Low Power Elliptic Curve Digital Signature Design for Constrained Devices

Elhadjyoussef Wajih *Elhadjyoussef_wajih@yahoo.fr*
Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, 5000, Tunisia

Benhadjyoussef Noura *noura-benhadjyoussef@laposte.net*
Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, 5000, Tunisia

Machhout Mohsen *Machhout@yahoo.fr*
Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, 5000, Tunisia

Tourki Rached *Rached.tourki@planet.tn*
Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, 5000, Tunisia

Abstract

Digital signatures represent one of the most widely used security technologies for ensuring unforgeability and non-repudiation of digital data. In this paper a reduced power dissipation of hardware Elliptic Curve Digital Signature design has been developed.

Our proposed architecture is based on the Globally Asynchronous Locally Synchronous (GALS) design methodology. In GALS system, modules that are not used frequently can be made to consume less power by pausing their local clocks until they are needed. Our design consists of using units that are clocked independently. The whole ECDSA design is captured using VHDL language, over the finite field GF (2163), and the Virtex IV FPGA device is used for the hardware implementation of the architecture.

Keywords: Elliptic Curve Digital Signature Algorithm (ECDSA), Locally Synchronous (LS), Low power, Globally Asynchronous Locally Synchronous (GALS).

1. INTRODUCTION

Digital signature is a method of authenticating digital information often treated, as analogous to a physical signature on paper. The purpose of digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing entails transforming the message and some secret information held by the entity into a tag called a signature [1]. Digital signature is divided into a public-key digital signature using public key cryptosystem and the arbitrated digital signature using a conventional cryptosystem [2].

The signature generation and verification of arbitrated digital signature should be done by a third trusted entity because the signature is generated and verified by a symmetric key. But, the signature generated by a public-key digital signature can be verified sby any one since the

information for verification - public key - is open to the public. The Requirements of digital signature are:

- Unforgeable: Only appropriate signer can generate a signature.
- User authentication: Anyone can authenticate the signer.
- Non-repudiation: A signer can't repudiate his/her signature.
- Unalterable: Signed digital document should not be altered.
- Not reusable: A signature can't be reused for other digital document.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is one of the public-key digital signature algorithms [3]. It is the elliptic curve analogue of the digital signature algorithm (DSA) and its security is based on the elliptic curve discrete logarithm problem (ECDLP).
However, when implementing such cryptographic scheme on constrained devices as smart cards, area occupation and time computation must be considered. A main issue in the design of Cryptographic applications is the reduction of power consumption, especially for portable systems.

In this article, an asynchronous design of ECDSA scheme with a low power is, first, presented. Low-cost implementation technique based on Globally Asynchronous Locally Synchronous (GALS) architecture is studied. The basic idea of GALS technique is to partition a large design into isochronous blocks with limited size, each block can be considered as synchronous. Later in this paper, the content will be organized as follows. Section 2 describes characteristics of Elliptic Curve Digital Signature Algorithm and consideration of low power ECDSA module. In Section 3, the building blocks of our low power ECDSA design are detailed. A brief description of the Elliptic Curve Cryptography, SHA hash function and random number generator are presented. And in Section four, the analysis of the implementation performances over binary fields is provided. Finally, conclusions are drawn.

2. LOW PEWER ECDSA ARCHITECTURE

Embedded systems have limited circuit area and computing power by its nature. A special architectural consideration is needed to design ECDSA algorithm. In the following, we review the characteristics of ECDSA algorithm and analyze its structure for hardware implementation at first. And then, we describe the architectural features of our low power ECDSA design.

2.1 Elliptic Curve Digital Signature Algorithm

The Elliptic Curve Digital Signature Algorithm is the Elliptic Curve analogue of the more widely used Digital Signature Algorithm (DSA). It is the application of ECC to digital signature generation and verification [16]. It is now in many standards or recommendations, such as IEEE (Institute of Electrical and Electronics Engineers) standard (IEEE 1363-2000) and FIPS standards (FIPS 186-3).

An elliptic curve E over GF (2^n) with large group of order n and a point P of large order is firstly selected and made public to all users. Then, the following key generation primitive is used by each party to generate the individual public and private key pairs. Furthermore, for each transaction the signature and verification primitives are used.

In the following, we briefly outline the Elliptic Curve Digital Signature Algorithm, as defined in ANSI X9.62, details of which can be found in [17].

