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

Hayssam El-Razouk, Arash Reyhani-Masoleh, and Guang Gong

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 65nm 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 4G specifications.

Index Terms—Finite Fields, Linear Feedback Shift Registers, Polynomial Basis, Pseudo Random Key Generators, Stream Ciphers, Trace Function, WG Transformation.

I. INTRODUCTION

A STREAM cipher is a symmetric key crypto-system which generates a unique key-stream for each seed (secret key), where the plaintext (or ciphertext) is bitwise XORed with the generated key bits 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. This cipher was not advanced to the eSTREAM finalists because of the missing of a hardware realization until 2009 [9]. The WG(29, 11) is based on Linear Feedback Shift Register (LFSR) of length 11 and a 29-bit WG transform. It offers theoretically proved randomness properties [10], [8], [11], [9]. The revised version of the WG(29, 11) [11] does not suffer the chosen IV (Initial Value) attack in [12], [13]. 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 [14]. There are many hardware designs proposed for the WG(29, 11) [8], [7], [11], [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], [11], [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 6 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 6 multipliers. Also, a pipelined and, a serialized, instances of this standard design are presented (see Figure 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 [9]. Accordingly, there is a continuous effort demonstrated by the security specification group of the third generation partnership project (3GPP-TSG) [18] to address the security threats concerning the 4G network domain [17]. The cipher suite of 4G LTE consists of two stream ciphers, SNOW 3G and ZUC, 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, which

H. El-Razouk and A. Reyhani-Masoleh are with the Department of Electrical and Computer Engineering, Western University, London, Canada, e-mail: helrazou@uwo.ca, areyhani@uwo.ca.

G. Gong is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada, e-mail: ggong@uwwaterloo.ca.
offers proved randomness properties such as: long period, balance, ideal 2-level autocorrelation, ideal tuple distribution, and exact linear complexity [17]. The WG-16 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 and then by utilizing the proposed trace method, a PB-based WG-16 design (standard) is proposed which requires 6 multipliers for the WG transform. Also, a pipelined and a serialized versions of this standard design, are presented and for each design both the traditional PB and Karatsuba multipliers are considered (see Figure 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].

The contributions of this paper which include a novel trace method and nine new designs of the WG stream ciphers (three for WG(29, 11) and six for WG-16) are summarized in Figure 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].

The paper is organized as follows. Section II defines the terms, notations, and gives brief background about the WG(29, 11) and the WG-16. Section III presents the proposed WG(29, 11) hardware designs based on the PB. Section IV presents the proposed WG-16 hardware designs based on the PB. Results based on the ASIC implementations are discussed in Section V. Section VI concludes the paper.

II. PRELIMINARIES

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

- $GF(2)$, binary finite field with elements $\{0, 1\}$.
- $GF(2^m)$, binary extension field with $2^m$ elements. $a = (a_0, a_1, \ldots, a_{m-1})$ 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 $GF(2^m) \rightarrow GF(2)$.
- Let $p(x)$ be an irreducible polynomial of degree $m$ over $GF(2)$ and let $p(\alpha) = 0$, then $\{1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{m-1}\}$ is a polynomial basis of $GF(2^m)$ over $GF(2)$.
- The inner product of the vectors $a = (a_0, a_1, \ldots, a_{m-1})$ and $b = (b_0, b_1, \ldots, b_{m-1})$, $a_i, b_i \in \{0, 1\}$, $0 \leq i < m$, is computed as $ab^T = \sum_{i=0}^{m-1} a_i b_i \in \{0, 1\}$, where $T$ denotes the 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 $A_{l+j} = \sum_{i=0}^{l-1} C_i A_{i+j}$, 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.

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

![Figure 2. General Block Diagram for the WG Stream Ciphers.](image)

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 Figure 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 $2^{ml} - 1$ [11], [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 + Initial Feedback is the input to the LFSR in Figure 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 Figure 2, the finite state machine (FSM) controls the three phases of operations.

2) Parameters of the WG(29, 11): The permutation for the WG(29, 11) is

$$WGP29 = 1 \oplus Y \oplus Y^{2^{10}+1} \oplus Y^{2^{20}+2^{10}+1} \oplus Y^{2^{20}−2^{10}+1} \oplus Y^{2^{20}+2^{10}−1},$$

where $Y = 1 \oplus A_{i+10}$ and $A_{i+10}$ is the LFSR’s output. The WG transform is given as follows [8], [11], [9], [16], [23]

$$WGT29 = Tr(WGP29).$$
The reader is referred to Section III-B of this paper for the field and the characteristic polynomials for the \(2(29, 11)\).

