Computers and Electrical Engineering 39 (2013) 542–549

Contents lists available at SciVerse ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Area/performance trade-off analysis of an FPGA digit-serial $\mathbb{GF}(2^m)$ Montgomery multiplier based on LFSR ☆

M. Morales-Sandoval [a], C. Feregrino-Uribe [b], P. Kitsos [c], R. Cumplido [b,*]

[a] Polytechnic University of Victoria, Information Technology Department, Mexico
[b] National Institute for Astrophysics, Optics and Electronics, L. Enrique Erro No. 1, Santa. Ma. Tonantzintla, Puebla 72840, Mexico
[c] Hellenic Open University, School of Science and Technology, Digital Systems & Media Computing Laboratory, Tsamadou 13-15, GR-26222 Patras, Greece

ARTICLE INFO

Article history:
Received 13 October 2011
Received in revised form 29 August 2012
Accepted 30 August 2012
Available online 11 October 2012

ABSTRACT

Montgomery Multiplication is a common and important algorithm for improving the efficiency of public key cryptographic algorithms, like RSA and Elliptic Curve Cryptography (ECC). A natural choice for implementing this time consuming multiplication defined on finite fields, mainly over $\mathbb{GF}(2^m)$, is the use of Field Programmable Gate Arrays (FPGAs) for being reconfigurable, flexible and physically secure devices. FPGAs allow the implementation of this kind of algorithms in a broad range of applications with different area–performance requirements. In this paper, we explore alternative architectures for constructing $\mathbb{GF}(2^m)$ digit-serial Montgomery multipliers on FPGAs based on Linear Feedback Shift Registers (LFSRs) and study their area–performance trade-offs. Different Montgomery multipliers were implemented using several digits and finite fields to compare their performance metrics such as area, memory, latency, clocking frequency and throughput to show suitable configurations for ECC implementations using NIST recommended parameters. The results achieved show a notable improvement against FPGA Montgomery multiplier previously reported, achieving the highest throughput and the best efficiency.

1. Introduction

Public key cryptography [1] is a kind of cryptography used for ensuring the security services of confidentiality, integrity and authentication on digital information. Generally, the security of a public key cryptographic algorithm is based on a conjectured difficult problem, such as integer factorization [2], discrete logarithm [3] or the elliptic curve discrete logarithm (ECDL) [4]. Elliptic Curve Cryptography (ECC) is based on the ECDL problem defined on a mathematical structure called elliptic curve, a set of points satisfying an equation which is defined over a finite field [5,6]. The ECC cryptographic algorithms for confidentiality, integrity, and authentication services require arithmetic operations on the elliptic curve such as scalar multiplication, implemented as several additions of points in the elliptic curve. A point addition operation in ECC is implemented using several finite field arithmetic operations, like addition, inversion, division, and multiplication. It has been shown that efficient implementations of ECC are achieved by using projective coordinates [7] to represent the points of the elliptic curve. Under this representation, the point addition operation is implemented using only field additions, subtractions and multiplications. While field additions and subtractions are considered fast operations, multiplications are significantly

☆ Reviews processed and proposed for publication to Editor-in-Chief by Associate Editor Dr. Aly El-Osery.

* Corresponding author. Tel.: +52 222 2663100x8225; fax: +52 222 2663152.
E-mail addresses: mmoraless@upv.edu.mx (M. Morales-Sandoval), cferegrino@ccc.inaoep.mx (C. Feregrino-Uribe), pkitsos@eap.gr (P. Kitsos), rcumplido@inaoep.mx (R. Cumplido).

0045-7906/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compeleceng.2012.08.010

more time demanding, becoming the bottleneck of cryptographic algorithms like ECDSA [8]. This is the reason why efficient implementation of field multiplication has been one of the main topics studied in recent times. Several algorithms for field multiplication have been proposed [9], one of the most attractive has been the Montgomery algorithm [10]. Several implementations of this algorithm have been reported in the literature, mainly hardware architectures for FPGAs. The Montgomery multiplication algorithm performs several iterations to achieve a field multiplication in a finite field, such as $\mathbb{GF}(2^m)$. The bit-serial version of this algorithm processes one bit from one of the involved operands at each iteration and delivers the multiplication after m iterations. The digit-serial version reduces the latency of field multiplication from m to $\lceil m/D \rceil$ iterations by processing a group of D bits (digit) at each iteration. However, this last kind of multiplier requires more area resources as D grows, increasing the delay in the critical path. Bit-parallel multipliers are built by taking $D = m$, performing a field multiplication in only one iteration. Bit-serial multipliers exhibit the highest latency compared to digit-serial and bit parallel multipliers but bit serial multipliers use less area resources and can achieve higher clock frequencies. In most cases, application requirements determine which multiplier configuration is better to use, ranging from a pure bit-serial implementation to a fully parallel one. The digit-serial approach could be a better choice for getting a better performer multiplier compromising area and speed as the application demands.

