Elliptic Curve Cryptoprocessor Implementation on a Nano FPGA: Interesting for Resource-Constrained Devices

Hilal Houssain
LIMOS Laboratory
CNRS France

Turki F. Al-Somani
Computer Engineering Department Umm Al-Qura University
Makkah, Saudi Arabia

Abstract

This paper presents an implementation of an Elliptic Curve Cryptography (ECC) cryptoprocessor on a Nano Field Programmable Gate Array (FPGA), that maybe be interesting for resource-constrained devices that require moderate level of security. Nano FPGAs offer groundbreaking possibilities in power, size, lead-times, operating temperature and cost. To the best of our knowledge, this is the first ECC implementation on Nano FPGAs. The proposed ECC cryptoprocessor was modeled using VHDL and synthesized on Actel IGLOO AGLN250V2-VQFP100 Nano FPGA. The synthesis results show that the targeted Nano FPGA can't exceed the values of m ≤ 11 bits. This is because of the limited number of resources available on Nano FPGAs, which opens a new challenging opportunity for future Nano FPGAs to satisfy the needs of critical portable applications and resource-constrained devices, such as Wireless Sensor Network (WSN).

1. Introduction

Elliptic Curve Cryptography (ECC), which was originally proposed by Niel Koblitz and Victor Miller in 1985 [1, 2] is seen as a serious alternative to RSA [3] with much shorter key size. ECC with key size of 128-256 bits is shown to offer equal security to that of RSA with key size of 1 – 2K bits [4, 5]. To date, no significant breakthroughs have been made in determining weaknesses in the ECC algorithm, which is based on the discrete logarithm problem over points on an elliptic curve. The fact that the problem appears so difficult to crack means that key sizes can be reduced in size considerably, even exponentially [6]. This made ECC become a serious challenge to RSA. The advantage of ECC is being recognized recently where it is being incorporated in many standards. ECCs have gained popularity for cryptographic applications and are considered particularly suitable for implementations on smart cards or mobile devices. Several software implementations of ECC have been reported in recent years [8 – 10]. The advantages of software implementations include ease of use, ease of upgrade, portability, low development cost and flexibility. Their main disadvantages, on the other hand, are their lower performance and limited ability to protect private keys from disclosure compared to hardware implementations. These disadvantages have motivated many researchers to investigate efficient architectures for hardware implementations of ECC. Many hardware implementations of ECC, on the other hand, have been reported [11 – 15]. Hardware implementations offer improved speed and higher security over software implementations, because they cannot be read or modified by an outside attacker as easily as software implementations. Application Specific Integrated Circuits (ASIC) implementations show lower price per unit, high speeds, and low power dissipation. The main disadvantages of ASIC implementations, however, are higher development costs and the lack of flexibility. Field Programmable Gate Array (FPGA) technology offers a good compromise between the speed of ASIC implementations, the short development times, and adaptability of software implementations.

An FPGA contains multiple programmable logic blocks and a network of customizable interconnects for connecting the blocks together. Design teams program the FPGA by connecting logic blocks to perform specific functions. In general, FPGAs give design teams two key benefits: flexibility and fast time-to-market. Despite their utility, however, FPGAs have a couple of drawbacks. Traditionally, FPGAs have relied on static random access memory (SRAM) technology, resulting in a large FPGA footprint and high power draw. These drawbacks alone preclude using FPGA technology in high-volume portable electronic device applications. Without revolutionary advances in FPGA

technology, their utility in portable applications is limited.

Recently, Nano FPGAs have been introduced to offer groundbreaking possibilities in power, size, lead-times, operating temperature, and cost. In addition, Nano FPGAs have been designed for high-volume applications where power and size are key decision criteria [16]. This paper presents an implementation of an Elliptic Curve Cryptography (ECC) cryptoprocessor on a Nano Field Programmable Gate Array (FPGA), that maybe be interesting for resource-constrained devices that require moderate level of security. The rest of the paper is organized as follows. In section 2, we present a brief introduction to the arithmetic of the finite field $GF(2^m)$. Section 3 presents ECC. The proposed architecture of the ECC cryptoprocessor is described in section 4. Discussion of the synthesis results is presented in section 5. Section 6 concludes the presented work.