- *ECDSA Key Generation:* The user A follows these steps:
  1. Select an elliptic curve E and a point G $\in$ E of order n.
  2. Select a random integer d $\in$ [1, n-1].
  3. Compute Q = d.G on the Montgomery-form. (Compute using the scalar multiplication algorithm and the y-coordinate recovery method).
  4. The public and private keys of the user A are (E, G, n, Q) and d, respectively.
- *ECDSA Signature Generation:* The user A signs the message *m* using the following steps:

1. Select a random integer k $\in$ [1, n-1].
2. Compute kG = (x_1, y_1) and r = $x_1$mod n.
   If r = 0 then go to step 1.
3. Compute k^{-1} mod n.
4. Compute s = k^{-1} (e + d.r) mod n.
   If s = 0 then go to step 1.
   Here e = H (m) is the secure hash algorithm (SHA-1).
   If s = 0 then go to step 1.
5. The signature for the message m is the pair (r, s).

- *ECDSA Signature Verification:* The user B verifies A's signature (r, s) on the message *m* by applying the following steps:
  1. Compute w = s^{-1}mod n and e = H (m).
  2. Compute u_1 = e.w mod n and u_2 = r.w mod n.
  3. Compute X = u_1G + u_2Q = (x_1, y_1) (Compute using the addition formulae in projective coordinates with Y-coordinate) and v = x_1 mod n.
  4. Accept the signature if v=r.

Thus $u_1G + u_2Q = (u_1 + u_2d).G = k.G$, and so $v = r$ as required.

- *Proof that Signature Verification Works:*
  If a signature (*r, s*) on a message *m* was indeed generated by *A*, then

$S \equiv k^{-1} (e + d.r) \bmod n$. Rearranging gives

$$K \equiv s^{-1} (e + d.r) \equiv s^{-1}e + s^{-1}r.d$$
$$\equiv w.e + w.r.d \equiv u_1 + u_2d \pmod{n} \quad (2)$$

The ECDSA module design is presented in Figure 1. The proposed hardware description is implemented over GF (2^n).

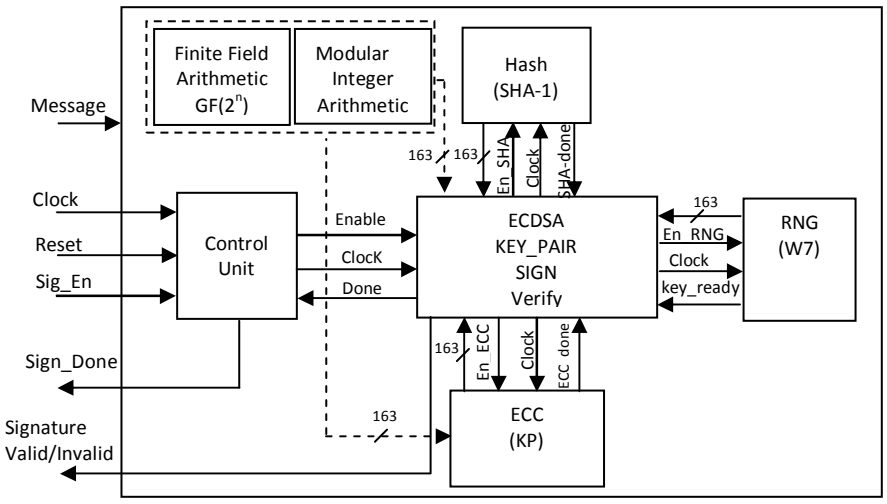

FIGURE 1: Proposed ECDSA module design.

It consists of the following blocks: 163-bit ECC algorithm, SHA-1 function and a Random Number generator (W7).
The scalar multiplication (KP) is based on the Montgomery point multiplication method previously described. Finite field arithmetic over GF (2^n), as well as modular integer arithmetic functions libraries are implemented. A control Unit generates all the appropriate control activation signals. The main concern of design for our low power ECDSA architecture is how to implement these blocks.

2.2 ECDSA Architecture for Low Power Computing

In this section, we discuss the structural access to implement low power ECDSA design. From view point of low power consumption, we designed each component used in ECDSA design and optimized each modules. Then optimized components are assembled for bigger functional unit.

Elliptic Curve Processor

The use of elliptic curves in cryptography was first proposed by Koblitz [8] and Miller [9] in 1985. Since then, Elliptic Curves (EC) over large finite fields has become a common way to implement public-key protocols. It is based on ECDLP, a mathematical problem with no sub-exponential solution as yet. Generally the Elliptic curve is defined either prime field GF (p) or binary field GF (2n). Since the arithmetic in the second field is much faster, we chose to work in GF (2n).

