CP524 05/4/12

Restricted Moduli Symmetrical Signed Residue Addition: Part I

First M. N. Daikpor, Second O. Adegbenro, Member, IEEE

Abstract — In this paper we propose a scheme for the design of a Symmetrical Multiple Valued Logic (SMVL) arithmetic circuit based on the use of restricted moduli Symmetrical Signed digit Residue Number system (SSRNS). Sign and overflow detection as well as magnitude comparison operations are accomplished without recourse to the traditional complex Mixed Radix number System (MRS) conversion process and multiplicative inverse computation. The method is particularly general purpose systems oriented. Addition operations are executed economically, fast and at constant speed.

Keywords — **conversion, carry-free, full adder, magnitude, number system, symmetrical-signed-residue, weighted.**

I. INTRODUCTION

PRACTICALLY all human endeavours today are Information and Communication Technology Systems (ICTS) driven. These systems rely on high speed, secured and trusted communication gadgets whose operations depend on special classes of very big integer arithmetic circuits. The type of arithmetic operations employed in these devices is not only fixed point but also must be carry-free. The method of number representation is a critical design factor in order to attain the desired high-speed operations of these circuits; designers still have to find optimal ways of managing carry propagation chains [1], [2].

It has long been established that the non-redundant non-weighted Residue Number System (RNS) and the weighted highly redundant Signed-Digit Number System (SDNS), attract fast and efficient arithmetic. [2], [3], [4], [5], [6]. However, RNS arithmetic is beset with complex conversion procedure, difficult sign detection, cost intensive overflow detection and magnitude comparison strategies that require Mixed Radix number System (MRS) and practically none in-existence simple division algorithm. Hence, RNS arithmetic found application in special areas such as: error correction, fault tolerance and digital filter design, power dissipation reduction in VLSI design, fast Fourier transform structures and cryptography [7], [8], [9], [10], [11]. On the other hand, the SDNS representation in addition to the above benefits of RNS provides a means of presenting operands at higher radices thereby supporting SMVL systems design which find application in image processing, robotic, and finite field arithmetic [6], [7]. The SDSN drawback is that it requires more character digits for operands representation. Thus in systems of high frequency addition operation this requirement is a setback in terms of circuit power consumption.

It is already established that combined Signed Digit (SD) and Residue Number (RN) arithmetic could result in reduced carry propagation delay and power consumption but the deployment of this in fast arithmetic circuit design is still at its infancy. For example the related arithmetic circuits in [2] are parameterized to provide basic building blocks for binary logic signal VLSI processors. Binary logic VLSI circuit's main objective is miniaturization with improved circuit complexity all at reduced cost. The Multiple Valued Logic (MVL) systems extend the horizon of this objective by virtue of its higher information per line capacity. Very recently, a Symmetrical Multiple Valued Logic (SMVL) developed from Restricted radix-7 Quaternary Signed Digit (Rr7SqSd) number system has been proposed [7]. It is highly probable that an interesting cross line could evolve when the character digit set of a symmetrical SDNS coalesces with a symmetrical signed digit RNS character digit set.

The proposed SSRNS addition scheme widens the scope of RNS arithmetic application by removing the inherent bottle neck in RNS arithmetic operations namely: sign detection, overflow detection and magnitude comparison. This contribution will thus make RNS arithmetic possible for use in general purpose digital systems. The organization of the remaining part of this paper is as follows. The background to this paper is presented in section II. Section III presents the SSRNS and the conversion procedures. SSRNS addition is presented in section IV.

II. BACKGROUND

Exegesis of RNS arithmetic can be found in popular works such as [2], [4], and [5]. Similar works on Signed-Digit Number System (SDNS) arithmetic can be found in [6]. The concept of restricted radix-7 Symmetrical quaternary Signed-digit number system (Rr7SqSd) and its arithmetic is in the most recent works of [7], [12], [13], and [14]. Hence, by extension restricted radix-5 Symmetrical ternary Signed-digit (Rr5StSd) number system with the character digit set $L \in \{-2,-1,0,1,2\}$ and Restricted radix-3

Corresponding First M. N. Daikpor, is with Faculty of Engineering, Department of Electrical and Electronics Engineering, University of Lagos, Akoka Lagos , Nigeria e-mail michaelnnaseimodaikpor@rocketmail.com
Second O. Adegbenro, is with the Centre for Energy Efficiency and Conservation University of Lagos, Akoka Lagos , Nigeria e-mail wole_adegbenro@yahoo.com

Symmetrical binary Signed-digit (Rr3SbSd) number system exist with the character digit set $L \in \{-1,0,1\}$.

III. THE SSRNS

The number system we are proposing here provides a cross over point between the SDNS and the RNS. It therefore possesses the properties of the two number systems as it is composed of symmetrical signed residues numbers x_k, of the moduli set $p_1 p_2 .. p_k .. p_n$. A symmetrical signed residue number x_k, is a unique representation of the signed integer x, such that for a set of unsigned relative prime moduli set $P_k; k = 1,2,..,n$ the signed integer x can be described as

$$x = (x_1 | x_2 | .. | .x_n) SSRNS(P_1 | P_1 | ... | .P_N) \quad (1)$$

for all $|x| \in [-M, M]$, where: $x_k = x \bmod p_k$, $|M| = \prod_1^n p_k$ and if ζ_k is a restrictor $\leq |p_k - 1|$ of modulo p_k residues then the SSRNS digit x_{p_k} takes the values

$$x_{p_k} \in \left(\left(\overline{(p_k - (\zeta_k - 1))} \right), \bar{1}, 01, , (p_k - (\zeta_k - 1)) \right) \quad (2)$$

Now for bridging Rr7SqSd and SSRNS we take as an example $k = 3$ and the relative prime moduli set as $(7|5|3|)$ which provide a unique representation of any signed integer x, in the dynamic range $-105 < x < 105$. It then follows that with $(\zeta_7, \zeta_5, \zeta_3) = (3,2,1)$

$$\left.\begin{aligned} x_{7_k} &\in \{-3,-2,-1,0,1,2,3\} \\ x_{5_k} &\in \{-2,-1,0,1,2\} \\ x_{3_k} &\in \{-1,0,1\} \end{aligned}\right\} \quad (3)$$

The following procedures apply for the specified cases.

A. Decimal to SSRNS conversion

1. Obtain the $radix - M/2$ form $X^* = X^*_{m-1} X^*_{m-2} .... X^*_0$, of the number.
2. For each X^*_i, compute the RNS residue $T_{i_k} = X^*_i \bmod p_k$.
3. Represent T_{i_k} in SSRNS i.e. $t_{i,l} = f(T_{i_k})$ and $l \in \{1,2,3\}$

Hence, $X = \{X^*_i\} = (t_{i,1} t_{i,2} t_{i,5}) SSRNS(7|5|3)$,

e,g $X = 65$; $X_i = (2|1|\bar{1}) SSRNS(7|5|3)$.

B. Rr7SqSd to SSRNS

1.Partition the given number $X_{Rr7SqSd} = x_{Rj-1} x_{Rj-2} .... x_{Rj0}$ where $x_{Rj-i} \in \{-3,-2,-1,0,1,2,3\}$, from the right into m groups of 3-Rr7SqSd

$$x_{Rj,3} x_{ij2} x_{j,1} = (\alpha_j \beta_j \lambda_j),$$
$$-172 < \alpha_j, \beta_j, \lambda_j < 172$$
$$\alpha, \beta, \lambda \in \{-3,-2,-1,0,1,2,3\}.$$

2.Compute the SSRNS equivalent $t_{R1j}, t_{R2j}, t_{R3j}$ of the Rr7SqSd j^{th} partition $\alpha_j \beta_j \lambda_j$ as follows;

$$\left.\begin{aligned} t_{Rj1} &= \lambda_j \\ t_{R2j} &= (7(7\alpha_j + \beta_j) + \lambda_j) \bmod p_2 \\ t_{R3j} &= |(\alpha_j + \beta_j + \lambda_j)| \bmod p_3 \end{aligned}\right\} \quad (4)$$

e.g.

$$1\bar{3}23 = 213_{10} = 2|3_{Radix-M/2} = (22\bar{1}|30\bar{1}) SSRNS(7|5|3).$$

C. SSRNS to decimal

Rather than using the Mixed Radix number System (MRS) , multiplicative or Look Up Tables (LUT) the following formular is developed heuristically. If $p_1 p_2 ... p_k$ is a set of relatively prime moduli in the interval $\left(-\frac{M}{2}, \frac{M}{2}\right)$ $M = 2\prod_{k=1}^{l} p_k$ then there exist the set of integers $u_1, ..., u_k$, $-\frac{M}{2} \leq u_i \leq \frac{M}{2}$ such that

$$u_i \bmod p_j = \begin{cases} \pm 1 \ ; \text{ if } i = j \\ 0 \ ; \text{if } i \neq j \end{cases} \quad (5)$$

Satisfying the equation (10)

$$u_1 + ... + u_l = \frac{M}{2} + 1 \text{ or } u_1 - \frac{M}{2} +$$
$$.. + u_l - \frac{M}{2} = -(M-1) \text{ ;for } u_1 \bmod p_1 =$$
$$.... = u_l \bmod p_l = 1 \quad (6)$$

Similarly,

$$-u_1 - ... - u_l = -\left(\frac{M}{2} + 1\right) \text{ or } \frac{M}{2} - u_1 +$$
$$.. + \frac{M}{2} - u_l = M - 1 \text{ ; for } u_1 \bmod p_1 =$$
$$.. = u_l \bmod p_l = -1 \quad (7)$$

If the decimal equivalent of the SSRNS digits $t_1 t_2 .... t_l$ is φ_i $-157 \leq \varphi_i \leq 157$ and denoting $\mu_k = u_k \bmod p_k$, $k = 1,2,..,l$ then,

$$\left(\mu_1 \prod_{h=2}^{l} p_h\right) \bmod p_1 = 1 \Rightarrow \mu_1 = 1$$

.....

$$\left(\mu_m \prod_{h^*=1}^{m-1} p_{h^*} \cdot \prod_{h=m+1}^{l} p_h\right) \bmod p_m = 1 \Rightarrow \mu_m = 1 \quad (8)$$

.....

$$\left(\mu_l \prod_{h=1}^{l-1} p_h\right) \bmod p_l = 1 \Rightarrow \mu_l = l-1$$

provided: $u_k = \mu_k \dfrac{M/2}{p_k}$. Consequently,

$$\varphi = \sum_{k=1}^{l} \mu_k \frac{M/2}{p_k} t_k = \sum_{k}^{l} u_k t_k \qquad (9)$$

