Tatra Mt. Math. Publ. **33** (2006), 41–50

TATRA MOUNTAINS
Mathematical Publications

EFFICIENT HARDWARE IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOSYSTEMS

Krzysztof Gołofit

ABSTRACT. The paper focuses on a fast hardware implementation of the most difficult and time consuming operation within elliptic curve cryptosystems: the multiplication of point. When applied, it can significantly boost the overall performance making it possible to be implemented at the lowest level of design: *full custom* VLSI. The paper starts with a short introduction to ECC using optimal normal bases over finite fields of characteristic two. The main part describes an idea of a cryptographic accelerator and its model designed in the VLSI technology (a 0.35 μm process using 3 metallization layers). Then the results are analyzed and projected for more advanced technologies giving an estimation of achievable parameters.

1. Introduction

It is a common knowledge (among cryptographers at least) that the ECC algorithms provide more security than the traditional public key schemes with the same key length and result in shorter key lengths as their classic equivalents of the same security. It is because the exponentiation in a multiplicative group of a finite field is replaced with the multiplication of a point on an elliptic curve with the underlying hard problem much more difficult to be solved (as it is believed). It has been estimated that security level for 160-bit elliptic curve key is an equivalent to 1024-bit RSA key. But there is the usual tradeoff—shorter key with more security means, more difficult implementation (both in software and hardware) originating from larger computational complexity of ECC algorithms.

Actually most of the arithmetical operations in finite fields (focusing on those of characteristic two) may be reduced to series of multiplications (exponentiation is obvious, inversion and division to follow). Addition and squaring are trivial, and are used to reduce the computational problems of multiplication. Still, when elliptic curves are concerned, we have to perform much more elementary multiplications than in the traditional algorithms. Comparing computation

2000 Mathematics Subject Classification: 14G50.
Keywords: elliptic curves, hardware accelerator, fast multiplication, VLSI.

complexity of both methods (the multiplicative group exponentiation and the elliptic curve point multiplication) in traditional schemes we need 1/2 statistical multiplication per 1 bit of the randomly chosen key (assuming that squaring is done by rotating bits—therefore insignificant in computations), and from 16.5 to 22.5 statistical multiplications per 1 bit in ECC schemes (depending on binary field extension 163–571, according to [8]). This deficiency may be made up by implementations at the lowest possible design level—VLSI (Very Large Scale Integration) technology. With some specific design tips taking into account the fixed nature of ECC parameters one may expect to get even more efficient solutions when the newest achievements of the technology become available.

2. Efficient multiplication

From the contemporary number theory we get many methods to simplify and optimize almost every arithmetical operation in finite fields. Of course multiplication still remains the most computationally complex operation. For hardware implementation, binary fields (arithmetic in $\mathrm{GF}(2^m)$), are useful, since the operations involve only shifts and elementary operations like conjunction and bitwise addition modulo 2. When Optimal Normal Bases (ONB) are used multiplication becomes especially undemanding. Simplicity results from optimal complexity of the multiplication matrix T_k (size $m \times m$), which is used for multiplication of one bit c_k.

$$c_k = AT_kB^t \tag{1}$$

$A = [a_0a_1 \ldots a_{m-1}]$ and $B = [b_0b_1 \ldots b_{m-1}]$ are the multiplied vectors, c_k is the indexed bit from the result vector $C = [c_0c_1 \ldots c_k \ldots c_{m-1}]$, $0 \leqslant k \leqslant m-1$, and m is the extension of the binary field $\mathrm{GF}(2^m)$.

Optimal complexity means that the quantity of the nonzero elements C_N in every multiplication matrix T_k has the minimal value ($C_N = 2m-1$). Practically we have at the most two nonzeros in every row and every column in the matrix. Transformation from the matrix T_k to the matrix T_{k+1} results in a regular slanting shift of the elements in the matrix which can be easily implemented in hardware.

A more accurate description of optimal normal bases, used multiplication methods, construction of the multiplication matrixes as well as specification of the following implemented algorithms can be found in [8] and [9].