3) Parameters of the WG-16: The WG permutation is [17]

\[
\text{WGP16} = 1 \oplus Y \oplus Y^{211 + 1} \oplus Y^{211 + 2^6 + 1} \\
\quad \text{+} Y^{-2^{11} + 2^6 + 1} \oplus Y^{211 + 2^6 - 1},
\]

(3)

where \(Y = (A_{i+31})^{1057} \oplus 1\) and \(A_{i+31}\) is the output of the LFSR. In [22], WGP16 is computed as

\[
1 \oplus Y \oplus Y^{211 + 1} \oplus Y^{211 (2^11 - 1) + 1} \oplus Y^{2^6 (Y^{211 + 1} \oplus Y^{211 - 1})},
\]

(4)

where

\[
Y^{211 - 1} = Y \{(1+2)(1+2^7)(1+2^8)\} \{1+2^5\} + 2^{10},
\]

It is noted that (4) requires 10 multiplications (including 2 for computing \((A_{i+31})^{1057}\)). The WG transform is \(WGT16 = Tr (WGP16)\). The characteristic polynomial of the WG-16's LFSR\(^1\) is [17]

\[
Z^{32} \oplus Z^{31} \oplus Z^{22} \oplus Z^9 \oplus \omega^{11}
\]

(5)

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

\[
x^{16} + x^5 + x^3 + x^2 + 1.
\]

(6)

III. ARCHITECTURES OF THE WG\( (29, 11)\) STREAM CIPHER

The WG\( (29, 11)\) uses exponentiation over \(GF\( (2^{29})\), and therefore, an Optimal Normal Basis (ONB) was assumed to be more efficient, compared to other representations, due to the free cost of squaring operations [8], [11], [16]. The authors of [8] propose a direct design of the WG\( (29, 11)\) based on ONB using 7 multiplications and an inversion over \(GF\( (2^{29})\), where the field inversion requires 6 multiplications and 28 squarings in \(GF\( (2^{29})\) [24]. In [11], the authors reduce the field multiplications to only 9, compared to [8]. 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 6 field multiplications.

In this paper, we propose three PB-based designs for the WG\( (29, 11)\) cipher. The three 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.

A. Formulation of the WG Transform

Since replacing \(\left(\left(Y^{2^{20} - 2^{10} + 1}\right)\right)\) with \(\left(\left(Y^{2^{20} - 2^{10} + 1}\right)^{2^{20}}\right)\) in (1) does not affect \(Tr (WGP29)\), the transform of the WG\( (29, 11)\) becomes

\[
WGT29 =
\]

\[
Tr \left(1 \oplus Y \oplus Y\left(2^{25}\right)^{2^5}\right) +
\]

\[
Tr \left(\left(Y^{2^{25}}\right)^{2^{10}} \left(Y\left(2^{25}\right)^{2^5} \oplus Y^{2^{10} - 1} \oplus Y^{2^{10} - 1}\right)^{2^{30}}\right),
\]

(7)

where \(Y = A_{i+10} \oplus 1\). 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 \(Y^{2^{20} - 1}\) as follows:

\[
Y^{2^{10} - 1} = \left(\left(Y^{2^{5} + 1}\right)^{2^{10}} \left(Y^{2^{5} + 1}\right)^{2^{10}} \left(Y^{2^{5} + 1}\right)^{2^{10}}\right).
\]

(8)

The following subsection introduces the WG\( (29, 11)\)'s design parameters.

B. 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.

1) Field Polynomial and Squaring Matrices: To compute (7) and (8), field multiplications and squarings are used. The complexities of these finite field 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\( (2)\)

\[
t_1 (x) = x^{29} + x^2 + 1,
\]

(9)

and its reciprocal function \(t_2 (x) = x^{29} (t_1 (x^{-1})) = x^{29} + x^{27} + 1\). Between these two polynomials, (9) offers multiplication and squaring operations with lower space complexities.

In the original design of the WG\( (29, 11)\) [8] and all reported schemes to date [11], [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 multiplication using PB requires lower complexity than the one using NB. Specifically, a trinomial-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 squarings using PB representation.

