A RECONFIGURABLE GF(2^M) ELLIPTIC CURVE CRYPTOGRAPHIC COPROCESSOR

M. Morales-Sandoval[1,*], *C. Feregrino-Uribe*[2,◇], *R. Cumplido*[2,▽] *and I. Algredo-Badillo*[3,+]

[1]Polytechnic University of Victoria
Cd. Victoria, Tam. Mexico. 87138
*mmoraless@upv.edu.mx
[2]National Institute for Astrophysics, Optics and Electronics
Tonantzintla, Pue. Mexico. 72840
◇cferegrino@inaoep.mx, ▽rcumplido@inaoep.mx
[3]University of Istmo
Tehuantepec, Oax. Mexico. 70760
+algredobadillo@gmail.com

ABSTRACT

Elliptic Curve Cryptography (ECC) is a kind of cryptography that provides the security information services using shorter keys than other known public-key crypto-algorithms without decreasing the security level. This makes ECC a good choice for implementing security services in constrained devices, like the mobile ones. However, the diversity of ECC implementation parameters recommended by international standards has led to interoperability problems among ECC implementations. This work presents the design and implementation results of a novel FPGA coprocessor for ECC than can be reconfigured at run time to support different implementation parameters and hence, different security levels. Regardless there are several related works in the literature, to our knowledge this is the first ECC coprocessor that makes use of a partial reconfigurable methodology to deal with interoperability problems in ECC. A suitable application of the proposed reconfigurable coprocessor is the security protocol IPSec, where the domain parameters for ECC-based cryptographic schemes, like digital signature or encryption, have to be negotiated and agreed upon by the communication partners at run time.

1. INTRODUCTION

Elliptic curve cryptography (ECC) [1, 2] is a kind of public key cryptography whose main advantage is that it requires smaller key sizes compared to the keys used by traditional public key cryptosystems like RSA with the same security. Elliptic curve cryptography is founded on the mathematical properties of elliptic curves [3]. An elliptic curve over a field K is formed by the point O called point at infinity, and the set of points $P = (x, y) \in K \times K$ satisfying a (non-singular) Weierstrass equation 1.

$$E(K) : y^2 + a_1xy + a_3y = x^3 + a_2x_2 + a_4x + a_6 \quad (1)$$

The set $E(K)$ together with the identity point O form an additive abelian group $S = (E(K) \cup O, +)$ respect to the point addition operation '+'. The security of elliptic curve cryptography is based on the difficulty to solve the discrete logarithm problem, which is defined on S as: given two points $P, Q \in E(K)$, find the scalar d such that $Q = dP$. The operation dP is called scalar multiplication and it is the most time consuming operation in elliptic curve cryptographic schemes such as ECDSA (Elliptic Curve Digital Signature Algorithm) or ECDH (Elliptic Curve Diffie-Hellman) [4].

ECC-based cryptographic schemes need to define a tuple $T = (K, R, E, G, n, h)$, where K is a finite field, R is a basis for the elements of K, E is the elliptic curve defined over K, G is a generator point of the additive abelian group S, n is the order of G, that is, the smaller integer n such that $nG = O$, and h is the co-factor, the total number of points in the curve divided by n. Different fields K can be used and also different curves E on a fixed field. Several tuples T have been recommended for standards, like the National Institute of Standards and Technology NIST [5] or the Standards for Efficient Cryptography Group SECG [4]. The diversity of choices to implement ECC and the several tuples T recommended by international standards has led to interoperability problems.

In this context, interoperability is understood as the ability of two ECC implementations (either in software or hardware) to work together and communicate, for example one ciphering and the other deciphering. To achieve this interoperability, it is necessary that all participants in the communication have the standardized versions of common ECC-

978-1-4244-8848-3/11/$26.00 ©2011 IEEE

based crypto-algorithms using the same parameters T. However, most of the ECC hardware implementations of elliptic curve cryptography are focused on implementing efficiently the scalar multiplication operation dP [6, 7, 8, 9, 10, 11]. This efficiency is usually achieved by optimizing the circuits for specific tuples T which increases even more interoperability problems. Different to those approaches, this work aims to provide a flexible solution that can dynamically switch to different implementation parameters, instead of custom high performance solutions for a specific tuple T.