In order to find this better multiplier configuration, the area–performance of Montgomery multiplication can be evaluated by implementing digit-serial multipliers for different digits while analyzing how the area-time (AT) metric is affected. FPGAs are very attractive for this study as they join the flexibility of software and the performance of hardware. The design flow is achieved by using CAD tools and several versions of the circuit can be tested on the same hardware resources, reducing costs and increasing productivity. This capability of FPGAs allows the exploration of different versions of the digit-serial multiplier in order to select the most appropriate according to the application requirements in terms of area resources or performance.

In this work we present an area/performance trade-off analysis of a digit-serial Montgomery Multiplier based on a Linear Feedback Shift Register [11] well suited for use in ECC cryptographic algorithms. The multiplier is defined over the finite field $\mathbb{GF}(2^m)$ using polynomial basis. We have studied this multiplier using different digits and different finite fields currently recommended in standards of ECC by organizations like IEEE [8], NIST [12] and SEC [13]. A related work to the one presented in this paper has been published in [14], that includes an FPGA area–performance trade offs analysis of a $\mathbb{GF}(2^m)$ "classic" multiplier. Unlike this paper, we are working with $\mathbb{GF}(2^m)$ multiplication in the Montgomery domain. The algorithm for Montgomery multiplication is quite different to the one studied in [14], and hence, the architectural design and corresponding results achieved here cannot be directly compared. In [11], the complexity of the digit-serial Montgomery multiplier was analyzed theoretically and expressed in terms of the digit D and field size m, and the complexity of hardware designs for bit serial and digit-serial $\mathbb{GF}(2^m)$ multipliers is presented in terms of ANDs, XORs, latency and critical path delay. On the contrary, the contributions presented in this work are:

(i) A $\mathbb{GF}(2^m)$ Montgomery multiplier implemented in FPGA, for which area/performance trade-off is studied.
(ii) An area/performance trade off study of $\mathbb{GF}(2^m)$ Montgomery multiplier for elliptic curve cryptography. Different configuration for the digit-serial multiplier were considered, using the finite fields $m = 193$, 233, 239, 277, 409 and 571 and the digits $D = 2$, 4, 8, 16, 32, and 64.
(iii) A comparison against previous FPGA implementations of Montgomery multipliers, in order to demonstrate the advantage of using LFSR in the construction of the multiplier for practical applications.
(iv) An evaluation of the $\mathbb{GF}(2^m)$ digit-serial Montgomery multiplier in practical FPGA implementation using several metrics, such as throughput, efficiency and AT metric.

To our knowledge, this is the first work that provides an area/performance trade-off analysis for digit-serial Montgomery multiplier over $\mathbb{GF}(2^m)$. As an application example, consider the design of a security protocol on chip. Depending on the available area for the whole protocol or the performance it must meet, the study presented in this work allows the designer to compare performance metrics of the multiplier such as area, memory, latency, clocking frequency and throughput, in order to select the most suitable digit that meets the application requirements.

The rest of this paper is organized as follows: next section overviews the digit-serial Montgomery Multiplier and its architecture. The results, analysis, and comparison are presented and discussed in Section 3. Finally, the conclusions are pointed out in Section 4.

2. Digit-serial Montgomery multiplication and hardware architecture

The Montgomery multiplier considered in this work is for arbitrary finite fields of the form $\mathbb{GF}(2^m)$ defined by an arbitrary irreducible polynomial $f(x)$, where the main component in the architecture is a Linear Feedback Shift Register (LFSR) implementing the multiplication of a polynomial $A(x)$ by $x^{-1} \pmod{f(x)}$. The objective of using an LFSR was to reduce the time complexity of the Montgomery multiplier, reducing both the latency and the critical path delay to increase the multiplier throughput.

2.1. $\mathbb{GF}(2^m)$ Montgomery multiplication

In polynomial basis, each element $e \in \mathbb{GF}(2^m)$ corresponds to a binary polynomial $e(x)$ of degree less than m defined as $e(x) = e_{m-1}x^{m-1} + \cdots + e_1x + e_0$ with $e_i \in \{1,0\}$. Usually, e is denoted by the bit-vector $(e_{m-1}, e_{m-2}, \ldots, e_1, e_0)$ of length m [15,16].

The $\mathbb{GF}(2^m)$ Montgomery multiplication [10] between $A(x)$ and $B(x)$ is defined as $A(x) \times B(x) \times R^{-1}(x) \bmod f(x)$, where $f(x)$ is an irreducible polynomial that generates the field $\mathbb{GF}(2^m)$ and $R(x)$ is a fixed field element in $\mathbb{GF}(2^m)$. The field element $R^{-1}(x)$ denotes the multiplicative inverse of the element $R(x) \in \mathbb{GF}(2^m)$; in this work, $R(x) = x^m$. Using this polynomial, most of the modulo reduction operations required in the algorithm consist on ignoring terms which have powers of x larger than or equal to m. Also, the division of an arbitrary polynomial by x^m is accomplished by shifting the polynomial to the right by m places, which are faster operations in hardware.