\footnote{For the field polynomial (6), the multiplication with the constant \(\omega^{11}\) in (5) requires only 33 XOR gates and a delay of \(2T_X\).}
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, $v$ is obtained as $v = aS$, where $S$ is the binary $m \times m$ squaring matrix whose entries are either 0 or 1 [27]. In general, $W = A^2$ is obtained as $w = aS^e$. This formulation involves $m$ inner products $aS^e_j$, where $S^e_j$ denotes the $j$-th column vector of $S^e$, $0 \leq j < m$. Let $N_X$ denote the number of XOR gates. Then, the hardware realization of $aS^e$ requires $N_X = \sum_{h(S^e_j)>1,0<j<m} (H(S^e_j) - 1)$ and $T_{S^e} = \lceil \log_2(\theta) \rceil T_X$, where $T_{S^e}$ is the propagation delay for computing $aS^e$. $H(\Omega)$ is the Hamming weight of a vector $\Omega$, and $\theta = \max_{h(S^e_j)>1} \{ H(S^e_j) \mid 0 \leq j < m \}$.

For the PB defined by (9), the squaring matrix $S$ is shown in Figure 3. Table I lists the space and time complexities, before and after signal reuse, for the different squaring matrices used in the $\text{WG}(29, 11)$’s implementations.

### Table I

The space and time complexities of the different squaring matrices used in the $\text{WG}(29, 11)$ stream cipher. PD = Propagation Delay.

<table>
<thead>
<tr>
<th>No Sig Reuse</th>
<th>XOR</th>
<th>PD</th>
<th>Sig Reuse</th>
<th>XOR</th>
<th>PD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{S}^{30}$</td>
<td>15</td>
<td>$T_X$</td>
<td>15</td>
<td>$T_X$</td>
<td></td>
</tr>
<tr>
<td>$\text{S}^4$</td>
<td>27</td>
<td>$2T_X$</td>
<td>91</td>
<td>$2T_X$</td>
<td></td>
</tr>
<tr>
<td>$\text{S}^8$</td>
<td>118</td>
<td>$3T_X$</td>
<td>65</td>
<td>$3T_X$</td>
<td></td>
</tr>
<tr>
<td>$\text{S}^8$</td>
<td>232</td>
<td>$27T_X$</td>
<td>97</td>
<td>$27T_X$</td>
<td></td>
</tr>
<tr>
<td>$\text{S}^8$</td>
<td>238</td>
<td>$27T_X$</td>
<td>214</td>
<td>$27T_X$</td>
<td></td>
</tr>
</tbody>
</table>

2) **Characteristic Polynomial of the LFSR:** A primitive characteristic polynomial of degree 11 over $GF(2^{29})$ is required in order for the $\text{WG}(29, 11)$ to produce key-streams with maximal period of $2^{219} - 1$ [8], [11], [9]. For space efficiency, the following primitive pentanomial is selected

$$Z^{11} \oplus Z^6 \oplus Z^2 \oplus Z \oplus \alpha \quad (10)$$

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 $\{ A_i, 0 \leq i \leq 2^{319} - 1 \}$ denote the sequence generated by (10). According to [16], the following recurrence relation generates the sequence $\{ B_i = A_i \oplus 1, 0 \leq i \leq 2^{319} - 1 \}$

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

where $\{ B_i = A_i \oplus 1, 0 \leq i \leq 10 \}$ is the initial state of the LFSR. By constructing the LFSR based on (11) instead of (10), 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 $\alpha$ (a constant), and one NOT gate (for addition with $\alpha^2$).

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 $Tr(A) = \alpha^T$, where $\tau = (\tau_0, \tau_1, \ldots, \tau_{m-1})$ is a unique and a constant $m$-bit vector such that $\tau_i = Tr(\alpha^i) \in GF(2)$, $0 \leq i < m$ and $f(\alpha) = 0$ [27]. Therefore, for the PB $\{ 1, \alpha, \alpha^2 \}$ which is defined by (9), one obtains $\tau_i = 1$ for $i \in \{ 0, 27 \}$ and $\tau_i = 0$ otherwise. Thus,

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

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 used to optimize the space complexity of the PB based implementations of the $\text{WG}(29, 11)$ and the $\text{WG}$-16 stream ciphers used in this paper (see Sections III-C and IV-D).

