New Hardware Implementations of WG(29, 11) and WG-16 Stream Ciphers Using Polynomial Basis

Hayssam El-Razouk, Arash Reyhani-Masoleh, *Member, IEEE*, and Guang Gong, *Fellow, IEEE*

Abstract—The WG stream ciphers are based on the WG (Welch-Gong) transformation and possess proved randomness properties. In this paper we propose nine new hardware designs for the two classes of WG(29,11) and WG-16. For each class, we design and implement three versions of standard, pipelined and serial. For the first time, we use the polynomial basis (PB) representation to design and implement the WG(29,11) and WG-16. We consider traditional PB multiplier for the WG(29,11), and, the traditional and Karatsuba multipliers for the WG-16. For efficient field operations, we propose an irreducible trinomial for the WG(29,11). For the WG-16, a new formulation of its permutation which requires only 8 multipliers is introduced. In these designs, the multipliers in the transforms are further reduced by utilizing a novel computation for the trace of the multiplication of two field elements. We have implemented the proposed designs in ASIC using CMOS 65 nm technology. The results show that the proposed standard WG(29,11) consumes less area and slightly enhances the normalized throughput, compared to the existing counterparts. For the WG-16, throughput of the proposed pipelined instance outperforms the previous designs. Moreover, the speed of the proposed WG-16 designs meet the peak bit rates for the 4 G specifications.

Index Terms—Finite fields, linear feedback shift registers, polynomial basis, pseudo random key generators, stream ciphers, trace function, WG transformation

1 Introduction

A stream cipher is a symmetric key crypto-system which generates a unique key-stream for each secret key, where the plaintext (or ciphertext) is bitwise XORed with the generated key to produce the ciphertext (or plaintext). Stream ciphers are used in different communication applications, such as, RFID tags [1], bluetooth [2], network protocols (SSL, TLS, WEP and WPA) [3], and 3GPP long term evolution (LTE) security suite [4], [5].

The eSTREAM project [6] is the most significant effort for finding secure stream ciphers [7]. The WG$(29, 11)$ [8] is a stream cipher submitted to the hardware profile of phase 2 of this project. The WG$(29, 11)$ offers the proved randomness properties of the WG family of ciphers [8], [9], [10], [11]. The two attacks [12], [13] were launched on WG$(29, 11)$ during this project. However, it is noted that the revised version of the cipher [10] does not suffer the chosen IV (Initial Value) attack in [12], [14]. Also, as per design, the number of key-stream bits per a single key/IV pair is strictly less than the number of key-stream bits required to perform a linear span attack introduced in [13], [10]. In the literature, there are many proposed WG$(29, 11)$ hardware designs [7], [8], [10], [15], [16]. The original submission uses normal basis (NB) representation [8] and hence all of presented designs until now have used the NB representation [8], [10], [16]. Among them, the most optimal design is based on the Type-II optimal normal basis (ONB), which is presented in [16]. Using the novel trace property presented by the authors of [16], their design requires only six field multipliers. In this paper, for the first time we consider PB representation in the WG stream ciphers. We propose a novel method for computing the trace of the multiplication of two field elements represented in the PB. It is noted that the proposed trace method is applicable to any $GF(2^m)$, while the one presented in [16] only applies to fields where self-dual bases exist. Based on this trace method, we present a PB-based hardware design of the WG$(29, 11)$, which uses six multipliers. Also, pipelined and serialized instances of this standard design are presented (see Fig. 1). The reported results for the 65nm CMOS ASIC realization of the proposed standard WG$(29, 11)$ design shows smaller area and, slightly improved normalized throughput, compared to the best result presented in [16].

Another initiative for designing secure stream ciphers is the LTE mobile technology. LTE is being established as the fourth generation (4G) mobile technology, where a flat all internet protocol (IP) infrastructure has been adopted [17]. This has changed the threat model of the 4G mobile domain to include the security issues which are applied to the IP networks [17]. Accordingly, there is a continuous effort demonstrated by the security specification group of the third generation partnership project (3GPP-TSG) [18] to address these security threats [17]. The cipher suite of 4G LTE consists of two stream ciphers, SNOW 3G and ZUC,

- *H. El-Razouk and A. Reyhani-Masoleh are with the Department of Electrical and Computer Engineering, Western University, London, ON, Canada. E-mail: {helrazou, areyhani}@uwo.ca.*
- *G. Gong is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. E-mail: ggong@uwaterloo.ca.*

Manuscript received 18 Oct. 2013; revised 8 Apr. 2014; accepted 16 June 2014. Date of publication 6 Aug. 2014; date of current version 10 June 2015. Recommended for acceptance by J.-M. Muller.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2346207

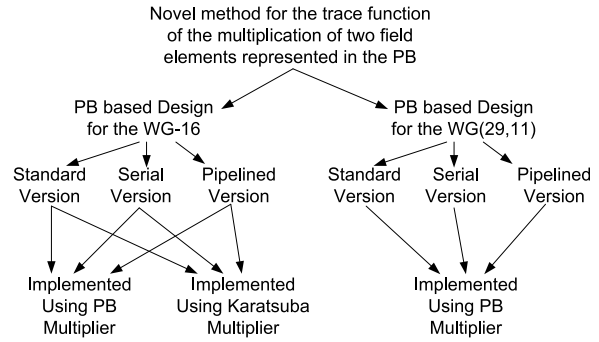

Fig. 1. Contributions of this work.

and the block cipher AES in the counter mode [4], [5]. It is noted that the randomness of the key-streams generated by the 4G LTE cryptographic algorithms is hard to analyze and, more importantly, some weaknesses concerning these ciphers have already been discovered [19], [20]. Furthermore, some security flaws in the LTE integrity protocols have been recently recognized [21]. The authors of [17] propose confidentiality and integrity protection schemes for securing the 4G network domain against the attack in [21]. These schemes are based on the WG-16 stream cipher. The WG-16 offers the proved randomness properties of the WG family of ciphers [17]. In addition, it is secure and resists to all known attacks [17]. The only WG-16 hardware design, which uses NB, is presented in [22]. This design is based on composite field arithmetic and properties of the trace function in the tower field representation. In this paper, we propose a new formulation of the WG-16 permutation which requires 8 multiplications compared to 10 in the formulation of [22]. Furthermore, we have derived a new formulation for the trace function of the multiplication of two field elements, based on which a PB-based WG-16 design is proposed using only six multipliers for its transform. Also, pipelined and serialized versions of this standard design, are presented and for each design both the traditional PB and Karatsuba multipliers are considered (see Fig. 1). According to our ASIC (CMOS 65 nm) implementations, the proposed pipelined instance of the WG-16 offers double the throughput, while it slightly reduces the area, compared to the results reported in [22].

WG ciphers provided guaranteed randomness properties such as long period, balanced 0-1 distribution, ideal tuple distribution, exact linear complexity, and delta like autocorrelation functions, for which no other existing ciphers could provide [9], [10]. The goal of this paper is to show hardware implementations for WG ciphers, which in return, provides trade-offs between randomness properties and performance for a selection of ciphers for a particular application. In particular, it is shown that the proposed WG-16 implementations comply with the throughput requirements of the 4G domain. The contributions of this paper which include a novel trace method and nine new designs of the WG stream ciphers are summarized in Fig. 1. In this figure, the standard WG$(29, 11)$ implementation shows lower space and slightly improved normalized throughput, compared to the one in [16]. Also, the pipelined instance of the proposed WG-16 reports higher throughput and lower area compared to the corresponding ones in [22].

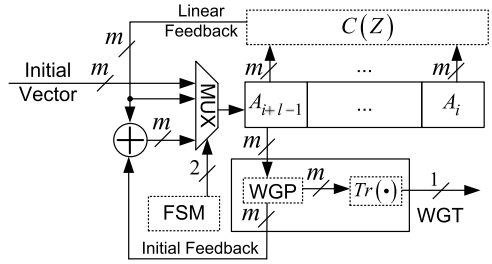

Fig. 2. General block diagram for the WG stream ciphers.

The paper is organized as follows. Section 2 defines the terms, notations, and gives brief background about the WG$(29, 11)$ and WG-16. Section 3 presents the proposed WG$(29, 11)$ hardware designs based on the PB. Section 4 presents the proposed WG-16 hardware designs based on the PB. Results based on ASIC implementations are discussed in Section 5. Section 6 concludes the paper.

2 Preliminaries

The following are notations used throughout this paper to describe the architectures and operations of the WG$(29, 11)$ and WG-16 ciphers.

- $GF(2)$, binary finite field with elements $\{0, 1\}$.
- $GF(2^m)$, binary extension field with 2^m elements. $\mathbf{a} = (a_0, a_1, \ldots, a_{m-1})$ denotes the m-bit row vector representation of $A \in GF(2^m)$.
- $\oplus$ represents the addition operator in $GF(2^m)$.
- $Tr(Z) = \sum_{i=0}^{m-1} Z^{2^i}$, $Z \in GF(2^m)$, the trace function from $GF(2^m) \longrightarrow GF(2)$.
- Let $p(x)$ be an irreducible polynomial of degree m over $GF(2)$ and let $p(\alpha) = 0$, then $\{1, \alpha, \ldots, \alpha^{m-1}\}$ is a polynomial basis of $GF(2^m)$ over $GF(2)$.
- The inner product of the vectors $\mathbf{a} = (a_0, a_1, \ldots, a_{m-1})$ and $\mathbf{b} = (b_0, b_1, \ldots, b_{m-1})$, $a_i, b_i \in \{0, 1\}$, $0 \leq i < m$, is computed as $\mathbf{ab}^T = \sum_{i=0}^{m-1} a_i b_i \in \{0, 1\}$, where T denotes transposition.
- $C(Z) = Z^l \oplus \sum_{i=0}^{l-1} C_i Z^i$, $C_i \in GF(2^m)$ is the characteristic polynomial of an l-stages LFSR over $GF(2^m)$, from which the feedback recurrence relation can be derived as $A_{j+l} = \sum_{i=0}^{l-1} C_i A_{i+j}$, where $j \geq 0$, $A_i \in GF(2^m)$, and $(A_0, A_1, \ldots, A_{l-1})$ is the initial state of the LFSR.

2.1 Components, Operation, and Parameters of the WG$(29, 11)$ and WG-16 Stream Ciphers

2.1.1 Components and Operation

The WG$(29, 11)$ and WG-16 are bit-oriented filter generators. The sequence generator consists of an orthogonal m-bit WG transform which is applied to the leftmost cell of a primitive LFSR of degree l over $GF(2^m)$. This can be seen in the block diagram of Fig. 2. For the WG$(29, 11)$ $m = 29$ and $l = 11$, while $m = 16$ and $l = 32$ for the WG-16. This construction generates m-sequences of period $2^{ml} - 1$ [10], [17]. The WG$(29, 11)$ and the WG-16 have three phases of operation, namely, loading phase (which requires l clock cycles, with Initial Vector applied to the LFSR's input), key initialization phase (which requires $2 \times l$ clock cycles, Linear

Feedback $\oplus$ Initial Feedback is the input to the LFSR in Fig. 2), and run phase which generates an output bit in each clock cycle (for this phase Linear Feedback is the input to the LFSR). As shown in Fig. 2, the finite state machine (FSM) controls the three phases of operations.

2.1.2 Parameters of the WG$(29, 11)$

The permutation for the WG$(29, 11)$ is

$$WGP29 = 1 \oplus Y \oplus Y^{2^{10}+1} \oplus Y^{2^{20}+2^{10}+1} \oplus Y^{2^{20}-2^{10}+1} \oplus Y^{2^{20}+2^{10}-1}, \quad (1)$$

where $Y = 1 \oplus A_{i+10}$ and A_{i+10} is the LFSR's output. The WG transform is given as follows [8], [10], [16], [23]

$$WGT29 = Tr(WGP29). \quad (2)$$

The reader is referred to Section 3.2 of this paper for the field and characteristic polynomials of the WG$(29, 11)$.