This article discusses a configurable system on programmable chip (CSoPC) that allows switching to different implementation parameters T of ECC at run time. The key piece in this CSoPC is a coprocessor that accelerates the elliptic curve computations and can be configured at run time to support different tuples T. The CSoPC was prototyped to the ML403 board, which includes a Virtex4 FPGA. Different tuples T recommended by NIST and SECG were used for validation. Although other works have used FPGAs to implement processors or coprocessors for ECC computations, this is the first work that reports a system where reconfiguration of the FPGA is performed at run time to support specific implementation parameters T of ECC. A suitable application of the proposed system is the security protocol IPSec, where the domain parameters for ECC-based cryptographic schemes, like digital signatures, have to be negotiated and agreed upon by the communication partners at run time.

The rest of this paper is organized as follows. Next section describes the coprocessor architecture and gives details about the practical implementation. The results are discussed in section 3 and finally, concluding remarks and further directions are presented in section 4.

2. ECC COPROCESSOR ARCHITECTURE FOR SCALAR MULTIPLICATION DP

An scalar multiplication dP is the result of adding the point $P \in E(K)$ to itself $d-1$ times. That is:

$$dP = \underbrace{P + P + P + \cdots + P}_{d-1 \ sums}$$

The scalar d must be in the range $[1, n-1]$. The '+' operation for elliptic curve point addition is defined for two operations: addition ECC-Add to sum two distinct points $P, Q \in E(K)$ and doubling ECC-Dbl to sum a point $P \in E(K)$ to itself. A software or hardware implementation of the scalar multiplication implies the choice of the algorithms to perform finite field arithmetic, to select the coordinate system, to represent the elliptic curve points and to select the algorithm to compute dP. Most of the works reported in the literature argue that the Montgomery ladder [12] is the best choice for computing dP while projective coordinates [13] are the best way to represent the elliptic curve points. This argument is based on the fact that finite field inversion is a very time consuming operation, requiring for its computation the same time required to compute six or more multiplications. However, for small area implementations, affine coordinates are better preferred because they require less operations and also less intermediate registers during the computations. For compact implementations, bit-serial field multipliers are preferred instead of digit-serial field multipliers. Field inversion or direct division with average cost of $2m-1$ clock cycles results in a similar cost of two serial multiplications. So in these cases the argument for using projective coordinates does not apply.

The operation dP is implemented using the coprocessor approach, avoiding the Fetch-Decode-Execute-Store cycle in the processor approach. The coprocessor approach requires less clock cycles because data are used immediately after they are available from previous modules, reducing the number of registers for storage but increasing the number of buses in the system. The coprocessor presented in this article is based on scalable $GF(2^m)$ arithmetic modules together with combinatorial logic, and finite state machines for control.

Hardware implementations of ECC should not only be efficient but also resistant to side channel attacks [14]. In these attacks extra source information such as timing, power consumption, electromagnetic emanations or even sound can be exploited to break the system. The binary method for computing dP parses every bit value of scalar d and executes at each iteration one ECC-Dbl operation followed by one ECC-Add only if the current bit value of d is '1'. The direct hardware implementation of this dP method is vulnerable to side channel attacks, such as the simple power analysis attack (SPA). In SPA, the attacker measures the power produced by the hardware executing the operation dP and tries to reveal the private key from those traces. An SPA attack for the hardware implementation of the binary method for dP is possible because ECC-Add and ECC-Dbl are different in essence and they will produce different power traces. Due the operations ECC-Add and ECC-Dbl are strongly related to the d's bits, the security of the system could be compromised.

One approach for preventing SPA attacks is to rewrite the addition formulas ECC-Add and ECC-Dbl so that a single formula can be used for both kinds of point sums, indistinctly [15]. This approach has been considered in the literature but using projective coordinates [6] or special forms of the elliptic curve [16]. However, addition and doubling operations are very similar in affine representation for elliptic curves defined over binary fields GF(2^m) so these two operations could also be unified. In [17] is presented a new formulation that unifies both ECC-Add and ECC-Dbl operations using affine coordinates. This new formulation al-

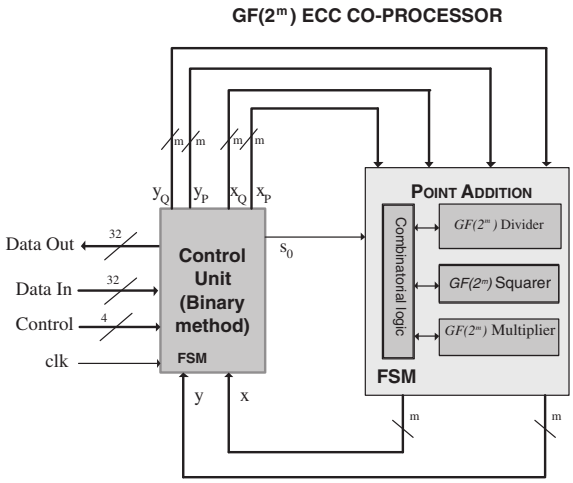

Fig. 1. Elliptic curve coprocessor for dP.