**Proposition 1.** Consider the $m$-bit trace vector $\tau = (\tau_0, \ldots, \tau_{m-1})$, $\tau_i = Tr(\alpha^i)$, where $\alpha$ 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:

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

where $Q_{(m-1)\times m} = [q_{k,i}]$ is the reduction matrix and, $U_{(m-1)\times m} = [u_{k,j}]$ and $L_{m\times m} = [l_{i,j}]$ are as follows [25]

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

and

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

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

Figure 3. The matrix $S$ for $\text{WG}(29, 11)$. 

and after signal reuse, for the different squaring matrices used in the $\text{WG}(29, 11)$’s implementations.
Proof: Let \( b = (b_0, \ldots, b_{m-1}) \) and \( c = (c_0, \ldots, c_{m-1}) \) be the vector representations of \( B \) and \( C \), respectively, then, from [25] one has
\[
c = Lb^T + Q^T Ub^T, \tag{14}
\]
where \( Q^T \) is the transpose of \( Q \). We have the following computations
\[
Tr(C) = c\tau^T = (Lb^T)^T \tau + (Q^T Ub^T)^T \tau^T
\]
\[
= \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} l_{i,j}b_j \tau_i + \sum_{i=0}^{m-1} \sum_{j=0}^{m-2} q_{k,i}u_{k,j}b_j \tau_i
\]
\[
= \sum_{i=0}^{m-1} \sum_{j=0}^{i} l_{i,j}b_j + \sum_{i=0}^{m-1} \sum_{j=k+1}^{m-1} q_{k,i}u_{k,j}b_j,
\]
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 \( L \) and \( U \), 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 \( n = \sum \tau_{i,j} (i + 1) \) is the upper bound of the number of the terms \((a_{i,j} b_j)\) and \((a_{m-j+k} b_j)\) in (13). It is noted that if \( \tau \) and \( Q \) 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) stream cipher 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:
\[
Tr(AB) = (a_0 + a_{27})b_0 + \sum_{j=1}^{25} (a_{27-j} + a_{29-j})b_j +
\]
\[
(a_1 + a_{26})b_{26} + \sum_{j=26}^{27} (a_{27-j} + a_{29-j} + a_{54-j})b_j,
\]
\[
\tag{15}
\]
Proof: It is noted that the trace has only two nonzero components, \( \tau_0 \) and \( \tau_{27} \) (see Subsection III-B3). We have computed the \( Q \) matrix for the field polynomial (9) and found that the only nonzero entries in the 1-st and the 28-th columns in this matrix are \( q_{0,0}, q_{27,0}, q_{25,27}, \) and \( q_{27,27} \). Hence, (15) results from substituting these \( Q \) values in (13).

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

C. Architecture and FSM

1) Architecture of the WG(29,11) Stream Cipher: The PB based architecture of the WG(29, 11) stream cipher (PB defined by (9)), according to the WGT29 formulations in (7) and (8), and the linear recurrence (11), is shown in Figures 4a and 4b. The squaring matrices are implemented using the signal reuse constructions, the complexities of which are presented in Table I. The complement operator, i.e. \( \sim \), invert the first bit of the input, which requires only one NOT gate. Notice that \( \alpha B_i \) 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 Figure 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 (III-C2)). 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 Figure 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 \( WGP_{29} \) in (1) can be written as \( Tr \left( 1 + Y + Y^{r_5} + Y^{r_6} + Y^{r_7} + Y^{r_8} \right) \), and is recovered using serial computation over 3 clock cycles as described in Table II. During the initialization phase, the LFSR shifts once every 3 clock cycles and loads its leftmost cell with \( IF \oplus \hat{LF} \), where \( \hat{LF} \) is the complement of the original linear feedback given by (10).

### Table II

**Computation of the IF = WGP_{29} Signal over 3 Clock Cycles during the Initialization Phase.** Ctrl0 and Ctrl1 are generated by the FSM. \( WGP_{29} \) is the next state of Register 2 in stage 3. NEXT STATE OF REGISTER 3 IS ALWAYS \( Y^g \). ROWS ARE LISTED IN ORDER OF COMPUTATION STAGES (FIRST TO LAST).