2.1.3 Parameters of the WG-16

The WG permutation is [17]

$$WGP16 = 1 \oplus Y \oplus Y^{2^{11}+1} \oplus Y^{2^{11}+2^6+1} \oplus Y^{-2^{11}+2^6+1} \oplus Y^{2^{11}+2^6-1}, \quad (3)$$

where $Y = (A_{i+31})^{1,057} \oplus 1$ and A_{i+31} is the output of the LFSR. In [22], $WGP16$ is computed as

$$1 \oplus Y \oplus Y^{2^{11}+1} \oplus Y^{2^{11}(2^{11}-1)+1} \oplus Y^{2^6}(Y^{2^{11}+1} \oplus Y^{2^{11}-1}), \quad (4)$$

where

$$Y^{2^{11}-1} = Y^{((1+2)(1+2^2)+2^4)(1+2^5)+2^{10}}.$$

It is noted that (4) requires 10 multiplications (including 2 for computing $(A_{i+31})^{1,057}$). The WG transform is $WGT16 = Tr(WGP16)$. The characteristic polynomial of the WG-16's LFSR[1] is [17]

$$Z^{32} \oplus Z^{31} \oplus Z^{22} \oplus Z^9 \oplus \omega^{11}, \quad (5)$$

which is primitive over $GF(2^{16})$, where ω is the root of the $GF(2^{16})$'s field polynomial

$$x^{16} + x^5 + x^3 + x^2 + 1. \quad (6)$$

3 Architectures of the WG$(29, 11)$ Stream Cipher

The WG$(29, 11)$ uses exponentiation over $GF(2^{29})$, and therefore, an ONB was assumed to be more efficient for hardware design, compared to other representations, due to the free cost of squaring operations [8], [10], [16]. The authors of [8] propose a direct design of the WG$(29, 11)$ based on ONB using seven multiplications and an inversion over $GF(2^{29})$, where the field inversion requires six multiplications and 28 squarings in $GF(2^{29})$ [24]. In [10], the authors reduce the field multiplications to only 9. The author of [15] proposes a design which uses modulo 2 computations and requires 2^{29} bits of ROM to store the precomputed WG transform sequence [15]. The authors of [16] utilize some properties of the trace function for type-II ONB in order to build the cipher using only six field multiplications.

In this paper, we propose three PB-based designs for the WG$(29, 11)$. These designs include a standard architecture, its serial version, and its pipelined version. The serial version is suitable for low-area applications whereas the pipelined one is proposed for high-speed applications. To the best of our knowledge, this is the first implementation of the WG cipher based on the PB representation. The parameters of the cipher are chosen carefully for a low area design. Also, for further area reduction, the proposed implementation uses properties of the trace function for PB in order to optimize the WG transform. The proposed scheme offers smaller area and a slightly higher normalized throughput, compared to the best results presented in [16], at the expense of a small decrease in the speed. In this section, first, the WG transform formulations are derived. This is followed by finding the design parameters. After that, the proposed architecture of the WG$(29, 11)$ is introduced.

3.1 Formulation of WGT29

Since replacing $(Y^{2^{20}-2^{10}+1})$ with $(Y^{2^{20}-2^{10}+1})^{2^{20}}$ in (1) does not affect $Tr(WGP29)$, therefore

$$WGT29 = Tr(1 \oplus Y \oplus Y(Y^{2^5})^{2^5}) + Tr(((Y^{2^5})^{2^5})^{2^{10}}(Y(Y^{2^5})^{2^5} \oplus Y^{2^{10}-1} \oplus (Y^{2^{10}-1})^{2^{30}})). \quad (7)$$

It is noted that (7) shows the order of computing the squarings in the transform. To reduce propagation delay due to squarings in the PB, we compute $Y^{2^{10}-1}$ as follows:

$$Y^{2^{10}-1} = (((Y^{2^5+1})^{2+1})(Y^{2^5+1})^{2^4})((Y^{2^5+1})^{2+1})^{2^2}. \quad (8)$$

The following section introduces the WG$(29, 11)$'s design parameters.

3.2 Design Parameters

This section presents the design parameters for the proposed PB implementation of the WG$(29, 11)$. In what follows, the field polynomial, the squaring matrices, the LFSR's characteristic polynomial, the trace vector, and the formulation for directly computing the trace of the multiplication of two field elements are presented.

3.2.1 Field Polynomial and Squaring Matrices

To compute (7) and (8), field multiplications and squarings are used. In the original design of the WG$(29, 11)$ [8] and all reported schemes to date [10], [16], NB representation is used. The squaring is obtained by cyclic shift in NB and hence it is free in hardware implementation. However, such an operation in PB is not free. On the other hand, field

1. For the field polynomial (6), the multiplication with the constant ω^{11} in (5) requires only 33 XOR gates and a delay of $2T_X$.

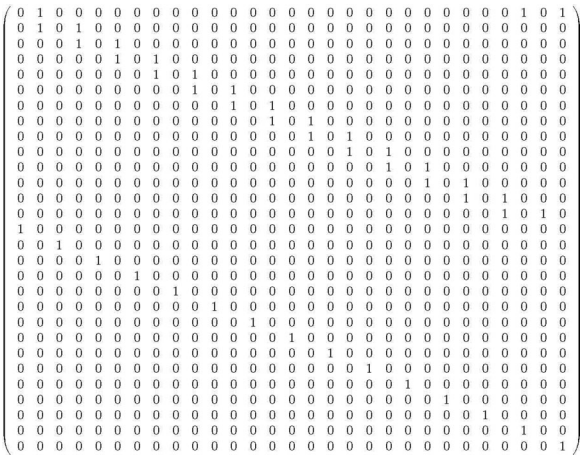

Fig. 3. The matrix S for WG$(29, 11)$.

multiplication using PB requires lower complexity than the one using NB. In PB, the complexities of these operations depend on the irreducible polynomial that constructs the finite field. It is known that irreducible trinomials define PBs with low space and time complexities [25], [26], [27]. For $GF(2^{29})$, the following two trinomials are irreducible over $GF(2)$

$$t_1(x) = x^{29} + x^2 + 1, \tag{9}$$

and its reciprocal function $t_2(x) = x^{29}(t_1(x^{-1})) = x^{29} + x^{27} + 1$. Between t_1 and t_2, t_1 offers operations with lower space complexities. Specifically, the t_1-based PB multiplier requires $29^2 = 841$ ANDs and $29^2 - 1 = 840$ XORs with a propagation delay of $T_A + 7T_X$ [25], where T_A and T_X are the delays in an AND and an XOR, respectively. In the following, we obtain the complexities of the squarings using the PB defined by (9).

Let A be an arbitrary element of $GF(2^m)$ represented in the PB, and let $V = A^2$. Denote by **a** and **v**, the row vector representations of A and V w.r.t the PB, respectively. Then, $\mathbf{v} = \mathbf{aS}$, where **S** is the binary $m \times m$ squaring matrix whose entries are either 0 or 1 [27]. In general, $W = A^{2^e}$ is obtained as $\mathbf{w} = \mathbf{aS}^e$. This formulation involves m inner products $\mathbf{aS}^e_j$, where $\mathbf{S}^e_j$ denotes the jth column vector of $\mathbf{S}^e$, $0 \le j < m$. Let N_X denote the number of XOR gates. Then, the hardware realization of $\mathbf{aS}^e$ requires $N_X = \sum_{H(\mathbf{S}^e_j)>1, 0\le j<m}(H(\mathbf{S}^e_j) - 1)$ and $T_{\mathbf{S}^e} = \lceil \log_2(\theta) \rceil T_X$, where $T_{\mathbf{S}^e}$ is the propagation delay for computing $\mathbf{aS}^e$, $H(\Omega)$ is the Hamming weight of a vector Ω, and $\theta = \max_{H(\mathbf{S}^e_j)>1} \{H(\mathbf{S}^e_j) \,|\, 0 \le j < m\}$.

For the PB defined by (9), the squaring matrix **S** is shown in Fig. 3. Table 1 lists the space and time complexities, before and after signal reuse, for the different squaring matrices used in the WG$(29, 11)$'s implementations.

3.2.2 Characteristic Polynomial of the LFSR

A primitive characteristic polynomial of degree 11 over $GF(2^{29})$ is required in order for the WG$(29, 11)$ to produce key-streams with maximal period of $2^{319} - 1$ [8], [10]. For space efficiency, the following primitive pentanomial is selected:

TABLE 1
The Space and Time Complexities of the Different Squaring Matrices Used in the WG$(29, 11)$ Stream Cipher

	No Sig. Reuse		Sig. Reuse	
	XOR	PD	XOR	PD
$\mathbf{S}, \mathbf{S}^{30}$	15	T_X	15	T_X
$\mathbf{S}^2$	37	$2T_X$	30	$2T_X$
$\mathbf{S}^4$	118	$3T_X$	65	$3T_X$
$\mathbf{S}^5$	182	$4T_X$	97	$4T_X$
$\mathbf{S}^{10}$	374	$5T_X$	214	$5T_X$
$\mathbf{S}^{20}$	338	$5T_X$	200	$5T_X$

PD = Propagation delay.

$$Z^{11} \oplus Z^6 \oplus Z^2 \oplus Z \oplus \alpha, \tag{10}$$

where $\alpha \in GF(2^{29})$ is a root of the defining polynomial (9). The primitive property of the polynomial has been verified using the "is_primitive()" method provided by the Sage Notebook online tool [28]. Let $\{A_i,\ 0 \le i < 2^{319} - 1\}$ denote the sequence generated by (10). According to [16], the following recurrence relation generates the sequence $\{B_i = A_i \oplus 1,\ 0 \le i < 2^{319} - 1\}$:

$$B_{j+11} = (B_{j+6} \oplus B_{j+2} \oplus B_{j+1} \oplus \alpha B_j) \oplus \alpha,\ \ j \ge 0, \tag{11}$$

where $\{B_i = A_i \oplus 1,\ 0 \le i \le 10\}$ is the initial state of the LFSR. By constructing the LFSR based on (11) instead of (10), then, one obtains $Y = 1 \oplus A_{i+10} = B_{i+10}$ in (7) and (8). In addition, notice that (11) requires only three field additions, one field multiplication with α (a constant), and one NOT gate (for addition with α).[2]

3.2.3 Trace Vector

Let the elements in $GF(2^m)$ be represented in the PB which is defined by an irreducible polynomial $f(x)$ of degree m over $GF(2)$. Then, the trace of an element $A \in GF(2^m)$ is obtained as $Tr(A) = \mathbf{a}\tau^T$, where $\tau = (\tau_0, \tau_1, \ldots, \tau_{m-1})$ is a unique and constant m-bit vector such that $\tau_i = Tr(\alpha^i) \in GF(2),\ 0 \le i < m$ and $f(\alpha) = 0$ [27]. Therefore, for the PB $\{1, \alpha, \ldots, \alpha^{28}\}$ defined by (9), one obtains $\tau_i = 1$ for $i \in \{0, 27\}$ and $\tau_i = 0$ otherwise. Thus,

$$Tr(A) = a_0 + a_{27}. \tag{12}$$

3.2.4 Trace of Multiplication of Two Field Elements

The authors of [16] present a method for the direct computation of the trace of the multiplication of two elements represented in the type-II ONB. In the following, a formulation for the direct computation of the trace of the multiplication of two field elements represented in PB is constructed. This method is then used to optimize the space complexity of the PB based implementations of the WG$(29, 11)$ and the WG-16 (see Sections 3.3 and 4.4).

2. For the field polynomial (9), one can easily find that the multiplication with the constant α requires only one XOR gate with a propagation delay T_X.

Proposition 1. *Consider the m-bit trace vector $\tau = (\tau_0, \ldots, \tau_{m-1})$, $\tau_i = Tr(\alpha^i)$, where α is the root of the defining polynomial of $GF(2^m)$ over $GF(2)$ [27]. For any two field elements $A = (a_0, \ldots, a_{m-1})$ and $B = (b_0, \ldots, b_{m-1})$, let $C = AB \in GF(2^m)$. Then, we have:*

$$Tr(C) = \sum_{i=0}^{m-1} \tau_i \sum_{j=0}^{i} a_{i-j} b_j + \sum_{i=0}^{m-1} \tau_i \sum_{k=0}^{m-2} q_{k,i} \sum_{j=k+1}^{m-1} a_{m-j+k} b_j, \tag{13}$$

where $\mathbf{Q}_{(m-1)\times m} = [q_{k,i}]$ is the reduction matrix and, $\mathbf{U}_{(m-1)\times m} = [u_{k,j}]$ and $\mathbf{L}_{m\times m} = [l_{i,j}]$ are as follows [25]:

$$\mathbf{U} = \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix},$$

and

$$\mathbf{L} = \begin{bmatrix} a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 \end{bmatrix}.$$

Proof. Let $\mathbf{b} = (b_0, \ldots, b_{m-1})$ and $\mathbf{c} = (c_0, \ldots, c_{m-1})$ be the vector representations of B and C, respectively, then, from [25] one has

$$\mathbf{c}^T = \mathbf{L}\mathbf{b}^T + \mathbf{Q}^T\mathbf{U}\mathbf{b}^T, \tag{14}$$

where $\mathbf{Q}^T$ is the transpose of $\mathbf{Q}$. We have the following computations:

$$\begin{aligned} Tr(C) = \mathbf{c}\tau^T &= (\mathbf{L}\mathbf{b}^T)^T\tau^T + (\mathbf{Q}^T\mathbf{U}\mathbf{b}^T)^T\tau^T \\ &= \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} l_{i,j} b_j \tau_i + \sum_{i=0}^{m-1}\sum_{j=0}^{m-1}\sum_{k=0}^{m-2} q_{k,i} u_{k,j} b_j \tau_i \\ &= \sum_{i=0}^{m-1} \tau_i \sum_{j=0}^{i} l_{i,j} b_j + \sum_{i=0}^{m-1} \tau_i \sum_{k=0}^{m-2} q_{k,i} \sum_{j=k+1}^{m-1} u_{k,j} b_j, \end{aligned}$$

where the last result is obtained by noticing that $l_{i,j} = 0$ for $j > i$ and $u_{k,j} = 0$ for $j \leq k$ [25], and by replacing $l_{i,j}$ and $u_{k,j}$ with the corresponding entries from $\mathbf{L}$ and $\mathbf{U}$, respectively, one obtains (13). □

The hardware realization of (13) requires n ANDs, $n-1$ XORs, and a propagation delay of $T_A + \lceil \log_2(n) \rceil T_X$, where $n = \sum_{\tau_i \neq 0}(i+1) + \sum_{\tau_i \neq 0, q_{k,i} \neq 0}(m-k-1)$ is the upper bound of the number of terms $(a_{i-j}b_j)$ and $(a_{m-j+k}b_j)$ in (13). It is noted that if τ and $\mathbf{Q}$ have low Hamming weights, then, the computation of $Tr(AB)$ using (13) becomes more efficient (in terms of space) than the straight forward method. In what follows, the realization of (13) for the WG$(29, 11)$ is derived.