2. GF(2m) Arithmetic Background

Efficient finite field modular multiplication is important in elliptic curve cryptosystems (ECC) [1]. Finite fields could be either polynomial fields $GF(p)$ or binary fields $GF(2^m)$. The finite $GF(2^m)$ field has particular importance in cryptography since it leads to particularly efficient hardware implementations. Elements of the field are represented in terms of a basis. Most implementations use either a *Polynomial Basis* or a *Normal Basis* [17]. For the implementation described in this paper, normal basis is chosen since it leads to more efficient hardware implementations. Normal basis is more suitable for hardware implementations than polynomial basis since operations are mainly comprised of rotation, shifting and exclusive-OR operations which can be efficiently implemented in hardware. A normal basis of $GF(2^m)$ is a basis of the form

$(\beta^{2^{(m-1)}}, \ldots\ldots, \beta^8, \beta^4, \beta^2, \beta)$, where $\beta \in GF(2^m)$

In a normal basis, an element $A \in GF(2^m)$ can be uniquely represented in the form

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i}, \quad (1)$$

where $a_i \in \{0, 1\}$. $GF(2^m)$ operations using normal basis are performed as follows:

1. **Addition and Subtraction***:* Addition and subtraction are performed by a simple bit-wise exclusive-OR (XOR) operation.
2. **Squaring:** Squaring is simply performed by a rotate left operation.
3. **Multiplication**: $\forall A, B \in GF(2^m)$, where

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$$

And

$$B = \sum_{i=0}^{m-1} b_i \beta^{2^i},$$

The product C = *A*B, is given by:*

$$C = A * B = \sum_{i=0}^{m-1} c_i \beta^{2^i} \quad (2)$$

Then multiplication is defined in terms of a multiplication table $\lambda_{ij} \in \{0, 1\}$

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_{ij} a_{i+k} b_{j+k} \quad (3)$$

An optimal normal basis (ONB) [18] is one with the minimum number of terms in (3), or equivalently, the minimum possible number of nonzero λ_{ij}. This value is $2m - 1$, and since it allows multiplication with minimum complexity, such a basis would normally lead to more efficient hardware implementations.

4. **Inversion**: Inverse of $a \in GF(2^m)$, denoted as a^{-1}, is defined as follows.

$$aa^{-1} = 1 \mod 2^m \quad (4)$$

Most inversion algorithms used are derived from Fermat's Little Theorem:

$$a^{-1} = a^{2^m - 2} = (a^{2^{m-1}-1})^2 \quad (5)$$

for all $a \neq 0$ in $GF(2^m)$.

3. Elliptic Curve Cryptography

Here we present a brief introduction to elliptic curves. Let $GF(2^m)$ be a finite field of characteristic two. A non-supersingular elliptic curve E over $GF(2^m)$ is defined to be the set of solutions $(x, y) \in GF(2^m)$ X $GF(2^m)$ to the equation,

$$y^2 + xy = x^3 + ax^2 + b, \quad (6)$$

where a and $b \in GF(2^m)$, $b \neq 0$, together with the point at infinity denoted by O. It is well known that E forms a commutative finite group, with O as the group identity, under the addition operation known as the tangent and chord method. Explicit rational formulas for the addition rule involve several arithmetic operations (adding, squaring, multiplication and inversion) in the underlying finite field. In affine coordinates, the elliptic group operation is given by the following:

Let $P = (x_1, y_1) \in E$; then $-P = (x_1, x_1 + y_1)$. For all $P \in E$, $O + P = P + O = P$. If $Q = (x_2, y_2) \in E$ *and* $Q \neq -P$, then $P + Q = (x_3, y_3)$,

where

$$x_3 = (\frac{y_1 + y_2}{x_1 + x_2})^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a \quad (7)$$

$$y_3 = (\frac{y_1 + y_2}{x_1 + x_2}) \cdot (x_1 + x_3) + x_3 + y_1 \quad (8)$$

if $P \neq Q$ and,

$$x_3 = x^2{}_1 + \frac{b}{x^2{}_1} \quad (9)$$

$$y_3 = x^2{}_1 + (x_1 + \frac{y_1}{x_1})x_3 + x_3 \quad (10)$$

if $P = Q$.

Computing $P + Q$ is called elliptic curve point addition if $P \neq Q$ and is called elliptic curve point doubling if $P = Q$. Point subtraction is a useful operation in some algorithms. This operation can be performed with the point addition or point doubling formulas using the additive inverse of the point to be subtracted. For example, the point subtraction $P - Q$ can be computed using the point addition operation where: $P - Q = P + (-Q)$. The additive inverse of a point $P = (x, y)$ is the point $(x, x + y)$ for curves defined over the $GF(2^m)$ fields.