For $l_{\max} = 3$; $\mu_1 = 1$, $\mu_2 = 1$ and $\mu_3 = 2$ therefore; $u_1 = 15$, $u_2 = 21$ and $u_3 = 70$. Hence,

$$\varphi_i = \left(15t_1 + 21t_2 + 70t_3\right) \qquad (10)$$

Long SSRNS digits strings are partitioned into g groups of $3-SSRNS$ digits and

$$\varphi = \sum_{i=0}^{g-1} \varphi_i = \sum_{i=0}^{g-1} \left(15t_{1i} + 21t_{2i} + 70t_{3i}\right)\left(M/2\right)^i \quad (11)$$

IV. SSRNS ADDITION SCHEME

In this contribution we are more concerned with very big integer operand SSRNS addition operations that are operands are converted to $radix - M/2$ number system character digits and subsequently to SSRNS. We describe the addition operation by equation (12)

$$\left.\begin{aligned} \delta_{ip} &= \begin{cases} \overline{1} & ;\text{if } \alpha_{ip} + \beta_{ip} < \overline{a}_p \\ 1 & ;\text{if } \alpha_{ip} + \beta_{ip} > a_p \\ 0 & ;\text{otherwise} \end{cases} \\ \gamma_{ip} &= \alpha_{ip} + \beta_{ip} - p\delta_{ip} \\ a_p &\in (3,2,1)RNS(7|5|3) \end{aligned}\right\} \qquad (12)$$

Where $\alpha_{ip_k}, \beta_{ip_k}$ are operand pairs and γ_{ip_k} sum of the signed $radix - M/2$ character digits in the SSRNS representation addition operation. $\alpha_{7i}, \beta_{i7}, \gamma_{7i} = \{x_{7_k}\}$, $\alpha_{5i}, \beta_{5i}, \gamma_{5i} = \{x_{5_k}\}$, and $\alpha_{3i}, \beta_{3i}, \gamma_{3i} = \{x_{3_k}\}$ as earlier defined in equation (3). There are inherent problems here, i) since the addition is $radix - M/2$ pair-wise there is danger of the weighted number systems equivalent of operand's $radix - M/2$ character digit-partitions to acquire varying signs that may lead to inaccurate computed final result. ii) The magnitude ψ_o, of any operand participating in an SSRNS arithmetic lies in the dynamic range $-M/2 < \psi_o < M/2$ just as that of the computed sum φ_s lies in the interval $[-M, M]$. The implication of these observations is that there is bound to be numerical trimming of any computed sum magnitude that lies outside the SSRNS dynamic range. This of course can bring about fictitious sums of the addition operation. Which means long SSRNS digit input string addition schemes must have mechanism for identifying/detecting $radix - M/2$-partition sign, operand-pair and result pairs parity (odd or even) status, extend of partition sum overflows detection and it's reporting. Existing methods of solving these difficult-to-handle problems in the RNS domain [8], [9] as earlier enunciated require conversion of the RNS operands to MRS domain for computability as well as multiplication inverse computations.

In this paper, these problems are again solved heuristically in a very simple way using the Rr7SqSd addition. The $Radix - M/2$ character digits input stream of operands appear both in 3-Rr7SqSd and in the corresponding 3-SSRNS digits packet streams. The 3-Rr7SqSd operand-pairs are added together per Rr7SqSd in parallel in a four level Rr7SqSd addition operation described by equation (13)

$$\left.\begin{aligned} z_i &= \alpha_i + \beta_i \\ \delta_i &= \begin{cases} -1 \text{ ; If } z_i < \overline{a} \\ 1 \text{ ; if } z_i > a \\ 0 \text{ ; if } otherwise \end{cases} \\ \tau_i &= z_i - 7\,\delta_i \\ \lambda_i &= \tau_i + \delta_{i-1} \end{aligned}\right\} \qquad (13)$$

where: $\alpha_i, \beta_i, \tau_i, \lambda_i \{-3,-2,-1,0,1,2,3\}$ and δ_i, δ_{i-1} $\in$ { -1,0,1}. The addition accomplishes the magnitude comparison aspect. The sign of the i^{th} $radix - M/2$ character digit pair Rr7SqSd addition operation is the sign of the most significant Rr7SqSd . of that partition and the corresponding outputs $\lambda_{i=} = \delta_i$, $\lambda_{i,3}$, $\lambda_{i,2}$ and $\lambda_{i,1}$ facilitates overflow discussion making. The actual $radix - M/2$ character digit-pair addition operation is executed in SSRNS domain modulo-wise using equation (16). Overflows appear just as inter-radix carries.

One immediate area of application for the SSRNS arithmetic circuits is in SMVL systems where the processing signal profile, rather than the binary logic, is the Rr7SqSd logic levels. The Rr7SqSd number system being weighted is susceptible to carry propagation chains that adversely affect system operation speed. To avoid this problem, we embed SSRNS addition procedure in Rr7SqSd addition operation. Long Rr7SqSd-input-string operands $x_i = x_{i_3}, x_{i_2}, x_{i_1}$, $y_i = y_{i_3}, y_{i_2}, y_{i_1}$ are divided into m-groups of 3-Rr7SqSd each with $-171 \le x_i, y_i \le 171$. Let $-342 \le z_i \le 342$, be the

sum of the i^{th} partition addition be such that $z_i = z_{i3}z_{i2}z_{i1}$, $z_{il} \in \{-3,-2,-1,0,1,2,3\}$ $l \in \{3,2,1\}$. In the SSRNS domain, two $radix - M/2$ character digits are required to represent x_i, y_i and z_i. Taking $(\theta_{0i}\theta_{1i}),(\vartheta_{0i}\vartheta_{1i})$ and $(\gamma_{0i}\gamma_{1i})$ as x_i, y_i and z_i then $-(M/2-1) \le \theta_{1i}, \vartheta_{1i}, \gamma_{1i} \le (M/2-1)$, $-1 \le \theta_{0i}\vartheta_{0i} \le 1$ and $-3 \le \gamma_{0i} \le 3$. To reduce cost and enhance operation speed operands are first represented in $radix - M/2$ character digits and then each $radix - M/2$ character digit is converted to both Rr7SqSd and SSRNS presentations. In terms of $radix - M/2$ the addition process is represented by equations (17), (18) and (19)

$$\gamma_{1i} = \theta_{1i} + \vartheta_{1i} \tag{17}$$

$$c_i = \begin{cases} \pm 1 \text{ ; if } \gamma_{1i} \ge M/2 \text{ or } = \pm(M/2-1), \\ \gamma_{1i} = \pm M/2 - 1, \gamma_{1i-1} \text{ and } c_{i-2} = \pm 1 \\ 0 \quad \text{; Otherwise} \end{cases} \tag{18}$$

$$\gamma_{1,i+1} = \begin{cases} \gamma_{1i+1} \mp lM/2 \text{ ; if } c_i = \pm 1 \\ \gamma_{1i+1} \quad \text{; otherwise} \end{cases} \tag{19}$$

Where $l \in \{-3,-2,-1,0,1,2,3\}$ corresponding to $\{-315,-210,-105,0,105,210,315\}$ and $\theta_{1p} \in \{\alpha_{7i}, \alpha_{5i}, \alpha_{i3}\}, \vartheta_{1p} \in \{\beta_{7i}, \beta_{5i}, \beta_{3i}\}$.

Computational experiments conducted showed 50% operation execution speed increment using this detection-compare-migrate-and-return for alignment approach. There is a 75% increase in speed when magnitude alignment is also carried out in the SSRNS domain though with a higher complexity trade-off. The approach does not need a Chinese Remainder Theory (CRT) and the Extended Euclidean Algorithm (EEA) for backward conversion operation.

REFERENCES

[1] R. Zimmermann, "Lecture notes on computer arithmetic: principles, architectures and VLSI design," Integrated system Laboratory, Swiss Federal Institute of Technology (ETH), CH-8092 Zurich, Switzerland. March 16, 1999. http://www.iis.ee.ethz.ch/

[2] A. Lindstrom, M. Nordseth, L. Bentsson and A. Omondi,"Arithmetic circuits combing residue and signed-digit representations," in Proc. of the 8th Asia-Pacific Computer Systems Architecture conference (ACSAC"2003) Aizu-Wakamatsu City Japan , Sept. 2003, Published by Springer-verlag in lecture Notes in Computer Science(LNCS) Vol. 2823.

[3] S. Shieh and C. Wu, "Asymmetric high-radix signed-digit number system for carry-free addition- Accepted for publication," Journal of Information Science and Engineering 19, February 24, 2003, pp. 1015-1039.

[4] B. Parhami, "RNS Representations with redundant residues- Invited Paper," IEEE (2001) 0-7803-7147-X/01/$10.00, pp. 1651-1655.

[5] S. Wei, "Number conversions between RNS and mixed-radix number system based on modulo (2^p-1) signed-digit arithmetic," Proc. SBCCT'05 September 4-7, 2005, Florianopolis, Brazil, ACM 1-59593-174-0/05/0009, pp. 160- 165.

[6] M. Kameyama and Tatsuo Higuchi, "Design of a radix- 4 signed-digit arithmetic circuit for digital filters,"Proc. of the 12th IEEE International Symposium on multiple-Valued Logic, North-western University, Evanston, IL. USA 1980, pp 272 -277.

[7] M.N Daikpor and O. Adegbenro," Radix - 7 signed digit element finite field arithmetic," Proc of the IEEE International conference on signal processing and communication, Dubai November 2007, pp 708 – 711.

[8] C. K. Cheng, "CSE: Computer arithmetic algorithms and hardware design lecture 2: redundant and residue number systems," 2006, http://cseweb.ucsd.edu/classes/fa06/cse246/lect2.pdf.

[9] N. Stamenkovic, "Digital FIR filter architecture based on residue number system," FACTA UNIVERSITATIS (NIS) Ser.:ELEC.ENERG. vol.22, N0. 1, April 2009, pp 125-140.

[10] M. Hosseinzadeh, S. Jafarali and K Navi., "A novel multiple valued logic OHRNS modulo r^n adder circuit," Proc of World academy of Science, Engineering and Technology, Vol. 25 November 2007 ISSN 1307-6884, pp 128 – 132.

[11] F. Barsi and P Maestrini, "Error correcting properties of redundant residue number systems," IEEE Transactions on Computers, Vol. C-22.N0. 3, March 1973, pp 307 – 314.

