Performance Analysis of Elliptic Curve Cryptography on Reconfigurable Hardware

Prof. Renu Vig ; Ravi Tandon

Abstract

This paper presents an efficient FPGA implementation approach of the elliptic curve cryptography. There are many drawbacks in current encryption algorithms (RSA; AES) in respect of security, power & resources at real-time performance. The Elliptic Curve Cryptography (ECC) is evolving as an important cryptography, and shows a promise to be an alternative of RSA. Small size, high security and other features characterize ECC. Based on the theory of ECC, this paper analyzes its advantages over other cryptographies and focuses on its principle.

Key terms*:* Elliptic curve point addition; point doubling; Finite field arithmetic; Point multiplication; virtex FPGA.

1. Introduction

It is widely recognized that security issues play a crucial role in the majority of computer and communication systems. A central tool for achieving software protection is Cryptography. Cryptographic algorithms are most efficiently implemented in custom hardware than in software running on general purpose processors. Hardware implementations are of extreme importance in case of high performance, security against system intruders and busy systems, where a cryptographic task consumes too much time. Traditional ASIC solutions have the well-known drawback of reduced flexibility compared to software solutions. Since modern security protocols are increasingly becoming algorithm independent, a high degree of flexibility with respect to the cryptographic algorithms is desirable. The security degrees of all the techniques are based on the hardness of mathematical problems. Among them, Elliptic curve cryptography shows a promise to be an alternative of RSA.
A promising solution which combines high flexibility with the speed and physical security of traditional hardware is the implementation of cryptographic algorithms on reconfigurable devices such as FPGAs. FPGAs are hardware devices whose function is not fixed and which can be programmed in-system. An FPGA implementation can be easily upgraded to incorporate any protocol changes without the need for expensive and time consuming physical design, fabrication and testing required in case of ASICs

The authors would like to thank Government of India, Ministry of Communications and Information Technology, Department of Information Technology, New Delhi, for funding the Project "Development of Software Protection Tools", under which this work has been done.
* Prof. Renu Vig is with the University Institute of Engineering and Technology; Panjab University; India;Email:renuvig@hotmail.com.
** Ravi Tandon (Project Associate; master of electronics and communication; Panjab University; India; Email: robintandon_007@yahoo.co.in

This work presents the ECC processor structure of moderate gate count and high speed and it is organized as follows. Section 2 focuses on the architecture of the ECC design.
Section 3 explores the implementation techniques used for ECC Encryption and Decryption. Section 4 is endowed with experimental results on FPGA platform Section 5 shows the comparisons and finally section 6 winds up with future work and conclusion.

2. Architecture of the Circuit

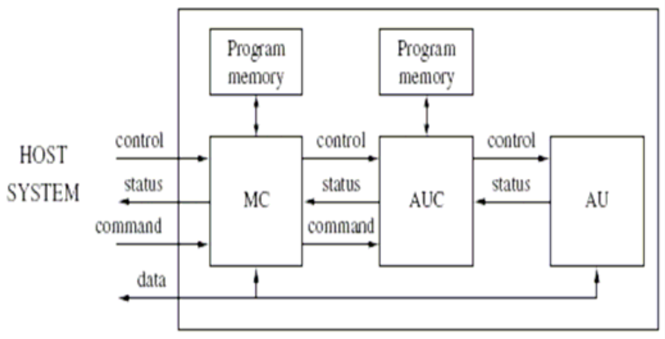

Fig. 1: ECC processor architecture

The operation that dominates the execution time of an elliptic curve cryptographic protocol is point multiplication. Efficient implementation of point multiplication can
be separated into three distinct layers:
1. Finite field arithmetic;
2. Elliptic curve point addition and doubling
3. Point multiplication technique.

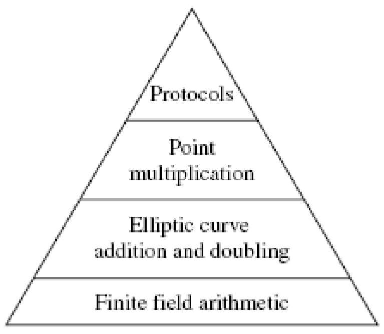

Fig. 2: Hierarchy of operations in ECC

Accordingly, there is a hierarchy of operations involved in point multiplication with point multiplication techniques near the top and the fundamental finite field arithmetic at the base. The hierarchy, depicted in above Fig 2, has been extended to the protocol level. For example, one could decide to implement ECDSA signature generation entirely