2.2. Digit-serial $\mathbb{GF}(2^m)$ Montgomery multiplier

The main idea in a digit-serial multiplier (word level) is to process a group of D bits at a time from $A(x)$ instead of one bit at each clock cycle. The word level description of the polynomial $A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0$ implies a partition of $A(x)$ into blocks of equal length. Let D be the size of these blocks such that $A(x)$ has s blocks, $s = \lceil m/D \rceil$. Thus, $A(x) = A_0(x) = A_0(x) + A_1(x) + \cdots + A_{s-2}(x) + A_{s-1}(x)$, where each $A_i(x)$ is of length D and defined as in Eq. (1).

$$A_i(x) = a_{m-iD-1}x^{m-iD-1} + a_{m-iD-2}x^{m-iD-2} + \cdots + a_{m-iD-D-1}x^{m-iD-D-1} + a_{m-iD-D}x^{m-iD-D} \tag{1}$$

Using the word level representation of $A(x)$, we can express $C(x) = A(x) \times B(x) \times R^{-1}(x) \bmod f(x)$ as

$$C(x) = \sum_{i=0}^{s-1} A_i(x) \times B(x) \times R^{-1}(x) \bmod f(x) \tag{2}$$

Let $C_i(x) = A_i(x)B(x)R^{-1}(x) \bmod f(x)$ and $R(x) = x^m$. Eq. (3) defines $C_i(x)$:

$$\begin{aligned} C_i(x) &= \frac{a_{m-iD-1}x^{m-iD-1}B(x)}{x^m} + \frac{a_{m-iD-2}x^{m-iD-2}B(x)}{x^m} + \cdots + \frac{a_{m-iD-D}x^{m-iD-D}B(x)}{x^m} \\ &= a_{m-iD-1}\frac{B(x)}{x^{iD+1}} + a_{m-iD-2}\frac{B(x)}{x^{iD+2}} + \cdots + a_{m-iD-D}\frac{B(x)}{x^{iD+D}} \\ &= a_{m-iD-1}B(x)x^{-(iD+1)} + a_{m-iD-2}B(x)x^{-(iD+2)} + \cdots + a_{m-iD-D}B(x)x^{-(iD+D)} \end{aligned} \tag{3}$$

According to the last expression of $A_i(x) \times B(x) \times x^{-m} \bmod f(x)$ in Eq. (3), $j (1 \leqslant j \leqslant D)$ consecutive outputs of the LFSR are processed instead of a single one. Each output $B(x)x^{-(iD+j)}$ is multiplied by the bit a_{m-iD-j} from $A_i(x)$. As in a bit-serial implementation, this multiplication is implemented by ANDing each bit value of polynomial $B(x)x^{-(iD+j)}$ with the bit a_{m-iD-j}. At each clock cycle $i (0 \leqslant i \leqslant s-1)) j$ multiplications $a_{m-iD-j}B(x)x^{-(iD+j)}$ are performed in parallel and added all together to get $C_i(x) = A_i(x)B(x)x^{-m} \bmod f(x)$.

Generalizing the idea of an LFSR computing $A(x) \times x^{-1} \bmod f(x)$ we built a parallel LFSR (PLFSR) that computes $A(x) \times x^{-D} \bmod f(x)$ in a single iteration. This was achieved by replicating the combinatorial logic (CL-LFSR block in Fig. 1b) for computing $A(x) \times x^{-1} \bmod f(x)$ in a cascade way, as depicted in Fig. 1a).

The CL-LFSR block computes $B(x)x^{-(i+1)}$ from $B(x)x^{-i}$. The word $A_i(x) = (a_{m-iD-1}, a_{m-iD-2}, \ldots, a_{m-iD-D})$ could be obtained by a D-bit shift register. Each bit a_{m-iD-j} is multiplied by the corresponding polynomial $B(x)x^{-(i+j)}$, $(1 \leqslant j \leqslant D)$ and all these partial multiplications are added to get $C_i(x)$. After $s = \lceil m/D \rceil$ clock cycles the whole multiplication $A(x) \times B(x) \times R^{-1}(x) \bmod f(x)$ is finally computed.

The factor $R(x) = x^{m-1}$ could be used by taking the D outputs of PLFSR before the first CL-LFSR block. Our digit-serial multiplier could be configured for any arbitrary irreducible polynomial $f(x)$ as it is shown in Fig. 1b). Since the finite fields used in ECC are typically defined on trinomials or pentanomials, the area usage is lower because several AND gates are not required in the CL-LFSR blocks. The complexity of the digit-serial multiplier shown in Fig. 1 is presented in Table 1. Three configurations are considered for the linear feedback register, using general irreducible polynomials, trinomials and pentanomials. The delay is mainly determined by the depth of a binary tree of D XOR gates used in the multiplication phase of a digit in $A_i(x)$ by $B(x)x^{-i}$.