The first idea for the hardware VLSI implementation is to build an array following almost directly the theoretical basis. The main cell of the array consists of three elements: flip-flop, binary conjunction and addition modulo 2. Multiplied vectors A and B are applied (respectively) horizontally and vertically to the

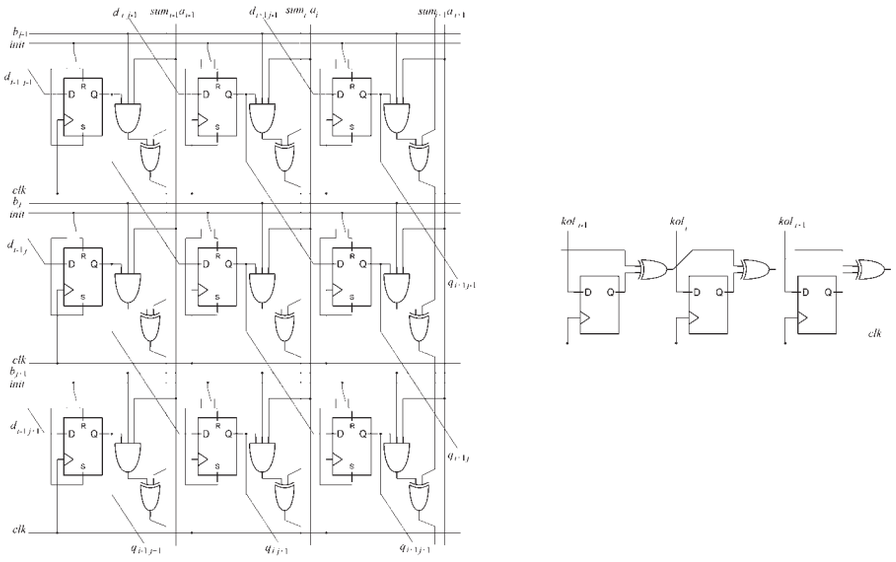

FIGURE 1. Cyclic multiplication: multiplication array (left) and vertical signals addition (right).

array while flip-flops are holding values analogous to the current multiplication matrix T_k.

Following the formula

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j t_{ij}^{(k)} \tag{2}$$

every single cell of the array is performing binary multiplication of bits $t_{ij}^{(k)}$ (elements t_{ij} of the matrix T_k), a_i and b_i. All the results are added first vertically (in the columns), then horizontally (under the array) consequently giving the binary value of c_k. Cyclic process together with regular slanting shifts (between the flip-flops) provides us with the whole vector C. Unfortunately we can observe cubic dependence of processing time on field extension m (this is due to the sequential nature of the algorithm: vertical addition, horizontal addition, cyclic process). Computation time can be shortened from cubic to square dependence by making the second addition in the cycle for the next bit while the next vertical addition is processed. Nevertheless, such dependencies can result in too long times of circuit operation.

An example of a practical implementation of the complete array model for GF(2^{11}) with the vertical signals addition (at the top of the array) is presented below.

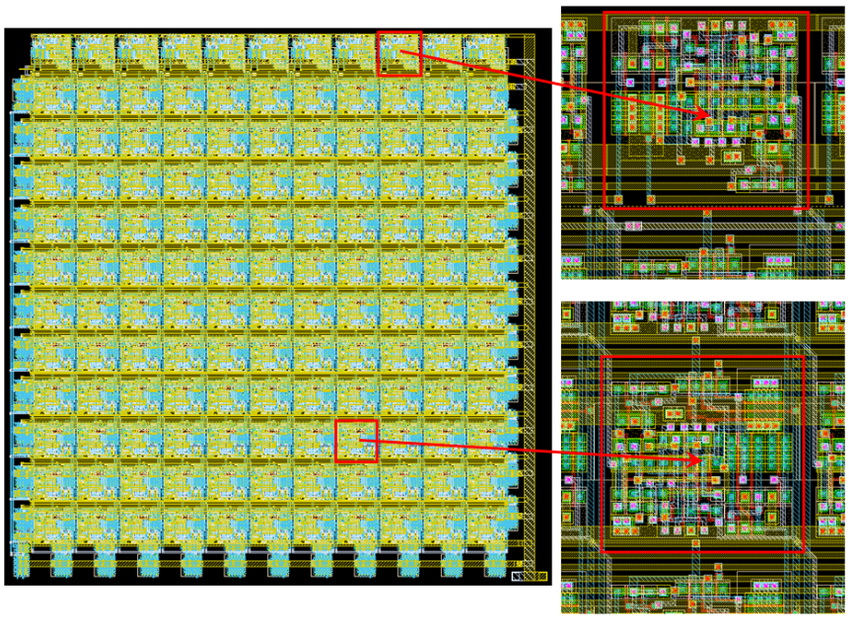

FIGURE 2. Layouts of the cell structure for the cyclic multiplication circuit.

