eyupoglu, "Investigation of the Performance Nikhilam Multiplication algorithm", In *Proceeding of the world Conference on Technology, Innovation and Entrepreneurship. Elsevier*, Procedia-Social Sciences 195(2015), pp. 1959–1965, 2015.

[5]. Tiwari H.D., Gankhuyag, G., Kim, M. , and Cho, B.: "Multiplier Design Based on Ancient Indian Vedic Mathematics," In Proceedings of the IEEE InternationalConference on SoCDesign, ISOCC, Busan, pp. II-65-II-68,2008.

[6]. Kunchigi, V., Kulkarni, L. and Kulkarni. S.: "High Speed and Area Efficient Vedic Multiplier," In ProceedingsofIEEEInternational Conference on Devices,Circuits and Systems (ICDCS), Coimbatore, pp. 360 – 364, 2012.

[7]. PrabirSaha, Arindam Banerjee, Partha Bhattacharyya, AnupDandapat.:"High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics", in Proceeding of the IEEE Students' TechnologySymposium, lITKharagpur, pp. 237-242, 2011.

[8]. Saha, P., Banerjee, A., Dandapat, A., and Bhattacharyya , P. : "ASICDesign of a High Speed Low Power Circuit for Factorial Calculation using AncientVedic Mathematics," in Elsevier Microelectronics Journal, vol. 42, pp. 1343-1352, 2011.

[9]. Huddar S R., Rupanagudi, S.R., M., Mohan, S.: "Novel High Speed Vedic Mathematics Multiplier using Compressors", in Proceedings of IEEE InternationalMultiConference, pp. 465-469, 2013.

[10]. Kumaravel, Ramalatha, Marimuthu.: "VLSI Implementation of High Performance RSA Algorithm using Vedic Mathematics", in Proceedings of International conference on Computational Intelligence and Multimedia Applications, pp. 126-129, 2007.

[11]. SSChopada, Rama Mehta. "Performance Analysis of Vedic Multiplication Technique Using FPGA", in IEEE Bombay Section Symposium (IBSS), India, 2015.

[12]. Pramod S. Aswale, PriyankaNirgude, Bhakti Patil, RohiniChanderi.: "Design and Implementation of High Speed Multiplier based on Vedic Mathematics", in International Journal of Computer Applications, Vol.155, No.8,2016.

[13]. R Raju., S.Veerakumar .: "Design and Implementation of Low Power and High performance Vedic Multilier", in Proceedings of IEEE International Conference on Communication and Signal Processing, pp. 0601-0605, 2016.

[14]. PritamGajkumar Shah, Xu Huang, Dharmendra Sharma.: “Algorithm based on one’s complement for fast scalar multiplication in ECC for Wireless Sensor Network”, in Proceedings of IEEE International Conference on Advanced Information Networking and Applications Workshops, pp. 571-576, 2010.

[15]. SoramRanbir Singh, Ajoy Kumar Khan, SoramRakesh Singh.: “Performance Evaluation of RSA and Elliptic Curve Cryptography”, in Proceedings of IEEE International Conference on 2nd International Conference on Contemporary Computing and Informatics, pp. 302-306, 2016.

[16]. SoramRanbir Singh, Ajoy Kumar Khan, TalhellambamSonamani Singh.: “ACritical Review on Elliptic Curve Cryptography”, in Proceedings of International Conference on Automatic Control and Dynamic Optimization Techniques, pp. 13-19, 2016.

[17]. Xianjin Fang, Yanting Wu.: “Investigation into the Elliptic Curve Cryptography”, in Proceedings of International Conference on Information Management, pp. 412-415, 2017.

[18]. Shaila K, Nalini L, Tejaswi V, Thriveni J, Venugopal K R, L M Patnaik, ”Secure QoS-Aware Data Fusion to Prevent Node Misbehavior in Wireless Sensor Networks,” *in International Journal of ComputerScience and Network Security,* ISSN: 0975-3397, vol. 11, no. 3, pp. 31- 41, 2011.

[19]. HimanshuThapliyal and M B Srinivas, ”An Efficient Metho d of Elliptic Curve Encryption Using Ancient Indian Vedic Mathematics,” in Procedings of the 48th IEEE Midwest Symposium on Circuit and Systems, ISBN:0-7803-9197-7, vol. 1, pp. 826-828, 2005.

[20]. HilalHoussain, MohamadBadra and Turki F Al-Somani,” Comparative Study of Elliptic Curve Cryptography Hardware Implementations in Wireless Sensor Networks,” in International Journal of RFID Securityand Cryptography (IJRFIDSC), vol. 1, no. 1/2, pp. 67-74, March/June2012.

[21]. Gustavo D Sutter, Jean-Pierre Deschamps and Jos Luis Imaa, ”Efficient Elliptic Curve Point Multiplication Using Digit-Serial Binary Field Operations,” *in IEEE Transactions on Industrial Electronics,* ISSN:0278-0046, vol. 60, no.1, pp. 217-225, 2013.

[22]. J W Lee and H C Chang, ”Efficient Power-Analysis-Resistant Dual-Field Elliptic Curve Cryptographic Processor Using Heterogeneous Dual-Processing-Element Architecture,” *in IEEE Transactions on VeryLarge Scale Integration Systems,* ISSN:1063-8210, vol. 22, no. 1, pp.49-61, 2014.

[23]. Gustavo D Sutter, Deschamps, Jean-Pierre and Imana, Jos´e Luis, “Modular Multiplication and Exponentiation Architectures for Fast RSA Cryptosystem Based on Digit Serial Computation.,” IEEE Transactions on Industrial Electronics, vol. 58, no.7 pp. 3101-3109, 2011.

Leelavathi G "Design of RSA Processor and Field Arithmetic of ECC with Vedic Multipliers for Nodes in Wireless Sensor Networks.” IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) , vol. 8, no. 1, 2018, pp. 01-15.