3. Results

The $\mathbb{GF}(2^m)$ digit-serial Montgomery multiplier was coded and verified using the VHDL hardware description language and Xilinxs tools. The design has been synthesised using Xilinxs ISE 13.2 design tools targeting a Xilinx Virtex6 FPGA. A single slice in this device contains four LUTs and eight flip flops.

The multiplier architecture was tested using digits 2, 4, 8, 16, 32 and 64 for recommended ECC key-lengths. Table 2 shows the finite field used and the corresponding irreducible polynomials. These parameters are widely used and recommended in several standards of elliptic curve cryptography, as in [8,12,13,17].

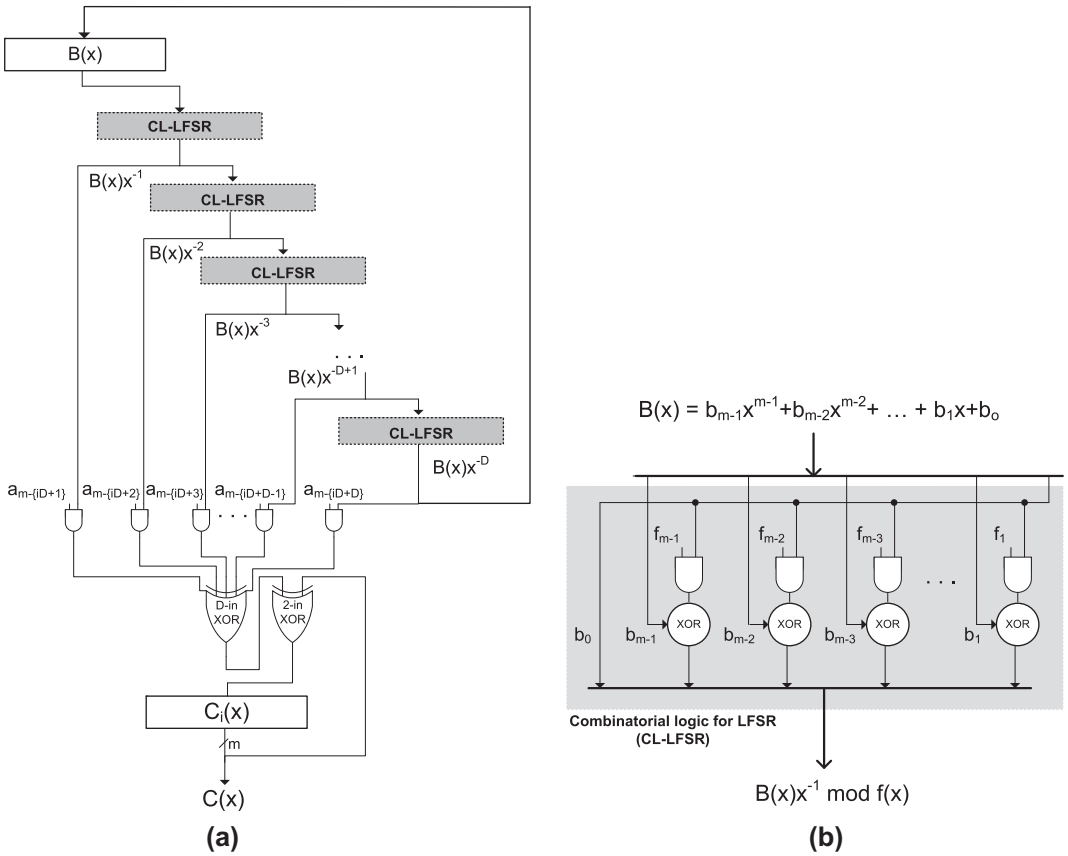

Fig. 1. (a) Digit-serial Montgomery multiplier using a parallel LFSR. (b) Combinatorial logic block for computing $A(x) \times x^{-1} \bmod f(x)$.

Table 1
Area and time complexity of digit-serial $\mathbb{GF}(2^m)$ Montgomery multiplier.

Multiplier complexity	$f(x)$		
	General	Trinomials	Pentanomials
1-bit FF	$2m$	$2m$	$2m$
2-in AND	$D(2m-1)$	Dm	$D(m+2)$
2-in XOR	$D(2m-1)$	$D(m+1)$	$D(m+3)$
Latency	$\lceil m/D \rceil$	$\lceil m/D \rceil$	$\lceil m/D \rceil$
Delay	$(D+1)T_A + [D+P]T_X$	$T_A + [1+P]T_X$	$T_A + [1+P]T_X$

T_X: Time delay of an XOR gateT_A: Time delay of an AND gate$P$: Depth of a binary tree of D XOR gates = $log_2(D)$.

Table 2
Finite fields and irreducible polynomials used for implementation.

