Efficient RNS Implementation of Elliptic Curve Point Multiplication Over GF(p)

Mohammad Esmaeildoust, Dimitrios Schinianakis, Hamid Javashi, Thanos Stouraitis, and Keivan Navi

***Abstract*— Elliptic curve point multiplication (ECPM) is one of the most critical operations in elliptic curve cryptography. In this brief, a new hardware architecture for ECPM over $GF(p)$ is presented, based on the residue number system (RNS). The proposed architecture encompasses RNS bases with various word-lengths in order to efficiently implement RNS Montgomery multiplication. Two architectures with four and six pipeline stages are presented, targeted on area-efficient and fast RNS Montgomery multiplication designs, respectively. The fast version of the proposed ECPM architecture achieves higher speeds and the area-efficient version achieves better area–delay tradeoffs compared to state-of-the-art implementations.**

***Index Terms*— Elliptic curve cryptography (ECC), Montgomery multiplication, residue arithmetic, residue number system (RNS).**

I. Introduction

Elliptic curve cryptography (ECC) was introduced independently by Koblitz [1] and Miller [2] in 1985. ECC is based on finite field arithmetic, where prime fields of the form GF(p) and binary extension fields of the form GF (2^m) are the most prominent choices for ECC implementation [3]. Thus, efficient point multiplication depends on efficient implementation of the underlying field arithmetic. One way to speed up the elliptic curve point multiplication (ECPM) is through the residue number system (RNS) representation [4].

By employing RNS, large dynamic ranges are decomposed into smaller parallel paths, each employing a different modulus. Thus, higher speeds and less power consumption are achieved [4]. Among various ECC implementations [4]–[8], RNS has been recently exploited for ECPM [4], [8]. In [4], reductions modulo the field characteristic p are removed during intermediate calculations and are performed only once during the final step. This leads to an increase of the dynamic range because, for the 192-b key, twenty 42-b moduli are required to cover the data range.

In [8], RNS bases with moduli of the form $2^k - c_i$, with $c_i < 2^q$ and $q < k/2$, were employed. For the 192-b key, six 33-b moduli are required in each base. For larger key sizes, more moduli are required. For the base extensions in [8], the method proposed in [9] is used. As the number of moduli increases, the complexity of the required base extensions in the process of RNS Montgomery multiplication also increases.

Manuscript received February 2, 2012; revised May 24, 2012; accepted July 21, 2012. Date of publication September 4, 2012; date of current version July 22, 2013. This work was supported in part by the Iran Telecommunication Research Center, the European Union (European Social Fund ESF), and Greek national funds through the Education and Lifelong Learning program of the National Strategic Reference Framework's Research Funding Program: Heracleitus II—Investing in Knowledge Society through the European Social Fund.

M. Esmaeildoust and K. Navi are with the Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran 1983963113, Iran (e-mail: m_doust@sbu.ac.ir; navi@sbu.ac.ir).

D. Schinianakis and T. Stouraitis are with the Electrical and Computer Engineering Department, University of Patras, Patras 13231, Greece (e-mail: dsxoinianakis@upatras.gr; thanos@upatras.gr).

H. Javashi is with the Microelectronic Laboratory, Shahid Beheshti University, Tehran 1983963113, Iran (e-mail: mrc-ecef@sbu.ac.ir).

Digital Object Identifier 10.1109/TVLSI.2012.2210916

This brief employs efficient RNS bases for the implementation of RNS Montgomery multiplication. Sets of three and four moduli are used to support ECC for keys of 160, 192, 224, and 256 b. The small number of moduli allows the exploitation of moduli of the form 2^k, $2^k - 1$, and $2^k - 2^{t_i} \pm 1$, and results in efficient arithmetic operations and residue-to-binary and binary-to-residue conversions [10], [11]. Two different designs are presented with a focus on area and speed, respectively. In the first design, in order to achieve an area-efficient implementation, one multiplier is used in each base and the required converters are implemented using an adder-based structure. In the fast design, a multiplier is used for each modulus, while both designs encompass a new pipelined structure.

The rest of this brief is organized as follows. Section II presents an overview of RNS and modular multiplication. ECC arithmetic is reviewed in Section III. The RNS exploitation in ECC arithmetic is presented in Section IV. In Section V, the proposed RNS Montgomery multiplication is analyzed in detail, while in Section VI the proposed ECPM architecture and its comparisons with state-of-the-art implementations are offered. Section VII concludes this brief.