Corollary 1. *Let $\{1, \alpha, \ldots, \alpha^{28}\}$ be the PB of $GF(2^{29})$ over $GF(2)$ which is defined by (9). Then, the trace of the multiplication of two field elements $A = \sum_{i=0}^{28} a_i \alpha^i$ and $B = \sum_{i=0}^{28} b_i \alpha^i$ is computed as follows:*

$$\begin{aligned} Tr(AB) = (a_0 + a_{27})b_0 &+ \sum_{j=1}^{25}(a_{27-j} + a_{29-j})b_j \\ &+ (a_1 + a_{26})b_{28} + \sum_{j=26}^{27}(a_{27-j} + a_{29-j} + a_{54-j})b_j. \end{aligned} \tag{15}$$

Proof. It is noted that τ has only two nonzero components, τ_0 and τ_{27} (see Section 3.2.3). We have computed the $\mathbf{Q}$ (reduction) matrix for the field polynomial (9) and found that the only nonzero entries in the first and the 28th columns of this matrix are $q_{0,0}$, $q_{27,0}$, $q_{25,27}$, and $q_{27,27}$. Hence, (15) results from substituting these values in (13). □

It is noted that the realization of (15) requires 29 AND and 59 XOR gates with a time delay of $T_A + 6T_X$.

3.3 Architecture and FSM

3.3.1 Architecture of the WG$(29, 11)$ Cipher

The PB (defined by (9)) based architecture of the WG$(29, 11)$, according to the $WGT29$ formulations in (7) and (8), and the linear recurrence (11), is shown in Figs. 4a and 4b. The squaring matrices are implemented using the signal reuse constructions, the complexities of which are presented in Table 1. The complement operator, i.e. $\sim$, invert the first bit of the input, which requires only one NOT gate. Notice that αB_i, which is required for generating the LFSR feedback signal, is stored in the right most cell of the LFSR (i.e. B_i') as shown in Fig. 4a. This is done to reduce the propagation delay through the LFSR feedback by one multiplier. This construction avoids having the LFSR's critical path constraining the speed of the cipher when pipelining is applied to the transform.

The finite state machine (FSM) controls the cipher during three different phases of operation (see Section (3.3.2)). During the load phase, the LFSR shifts at each clock cycle, where its leftmost cell is loaded with $1 \oplus IV$ (IV is the initial vector).

It is noted that the initial feedback signal $IF = WGP29$, which is needed for initialization phase, is missing in Fig. 4a. This is a result of computing $WGT29$ according to (7) using (15). Let $q = 2^{10} - 1$, $r_1 = 2^{10} + 1$, $r_2 = 2^{20} + r_1$, $r_3 = 2^{20} - q$, and $r_4 = 2^{20} + q$. Therefore, the $WGP29$ in (1) can be written as $1 \oplus Y \oplus Y^{r_1} \oplus Y^{r_2} \oplus Y^{r_3} \oplus Y^{r_4}$, and is recovered using serial computation over three clock cycles as described in Table 2. During the initialization phase, the LFSR shifts once every three clock cycles and loads its leftmost cell with $IF \oplus \overline{LF}$, where $\overline{LF} = LF \oplus 1$ and LF is the original linear feedback given by (10).

In the running phase, the LFSR updates its state in each clock cycle, where B_{i+10} is loaded with $\overline{LF}$. In Fig. 4a, the keystream bits are obtained from XORing $Tr(1 \oplus Y \oplus Y^{r_1})$ with $Tr(Y^{r_2} \oplus (Y^{r_3})^{2^{20}} \oplus Y^{r_4})$. $Tr(1 \oplus Y \oplus Y^{r_1})$ is the result of XORing $Tr(1 \oplus Y)$ and $Tr(Y^{r_1})$. $Tr(1 \oplus Y)$ and $Tr(Y^{r_1})$ are

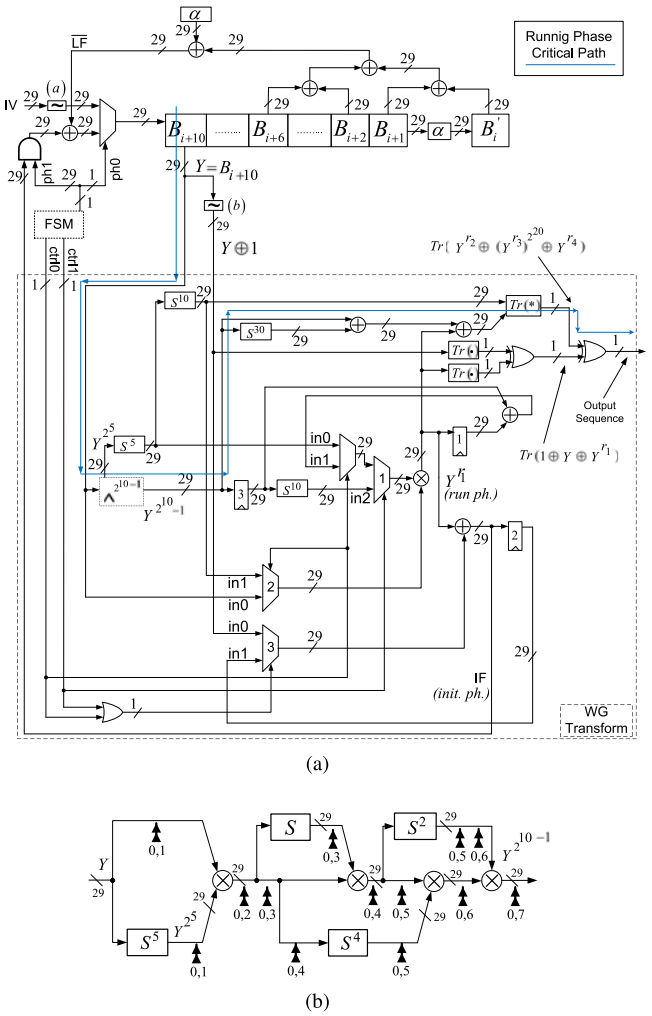

Fig. 4. a) Architecture of the WG$(29, 11)$ stream cipher. $Tr(\bullet)$ generates the trace of a $GF(2^{29})$ element (see Section 3.2.3). $Tr(\star)$ generates the trace of the multiplication of two $GF(2^{29})$ elements using (15). Y is the output of the LFSR represented by (11). b) Implementation of Y^q ($q = 2^{10} - 1$) based on (8). An arrow represents a register which is inserted for pipelining. A number n under a register means it is clocked at end of the nth clock cycle during initialization phase. A zero under a register indicates that the register's clock input is always enabled during the run phase. $r_1 = 2^{10} + 1$, $r_2 = 2^{20} + 2^{10} + 1$, $r_3 = 2^{20} - 2^{10} + 1$, and $r_4 = 2^{20} + 2^{10} - 1$.

produced by applying operator $Tr(\bullet)$ to $1 \oplus Y$ and Y^{r_1}, respectively. The operator $Tr(\bullet)$ generates its output according to (12). Y^{r_1} is generated by multiplying Y with $Y^{2^{10}}$ in

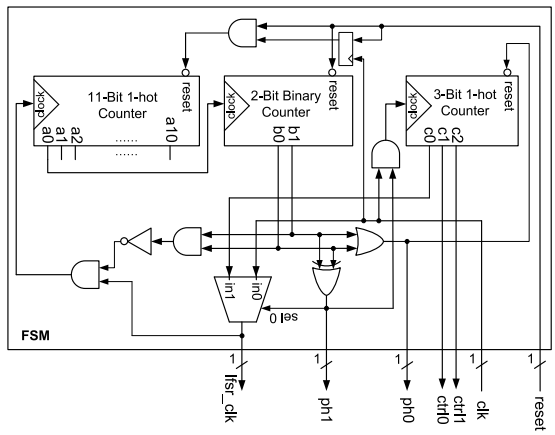

Fig. 5. FSM for the PB based implementation of the WG$(29, 11)$ stream cipher.

$GF(2^{29})$ (by setting $\text{ctrl0} = \text{ctrl1} = 0$ for the running phase in Fig. 4a). Y is the output B_{i+10} of the LFSR and $Y^{2^{10}}$ is obtained from the squarer $\mathbf{S}^5$ operating on Y^{2^5}, which in turn, is available from the generator of $Y^{2^{10}-1}$ (see Fig. 4b). $1 \oplus Y$ is the addition of $Y \in GF(2^{29})$ with the unity element $1 = (1, 0, 0, \ldots, 0)$ represented w.r.t. PB. Thus, $1 \oplus Y$ results from inverting the least significant bit of B_{i+10} by the complement operator $\sim$. $Tr(Y^{r_2} \oplus (Y^{r_3})^{2^{20}} \oplus Y^{r_4})$ is generated by applying (15) to $Y^{2^{20}}$ and $(Y^{r_1} \oplus Y^q \oplus Y^{2^{30}q})$. The signal $Y^{2^{20}}$ is the result of $\mathbf{S}^{10}$ operating on $Y^{2^{10}}$. Signal $Y^{r_1} \oplus Y^q \oplus Y^{2^{30}q}$ is the bitwise XOR of Y^{r_1}, Y^q, and $Y^{2^{30}q}$, where $Y^{2^{30}q}$ is obtained from $\mathbf{S}^{30}$ operating on Y^q and Y^q is generated as presented in Fig. 4b.

3.3.2 The Finite State Machine

The architecture of the FSM is shown in Fig. 5. The FSM controls the inputs to the LFSR during the three phases of operations through signals ph0 and ph1. As presented in Table 3 for the column of Fig. 4a the loading phase takes 11 clock cycles followed by the initialization phase which stays for $33 + 33 = 66$ clock cycles, then starts the run phase. The FSM is built from a 2-bit binary counter, an 11-bit 1-hot counter, and a 3-bit 1-hot counter. The first counter generates ph0 and ph1. The 11-bit counter triggers the clock of the 2-bit counter, every 11 counts, during the loading and initialization. The 3-bit counter, generates ctrl0 and ctrl1,

TABLE 2
Computation of the $IF = WGP29$ Signal over Three Clock Cycles during the Initialization Phase

ctrl0	ctrl1	Output			Next State	
		MUX # 1	MUX # 2	MUX # 3	Register 1	Register 2
0	0	$Y^{2^{10}}$	Y	$Y \oplus 1$	Y^{r_1}	$Y^{r_1} \oplus Y \oplus 1$
1	0	$Y^{r_1} \oplus Y^q$	$Y^{2^{20}}$	$Y^{r_1} \oplus Y \oplus 1$	$Y^{r_4} \oplus Y^{r_2}$	$Y^{r_4} \oplus Y^{r_2} \oplus Y^{r_1} \oplus Y \oplus 1$
0	1	$Y^{2^{10}q}$	Y	$Y^{r_4} \oplus Y^{r_2} \oplus Y^{r_1} \oplus Y \oplus 1$	Y^{r_3}	$Y^{r_4} \oplus Y^{r_3} \oplus Y^{r_2} \oplus Y^{r_1} \oplus Y \oplus 1$

ctrl0 and ctrl1 are generated by the FSM. $WGP29$ is the next state of Register 2 in stage 3. Next state of Register 3 is always Y^q. Rows are listed in order of computation stages (first to last).

