FPGA IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHY

Lavanya .P [1], Prof. Leelavathi .G[2], Dr. Siva S. Yellampalli[3]

[1,2,3]*Department of VLSI Design and Embedded Systems,*

UTL Technologies, VTU Extension Centre, (India)

ABSTRACT

The rising growth of data communication and electronic transactions over the internet has made security to become the most important issue over the network. The widely used algorithms for public-key cryptosystems are RSA, Diffie-Hellman key agreement, the digital signature algorithm and systems based on elliptic curve cryptography (ECC). ECC offer the smallest key size and highest strength per bit compared to any other public key cryptosystem, since there is currently no known sub-exponential time algorithm to solve the discrete logarithm problem. Smaller key sizes make them highly suitable for hardware implementation on FPGAs.

The main objective of the project is to design and develop the Montgomery Modular Multiplier for ECC cryptosystem system with optimized design for increasing the efficiency of the system by optimizing the area and throughput parameters based on FPGA.

***Keywords:* Cryptography, ECC, Montgomery modular multiplier, FPGA, Atrix.**

I. INTRODUCTION

As the Internet and other forms of electronic communication become more prevalent, electronic security is becoming increasingly important. Cryptography is used to protect e-mail messages, credit card information, and corporate data. One of the most popular cryptography systems used on the Internet is Pretty Good Privacy because it's effective and free.

Cryptography is used for confidentiality, authentication, data integrity, and non-repudiation, which can be divided into two families: secret-key cryptography and public-key cryptography. Secret-key cryptography which usually has a relatively compact architecture and smaller key size than public-key cryptography is often used to encrypt/decrypt sensitive information or documents. Public-key cryptography offers fundamental technology for key agreement, encryption/decryption (two keys), and digital signatures. In contrast to secret-key cryptography, public-key cryptography usually has a lower throughput rate and more complicated computation. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

In recent years the need for secure communication over computer networks has grown significantly, especially with the widespread use of a possibly transparent medium like Internet for online banking and other forms of e-commerce. Such applications use public key cryptosystems like RSA and Elliptic Curve Cryptosystem (ECC) [1][2].

Elliptic Curve Cryptosystems are emerging as a new generation of cryptosystems based on public key cryptography. They offer the smallest key size and highest strength per bit compared to any other public key cryptosystem, since there is currently no known sub-exponential time algorithm to solve the discrete logarithm problem. Smaller key sizes make them highly suitable for hardware implementation on FPGAs.

II. ELLIPTIC CURVE CRYPTOGRAPHY

In 1985, Elliptic Curve Cryptography (ECC) was proposed independently by cryptographers Victor Miller (IBM) and Neal Koblitz (University of Washington). ECC is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). Like the prime factorization problem, ECDLP is another "hard" problem that is deceptively simple to state: Given two points, P and Q, on an elliptic curve, find the integer *n*, if it exists, such that P = nQ.

ECC may be employed with many Internet standards, including CCITT X.509 certificates and certificate revocation lists (CRLs), Internet Key Exchange (IKE), Transport Layer Security (TLS), XML signatures, and applications or protocols based on the cryptographic message syntax (CMS).

RSA had been the mainstay of PKC for over a quarter-century. ECC, however, is emerging as a replacement in some environments because it provides similar levels of security compared to RSA but with significantly reduced key sizes. NIST use the following table to demonstrate the key size relationship between ECC and RSA, and the appropriate choice of AES key size as shown in TABLE 1.

ECC Key Size	**RSA Key Size**	**Key-Size Ratio**	**AES Key Size**
163	1,024	1:6	n/a
256	3,072	1:12	128
384	7,680	1:20	192
512	15,360	1:30	256
Key sizes in bits.		*Source: Certicom, NIST*	

TABLE 1: Comparison of Key sizes

III. ENCRYPTION & DECRYPTION

This scheme will be demonstrated using Alice and Bob as sender and receiver of a secret message, respectively. Typically, the message consists of some large secret number, which is subsequently used by the two parties to open a conventional secure communication channel. The coordinates of the points on the elliptic curve itself serve as a pool of numbers to choose from.

3.1 The Encryption Operation

Step 1: Bob chooses a point α on an elliptic curve E over some Zp and an integer z between 1 and the order of the abelian group E.