[12] B.Tseng, G.A. Jullien and W. Miller, "Implementing FFT structures Using the Residue Number System," IEEE Transactions on Computer, vol. C-28, N0 11, November 1979, pp 831 – 845.

[13] F. J. Taylor, "A VLSI Residue Arithmetic Multiplier," IEEE Transactions on Computers, Vol. C-31, N0. 6, June 1982, pp 540 – 546.

[14] V. Parlours and T. Strouraitis, "Novel High-radix Residue Number System Architectures, " IEEE transaction circuits and systems-II Analog and Digital Processing , Vol. 47, N0. 10 October 2000, pp 1059 -1073.

[15] M.N. Daikpor and O. Adegbenro, "Efficient carry-free Rr7SqSd addition algorithm". Proc of the 3rd International Conference on Emerging Trends, Research directions and Training Requirements of the 21st century Electrical and Electronics Engineering, University of Lagos . 22nd to 24th, July 2009. Pp 102- 107.

IEEE Region 8

IEEE Bosnia and Herzegovina Section

IEEE Bosnia and Herzegovina Section - Chapter of the Communications Society

IEEE Bosnia and Herzegovina Section - Chapter of the Computer Society

IEEE Croatia Section

EURASIP - European Association for Signal Processing

2011 18th International Conference on Systems, Signals and Image Processing

PROCEEDINGS IWSSIP 2011

Edited by: Branka Zovko-Cihlar
Narcis Behlilović
Mesud Hadžialić

16-18. June 2011, Sarajevo, Bosnia and Herzegovina

ISBN: 978-9958-9966-1-0
IEEE Catalog number: CFP1155E-PRT

Author Index

Restricted Moduli Symmetrical quaternary Signed-digit Addition: A design Implementation Overview

Michael Naseimo Daikpor
Department of Electrical and Electronics Engineering,
University of Lagos - Akoka,
Lagos, Nigeria.
e-mail: michaelnaseimodaikpor@rocketmail.com

Oluwole Adegbenro, Member IEEE
National Centre for Energy Efficiency and conservation,
University of Lagos – Akoka,
Lagos, Nigeria.
e-mail: wole_adegbenro@yahoo.com

***Abstract*—In this paper we present an overview of design implementation of a Symmetrical Multiple Valued Logic (SMVL) arithmetic circuit based on the use of restricted moduli Symmetrical Signed Residue Number System (SSRNS). Restricted radix-7 Symmetrical quaternary Signed digit (Rr7SqSd) T-gate based interconnections and full adders are used to implement sign detection, overflow detection and magnitude comparison without recourse to Mixed Radix number System (MRS) converters design or Chinese Remainder Theorem (CRT) computation.**

Keywords-component; circuit design; full adders; magnitude generator; signed residue; T-gates

I. INTRODUCTION

High speed arithmetic circuits are the kernel of the emerging information and secured communication systems. They operate in such manners that very big integer operand chunks are fetched and executed in one clock cycle. This requires precise and concise arithmetic operation carry propagation chain reduction techniques [1],[2] during the VLSI circuit design stage. The developments of quantum functional semiconductor devices such as the Resonant Tunneling Device (RTD) and the Single Electron Tunneling Transistor (SETT) as well as the SMVL concept [3] are some of the other current research directions. An absolute carry-free arithmetic circuit that can bring about ultra-high speed system operation is the ultimate target.

It is already established that the Residue Number System (RNS) and the Signed-Digit Number System (SDNS) separately provide efficient arithmetic and parallel system architecture, necessary for high speed operations [4],[5],[6],[7] . However, the RNS is beset with difficult sign and overflow detection as well as complex magnitude comparison procedure and SDNS has the drawback of requiring more character digits to represent operands.

In the part I of this work [8] we suggested the cross over point for RNS and SDNS arithmetic in a way that resulted in tremendous increase in the addition operation execution speed. In this part II, we present the hardware implementation considerations of the scheme. This paper is organized as follows. In section II we present a brief review of the SSRNS addition scheme. The circuit realization is presented in section III while performance evaluation is presented in section IV.

II. SSRNS ADDDTION REVIEW

The SSRNS is composed of symmetrical signed residue numbers x_k, of the moduli set $p_1 p_2 .. p_k .. p_n$. A symmetrical signed residue number x_k, is a unique representation of the signed integer x, such that for a set of unsigned co-prime moduli set $p_k; k = 1,2,..,n$ the signed integer x can be described as

$$x = (x_1 | x_2 | ... | x_n) SSRNS(p_1 | p_2 | ... | p_n) \quad (1)$$

for all $|x| \in [-M, M]$, where: $x_k = x \bmod p_k$,

$|M| = \prod_1^n p_k$. If ζ_k is a restrictor $\leq |p_k - 1|$ of modulo p_k residues then the Symmetrical Signed Residue Digit (SSRD) x_{p_k} takes the values

$$x_{p_k} \in \{(\overline{p_k - (\zeta_k - 1)}), .., \bar{1}, 0, 1, ..., (p_k - (\zeta_k - 1))\} \quad (2)$$

For the co-prime moduli set $(7|5|3|)$ a unique representation of any signed integer x in the dynamic range $-105 < x < 105$ can obtained. If the restrictors $(\zeta_1, \zeta_2, \zeta_3) = (3,2,1)$ corresponds to the moduli set $\{7,5,3\}$ then

$$\left.\begin{aligned} x_{7_k} &\in \{-3,-2,-1,0,1,2,3\} \\ x_{5_k} &\in \{-2,-1,0,1,2\} \\ x_{3_k} &\in \{-1,0,1\} \end{aligned}\right\} \quad (3)$$

In our design, sign and overflow detection as well as magnitude comparison operations are performed in Rr7SqSd environments. The exact addition operation is conducted in SSRN domain. Operands pairs are initially converted into packets of $radix - M/2$ character digit strings. Each

$radix-M/2$ character digit is then converted to Rr7SqSd and SSRNS forms of number representations that yield 3-Rr7SqSd and 3-SSRNS digits per partition respectively. The addition operation is performed simultaneously on each $radix-M/2$ character digit pair now appearing as 3-SSRNS digit partition-pair. Equations (4),(5), and (6) describe the addition process in $radix-M/2$ character digit, Rr7SqSd and SSRNS respectively.

$$\left.\begin{aligned} \gamma_i &= \theta_i + \vartheta_i \qquad (4a)\\ c_i &= \begin{cases} \mp 1 \text{ ; if } \gamma_i \geq M/2 \text{ or } \gamma_i = \pm\left(M/2 - 1\right), \\ \qquad \gamma_{i+1} = \pm M/2 - 1 \text{ and } c_{i-1} = \pm 1 \\ \qquad\qquad (4b) \\ 0 \text{ ; otherwise.} \end{cases} \\ \gamma_{i+1} &= \begin{cases} \gamma_{i+1} \mp l\, M/2 \text{ ; if } c_i = \pm 1 \\ \qquad\qquad (4c) \\ \gamma_{i+1} \text{ ; otherwise.} \end{cases} \end{aligned}\right\} \quad (4)$$

$\gamma_i \in \{-315,-314,.....,-1,0,1,...,314,315\}$ θ_i, ϑ_i are operand pairs in $radix-M/2$ but in Rr7SqSd or SSRNS forms of representation. $\theta_{p_i} \in \{\alpha_{7_i}, \alpha_{5_i}, \alpha_{3_i}\}$, $\vartheta_{p_i} \in \{\beta_{7_i}, \beta_{5_i}, \beta_{3_i}\}$ and $l \in \{-3,-2,-1,0,1,2,3\}$ is the modulo $M/2$ multiplier corresponding to $l\, M/2 \in \{-315,-210,-105,0,105,210,315\}$.

$$\left.\begin{aligned} z_i &= \alpha_i + \beta_i \\ \delta_i &= \begin{cases} -1 \text{ ; if } z_i < \bar{a} \\ 1 \text{ ; if } z_i > a \\ 0 \text{ ; if otherwise} \end{cases} \\ \tau_i &= z_i - 7\delta_i \\ \lambda_i &= \tau_i + \delta_{i-1} \end{aligned}\right\} \quad (5)$$

Where i) $\alpha_i, \beta_i, \tau_i, \lambda_i \in \{-3,-2,-1,0,1,2,3\}$ and ii) $\delta_i, \delta_{i-1} \in \{-1,0,1\}$. Again with $\alpha_{i_p}, \beta_{i_p}$ as the SSRNS operands pairs and γ_{i_p} sum of the signed $radix-M/2$

$$\left.\begin{aligned} \delta_{i_p} &= \begin{cases} -1 \text{ ; if } \alpha_{i_p} + \beta_{i_p} < \bar{a}_p \\ 1 \text{ ; if } \alpha_{i_p} + \beta_{i_p} > a \\ 0 \text{ ; if otherwise.} \end{cases} \\ \gamma_{i_p} &= \alpha_{i_p} + \beta_{i_p} - p\delta_{i_p} \end{aligned}\right\} \quad (6)$$

character digits in the SSRNS representation. $\alpha_{7_i}, \beta_{7_i}, \gamma_{7_i} \in \{x_{7_k}\}$, $\alpha_{5_i}, \beta_{5_i}, \gamma_{5_i} \in \{x_{5_k}\}$ and $\alpha_{3_i}, \beta_{3_i}, \gamma_{3_i} \in \{x_{3_k}\}$ as defined in (3).

III. THE SSRNS VLSI CIRCUIT

The SSRNS VLSI circuit is shown in fig. 1. All the functional blocks are implemented on Complementary Pass (CP) gate derived Rr7SqSd T-gates [3]. The $radix-M/2$ character digit-packet-streams are passed, operand pair-wise simultaneously to the Rr7SqSd and SSRNS converters. The first SSRNS adder performs the SSRNS addition on the input operand while the second handles possible inter $radix-M/2$ carry addition operation. Inter $radix-M/2$ carry digit generation and the overflow evaluation operations are accomplished in the 4-level 3,2,1,1-Rr7SqSd adders, the magnitude generator cascade and the wired AND-OR connection. The sign of a $radix-M/2$ character-digit partition addition operation result is simply the sign of the most significant digit Rr7SqSd addition operation's sum. The cascade of 3,2,1,1-Rr7SqSd adders execute the operations $\alpha_{3_i} + \beta_{3_i}$, $\alpha_{2_i} + \beta_{2_i}$, $\alpha_{1_i} + \beta_{1_i}$ and $\alpha_{1_i} + \beta_{1_i}$ according to equation (5).