Elliptic curve cryptosystems can use much smaller key sizes, typically around 163 bits, providing the same security level as 1024 key bits of RSA. In addition, ECCs show better performance and computation speed than other multiplication groups such as RSA and ElGamal at the same security level [10]. This makes ECCs very attractive for implementations on devices with limited memory and computation capabilities, like smart cards. An elliptic curve E is defined by the simplified projective coordinates as follow:

$$Y^2Z+XYZ=X^3+aX^2Z+bZ^3 \quad (1)$$

The primary operation in ECC is the scalar point multiplication Q = k.P, where k is an integer and P is a point on the elliptic curve. Point multiplication can be separated into three distinct layers: Finite field arithmetic (Multiplication, Inversion, and Reduction), Elliptic curve point addition (EC_add) and doubling (EC_dbl) operations and Point multiplication technique (KP) [11]. In this document, Montgomery method [26] is applied to implement scalar multiplication as illustrated bellow in Figure 2.

Input: $k=(k_{n-1}, k_{n-2}....., k_1, k_0)2$ with $k_{n-1} = 1$, $P(X_1, Z_1) \in E (GF (2^n))$
Output: Q = kP
1. $P_1 \leftarrow P$; $P_2 \leftarrow 2P$
2. For i from n – 2 downto 0 do
3. if (ki = 1) then
4. $P_1 \leftarrow P_1 + P_2$; $P_2 \leftarrow 2.P_2$
5. else
6. $P_2 \leftarrow P_2 + P_1$; $P_1 \leftarrow 2.P_1$
7. end if
8. end for
9. Return ($Q=P_1$)

FIGURE 2: Montgomery point multiplication Algorithm.

Finite field arithmetic must be designed into any hardware implementation. 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 Figure 3.

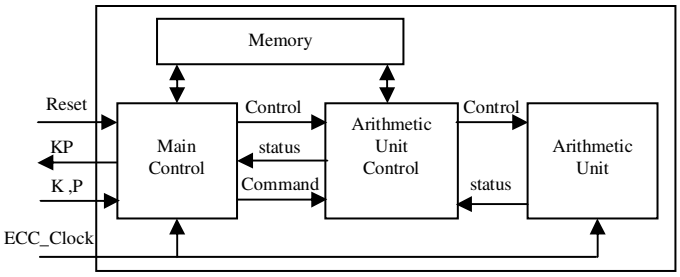

FIGURE 3: Elliptic curve processor architecture.

The three main components are:

- *Arithmetic logic unit (AU):* The AU performs the basic field operations of addition, squaring, multiplication, and inversion, and is controlled by the AUC.
- *Arithmetic unit controller (AUC):* The AUC executes the elliptic curve operations of point addition and doubling.
- *Main controller (MC):* The MC coordinates and executes the method chosen for point multiplication, and interacts with other modules.

- **Hash Function SHA-1**

A function that compresses an arbitrarily large message into a fixed small size is known as a hash function. Any change to the message invariably produces a different hash result when the same hash function is used. Since it was digitally signed, no modification could be made on the message, therefore assurance is guaranteed. In fact, the secure hash algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as Federal Information Processing Standard (FIPS 180): Secure Hash Standard (SHS) [12]. And here as shown in figure 4, the SHA-1 is considered.

A function that compresses an arbitrarily large message into a fixed small size is known as a hash function. Any change to the message invariably produces a different hash result when the same hash function is used. Since it was digitally signed, no modification could be made on the message, therefore assurance is guaranteed. In fact, the secure hash algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as Federal Information Processing Standard (FIPS 180): Secure Hash Standard (SHS) [12]. And here as shown in figure 4, the SHA-1 is considered.

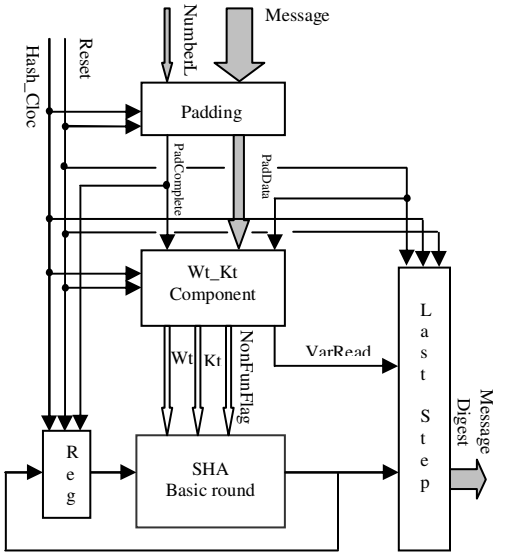

FIGURE 4: SHA_1 Hash function Architecture.