lows to reduce the hardware used for implementing the addition operation in elliptic curves and increases the resistance of the dP hardware implementation to side channel attacks by expressing two different operations as a single one. The ECC coprocessor presented in this work uses the single formulation of [17] as main block for field and elliptic curve arithmetic and its design incorporates the partial reconfigurable flow of Xilinx [18] to achieve a flexible hardware able to adapt dynamically its architecture to support different implementation parameters T of ECC.

The dP architecture presented in this work is based on finite state machines for reading the input parameters from a host computer, implementing the left to right version of the binary method [19] for scalar multiplication in elliptic curve cryptography, and finally, delivering the result back to the host computer when dP is obtained. The input parameters d and P are read as 32-bit words using a simple handshake protocol. In the same way, the result dP is delivered to the host computer as several 32-bit words. The block diagram of the dP processor is shown in figure 1.

3. RESULTS

The dP coprocessor was implemented in a Virtex4 FPGA using different tuples T. In order to study the impact, in terms of performance and space, of every module in the overall dP coprocessor, each module was synthesized separately..

Being GF(2^m) arithmetic the critical part in the dP coprocessor, different GF(2^m) arithmetic modules were designed and tested leading to different versions for the ECC coprocessor. The HDL design was described in such a way that the synthesis tools had a better translation from the high level description to the hardware resources in the FPGA. All the different versions of the arithmetic units were synthesized and optimized for speed or area so different results were achieved. The GF(2^m) divider was by far the critical module in the hardware architecture of dP coprocessor, occupying about one third of the area for the entire dP coprocessor. The results of the implementation of the dP coprocessor in the FPGA are shown in table 1.

The area optimization criteria allows to design more area reduced architectures at the cost of lower frequencies, which implies slower circuits for performing dP. The Virtex4 FPGA enables to implement the security levels of m=113, 131, 163, 233, 277 and 283 (this last only if area optimization is used). All the designs for these security levels were tested and validated by comparing the hardware results against software results.

3.1. The configurable system on programmable chip for interoperable Elliptic Curve Cryptography

The CSoPC for interoperable ECC is based on a partially reconfigurable design. Figure 2 shows the target system, which includes an embedded microprocessor, data buses, memory blocks, an universal asynchronous UART module and the dP coprocessor wrapped by the IPIF EDK core. The dP coprocessor could be partially configured for arbitrary tuples T. A C program running in the microprocessor enables the ECC coprocessor and sends the point P and scalar d through the PLB bus as a group of 32-bit words. After reading the input parameters, the ECC coprocessor starts the computation while the processor waits for the results. By asserting a signal, the coprocessor notifies the end of the computation and then the microprocessor reads back the results and shows them via the UART module to the user. The CSoPC was implemented using the ISE design flow for partial reconfiguration [18] EDK and PlanAhead from Xilinx.

The system in figure 2 was divided in a static part and a reconfigurable part. The static part is that part of the system implemented in the FPGA that will never change. It was composed of all the modules in figure 2 except the ECC coprocessor wrapped by the IPIF module, which is the reconfigurable part. All signals that connect the fixed and reconfigurable part cross through busmacros. Also, global FPGA resources like the buffered clock were locked and specified. The assignment of area for each part in the design, fixed and reconfigurable, the busmacro placement and global resources were specified in a constraint file using PlanAhead.

The ECC coprocessor uses three parameters from a given tuple T: the size m of the finite field GF(2^m), the irreducible polynomial $f(x)$ defining GF(2^m) and its arithmetic, and the elliptic curve E. All these parameters are available from the standards, like the ones proposed by NIST. The size m affects the size of the data buses in the coprocessor, the number of iterations in the finite field arithmetic modules and

Table 1. Synthesis results for the dP coprocessor optimized by speed or area.