II. Overview of RNS and Modular Multiplication

A. RNS Background

RNS consists of a set of relatively prime integers $(m_1, m_2, \ldots, m_n)$ called the base. The dynamic range of the system is defined by $M = m_1 \times m_2 \times \cdots \times m_n$ and any integer $X \in [0, M-1]$ has a unique RNS representation given by $X = (x_1, x_2, \ldots, x_n)$, where $x_i = (X \bmod m_i) = \langle X \rangle_{m_i}$.

Residue-to-binary conversion can be performed using mixed radix conversion (MRC). By MRC, the number X can be calculated from its residues by

$$X = v_n \prod_{i=1}^{n-1} m_i + \cdots + v_3 m_2 m_1 + v_2 m_1 + v_1 \tag{1}$$

where

$$\begin{aligned} v_1 &= x_1 \\ v_2 &= \left\langle (x_2 - v_1) \left\langle m_1^{-1} \right\rangle_{m_2} \right\rangle_{m_2} \\ &\vdots \\ v_n &= \left\langle \left((x_n - v_1) \left\langle m_1^{-1} \right\rangle_{m_n} - v_2 \right) \left\langle m_2^{-1} \right\rangle_{m_n} \right. \\ &\qquad \left. - \cdots - v_{n-1} \right) \left\langle m_{n-1}^{-1} \right\rangle_{m_n} \Big\rangle_{m_n} \end{aligned}$$

and $\left\langle m_i^{-1} \right\rangle_{m_j}$ is the multiplicative inverse of $m_i \bmod m_j$. $v_i s$ in (1) are the MRC digits of an intermediate system called the mixed radix system (MRS).

B. Modular Multiplication in RNS

RNS Montgomery modular multiplication is briefly presented in this section. Two RNS bases $\mathrm{B}_n = \{m_1, \ldots, m_n\}$ and $\mathrm{B}'_n = \{m'_1, \ldots, m'_n\}$ are considered, with $M = \prod_{i=1}^{n} m_i$ and $M' = \prod_{i=1}^{n} m'_i$, respectively. Assume two integers A and B, with the RNS representation in the first base given by $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$, and in the auxiliary base B'_n, given by $(a'_1, \ldots, a'_n)$ and $(b'_1, \ldots, b'_n)$, respectively. Considering that $p < M < M'$ and $\gcd(p, M) = \gcd(p, M') = \gcd(M, M') = 1$, the RNS Montgomery multiplication computing $A \times B \times M^{-1} \pmod{p}$ is given in Algorithm 1 [10], [12].

Algorithm 1 RNS Montgomery Multiplication

$1: D = A \times B$ ($d_i = |a_i \times b_i|_{m_i}$ in base B_n and $d'_i = |a'_i \times b'_i|_{m'_i}$ in base B'_n)

$2: q_i = \left| d_i \times |p|_{m_i}^{-1} \right|_{m_i}$ in B_n

$3: q_i$ in $B_n \longrightarrow q'_i$ in B'_n Base extension

$4: r' = (d'_i - q'_i \times N'_i) M^{-1}$ in B'_n

$5: r$ in $B_n \leftarrow r'$ in B'_n Base extension

TABLE I
PROPOSED RNS BASES

	Field (b)	First base	Second base
Three-modulus RNS bases (D1)	160	$\{2^{56}-2^{11}-1,$ $2^{56}-2^{16}-1,$ $2^{56}-2^{20}-1\}$	$\{2^{56},$ $2^{56}-1,$ $2^{57}-1\}$
Three-modulus RNS bases (D2)	192	$\{2^{66}-2^{17}-1,$ $2^{66}-2^{18}-1,$ $2^{66}-2^{24}-1\}$	$\{2^{66},$ $2^{66}-1,$ $2^{67}-1\}$
Four-modulus RNS bases (D3)	192	$\{2^{50}-2^{20}-1,$ $2^{50}-2^{22}-1,$ $2^{50}-2^{18}-1,$ $2^{50}-2^{10}-1\}$	$\{2^{50},$ $2^{50}-1,$ $2^{51}-1,$ $2^{49}-1\}$
Four-modulus RNS bases (D4)	224	$\{2^{58}-2^{22}-1,$ $2^{58}-2^{13}-1,$ $2^{58}-2^{10}-1,$ $2^{58}-2^{16}-1\}$	$\{2^{58},$ $2^{58}-1,$ $2^{59}-1,$ $2^{57}-1\}$
Four-modulus RNS bases (D5)	256	$\{2^{66}-2^{22}-1,$ $2^{66}-2^{24}-1,$ $2^{66}-2^{18}-1,$ $2^{66}-2^{17}-1\}$	$\{2^{66},$ $2^{66}-1,$ $2^{67}-1,$ $2^{65}-1\}$