The proposed architecture consists of the following steps:

- *Padding: The message, shall be padded before hash computation begins. The purpose of this padding is to ensure that the padded message is a multiple of 512 bits, depending on the algorithm.*
- *WT_KT component: Operations applied on constants.*
- *REGISTER (REG): Intermediate memory.*
- *SHA BASIC ROUND: The SHA-1 hash computation uses functions previously defined as termed by FIPS.*
- *LAST step: The final result of SHA-1 is a 160-bit message digest, saved in this register.*

Random Number Generator

The security of an entire cryptographic system may depend on the quality and unpredictability of the random numbers used. So, in digital signature protocol, an adversary, who does not know the private key of the signatory, cannot feasibly generate the correct signature [13, 14].

In this paper, the random number generator W7 is used. It consists of a control register and a majority function unit. The majority function is responsible for the key stream generation. This unit contains eight similar cells. Each model consists of three LFSR's 38, 43 and 47 bit long and one majority function unit. Their initial state which is the same for all is the symmetric encryption key.
The majority function unit is responsible of the key stream generation. The three LFSRs together determine when each shift register is clocked. One bit in each register is designated as the clock tap for that register. At each clock cycle the majority value for these taps determines which LFSRs advance. The proposed architecture for the hardware implementation of one cell is presented in Figure 5.

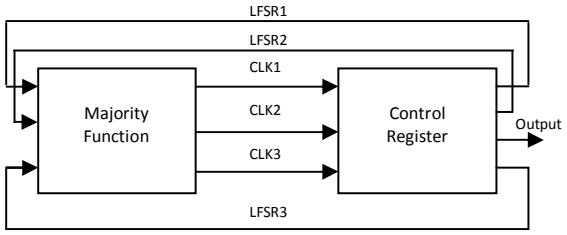

FIGURE 5: W7 key stream generator architecture.

Each cell has two inputs and one output. The first input is the key and it is the same for all the cells. The second input consists of control signals. Finally, the output is of 1-bit long. The outputs of each cell form the key stream byte [15].

3. LOW POWER ECDSA MODULE DESIGN

An architectural design with some low power technique is needed to design our low power ECDSA module. In this chapter, we first present the selected technique to be applied for our design. Then, the design of our low power ECDSA module is detailed.

3.1 GALS System Technique

Almost every digital system is built based on synchronous design techniques. All activities are ordered by a reference clock. The timing of all operations of the circuit design is derived from the global clock signal [4]. In a synchronous system the slowest communicating operations would determine the overall clock rate. Furthermore, a synchronous circuit continues to operate even if it has nothing to do, and it consumes dynamic power during such idle states. A self-timed circuit would not be triggered and it would simply wait.

In a GALS system, the design is partitioned into a number of synchronous blocks, which are locally synchronized by a clock and communicate asynchronously with other synchronous blocks [5]. These blocks contain the functionality of the module and are developed using conventional synchronous design techniques. In such systems, modules that are not used frequently can be made to consume less power by either pausing their local clocks until they are needed, or simply by using a reduced local clock frequency (and/or supply voltage) for that particular module. It is also possible to optimize this approach by designing systems that dynamically adjust their frequency and or supply voltage on demand.
A GALS module design consists of port controllers that control the I/O ports. These controllers permit request stretching of local clock signals as shown in Figure 6.

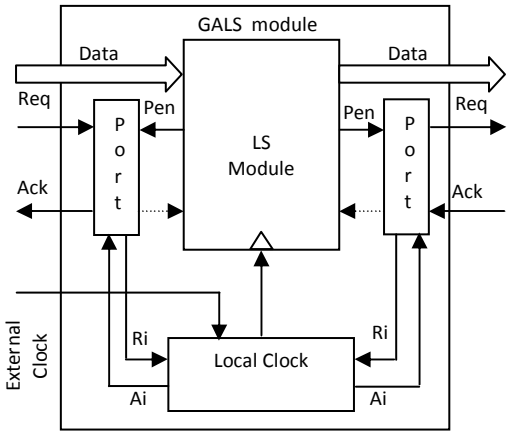

FIGURE 6: GALS module.