In view of the fact that the computations on an elliptic curve require a few times shorter field extensions than the exponentiation in a multiplicative group (between 163 and 571 according to NIST), it is possible to consider different design criteria. The difference lies in the fact that we resign from the universal solution in favour of the parallel and fixed method. Making m multiplication arrays (each for the different matrix T_k) there is no more need to build the flip-flops (as a matter of fact most of area-consuming factor is omitted). Remembering that two nonzero elements at the most appear in every column, we can arrange all the computation gates into the line and cross only connections of them.

The square dependence of gates area on m is obvious, as we have to build m similar constructions (as illustrated on Figure 3)—but unfortunately crossed connections result in cubic dependence of area on m (this parameter will be inseparably related to applied technology).

Systematic structure of the cell construction for the multiplication over GF(2^{11}) is illustrated below. The cubic dependence of area is marked at the top of the middle picture.

Comparing the two solutions (the cyclic and the parallel), the most significant point is that the square dependence of computation time and circuit area

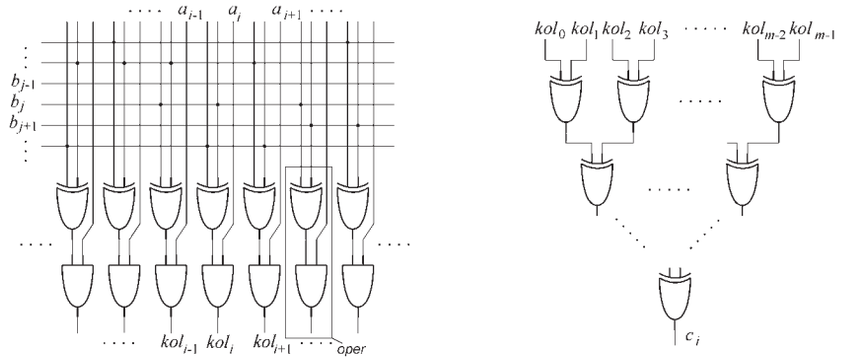

FIGURE 3. Parallel multiplication: fast multiplication for one bit (left) and fast vertical signals addition (right).

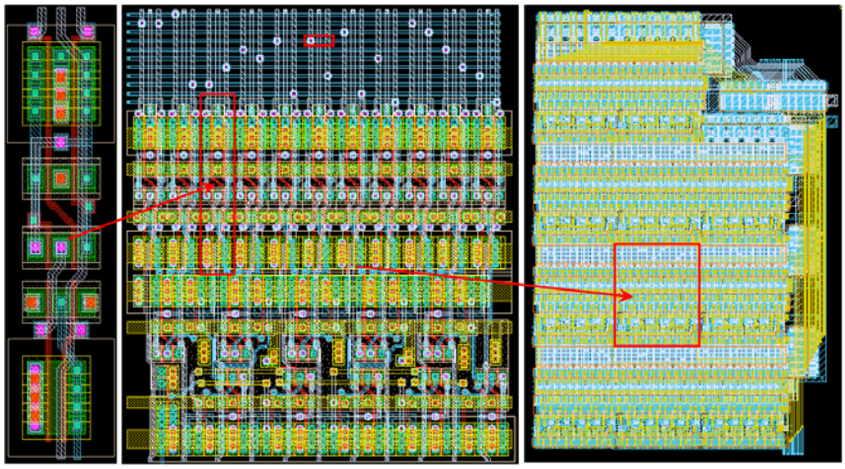

FIGURE 4. Layouts of the cell structure for the parallel multiplication circuit.

was exchanged for the cubic dependence of area and negligible (approximately $\lceil \log_2 m \rceil$) dependence of time on m. There are also other meaningful features set in Table 1.