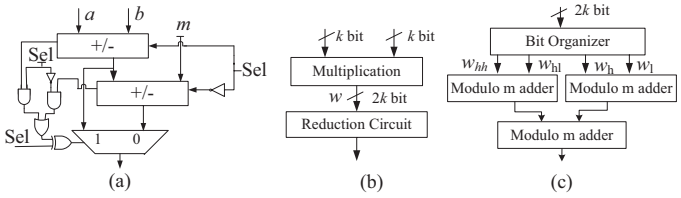

Fig. 1. (a) Modulo m adder/subtractor [4]. (b) Proposed modulo $2^k - 2^{t_i} - 1$ multiplier. (c) Reduction circuit.

III. ECC Arithmetic

An elliptic curve E is defined by

$$y^2 \equiv x^3 + ax + b \pmod{p} \tag{2}$$

where $a, b \in \mathrm{GF}(p)$ and $4a^3 + 27b^2 \neq 0 \pmod{p}$, together with a special point O called the point at infinity. All points (x, y) as defined by (2) together with O are solutions of the elliptic curve E over GF(p). The group law defines the addition of two points on an elliptic curve. Point doubling is a special case of point addition, and refers to adding a point to itself. The required operations for point addition and doubling are detailed in [4]. Point multiplication $[K] \times P$, where K is an integer and P is a point on the curve, is realized via the binary method algorithm, based on the binary expansion of K [3], [4].

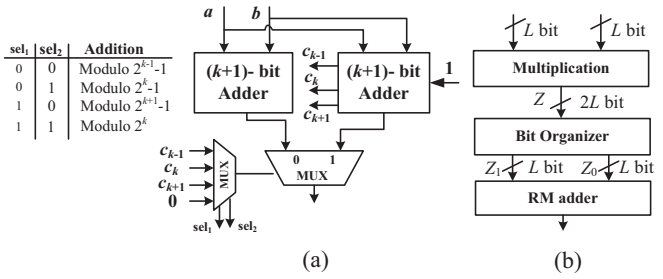

Fig. 2. (a) Proposed RM adder. (b) Proposed RM multiplier, L = k, $k-1$, and $k+1$.

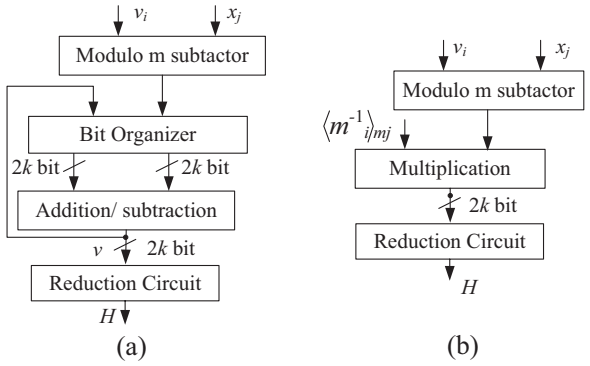

Fig. 3. Calculation of H in RNS-to-MRS conversion for the first base. (a) Area-efficient design. (b) Fast design.

IV. RNS and ECC

Sets of three and four moduli are proposed in order to implement the RNS Montgomery multiplication of Algorithm 1. The form of the moduli determines the efficiency of the RNS arithmetic operations and the structure of the residue-to-binary and binary-to-residue converters [11]. The RNS bases employed are shown in Table I. In the first base, RNS moduli are of the form $2^k - 2^{t_i} - 1$, where $t_i < k/2$ are employed, which offer simple modulo reduction operations [10].