Step 2: Bob computes β= z α on the curve and publishes α, β, E, and p. He keeps his private key z secret.

Step 3: Suppose Alice wants to send a message to Bob. Alice picks an integer k between 1 and the order of E, which will be her private key.

Step 4: To encrypt a message, Alice looks up Bob's public key. As the message, she selects a point x on the elliptic curve E. Next, Alice performs the following encryption operation to encrypt the message:

$$e_k(x, k) = (k\alpha, x + k\beta) = (y1, y2).$$

The encrypted message is y = (y1, y2); it includes Alice's public key y1.

3.2 The Decryption Operation

Step 5: Alice sends Bob the encrypted message. To decrypt the message, Bob uses the decryption operation:

$$d_z(y1, y2) = y2 - zy1 = (x + k\beta) - z(k\alpha) = x + k(z\alpha) - z(k\alpha) = x,$$

where z is Bob's private key.

Note the interlocking of public and private keys here: Bob's private key z will decrypt this message correctly, because it matches his public key β = zα, and he can be sure that it was Alice who transmitted this message, since nobody else is in possession of the private key k that matches her public key y1 = kα.

IV. MONTGOMERY'S POINT MULTIPLICATION

Point multiplication is to compute kP, where k is an integer and P is an point on an elliptic curve E defined over a field Fq. Point multiplication is also called scalar multiplication, and it dominates the execution time of elliptic curve cryptographic schemes. There are several algorithms for point multiplication over elliptic curve. The Montgomery point multiplication algorithm used in our work. Block diagram for point multiplier is shown in Fig. 1

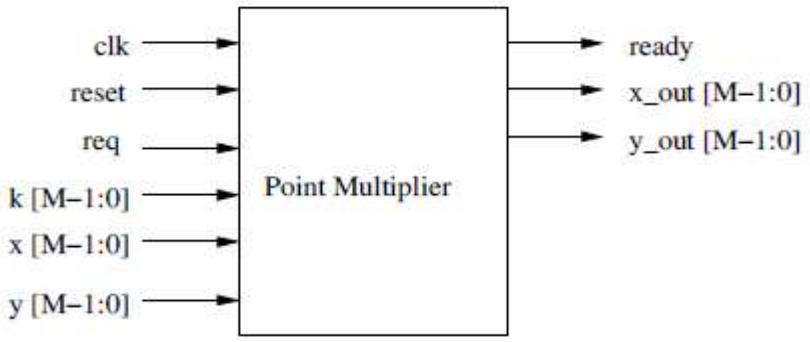


FIGURE 1: Block diagram for point multiplier for GF(2^m)

V. HARDWARE AND SOFTWARE IMPLEMENTATION OF ECC

5.1 Software Requirement Specification

- Operating System: Windows XP with SP2
- Synthesis Tool: Xilinx 12.2.
- Simulation Tool: Modelsim6.3c.

5.2 Hardware Requirement Specification

- Minimum Intel Pentium IV Processor
- Primary memory: 2 GB RAM,
- Artix-7FPGA
- Artix 7 FPGA development board
- Device- XC7A100T-3 CSG324
- USB Supple cable

V. VHDL SIMULATION RESULTS

The functional working of the whole system was verified including simulations and tests of each module. The Results obtained after VHDL implementation of ECC algorithm with Montgomery multiplication at various stages of implementation is summarized below:

To evaluate the results, encryption and decryption of the message is implemented in VHDL. The key based information hiding are analyzed, synthesized and simulated with different input patterns. Fig. 2 shows the snapshot of schematic view of entire code represented in VHDL. It consists of 11 bit data i.e, data (0:10), input of 112 bit i (0:111), reset input with rst of single bit, clock signal given by clk.