The two implementations have been modeled for GF(2^{11}) in the VLSI technology 0.35 μm (full custom) using 3 layers of metallization at a Cadence platform available at the university. The models are obviously not practical and are not in the range of cryptographic interest, but they give sound grounds for some assessment of basic parameters in more advanced technologies. The numbers are not precise, but their orders of values are reliable.

TAB. 1. The comparison of the two solutions of multiplication.

Cyclic multiplication (universal)	Parallel multiplication (fast)
− square dependence of time on m	+ negligible dependence of time on m
± square dependence of area on m	− cubic dependence of area on m
± square dependence of power on m	± square dependence of power on m
− need of synchronization	+ asynchronous mechanism
+ applicable to any field of char. 2	− applicable only to ONB
+ exchangeability of speed versus dissipated power and circuit area	− fixed circuit structure
+ simple implementation resulting from highly modular structure	− difficult implementation resulting from irregular interconnections among m matrixes

TAB. 2. The obtainable parameters extrapolated for different technologies.

Technology	Type of circuit	Max. field extension	Area [cm^2]	Speed [mul./sec.]	Number of transistors
0.80 μm, 3 met.	*Cyclic*	$m = 283$	1.08	15 k	3.4 M
0.80 μm, 3 met.	*Parallel*	$m = 163$	0.82	120 M	270 k
0.35 μm, 5 met.	*Cyclic*	$m = 409$	0.99	16 k	7.0 M
0.35 μm, 5 met.	*Parallel*	$m = 283$	1.33	260 M	800 k
0.18 μm, 6 met.	*Cyclic*	$m = 571$	0.99	16 k	14 M
0.18 μm, 6 met.	*Parallel*	$m = 409$	0.65	510 M	1.7 M
0.09 μm, 7 met.	*Cyclic*	$m = 571$	0.49	33 k	14 M
0.09 μm, 7 met.	*Parallel*	$m = 571$	0.75	920 M	3.3 M

Attempts were made to fit both circuits onto a chip of an area of about one square centimeter (4^{th} column) with the field extensions according to NIST standards (3^{rd} column). Considering ECC approach (for computational and cryptographical reasons) we stay focused only on solutions applicable in practice. For example using 0.80 μm technology we can build one cyclic multilayer for m equal to 283 at most and one parallel for the extension 163. The biggest differences between circuits parameters can be observed for 0.09 μm technology where both implementations can be done for m equal to 571 (the cyclic circuit even with the area of $0.49\,\mathrm{cm}^2$). This technology makes it possible to perform almost 1 billion multiplications per second with the aid of the parallel solution.

The achievements of contemporary technologies (like additional layers of metallization, for example) give us opportunities to significantly decrease the circuits areas. An exact report on the approach of the implementation, series of the optimization depending on technology trumps, useful equations and detailed description of parameters prediction are out of scope of this paper, and can be found in [8] and [9].

3. Elliptic curve computation

An effective implementation of the point multiplication $k \cdot P$ on an elliptic curve is usually done as a sum of doubled points (following this equation)

$$k \cdot P = \left(\sum_{i=0}^{n-1} k_i 2^i \right) \cdot P = \sum_{i=0}^{n-1} k_i \cdot \left(2^i \cdot P \right), \tag{3}$$

where n is the binary length of multiplier k and $k_i = \{0, 1\}$. Statistically we need one and a half of the points addition (treating the point doubling as an addition of the point to itself) per one bit of the randomly chosen key k. The operations for a sample key can be illustrated in the following way:

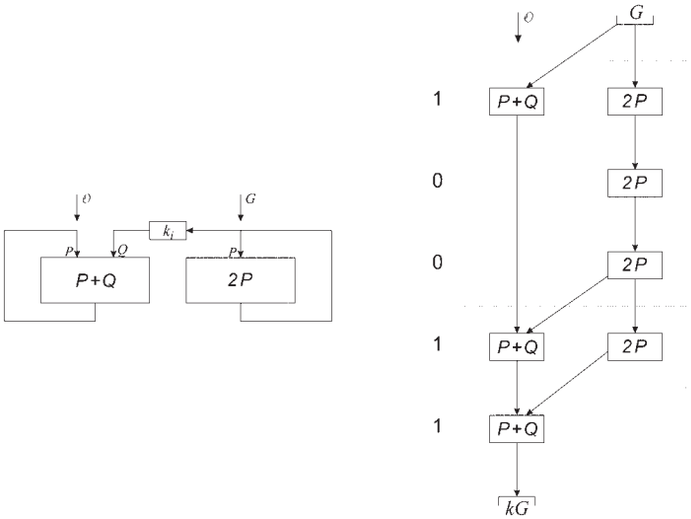

FIGURE 5. An effective algorithm of the point P multiplication on an elliptic curve (left) and an example of usage for the key (10011) (right).

where $P + Q$ represents the points addition on an elliptic curve and 2P the doubling of the point P.

The number of concurrent multiplications to be performed in one step addition forms the main barrier for a practical implementation of such circuits. When optimal normal bases are used, the addition and doubling of a point $P = \{x, y\}$ follow these two equations (Figure 6 presents layouts of the points addition)

$$P + Q = \begin{cases} x_s = \left(\frac{y_p + y_q}{x_p + x_q} \right)^2 + \frac{y_p + y_q}{x_p + x_q} + x_p + x_q + a\,, \\ y_s = \left(\frac{y_p + y_q}{x_p + x_q} \right)(x_p + x_s) + x_s + y_p\,, \end{cases} \tag{4}$$

$$2 \cdot P = \begin{cases} x_r = \left(\frac{x_p^2+y_p}{x_p}\right)^2 + \left(\frac{x_p^2+y_p}{x_p}\right) + a\,, \\ y_r = \left(\frac{x_p^2+y_p}{x_p}\right)(x_r + x_p) + y_p\,. \end{cases} \tag{5}$$

Each equation needs one multiplication and one division—squaring (as a one bit rotation) and addition are insignificant. Division can be performed as a multiplication by an inversion. The inversion depending on the binary finite field extension can be done as several multiplications and squarings (what is known from computational number theory). For example over $\mathrm{GF}(2^{11})$, one inversion needs four multiplications, and what follows one points addition requires six multiplications altogether. The computation complexity of one inversion for the standard field extensions are shown below.

TAB. 3. Computation complexity of the inversion for different field extension (according to NIST).

Field extension (optimal normal bases)	Number of multiplications needed for one inversion
$m = 163$	9 mul./inv.
$m = 233$	10 mul./inv.
$m = 283$	11 mul./inv.
$m = 409$	11 mul./inv.
$m = 571$	13 mul./inv.

Essentially we need two multiplications and one inversion for each curve operation (each of the equations: 4 and 5), what gives us from 16.5 to 22.5 statistical multiplications per one bit depending on field extension. Shamir's trick in such hardware solution cannot be implemented.

Such schemes were implemented in a model of an elliptic curve arithmetic unit using fast multiplication (characterized by parallel computations). Consequently, it was used for determining the implementation perspectives.

It is worth mentioning that many practical solutions (as well as the discussed case) take advantage of the Koblitz curves but it is out of scope of this paper.

In the Table 4 we have three parameters for each selected extension and technology: the implementation possibilities (construction fitting onto a chip of an area of about one square centimeter), the bit rate (one key bit operations per one second) and number of possible signatures per one second. Digital signatures like DSA, Schnorr, ElGamal (as well as their ECC equivalents: EC-DSA, EC-Schnorr, EC-ElGamal) or signatures verifications consist of two asymmetric transformations (two point multiplications in ECC case). Also the proper lengths of the keys commensurate with the provided security were taken into consideration.