Finite field	$f(x)$	Recommended by
$\mathbb{GF}(2^{193})$	$x^{193} + x^{15} + 1$	SEG group [13]
$\mathbb{GF}(2^{233})$	$x^{233} + x^{74} + 1$	IEEE [8], NIST [12]
$\mathbb{GF}(2^{239})$	$x^{239} + x^{36} + 1$	SEG group [13], NIST [12]
$\mathbb{GF}(2^{277})$	$x^{277} + x^{12} + x^6 + x^3 + 1$	IPSec [17]
$\mathbb{GF}(2^{409})$	$x^{409} + x^{87} + 1$	IEEE [8], NIST [12]
$\mathbb{GF}(2^{571})$	$x^{571} + x^{10} + x^5 + x^2 + 1$	IEEE [8], NIST [12]

The area and time implementation results are presented in Table 3. This table shows how area resources increase as the digit size gets bigger. While the sequential logic remains the same for different digits, the cost in the FPGA comes in terms of LUTs for combinatorial logic for implementing the replication of CL-LFSR blocks. This increase in combinatorial logic affects

Table 3
Area/performance results of digit-serial $\mathbb{GF}(2^m)$ Montgomery multiplier for different digits and finite fields in Virtex6 FPGA.

m = 193						
Digit	2	4	8	16	32	64
FlipFlops	590	588	588	590	533	512
LUTs	590	785	1173	1954	2494	3395
Slices	148	196	293	489	624	849
Latency	97	48	24	12	6	3
Freq. (MHz)	777.363	647.627	583.805	557.693	466.408	450.348
m = 233						
Digit	2	4	8	16	32	64
FlipFlops	710	708	708	710	635	629
LUTs	710	945	1413	2352	2995	4123
Slices	178	236	353	588	749	1031
Latency	117	58	29	15	7	4
Freq. (MHz)	777.363	641.807	583.805	539.055	482.346	450.659
m = 239						
Digit	2	4	8	16	32	64
FlipFlops	728	726	726	728	658	643
LUTs	728	969	1449	2412	3069	4233
Slices	182	242	362	603	767	1058
Latency	120	60	30	15	7	4
Freq. (MHz)	777.363	641.807	583.805	539.055	480.491	450.628
m = 277						
Digit	2	4	8	16	32	64
FlipFlops	841	840	840	844	762	849
LUTs	846	1130	1693	2847	3676	5293
Slices	212	283	423	712	919	1323
Latency	139	69	35	17	9	4
Freq. (MHz)	777.363	647.333	558.424	445.752	370.636	390.778
m = 409						
Digit	2	4	8	16	32	64
FlipFlops	1238	1236	1236	1238	1112	1067
LUTs	1238	1649	2469	4112	5094	6986
Slices	310	412	617	1028	1274	1747
Latency	205	102	51	26	13	6
Freq. (MHz)	777.363	641.807	583.805	539.055	489.285	450.659
m = 571						
Digit	2	4	8	16	32	64
FlipFlops	1723	1722	1723	1727	1546	1727
LUTs	1728	2304	3957	5075	6428	10983
Slices	432	576	989	1269	1607	2746
Latency	286	143	71	36	18	9
Freq. (MHz)	777.363	638.121	577.35	379.313	254.321	258.558

the maximum delay which is reflected in a decreasing of the clock frequency. However, even for the greatest digit ($D = 64$) the reached frequency is at least above 250 MHz, which is higher than the best frequency previously reported in the literature. The best clock frequency achieved by the multiplier is around 770 MHz using the minimum finite field $m = 193$, an acceptable security key-length for ECC cryptography comparable to the existing architectures.

Fig. 2 shows area usage for the digit-serial Montgomery multiplier in terms of slices used. Graphically, we can observe the regular structure of the digit-serial multiplier, exhibiting a linear behavior with respect the digit and finite field used. This structure is essential to achieve high clock frequencies and high throughputs.

In all cases, the ratio between the area required by the multiplier using the maximum digit (64) and the minimum (2) is about 5.8, while the improvement in the timing for computing a Montgomery multiplication using the greatest digit respect to the minimum is around 30×.

In order to find a relation between the area and performance achieved by the multiplier, the Area-Time (AT) metric is used to characterize the FPGA-based implementations, which is shown in Fig. 3 for evaluating the implemented multipliers. In this work, AT is defined as the number of slices used (area) multiplied by the time to compute a single field multiplication, in μs. The computing time of $\mathbb{GF}(2^m)$ Montgomery multiplications is determined by the latency and period of the clock cycle. In all cases, latency is $\lceil m/D \rceil$ and the period is obtained as $1/F$, being F the operational frequency of the circuit. The behavior of the AT metric in Fig. 3 demonstrates that the increase on area resources in the Montgomery multiplier is justified, as the curves tend to decrease rapidly as the digit size increases. The AT value is the smallest when the digit is the biggest. Regardless the frequency decreases as the digit increases, the reduction in the latency allows smaller AT values.

The results shown in Table 3, Figs. 2 and 3, have the purpose of serving as a guide to select the most suitable configuration for the $\mathbb{GF}(2^m)$ Montgomery multiplier. For example, a multiplier with small digit size is well suited for area constrained implementations while the one with large digit size is better suited for applications with high performance requirements.