in hardware so that the only input to the device is the message to be signed, and the only output is the signature for that message.
An important element of hardware design is to determine those layers of the hierarchy that should be implemented in silicon. Clearly, finite field arithmetic must be designed into any hardware implementation accelerator for finite field arithmetic only and then use an off-the-shelf microprocessor to perform the higher-level functions of elliptic curve point arithmetic. It is important to note that an efficient finite field multiplier does not necessarily yield an efficient point multiplier—all layers of the hierarchy need to be optimized. Moving point addition and doubling and then point multiplication to hardware provides a more efficient ECC processor at the expense of more complexity. In all cases a combination of both efficient algorithms and hardware architectures is required. One approach to higher functionality is the processor depicted in Fig 1. Along with program and data memory, the three main components are an arithmetic logic unit (AU), an arithmetic unit controller (AUC), and a main controller (MC). The AU performs the basic field operations of addition, squaring, multiplication, and inversion, and is controlled by the AUC. The AUC executes the elliptic curve operations of point addition and doubling. The MC coordinates and executes the method chosen for point multiplication, and interacts with the host system.

3. Implementation Approaches:

Elliptic curve key generation:
Let E be an elliptic curve defined over a finite field Fp. Let P be a point in E (Fp), and suppose that P has prime order n. Then the cyclic subgroup of E (Fp) generated by P is P = {∞, P, 2P, 3P, (n−1) P}.The prime p, the equation of the elliptic curve E, and the point P and its order n, are the public domain parameters. A private key is an integer‘d’ that is selected uniformly at random from the interval [1, n −1], and the corresponding public key is Q = d*P.

Algorithm 1:
Elliptic curve key pair generation
Input: Elliptic curve domain parameters (p, E, P, n).
Output: Public key Q and private key d.
1. Select d belongs to R in range [1, n−1].
2. Compute Q = d*P.
3. Return (Q, d).

Elliptic curve encryption and decryption scheme:
We present the encryption and decryption procedures for the elliptic curve. A
Plaintext ‘m’ is first represented as a point M, and then encrypted by adding it to k*Q where k is a randomly selected integer and Q is the intended recipient’s public key. The sender transmits the points C1 = k*P and C2 = M + (k*Q) to the recipient who uses her private key‘d’ to compute dC1 = d (k*P) = k (d*P) = k*Q, and thereafter recovers M = C2 − (k*Q). An eavesdropper who wishes to recover M needs to compute k*Q. This task of computing k*Q from the domain parameters, Q, and C1 = k*P, is the elliptic curve analogue of the Diffie-Hellman problem.

Algorithm 2:

Basic elliptic curve encryption:
Input: Elliptic curve domain parameters (p, E, P, n), public key Q, plaintext m.
Output: Cipher text (C1, C2).
1. Represent the message ‘m’ as a point M in E (Fp).
2. Select ‘k’ belongs to R in range [1, n−1].
3. Compute C1 = k*P.
4. Compute C2 = M + (k*Q)
5. Return (C1, C2).

Algorithm 3:

Basic elliptic curve decryption
Input: Domain parameters (p, E, P, n), private key d, cipher text (C1, C2).
Output: Plaintext m.
1. Compute M = C2−dC1, and extract m from M.
2. Return (m)

(1) *Finite field arithmetic*: - Fields are abstractions of familiar number systems (such as the rational numbers Q, the real numbers R, and the complex numbers C) and their essential properties. They consist of a set F together with two operations, addition (denoted by +) and multiplication (denoted by ·), that satisfy the usual arithmetic properties.
If the set F is finite, then the field is said to be finite. A field F is equipped with two operations, addition and multiplication. Subtraction of field elements is defined in terms of addition: a −b = a + (−b) where −b is the unique element in F such that b+ (−b) = 0 (−b is called the negative of b).Similarly, division of field elements is defined in terms of multiplication: with b = 0, a/b = a ·b−1 where b−1 is the unique element in F such that b ·b−1 = 1.(b−1 is called the inverse of b.)
Arithmetic unit shown in Fig.3 is carrying out these finite field operations as:

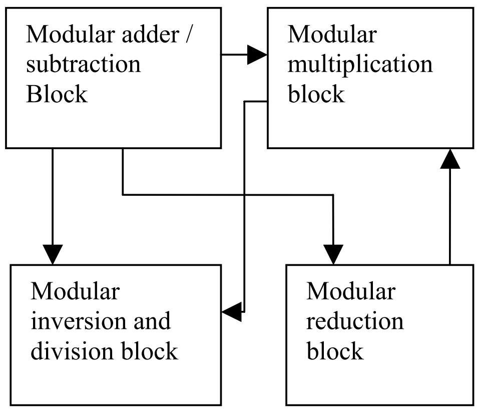
Modular adder / subtraction Block
Modular multiplication block
Modular inversion and division block
Modular reduction block

Fig.3: Arithmetic unit block diagram of ECC

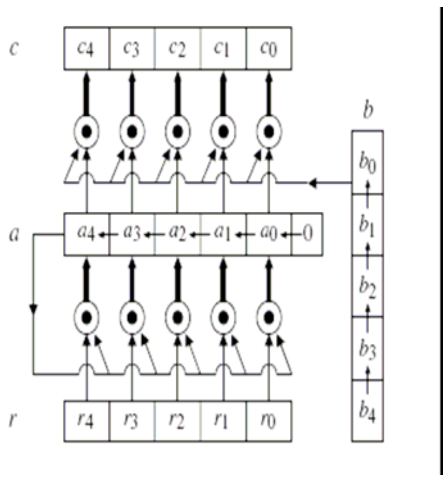

Fig.4: ECC multiplier logic for m=5

ECC multiplier algorithm 4:

```
Input: a =($a_{m-1}$, . . .,$a_1$,$a_0$),b=($b_{m-1}$, . . .,$b_1$,$b_0$) belongs $F_{2m}$(binary field) , and reduction polynomial
f (z) = $z^m$ +r (z).
Output: c = a ·b.
1. Set c←0.
2. For i from 0 to m −1 do
    2.1 c ← c + (b (i)* a).
    2.2 a ← (left shift (a)) +$a_{m-1}$*r.
3. Return(c).
```

(2) *Elliptic curve point addition and doubling:-*

Law for non-super singular (E/F2m): $y^2+x\ y = x^3+ax^2+b$.

Point addition. Let P = (x1, y1) belongs to E (F2m) and Q = (x2, y2) belongs to E (F2m), where P not equal to Q. Then P + Q = (x3, y3), where x3 = λ^2+λ+x1+x2 +a and y3 = λ(x1 +x3) +x3 + y1 with λ = (y1 + y2)/(x1 +x2).

Point doubling. Let P =(x1, y1) belongs to E (F2m), where P = −P. Then 2P =(x3, y3), and x3 = λ2+λ+a = $x1^2 + b/x1^2$ and y3 = $x1^2$ +λx3 +x3 with λ = x1+ y1/x1.

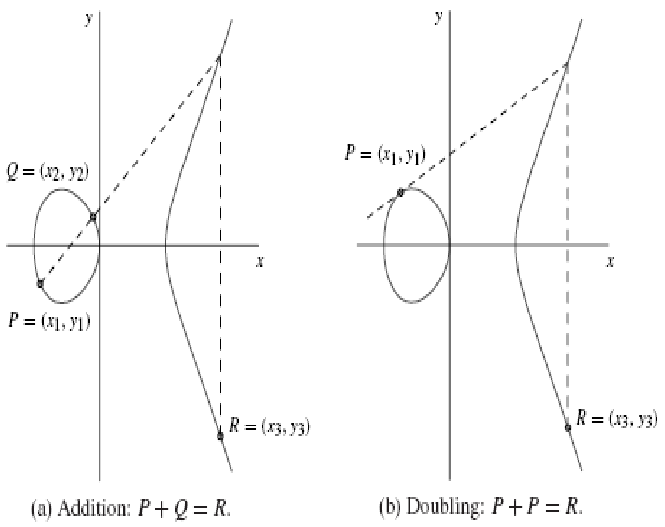

Fig5 :(a) point addition (b) point doubling

(3) *Point Multiplication:*

```
Algorithm 5: Right-to-left binary method for point multiplication
Input: k = ($k_{t-1}$, $k_1$, $k_0$) in binary, P belongs to E (F).
Output: k*P.
1. Q←∞.
2. For i from 0 to t −1 do
    2.1 If k(i) = 1 then Q←Q + P.
    2.2 P←2P.
3. Return (Q).
```