The second base is realized by sets of three and four moduli of the special forms $\{2^k, 2^{k+1}-1, 2^k-1\}$ [13], and $\{2^k, 2^k-1, 2^{k+1}-1, 2^{k-1}-1\}$, which also provide efficient arithmetic operations and residue-to-binary and binary-to-residue conversions [10]. In order to use the result of RNS Montgomery multiplication in subsequent modular multiplications, it is required that $4p < M < M'$ [10]. It is easy to check that the employed bases are sufficient to cover this requirement.

V. Design of Montgomery Modular Multiplication Using Proposed RNS Bases

A. Modular Adders and Multipliers

For the first base, where moduli of the form $2^k - 2^{t_i} - 1$ are used, the modular adder and multiplier depicted in Fig. 1 are employed. Regarding modular multiplication, two k-b operands are multiplied and a $2k$-b value is obtained. Modular reduction of a $2k$-b value w with moduli of the form $2^k - 2^{t_i} - 1$ can be written using its w-b higher, w_h, and lower, w_l, parts, as

$$w = \left\langle w_h 2^k + w_l \right\rangle_{2^k - 2^{t_i} - 1}. \tag{3}$$

Since $2^k \bmod (2^k - 2^{t_i} - 1) = 2^{t_i} + 1$, it holds that

$$w = \left\langle \overbrace{\underbrace{w'_{hh}}_{t_i \text{ bit}} 2^k + \underbrace{w'_{hl}}_{k \text{ bit}}}^{w'} + w_h + w_l \right\rangle_{2^k - 2^{t_i} - 1} \tag{4}$$