The dynamic range of input operands to these adders is $-\left(M/2 - 1\right) \leq x \leq \left(M/2 - 1\right)$ which in the Rr7SqSd number system is $\bar{2}\,\bar{1}\,1 \leq x \leq 21\,\bar{1}$. The magnitude of a partition sum outside this range indicates an overflow. The least value of this sum is $\pm M/2 \equiv (210,\bar{2},\bar{1},0)_{Rr7SqSd}$ and the maximum is $\pm(171+171+1=343) \equiv (1000,\bar{1}\,000)_{Rr7SqSd}$. The output of Rr7SqSd adders cascade w_i, x_i, y_i and z_i goes into the 'magnitude generator' where equations (7) and (8) are implemented.

$$\omega_i^* = \begin{cases} \pm 1; \text{ if } \gamma_i^* \geq \pm M/2 \\ \qquad\qquad (7) \\ 0 \text{ ; otherwise.} \end{cases}$$

$$\varpi_{i-1} = \begin{cases} \pm 1; \text{ if } \omega_{i-2}^* = \pm 1,\ \gamma_i = \left(M/2 - 1\right) \\ \qquad \text{and } \gamma_{i-1} = \pm\left(M/2 - 1\right) \\ \qquad\qquad (8) \\ 0 \text{ ; otherwise} \end{cases}$$

The magnitude generator thus generates four outputs namely: ω_{1i}^* for $\gamma_i^* \geq \left|M/2\right|$, ω_{2i}^* for $\gamma_i^* = \left|M/2 - 1\right|$, ω_{3i}^* for

$\gamma_i^* < \left|M/2 - 1\right|$ and ω_{4i}^* for "sign of the addition" (sgn) operation. The first two outputs can be described by equations (9) and (10) respectively.

$$\omega_{1i}^* = \begin{cases} \pm 1 \text{ ; if } w_i = \pm 1 \\ 0 \text{ ; otherwise.} \end{cases} \quad (9)$$

$$\omega_{2i}^* = \begin{cases} \pm 1 \text{ ; if } |x_i| \le 2, |y_i| \le 1, z_i \ge 0 \text{ or} \\ \quad |x_i| = |y_i| \ge 2 \text{ or } w_i = 0, |x_i| = 3 \\ 0 \text{ ; if otherwise.} \end{cases} \quad (10)$$

Detailed explanation of the Rr7SqSd full adder circuit can be found in [9]. The same procedure is used to design the $SSRNS(7|5|3)$ adder.

IV. EVALUATION OF THE EXPECTED PERFORMANCE

A major cause of information delay in VLSI circuits is the signal propagation time between interconnected subunits. In our proposal, we adopted the most popular solution to this problem which is embracing an interconnection-free architecture by integrating several subunits into a single functional block. For example, the $SSRNS(7|5|3)$ adder block consists of moduli 7,5 and 3 SSRNS adders. Similarly, the $3-Rr7SqSd$ adder unit consists of 3 separate Rr7SqSd full adders. Consequently, the signal propagation delay time metric is used to evaluate the expected performance of the proposed VLSI circuit.

Let $\tau_{Rr7}, \tau_{MG}, \tau_{SSRNS+}, \tau_{Rr7+}, \tau_{AND\_OR}$ be the signal delay times of a 7-value CP-gate derived T-gate, the magnitude generator, $SSRNS(7|5|3)$ and Rr7SqSd full adders respectively and the wired AND-OR interconnection. The ratio of the number of $radix-M/2$ character digits n to a given operand length of $m >> 1$ decimal digits, we found to be $1:2$. Neglecting the time for converting operands to $Rr7SqSd\ /\ SSRNS$ representation the speed T, of the addition circuit can be described by equation (11)

$$T = \max(4\tau_{Rr7+}, \tau_{SSRNS+}) + \tau_{MG} + n/2(\tau_{AND-OR}) + \tau_{SSRNS+} \quad (11)$$

In [3] $\tau_{Rr7} = 300n\,\text{sec}$ while from our circuit's synthesis $\tau_{SSRNS+} = 2\tau_{Rr7}$, $\tau_{MG} = \tau_{AND-OR} = 4\tau_{Rr7}$, $\tau_{Rr7+} = 3\tau_{Rr7}$. Hence, the speed of a $radix-M/2$ character digit operands restricted moduli signed-digit residue addition operation is expected to be $6\mu\,\text{sec}$. The $\text{mod}-23$ $(\approx 6-bits)$ also a single $radix-M/2$ character digit operands addition in a bit-slice MVL architecture [10] required $10m\,\text{sec}$ [5] operation-execution time. Ordinary binary circuits where component interconnections are necessary for parallel processing required $100m\,\text{sec}$ to perform $\text{mod}-2^{16}$ or a 3, $radix-M/2$ character-digits arithmetic [5]. Using our proposed circuit the estimated periods for, $\text{mod}-2^{16}$, $\text{mod}-2^{64}$ and 2^{512} (corresponding to 3,10 and 96 $radix-M/2$ character-digits arithmetic operations) are: $7.2\mu\,\text{sec}$, $11.4\mu\,\text{sec}$ and $63\mu\,\text{sec}$ respectively. Most merging Information Technology and Secured Communication devices are elliptic curve cryptosystem supported. These devices require just $160-decimal$ digits or $512-bits$ key-size for an acceptable transmitted message security level. It is our opinion that the performance of such devices using our proposed scheme for the design implementation of the arithmetic circuits will no doubt have an edge over their contemporaries.

REFERENCES

[1] R. Zimmermann, "Lecture notes on computer arithmetic: Principles architecture and VLSI design." Integrated system Laboratory, Swiss Federal Institute of Technology (Technology (ETH), CH-8092 Zurich, Switzerland, March 16, 1999.

[2] Israel Koren, "Digital computer arithmetic ECE 666 Part 2 Unconventional number systems," (2004). http://www.ece.Umass.edu/ece/koren/arith/slides/Part2-unconv....

[3] Oluwole Adegbenro and Daikpor, M.N (2005), "Design of a 7-value Complementary Pass gate derived T-gate," Proc. International Engineering Conference, University of Lagos, 2005, Lagos Nigeria, Vol. 1 pp 146 – 158.

[4] S.Shieh and C. Wu, "Asymmetric high-radix signed-digit number system for carr-free addition. Accepted for publication", Journal of Information Science and Engineering 19, February 24, 2003, pp. 1015 – 1039.

[5] B. Parhami, "RNS Representation with redundant residues-Invited Paper," IEEE(2001) 0-7803-7147-x/01/S10.00, pp. 1651-1655.

[6] S.Wei, "Number conversion between RNS and mix-radix number system based on modulo (2^p-1) signed-digit arithmetic," Proc. SBCCT'05 September 4-7, 2005, Florianopolis, Brazil, ACM 1-59593-174-0/05/0009, pp. 160-165.

[7] M. Kameyama and Higuchi, "Design of a radix-4 signed-digit arithmetic circuits for digital filters," Proc. Of the 12th IEEE International Symposium on Multiple-valued Logic, North-western University, Evanston, IL. USA 1980. Pp 272 – 277.

[8] M.N. Daikpor and O. Adegbenro. "Restricted Moduli Symmetrical Signed Residue Addition Part I". Proc of the 18th Telecommunication forum TELFOR 2010, Serbia, Belgrade, November 23-25, 2010, pp 646 – 649.

[9] M.N. Daikpor and O. Adegbenro, " Efficient Carry-free Rr7SqSd Addition Algorithm," Proc of the 3rd International Conference on Emerging Trends, Research directions and Training Requirements of the 21st century

[10] M. Honda, M. kameyama and T. Higuchi, " Residue Arithmetic based Multiple-Valued VLSI Image Processor," Proc of the 22nd IEEE

International Symposium on Multiple-Valued Logic, May 27-29, 1992,
Sendai Japan, pp 330-336.

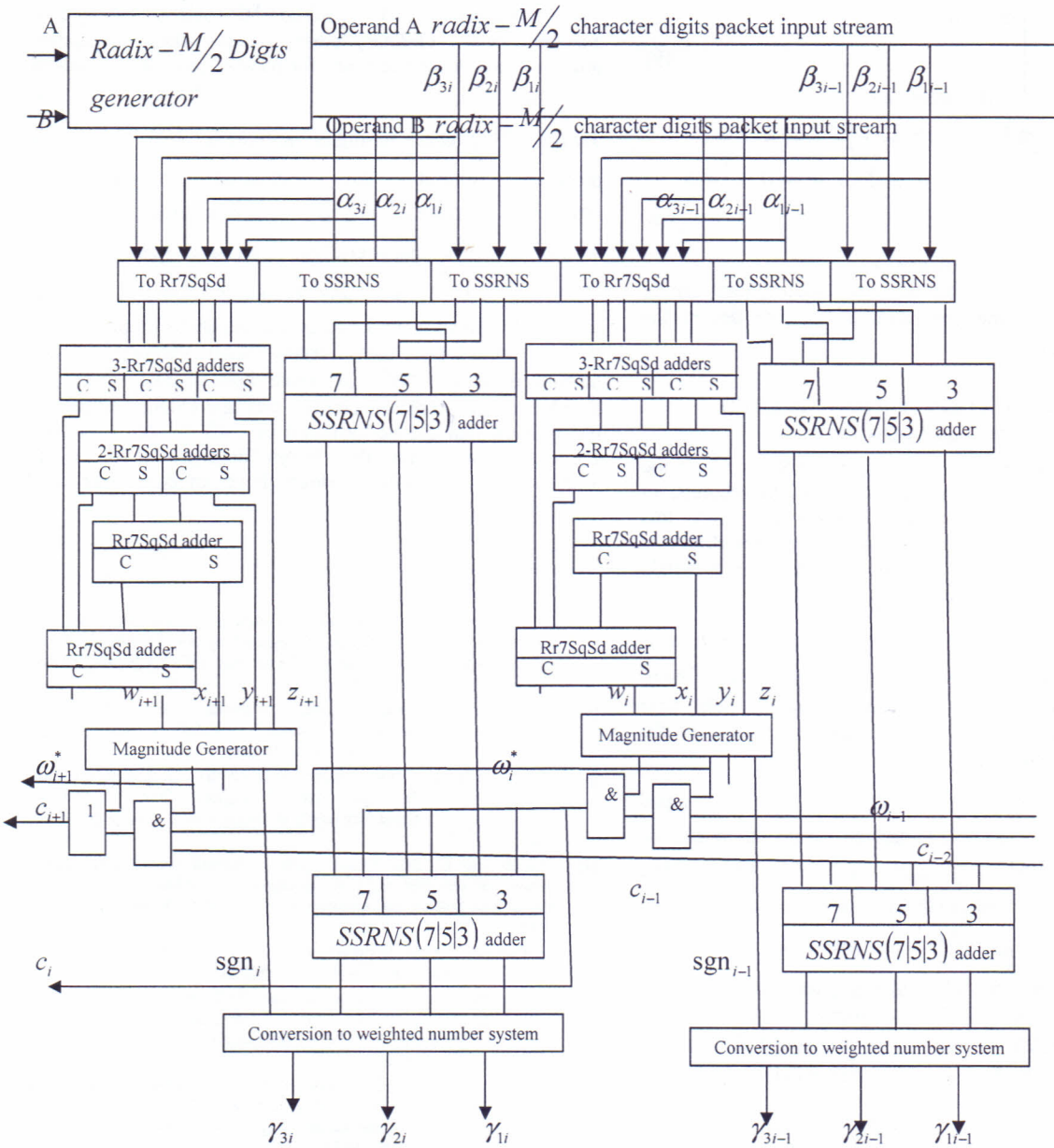


Fig. 1 Restricted moduli symmetrical SD residue parallel addition circuit

6th International Conference on Internet Technology and Secured Transactions, 11-14 December 2011, Abu Dhabi, United Arab Emirates