<table>
<thead>
<tr>
<th>ctrl0</th>
<th>ctrl1</th>
<th>Output</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( Y_r^{20} )</td>
<td>( Y )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( Y_r^{1} \oplus Y^{20} )</td>
<td>( Y_r^{1} \oplus Y \oplus 1 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( Y_r^{20} \oplus Y )</td>
<td>( Y_r^{20} \oplus Y \oplus 1 )</td>
</tr>
</tbody>
</table>

The 11-bit 1-hot counter, generates \( ctrl_0 \) and \( ctrl_1 \), and triggers the clock of the 2-bit counter, every 11 counts, during the loading and initialization. The 3-bit counter, generates \( ctrl_0 \) and \( ctrl_1 \), and triggers the clock of the 11-bit counter as well as the clock of the LFSR, every 3 counts, during initialization.

In the running phase, the LFSR updates its state in each clock cycle, where \( B_{i+10} \) is loaded with \( LF \). In Figure 4a, the keystream bits are obtained from XORing \( Tr \left( 1 + Y + Y^{r_5} \right) \) with \( Tr \left( Y^{r_5} \oplus Y^{r_6} + Y^{r_7} \right) \). \( Tr \left( 1 + Y + Y^{r_5} \right) \) is the output of XORing \( Tr \left( 1 + Y \right) \) and \( Tr \left( Y^{r_5} \right) \). \( Tr \left( 1 + Y \right) \) and \( Tr \left( Y^{r_5} \right) \) are produced by applying operator \( Tr \) (•) to \( 1 + Y \) and \( Y^{r_5} \), respectively. The operator \( Tr \) (•) generates its output according to (12). \( Y^{r_5} \) is generated by multiplying \( Y \) with \( 2^{20} \) in \( GF \left( 2^{29} \right) \) (by setting \( ctrl_0 = \) \( ctrl_1 = 0 \) for the running phase in Figure 4a). \( Y \) is the output \( B_{i+10} \) of the LFSR and \( Y^{20} \) is obtained from the squarer \( S^0 \) operating on \( Y^2 \), which in turn, is available from the generator of \( Y^{20} \) (see Figure 4b). \( 1 + Y \) is the addition of \( Y \in GF \left( 2^{29} \right) \) with the unity element \( 1 = (1, 0, 0, \ldots, 0) \) represented w.r.t. PB. Thus, \( 1 + Y \) results from inverting the least significant bit of \( B_{i+10} \) by the complement operator \( \sim \). \( Tr \left( Y^{r_5} \oplus Y^{r_6} + Y^{r_7} \right) \) is generated by applying (15) to \( Y^{20} \) and \( Y^{r_5} \oplus Y^q \). Signal \( Y^{r_5} \oplus Y^q \oplus Y^{20} \) is the bitwise XOR of \( Y^{r_5}, Y^q, \) and \( Y^{20} \), where \( Y^{20} \) is obtained from \( S^{10} \) operating on \( Y^q \) and \( Y^q \) is generated as presented in Figure 4b.

The Finite State Machine (FSM): The architecture of the FSM is shown in Figure 5. The FSM controls the inputs to the LFSR during the three phases of operations through generating the signals \( ph_0 \) and \( ph_1 \). As presented in Table III for the column of Figure 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 \( ph_0 \) and \( ph_1 \). 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 \( ctrl_0 \) and \( ctrl_1 \), and triggers the clock of the 11-bit counter as well as the clock of the LFSR, every 3 counts, during initialization.

### Table III

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

### D. Serialized Implementation of the PB Based WG \( (29, 11) \) Design

1) Architecture of the Serialized WG \( (29, 11) \): Here we present a serialized \( WGP_{29}/WGT_{29} \) 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 Figure 4a; however, the WG transform and the FSM are modified. Figure 6 presents the proposed serial \( WGP_{29}/WGT_{29} \) 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 IV over seven clock cycles.

It is noted that no changes are required for the loading phase of the serial WG \( (29, 11) \). However, in the architecture of Figure 6, an initialization round takes 7 clock cycles to generate...
Table IV
Steps for computing the WGT29 and WGT’29 in the serial implementation of the PB based WG(29, 11) design, $Y = 2^{10} + 1$ is the LFSR’s output (see Figure 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 $q = 2^{10} - 1$.