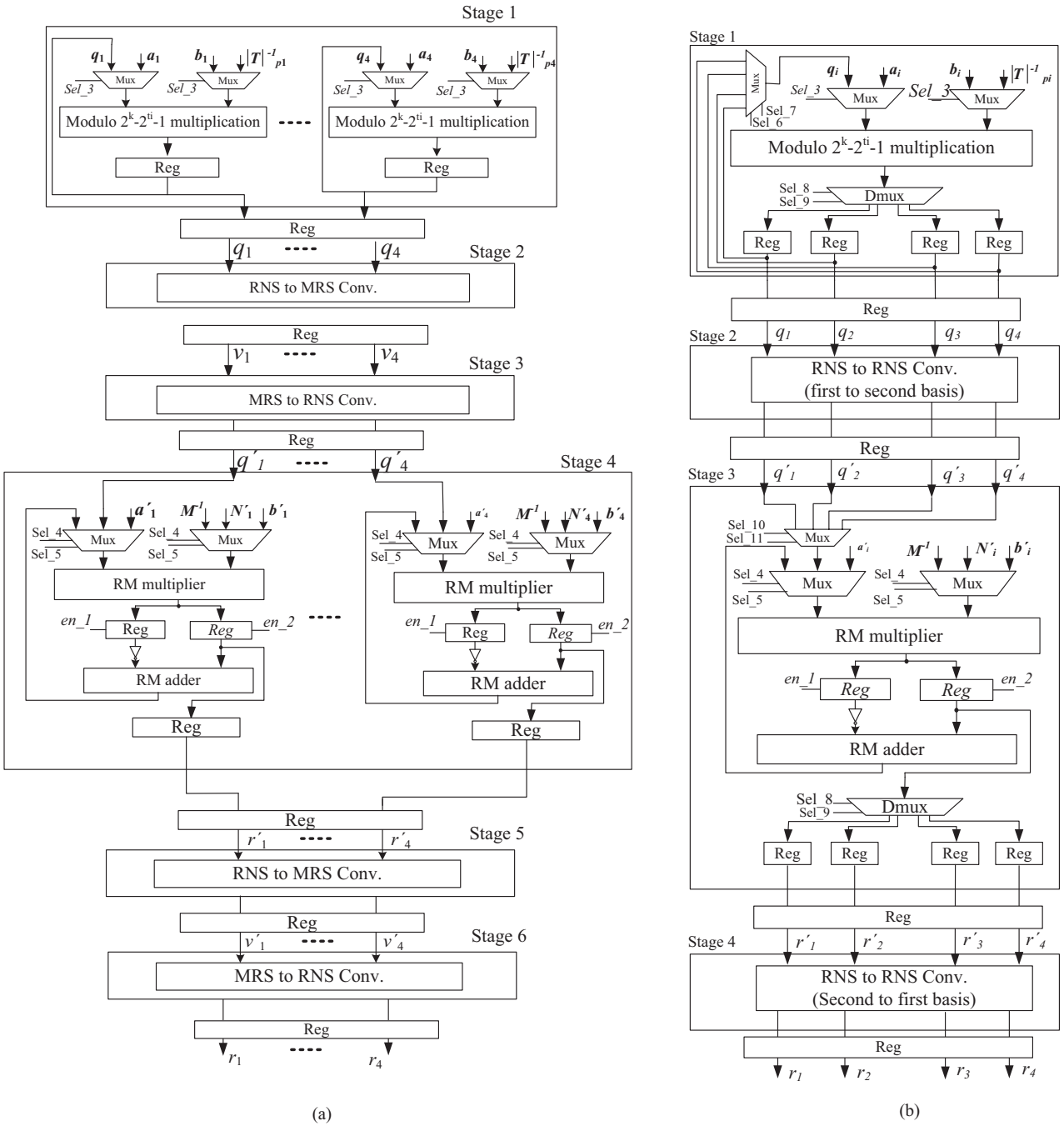

Fig. 4. RNS Montgomery multiplication architecture. (a) Fast design. (b) Area-efficient design.

$$w' = \left\langle \underbrace{w'_{hh}}_{t_i \text{ bit}} \underbrace{0, \ldots, 0}_{t_i \text{ bit}} + \underbrace{w'_{hh}}_{t_i \text{ bit}} + w'_{hl} \right\rangle_{2^k - 2^{t_i} - 1}. \quad (5)$$

Since w'_{hh} has t_i b, it can be concatenated at the end of $w'_{hh} \underbrace{0, \ldots, 0}_{t_i \text{ bit}}$. Therefore w can be calculated by

$$w = \left\langle \underbrace{w'_{hh} w'_{hh}}_{w_{hh}} + w'_{hl} + w_h + w_l \right\rangle_{2^k - 2^{t_i} - 1}. \quad (6)$$

For the second base, a reconfigurable modular (RM) adder is employed, as shown in Fig. 2. Based on the proposed adder, addition and multiplication modulo 2^k, $2^{k-1}-1$, 2^k-1, and $2^{k+1}-1$ can be done without hardware redundancy. Note that the RM adder shown in Fig. 2 has $(k-1)$-b delay of full adder less than the modulo m adder (Fig. 1) in the worst case; thus the second base supports more efficient arithmetic operations. The designed RM multiplier is shown in Fig. 2. After multiplication, the $2L$-b result R is split into two L-b LSB and MSB parts ($L = k$, $k-1$, $k+1$) and reduction modulo $2^L - 1$ can be achieved by a modular addition of R_l and R_h [11].

B. Conversion from First to Second Base

In step 3 of the RNS Montgomery multiplication, base extension from first to second base is required, which consists of a residue-to-MRS conversion in the first base and then an MRS-to-residue conversion in the second base.

Efficient RNS to MRS conversion in first base is reported in [10]. The basic operation in the calculation of v_i ($i = 2$, 3, and 4) in (1) is

$$H = \left\langle (x_j - v_i) \left\langle m_i^{-1} \right\rangle_{m_j} \right\rangle_{m_j}. \tag{7}$$

Hardware implementations of (7) for area- and time-efficient designs are shown in Fig. 3. Considering four-modulus RNS bases, for each v_i ($i = 2$, 3, and 4), the implementation shown in Fig. 3 is employed. The bit organizer provides the required shifts according to precalculated multiplicative inverses.

Residues in second base must be calculated after the calculation of MRNs. In the calculation of MRS to RNS from the first to the second base for four-modulus RNS bases, it holds that

$$x_j = \left\langle v_1 + m_1(v_2 + m_2 \underbrace{(v_3 + m_3 v_4)}_{Z}) \right\rangle_{m_j} \tag{8}$$

where m_j are the moduli 2^k, $2^k - 1$, $2^{k+1} - 1$, and $2^{k-1} - 1$. $m_i = m_1$, m_2, and m_3 are the moduli of the form $2^k - 2^{t_i} - 1$. According to form of selected bases with simple multiplicative inverses, for fast and area-efficient design, adder-based structure can be simply realized by using one RM adder for each modulus.

