

An FPGA Implementation of an Elliptic Curve Cryptosystem Coprocessor over Prime Fields

Qian Ding, William H. Robinson

EECS Department, Vanderbilt University, Nashville, TN, U.S.A.

Abstract – *Elliptic curve cryptography (ECC) is an alternative to traditional techniques for public key cryptography. It offers a smaller key size without sacrificing security strength. Previous work found that a hardware implementation of ECC has better performance than the software implementation. Since the performance of a typical elliptic curve cryptosystem is based on the execution time of the encryption/decryption process, hardware can be used to accelerate the computation time of several importance parameters. We propose an efficient field-programmable gate array (FPGA) implementation of the ECC over a prime Galois Field, $GF(p)$. We have designed and synthesized each individual component of the coprocessor for Altera's Cyclone II FPGA. Experimental results demonstrate that the FPGA implementation can improve the performance of the encryption/decryption process by at least 11.6 times compared to software-based implementation.*

Keywords: Elliptic Curve Cryptosystems, Modular Arithmetic, FPGA

1 Introduction

Cryptography is the practice and study of hiding information. There are two categories of cryptographic schemes, i.e., public-key cryptography and symmetric key cryptography. Public-key cryptography is easy for key management, but it is not as efficient as symmetric-key cryptography [1]. Therefore, it is necessary to use dedicated hardware for public-key cryptography to improve the performance. RSA is a well-known public-key cryptography algorithm [2]; its security is based on the difficulty of the integer factorization problem. Elliptic Curve Cryptography (ECC) is an alternative for RSA. The security of ECC is based on the difficulty of the elliptic curve discrete logarithm problem (ECDLP). When compared to RSA, ECC can improve the performance because ECC can provide the same security level with a much smaller key length [3]. Smaller key length results in: (1) faster computation, (2) lower power consumption, and (3) lower memory and bandwidth usage. Despite ECC's advantages over RSA, software-based ECC implementations usually require a long computation time, which makes it difficult to be utilized effectively in a real-time embedded system. Recent work has examined full hardware implementations of ECC as an option to software-based implementations [4, 5]. We propose an efficient FPGA-based implementation of an ECC coprocessor over $GF(p)$. Elliptic curves are typically

defined over two types of finite fields: prime fields of odd characteristic ($GF(p)$, where $p > 3$ is a large prime number), and binary fields of characteristic two ($GF(2^m)$). In $GF(p)$, the elements are integers ($0 \leq x < p$) which are combined using modular arithmetic. We define the elliptic curve over prime field using the following equation:

$$y^2 = x^3 + ax + b \text{ where } a, b \in GF(p) \text{ and } 4a^3 + 27b^2 \neq 0 \quad (1)$$

The simulation results show that our implementation of the encryption/decryption process is at least 11.6 times faster than the software implementation.

2 ECC Coprocessor Implementation

2.1 Basic Design

Figure 1 shows the block diagram of the ECC coprocessor. It consists of six function blocks: an interface block, a parameter generation block, a Prime field arithmetic logic unit (PFALU) block, an encryption/decryption (ENDEC) block, a control block, and a storage block. The interface block controls the communications between the main processor and coprocessor. The parameter generation block computes some important parameters used by the elliptic curve cryptosystem, such as the public key, and does the conversion between the received plaintext message and its corresponding elliptic curve point. The PFALU block consists of all the fundamental operations over Prime field

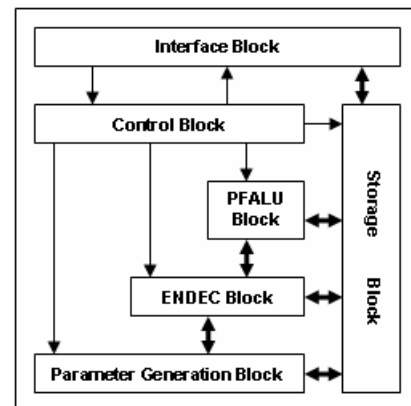


Figure 1. Block Diagram of the ECC Coprocessor (i.e., modular addition/subtraction, modular inversion, and modular multiplication) and the basic elliptic curve operations (i.e., point addition/subtraction and point multiplication). The ENDEC block performs either the encryption process or decryption process of the elliptic

curve cryptosystem. The control block controls the operation mode of the other five function blocks and makes the coprocessor reconfigurable to perform operations over the Prime field and operations required by the elliptic curve cryptosystem. The storage block stores the input, output, and intermediate values.