<table>
<thead>
<tr>
<th></th>
<th>Clock Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Register 1</td>
<td>Y$r^{14}$</td>
</tr>
<tr>
<td>Register 2</td>
<td>-</td>
</tr>
<tr>
<td>Register 3</td>
<td>1 ⊕ Y ⊕ Y$r^{14}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Clock Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Register 1</td>
<td>Y$r^{14}$</td>
</tr>
<tr>
<td>Register 2</td>
<td>Y$r^{14}$</td>
</tr>
<tr>
<td>Register 3</td>
<td>1 ⊕ Y ⊕ Y$r^{14}$</td>
</tr>
</tbody>
</table>

the WGP29 signal. The LFSR is updated at the 8-th clock cycle. During the run phase, a stream bit is produced every 6 cycles. During these two phases, the multiplexers provide the inputs to the multiplier and the adder. The multiplexers’ inputs are multiplexed through the selectors $m_0$ - $m_4$ during the clock cycles of the computations. The 3 registers are clocked as it is specified by the clocking table which is shown in Figure 6. The clocking of the different registers is enabled by means of clock enable signals (see Section III-D2). In this design, the lfsr_clk signal in Figure 7a is required in order to clock the LFSR once every 1 clock cycle, 8 clock cycles, and 6 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 Figure 6 are presented in the corresponding column of Table III. Moreover, the signal EO in Figure 7a is used to enable the keystream output every 6 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.

2) FSM for the Serialized PB based WG(29, 11): Figure 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 III for Figure 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 $ph_0$ and $ph_1$ according to Table III. 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 Figure 6), and triggers the clock of the 11-bit counter as well as the clock of the LFSR, every 8 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 6 counts. From the starting of the run phase, the 6-bit counter enables the output of the cipher every 6 counts.

E. Pipelined Implementation of the PB Based WG(29, 11) Design

1) Architecture: Figures 8 and 4b present the pipelined version of the PB based implementation of the WGT’29. 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 III-E2).

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 Figure 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 III for Figure 8. Also, the multiplexers’ outputs in Figure 8 are controlled through the signals $ctrl_0$ and $ctrl_1$ (Figure 9b) during the initialization and the run phases. For the run phase, an output enabled signal, EO in Figure 9a, is used to enable the keystream output after the first 10 clock cycles. The following section presents the FSM and show how
Figure 8. Pipelined version of the PB based architecture of the WG(29).

Figure 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.

IV. ARCHITECTURES OF THE WG-16 STREAM CIPHER

The WG-16 stream 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 WG-16 is secure and resists to all known attacks [17]. In addition, it has proved randomness properties such as: long period, balance, ideal 2-level autocorrelation, ideal tuple distribution, and exact linear complexity [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.

Here, we propose a new formulation of the WG-16 permutation which requires 8 multiplications compared to 10 in the formulation of [22]. Based on this formulation, and using the trace property in (13), this section present six hardware architectures of the WG-16, based on the PB representation of its field elements 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 introduced. Then, the proposed standard architecture of the WG-16 is shown. The section ends by presenting a serialized version and a pipelined version of the standard design.
A. Formulations of the WG-16 Permutation and the WG-16 Transformation

The WG16’s formulation in (4) requires 10 multiplications when the field elements are represented in the PB. In the following, a new WG16 formulation is derived which requires 8 multiplications.

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

\[
WGP16 = 1 \oplus Y \oplus Y^{2^{i+1}} \oplus Y^{2^{i+1}}(2^{i+1})^2 + 2^6 \\
\quad \oplus Y^{2^{i+1}} \left( Y^{2^6} \oplus Y^{2^{i+1}} \right),
\]

(16)

where \( Y = (A_{i+31})^{1057} \oplus 1, A_{i+31} \) is the output of the LFSR described by (5), and \( Y^{2^{i+1}} \) is computed as follows

\[
Y^{2^{i+1}} = \left( Y^{2^{i+1}} \right)^{2+1} Y^{2^i}.
\]

(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 8 \( GF(2^{16}) \) multiplications: 1 for computing \( Y^{2^{i+1}} \), 3 for computing \( Y^{2^i} \), 1 for computing \( Y^{2^{i+1}}(2^{i+1})^2 + 2^6 \), 1 for computing \( Y^{2^{i+1}} \left( Y^{2^6} \oplus Y^{2^{i+1}} \right) \), and 2 for computing \( 1 \oplus Y = (A_{i+31})^{1057} \). In addition to this, (16) requires 7 squarings and 5 \( GF(2^{16}) \) additions. For the transform, the computation of the trace of \( WGP16 \) is required. Section IV-C presents a method which reduces the number of multiplications in the WGT16 to only 6 through computing \( Tr \left( Y^{2^{i+1}}(2^{i+1})^2 + 2^6 \right) \) directly from \( Y^{2^{i+1}}(2^{i+1})^2 + 2^6 \), and \( Tr \left( Y^{2^{i+1}} \left( Y^{2^6} \oplus Y^{2^{i+1}} \right) \right) \) directly from \( Y^{2^{i+1}} \) and \( Y^{2^6} \). Without performing the multiplications.