C. Conversion from Second to First Base

In order to mechanize RNS to MRS conversion in the second base $\{2^k, 2^k - 1, 2^{k+1} - 1\}$, based on (1) and considering $m_1 = 2^k$, $m_2 = 2^k - 1$, and $m_3 = 2^{k+1} - 1$, we get

$$v_1 = x_1 \tag{9}$$

$$v_2 = \left\langle \underbrace{(x_2 - v_1) \left\langle m_1^{-1} \right\rangle_{m_2}}_{Z} \right\rangle_{m_2} \tag{10}$$

$$v_3 = \left\langle \underbrace{((x_3 - v_1) \left\langle m_1^{-1} \right\rangle_{m_3}}_{Z} - v_2) \left\langle m_2^{-1} \right\rangle_{m_3} \right\rangle_{m_3}. \tag{11}$$

The required multiplicative inverses in (10) and (11) are $<m_1^{-1}>_{m_2} = 1$, $<m_1^{-1}>_{m_3} = 2$, and $<m_2^{-1}>_{m_3} = -2$ [13]. Because of the simple form of multiplicative inverses, our adder-based structure was employed for both fast and area-efficient design.

Regarding the MRS to RNS conversion in the first base, it holds that

$$x_j = \langle v_1 + m_1(v_2 + m_2(v_3 + m_3 v_4)) \rangle_{m_j}. \tag{12}$$

Therefore operations in (12) consist of some shifts and additions.

D. Hardware Architecture for RNS Montgomery Multiplication

The proposed architectures for the RNS Montgomery multiplication are shown in Fig. 4. The area-efficient architecture consists of one modulo $(2^k - 2^{t_i} - 1)$ multiplier, one RM multiplier, one RM adder, and two base extension units with adder-based structure, connected in a four-stage pipelined fashion [Fig. 4(b)].

The alternate design optimized for high speed is implemented in a six-stage pipelined architecture, shown in Fig. 4(a). Modular multipliers and adders in Figs. 1 and 2 are employed in each modulus channel in stages 1 and 4 of the pipelined implementation. For the base extension operations, modulo adders and multipliers are used as described in previous subsections.

TABLE II
DELAY AND AREA COMPARISON FOR DIFFERENT ECC ARCHITECTURES

	Field (b)	Max freq. (MHz)	EC mult. time (ms)	Size
[1]Three moduli (area efficient)	192 160	34.7 38.2	2.56 1.83	20 014 LUT 15448 LUT
[1]Four moduli (area efficient)	256 224 192	34.7 37.0 38.4	3.41 2.92 2.67	28318 LUT 25912 LUT 21 380 LUT
[1][4]	256 224 192	39.7 46.8 52.9	3.95 3.31 2.97	32 716 LUT 29610 LUT 25012 LUT
[1][6]	192	40	3	11 416 LUT
[1][7]	160	91.3	14.41	11 227 LUT
[2]Three moduli (area efficient)	192 160	50.2 52.5	1.82 1.36	9310 LUT 8742 LUT
[2]Four moduli (area efficient)	256 192	50.2 53.6	2.62 2.07	18942 LUT 14782 LUT
[2]Three moduli (fast design)	192 160	50.2 52.5	0.35 0.31	19224 LUT 18114 LUT
[2]Four moduli (fast design)	256 192	50.2 53.6	0.59 0.52	28746 LUT 25304 LUT
[2][5]	256	39.46	3.86	35 450 LUT
[2][15]	160	75	2.41	50 000 LUT
[2][14]	256 192 160	94.7 94.7 94.7	2.66 1.25 0.78	41 595 Slices 40 219 Slices 39531 Slices
[3]Three moduli (fast design)	192 160	54.1 56.8	0.33 0.29	5248 ALUT 4892 ALUT
[3]Four moduli (fast design)	256 192 160	54.1 59.9 61.2	0.54 0.42 0.38	12324 ALUT 7932 ALUT 5148 ALUT
[3][8]	256 192 160	157.2 160.5 165.5	0.68 0.44 0.32	9177 ALM 6203 ALM 5896 ALM

[1] Implemented on Xilinx VirtexE.
[2] Implemented on Xilinx Virtex 2 Pro.
[3] Implemented on Altera Startix II.