An asynchronous circuit consists of many sub-blocks that use handshake signals to request data from connected sub-blocks, and to respond to such requests. These handshake signals are generated locally in each sub-block (See Figure 7).

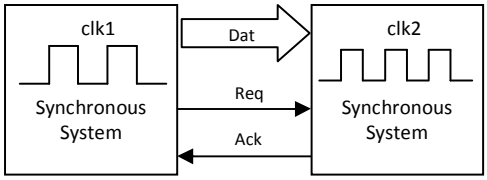

FIGURE 7: Small example of handshake communication.

The communicating parts are divided into active and passive parts, which are respectively either producing a request or waiting for a request. In Figure 8 the active port is the data sender. It outputs data and raises the request signal.

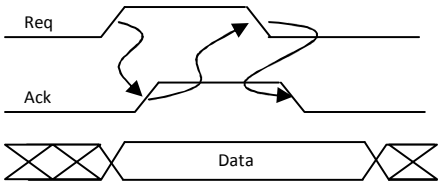

FIGURE 8: Four phase handshake protocol.

Furthermore, demand ports stop the internal clock as soon as the synchronous module tries to send/receive data. The internal clock is stopped under the handshake process (See Figure 9).

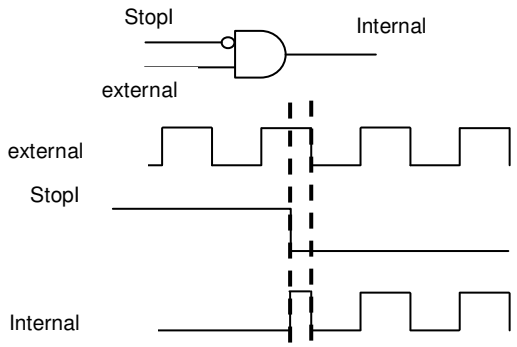

FIGURE 9: Clock synchronization with AND-Gate.

In GALS concept, the clock which skew constraints is reduced due to the fact that the extensions of the local clock trees become smaller [6]. The clock speed and power supply voltage of the synchronous blocks can be adjusted individually. The efficiency of GALS relies heavily on the presence of a small area, and low power on-chip clock generators for the local synchronous blocks [7].

3.2 Low Power GALS Based ECDSA Design

The synchronous ECDSA hardware description consists on global clock gating for each block (KP, RNG, and SHA). In this description only one clock signal is applied for all blocks. It is generated either if these blocks are functional or not. All events in the architecture are ordered by

this global clock signal as described in Section xx. In synchronous design the clock qualifies all data signals.

The circuit operates correctly as long as all signals within the design have their intended values at the time of the clock event. The clock signal for the locally synchronous unit is generated by a local clock generator. And, the data communication between GALS modules is governed by a specialized port controllers integrated in each block. These controllers are asynchronous finite state machines (AFSM) that can pause the local clock generator during data transfers in order to ensure data integrity (Figure 10).

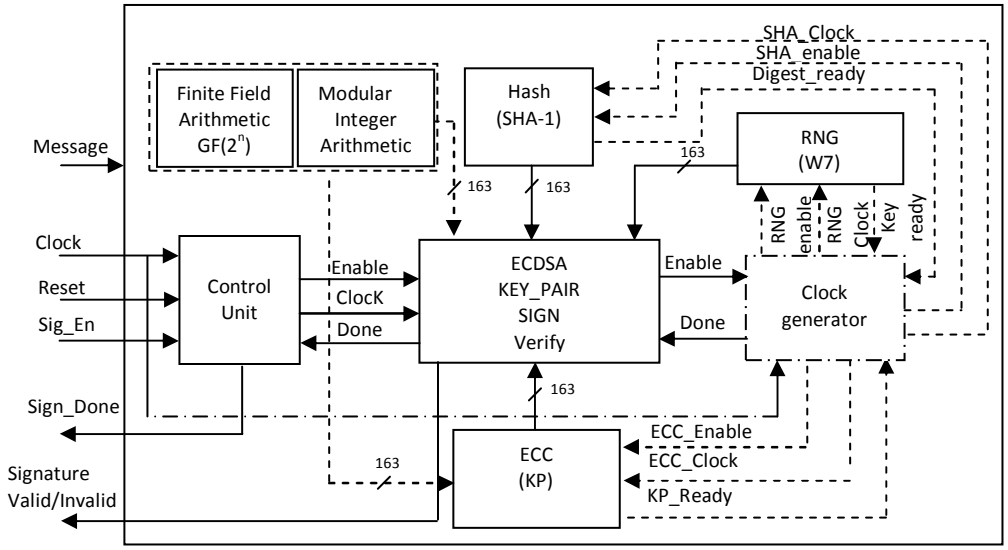

FIGURE 10: Proposed ECDSA GALS design.