Concurrent Realization of the Multiply-by-7 Elliptic Curve Scalar Multiplication algorithm

Michael NaseimoDaikpor
Department of Electrical and Electronic Engineering,,
Faculty of Engineering,
University of Lagos,
Lagos, Nigeria
mndaikpor@yahoo.com

Oluwole Adegbenro
National Research Center for Energy Efficiency and Conservation
University of Lagos
Lagos, Nigeria
Oluwole_adegbenro@yahoo.com

Abstract— **this paper investigates the multiply-by-7 Elliptic Curve (EC) point P Scalar Multiplication algorithm for reduced computational complexity and enhanced inherent parallel property based on the Area-Time (AT^2) metric. The findings were compared with those obtained when the algorithm was again realized on the Jacobian projective coordinate and the Non-Adjacent Form (NAF). The investigation revealed 27% and 8% computational complexity reductions over the Jacobian and NAF realizations. The algorithm also presents the best AT^2 value.**

Keywords- concurrent processing; computational complexity; data flow graph; scheduling algorithm

I. INTRODUCTION

Todays' emerging Information Technology and secure Communication Systems (ITCS) such as: pocket size laptop computers, palm held operating systems , Personal digital assistant, ubiquitous sensor networks, automated teller machines, smart cards, mobile phones are beneficiaries of Elliptic Curve Cryptography (ECC) [1], [2]. These gadgets are driven by EC arithmetic operations of which scalar or point [3] multiplication is the major operation. EC scalar multiplication is essentially scaling a random point $P(x, y)$ on the elliptic curve with a scalar quantity $m >> 1$, in such manner that the product $W = mP$ is also a point $P(x_W, y_W)$ on the same elliptic curve. Finding W, from known values of P and m is easy; but finding m from given values of W and P is an exponentially hard mathematic problem known as the Elliptic Curve Discrete Logarithm Problem (ECDLP). While ECDLP is the main attraction of EC into Cryptography the manner of computing mP is also a critical issue in the design of secure communication systems [4], [2]. Elliptic curve scalar multiplication computation is based on two basic arithmetic operations namely: EC points' addition and EC point doubling. They are in turn calculated using finite field addition, multiplication, squaring and inversion operations. The inversion operation is slow and expensive.

The conventional method of computing $W = mP$ by $m-1$ consecutive EC point addition operations in the affine coordinate system requires $m-1$ field inversions. For large values of m the procedure is sluggish, slow and cost prohibitive. There are now various high speed EC scalar multiplication techniques. The Double Base Number System (DBNS) [6], the direct Tripling $3P$ and Quadrupling $4P$ approach in [3], the binary addition/subtraction chain methods of [7] have become house hold methods. Another popular method of accelerating EC scalar multiplication operation is to perform the main computation in projective coordinate system [5] that is inversion free and return to the affine system for the final x_W, y_W computation. Many forms of projective coordinate systems now exist but the choice of a particular is determined by the dominating EC arithmetic operation. Most of the existing scalar multiplication algorithms are serial or sequential machine implement-oriented. This poses on devices upper speed limit. Consequently, current solutions have not been able to totally meet the intended high speed requirements such as in real time critical services delivery system.

Naturally EC arithmetic is concurrent implement-friendly in nature so attention now focuses on exploiting this inherent parallelism to accelerate scalar multiplication operation. Concurrent realization of an operation is all about performing several sub operations simultaneously provided resources are available to extract optimal performance from the system. The resources are usually the hardware circuits of the most dominant arithmetic operation(s). Generally with EC arithmetic, field multiplication is the dominant operation hence; the resources in this case are multipliers. Pioneer works on concurrent EC scalar multiplication computation procedures can be found in [8] where non adjacent binary sequences are used to reduce the total number of addition. In [9], where scalable multipliers are used to replicate design for varying key sizes and [10] in which EC processors are designed and simulated on FGPA. These works remain novel and laudable contributions. However, they are all based on the projective coordinate systems that trade fewer number of

inversion operations for increased computational complexity cost.

In this contribution, we model concurrent realization of EC scalar multiplication in the affine coordinate system based on a simple multiply-by-7 algorithm to reduce computational complexity and enhance operation speed. The rest of this paper is organised as follows. In section II we recall the normal EC basic arithmetic operation equation in the affine and Jacobian coordinate systems and a slight modification we have made in the original affine coordinate version. The multiply-by-7 scalar multiplication algorithm is presented in section III. The concurrent computation model is presented in section IV. In section V we briefly discuss the results of our investigation. Conclusion is given in section VI.

II. EC BASIC OPERATION EQUATION

EC rank after straight lines and conics in the hierarchy of curves. They are projections of non-singular genus one algebraic curves defined over some **K** fields with k rational points and a point at infinity constituting a group. It is customary [11], [12], and [13] to concisely describe an elliptic curve E, together with the point O, at infinity defined over either binary or prime fields by (1).

$$E:\begin{cases} y^2 = x^3 + ax + b \ ; \text{for prime fields} \\ p \geq 3; a,b,(x,y) \in GF(p^k) \text{ and} \\ 4a^3 + 27b^2 \neq 0 \\ y^2 + xy = x^3 + ax^2 + b \ ; \text{for binary Galois} \\ \text{fields and } \ a,b,(x,y) \in GF(2^k) \end{cases} \quad (1)$$

If the three points $U = P(x_U, y_U), V = P(x_V, y_V)$ and $W = P(x_W, y_W) \in E$ then, $W = V + U$ and $W = U + U = 2U$ are respectively referred to as the EC points' addition and point doubling operations. These operations are traditionally computed in two phases namely, pre-computation of the quantity λ, phase and a cardinal point components x_W, y_W computation phase. The prime field versions of the two phases can be represented by (2).

$$\begin{cases} \lambda = \begin{cases} \dfrac{y_V - y_U}{x_V - x_U} \ ; \text{if points' addition} \\ \dfrac{3x_U^2 + a}{2y_U} \ ; \text{if point doubling} \end{cases} \\ x_W = \begin{cases} \lambda^2 - x_U - x_V \ ; \text{if points addition} \\ \lambda^2 - 2x_U \ ; \text{if point doubling} \end{cases} \\ y_W = \lambda(x_U - x_W) - y_U \end{cases} \quad (2)$$

As earlier mentioned performing EC arithmetic operations in the projective coordinate system increases computational complexity but it is less expensive. In the projective and in particular the Jacobian projective coordinate system a point $P(x,y)$ in the affine coordinate is equivalent to $P\left(X/Z^2, Y/Z^3\right)$. Thus, (2) in the Jacobian projective coordinate is represented as in (3)

$$\begin{cases} \lambda = \begin{cases} \dfrac{Y_v/Z_v^3 - Y_U/Z_U^3}{X_V/Z_V^2 - X_U/Z_U^2} \ ; \text{if points' addition} \\ \dfrac{3X_U^2 + a}{2Y_U/Z_U^3} \ ; \text{if point doubling} \end{cases} \\ X_W = \begin{cases} \lambda^2 - X_U/Z_U^2 - X_V/Z_V^2 \ ; \text{if points' addition} \\ \lambda^2 - 2X_U \ ; \text{if point doubling} \end{cases} \\ Y_W = \lambda^2\left(X_U/Z_U^2 - X_W/Z_W^2\right) - Y_U/Z_U^3 \end{cases} \quad (3)$$

Equations (2) or (3) form the bases of all scalar multiplication techniques. However, in both (2) and (3) y_W component is computed based on the availability of the x_W component. We eliminated this x_W before y_W requirement by modifying (2) in such manner that x_w or y_W is computed directly from source to enhance parallel hardware-implementation friendly architecture. The modification is described by (4).

$$\begin{cases} x_W = \begin{cases} \dfrac{\Delta^2 y - \Delta^2 x \nabla x}{\Delta^2 x} \ ; \text{if EC points' addition} \\ \dfrac{A_0^2 - 8x_u y_u^2}{(2y_u)^2} \ ; \text{if EC point doubling} \end{cases} \\ y_W = \begin{cases} \dfrac{\Delta^3 y - \Delta^2 x(\nabla x \Delta y - \Delta x y)}{\Delta^3 x} \ ; \text{if} \\ \text{points' addition operation} \\ \dfrac{-A_o^3 + 12A_o x_u y_u^2 - 8y_u^4}{(2y_u)^3} \ ; \text{if} \\ \text{point doubling operation} \end{cases} \end{cases} \quad (4)$$

Where: $\nabla x = x_V + x_U$, $\Delta x = x_V - x_U$, $\Delta y = y_V - y_U$, $\Delta xy = x_V y_U - x_U y_V$ and $A_0 = 3x_U^2 + a$. It is important to note that migrations to projective coordinate system only postpone inversion operation and never completely remove it. Practical applications require the computed result in the affine domain. The implication is that there is at least one inversion operation for every terminal computation in the projective coordinate system. Using: A, M, S, I to represent field Addition, Multiplication, and Squaring and Inversion operations respectively the computational complexity costs of executing (2), (3) and (4) are as presented in table 1.