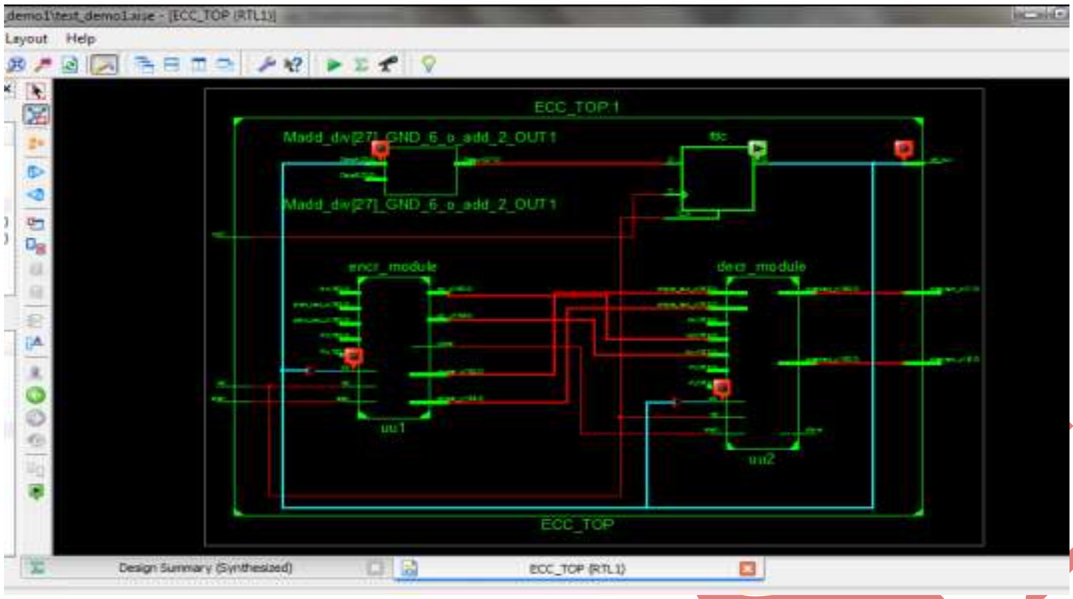
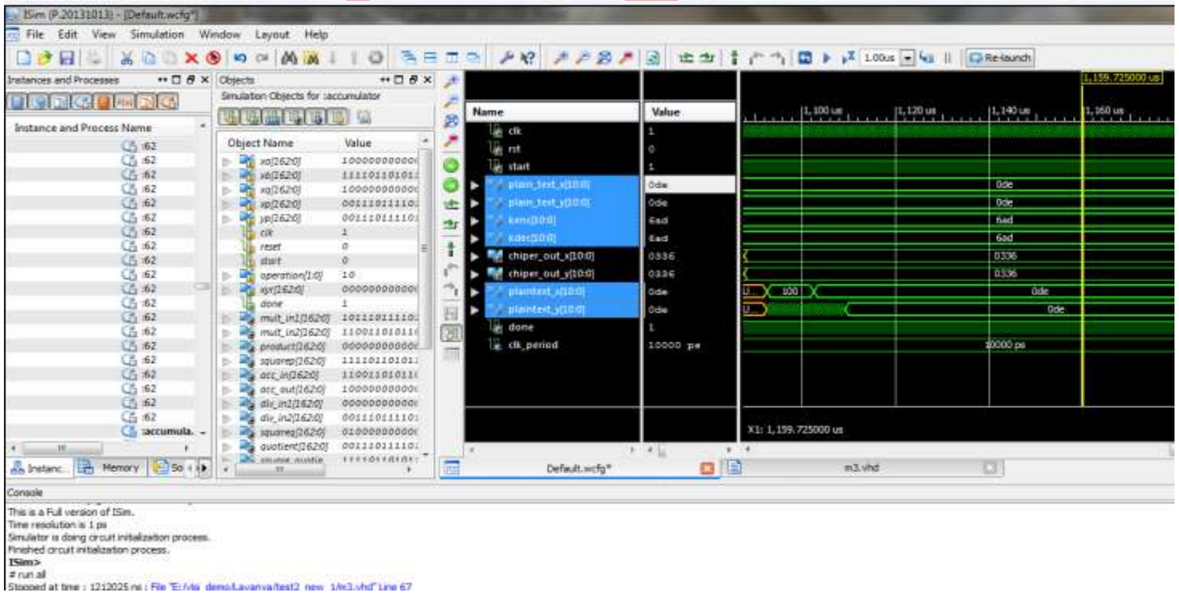

demo1\test_demo1.xise - [ECC_TOP (RTL1)]
Layout
Help
ECC_TOP:1
Madd_div[27]_GND_6_o_add_2_OUT1
Madd_div[27]_GND_6_o_add_2_OUT1
encr_module
decr_module
uu1
uu2
ECC_TOP
Design Summary (Synthesized)
ECC_TOP (RTL1)

FIGURE 2: Snapshot of Schematic of test file

For a 11 bit input data "0DE" and 11 bit key "6AD" , Fig. 3 shows the timing diagram and encryption and decryption results. Fig. 4 shows the simulation clock frequency of 403.177 MHz and time 2.480 ηSec and Fig.5 shows the area utilization of 846 slices and 0% of the device utilization for the same 11 bit input data and key size.