When a subsystem is called, the block is activated and the appropriated clock is unlocked. Once achieved, the block is locked and the local clock is paused. Therefore, no new clock pulse will be generated, and the module will not consume dynamic power.

4. PERFORMANCE ANALYSIS

In this section, the performance of our proposed ECDSA blocks will be analyzed. We have prototyped our ECDSA hardware scheme using a reconfigurable hardware platform Virtex IV XC5VFX70t-2FF1136 FPGA [18]. All blocks are described in VHDL (VHSIC Hardware Description Language) using ModelSim ISE and synthesized using Xilinx ISETM Tools.

4.1 Power Consumption

Power efficiency is becoming important factors for algorithms running on different platforms. It is generally considered as the sum of static power and dynamic power [19]. In FPGAs, the static power consumption is dominated by the leakage current in the transistors. It represents the power used when the design is idle. The static power is equal to:

$$\text{Static Power} = VI \qquad (3)$$

The static power consumption of Virtex IV XC5VFX70t FPGA is about 1.099 mW. In the other side, the dynamic power dissipation is caused by signal transitions in the circuit. Frequent signal transitions and increased power dissipation are caused by higher operating frequency. The simple equation governing dynamic power consumption is:

$$\text{Dynamic Power} = C.V^2.f \qquad (4)$$

Where C is the capacitance of the node switching, V is the supply voltage, and f is the switching frequency.

4.2 Synthesis Results

The implementation of Elliptic Curve Digital Signature follows the guidelines as specified in ANSI X9.62. Elliptic curve domain parameters are defined over the finite field GF (2^n).

Table 1 summarizes the cost and performance of SHA-1 hash function, W7 random number generator and KP scalar multiplication designs. The cost is given as the number of maximum frequency, cycle's number, Slice LUTs and Flip Flops (FFs).

Design	Max. Freqency (MHz)	Cycles number	Slice LUTs	Flip Flops (FFs)
SHA-1	340.994	161	461	957
RNG	315.766	82	627	116
KP	194.876	109687	13016	6823

TABLE 1: ECDSA blocks design performances.

The scalar multiplication clearly dominates the design size with 29% of Slice Luts, 6823 of Flip Flops and a speed up of 0.562ms. The hash function SHA-1 presents a high frequency with 340.994 MHz (0.472 ms). The random number generator runs with a speed up of 0.258ms. As can be noticed, scalar multiplication presents the most consumed block in ECDSA design.

Table 2 deals with the comparison of our two proposed implementation of ECDSA hardware designs. The first is a synchronous design, and the second is a GALS based ECDSA design as described in Section 3. Measurement results showed that the performance metrics (Frequency, area, power consumption) of the GALS integration are comparable to circuits that were designed using conventional synchronous method.

	Max. Freqency (MHz)		Occupied Slices (Slices)		Power Consumption (mW)	
Design	**Synchro -nous**	**GALS based**	**Synchro -nous**	**GALS based**	**Synchro -nous**	**GALS based**
ECDSA key pair	210.172	211.356	3870	4916	114	101
ECDSA Generation	205.262	214.483	5157	5408	164	60
ECDSA Verification	141.699	148.963	10118	10304	185	67

TABLE 2: GALS and synchronous ECDSA design performances comparison over GF (2^{163}).

The experimental results show that, between the two proposed designs, the GALS based architecture benefits of less dynamic power consumption. We notice a reduction in power consumption by up to 13 mW in key pair generation process, 104 mW in signature generation and 118 mW in verification process.

According to what is reported in table 2, we notice that the different GALS based asynchronous processes architectures run with higher frequency than synchronous designs.