Scalar multiplication is the basic operation for ECC. Scalar multiplication in the group of points of an elliptic curve is the analogous of exponentiation in the multiplicative group of integers modulo a fixed integer m. Computing kP can be done with the straightforward *double-and-add* method [5], as described in Algorithm 1, based on the binary expression of $k = (k_{m-1},\ldots,k_0)$ where k_{m-1} is the most significant bit of k. However, several scalar multiplication methods have been proposed in the literature. A good survey is presented in [5].

Algorithm 1: The double-and-add method.
Inputs: P: Base Point, k: Secret key.
Output: kP.
Scalar Multiplication (kP):

1. $Q \leftarrow P$
2. for i from $m-2$ to 0 do
   2.1 $Q \leftarrow 2Q$
   2.2 if $k_i = 1$ then $Q \leftarrow Q + P$
3. output Q

Projective coordinate systems define points over the projective plane as triplets (X, Y, Z). Projective coordinate systems are used to eliminate the need for performing inversion. For elliptic curve defined over $GF(2^m)$, many different forms of formulas are found [19] for point addition and doubling. The projective coordinate system (Pr), so called homogeneous coordinate system, takes the form $(x, y) = (X/Z, Y/Z)$, while the Jacobian coordinate system takes the form $(x, y) = (X/Z^2, Y/Z^3)$ and the Lopez-Dahab coordinate system takes the form $(x, y) = (X/Z, Y/Z^2)$. The Mixed coordinate system, on the other hand, adds two points where one is given in a certain coordinate system while the other is given in another coordinate system. The coordinate system of the resulting point may be in a third coordinate system [19].

In Elliptic Curve Diffie-Hellman Protocol [4], the base point P and the elliptic curve equation are public. User's A private and public keys are k_A and P_A respectively. User's A public key is equal to k_AP. User's B, on the other hand, private and public keys are k_B and P_B respectively. Similarly, User's A public key is equal to k_BP. The message to be encrypted is embedded into the x-coordinate of a point on the elliptic curve ($P_m = (x_m, y_m)$). The shared secret key S between two parties A and B is easily calculated by

$$S = k_A(k_BP) = k_B(k_AP) \quad (11)$$

Whenever one of the users need to send a message to the other party, he/she needs to add the shared secret key to the message to produce the ciphertext point P_c which is

$$P_c = P_m + S \quad (12)$$

To decrypt the ciphertext point, the secret key is subtracted from the ciphertext point to give the plaintext point P_m as follows

$$P_m = P_c + S \quad (13)$$

In elliptic curve ElGamal protocol [4], on the other hand, for some user to encrypt and send the message point P_m to user A, he/she chooses a random integer "l" and generates the ciphertext C_m which consists of the following pair of points:

$$C_m = (lP, P_m + lP_A) \quad (14)$$

The ciphertext pair of points uses A's public key, where only user A can decrypt the plaintext using his/her private key. To decrypt the ciphertext C_m, the first point in the pair of C_m, lP is multiplied by A's private key to get the point $k_a(lP)$. This point is subtracted from the second point of C_m to produce the plaintext point P_m.

The complete decryption operations can be summarized in the following equation

$$P_m = (P_m + lP_A) - k_A(lP) = P_m + l(k_AP) - k_A(lP) \quad (15)$$

4. Architecure of The ECC Cryptoprocessor

This section presents the architecture of the proposed ECC cryptoprocessor. The proposed cryptoprocessor architecture is modeled using VHDL and is fully parameterized. The basic units of this architecture are: (1) the main controller, (2) the data

embedding unit, (3) the point addition and doubling units and (4) the field arithmetic units (adder, multiplier and inverter). In the following subsections, these units are described in detail (see Figure 1).

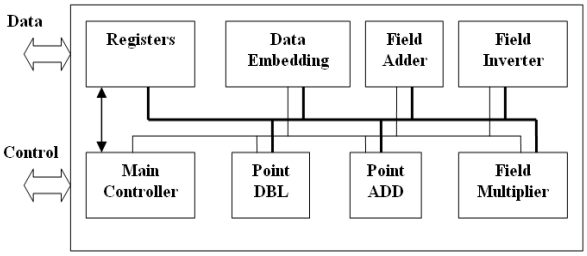

Figure 1: The architecture of the cryptoprocessor.