Security level	113		131		163		233		277		283	
Optimization	Speed	Area	Speed	Area	Speed	Area	Speed	Area	Speed	Area	Speed	Area
Slices	2566	2109	3217	2467	3883	3034	4834	4236	5632	5086	5743	5191
Freq. (MHz)	136	93	139	90	145	87	134	78	114	84	114	84

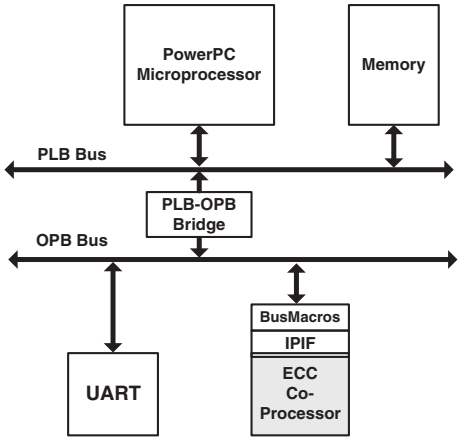

Fig. 2. Block diagram of the configurable system for ECC.

in the method for performing scalar multiplication. The irreducible polynomial $f(x)$ is used in finite field arithmetic and the elliptic curve is used in the point addition operation. Hence, for each tuple T for ECC there are specialized modules for field arithmetic, point addition and scalar multiplication using the parameters (m, $f(x)$, E). Several versions of the reconfigurable modules were implemented, not only because of the different tuples T but also because of the different hardware architectures designed for finite field arithmetic.

The CSoPC for interoperable ECC was prototyped using the ML403 board from Xilinx for validation and proof of concept purposes. The FPGA device is a Virtex4 which has 5,472 slices available and includes a PowerPC processor. The system was validated using test vectors generated by software models for different recommended security levels in [4] and [5]. The implementation results of the CSoPC are shown in table 2. In this implementation phase, the dP coprocessor was implemented using the area optimization criteria and the constraint for working with a clock of 100MHz, which is the working frequency of the ML403 board.

The time to perform the scalar multiplication is given in table 3. The coprocessor uses the clock frequency of the bus system, which is the same that the microprocessor clock, 100 MHz. All results were validated by applying test vectors and comparing the results from the coprocessor against the results obtained from a software implementation.

The aim of this work was to explore the design of an ECC coprocessor in FPGA that exhibits better performance

Table 2. Area results for the reconfigurable system.

Hw Resources	**Fixed part**	**Reconfigurable part** Security level (bits)	
		113	131
Flip-Flops	942	1,966	2,233
LUTs	891	4,535	4,053
Slices	1027	2,748	2,336
Dual Port RAMs	216	0	0
Shift registers	64	0	0
RAMB16s	16	0	0
Gate count	1,079,972	44,411	43,751

Table 3. Timing results for the reconfigurable dP coprocessor.

Sec. level	**Cycles/**dP	**Time** (*ms*)
113	51,730	0.52 ms
131	68,887	0.69 ms
163	107,043	1.07 ms
233	211,210	2.11 (Estimated)
277	305,922	3.05 (Estimated)
283	318,348	3.18 (Estimated)

than software solutions and at the same time, retains the flexibility of software to support different implementation parameters at run time. Then, the main contribution of this work lies in the ability of the system to be configured at run time depending on the security level required by an ECC-based cryptographic algorithm executed in the general purpose microprocessor.

As no other interoperable ECC hardware implementation has been reported in the literature, although it is not so fair, we have compared this work against not interoperable and higher performer ECC architectures. We provide a comparison table of the results achieved by this architecture against results reported in the literature, whose main purpose has been to achieve the highest performance without considering the ECC interoperability problems. In several cases, the reported works deal with custom implementations optimized for single implementation parameters T. In order to provide a fair comparison, the results achieved in this

work are compared against those works in the literature using the same conditions, that is, against related works using the same method for dP, the same elliptic curve representation and the same finite field. Also, the ECC coprocessor was implemented using the same FPGA than related works. The comparison results are shown in table 4.

Table 4. Comparison results against works under the same conditions

Ref.	m	Time *(ms)*	Device	Slices
[20]	151	5.1	XCV2000E	4048
	176	6.9		
	191	8.2		
	239	12.8		
[21]	113	3.7	XCV300-4	1290
	155	6.8		1567
	281	14.4		2622
This work	113	0.84	XCV2000E	2449
	131	1.25		2582
	163	2.09		3324
	113	1.05	XCV300-4	2515
	131	1.58		2516
	113	0.52	Virtex4	2405
	131	0.69		2871
	163	1.07		3528

Besides custom hardware implementations, some parameterizable hardware architectures have been reported in the literature that have used different arithmetic algorithms in order to achieve a faster dP operation. In this work it was found that reduced area designs are desired due the problems experienced when the reconfigurability of the coprocessor was implemented. The proposed coprocessor uses fewer resources than Kerins *et al.* [20] and computes dP faster. Our coprocessor is also faster than the coprocessor reported by Leong and Leung [21] at the expense of higher area resources.