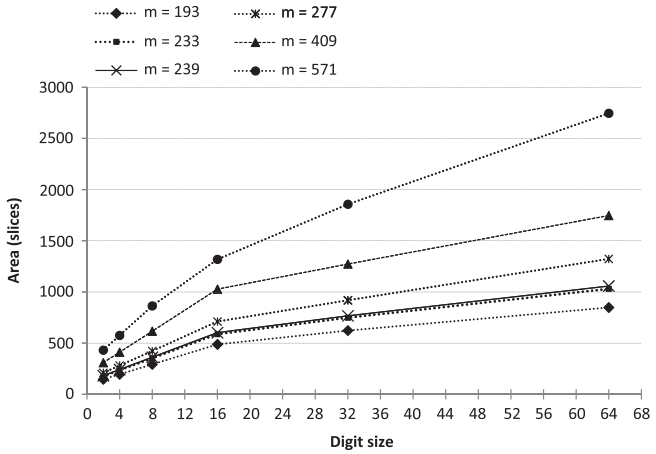

Fig. 2. Area results (slices) of LFSR-based $\mathbb{GF}(2^m)$ digit-serial Montgomery multiplier.

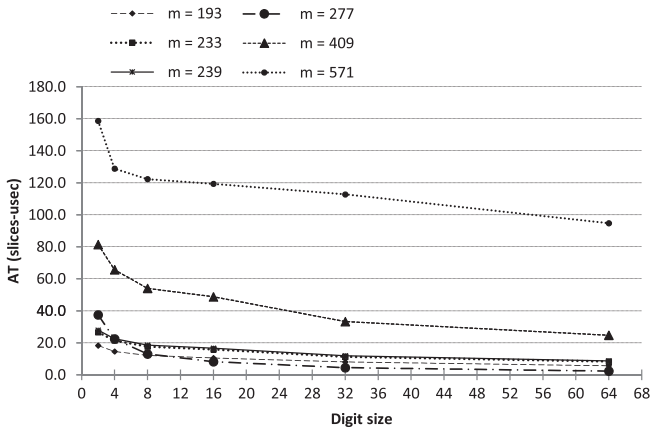

Fig. 3. AT results (slices-μs) of $\mathbb{GF}(2^m)$ LFSR-based digit-serial Montgomery multiplier.

Table 4
Comparison results.

Ref.	Precision	Time (ms)	Device	FPGA slices	Throughput (Mbps)	Efficiency (Mbps/slices)
[20]	512 (prime field)	–	Virtex II	5782	121.55	0.021
[21]	256 (prime field)	0.38	VirtexII	LUTs = 4997, REGs = 4051	–	–
[22]	192 (prime field)	–	VirtexII	2347	–	–
[22]	256 (prime field)	–	VirtexII	3109	–	–
This work	409 (binary field), $D = 8$	0.00019	VirtexII	1800	2146.18	1.192
[23]	512 (prime field)	–	Virtex2Pro	5671	148.32	0.026
[24]	256 (prime field)	–	Virtex2Pro	4663	209.66	0.045
This work	409 (binary field), $D = 8$	0.00015	Virtex2Pro	1792	2645.18	1.476
[25]	512 (prime field)	–	Virtex-E	6293	85.06	0.014
[18]	256 (prime field)	0.9	Virtex-E	LUTs = 1407, REGs= -	–	–
[18]	163 (binary field)	0.9	Virtex-E	LUTs = 1417, REGs= -	–	–
This work	409 (binary field), $D = 8$	0.00038	Virtex-E	1800	1058.33	0.588
This work	233 (binary field), $D = 2$	0.00016	VirtexII	523	2529.96	4.837
This work	233 (binary field), $D = 2$	0.00013	Virtex2Pro	523	3027.50	5.789
This work	233 (binary field), $D = 2$	0.00030	VirtexE	536	1361.53	2.540

Few works [18,19] have considered to study digit-serial $\mathbb{GF}(2^m)$ Montgomery multiplier trade-off so it is hard to provide a fair comparison. Most of the related works of Montgomery multipliers are for the field $\mathbb{GF}(p)$ (see Table 4) and the precision used in these fields does not correspond exactly to the precision used in $\mathbb{GF}(2^m)$ fields used for elliptic curve cryptography. For these fields, m is not an exact integer multiple of the word size as in the case of $\mathbb{GF}(p)$. In the case of $\mathbb{GF}(p)$, currently

accepted security levels for ECC are for a key length around 256 bits. On the contrary, for $\mathbb{GF}(2^m)$, key length in the range 233–409 bits is considered secure. Regardless of this fact, in Table 4 we provide a comparison of our results against representative works in the literature. Since related works have used FPGAs no longer supported by current synthesis tools, the HDL designs were implemented using Xilinx's ISE 8.2 for the same FPGAs used in related works. According to Table 4, our proposed digit-serial multiplier exhibits better performance than previous approaches for constructing Montgomery multipliers. For $m = 233$ and $D = 2$, the throughput of the proposed multiplier on the Virtex2Pro is up to 3 Gbps and 29 Gbps on the Virtex6. In all cases the $\mathbb{GF}(2^m)$ Montgomery multiplier discussed in this work obtained the best efficiency, expressed as the ratio Mbps/Slices. In [18] authors have proposed a Montgomery multiplier implemented on FPGA for the binary field $\mathbb{GF}(2^{163})$. Their design is based on modified four-to-two carry-save adders (CSAs). However, their implementation uses more area resources than the multiplier reported in this work even though they used a minor finite field order.