VI. IMPLEMENTATION DETAILS OF ECPM AND COMPARISONS

Tables II summarizes the delay and area comparisons of the proposed ECPM with recent state-of-the-art works. The field characteristic is $p = 2^{160} + 7$ for a 160-b implementation and NIST recommendations for $p - 192$, $p - 224$, and $p - 256$ corresponding to $2^{192} - 2^{64} - 1$, $2^{224} - 2^{96} + 1$, and $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, respectively [4].

As described in Section III, the binary method is employed to implement the ECPM. For the point addition and doubling operations, the optimized DFGs presented in [4] are employed. The architecture for the proposed ECPM consists of a binary-to-residue converter, a register file, one RNS Montgomery multiplier, a control unit, and a residue-to-binary converter. In the proposed architecture, a binary-to-residue converter is employed to compute the RNS representation of the projective coordinates (X, Y, Z). Because of the use of well-formed moduli, an adder-based structure can be utilized for the binary-to-residue conversion as discussed in [11]. The control unit provides the required input operands and control signals for the arithmetic unit, while the distributed RAMs on field-programmable gate arrays are used in designing the register file. Residue-to-binary conversion can be also realized by an adder-based structure according to (1).

Because of the pipelined implementation of the RNS Montgomery multiplication, reduction in the execution time of a point multiplication is also achieved. Unlike in [4] and [8], fewer moduli

are required, and the use of well-formed RNS bases results in efficient realization of modular addition and multiplication required in the RNS Montgomery multiplication. Another advantage of the proposed architecture is the simple adder-based implementation of the residue- to-binary converter, compared to that in [4] which requires multiplications by large operands. The proposed ECPM architecture is implemented on Xilinx VirtexE, Virtex 2 Pro, and Altera Startix II to achieve a fair one-to-one comparison with the state-of-the-art implementations in [4], [6], [7], [8], and [5]–[15] presented in Table II. In [8], although moduli of smaller word-length are employed, a large number of moduli are used and thus the RNS base extensions are complex resulting in worse performance, as shown in Table II.

VII. Conclusion

In this brief, a new architecture for ECPM was presented, based on efficient RNS bases. Arithmetic-friendly RNS bases in conjunction with a pipelined design for RNS Montgomery multiplication resulted in noticeable improvements in terms of speed and area. Two versions of fast and area-efficient designs for RNS Montgomery multiplication in six- and four-stage pipelined architectures were presented. Comparisons with state-of-the-art implementations proved the efficiency of the proposed architecture.

References

[1] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comp.*, vol. 48, no. 177, pp. 203–209, 1987.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Adv. Cryptology LNCS*, 1986, pp. 47–426.

[3] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[4] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Stouraitis, "An RNS implementation of an F_p elliptic curve point multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 6, pp. 1202–1213, Jun. 2009.

[5] C. J. McIvor, M. McLoone, and J. V. McCanny, "Hardware elliptic curve cryptographic processor over $GF_{(P)}$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 9, pp. 1946–1957, Sep. 2006.

[6] G. Orlando and C. Paar, "A scalable $GF_{(P)}$ elliptic curve processor architecture for programmable hardware," in *Proc. Workshop Cryptograph. Hardware Embed. Syst. LNCS*, 2001, pp. 348–363.

[7] S. B. Ors, L. Batina, B. Preneel, and J. Vandewalle, "Hardware implementation of an elliptic curve processor over $GF_{(P)}$," in *Proc. IEEE Appl.-Specific Syst. Arch. Process.*, Jun. 2003, pp. 433–443.

[8] N. Guillermin, "A high speed coprocessor for elliptic curve scalar multiplications over F_p," in *Proc. CHES 12th Int. Conf. Cryptograph. Hardware Embed. Syst.*, 2010, pp. 48–64.

[9] S. Kawamura, M. Koike, F. Sano, and A. Shimbo, *Cox Rower Architecture for Fast Parallel Montgomery Multiplication*. New York: Springer-Verlag, 2000, pp. 523–538.

[10] J. C. Bajard, M. Kaihara, and T. Plantard, "Selected RNS bases for modular multiplication," in *Proc. IEEE 19th Int. Symp. Comput. Arith.*, Jun. 2009, pp. 25–32.

[11] K. Navi, A. S. Molahosseini, and M. Esmaeildoust, "How to teach residue number system to computer scientists and engineers?" *IEEE Trans. Edu.*, vol. 54, no. 1, pp. 156–163, Feb. 2011.