4.1. Main Controller

The *double-and-add* algorithm was selected for scalar multiplication (Algorithm 1) to reduce the design complexity. For the encryption/decryption process, the selected encryption protocol is the elliptic curve Diffie-Hellman protocol. The pseudo-code of the cryptoprocessor is given in Algorithm 2 below.

The inputs of Algorithm 2 are: (1) the base point P, (2) the elliptic curve parameters a, b, (3) the secret key k, (4) the encryption/decryption mode and (5) the plaintext/ciphertext. The output is either the ciphertext or the plaintext depending on the encryption/decryption mode. Referring to the cryptoprocessor pseudo-code (Algorithm 2, scalar multiplication starts at Step 1 by executing the double and algorithm (Algorithm 1). The encryption process starts at Step 2 by embedding the plaintext into a random point on the elliptic curve. The scalar multiplication result (kP) is added to this point to produce a ciphered point. The decryption process (Step 3), however, subtracts (kP) from the ciphered point.

4.2. Data Embedding

Data embedding is performed within the x-coordinate of a point on the elliptic curve. A random number is picked to fill the 2 most significant bits and the remaining bits will contain the data to be encrypted. If this x-coordinate is not a valid point on the elliptic curve, another random number is picked until a valid elliptic curve point is obtained. Note that we used only to the 2 most significant bits because of the resource limitations of Nano FPGAs that limits the use of large values of m as discussed later in the implementation results section.

4.3. Point Doubling and Addition

Point addition and doubling are performed using Lopez-Dahab projective coordinate system which takes the form $(x, y) = (X/Z, Y/Z^2)$ [19]. Point addition and point doubling require only 14 and 5 field multiplications, respectively.

Algorithm 2: Pseudo code of the cryptoprocessor.

Inputs: P: Base Point, k: Secret key, a, b: Elliptic curve parameters, Plaintext/Ciphertext, Encryption/Decryption.

Outputs: Ciphertext/Plaintext.

Scalar Multiplication (kP):

1. Algorithm 1(P, k).

Encryption/Decryption Process:

2. if (Encrypt) then

2.1. Embed the plaintext in random points on the elliptic curve.

2.2. ADD (kP) to data points.

2.3. Output (ciphertext).

3. else

3.1. ADD ($-kP$) to ciphered points.

3.2. Extract the plaintext from the data points.

3.3 Output (plaintext).

4. end if.

4.4. Field Operations

One key advantage of optimal normal basis representation is the simplicity of the squaring operation. Field squaring is simply a cyclic shift operation. Field addition is a Boolean XOR operation and is implemented using an m-bit XOR unit. Thus, only one clock cycle is required to perform either of the two operations, i.e., field squaring or field addition.

Field multiplication is more complicated than addition and squaring. An efficient multiplier is highly needed and is the key for efficient finite field computations. Massey-Omura multiplier was selected for field arithmetic [20]. The space complexity of the Massey-Omura multiplier is $(2m - 1)$ AND gates + $(2m - 2)$ XOR gates, while the time complexity is $T_A + (1 + log_2(m - 1))\ T_X$, where T_A and T_X are the delay of one AND gate and one XOR gate respectively. One advantage of the Massey-Omura multiplier is that it can be used with both types of the optimal normal basis (Type I and Type II). Another advantage is that it is a bit-serial multiplier and hence the same circuitry used to generate c_0 can be used to generate c_i (i = 1,2, … $m - 1$) as shown in Figure 2 [21].

The encryption/decryption process requires only one inversion since we are using projective coordinate, while an inversion per trial is required for data embedding in a valid x-coordinate. Thus, an

efficient inverter is required. The selected inverter is the Itoh and Tsujii inverter [22]. The Itoh–Tsujii inversion algorithm requires only $O(log_2(m))$ field multiplications, which is the best among other inversion algorithms reported thus far [21].

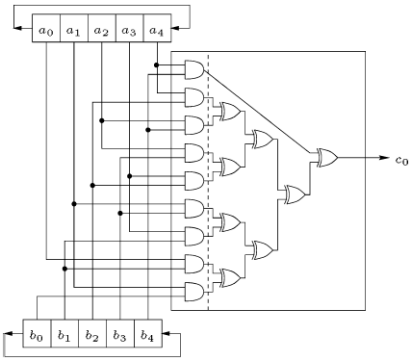

Figure 2: The Massey-Omura bit-serial multiplier.