TABLE 3
Phase of Operation in the Proposed PB Based WG Designs (Figs. 4A, 6, 8, 11A, 12, and 14) as a Function of the State of the 2-Bit Binary Counter

2-bit counter		ph1/ph0	phase of operation	Number of Clock Cycles for the Proposed Designs					
a_1	a_0			Figure 4a	Figure 6	Figure 8	Figure 11a	Figure 12	Figure 14
0	0	0/0	Load	11	11	11	32	32	32
0	1	1/1	Init.	33	88	132	96	288	416
1	0	1/1	Init.	33	88	132	96	288	416
1	1	0/1	Run	-	-	-	-	-	-

and triggers the clock of the 11-bit counter as well as the clock of the LFSR, every three counts, during initialization.

3.4 Serialized Implementation of the PB Based WG$(29, 11)$

3.4.1 Architecture of the Serialized WG$(29, 11)$

Here we present a serialized $WGP29/WGT29$ design for area constrained applications. The serial WG$(29, 11)$ which is proposed in this section has the same LFSR, compared to the standard design in Fig. 4a; however, the WG transform and the FSM are modified. Fig. 6 presents the proposed serial $WGP29/WGT29$ architecture. In this architecture, only one multiplier is used. The computations of the different variables used in (7) and (8) are accomplished sequentially according to Table 4.

It is noted that no changes are required for the loading phase of the serial WG$(29, 11)$. However, in the architecture of Fig. 6, an initialization round takes seven clock cycles to generate the $WGP29$ signal. The LFSR is updated at the eighth clock cycle. During the run phase, a stream bit is produced every six cycles. During these two phases, the multiplexers provide the inputs to the multiplier and the adder. The multiplexers' inputs are multiplexed by selectors m_0-m_4. The three registers are clocked as it is specified by the clocking table in Fig. 6. The clocking of the different registers is enabled by means of clock enable signals (see Section 3.4.2). In this design, the lfsr_clk signal in Fig. 7a is required in order to clock the LFSR once every one clock cycle, eight clock cycles, and six clock cycles, during loading, initialization, and run phases, respectively. This means that the initialization phase takes a total of $8 \times 22 = 176$ clock cycles. The number of clock cycles needed for different phases of Fig. 6 are presented in the corresponding column of Table 3. Moreover, the signal EO in Fig. 7a is used to enable the keystream output every six clock cycles during the run phase. These selectors, clock enables, lfsr_clk, and EO signals are generated through the FSM, as it is presented next.

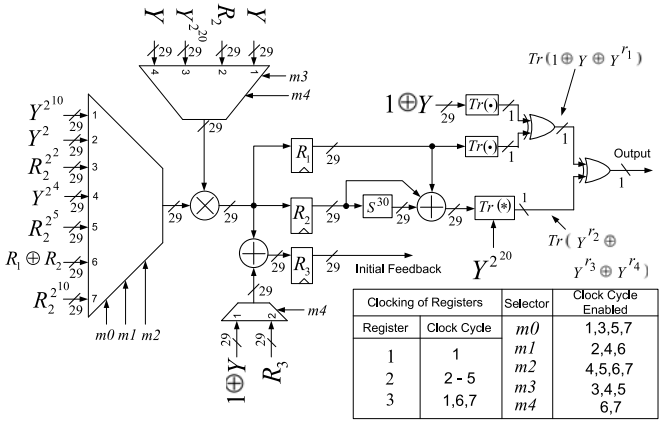

Fig. 6. Architecture of the serial $WGP29/WGT29$ implementation for the PB based design of the WG$(29, 11)$. $Y = B_{i+10}$ is the LFSR's output (see Fig. 4a), $r_1 = 2^{10} + 1$, $r_2 = 2^{20} + 2^{10} + 1$, $r_3 = 2^{20} - 2^{10} + 1$, and $r_4 = 2^{20} + 2^{10} - 1$.

3.4.2 FSM for the Serialized PB based WG$(29, 11)$

Fig. 7a is a block diagram for the FSM which is used for the serialized PB based WG$(29, 11)$. The FSM controls the inputs to the LFSR during the three phases of operations. As shown in Table 3 for Fig. 6, the loading phase takes 11 clock cycles followed by the initialization phase which stays for 176 clock cycles, then starts the run phase. The FSM is built from a 2-bit binary counter, an 11-bit 1-hot counter, an 8-bit 1-hot counter, and a 6-bit 1-hot counter. The 2-bit counter generates ph0 and ph1 according to Table 3. The 11-bit counter triggers the clock of the 2-bit counter, every 11 counts, during the loading and initialization. The 8-bit counter, generates the clock enable signals and the multiplexers' selectors (see Fig. 6), and triggers the clock of the 11-bit counter as well as the clock of the LFSR, every eight counts, during initialization. In the run phase, the 6-bit counter, generates the clock enable signals and the multiplexers' selectors, and triggers the clock of the LFSR, every six counts. From the starting of the run phase, the 6-bit counter enables the output of the cipher every 6 counts.

3.5 Pipelined Implementation of the PB Based WG$(29, 11)$

3.5.1 Architecture

Figs. 8 and 4b present the pipelined version of the PB based implementation of the $WGT29$. The pipeline has been constructed with 10-stages during the run phase and 12-stages during the initialization phase, in order to achieve a critical path with only one multiplier. In these figures, the double headed arrows point to the locations where the registers are inserted, for the pipeline. The numbers under these arrows indicate the clock cycles, throughout initialization, during which the registers will be clock-enabled. A zero below a register means that its clock input will always be enabled during the run phase. The clocking of the different registers in the transform is controlled by means of clock enable signals (see Section 3.5.2).

TABLE 4
Steps for Computing the $WGP29$ and $WGT29$ in the Serial Implementation of the PB Based WG$(29, 11)$ Design

		Clock Cycle			
		1	2	3	4
Next State	Register 1	Y^{r_1}	Y^{r_1}	Y^{r_1}	Y^{r_1}
	Register 2	-	Y^{2+1}	$Y^{\sum_{i=0}^{3} 2^i}$	$Y^{\sum_{i=0}^{4} 2^i}$
	Register 3	$1 \oplus Y \oplus Y^{r_1}$	$1 \oplus Y \oplus Y^{r_1}$	$1 \oplus Y \oplus Y^{r_1}$	$1 \oplus Y \oplus Y^{r_1}$
		Clock Cycle			
		5	6	7	
Next State	Register 1	Y^{r_1}	Y^{r_1}	Y^{r_1}	
	Register 2	Y^{q}	Y^{q}	Y^{q}	
	Register 3	$1 \oplus Y \oplus Y^{r_1}$	$1 \oplus Y \oplus$ $Y^{r_1} \oplus Y^{r_2} \oplus Y^{r_4}$	$1 \oplus Y \oplus Y^{r_1} \oplus$ $Y^{r_2} \oplus Y^{r_3} \oplus Y^{r_4}$	

$Y = B_{i+10}$ *is the LFSR's output (see Fig. 4A),* $r_1 = 2^{10} + 1$, $r_2 = 2^{20} + 2^{10} + 1$, $r_3 = 2^{20} - 2^{10} + 1$, $r_4 = 2^{20} + 2^{10} - 1$, *and* $q = 2^{10} - 1$.

It is noted that no changes are required for the loading phase. However, during the initialization and the run phases, an input signal now requires 12 and 10 clock cycles, respectively, to propagate to the output of the transform/permutation. Therefore, for the initialization phase, the lfsr_clk signal in Fig. 9a triggers the LFSR once every 12 cycles. This means that the initialization phase takes a total of $12 \times 22 = 264$ clock cycles as presented in Table 3 for Fig. 8. Also, the multiplexers' outputs in Fig. 8 are controlled through the signals ctrl0 and ctrl1 (Fig. 9b) during the initialization and the run phases. For the run phase, an output enable signal, EO in Fig. 9a, is used to enable the keystream output after the first 10 clock cycles. The following section presents the FSM and show how the different control signals are derived.

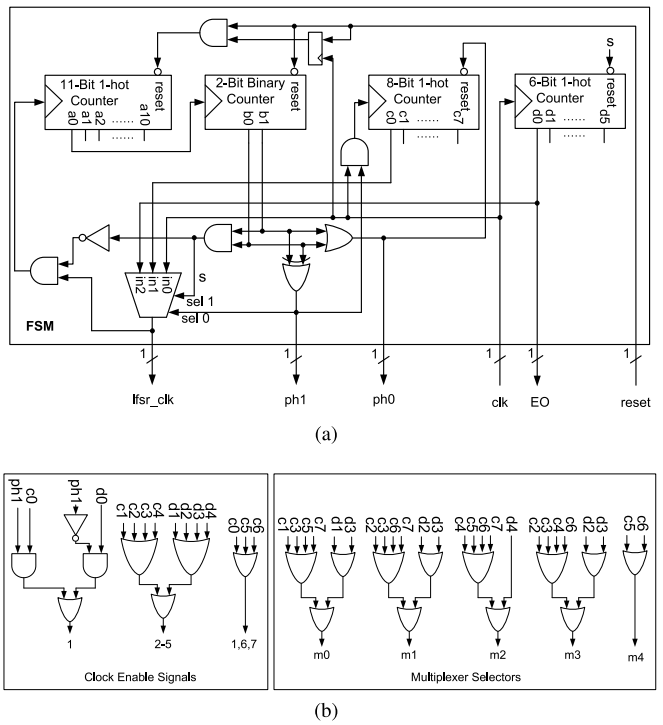

Fig. 7. a) Architecture of the FSM for the serialized PB based implementation of the WG$(29, 11)$. b) Generating the clock enable control signals and the multiplexers' Selectors. For the clock enable signals, the number at the output of an OR gate indicates the number of the enabled clock cycle during the initialization phase. m_0, m_1, m_2, m_3, and m_4 are the selectors for the multiplexers.

3.5.2 FSM for the Pipelined PB Based Implementation of the WG$(29, 11)$

Fig. 9a presents the architecture of the FSM which is used for the pipelined version of the PB based implementation of the WG$(29, 11)$. Similar to the previously introduced FSMs in this paper, the FSM controls the inputs to the LFSR during the three phases of operations through generating the signals ph0 and ph1. According to column of Fig. 8 in Table 3, the loading phase takes 11 clock cycles, followed by the initialization phase which stays for 264 clock cycles, followed by the run phase. The FSM is built from a 2-bit binary counter, an 11-bit 1-hot counter, and 12-bit 1-hot counter. The 2-bit counter generates ph0 and ph1 according to

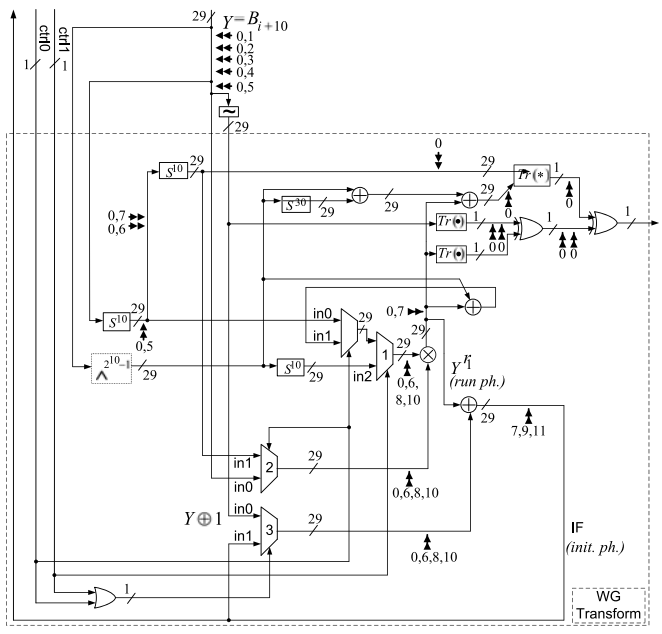

Fig. 8. Pipelined version of the PB based architecture of the $WGT29$. The double headed arrows indicate the location of the inserted registers for the pipeline. The numbers under the arrows indicate the clock cycles, throughout initialization, during which the registers will be clock-enabled. A 0 under an arrow means the clock input of this register will always be enabled during the run phase.