The simulation results show also that the GALS design for ECDSA reduce power consumption compared with their synchronous design, albeit with an increase in processing time and area occupation. This is because of added blocks and clocks used to reduce power dissipation of our proposed ECDSA architecture.
In spite of, the key pair generation runs with approximately the same rate, the signature generation process upgrade in GALS architecture compared with synchronous one, respectively from 0.609 to 1,901 ms. The ECDSA verification presents also latency up to 1.291 ms. In terms of area occupation, we notice that ECDSA GALS based design occupied slices, increases a little. However, the waste of occupied slices is about 2%, of total FPGA slices area, for digital signature generation/verification and 8% for key generation process.

In Table 3, we present performance of ECDSA signature generation and verification compared with the state of the art on different hardware resources. Table 3 gives an overview in comparison to some implementation in the literature. There are a lot more software implementations of ECDSA, on PC processors, over Galois fields (GF (p) and GF (2^n)), but it is not meaningful to compare our work with these results because of platform differences. Not many documents where we find the hardware ECDSA signature and the verification implementations, on FPGAs.

References	Hardware Resources	Finite Field Size (bits)	ECDSA Generation (ms)	ECDSA Verification (ms)
	PC processor implementations			
Cronin et all.[20]	**Pentium system**	GF(192)	31	47
eBACS [21]	**Intel Core 2 Duo**	GF(256)	1.88	2.2
Westhoff et all. [22]	**SHARP Zaurus**	GF(2^{163})	5.7	17.9
	FPGA implementations			
Benjamin et all. [23]	**Xilinx Virtex 5**	GF(256)	7.15	9.09
Gura et all.[24]	**Xilinx Virtex 2**	GF(2^{163})	0.635	0.815
Jarvinen et al. [25]	**Altera Cyclone II**	GF(2^{163})	0.94	1.61
Wajih et al. [27]	**Xilinx Virtex 2P**	GF(2^{163})	3.208	3.821
This paper Synchronous design	**Xilinx Virtex5**	GF(2^{163})	0.609	0.652
This paper GALS design	**Xilinx Virtex5**	GF(2^{163})	1,901	1,943

TABLE 3: Performance comparison of ECDSA signature generation/verification

compared with the state of the art.

These implementations are made over different finite fields: GF (p =192 and 256), in addition to Galois field GF (2^{163}). In [25] Jarvinen et al. present a Nios II-based ECDSA system on an Altera Cyclone II FPGA for a key length of 163-bit performing signature generation in 0.94ms and verification in 1.61ms. Benjamin et al. present in [23] an FPGA-based autonomous ECDSA system in, Virtex 5 FPGA, for longer key lengths of 256 bit containing all necessary subsystems for application in embedded systems on a reconfigurable hardware. They perform the two signature processes in 7.15 and 9.09 ms. Over the same finite field GF (2163) Gura et al. note times of 0.635and 0.815ms in ECDSA processes. In a previous work, wajih et al. performs the signature generation and verification respectively in 3.208 and 3.821 ms.

5. CONCLUSION

Power efficiency is becoming an important factors for algorithms running on different platforms. In this article, two hardware ECDSA implementations are studied: synchronous and GALS based designs. The synchronous architecture is a standard conception. It represents good results compared to works presented in literature. The main motivation for adapting GALS design techniques to ECDSA hardware implementation is to achieve reduced power consumption due to the global clock distribution. The power saving that can be achieved by this method depends entirely on how often the module is utilized.

The described GALS based ECDSA design possess many advantages, not only is there a mitigation in the clock distribution problems such as the decreased power consumption, and the problem of clock skew arising due to the large chip area, there is also a simplification in the reuse of modules. This task is made even more difficult by varying the clock period of each LS island randomly. This can be exploited to make DPA attacks more difficult.

Finally, because the security of an entire cryptographic system may depend on the quality and the unpredictability of the random numbers used, a random number generator is integrated in our proposed ECDSA design.

6. REFERENCES

[1] U.S. Department of Commerce, National Institute of Standards and Technology, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication FIPS PUB 186-2, January 2000.

[2] T.S. Chen, J.Yan Huang and T.L. Chen, “An efficient undeniable group-oriented signature scheme,“ *Applied Mathematics and Computation* 165, 2005, pp.95–102.

[3] D. Johnson, A. J. Menezes, and S. A. Vanstone, “The elliptic curve digital signature algorithm (ECDSA)”, *International Journal of Information Security*, 2001, pp.1:36-63.

[4] EBY G. FRIEDMAN , “Clock Distribution Networks in Synchronous Digital Integrated Circuits,“ proceeding of the IEEE, Vol. 89, NO. 5, MAY 2001.