2.2 Implementation Results

The ECC coprocessor has been implemented on an Altera Cyclone II EP2C70F896C8 FPGA. The device has 68,416 logic elements and 622 pins in total. The design is synthesized with Quartus II v8.1 Web Edition. In order to show the effectiveness of hardware implementation over a software-based approach, we have simulated the coprocessor for prime field operations and elliptic curve point operations individually and compare the timing results with a previous software implementation done by Brown et al. [6]. The timing results for both software and hardware implementation are shown in Table 1. For the point operations, the coefficients of the elliptic curve are set the same as the NIST-recommended prime curve. In addition, we have also implemented NIST-recommended prime curve *P-256* (P stands for Prime field and the number after the hyphen stands for the bitlength of P) which is not presented in Table 1 due to space limitations.

Table 1. Timing results (μs) for operations on NIST-Recommended Prime Curves on both software implementation (SW) and hardware implementation (HW)

	P-192		P-224	
	SW	HW	SW	HW
Mod/Add	0.094	0.052	0.112	0.063
Mod Mult	0.705	0.089	0.913	0.115
Mod Inv	66.3	12.54	88.26	14.26
Point Add/Sub	32.89	0.583	45.23	0.695
Point Mult	681	59.93	1052	81.59

Table 2. The Speedup of the Encryption/Decryption Process for NIST-Recommended Prime Curves on FPGA-based Implementation

	Latency for SW Only (μs)	Full HW Implementation		HW for Point Mult Only	
		Latency (μs)	Speedup	Latency (μs)	Speedup
P-192	2790	241	11.6	306	9.1
P-224	4298	328	13.1	417	10.3

According to a typical El Gamal Cryptosystem, one entire encryption/decryption process (including the operation of the public key computation) requires 4 point multiplications, 1 point addition, and 1 point subtraction. Using the results from the Table 1, we have calculated the speedup of the encryption/decryption process for the NIST-recommended elliptic curves on FPGA-based implementation and the results are shown in Table 2. From

Table 2, we can observe that the speedup of hardware implementation is a factor of 11.6 or greater, and it also increases with the increase in the key size of the NIST-recommended prime curves.

3 Conclusions

We have described an efficient implementation of an elliptic curve coprocessor over $\text{GF}(p)$. The coprocessor can be programmed to execute basic operations for ECC, such as modular addition/subtraction, modular multiplication, modular inversion, elliptic curve point addition/doubling and multiplication. And it can also be used to compute some important parameters used in an elliptic curve cryptosystem and to do the mapping from plaintext message to points on elliptic curve. We compared the FPGA implementation with a software implementation and the experimental results show that the hardware based implementation can improve the latency of encryption/decryption process for NIST-recommended elliptic curves over prime fields by a factor of 11.6 or greater.

4 Acknowledgement

This work was supported in part by DARPA's Computer Science Study Panel – Phase I (HR0011-08-1-0017). In addition, this work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244), BT, Cisco, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies.

5 References

- [1] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*, Springer, 2004.
- [2] R. Rivest, A. Shamir, and L. Adleman, "A Method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, 1978.
- [3] V. Gupta, S. Gupta, and S. Chang and D. Stebila, "Performance analysis of elliptic curve cryptography for SSL," *3rd ACM workshop on wireless security*, pp. 87-94, 2002.
- [4] A. Daly, W. Marnane, T. Kerins, and E. Popovici, "An FPGA Implementation of a $\text{GF}(p)$ ALU for Encryption Processors," *Microprocessors and Microsystems*, vol. 28, no. 5-6, pp. 253-260, 2004.
- [5] S. B. Ors, L. Batina, B. Preneel, J. Vandewalle, "Hardware implementation of an elliptic curve processor over $\text{GF}(p)$," *Proc. of ASAP*, 2003, pp. 433-443.
- [6] M. Brown, D. Hankerson, J. Hernandez and A. Menezes, "Software implementation of the NIST elliptic curves over prime fields," *CT-RSA 2001*, LNCS 2020, pp. 250-265.