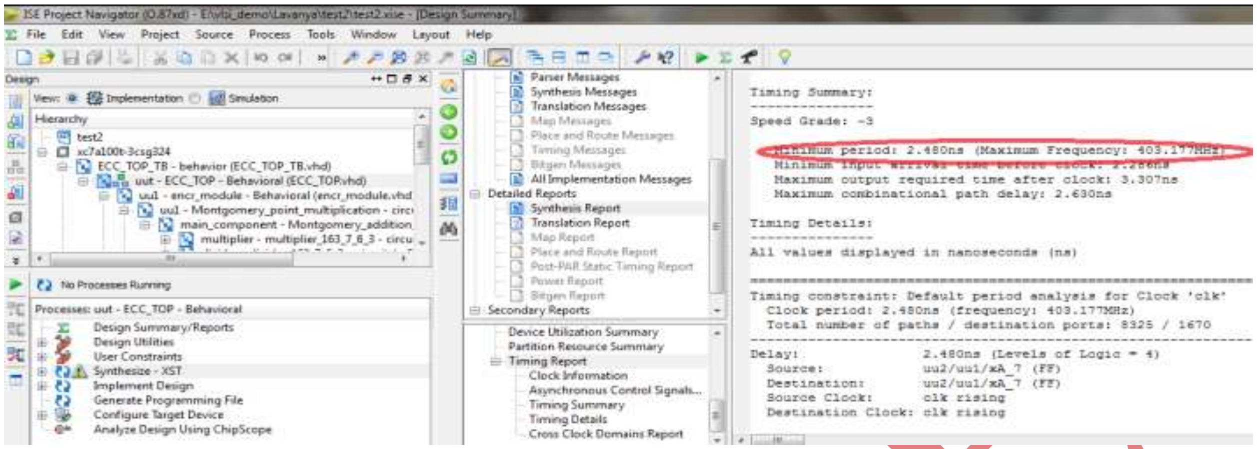
Object Name	Value
xa[162:0]	100000000000
xb[162:0]	111101101011
xq[162:0]	100000000000
xp[162:0]	001110111101
yp[162:0]	001110111101
clk	1
reset	0
start	0
operation[1:0]	10
xyr[162:0]	000000000000
done	1
mult_in1[162:0]	101110111101
mult_in2[162:0]	110011010111
product[162:0]	000000000000
squarep[162:0]	111101101011
acc_in[162:0]	110011010111
acc_out[162:0]	100000000000
div_in1[162:0]	000000000000
div_in2[162:0]	001110111101
squareq[162:0]	010000000000
quotient[162:0]	001110111101

Name	Value
clk	1
rst	0
start	1
plain_text_x[10:0]	0de
plain_text_y[10:0]	0de
kenc[10:0]	6ad
kdec[10:0]	6ad
chiper_out_x[10:0]	0336
chiper_out_y[10:0]	0336
plaintext_x[10:0]	0de
plaintext_y[10:0]	0de
done	1
clk_period	10000 ps

1,159.725000 us
X1: 1,159.725000 us
10000 ps

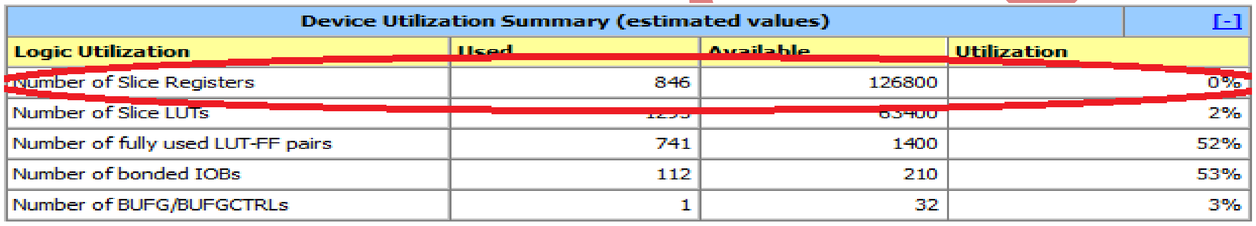
```
Console
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# run all
Stopped at time : 1212025 ns : File "E:/vla_demo/Lavanya/test2_new_1/m3.vhd" Line 67
```