[5] Krstic, M, Grass, E. and Gurkaynak, F.K., “Globally Asynchronous, Locally Synchronous Circuits: Overview and outlook, Design & Test of Computers,“ IEEE, Vol.24, Sept.-Oct. 2007, pp.430-441.

[6] A. Iyer and D. Marculescu, “Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors,“ In 29th Intl. Symp on ComputeArchitecture, May 2002.

[7] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott. Energy, “Efficient Processor Design Using Multiple Clock Domains with Dynamic

Voltage and Frequency Scaling," 8th Intl. Symp. on High-Performance Computer Architecture, Feb. 2002.

[8] N. Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 48:203, 209, 1987.

[9] V. Miller, Uses of elliptic curves in cryptography, In Lecture Notes in Computer Science 218: Advances in Cryptology-CRYPTO'85, pages417, 426 . Springer-Verlag, Berlin 1986.

[10] L. Batina, G. Bruin-Muurling, and S. B. Ors, "Flexible Hardware Design for RSA and Elliptic Curve Cryptosystems," Proceedings of Topics in Cryptology-CTRSA 2004, Lecture Note in Computer Science, Springer-Verlag, Vol. 2271, 2004, pp. 250-263.

[11] El hadj youssef WAJIH, Guitouni ZIED, Machhout MOHSEN, and Tourki RACHED, "Design and Implementation of Elliptic Curve Point Multiplication Processor over GF (2^n) ," International Journal of Computer Sciences and Engineering Systems (IJCSES), Vol.2, No.2, April 2008.

[12] National Institute of Standards and Technology (NIST). Secure Hash Standard, FIPS PUB 180-1, 2002.

[13] National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, Special Publication 800-57: Recommendation for Key Management. Part1: General Guideline. Draft 2003.

[14] S. Kim, K. Umeno, and Hasegawa, On the NIST Statistical Test Suite for Randomness, In IEICE technical Report, Vol. 103, No. 449, 2003 , pp. 21-27.

[15] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, O. Koufopavlou, and C.E. Goutis, "Comparison of the Hardware Architecture and FPGA Implementations of Stream Ciphers," Proceedings of 11th IEEE International Conference on Electronics, Circuits and Systems, Tel-Aviv, Israel, December 13-15, 2004.

[16] FIPS 186-2, Digital signature standard, National Institute of Standards ans Technology The Digital Signature Standard, 28 April 2004.

[17] ANSI X9.62, Public Key Cryptography for The Financial Service Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), American National Standards Institute, 16 novembre 2005.

[18] Virtex-5 FPGA Family Datasheet. Xilinx, Inc., San Jose, CA, 2007.

[19] Derek Curd, Power Consumption in 65nm FPGAs, Xilinx White Paper WP246 (v1.2), 1 February, 2007.

[20] CroninVipul Komathy K and Narayanasamy P, "Strengthening ECDSA Verification Algorithm to be More Suitable to Mobile Networks," Proceedings of the International Multi-Conference on Computing in the Global Information Technology, 2006.

[21] eBACS, ECRYPT Benchmarking of Cryptographic Systems, http://bench.cr.yp.to/ebats.html, 2010,

[22] D. Westhoff, B.Lamparter, C.Paar, and A. Weimerskirch, "On digitlal Signatures in Ad Hoc networks," European Transactions on Telecommunications, Special Issue: Self-Organisation in Mobile Networking Volume 16, Issue 5, pages 411–425, October 2005.

[23] Benjamin Glas, Oliver Sander, Vitali Stuckert, Klaus D.Muller-Glaser, and Jurgen Becker, "Prime Field ECDSA Signature Processing for Reconfigurable Embedded Systems," Hindawi Publishing Corporation, International Journal of Reconfigurable Computing, Article ID 836460, 12 pages, 2011.

[24] Gura, Chang Shantz, Eberle, Gupta, Gupta, Finchelstein, Goupy, Stebila, "An end-to-end systems approach to elliptic curve cryptography," In Proc. CHES 2002.

[25] K. Jarvinen and J. Skytta, "Cryptoprocessor for Elliptic Curve Digital Signature Algorithm (ECDSA) ," Tech. Rep., Helsinki University of Technology, Signal Processing Laboratory, 2007.

[26] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer Professional Computing Series, janvier 2003.

[27] Wajih, E.H.Y., Mohsen, M., and Rached, T., "A secure Elliptic Curve Digital Signature scheme for embedded devices," The 2nd International Conference on Signals, Circuits and Systems, 2008.