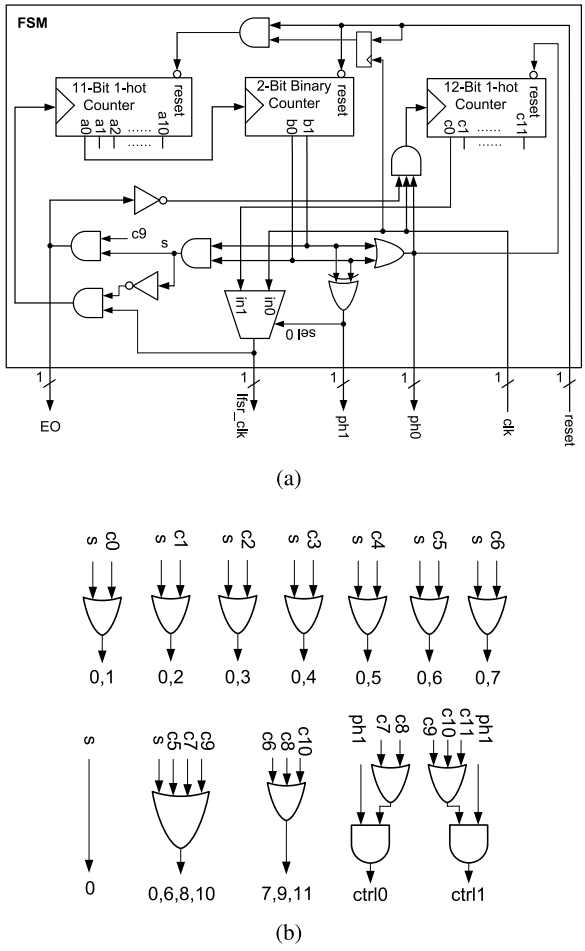

Fig. 9. a) Architecture of the FSM for the pipelined version of the PB based implementation of the WG$(29, 11)$. b) Generating the clock enable signals and, ctrl0 and ctrl1 signals. The numbers at the output indicate the clock cycles, throughout initialization, during which the register will be clock-enabled. A 0 at the output means the clock input will be always enabled during the run phase. Signals s and c_i, $0 \leq i \leq 11$, are shown in Fig. 9a.

Table 3. The 11-bit counter triggers the clock of the 2-bit counter, every 11 counts, during the loading and initialization. The 12-bit counter, generates the clock enable signals and the multiplexers' selectors (ctrl0 and ctrl1, see Fig. 8), and triggers the clock of the 11-bit counter as well as the clock of the LFSR, every 12 counts, during initialization. In the run phase, signal s in Fig. 9a and the 12-bit counter, generate the clock enable signals and the multiplexers' selectors (fixed at ctrl0=ctrl1=0), respectively. The 12-bit counter enables the output of the cipher after 10 counts from the start of the run phase. The LFSR is triggered with each clock cycle in the run phase.

4 Architectures of the WG-16 Stream Cipher

The WG-16 cipher has been proposed by the authors of [17] for securing the 4G's confidentiality and integrity protection schemes against the attack in [21]. The only WG-16 hardware design, which uses NB, is presented in [22]. This design is based on composite field arithmetic and properties of the trace function in the tower field representation.

Here, we propose a new formulation of the WG-16 permutation which requires eight multiplications compared to 10 in the formulation of [22]. Based on this formulation, and using the trace property in (13), this section presents six hardware architectures of the WG-16, based on the PB representation for the first time. The six designs include a standard architecture, its serial version, and its pipelined version using two different types of multipliers for each version. The serial version can be used for low-area applications whereas the pipelined one is suitable for high-speed applications. The pipelined instance of the proposed scheme offers almost twice the throughput which is reported by the implementations in [22], at a slightly smaller area. In what follows, the formulation of the WG-16 transform followed by the formulations used for squaring and trace function are derived. In addition, the formulation for computing the trace of the multiplication of two field elements, in the PB, is obtained. Then, the proposed standard architecture of the WG-16 is shown. The section ends by presenting serialized and pipelined versions of the standard design.

4.1 Formulations of $WGP16$ and $WGT16$

The $WGP16$'s formulation in (4) requires 10 multiplications when the field elements are represented in the PB. In the following, a new formulation is derived which requires eight multiplications.

Proposition 2. *The WG permutation of the WG-16 stream cipher is computed as follows:*

$$WGP16 = 1 \oplus Y \oplus Y^{2^{11}+1} \oplus Y^{2^{11}(2^5-1)+2^6} \oplus Y^{2^{11}+1}(Y^{2^6} \oplus Y^{2(2^5-1)}), \quad (16)$$

where $Y = (A_{i+31})^{1,057} \oplus 1$, A_{i+31} is the output of the LFSR described by (5), and Y^{2^5-1} is computed as follows:

$$Y^{2^5-1} = (Y^{2^2+1})^{2+1} Y^{2^4}. \quad (17)$$

Proof. Let $e_1 = 2^{11} + 1$, $e_2 = 2^{11} + 2^6 + 1$, $e_3 = -2^{11} + 2^6 + 1$, and $e_4 = 2^{11} + 2^6 - 1$ in (3). By noticing that $e_3 + 2^{16} - 1 \equiv e_3 \pmod{2^{16} - 1}$, then, one obtains

$$e_2 = e_1 + 2^6, \quad e_3 = 2^{11}s + 2^6, \quad e_4 = e_1 + 2s,$$

where $s = 2^5 - 1$, and the proof is completed by taking Y^{e_1} as a common factor between Y^{e_2} and Y^{e_4}. □

The WG transform is obtained by taking the trace of (16). Equation (16) requires eight $GF(2^{16})$ multiplications: 1 for computing $Y^{2^{11}+1}$, 3 for computing Y^{2^5-1}, 1 for computing $Y^{2^{11}(2^5-1)+2^6}$, 1 for computing $Y^{2^{11}+1}(Y^{2^6} \oplus Y^{2(2^5-1)})$, and 2 for computing $1 \oplus Y = (A_{i+31})^{1,057}$. In addition to this, (16) requires seven squarings and five $GF(2^{16})$ additions. For the transform, the computation of the trace of $WGP16$ is required. Section 4.3 presents a method which reduces the number of multiplications in the $WGT16$ to only 6 through computing $Tr(Y^{2^{11}(2^5-1)}Y^{2^6})$ directly from $Y^{2^{11}(2^5-1)}$ and Y^{2^6}, and $Tr(Y^{(2^{11}+1)}(Y^{2^6} \oplus Y^{2(2^5-1)}))$ directly from $Y^{(2^{11}+1)}$ and $Y^{2^6} \oplus Y^{2(2^5-1)}$, without performing the multiplications.

$$\begin{pmatrix}
0&1&0&0&0&0&0&1&0&0&0&1&1&1&1&1\\
1&1&0&1&0&0&0&0&0&1&0&1&1&0&1&0\\
1&0&1&1&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&1&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&1&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&1&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix}$$

Fig. 10. The matrix **S** for WG-16.

TABLE 5
The Space and Propagation Delay Complexities of the Different Squaring Matrices Used in the WG-16 Stream Cipher

	No Sig. Reuse		Sig. Reuse			No Sig. Reuse		Sig. Reuse	
	XOR	PD	XOR	PD		XOR	PD	XOR	PD
$\mathbf{S}$	30	$3T_X$	21	$3T_X$	$\mathbf{S}^6$	99	$4T_X$	63	$4T_X$
$\mathbf{S}^2$	82	$3T_X$	45	$3T_X$	$\mathbf{S}^9$	89	$4T_X$	58	$4T_X$
$\mathbf{S}^4$	103	$4T_X$	64	$4T_X$	$\mathbf{S}^{10}$	102	$4T_X$	60	$4T_X$
$\mathbf{S}^5$	89	$4T_X$	58	$4T_X$	$\mathbf{S}^{11}$	115	$4T_X$	62	$4T_X$

$GF(2^{16})$ elements are represented by the PB which is defined by (6). PD = Propagation Delay.

4.2 Squaring Matrices and Trace Vector

Similar to the WG$(29, 11)$, in what follows, the squaring matrices and the the trace vector for the field polynomial (6) are presented.

4.2.1 Squaring Matrices

Fig. 10 shows the squaring matrix **S** for the field polynomial (6). One can find the required squaring operations for the WG-16 permutation from (16) and (17). Table 5 lists the space and propagation delay complexities of the different squaring matrices used in the WG-16 implementation (before and after signal reuse).

4.2.2 Trace Vector

The trace vector for the PB $\{1, \alpha, \ldots, \alpha^{15}\}$ defined by (6) is $\tau = (\tau_0, \ldots, \tau_{15})$ where $\tau_i = 1$ for $i \in \{11, 13\}$ and $\tau_i = 0$ otherwise (see Section 3.2.3). Thus, for $A \in GF(2^{16})$

$$Tr(A) = a_{11} + a_{13}. \tag{18}$$

4.3 Trace of the Multiplication of Two Field Elements for the PB Based WG-16

The following is the realization of (13) when applied to WG-16.

Corollary 2. *Consider the $GF(2^{16})$ defined by (6) where $\{1, \alpha, \ldots, \alpha^{15}\}$ is its PB. Then, the trace of the multiplication of two field elements $A = \sum_{i=0}^{15} a_i\alpha^i$ and $B = \sum_{i=0}^{15} b_i\alpha^i$ is computed as follows:*

$$\begin{aligned}
Tr(AB) = &\sum_{j=0}^{11}(a_{11-j} + a_{13-j})b_j + \sum_{j=12}^{13} a_{13-j}b_j\\
&+ \sum_{j=7}^{9} a_{22-j}b_j + (a_{12} + a_{15})b_{10}\\
&+ \sum_{j=11}^{13}(a_{22-j} + a_{25-j} + a_{26-j})b_j\\
&+ \sum_{j=14}^{15}(a_{22-j} + a_{25-j} + a_{26-j} + a_{29-j})b_j.
\end{aligned} \tag{19}$$

Proof. τ has only two nonzero components, τ_{11} and τ_{13} (see Section 4.2.2). We have computed the **Q** (reduction) matrix for the field polynomial (6). Accordingly, the only nonzero entries for the 12th and the 14th columns of this matrix are $q_{6,11}$, $q_{8,11}$, $q_{9,11}$, $q_{11,11}$, $q_{8,13}$, $q_{10,13}$, $q_{11,13}$, and $q_{13,13}$. Hence, by replacing these values of τ_i and $q_{k,i}$ in (13), we get (19). □

It is noted that the realization of (19) requires 23 AND and 47 XOR gates and introduces a propagation delay of $T_A + 7T_X$.

4.4 Architecture and FSM

4.4.1 Architecture of the WG-16 Cipher

Let $e_1 = 2^{11} + 1$, $e_2 = 2^{11} + 2^6 + 1$, $e_3 = -2^{11} + 2^6 + 1$, $e_4 = 2^{11} + 2^6 - 1$, and $s = 2^5 - 1$. Figs. 11a , 11b, and 11c present the proposed architecture of the WG-16 according to the $WGP16$ formulations in (16) and (17), and the linear recurrence (5), based on the PB defined by (6). The squaring matrices are implemented using the signal reuse constructions of Table 5.

In Fig. 11a, the FSM controls the components of the cipher during the different phases of operation. This is accomplished through signals lfsr_clk, ph0, ph1, ctrl0 and ctrl1 (see Section 4.4.2 for details).

During the load phase, the LFSR shifts at each clock cycle while its leftmost cell is loaded through the Initial Vector input.

It is noted that the signal $Y^{e_2} \oplus Y^{e_4}$ is missing in Fig. 11a. This is due to the generation of $Tr(Y^{e_2} \oplus Y^{e_4})$ directly from $Y^{2^{11}+1}$ and $(Y^{2^6} \oplus Y^{2s})$ using (19). As a result, the Initial Feedback ($WGP16$) signal, which is needed for the initialization phase, does not exist. This is recovered by generating $WGP16$ over three clock cycles, during initialization, as presented in Table 6. The signals $(A_{i+31})^{1,057} = 1 \oplus Y$ and Y^{2^5-1} are generated as shown in Figs. 11c and 11b, respectively. During the initialization phase, the lfsr_clk signal triggers the LFSR every three clock cycles. The leftmost cell is loaded with the result from the field addition of the LFSR feedback and $WGP16$ (Initial Feedback).