5. Implementation Results

The proposed ECC cryptoprocessor was modeled using VHDL and synthesized on Actel IGLOO AGLN250V2-VQFP100 Nano FPGA [16]. The developed VHDL models are parameterized to allow synthesizing the cryptoprocessors with different architectural features. The developed VHDL allow for flexible definition of the following parameters: the elliptic curve parameters a and b, the underlying field $GF(2^m)$, the base point P and the secret key k.

The Actel IGLOO AGLN250V2-VQFP100 Nano FPGA core consists of 6144 cells, so called VersaTiles [16]. Each core cell can be configured as a three-input logic function, a D-flip-flop (with or without enable), or a latch by programming the appropriate flash switch interconnections. The versatility of the IGLOO nano core tile as either a three-input lookup table (LUT) equivalent or a D-flip-flop/latch with enable allows for efficient use of the FPGA fabric. Flash switches are distributed throughout the device to provide nonvolatile, reconfigurable interconnect programming. The available logic density on this Nano FPGA is 250,000 gates. It comes in a small footprint packages with the industry's smallest 5x5 mm micro chip scale package. Such Nano FPGA with low-power characteristics, starting at just 2 µW, opens new opportunities for designers of battery-powered handheld applications [16].

The performance of the proposed cryptoprocessor is mainly a function of the control strategy and architectural differences independent of field operations. For example, field multiplication requires m clock cycles because of the bit-serial Massey-Omura multiplier. Point doubling, using Lopez-Dahab coordinates system requires 5 field multiplications, 4 field additions and 6 squaring. Each field addition and squaring requires only one clock cycle as a result of using optimal normal basis. The total number of clock cycles required for performing point doubling is $5m$ + 10 clock cycles. Point addition, on the other hand, requires 14 field multiplications, 8 field additions and 6 squaring, which require $14m$ + 14 clock cycles. Scalar multiplication requires, on the average, m point doubles and $m/2$ point additions, using the double-and-add algorithm. Thus, the required time to perform scalar multiplication, on the average, is $[m\ (5m + 10) + m/2\ (14m + 14)]$, i.e. $(12m^2 + 17m)$ clock cycles.

Table 1 shows the synthesis result of scalar multiplication with m = 5, 9 and 11 bits. Despite that the selected Nano FPGA provides the maximum number of logic cells in its series of Nano FPGAs, we could not exceed the values of m > 11 bits. However, this opens an opportunities for new Nano FPGAs to provide larger number of logic cells needed by resource-constrained applications that require moderate level of security such as Wireless Sensor Networks (WSN) [23, 24].

Table 1: The synthesis results.

m	Clock(MHz)	Time (μsec)	Area (Cells)	Area Usage
5	27.4	14.05109	2681	44%
9	26.7	42.13483	4312	70%
11	26.4	62.08333	5126	83%

WSNs are ad hoc networks comprised of a large number of low-cost, low-power, and multi-functional sensor nodes and one or more base stations. A base station is a much more powerful laptop-class node that connects the sensor nodes to the rest of the world [23, 24] using radio interfaces. There exist a wide range of applications for WSN, such as health monitoring, industrial control, environment observation, as well as office and even military operations. In most of these scenarios, critical information is frequently exchanged among sensor nodes through insecure wireless channels. It is therefore crucial to add security measures to WSNs for protecting its data against threats in a way so integrity, authenticity or confidentiality can be guaranteed.

6. Conclusion

In this paper, an implementation of an Elliptic Curve Cryptography (ECC) cryptoprocessor is presented on a Nano Field Programmable Gate Array (FPGA), that maybe be interesting for resource-constrained devices that require moderate level of security. Nano FPGAs offers lower power, size, lead-times, operating temperature and cost than regular FPGAs. Up to our best knowledge, this is the first

ECC implementation on Nano FPGAs. The proposed ECC cryptoprocessor was modeled using VHDL and synthesized on Actel IGLOO AGLN250V2-VQFP100 Nano FPGA. The synthesis results showed that the targeted Nano FPGA could not exceed the values of $m \leq 11$ bits. This is because of the limited number of resources available on Nano FPGAs. The area result opened a new challenging opportunity for future Nano FPGAs to satisfy the needs of critical portable applications and resource-constrained devices, such as Wireless Sensor Network (WSN).

7. Acknowledgment

The authors would like to acknowledge the support of Umm Al-Qura University, Makkah, Saudi Arabia and the support of LIMOS, CNRS, University Blaise Pascal, Clermont-Ferrand II, France.

8. References

[1] N. Koblitz, Elliptic curve cryptosystems. Math. Comput., Vol. 48, No. 177, pp. 203-209, 1987.