TABLE 1 COMPARATIVE COMPUTATIONAL COMPLEXITY

EC Operation	Equation (2) in affine coordinates system	Equation (3) in projective coordinate	Equation (4) in affine coordinate
Addition	6A+3M+1I	6A+23M+1I	7A+12M+1I
Doubling	4A+4M+1I	4A+13M+1I	4A+11M+1I

Considering multiplication operation only the proposed modification is efficient. The algorithm we review in the next section is based on (4).

III. THE MULTIPLY-BY-7 SCALAR MULTIPLICATION ALGORITHM

As already enunciated there are several novel schemes [3], [6] for computing EC scalar multiplication operation. The work [14] in particular used special addition chains. The multiply-by-7 algorithm repeatedly multiply a random point on an EC by 7 and adds or subtracts 1, 2 or 3 multiples of the previous point. The exact multiple is determined from a Restricted radix-7 Symmetrical quaternary Signed digit (Rr7SqSd) representation [15] of the scalar quantity. Hence, we shall first review the Rr7SqSd number system and its addition/subtraction chain concept before introducing the algorithm.

A. Definition of the Rr7SqSd number system and its addition-subtraction chain

Definition 1: The Rr7SqSd number system is a radix-7 number system with its character digits 0,1,2,3,4,5,6 recoded in the symmetrical quaternary signed character digits $\bar{3},\bar{2},\bar{1},0,1,2,3$.

The overlay $\bar{i} = -i$ and the recoding are obtained by reducing all the radix-7 character digits with a restrictor $a = 3$. Table 2 shows the radix-7 character digit-Rr7SqSd-binary equivalents.

TABLE 2 RADIX-7 Rr7SqSd BINARY EQUIVALENTS

Radix-7	4	5	6	0	1	2	3
Rr7SqSd	-3	-2	-1	0	1	2	3
Binary	100	101	110	000	001	010	011

Arithmetic operations in the Rr7SqSd number system can be described by (5)

$$\left.\begin{array}{l} \delta_i = \begin{cases} \bar{1} & ; \text{if } (\alpha i \,\Theta\, \beta_i) < \bar{a} \\ 1; & \text{if } (\alpha i \,\Theta\, \beta_i) > a \\ 0 & ; \text{otherwise} \end{cases} \\ \tau_i = (\alpha_i \,\Theta\, \beta_i) - 7\delta_i \end{array}\right\} \quad (5)$$

where: the operator $\Theta \in \{+,-,*,/\}$ and $\alpha_i, \beta_i, \tau_i \in \{\bar{3},\bar{2},\bar{1},0,1,2,\}$. Decimal numbers are first converted to radix-7 before using the addition option of (5) to convert the radix-7 character digits to Rr7SqSd number system. Conversion from Rr7SqSd to decimal can be carried out using (6)

$$X = \sum_{i=n-1}^{0} \alpha_i 7^i \quad (6)$$

Where X is the decimal equivalence of the Rr7SqSd string $\alpha = \alpha_{n-1}\alpha_{n-2}.....\alpha_1\alpha_0$

The Rr7SqSd number system similar to the binary number system supports addition-subtraction chain representation [7] of an integer. A binary addition-subtraction chain of integer W is a sequence of integers $a_0\ a_1\ a_2 \ldots a_r$ where the starting value $a_0 = \pm 1$ and the ending term $a_r = w$ such that any $a_k = a_i + a_j$ for $0 < i, j < k$. For example one form of the integer $w = 23$ addition-subtraction chain is $\pm 1,2,4,5,7,11,18,23$. In a similar manner we define the Rr7SqSd addition-subtraction as follows.

Definition 2: If $\alpha = \alpha_{n-1}\alpha_{n-2}.....\alpha_1.\alpha_0$ is the Rr7SqSd representation of any integer w. then an Rr7SqSd addition-subtraction chain of w is the sequence of integers $w_0\ w_1\ w_2 \ldots\ldots w_r$, starting from $w_0 \in (1,2,3)$ and ending with $w_r = w$ such that any w_k is the sum or difference of the previous integer term w_{k-1} multiplied by 7 and the k^{th} character symbol $\alpha_k \in \{\bar{3},\bar{2},\bar{1},0,1,2,3\}$ of the integer $w's$ Rr7SqSd representation

$$w_k = 7w_{k-1} + \alpha_k \quad (7)$$

For example, the Rr7SqSd representation of $w = 1237$ is $1\bar{3}\bar{3}2\bar{2}$ and its Rr7SqSd addition-subtraction chain is 1 4 25 177 1237. An addition-subtraction chain is an algorithm for computing mP given the integer m and the random point P on an EC. Hence, we now present the multiply-by-7 EC scalar multiplication algorithm.

B. Multiply- by-7 EC scalar multiplications

Detailed treatment of the algorithm we review below can be found in [16]. The algorithm can be applied to both EC defined over both binary and prime fields however, in this work discussion is limited to prime fields only. Let E, denote an EC defined over the prime field $GF(p^k)$ together with the point O, at infinity such that

$$E: \quad y^2 \bmod p^k = x^3 + ax + b \bmod p^k \quad ; \quad 4a^3 + 27b^2 \bmod p^k \neq 0 \qquad (8)$$

If $P(x_G, y_G)$ be a generator point of E and $m, (m >> 1)$ a scalar quantity. Let the string $\alpha_{n-1}, \alpha_{n-2}, .., \alpha_j, ..., \alpha_1, \alpha_0$ with $\alpha_{n-1} \in \{1,2,3\}$ and $\alpha_j \in (\bar{3}, \bar{2}, \bar{1}, 0, 1, 2, 3)$, $n-1 < j \leq 0$ be the Rr7SqSd representation of m. Now if the elements of the sequence $W \in \{W_{n-1}, W_{n-2}, .., W_j .., W_1, W_0\}$, $W_j = P(x_j, y_j)$ are points on E, obtained by using (9)

$$W_j = \begin{cases} W_{n-1} = \alpha_{n-1} P(x_G, y_G) \text{ ; if } j = n-1 \\ \\ 7W_{j-1} + \alpha_j W_{j-1} \text{ ; if } n-2 \leq j \leq 0 \end{cases} \qquad (9)$$

Then, (9) computes the consecutive points of the EC scalar multiplication operation $W = mP$ by way of several one-stop or direct $7P$ computations We call (9) in this contribution the multiply-by-7 EC scalar multiplication algorithm. Obviously, (9): i) effectively reduce the number of inversions from $m-1$ to $j; (j >> m)$. ii) Decomposes the art of computing EC $W = mP$ arithmetic operation to several one-stop $7P$ computations. Fig. 1 shows the procedure for computing (9) using (4). First, depending on the most significant digit $\alpha_{n-1} \in \{1,2,3\}$ the EC generator point $P(x_G, y_G)$ is simply taken as the start point i.e., $Q = P(x_G, y_G)$ or doubled $Q = 2P(x_G, y_G)$ or tripled $Q = 3P(x_G, y_G)$ to become W_{n-1} or W_{j-1}. These pre computations processes are designated $1P_{SEL}$, $2P_{SEL}$ and $3P_{SEL}$ respectively in fig. 1. Their internal and external out variables are designated $\phi_i \in \{A, B, C, D, E, F, H\}, i \in \{0,1,2\}$ in the top square of fig. 1. This value of Q $(= W_{j-1})$ is separately multiplied by 7 and by $\alpha_j \in \{\bar{3}, \bar{2}, \bar{1}, 0, 1, 2, 3\}$ in the EC domain. The sum or difference of the two products $7W_{j-1}$ and $\alpha_j W_{n-1}$ represents the j^{th} iteration of the multiply-by-7 EC scalar multiplication operation. This sum/difference is field inverted to extract the x_w, y_w components which is the new generator

$1P_{sel}$ module,

$$A_0 = 0 \quad B_0 = X_G \quad C_0 = 1 \quad E_0 = Y_G \quad X_{UD} = B_0 \quad Y_{UD} = E_0 \quad W_{UD} = C_0$$

$2P_{sel}$ module

$$A_1 = 3X_G^2 + A \quad B_1 = A_1^2 - 8X_G Y_G^2 \quad C_1 = 2Y_G \quad D_1 = X_G C_1^{\,2} - B_1 \quad F_1 = X_G C_1^{\,2} + B_1 \quad E_1 = A_1 D_1 - 8Y_G \quad X_{UD} = B_1 \quad Y_{UD} = E_1$$

$3P_{sel}$ module

$$A_2 = Y_G C_1^3 - E_1 \quad B_2 = A_2^2 - F_1 D_1^2 \quad C_2 = C_1 D_1 \quad H_2 = B_1 D_1^{\,2} - B_2 \quad E_2 = A_2 H_2 - E_1 D_1 \quad X_{UD} = B_2 \; ; \quad Y_{UD} = E_2 \; ; \; W_{UD} = C_2 \quad W_{UD} = C_1$$

$$A_{17} = 3X_{17P}^2 + A \quad B_{17} = A_{17}^2 - 8X_{17P} Y_{17P}^2 \quad C_{17} = 2Y_{17P} \quad D_{17} = X_{17P} C_{17}^2 - B_{17} \quad F_{17} = X_{17P} C_{17}^2 + B_{17} \quad E_{17} = A_{17} D_{17} - 8Y_{17P}^4$$

$$A_{77} = E_{27} C_{37}^3 - B_{37} C_{27}^3 \quad B_{77} = A_{77}^2 - F_{37} D_{37}^2 \quad C_{77} = C_{27} C_{37} D_{37} \quad H_{77} = B_{37} (C_{27} D_{37})^2 - B_{77} \quad E_{77} = A_{77} H_{77} - E_{37} (C_{27} D_{37})^3$$

$$A_{27} = Y_{17_P} C_{17}^{\,3} - E_{17} \quad B_{27} = A_{27}^{\,2} - F_{17} D_{17}^{\,2} \quad C_{27} = C_{17} D_{17} \quad D_{27} = X_{17P} C_{27}^{\,2} - B_{27} \quad F_{27} = X_{17_P} C_{27}^{\,2} + B_{27} \quad H_{27} = B_{17} D_{17}^{\,2} - B_{27} \quad E_{27} = A_{27} H_{27} - E_{17} D_{17}^{\,3}$$

$$A_{37} = Y_{17P} C_{27}^3 - E_{27} \quad B_{37} = A_{37}^2 - F_{27} D_{27}^2 \quad C_{37} = C_{17} D_{17} D_{27}^2 \quad D_{37} = B_{27} C_{37}^2 - B_{37} C_{27}^2 \quad F_{37} = B_{27} C_{37}^2 + B_{37} C_{27}^2 \quad H_{37} = B_{27} D_{27}^2 - B_{37} \quad E_{37} = A_{37} H_{37} - E_{27} D_{27}^2$$

$$A_{UD} = E_{77} W_{UD}^3 - Y_{UD} C_{77}^3 \quad D_{UD} = B_{77} W_{UD}^2 - X_{UD} C_{77}^2 \quad F_{UD} = B_{77} W_{UD}^2 + X_{UD} C_{UD}^2 \quad B_{UD} = A_{UD}^2 - F_{UD} D_{UD}^2 \quad C_{UD} = C_{77} W_{UD} D_{UD} \quad H_{UD} = X_{UD} (C_{77} D_{UD})^2 - B_{UD} \quad E_{UD} = A_{UD} H_{UD} - Y_{UD} (C_{77} D_{UD})^3$$

$$x_W = \frac{B_{UD}}{C_{Ud}^2} \; , \; y_W = \frac{E_{UD}}{C_{UD}^3}$$

Figure.1 Multiply-by-7 EC scalar multiplication algorithm.

point for the $(j+1)^{th}$ iteration. Multiplying W_{j-1} by a negative value of $\alpha_j \in \{-3,-2,-1\}$ is merely a tripling, doubling or just $1P$ EC operation on $-P(x_{j-1}, y_{j-1})$ which is equal to $\alpha_j P(x_{j-1}, -y_{j-1})$.