[12] J. Bajard, L. Didier, and P. Kornerup, "An RNS Montgomery's modular multiplication algorithm," *IEEE Trans. Comput.*, vol. 47, no. 2, pp. 167–178, Feb. 1998.

[13] P. V. A. Mohan, "RNS-to-binary converter for a new three-moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$," *IEEE Trans. Circuits Syst., II*, vol. 54, no. 9, pp. 775–779, Sep. 2007.

[14] J. Yuan and C.-T. Huang, "Elixir: High-throughput cost-effective dual-field processors and the design framework for elliptic curve cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 11, pp. 1567–80, Nov. 2008.

[15] D. M. Schinianakis, A. P. Fournaris, A. P. Kakarountas, and T. Stouraitis, "An RNS architecture of an *F(p)* elliptic curve point multiplier," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 3369–3373.

Low-Power Correlation for IEEE 802.16 OFDM Synchronization on FPGA

Thinh H. Pham, Suhaib A. Fahmy, and Ian Vince McLoughlin

***Abstract*— This brief compares the use of multiplierless and DSP slice-based cross-correlation for IEEE 802.16d orthogonal frequency division multiplexing (OFDM) timing synchronization on Xilinx Virtex-6 and Spartan-6 field programmable gate arrays (FPGAs). The natural approach, given the availability of embedded DSP blocks on these FPGAs, would be to implement standard multiplier-based cross-correlation. However, this can consume a significant number of DSP blocks, which may not fit on low-power devices. Hence, we compare a DSP48E1 slice-based design to four different quantizations of multiplierless correlation in terms of resource utilization and power consumption. OFDM timing synchronization accuracy is evaluated for each system at different signal-to-noise ratios. Results show that even relatively coarse multiplierless coefficient quantization can yield accurate timing synchronization, and does so at high clock speeds. Multiplierless designs enjoy reduced power consumption over the DSP48E1 Slice-based design, and can be used where DSP Slice resources are insufficient, such as on low-power FPGA devices.**

***Index Terms*— Correlation, cognitive radio, field-programmable gate arrays (FPGA), IEEE 802.16 standards, orthogonal frequency division multiplexing (OFDM).**

I. Introduction

Orthogonal frequency division multiplexing (OFDM) is an effective modulation technique used in both wired and wireless communication systems. Particularly, thanks to the advantages of spectral efficiency and robustness to multipath fading, OFDM was specified for multiple applications in high bit-rate wireless transmission systems such as wireless local area networks adopted by IEEE 802.11 and metropolitan area networks in IEEE 802.16d. However, OFDM performance is sensitive to receiver synchronization. Frequency offset causes inter-subcarrier interference, and errors in timing synchronization can lead to inter-symbol interference [1]. Therefore, synchronization is critical for good performance in OFDM systems.

Much research has focused on improving OFDM synchronization performance and accuracy. Cyclic prefix (CP)-based methods were introduced [2]–[4] to determine frequency offset and symbol timing, but do not themselves find the start of a frame. To assist this, all OFDM frames begin with preamble symbols which can also be used to estimate the frequency offset [5]. This relies on the characteristic of a preamble symbol with two identical halves, using autocorrelation of the received signal, which can be computed iteratively at low cost and is robust to frequency offset. However, the metric used results in a plateau which leads to some uncertainty in determining the start of a frame. Work in [6]–[9] introduced modified timing metrics based on autocorrelation and the characteristic of specific preamble symbols to reduce the ambiguity of the plateau in finding the start of frame. However, the resulting autocorrelation operation is sensitive to additive white Gaussian noise (AWGN) and frequency selectivity.

Kishore and Reddy [10] presented an algorithm that requires knowledge of the time domain preamble in the receiver to compute a cross-correlation metric between the known and received preamble

Manuscript received February 2, 2012; revised June 6, 2012; accepted July 21, 2012. Date of publication September 7, 2012; date of current version July 22, 2013.

T. H. Pham is with Nanyang Technological University, 639798 Singapore, and also with the TUM-CREATE Centre for Electromobility, 138649 Singapore (e-mail: hung3@e.ntu.edu.sg).

S. A. Fahmy and I. V. McLoughlin are with Nanyang Technological University, 639798 Singapore (e-mail: sfahmy@ntu.edu.sg; mcloughlin@ntu.edu.sg).

Digital Object Identifier 10.1109/TVLSI.2012.2210917