[2] V.S. Miller, "Use of elliptic curves in cryptography", in Proc. of the Advances in Cryptology CRYPTO '85, Springer-Verlag, LNCS 218, pp: 417–426, 1986.

[3] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems". Commun. ACM, Vol. 21, No.2, pp. 120-126, 1978.

[4] A. Menezes, "Elliptic Curve Public Key Cryptosystems", 1st edition, Kluwer Academic Publishers, 1993.

[5] D. Hankerson, A. Menezes, and S. Vanstone, "Guide to Elliptic Curve Cryptography". 1st edition, Springer, 2004.

[6] C. Paar and J. Pelzl, "Understanding Cryptography: A Textbook for Students and Practitioners". 1st edition, Springer-Verlag, Berlin Heidelberg, 2010.

[7] T. ElGamal,. "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Advances in Cryptology", in Proc. of CRYPTO 84, August 19-22, Springer Verlag, pp. 10-18, 1985.

[8] D. Hankerson, J.L.Hernandez and A. Menezes, "Software Implementation Of Elliptic Curve Cryptography Over Binary fields. Cryptographic hardware and Embedded Systems", CHES 2000, LNCS 1965, Springer-Verlag, 1-24, 2000.

[9] N. Gura, A. Patel, A. S. Wander, H. Eberle, and S. Chang Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs", in Cryptographic Hardware and Embedded Systems — CHES 2004, vol. 3156 of Lecture Notes in Computer Science, pp. 119-132. Springer Verlag, 2004.

[10] S. Khajuria and H. Tange, "Implementation of diffie-Hellman key exchange on wireless sensor using elliptic curve cryptography", in Proc. 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology (Wireless VITAE '09), pp. 772–776, May 2009.

[11] J. Lutz and A. Hasan, "High performance FPGA based elliptic curve cryptographic co-processor," *in Proc. Int. Conf. Inf. Technol.: Coding Comput. (ITCC),* 2004, p. 486.

[12] G. Meurice de Dormale and J.-J. Quisquater, High-speed hardware implementations of Elliptic Curve Cryptography: A survey. Journal of Systems Architecture: the EUROMICRO Journal, Volume 53, Issue 2-3, pp. 72 – 84, February, 2007.

[13] K. Jarvinen and J. Skytta, On Parallelization of High-Speed Processors for Elliptic Curve Cryptography, *IEEE Trans. on VLSI,* Vol. 16, Issue 9, pp. 1162 – 1175, 2008.

[14] B. Ansari and M. A. Hasan, "High-Performance Architecture of Elliptic Curve Scalar Multiplication", *IEEE Trans. on Computers,* Vol.57, No.11, pp. 1443-1453, 2008.

[15] Kimmo Järvinen, "Optimized FPGA-based elliptic curve cryptography processor for high-speed applications", *Integration, the VLSI Journal,* Volume 44, Issue 4, pp. 270-279, 2011.

[16] Actel IGLOO Nano FPGA Data Sheet, available from: http://www.actel.com/documents/IGLOO_nano_DS.pdf

[17] R. Lidl, and H. Niederreiter, "Introduction to finite fields and their applications", Cambridge University Press, Cambridge, UK, 2nd edition, 1994.

[18] R. C. MULLIN and R. M. WILSON, "Optimal normal bases in $GF(p^n)$", Discrete Appl. Math., 22, pp. 149–161, 1988/1989.

[19] H. Cohen, G. Frey, R.M. Avanzi, C. Doche, T. Lange, K. Nguyen and F. Vercauteren, "Handbook of Elliptic and Hyperelliptic Curve Cryptography". Discrete Mathematics and Its Applications, Vol. 34, Chapman and Hall, CRC, USA, 2005.

[20] J.L. Massey and J.K. Omura, "Computational Method and Apparatus for Finite Field Arithmetic," US Patent No. 4,587,627, 1986.

[21] T. F. Al-Somani and A. Amin. "Hardware implementations of $GF(2^m)$ arithmetic using normal basis", J. Appl. Sci., Vol. 6, Issue 6, pp. 1362-1372, 2006.

[22] T. Itoh and S. Tsujii. "A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases", Information and Computation, Vol. 78, pp. 171-177, 1988.

[23] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", in Mobile Computing and Networking (MobiCom'99), Seattle, WA USA (1999) 263–270.

[24] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless Sensor Networks: a survey", Computer Networks, Volume 38, Issue 4, 15 March 2002, Pages 393-422.