The manner of realizing the one-stop or direct 7P computation is crucial to the scalar multiplication operation speed. To avoid the traditional method of 6 repetitive EC point addition operations which in turn requires 6 field inversion operations the operation is executed with one field inversion by using four sub processes: $2P = P + P$, $3P = 2P + P$, $4P = 3P + P$ and $7P = 3P + 4P$. The internal and external outputs of these four sub processes are similarly designated by the elements of the set $\varphi_{17} \in \{A, B, C, D, E, F, H\}$ where $i \in \{1,2,3,7\}$ correspond to $2P$, $3P$, $4P$ and the $7P$. They are the contents of the 3 left hand and first right hand rectangles that are below the top square.. Internal and external variables of the $7W_{j-1} + \alpha_j W_{j-1}$ computation (or update) are also designated $A_{UD}, D_{UD}, F_{UD}, B_{UD}, C_{UD}, H_{UD}, E_{UD}$ and presented in the rectangle directly below the φ_{77} sub process of fig. 1.

The following numerical example illustrates the working of fig. 1. Consider the EC E described by the equation $y^2 = x^3 - 53x + 218 \bmod 17011$ and let $P(8360,14564)$ be a generator point. Given the scalar quantity $m = 16299 \equiv 10\bar{1}\bar{3}\bar{3}3_{Rr7SqSd}$. The result of an EC scalar multiplication operation $W = mP$ using the multiply-by-7 algorithm would yield the 6 curve points:
$1P = (8360,14564), 7P = (13509,216),$
$48P = (12536,6665), 333P = (10407,4280),$
$2328P = (14924,5526), 16299P = (1594,2138).$
The algorithm execution time with Quick Basic version 4.5 on a Pentium III processor for values of m up to 640 decimal digits took 0.087 seconds.

Equation (9) was also realized in the Jacobian projective coordinate and in the Near Adjacent Form (NAF) projective coordinate domains using equation (3). The computational complexities for the NAF, Jocabian and the multiply-by-7 options are respectively: $34A + 94M + 1I$, $38A + 110M + 1I$ and $36A + 87M + 1I$. Thus, there is a relative 27% and 8% reduction in computational complexity over the Jacobian and the NAF respectively by the multiply-by-7 algorithm. In the next section we present the concurrent realization model of this algorithm and estimate a parallel mode-versus- sequential mode speed enhancement figure of merit ratio ξ.

IV CONCURRENT OPERATION MODEL OF THE MULTIPLY-BY-7 ALGORITHM

Concurrency is all about performing more than one operation simultaneously provided resources are available to extract optimal performance from a system. In order to balance devices chip areas and system speed it is necessary to determine the most appropriate number of resources that can run concurrently. Dependency graphs are usually employed to identify concurrency in a given computational processes and determine the optimal number of resources. To achieve this objective the operation processes are scheduled using popular scheduling algorithms such as: the As Soon As Possible (ASAP) and As Late As Possible (ALAP) scheduling algorithms described in [10], [9], [17] and [18]. In this work we adopted the method outlined in [10] to identify those statements in fig.1 that can be performed simultaneously. These are grouped together to form a batch so that the multiply-by-7 scalar multiplication operation is executed in batches. We judge the system performance by establishing the number of field multiplication operations in a batch and the total number of batches necessary to execute the entire algorithm. It is necessary to note that in this context a batch is a multiplication cycle. The time to execute a particular field multiplication operation is distinct hence, in a batch the longest execution time constitutes the batch processing time. This constituted our concurrent model of fig. 1. The data flow graph for the $2P$ sub process is given in appendix A as an example.

Our concurrent processing model of the multiply-by-7 scalar multiplication algorithm based on the ASAP scheduling principle consist of eight data flow graphs corresponding to the eight sub processes $1P_{SEL}$, $2P_{SEL}$, $3P_{SEL}$, $2P$, $3P$, $4P$, $7P$ and the $7W_{j-1} + \alpha_j W_{j-1}$. The computation flow of the graphs follows pipeline processing. Input variables of a i^{th} sub process are output variables of $(i-1)^{th}$ sub process. Thus, the input variables for $2P$ sub process are the outputs of either $1P_{SEL}$, $2P_{SEL}$ or $3P_{SEL}$ in the start-value selection stage. The $2P$ output variables: $C_{17}^3, E_{17}, F_{17}, D_{17}^2, B_{17}, D_{17}, C_{17}^2$ and C_{17} are part of the input values to the $3P$ sub process. These outputs in turn are the inputs for $4P$ sub process. Analyses of the entire sets of graphs provide an insight to the concurrent processing ability of the algorithm. Furthermore since a batch is essentially a multiplication cycle the total number of batches constitutes the number of multiplication cycles m_c of a given sub process.

The number of field operations n_{fo} in a batch is the number of hardware units that run simultaneously in that batch. Thus the number of hardware units required to execute a sub process equals the highest number of field operations $n_{fo\max}$ among the batches of the sub process. That is $n_{fo\max} = \max(n_{fo1}, n_{fo2}, \ldots n_{foh})$. We ignored the number of addition and subtraction operation as well as their durations in our models.

Similar models (8 data flow graphs each) were developed for the Jacobian and NAF option of realizing equation (9). Up to 7 field multiplication operations could run simultaneously in the $7P = 3P + 4P$ sub process of the NAF realization option. The information extracted from the models are:

i) Number of multiplication cycles per sub process.
ii) Total number of field multiplication operations per sub process per realization as the number of multipliers running concurrently increases from 1 to 7.
iii) The highest number of multipliers in one multiplication cycle per sub process per realization

The corresponding Area-Time squared (AT^2) values per sub process for each realization were computed. The expression $T_j = \sum_{i=1}^{h} t_{ji}$ where t - represent multiplication cycle duration of the j^{th} sub process evaluates the speed algorithm.

The trend in today's ITCS is for the system to be as small as possible. This has made device chip area (A) also as a factor of merit. Hence, in design works either area A, or time T is considered more important (sometimes A and T are considered equally important). Thus, the products metric: AT, A^2T or AT^2 are often used to judge operation parallelism. We employed the AT^2 metric. This automatically made the chip area A common denominator.

. One objective of this section is to determine the best method of parallelizing the scalar multiplication process of (9). Equation (2) based Jacobian, NAF realizations or a (4) based method of implementing (9). The relevant parameters for this decision are the: i) lowest number of multiplications in the sequential mode. ii) Best AT^2 value as the number of resources increases from 1 (sequential) to 7 (parallel mode). iii) Highest speed enhancement as a result of the concurrent realization. iv) The total number of multipliers in parallel to provide the figures of merit in the various implementation options. . It must be noted also that the grand total number of (multiplication cycles for the Jacobian, NAF and the proposed algorithm are 37,34 and 29 respectively which shows that the proposed algorithm has the smallest number of field multiplication operation cycles. The speed enhancement ratio ξ_i was determined using (10).

$$\xi_i = \frac{\sum_{i,j=1}^{1,7}\left(AT_j^2\right) - \sum_{i=2,j=1}^{7,7}\left(AT_j^2\right)}{\sum_{i=2,j=1}^{7,7}\left(AT_j^2\right)} \qquad (10)$$

Where i - is the number of multipliers in parallel and j - a subscript representing any of the sub processes: $1P_{SEL}, 2P_{SEL}, 3P_{SEL}, 2P, 3P, 4P, 7P$ and Ugp.

V DISCUSSION

Figures 2 ,3 and 4 show the graphical presentation of some major information from the various realizations. The bar chart plots of fig. 2 represent total number of field multiplication operations as the number of multipliers increases from 1 to 7. The plots in groups of three represent the Jacobian, NAF and the algorithms options of realization. All plots as expected are exponential decays as the resources on line increase. The
proposed method presents the smallest total number of field multiplication operations (87M) required to execute iteration. Similarly, fig. 3 shows the plot of AT^2 values for increasing number of resources in parallel The Jacobian and NAF approaches presented their best AT^2 values at 5 multipliers working concurrently. Our proposal do not only presents the lowest AT^2 value but also at 4 multipliers operating in parallel.