In the running phase, the LFSR updates its state at each clock cycle. The only feedback is the LFSR feedback. The keystream bits are obtained by XORing the signals $Tr(1 \oplus Y)$, $Tr(Y^{e_1})$, $Tr(Y^{e_3})$, and $Tr(Y^{e_2} \oplus Y^{e_4})$. $Tr(1 \oplus Y)$ and $Tr(Y^{e_1})$ are produced from $1 \oplus Y$ and Y^{e_1} using (18). Y^{e_1} is generated by multiplying Y with $Y^{2^{11}}$ in $GF(2^{16})$. Y is generated by complementing the least significant bit of $(A_{i+31})^{1,057}$, and $Y^{2^{11}}$ is obtained from the squarer $\mathbf{S}^{11}$ operating on Y. $1 \oplus Y$ is simply $(A_{i+31})^{1,057}$. $Tr(Y^{e_3})$ is generated

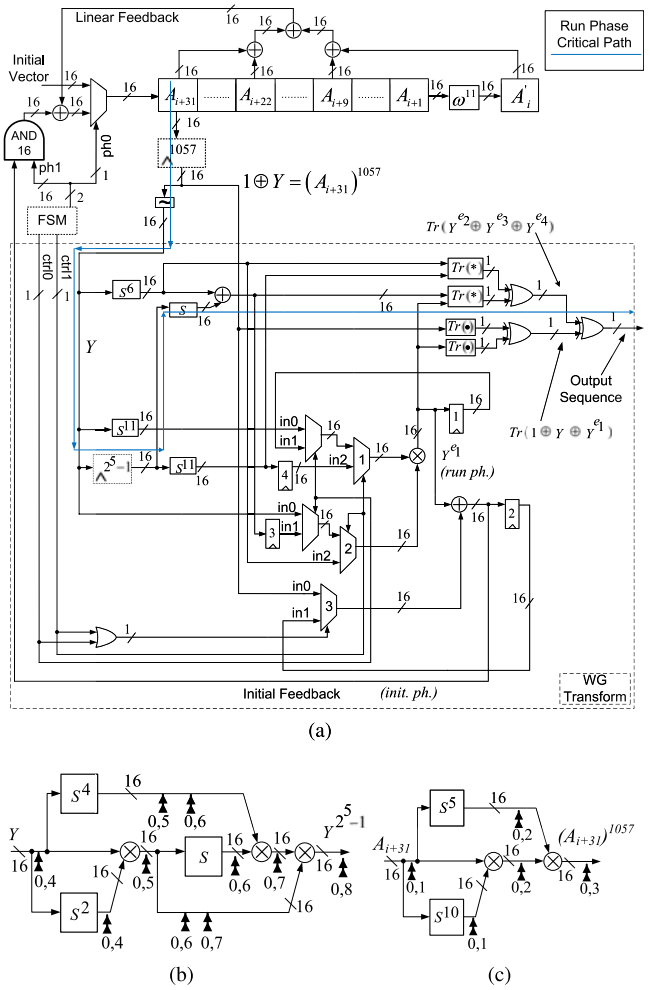

Fig. 11. a) Architecture of the WG-16 stream cipher based on the PB. $Tr(\bullet)$ generates the trace of a $GF(2^{16})$ element. $Tr(\star)$ generates the trace of the multiplication of two $GF(2^{16})$ elements. b) Generation of the signal Y^s $(s = 2^5 - 1)$. c) Generation of the signal $(A_{i+31})^{1,057}$. $e_1 = 2^{11} + 1$, $e_2 = 2^{11} + 2^6 + 1$, $e_3 = -2^{11} + 2^6 + 1$, $e_4 = 2^{11} + 2^6 - 1$, and $s = 2^5 - 1$. A double-headed arrow points to the location where a register is inserted for pipelining purposes (see Section 4.6.1).

by applying (19) to $Y^{2^{11}(2^5-1)}$ and Y^{2^6}. The signal Y^{2^6} is the result of $\mathbf{S}^6$ operating on Y. The signal $Y^{2^{11}(2^5-1)}$ is the result of $\mathbf{S}^{11}$ operating on Y^{2^5-1}. $Tr(Y^{e_2} \oplus Y^{e_4})$ is generated by applying (19) to $Y^{2^{11}+1}$ and $(Y^{2(2^5-1)} \oplus Y^{2^6})$. The signal $Y^{2(2^5-1)}$ is the result of $\mathbf{S}$ operating on Y^{2^5-1}, while signal $Y^{2(2^5-1)} \oplus Y^{2^6}$ is the bitwise XOR of $Y^{2(2^5-1)}$ and Y^{2^6}.

4.4.2 The Finite State Machine

The FSM for the PB based WG-16 is similar to the one used for the PB based implementation of the WG$(29, 11)$ (see Section 4.4.2). However, the WG-16's FSM replaces the 11-bit 1-hot counter with a 5-bit binary counter and, the clocking of the 2-bit binary counter occurs after a complete 32 counts for the 5-bit counter. As can be seen from column of Fig. 11a in Table 3, the loading phase takes 32 clock cycles. This is followed by the initialization phase which stays for 192 clock cycles, where each initialization round is extended to three clock cycles (for computing $WGP16$) by means of the 3-bit 1-hot counter. During this phase, the LFSR is clocked 64 times, once every three clocks, by means of the 3-bit 1-hot counter. After this starts the run phase. Also, the 3-bit counter controls the multiplexers' selectors, ctrl0 and ctrl1, during initialization and run phases.

4.5 Serialized Implementation of the PB Based WG-16

4.5.1 Architecture of the Serialized WG-16

The serialized computation of the WG-16 transform results in a lower space complexity, compared to the standard design in Fig. 11a. Fig. 12 presents the proposed architecture for the serial WG-16. In this architecture, the $WGP16$ is computed over eight cycles (initialization phase) while the $WGT16$ is computed over 6 cycles (run phase). The design uses only one field multiplier. The computations are accomplished according to Table 7.

It is noted that no changes are required for the loading phase, as a result of applying the serial computation. In this architecture, an initialization round takes eight clock cycles to generate the $WGP16$ signal. The LFSR is updated at the ninth clock cycle. During the run phase, a stream bit is produced every six cycles. During these two phases, the multiplexers provide the inputs to the multiplier and adder. The multiplexers' inputs are multiplexed through selectors m_0-m_3 during computations. The four registers are clocked as it is specified by the clocking table in Fig. 12. The clocking of the different registers is enabled by means of clock enable signals (see Section 4.5.2). In this design, the FSM's signal lfsr_clk is required in order to clock the LFSR once every

TABLE 6
Computation of the $\mathbf{WGP16}$ Signal over Three Clock Cycles during the Initialization Phase

$ctrl0/ctrl1$	Output			Next State		
	MUX # 1	MUX # 2	MUX # 3	Register 1	Register 2	Register 3
0/0	Y	$Y^{2^{11}}$	$Y \oplus 1$	Y^{e_1}	$Y^{e_1} \oplus Y \oplus 1$	$Y^{2^6} \oplus Y^{2s}$
1/0	Y^{e_1}	$Y^{2^6} \oplus Y^{2s}$	$Y^{e_1} \oplus Y \oplus 1$	$Y^{e_2} \oplus Y^{e_4}$	$Y^{e_4} \oplus Y^{e_2} \oplus Y^{e_1} \oplus Y \oplus 1$	$Y^{2^6} \oplus Y^{2s}$
0/1	Y^{2^6}	$Y^{2^{11}s}$	$Y^{e_4} \oplus Y^{e_2} \oplus Y^{e_1} \oplus Y \oplus 1$	Y^{e_3}	$Y^{e_4} \oplus Y^{e_3} \oplus Y^{e_2} \oplus Y^{e_1} \oplus Y \oplus 1$	$Y^{2^6} \oplus Y^{2s}$

The control signals ctrl0 and ctrl1 are generated by the FSM. $WGP16$ is the next state of Register 2 in stage 3. Next state of Register 4 is always Y^s. Rows are listed in order of computation stages (first to last). $e_1 = 2^{11} + 1$, $e_2 = 2^{11} + 2^6 + 1$, $e_3 = -2^{11} + 2^6 + 1$, $e_4 = 2^{11} + 2^6 - 1$, and $s = 2^5 - 1$.

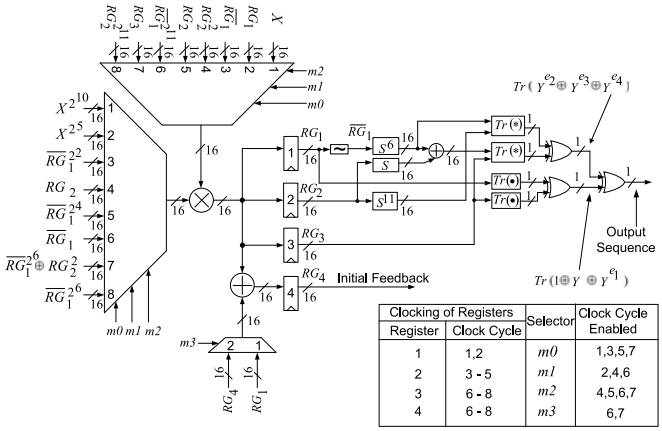

Fig. 12. Architecture of the serial implementation for the PB based design of the WG-16. In this architecture, $X = A_{i+31}$ and $Y = 1 \oplus X^{1,057}$. $e_1 = 2^{11} + 1$, $e_2 = 2^{11} + 2^6 + 1$, $e_3 = -2^{11} + 2^6 + 1$, $e_4 = 2^{11} + 2^6 - 1$, and $s = 2^5 - 1$.

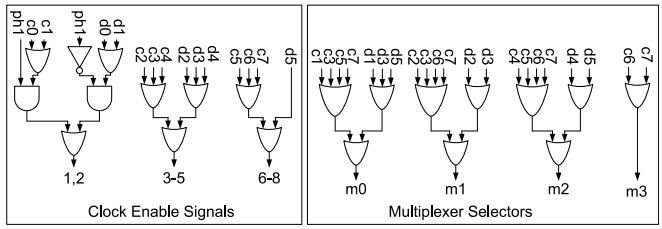

Fig. 13. Generating the clock enable control signals and the multiplexers' selectors for the serial version of the WG-16. For the clock enable signals, the number at the output of an OR gate indicates the number of the enabled clock cycle during the initialization phase. m_0, m_1, m_2, and m_3 are the selectors for the multiplexers. Signals s is shown in Fig. 7a, and c_i, $0 \le i \le 8$ and d_i, $0 \le i \le 5$, are the outputs of the 9-bit and the 6-bit 1-hot counters, respectively (see Section 4.5.2).

one, nine, and six clock cycles, during loading, initialization, and run phases, respectively. This means that the initialization phase takes a total of $9 \times 64 = 576$ cycles. Moreover, an output enable signal EO is used to enable the keystream output every six cycles during the run phase. These selectors, clock enables, lfsr_clk, and EO signals are generated through the FSM, as it is presented next.

4.5.2 FSM for the Serialized WG-16

The FSM for the serialized WG-16 is a modified version of the one in Section 3.4.2. The FSM of the serial WG-16 is obtained by replacing the 11-bit 1-hot counter with a 5-bit binary counter and the 8-bit 1-hot counter with a 9-bit 1-hot counter. The 2-bit binary counter generates ph0 and ph1, and is clocked once every 32 counts from the 5-bit binary counter. As it is shown in column of Fig. 12 in Table 3, the initialization phase takes 576 clock cycles. Each initialization round takes nine clock cycles. The LFSR is clocked at the arrival of the ninth clock cycle by means of the 9-bit 1-hot counter. During the run phase, the LFSR is clocked once every six clock cycles by means of the 6-bit counter. The cipher's output is enabled once every six clock cycles during the run phase, through the 6-bit counter. The clock enable signals which control the clocking of the registers in Fig. 12, and the multiplexers' selectors m, m_1, m_2, and m_3 are derived from the signal ph1, the outputs of the 9-bit 1-hot counter (initialization), and outputs of the 6-bit 1-hot counter (run phase), as it is shown in Fig. 13.

4.6 Pipelined Implementation of the PB Based WG-16

4.6.1 Architecture

A pipelined version of the PB based implementation of the WG-16 is presented in Figs. 14, 11b, and 11c. The critical path of this architecture has only one multiplier. This is accomplished through a pipeline which has 11-stages during the run phase and 13-stages during the initialization phase. In these figures, the double headed arrows point to the locations where the registers are inserted, for the pipeline. Also, the numbers under an arrow specify the corresponding clock cycles which trigger it during each $WGP16$ computation throughout the initialization phase (13 clock cycles for each computation). The clocking of the different registers in the transform is controlled by means of clock enable signals (see Section 4.6.2).

No changes are required for the loading phase. During the initialization and the run phases, an input signal requires 13 and 11 clock cycles, respectively, to propagate to

TABLE 7
Steps for Computing the $WGP16$ and $WGT16$ in the Serial Implementation of the PB Based WG-16 Design

		Clock Cycle			
		1	2	3	4
Next State	Register 1	$X^{2^{10}+1}$	X^{1057}	$X^{1,057}$	$X^{1,057}$
	Register 2	-	-	Y^{2^2+1}	$Y^{\sum_{i=0}^{3} 2^i}$
	Register 3	-	-	-	-
	Register 4	-	-	-	-
		5	6	7	8
Next State	Register 1	X^{1057}	X^{1057}	$X^{1,057}$	$X^{1,057}$
	Register 2	Y^s	Y^s	Y^s	Y^s
	Register 3	-	Y^{e_1}	$Y^{e_2} \oplus Y^{e_4}$	Y^{e_3}
	Register 4	-	$1 \oplus Y \oplus Y^{e_1}$	$1 \oplus Y \oplus Y^{e_1} \oplus Y^{e_2} \oplus Y^{e_4}$	$1 \oplus Y \oplus Y^{e_1} \oplus Y^{e_2} \oplus Y^{e_3} \oplus Y^{e_4}$

$X = A_{i+31}$ *and* $Y = 1 \oplus X^{1,057}$. $e_1 = 2^{11} + 1$, $e_2 = 2^{11} + 2^6 + 1$, $e_3 = -2^{11} + 2^6 + 1$, $e_4 = 2^{11} + 2^6 - 1$, *and* $s = 2^5 - 1$.