In [19], the digit serial systolic architecture reported for $\mathbb{GF}(2^m)$ Montgomery multiplication exhibits a latency of $2(\lceil m/D \rceil) - 1$, twice the latency of our proposed multiplier. The digits used were $D = 16$ and $D = 32$ with $m = 160$. This field is being no longer supported in current security standards. The clock cycle reported in [19] for this finite field is 20.53 ns and 16.47 ns using a digit size of 32 and 16 respectively. This leads to a clock frequency of 48.7MHz and 60.71MHz, that are lower than the frequencies achieved by our designs even using greater finite fields. The lower frequency and greater latency of the multiplier cause that the throughput achieved in [19] to be lower than the one achieved by the multiplier in this work. The multiplier presented in this work exhibits the highest throughput and efficiency among previously reported FPGA implementations of Montgomery algorithm for field multiplication. The main reason for this is the use of the LFSR as the core component in the multiplier architecture. LFSRs have a very simple and regular structure which minimizes the required hardware resources resulting in shorter critical paths thus increasing the operating clock frequency. As a result, the hardware architecture of the field multiplier is more compact and exhibits better performance when it is implemented in the FPGA.

4. Concluding remarks

This work discussed the area/performance trade-offs of digit-serial hardware architectures for $\mathbb{GF}(2^m)$ Montgomery multipliers implemented in FPGA technology. The main functional core is a Linear Feedback Shift Register (LFSR), which provides to the multiplier architecture a regular structure that allows it to achieve high clock frequencies at the cost of few area resources, obtaining an efficiency and throughput do not previously reported in the literature. The applications of this multiplier are for public key cryptography, particularly Elliptic Curve Cryptography (ECC). Because of this, the finite fields and corresponding irreducible polynomials used for implementation are the ones recommended by NIST, IEEE and SEC. In all cases, the ratio between the area required by the multiplier using the maximum digit (64) and the minimum (2) is about 5.8, while the improvement in the timing for computing a Montgomery multiplication using the greatest digit respect to the minimum is around 30×. The multiplier hardware requirements increase when the digit grows, it is justified because the area time (AT) metric decreases rapidly, obtaining the minimum AT metric for the greater digit. The area/performance trade off study presented in this work provides a useful guide to the designer in order to select the most suitable configuration for the multiplier, $\langle m, D \rangle$, compromising area, throughput and efficiency to meet given implementation requirements.

References

[1] Schneier B. Applied cryptography. NY: John Wiley & Sons; 1996.
[2] Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 1978;21(2):120–6.
[3] Diffie W, Hellman M. New directions in cryptography. IEEE Tran Inform Theory IT-22 1976(2/3):644–54.
[4] Hankerson D, Menezes AJ, Vanstone S. Guide to elliptic curve cryptography. Secaucus, NJ, USA: Springer; 2003.
[5] Koblitz N. Elliptic curve cryptosystems. Math Comput 1987;48(177):203–9.
[6] V. Miller, Use of elliptic curves in cryptography, in: Proc. of advances in cryptology, CRYPTO'85, Santa Barbara, CA, 1985, pp. 417–426.
[7] López J, Dahab R. Improved algorithms for elliptic curve arithmetic in GF (2^n). In: Proc. of selected areas in cryptography. Lecture notes in computer science, vol. 1556. Springer; 1998. p. 201–12.
[8] IEEE P1363 committee, Standards specification for public key cryptography, <http://grouper.ieee.org/groups/1363/> [accessed 27.08.12].
[9] Hankerson D, López L, Menezes A. Software implementation of elliptic curve cryptography over binary fields. In: Proc. of the second international workshop on cryptographic hardware and embedded systems CHES'2000. Lecture notes in computer science, vol. 1965. Worcester, MA: Springer; 2000. p. 1–24.
[10] Montgomery PL. Modular multiplication without trial division. Math Comput 1985;44:519–21.
[11] Morales-Sandoval M, Feregrino-Uribe C, Kitsos P. Bit-serial and digit-serial GF(2^m) Montgomery multipliers using linear feedback shift registers. IET Comput Digital Tech 2010;5(2):86–94.
[12] NIST, Recommended elliptic curves for federal government use; 1999. <http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf> [accessed 27.08.12].
[13] Standards for Efficient Cryptography (SEC), Section 1: elliptic curve cryptography; 2000. <http://www.secg.org/collateral/Section1_final.pdf> [accessed 27.08.12].
[14] Morales-Sandoval M, Feregrino-Uribe C, Cumplido R, Algredo-Badillo I. An area/performance trade-off analysis of a GF(2^m) multiplier architecture for elliptic curve cryptography. Comput Electr Eng Elsevier 2009;35(1):54–8. <http://dx.doi.org/10.1016/j.compeleceng.2008.05.008>.
[15] Menezes A, Blake I, Gao X, Mullin R, Vanstone S, Yaghoobian T. Applications of finite fields. Springer International Series in Engineering and Computer Science; 1993.
[16] Ryan WE, Lin S. Channel codes. Cambridge, NY: Classical and Modern; 2009.
[17] Panjwani P, Poeluev Y. Additional ECC groups for IKE, 1999. IPSecWorking group, INTERNET-DRAFT; 1999.
[18] Sudhakar M, Kamala RV, Srinivas MB. New and improved architectures for Montgomery modular multiplication. Mob Netw Appl 2007;12(4):281–91. <doi:http://dx.doi.org/10.1007/s11036-007-0019-z>.