Table 5 shows the comparison results of the proposed dP coprocessor against other works that have used projective coordinates, like [7] (10.9 ms for $m = 113$), [9] (3.8 ms for $m = 160$) or [6] (2.47 ms for m = 179). The use of projective coordinates supposes a better performance because inversions are avoided at each point addition operation at the cost of more multiplications. However, as it is shown in table 5, the proposed dP coprocessor achieves better timing than these works.

Other works using projective coordinates perform dP faster than the coprocessor presented in this article but they require higher area resources. For example, [10] uses 4,749 slices from a Virtex2 Pro and performs the dP in the field $GF(2^{163})$ in 0.49 ms. In [8], the area required is 19,000 slices from a XCV2000E FPGA while the dP operation in the field $GF(2^{163})$ is computed in 0.14 ms. While the area used in [8] is six times bigger that the area used by the proposed coprocessor, and the one used in [6] is three times bigger. In [22], the operation dP is performed in half the time of in this work but the required area is almost three times bigger.

Table 5. Comparison results against works using projective coordinates.

Ref.	m	Time *(ms)*	Device	Slices
[6]	179	2.47	XCV800	10,626
[23]	251	21.8	XC3S5000	19,304
[24]	163	0.66	V2Pro	8769
[7]	113	10.9	AT94K40	-
[8]	163	0.14	XCV2000E	19,000
[9]	160	3.81	XCV800	-
[10]	163	0.49	V2Pro	4,749
[25]	163	0.07	XCV2000E	5,008
[11]	191	0.05	VirtexE 3200	18, 314
[26]	192	2.21	FPSLIC	4907 CLBs
[22]	113	0.27	XC2V6000	6,961
This work	113	0.84	XCV2000E	2449
	131	1.25		2582
	163	2.09		3324
	113	1.05	XCV300-4	2515
	131	1.58		2516
	113	0.52	Virtex4	2405
	131	0.69		2871
	163	1.07		3528

The results obtained by the coprocessor discussed in this work are not only competitive or better that the reported in the literature but also are the results of the first coprocessor that provides the faculty of adapting at run time the hardware to support the costly arithmetic in elliptic curve cryptography.

4. CONCLUDING REMARKS AND DIRECTIONS

This paper presented a configurable system on reconfigurable chip for scalar multiplication dP. It is well suited for protocols like IPSec where the parameters of the ECC cryptoalgorithms are negotiated at run time. Dynamic reconfiguration of the ECC coprocessor allows supporting different security levels while retaining high performance. An efficient coprocessor for ECC using affine representation was discussed. This coprocessor performs as well as those that use projective representation while the architecture design is simpler and modular, so the system can be updated with better performer modules for finite field arithmetic, which still determine the whole latency for the dP operation. The single formula for ECC point addition helps to make the elliptic curve point addition operations indistinguishable, which increases the resistance of the dP hardware implementation to SPA attacks. This first reported partially reconfigurable solution is well suited to provide dynamic adaptation to different tuples T and hence to provide interoperability in elliptic curve cryptography. Further work is needed to design

extremely light-weight (low area) ECC in hardware. Low-power ECC is necessary mainly if the coprocessor is used in constrained devices. Additionally, self reconfiguration of the coprocessor could be pursued.

5. REFERENCES

[1] V. Miller, "Use of Elliptic Curves in Cryptography," in *Proc. of Advances in Cryptology, CRYPTO'85*, Santa Barbara, CA, 1985, pp. 417–426.

[2] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, November 1987.

[3] I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic curves in cryptography*. New York, NY, USA: Cambridge University Press, 1999.

[4] SEC 1, "Elliptic curve cryptography: Standards for efficient cryptography group," 2000, http://www.secg.org.

[5] NIST, "Recommended elliptic curves for federal government use," 1999, http://csrc.nist.gov/csrc/fedstandards.html.

[6] L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede, "Balanced point operations for side-channel protection of elliptic curve cryptography," in *IEE Proceedings of Information Security*, vol. 152, 2005, pp. 57–65.