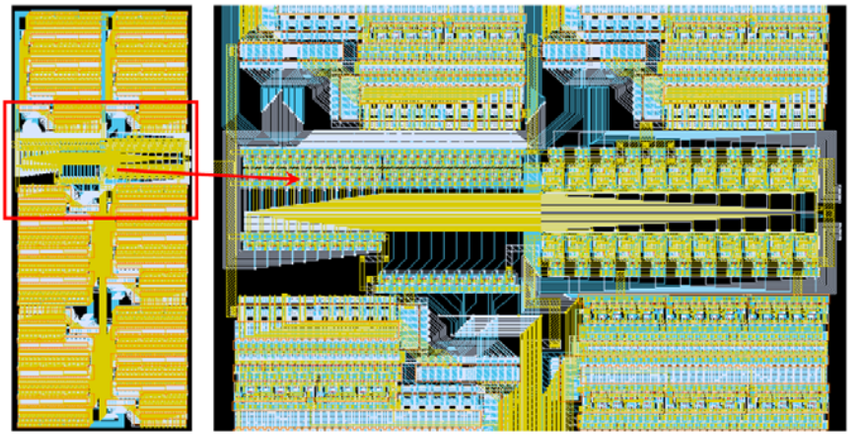

FIGURE 6. Layouts of the cell structure for the addition (left) and addition control unit (right).

TAB. 4. Implementation perspectives for fast elliptic curve arithmetic unit.

Tech. / Exten.	**0.18 μm; 6 metallization layers**		**0.09 μm; 7 metallization layers**	
	[key bits / sec.]	[signatures / sec.]	[key bits / sec.]	[signatures / sec.]
$m = 163$	1EC circuit, area: 1.41 cm^2		2 EC circuits, area: 1.13 cm^2	
	47 M	**144 k**	2 x 94 M	2 x **288 k**
$m = 233$	-		1 EC circuit, area: 1.63 cm^2	
	-	-	86 M	**185 k**
$m = 283$	-		1/2 EC circuit, area: 1.2 cm^2	
	-	-	37 M	**65 k**

A precise specification of circuit formation process, assumptions of modeling as well as a description of parameters prediction and formal equations can be found in [8].

4. Conclusions

Relatively small field extensions ($m \leqslant 571$) and advantages of contemporary technologies give us an opportunity for making parallel multiplications and fast elliptic curve implementations. Technology 0.09 μm makes it possible to do almost 300 thousand signatures (or signature verifications) for $m = 163$ and above 180 thousand signatures (or verifications) for $m = 233$.

As usual, the faster and bigger parallel circuits would probably be used in highly specialized hardware accelerators (coprocessors) in digital signature systems (or time-stamping servers), while the universal cyclic circuits can be implemented into not so demanding cryptographic cards used by individual users.

REFERENCES

[1] GAO, S.: *Normal Bases over Finite Fields*, PhD Thesis, University of Waterloo, Ontario, Canada 1993.

[2] GAO, S.—VANSTONE, S. A.: *On orders of optimal normal basis generators*, University of Waterloo, Ontario, Canada 1994, Math. Comp. **64** (1994), 1224–1233.

[3] KOBLITZ, N.: *Algebraiczne Aspekty Kryptografii*, WNT, Warszawa, 2000.

[4] NIST: *Digital Signature Standard* (*DSS*), (FIPS PUB 186-2), U.S. Department of Commerce, 2000.

[5] GAWINIECKI, J.—SZMIDT, J.: *Zastosowanie Ciał Skończonych i Krzywych Eliptycznych w Kryptografii*, WAT, Warszawa, 1999.

[6] KALISKI, B. S., JR.—KOC, K. C.—PAAR, C.: *Cryptographic Hardware and Embedded Systems—CHES '02*, Springer–Verlag, Berlin, 2002.

[7] ROSING, M.: *Implementing Elliptic Curve Cryptography*, Manning Publications Co., Connecticut, 1998.

[8] GOŁOFIT, K.: *Implementacja Systemów Podpisów Cyfrowych Wykorzystujacych Krzywe Eliptyczne*, Master thesis, Instytut Systemów Elektronicznych, Politechnika Warszawska, 2003. (In Polish)

[9] GOŁOFIT, K.: *Implementacja mnożenia w technologii VLSI dla zastosowań krzywych eliptycznych*, III Krajowa Konferencja Elektroniki, Kołobrzeg Politechnika Koszalińska – Wydział Elektroniki, 2004.

Received October 9, 2004

Faculty of Electronics and Information Technology
Warsaw University of Technology
ul. Nowowiejska 15/19 p. 249
PL–00-665 Warszawa
POLAND

E-mail: kgolofit@elka.pw.edu.pl