[19] Talapatra S, Rahaman H, Mathew J. Low complexity digit serial systolic montgomery multipliers for special class of GF(2^m). IEEE Trans Very Large Scale Integr (VLSI) Syst 2010;18(5):847–52. http://dx.doi.org/10.1109/TVLSI.2009.2016753.
[20] Sudhakar M, Kamala R, Srinivas M. An efficient, reconfigurable and unified Montgomery multiplier architecture. In: IEEE/ACM international conference on VLSI design. ACM; 2004. p. 750–5.
[21] Pinckney NR, Harris DM. Parallelized radix-4 scalable Montgomery multipliers. In: SBCCI '07: proceedings of the 20th annual conference on integrated circuits and systems design. New York, NY, USA: ACM; 2007. p. 306–11. <doi:http://dx.doi.org/10.1145/1284480.1284562>.
[22] Daly A, Marnane W, Kerins T, Popovici E. An FPGA implementation of a GF(p) ALU for encryption processors. Microprocess Microsyst 2004;28:253–60.
[23] Maddi S, Srinivas MB. A unified and reconfigurable Montgomery multiplier architecture without four-to-two CSA. In: SBCCI '07: proceedings of the 20th annual conference on integrated circuits and systems design. New York, NY, USA: ACM; 2007. p. 147–52. <doi:http://dx.doi.org/10.1145/1284480.1284525>.
[24] McIvor C, McLoone M, McCanny JV. FPGA Montgomery multiplier architectures – a comparison. In: FCCM '04: proceedings of the 12th annual IEEE symposium on field-programmable custom computing machines. Washington, DC, USA: IEEE Computer Society; 2004. p. 279–82.
[25] McIvor C, McLoone M, McCanny J. Modified Montgomery modular multiplication and RSA exponentiation techniques. IEE Proc Comput Digital Tech 2004;151(6):402–8.

Miguel Morales-Sandoval received the B.Sc. degree in Computer Science from the University of Puebla and the M.Sc. and Ph.D. degree in Computer Science from the National Institute for Astrophysics, Optics and Electronics (INAOE) in 2004 and 2008 respectively. Currently, he is a professor-researcher at the Information Technology department in the Polytechnic University of Victoria, Mexico. His research areas include efficient implementation of cryptographic algorithms, finite field arithmetic, reconfigurable computing and hardware design using HDLs.

Claudia Feregrino-Uribe received the M.Sc. degree from CINVESTAV Guadalajara, Mexico in 1997 and the Ph.D. degree from Loughborough University, UK, in 2001. She is a researcher at Computer Science Department at INAOE, Mexico since 2002. Her research interests include Digital Design with FPGAs, Data Compression, Cryptography and Watermarking. She is co-founder of the ReConFig International Conference, associate editor of the International Journal of Reconfigurable Computing and Microprocessors and Microsystems and reviewer for several international journals.

Paris Kitsos received the B.Sc. degree in Physics in 1999 and a Ph.D. in 2004 from the Department of Electrical and Computer Engineering, at the University of Patras. Currently he is with the Digital Systems & Media Computing Laboratory, Hellenic Open University. His research interests include VLSI design and efficient implementations of cryptographic primitives. He has published more than 70 scientific articles and technical reports, and serves as reviewer in International Journals and Conferences in the areas of his research.

Rene Cumplido holds a B.Sc. in Computer Systems from ITQ, Mexico (1995), a M.Sc. in electrical engineering from CINVESTAV, Mexico (1997), and a PhD in electrical engineering from Loughborough University, UK (2001). In 2002, he joined the Reconfigurable and High Performance Computing Group at INAOE. His research interests are reconfigurable computing and hardware architectures for signal processing and scientific computing. He is co-founder and general chair of the ReConFig conference and serves as associate editor of journals on computer and electrical engineering.