FIGURE 3: Encryption and decryption timing diagram for data and key length M=11 bits



```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 2.480ns (Maximum Frequency: 403.177MHz)
   Minimum input arrival time before clock: 2.286ns
   Maximum output required time after clock: 3.307ns
   Maximum combinational path delay: 2.630ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 2.480ns (frequency: 403.177MHz)
  Total number of paths / destination ports: 8325 / 1670
-------------------------------------------------------------------------
Delay:               2.480ns (Levels of Logic = 4)
  Source:            uu2/uu1/xA_7 (FF)
  Destination:       uu2/uu1/xA_7 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising
```

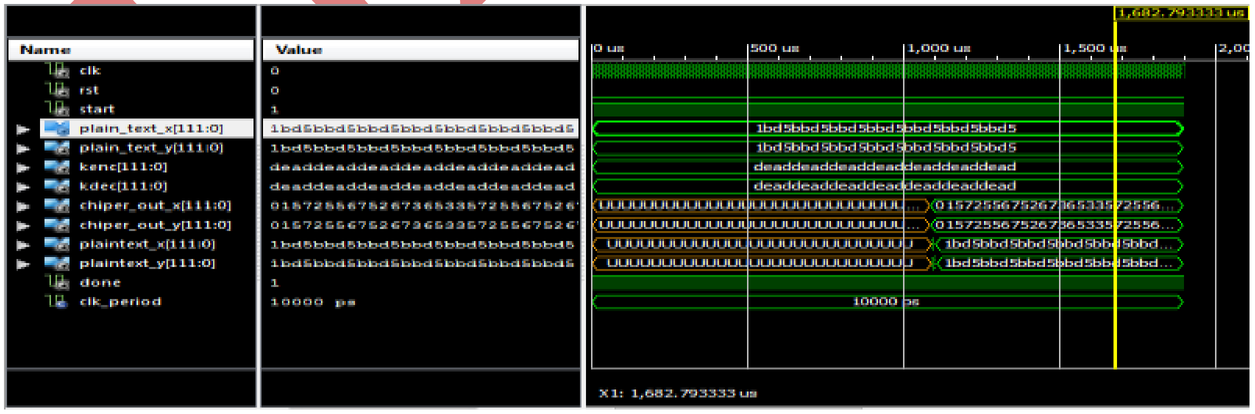
FIGURE 4: Clock frequency and time for 11 bit data and key length



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	846	126800	0%
Number of Slice LUTs	1293	63400	2%
Number of fully used LUT-FF pairs	741	1400	52%
Number of bonded IOBs	112	210	53%
Number of BUFG/BUFGCTRLs	1	32	3%