[7] M. Ernst, M. Jung, F. Madlener, S. Huss, and R. Blumel, "A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over GF(2^n)," in *Proc. of CHES'2002*, ser. LNCS, vol. 2523. Redwood Shores, CA: Springer, 2002, pp. 381–399.

[8] N. Gura, S. C. Shantz, H. Eberle, S. Gupta, and V. Gupta, "An End to End Systems Approach to Elliptic Curve Cryptography," in *Proc. of CHES 2002*, ser. LNCS, vol. 2523. Springer, 2002, pp. 349–365.

[9] N. Mentens, S. B. Ors, and B. Preneel, "An FPGA Implementation of an Elliptic Curve Processor GF(2^m)," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, Boston, MA, 2004, pp. 454–457.

[10] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "Superscalar coprocessor for high-speed curve-based cryptography," in *Proc. of CHES 2006*, ser. LNCS, vol. 4249. Springer-Verlag, 2006, pp. 415–429.

[11] N. Saquib, F. Rodriguez, and A. Diaz, "A Parallel Architecture for Fast Computation of Elliptic Curve Scalar Multiplication over GF(2^n)," in *Proc. of RAW'04*, Sta. Fe, USA, 2004, pp. 26–27.

[12] J. López and R. Dahab, "Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation," in *Proc. of CHES'99*, ser. LNCS, vol. 1717. Berlin: Springer, 1999, pp. 316–327.

[13] ——, "Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$," in *Proc. of Selected Areas in Cryptography*, ser. LNCS, vol. 1556. Springer, 1998, pp. 201–212.

[14] M. Joye, "Elliptic curves and side-channel analysis," *ST Journal of System Research*, vol. 4, no. 1, pp. 17–21, 2003.

[15] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity," *IEEE Trans. Comput.*, vol. 53, no. 6, pp. 760–768, 2004.

[16] T. Izu and T. Takagi, "A fast parallel elliptic curve multiplication resistant against side channel attacks," in *Proc. of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*. London, UK: Springer-Verlag, 2002, pp. 280–296.

[17] M. Morales-Sandoval, C. Feregrino-Uribe, R. Cumplido, and I. Algredo-Badillo, "A single formula and its implementation in FPGA for elliptic curve point addition using affine representation," *Journal of Circuits, Systems, and Computers*, vol. 19, no. 2, pp. 425–433, 2010, dOI: 10.1142/S0218126610006153.

[18] Xilinx Inc., "Two flows for partial reconfiguration: Module based or difference based," Application Note 290, XAPP290., September, 9. 2004, www.xilinx.com.

[19] D. Hankerson, L. López, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," in *Proc. of CHES'2000*, ser. LNCS, vol. 1965. Worcester, MA: Springer, August 2000, pp. 1–24.

[20] T. Kerins, E. Popovici, W. Marnane, and P. Fitzpatrick, "Fully Parameterizable Elliptic Curve Cryptography Processor over GF(2^m)," in *Proc. of 12th FPL'2002 Conference*, ser. LNCS, vol. 2438. Montpellier, France: Springer, 2002, pp. 750–759.

[21] P. Leong and K. Leung, "A Microcoded Elliptic Curve Processor Using FPGA Technology," *IEEE Trans. VLSI Syst.*, vol. 10, no. 5, pp. 550–559, October 2002.

[22] R. Cheung, N. Telle, W. Luk, and P. Cheung, "Customizable elliptic curve cryptosystems," *IEEE Trans. VLSI Syst.*, vol. 13, no. 9, pp. 1048–1059, Sept. 2005.

[23] N. Mentens, K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "A side-channel attack resistant programmable pkc coprocessor for embedded applications," in *International Conference on Systems, Architectures, Modeling and Simulation (IC-SAMOS 2007)*, 2007, pp. 194–200.

[24] L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede, "Flexible hardware architectures for curve-based cryptography," in *Proc. of ISCAS 2006*, 2006.

[25] J. Lutz and A. Hasan, "High performance fpga based elliptic curve cryptographic co-processor," in *ITCC'04: International Conference on Information Technology: Coding and Computing*, vol. 2. IEEE Society Press, 2004, pp. 486–492.

[26] S. Janssens, J. Thomas, W. Borremans, P. Gijsels, I. Verbauwhede, F. Vercauteren, B. Preneel, and J. Vandewalle, "Hardware/software co-design of an elliptic curve public-key criptosystem," in *2001 IEEE Workshop on Signal Processing Systems*, 2001, pp. 209 – 216.