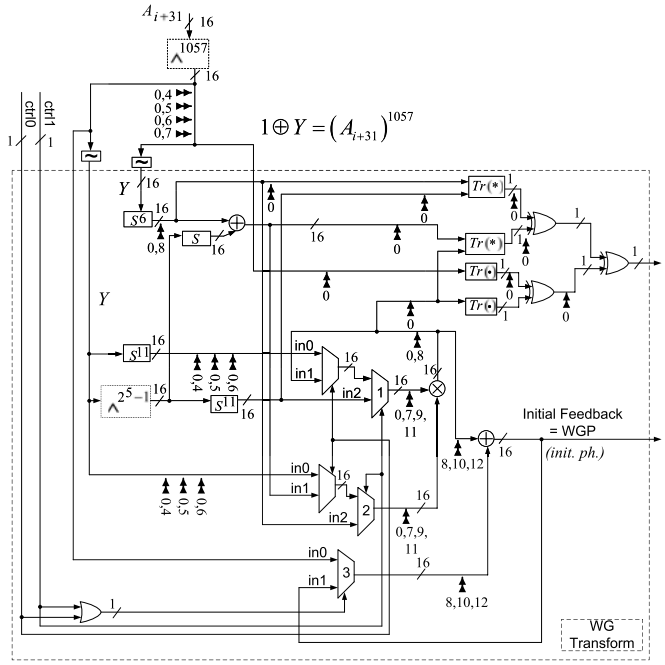

Fig. 14. Pipelined version of the PB based architecture of the WG-16 transform. The double headed arrows indicate the location of the inserted registers for the pipeline. The number under an arrow indicates the clock cycle which triggers it during initialization phase. A zero under an arrow indicates that the register is enabled during the run phase.

the output of the transform/permutation. Therefore, for the initialization phase, the lfsr_clk signal triggers the LFSR once every 13 cycles. This means that the initialization phase takes a total of $13 \times 64 = 832$ clock cycles. Also, the multiplexers' outputs in Fig. 14 are controlled through signals ctrl0 and ctrl1 during the initialization and the run phases. For the run phase, an output enable signal EO is used to enable the keystream output after the first 11 clock cycles. The following section presents the FSM and show how the different control signals are derived.

4.6.2 FSM for the Pipelined WG-16

The FSM for the pipelined version of the WG-16 is obtained from the one introduced in Section 3.5.2, where a 5-bit binary counter and a 13-bit 1-hot counter replace the 11-bit 1-hot counter and the 12-bit 1-hot counter, respectively. The 2-bit binary counter is clocked once every time the 5-bit binary counter completes 32 counts, during load and initialization. From column of Fig. 14 in Table 3, the loading phase takes 32 clock cycles followed by the initialization phase which stays for 832 clock cycles, then starts the run phase. The 13-bit 1-hot counter expands the initialization phase to a total of 832 clock cycles. At the end of each computation of the $WGP16$ (13 clock cycles), the LFSR is shifted once. The $WGP16$ computations are controlled through the two signals ctrl0 and ctrl1, which are generated by the 13-bit counter (Fig. 15). Signal ctrl0 is set during clock cycles 8 and 9, while signal ctrl1 is set during clock cycles 10, 11, and 12. These two signals always reset throughout the run phase. Signals ph0 and ph1 select the LFSR's input. These are derived from the output of the 2-bit binary counter according to Table 3. After 11 clock cycles from the

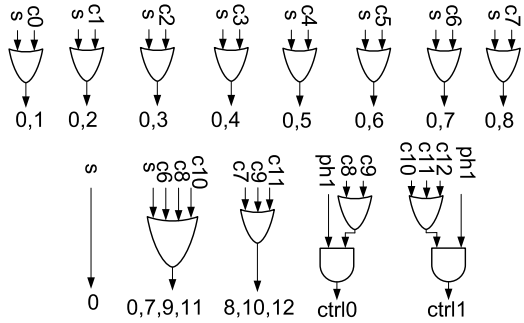

Fig. 15. Generating the clock enable signals and, ctrl0 and ctrl1 signals for the pipelined version of the WG-16. Signal s is shown in Fig. 9a, and c_i, $0 \le i \le 12$ and d_i, $0 \le i \le 5$, are the outputs of the 13-bit and the 6-bit 1-hot counters, respectively (see Section 4.6.2).

start of the run phase, the output of the cipher is enabled by means of the 13-bit counter. The clock enable signals are derived from the outputs of the 13-bit counter during initialization and from the outputs of the 2-bit counter (signal s is shown in Fig. 9a) during the run phase, as can be seen from Fig. 15.

5 Implementation Results and Comparisons

This section presents speed and area results based on ASIC implementations for the nine different proposed designs. The space and speed trade offs concerning the standard, pipelined, and serial versions of the proposed PB based WG$(29, 11)$ and WG-16 designs, are examined and compared to the counterparts.

5.1 ASIC Implementations

In Table 8, we present the speed and area readings for the nine WG designs which we have proposed, based on the ASIC implementations. The ASIC implementations provide speed and area results for the 65 nm CMOS technology with medium effort for optimizations using Synopsys Design Vision [29]. The results are based on Design Vision's estimate of area and clock speed prior to place-and-route. The PB realizations are accomplished using the multiplier presented in [25] for both the WG$(29, 11)$ and the WG-16. The WG-16 has been also realized using the Karatsuba multiplier [30]. We use the VHDL implementations presented in [31] for these two multipliers. Table 8 presents the area and speed results for the ASIC implementations of the different designs. The results for the hardware design of the WG$(29, 11)$ which is proposed in [15] are based on theoretical analysis. In addition, the results for the WG$(29, 11)$ design in [10] are reported in [16]. For the WG-16 which is presented in [22], the results are reported for post place and route.

5.2 Results and Comparisons

As shown in Table 8, the space complexity of the proposed standard WG$(29, 11)$ is reduced, w.r.t the ones previously presented in [10], [16], and the normalized throughput is improved. While the proposed standard WG$(29, 11)$ design shows higher throughput compared to the one in [10], it reports a slightly lower throughput compared to the type-II ONB based design presented in [16]. The WG design

TABLE 8
Results Obtained for Area and Speed from the ASIC Implementations

Implementation	Basis	WG Transform Architecture	Multiplier	Technology	GE	Speed (MHz)	TP ($Mbps$)	Normalized Throughput ($Kbps/Gate$)
SNOW 3G [32]	-	-	-	90 nm	34,000	-	1,900	55.88
SNOW 3G [33]	-	-	-	130 nm	25,016	249	7,900	315.97
ZUC [34]	-	-	-	65 nm	10,000	-	1,500	150
Grain128 (1-bit output version) [35]	-	-	-	130 nm	1,857	926	926	499
Trivium (1-bit output version) [35]	-	-	-	130 nm	2,599	358	358	138
Mickey128 [35]	-	-	-	130 nm	5,039	413	413	82
WG(29,11) [10]	ONB	Standard	[36]	65 nm	33,200	144	144	4.34
WG(29,11) [15]	-	Look-up Table (ROM)	-	-	319 Registers + 9000 XORs + 2^{29} ROM bits	-	-	-
WG(29,11) [16]	ONB	Standard	[36]	65 nm	19,900	224	224	11.26
WG(29,11) (This work, Fig. 4a)	PB	Standard	[25]	65 nm	17,165	202	202	11.77
WG(29,11) (This work, Fig. 6)	PB	Serialized	[25]	65 nm	7,050	610	101	14.32
WG(29,11) (This work, Fig. 8)	PB	Pipelined	[25]	65 nm	21,190	917	917	43.28
WG-16 [22]	NB	Pipelined (M_{16}/I_8)	-	65 nm	12,031	552	552	45.88
WG-16 [22]	NB	Pipelined (M_8/I_8)	-	65 nm	12,352	558	558	45.17
WG-16 (This work, Fig. 11a)	PB	Standard	[25]	65 nm	9,103	189	189	20.76
WG-16 (This work, Fig. 11a)	PB	Standard	[30]	65 nm	8,060	193	193	23.94
WG-16 (This work, Fig. 14)	PB	Pipelined	[25]	65 nm	11,795	1149	1,149	97.41
WG-16 (This work, Fig. 14)	PB	Pipelined	[30]	65 nm	10,681	1370	1,370	128.26
WG-16 (This work, Fig. 12)	PB	Serialized	[25]	65 nm	5,267	680	113	21.45
WG-16 (This work, Fig. 12)	PB	Serialized	[30]	65 nm	5,026	714	119	23.67

GE denotes Gate Equivalence in terms of number of NAND gates. TP denotes the throughput.

presented in [15] requires a number of ROM bits which is exponential in m (the dimension of the binary extension field). For the WG$(29, 11)$, this realization requires 2^{29}-bits of ROM in addition to $9{,}000$ XORs and 319 registers, as can be seen from Table 8. On the other hand, the space complexities of the proposed designs are based on the area of the multiplier, which is quadratic in m. For high speed applications, the throughput which is reported in Table 8 for the proposed pipelined version of the PB based WG$(29, 11)$ design is almost 4.5 times compared to the proposed standard one. This comes at an expense of almost 23 percent increase in the space complexity. On the other hand, for area constrained applications, the serial version shows up to 59 percent decrease in the space complexity compared to the standard design, according to the results in Table 8. This comes at the expense of reducing the throughput to the half.

In Table 8, the Karatsuba based PB implementations of the standard, pipelined, and serial WG-16 show optimal readings for throughput, space, and normalized throughput, compared to the same realizations using the multiplier in [25]. In the same table, in comparison with the pipelined WG-16 implementations presented in [22], the proposed pipelined PB based WG-16 demonstrates almost 2.5 times the throughput with even less space complexity. In addition, for low area applications, the serial version shows up to 42 percent decrease in the space complexity compared to the standard design. This comes at an expense of around 40 percent decrease in throughput. On the other hand, for high speed requirements, the pipelined version of the PB based WG-16 design increases the throughput by almost seven times compared to the standard one. This comes at an expense of almost 33 percent increase in space complexity.

Moreover, for WG-16, which is proposed by the authors of [17] to overcome the security flaws in the LTE integrity protocols [21], the reported results of the proposed designs in Table 8 clearly show that the different realizations offer bit rates greater than 100 Mbps and, hence, satisfy the LTE's peak bit rate requirements [37]. Although SNOW 3G [33] and ZUC [34] show better normalized throughput readings compared to our WG-16 designs in Table 8, the reported space complexities for the proposed WG-16 (specially, serial instances) are competitive to SNOW 3G and ZUC. Hence, WG-16 is an interesting, low area, candidate for the 4G domain. Table 8 also lists the 1-bit output versions of Grain and Trivium which show better performances compared to the proposed designs of the WG-16. On the other hand, our pipelined version of the WG-16 has higher throughput, and normalized throughput, while our serial WG-16 instance shows a very close area complexity, compared to Mickey128.

If even higher throughput is demanded, one can apply the unfolding technique which is presented in [38] to the proposed pipelined WG$(29, 11)$ and WG-16. In this technique, by implementing multiple transforms, the throughput will increase proportionally. Digit-level field multipliers [39], [40] can be considered if lower area is demanded; however, at the expense of adding more cycles for each multiplication.

6 Conclusion

In this paper, we have proposed for the first time new architectures for efficient computations of the WG stream ciphers using polynomial basis. The proposed architectures require fewer multiplication operations as compared to the WG counterparts. Moreover, we have derived an area efficient method for the direct computation of the trace of the multiplication of two $GF(2^m)$ elements. Unlike the trace method presented in [16] which applies only to type-II ONB, the trace method proposed in this paper applies to any PB. Based on the proposed trace properties, two classes of PB based designs (standard architecture) have been proposed, one for the WG$(29, 11)$ stream cipher and the other one for the WG-16 stream cipher. In addition, a serialized version and a pipelined version, has been proposed for each of the proposed standard designs.