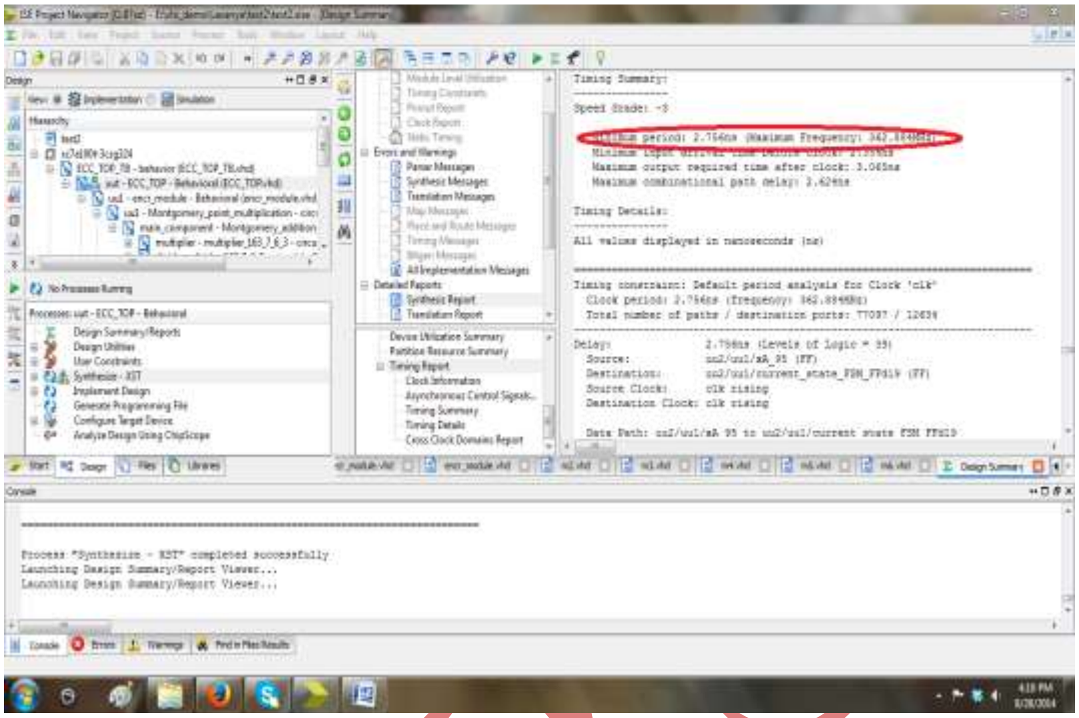
FIGURE 5: Device utilization for 11 bit data and key length

For a 112 bit input data “1bd5bbd5bbd5bbd5bbd5bbd5bbd5” and 112 bit key “deaddeaddeaddeaddeaddeaddead” , Fig. 6 shows the timing diagram and encryption and decryption results. Fig.7 shows the simulation clock frequency of 362.884 MHz and time 2.756 ηSec and Fig.8 shows the area utilization of 5722 slices and 4% of the device utilization for the same 112 bit input data and key size.



Name	Value
clk	0
rst	0
start	1
plain_text_x[111:0]	1bd5bbd5bbd5bbd5bbd5bbd5bbd5
plain_text_y[111:0]	1bd5bbd5bbd5bbd5bbd5bbd5bbd5
kenc[111:0]	deaddeaddeaddeaddeaddeaddead
kdec[111:0]	deaddeaddeaddeaddeaddeaddead
chiper_out_x[111:0]	0157255675267365335725567526'
chiper_out_y[111:0]	0157255675267365335725567526'
plaintext_x[111:0]	1bd5bbd5bbd5bbd5bbd5bbd5bbd5
plaintext_y[111:0]	1bd5bbd5bbd5bbd5bbd5bbd5bbd5
done	1
clk_period	10000 ps

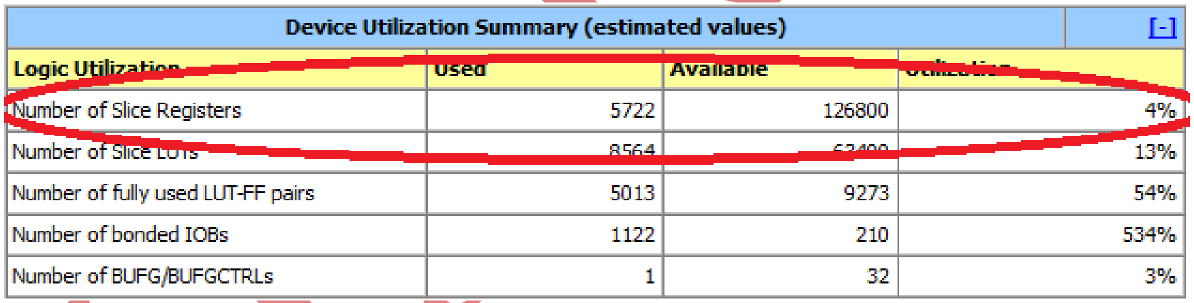
FIGURE 6 :Encryption and decryption timing diagram for data and key length M=112 bits



```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 2.756ns (Maximum Frequency: 362.884MHz)
   Minimum input arrival time before clock: 2.335ns
   Maximum output required time after clock: 3.065ns
   Maximum combinational path delay: 3.424ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 2.756ns (frequency: 362.884MHz)
  Total number of paths / destination ports: 77037 / 12638
-------------------------------------------------------------------------
Delay:               2.756ns (Levels of Logic = 35)
  Source:            uu2/uu1/xA_95 (FF)
  Destination:       uu2/uu1/current_state_FSM_FFd19 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: uu2/uu1/xA_95 to uu2/uu1/current_state_FSM_FFd19
```

```
=========================================================================

Process "Synthesize - XST" completed successfully
Launching Design Summary/Report Viewer...
Launching Design Summary/Report Viewer...
```