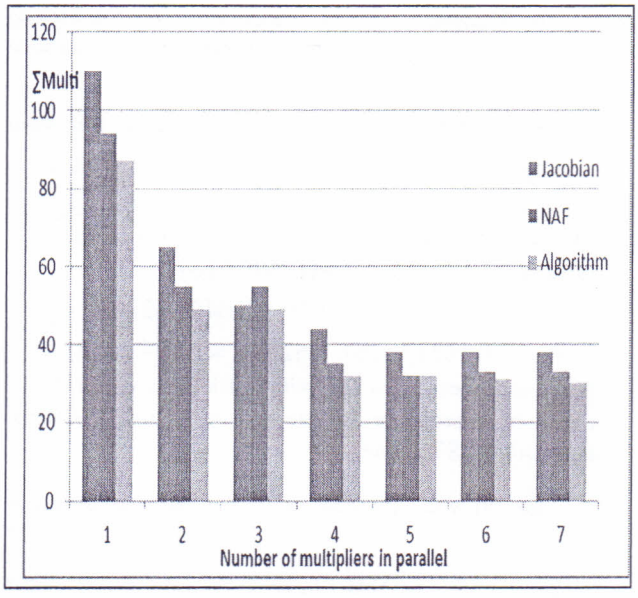


Figure. 2 Total number of field multiplication operations per realization versus number of multipliers

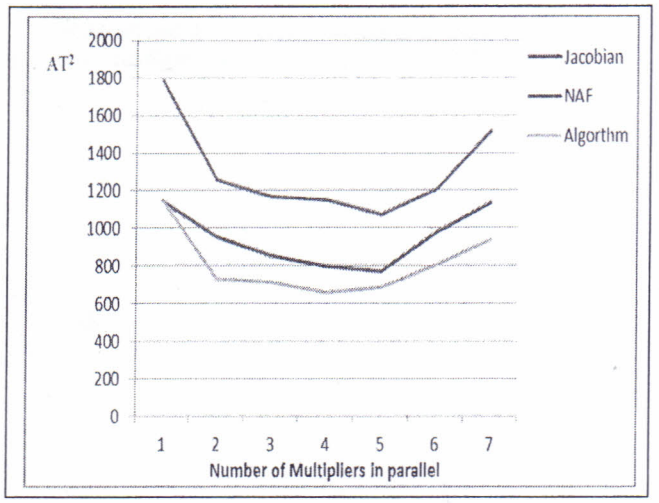


Figure. 3 AT^2 Values as a function of number of resources in parallel.

Finally the plot of fig. 4 shows the percentage speed enhancements as a result of concurrent processing of scalar multiplication operation. The NAF approach presents the lowest speed enhancement percentage while the proposed algorithm presents the highest speed enhancement.

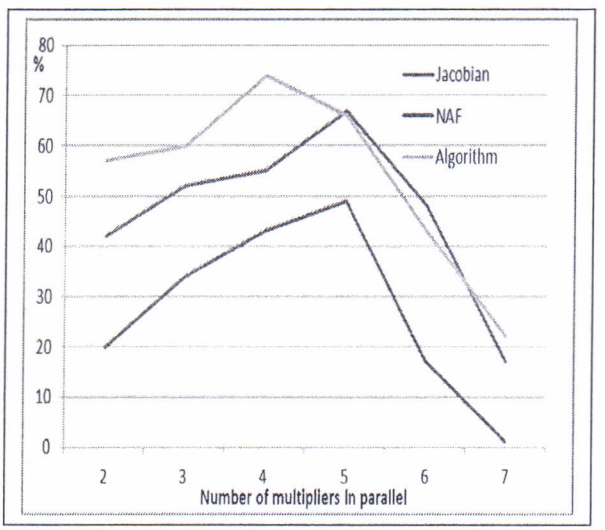


Figure. 4 Percentage speed enhancement over sequential regime.

The speed enhancements for the Jacobian and NAF approaches peaked at 5 multipliers working concurrently. In the related work of [19], the authors chose the standard projective coordinate as the appropriate efficient choice for parallel designs and implemented their parallel architecture. This was as a result of the corresponding speed enhancement peaking at 4 multipliers. The proposed algorithm speed enhancement also peaked at 4 multipliers working in parallel. This simply indicates that the approach is efficient and that it provides a 25% saving in hardware over Jacobian and NAF styled implementations.

VI Conclusion and Further Work

This investigation has established that an Rr7SqSd addition/subtraction chain based multiply-by-7 EC scalar multiplication scheme if concurrently realized can result in reduced computational complexity and high parallel operation execution speed. The approach is useful in the design of multiple-valued logic elliptic curve cryptosystem.

References

[1] Kefa Rabah ; "Elliptic curve cryptography over binary finite fields GF(2m)" Asian network for Scientific Information. ISSN 1812-5638. Information Technology Journal 5(i) :pp 204 -229, 2006.

[2] David Malan, "Crypto for tiny objects", Computer Science group , Harvard University Cambridge, Massachusetts , 2004. http://www.cs.havard-edu/malan/publication/tr-04-04.pdf..

[3] Yasuyuki Sakai and Kouichi Sakurai " Efficient scalar multiplication with direct computation of several Doublings" :IEICE Trans. Fundamentals, Vol. E84-A N01 January 2001 pp. 120-128

[4] "An introduction to Elliptic Curve Cryptography-Deviceforge.com ....tomorrow's device technology today," http://www.DeviceForge.com/articles & white papers /AT4234154468, July 20, 2004.

[5] M. S. Anoop, "Elliptic Curve Cryptography - An Implementation guide". Online implementation Tutorial, 2007. http://www.tataelxsi.cer/whitepapers/ECC_Tut_v1_0.pdf id

[6] Dimitrov,V.S.; Imbert, L.; Mishra,P.K.; "Fast Elliptic curve point multiplication using Double-base Chains"; University of Calgary, 2500 University drive NW Calgary AB, T2n In4, Canada 2005 pp1-13 http://eprint.iacr.org/205/069.ps.

[7] Cetin Kaya Koc "Elliptic curve cryptosystem" Department of Electrical and Computer engineering, Oregon State University, Oregon 97331, 2004.

[8] Jun-Hong Chen, Ming-Der Sheih and Chien-Ming Wu, "Concurrent algorithm got high-speed point multiplication in elliptic curve cryptography," 0-7803-8834-8/05$20.00, IEEE 2005, pp. 5254 – 5257.

[9] Hakim Khali and Ahcene Farah," Cost effective implementations of GF (p) elliptic curve cryptography computations," IJCSNS International journal of computer science and network security, Vol. 7 N0. 8, August 2007, pp. 29 -37.

[10] Adnan Abdul-Aziz Gutub, "Remodeling of elliptic curve cryptography scalar multiplication architecture using parallel Jacobian coordinate system," International Journal for computer science and security (IJCSS), Vol. 4: Issue 4, 2009, pp. 409 -425.

[11] "Certicom ECC Challenges". Ecc.project@edu.informatik.tu-darmstadt.de. 2003. http://www.certicom.com/index.php/the-certicom-ecc-challenge.

[12] Avikak "Lecture notes on 'Introduction to computer security". Avinagh Kak Purdue University 2010. http://cobweb.ecn.purdue.edu/~kak/compsec/newLectures/lecture14.pdf.

[13] Igbal H. Jubril, Rus Salleh and Al-Shawabkeh M. "Efficient algorithm in projective coordinates for ECC over ", Internal Journal of Computer, internet and management. Vol. 15 N0 1, April 2007, pp., 43 – 50.

[14] Nicolas Meloni "fast and secure EC Scalar multiplication over prime fields using special Addition chains"; Institute de Mathematique et de modisatic de Montpeller, UmR 5149, Montpeller France. Citeseerx.ist.psu.edu/viewdoc/download?doi=10.61.5861[1] pdf.

[15] Michael Naseimo Daikpor and Oluwole Adegbenro, " Efficient carry-free Rr7SqSd addition algorithm," Proceedings of the 3rd International

conference on emerging trends research direction and training requirements of the 21st century Electrical and Electronics Engineering, Lagos, Nigeria, 22nd – 24th July 2009, pp. 102 – 107.

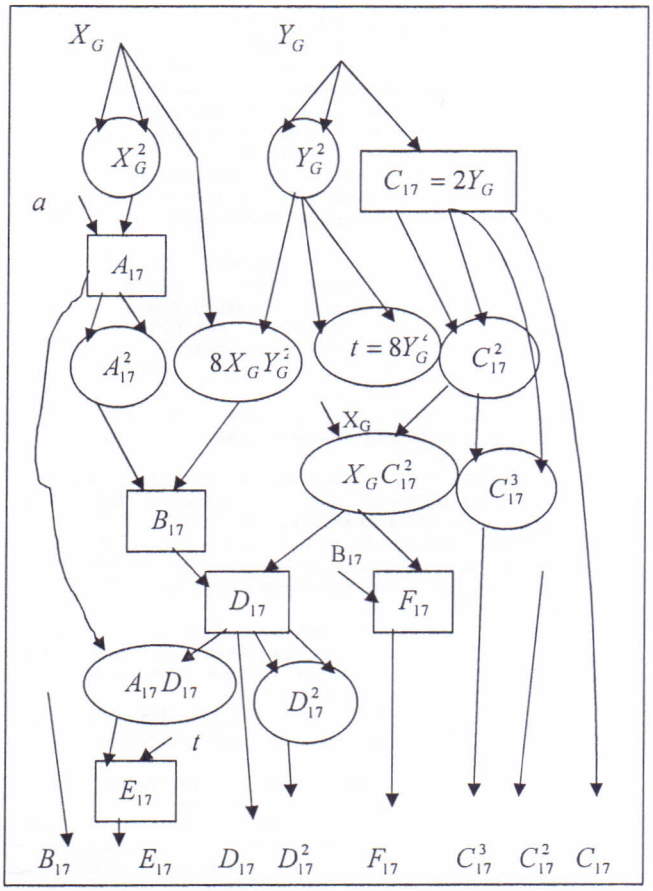
[16] Michael Naseimo Daikpor and Oluwole Adegbenro," Elliptic curve scalar multiplication with direct 7P computation for secure communication systems, " Proceedings of the International conference on innovations in Engineering and Technology (IET 2011)- Sustaining 21st century Engineering infrastructure, Lagos- Nigeria 8th – 10th August 2011, pp. 179 -189.

[17] Turki F. Al-Somani, M. K. Ibrahim and Adnan Gutub, "Highly efficient elliptic curve crypto-processor with parallel GF (2m) field multiplier," Journal for Computer Science 2 (5): 2006, ISSN 1549-3636, pp. 395 - 400.

[18] Robert A Walker, "Introduction to scheduling problem," IEEE Design and Test of Computers, 1995, pp. 60- 69.

[19] A. Gutub and M.K Ibrahim, "High radix parallel architecture for GF (p) elliptic curve processor," IEEE Conference on acoustic, speech and signal processing – ICASSP 2003, pages 625 – 628, Hong-Kong, April 6 -10, 2003.

Appendix A



i) Example of the $2P$ sub process data flow graph.

Notes to the appendix A:

i) The rectangular shapes represent field addition or subtraction operations while the circular and elliptical shapes represent field multiplication operations.

ii) Number of multiplication cycles is 4 and the highest number of field multipliers that can run simultaneously is 4.