We have realized nine different proposed designs through ASIC implementations using the 65nm CMOS technology. The ASIC implementations show that the proposed PB based WG$(29, 11)$ design achieves better area and normalized throughput results compared to all WG$(29, 11)$ counterparts which use NB. Also, it has been shown that the proposed pipelined PB based WG-16 provides almost double the throughput which is offered by the implementations presented in [22], at even smaller area. In addition, the throughput readings reported for the different designs of the WG-16 stream cipher meet the requirements for the peak bit rate specifications of the 4G mobile technology.

Based on these results, the proposed WG$(29, 11)$ and WG-16 designs using PB are competitive candidates, compared to the previously proposed implementations, for securing mobile and communication systems [4], [5], [41]. Specifically, the proposed WG-16 designs are promising for the 4G communications where the guaranteed randomness properties and security aspects are of significant importance.

Acknowledgments

The authors would like to thank the reviewers for their constructive comments. Also, the authors would like to thank Canadian Microelectronics Corporation (CMC) Microsystems for providing the required infrastructure and CAD tools that have been used in this work. This work was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) Alexander Graham Bell Canada Graduate Scholarships-Doctoral (CGS D) scholarship and Discovery and Discovery Accelerate Supplement (DAS) Grants.

References

[1] Y. Luo, Q. Chai, G. Gong, and X. Lai, "A lightweight stream cipher WG-7 for RFID encryption and authentication," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2010, pp. 1–6.

[2] Adopted Bluetooth core specifications, core version 4.0. (2010, Jun.). Bluetooth Special Interest Group [Online]. Available: https://www.bluetooth.org/

[3] S. Gupta, A. Chattopadhyay, K. Sinha, S. Maitra, and B. Sinha, "High-performance hardware implementation for RC4 stream Cipher," *IEEE Trans. Comput.*, vol. 62, no. 4, pp. 730–743, Apr. 2013.

[4] 3GPP TS 33.401 v11.0.1. 3rd Generation Partnership Project; Technical Specification Group Services and Systems Aspects; 3GPP System Architecture Evolution (SAE): Security Architecture, Jun. 2011 (Release 11).

[5] 3rd Generation Partnership Project; Long Term Evaluation Release 10 and Beyond (LTE-Advanced); Proposed to ITU at 3GPP TSG RAN Meeting, Spain, 2009.

[6] eSTREAM - The ECRYPT Stream Cipher Project. (2005) [Online]. Available: http://www.ecrypt.eu.org/stream/

[7] Y. Nawaz, "Design of stream ciphers and cryptographic properties of nonlinear functions," Ph.D. dissertation, Dept. Electrical Comput. Eng., Univ. Waterloo, Waterloo, ON, Canada, 2007.

[8] G. Gong and Y. Nawaz. (2005). The WG Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Rep. 2005/033, [Online]. Available: http://www.ecrypt.eu.org/stream/wgp2.html

[9] G. Gong and A. Youssef, "Cryptographic properties of the Welch-Gong transformation sequence generators," *IEEE Trans. Inf. Theory*, vol. 48, no. 11, pp. 2837–2846, Nov. 2002.

[10] Y. Nawazand G. Gong, "WG: A family of stream ciphers with designed randomness properties," *Inf. Sci.*, vol. 178, no. 7, pp. 1903–1916, 2008.

[11] L. Chen and G. Gong, *Communication System Security*. Boca Raton, FL, USA: CRC Press, 2012.

[12] H. Wu and B. Preneel, "Resynchronization attacks on WG and LEX," in *Proc. 13th Int. Conf. Fast Softw. Encryption*, 2006, vol. 4047, pp. 422–432.

[13] S. Ronjom and T. Helleseth. (2007). Attacking the filter generator over $GF(2^m)$. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/011, [Online]. Available: http://www.ecrypt.eu.org/stream/papersdir/2007/011.pdf

[14] A. Mirzaei, M. Dakhilalian, and M. Modarres-Hashemi. (2010, Apr.). An improved attack on WG stream cipher. *IJSNS Int. J. Comput. Sci. Netw. Security* [Online]. vol. 10 no. 4, pp. 45–52. Available: http://paper.ijcsns.org/07_book/201004/20100408.pdf

[15] E. Krengel, "Fast WG Stream Cipher," in *Proc. IEEE Region 8 Int. Conf. Comput. Technol. Elect. Electron. Eng.*, Jul. 2008, pp. 31–35.

[16] H. El-Razouk, A. Reyhani-Masoleh, and G. Gong, "New implementations of the WG Stream cipher," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 9, pp. 1865–1878, Sep. 2014.

[17] X. Fan and G. Gong. (2013). Specification of the stream cipher WG-16 based confidentiality and integrity algorithms. Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. CACR 2013-06 [Online]. Available: http://cacr.uwaterloo.ca/techreports/2013/cacr2013-06.pdf

[18] 3GPP Technical Specification Groups [Online]. Available: http://www.3gpp.org/Specification-Groups, 2013.

[19] H. Wu, T. Huang, P. Nguyen, H. Wang, and S. Ling, "Differential attacks against stream cipher ZUC," in *Proc. 18th Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2012, vol. 7658, pp. 262–277.

[20] A. Biryukov, D. Priemuth-Schmid, and B. Zhang, "Differential resynchronization attacks on reduced round SNOW 3G$^{\oplus}$," in *Proc. Int. Conf. e-Business Telecommun.*, 2012, vol. 222, pp. 147–157.

[21] T. Wu and G. Gong, "The weakness of integrity protection for LTE," in *Proc. 6th ACM Conf. Security Privacy Wireless Mobile Netw.*, Budapest, Hungary, Apr. 17–19, 2013, pp. 79–88. Also, appeared as Tech. Rep. CACR 2013–03, 2013, Univ. Waterloo, ON, Canada.

[22] X. Fan, N. Zidaric, M. Aagaard, and G. Gong, "Efficient Hardware Implementation of the Stream Cipher WG-16 with Composite Field Arithmetic," Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. CACR 2013-23, 2013. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2013/cacr2013-23.pdf

[23] C. Lam, M. Aagaard, and G. Gong. (2009). Hardware implementations of multi-output Welch-Gong ciphers. Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. CACR 2011-01 [Online]. Available: http://cacr.uwaterloo.ca/techreports/2011/cacr2011-01.pdf

[24] B. Ansari and M. Hasan, "High-performance architecture of elliptic curve scalar multiplication," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1443–1453, Nov. 2008.

[25] A. Reyhani-Masoleh and M. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over GF(2m)," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 945–959, Aug. 2004.

[26] E. D. Mastrovito. (1989). VLSI designs for multiplication over finite fields GF(2m). *Proc. 6th Int. Conf. Appl. Algebra, Algebraic Algorithms Error-Correcting Codes*, pp. 297–309 [Online]. Available: http://dl.acm.org/citation.cfm?id=646025.676557

[27] *IEEE Standard Specifications for Public-Key Cryptography*, IEEE Standard 1363-2000, p. i, 2000.

[28] The Sage Notebook [Online]. Avaialble: http://www.sagenb.org/, 2013.

[29] Synopsys [Online]. Available: http://www.synopsys.com/, 2008.

[30] A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata," *Soviet Physics-Doklady*, vol. 7, pp. 595–596, 1963.
[31] J. Deschamps, J. Imaña, and G. Sutter. (2009). *Hardware Implementation of Finite-Field Arithmetic*, serIES Electronic engineering. New York, NY, USA: McGraw-Hill Education [Online]. Available: http://books.google.ca/books?id=oBNumAEACAAJ
[32] "CLP-41: SNOW 3G Flow through Core. (2011). Elliptic Technologies [Online]. Available: http://www.elliptictech.com/products-clp-41.php
[33] P. Kitsos, G. Selimis, and O. Koufopavlou, "High performance ASIC implementation of the SNOW 3G stream cipher," presented at the Int. Conf. Very Large Scale Integration, Rhodes Island, Greece, Oct. 13–15, 2008.
[34] (2011). ZUC Key Stream Generator. Elliptic Technologies [Online]. Available: http://www.elliptictech.com/pdf/CLP-410 ZUC Key Stream Generator.pdf
[35] T. Good and M. Benaissa, "Hardware results for selected stream cipher candidates," in *Proc. Workshop Record State Art Stream Ciphers*, 2007, pp. 191–204.
[36] A. Reyhani-Masoleh and M. Hasan, "A new construction of Massey-Omura parallel multiplier over $GF(2^m)$," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 511–520, May 2002.
[37] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. (2012). A close examination of performance and power characteristics of 4G LTE networks. *Proc. 10th Int. Conf. Mobile Syst., Appl., Serv.*, pp. 225–238 [Online]. Available: http://doi.acm.org/10.1145/2307636.2307658
[38] C. Chengand K. Parhi,"High-speed parallel CRC implementation based on unfolding, pipelining, and retiming," *IEEE Trans. Circuits Syst. II*, vol. 53, no. 10, pp. 1017–1021, 2006.
[39] A. Reyhani-Masoleh and M. A. Hasan. (2004, Aug.) Efficient digit-serial normal basis multipliers over binary extension fields," *ACM Trans. Embed. Comput. Syst.* [Online]. vol. 3, no. 3, pp. 575–592. Available: http://doi.acm.org/10.1145/1015047.1015053
[40] A. Reyhani-Masoleh and M. Hasan, "Low complexity word-level sequential normal basis multipliers," *IEEE Trans. Comput.*, vol. 54, no. 2, pp. 98–110, Feb. 2005.
[41] L. Chen, J. Franklin, and A. Regenscheid. (2012, Oct.). Guidelines on hardware-rooted security in mobile devices (Draft) in *Special Publication 800-164*, Nat. Inst. Standards Technol. [Online]. Available: http://csrc.nist.gov/publications/drafts/800-164/sp800_164_draft.pdf

Hayssam El-Razouk received the BEng degree in electrical and computer engineering from Beirut Arab University in 2002, with the first rank, and the MESc degree in electrical and computer engineering from the University of Western Ontario in 2006. Currently, he is working towards the PhD degree at the Department of Electrical and Computer Engineering at Western University, Canada. From 2006 to 2011, he was a software engineer with RedIron Technologies, London, Ontario, Canada. He received the prestigious Natural Sciences and Engineering Research Council of Canada (NSERC) Alexander Graham Bell Canada Graduate Scholarships-Doctoral (CGS D) scholarship (September 2012 to August 2015).

Arash Reyhani-Masoleh received the BSc degree in electrical and electronic engineering from Iran University of Science and Technology in 1989, the MSc degree in electrical and electronic engineering from the University of Tehran in 1991, both with the first rank, and the PhD degree in electrical and computer engineering from the University of Waterloo in 2001. From 1991 to 1997, he was with the Department of Electrical Engineering, Iran University of Science and Technology. From June 2001 to September 2004, he was with the Center for Applied Cryptographic Research, University of Waterloo, where he received a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship in 2002. In October 2004, he joined the Department of Electrical and Computer Engineering, University of Western Ontario, London, Canada, where he is currently a tenured associate professor. His current research interests include fault-tolerant computing, algorithms and VLSI architectures for computations in finite fields, cryptography, and error-control coding. He has been awarded a NSERC Discovery Accelerator Supplement (DAS) in 2010. Currently, he serves as an associate editor for *Integration, the VLSI Journal* (Elsevier). He is a member of the IEEE and the IEEE Computer Society.

Guang Gong received the BS degree in mathematics in 1981, the MS degree in applied mathematics in 1985, and the PhD degree in electrical engineering in 1990, from Universities in China. She received the Postdoctoral Fellowship from the Fondazione Ugo Bordoni, in Rome, Italy, and spent the following year there. She was promoted to an associate professor at the University of Electrical Science and Technology of China. During 1995-1998, she was with several internationally recognized, outstanding coding experts and cryptographers, including Dr. Solomon W. Golomb, at the University of Southern California. He joined the University of Waterloo, Canada in 1998, as an associate professor in the Department of Electrical and Computer Engineering in September 2000. She has been a full Professor since 2004. His research interests include the areas of sequence design, cryptography, and communication security. She has authored or coauthored more than 250 technical papers and two books, *Signal Design for Good Correlation for Wireless Communication, Cryptography and Radar* (2005), coauthored with Dr. Golomb, *Communication System Security* (2012), coauthored with Lidong Chen. He serves/served as associate editors for several journals including associate editor for Sequences for *IEEE Transactions on Information Theory*, and served on a number of technical program committees and conferences as co-chairs or committee members. He received several awards including the Best Paper Award from the Chinese Institute of Electronics in 1984, Outstanding Doctorate Faculty Award of Sichuan Province, China, in 1991, the Premier's Research Excellence Award, Ontario, Canada, in 2001, NSERC Discovery Accelerator Supplement Award, 2009, Canada, and Ontario Research Fund - Research Excellence Award, 2010, Canada, Best Paper Award of IEEE ICC 2012, and IEEE Fellow 2014.