FIGURE 7 :Clock frequency and time for 112 bit data and key length



Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	5722	126800	4%
Number of Slice LUTs	8564	63400	13%
Number of fully used LUT-FF pairs	5013	9273	54%
Number of bonded IOBs	1122	210	534%
Number of BUFG/BUFGCTRLs	1	32	3%

FIGURE 8: Device utilization for 112 bit data and key length

VI. SUMMARY OF RESULTS

Device	Key size (Bits)	Clock (MHz)	Time (ƞSec)	Area (slices)	Usage (%)	Encryption time (ms)	Decryption time (ms)
Atrix-7	112	362.884	2.756	5722	4%	3.94	3.96
	11	403.177	2.480	846	0%	4.37	4.40

TABLE 2 : ECC simulation summary of results

- For a key length of 112 bits, the simulation time taken for encryption is 3.94 ms and decryption time is 3.96 ms. While, the clock frequency and time is 362.884 MHz and 2.756 ηs.5722 slices of area are utilized contributing to 6% of the total area on the device.
- For a key length of 11 bits, the simulation time taken for encryption is 4.37 ms and decryption time is 4.4 ms. While, the clock frequency and time is 403.177 MHz and 2.48 ηs.846 slices of area are utilized contributing to 0% of the total area on the device.

VII. CONCLUSION AND FUTURE SCOPE

We implemented all our designs on Xilinx Artix-7 device. We used Model Sim simulator and the Xilinx ISE toolset. Table 6.1 presents our results on the Artix-7 FPGA. No partitioning was required for 112-bit Montgomery point multiplier, comfortably fitting well within the FPGA. Hence we have presented an efficient technique for implementing the Elliptic Curve Cryptographic Scheme on Artix-7 FPGA.

Future enhancement of this work is to accommodate several other modules, as the device utilization is very less. Also the security can also be enhanced by increasing the key size without compromising on the area and time constraints.

REFERENCES

[1] Gerardo Orlando and Christof Paar, A High-Performance Reconfigurable Elliptic Curve Processor for GF(2m), Second International Workshop on Cryptographic Hardware and Embedded Systems -CHES 2000, pp. 41-56, 2000.

[2] M. Morales-Sandoval and C. Feregrino-Uribe, On the hardware design of an elliptic curve cryptosystem, Proceedings of the Fifth Mexican Internaional Conference in Computer Science, pp. 60-70,2004.

[3] M. Ernst, S. Klupsch, O. Hauck, and S.A. Huss,Rapid Prototyping for Hardware Accelerated Elliptic Curve Public-Key Cryptosystems, 12th International Workshop on Rapid System Prototyping, pp. 24-29, 2001.

[4] Leung, K.H., Ma, K.W., Wong, W.K., and Leong P.H.W., FPGA implementation of a micro-coded elliptic curve cryptographic processor, IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 68-76, 2000.

[5] B. Ansari and M. Anwar Hasan, High performance architecture of elliptic curve scalar multiplicatioin,Tech. Report CACR, 2006.

[6] H. Eberle, N. Gura, and S. Chang-Shantz, A cryptographic processor for arbitrary elliptic curves over GF(2m), Application-Speci_c Systems, Architectures and Processors (ASAP) (2003), 98-10.

[7] C. Shu, K. Gaj, and T. El-Ghazawi, Low latency elliptic curve cryptography accelerators for NIST curves on binary fields, IEEE Field-Programmable Technology (FPT), pp. 309-310, 2005.pp. 303-306, 2004.[14] Guerric Meurice de Dormale and Jean-Jacques Quisquater, High-speed hardware implementations of Elliptic Curve Cryptography: A survey, J. Syst.Archit., Vol. 53, pp. 72-84, 2007.