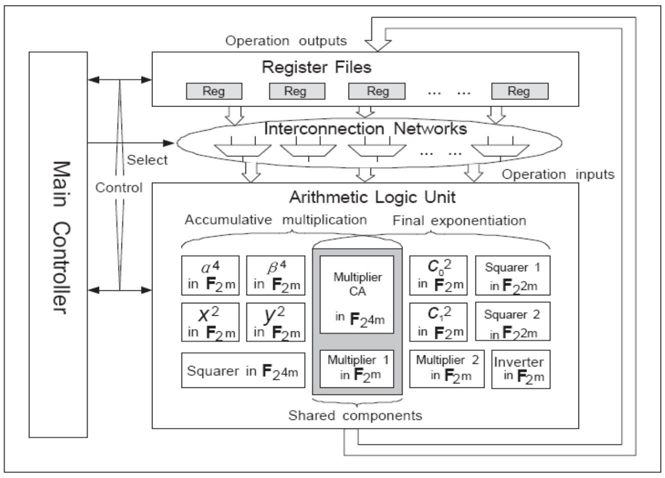

Fig 6: Complete architecture of ECC processor

4. Implementation Results

ISE 8.1i by Xilinx is used to synthesize the VHDL implementation of ECC algorithm and ModelSim 5.4a is used to simulate the design. Xilinx XC4VLX25-12 FPGA is taken as the target device

Table: I

Design	Device	FFs	LUTs	Clock period (ns)
Cast [F2^133]	XC2VP20-7	9988	18045	10.12
Algotronix [F2^133]	XC3S200-5	8970	17457	10.78
Hellicon F2^133]	VIRTEX4-11	4567	9876	7.89
Elbert [F 2^133]	XCV1000-4	5567	12009	8.34
Mcloone [F2^163]	XCV812-8	11234	28095	11.76
This work [F 2^163]	VIRTEX4-12	10938	26901	9.871

Synthesis Results:

Throughput = (163bits * 97.679 MHz)/ 15 = 1.06 Gb/sec
Throughput per Slice: = 1.06 /26901 = 0.039 Mb/sec/slice

5. Comparisons with Previous Implementations

Table I shows the comparison with selected existing FPGA implementations. As can be observed from Table 1 this architecture can achieve a good speed to area ratio, with very less LUTs and FFs utilization, than all prior FPGA implementations known to the authors.

6. Future Work and Conclusion

In this work, a FPGA implementation for ECC is presented. The whole design is captured entirely in VHDL language using a bottom-up design and verification methodology. An optimized coding for the implementation of ECC algorithm has been developed which results in a throughput of 1.06 Gbits/sec using a single FPGA device. Future development will include: Parameterization of the algorithm by selection of cipher key bits (233,283 or 353).

.

References

- *G. Orlando and C. Paar, "A High Performance Reconfigurable Elliptic Curve Processor for GF(2m)", CHES 2000*
- *Trade of analysis of FPGA based elliptic curve cryptographies 2002*
- *K. Fong etal, "Field Inversion and Point Halving Revisited", IEEE Trans on Comp, 2004*
- *N. A. Saqibetal, "A Parallel Architecture for Fast Computation of Elliptic Curve Scalar Multiplication over GF(2m)", Elsevier Journal of Microprocessors and Microsystems, 2004*
- *Sabiel Mercurioetal, " An FPGA Arithmetic Logic Unit for Computing Scalar Multiplication using the Half-and-Add Method", IEEE Reconfigure 2005*
- *A new approach to elliptic curve cryptography: RNS architecture. IEEE MELECON 2006*

- *Blake, I., Seroussi, G., Smart, N.: Elliptic Curves in Cryptography. London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.*
- *Borodin, A., Munro, I.: The Computational Complexity of Algebraic and Numeric Problems. Elsevier, New York, 1999*
- *NIST, \Recommended Elliptic Curves for Federal Government Use", July 1999, see http://csrc.nist.gov/csrc/fedstandards.html.*
- *V. Miller, \Uses of elliptic curves in cryptography",*
- *Lecture Notes in Computer Science 218: Advances in Cryptology - CRYPTO '85, pages 417-426, Springer-Verlag, Berlin, 1986.*
- *N. Koblitz, \Elliptic curve cryptosystems",*
- *Mathematics of Computation, 48:203-209, 1987 ANSI X9.62, \The Elliptic Curve Digital Signature Algorithm (ECDSA)", 1999.*