B. Squaring Matrices and Trace Vector

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

1) Squaring Matrices: Figure 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 V lists the space and propagation delay complexities of the different squaring matrices used in the WG-16 implementation (before and after signal reuse).

2) Trace Vector: The trace vector for the PB \( \{1, \alpha, \ldots, \alpha^{15}\} \) which is 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 III-B3). Thus, for \( A \in GF(2^{16}) \)

\[
Tr(A) = a_{11} + a_{13}.
\]

(18)

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

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

**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:

\[
Tr(AB) = \sum_{j=9}^{13} (a_{11-j} + a_{13-j}) b_j + \sum_{j=12}^{13} a_{13-j} b_j + \sum_{j=7}^{13} a_{22-j} b_j + \sum_{j=11}^{13} a_{25-j} + a_{26-j} b_j + \sum_{j=14}^{15} a_{22-j} + a_{25-j} + a_{26-j} + a_{29-j} b_j.
\]

(19)

**Proof:** \( \tau \) has only two nonzero components, \( \tau_{11} \) and \( \tau_{13} \) (see Subsection IV-B2). We have computed the \( Q \) (reduction) matrix for the field polynomial (6). Accordingly, the only nonzero entries for the 12-th and the 14-th columns in this matrix are \( q_{6,11}, q_{8,11}, q_{9,11}, q_{11,11}, q_{9,13}, q_{10,13}, q_{11,13}, \) and \( q_{13,13} \). Hence, by replacing these values of \( \tau_i \) and \( q_{k,i} \) in (13), we get (19).

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

D. Architecture and FSM

1) Architecture of the WG-16 Stream 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 \( s = 2^{5} - 1 \). Figures 11a, 11b, and 11c present the proposed architecture of the WG-16 stream cipher according to the WG permutation 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 V.

In Figure 11a, the FSM controls the components of the cipher during the different phases of operation. This is accomplished through signals lsfr_clk, ph0, ph1, ctrl0 and ctrl1 (see Section IV-D2 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 Figure 11a. This is due to the generation of \( Tr (Y^{e_2} \oplus Y^{e_4}) \) directly from \( Y^{2^{i+1}+1} \) and \( Y^{2^i \oplus Y^{2^k}} \) 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 3 clock cycles, during initialization, as described in Table VI. The signals \((A_{i+31})^{1057} = 1 \oplus Y \) and \( Y^{2^{-1}} \) are generated as shown in Figures 11c and 11b, respectively. During the initialization phase, the lsfr_clk signal triggers the LFSR every 3 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 from the LFSR feedback function. 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), respectively. \( Y^{e_1} \) is generated by multiplying \( Y \) with \( Y^{2^{i+1}} \) in GF \((2^{16})\). \( Y \) is generated by complementing the least significant bit of \((A_{i+31})^{1057} \), and \( Y^{2^k} \) is obtained from the squarer \( S^{11} \) operating on \( Y \). \( 1 \oplus Y \) is simply \((A_{i+31})^{1057} \).

\( Tr (Y^{e_3}) \) is generated by applying (19) to \( Y^{2^{i+1}}(2^{-1}) \) and \( Y^{2^k} \). The signal \( Y^{2^k} \) is the result of \( S^{11} \) operating on \( Y \). The signal \( Y^{2^{i+1}}(2^{-1}) \) is the result of \( S^{11} \) operating on \( Y^{2^k-1} \). \( Tr (Y^{e_2} \oplus Y^{e_4}) \) is generated by applying (19) to \( Y^{2^{i+1}+1} \) and \( Y^{2^{(2^{-1})}} \). The signal \( Y^{2^{(2^{-1})}} \) is the result of \( S^{11} \) operating on \( Y^{2^k-1} \), while signal \( Y^{2^{(2^{-1})}} \oplus Y^{2^k} \) is the bitwise XOR of \( Y^{2^{(2^{-1})}} \) and \( Y^{2^k} \).

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 IV-D2). 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 Figure 11a in Table III, 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 3 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 3 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.

E. Serialized Implementation of the PB Based WG-16 Design

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 of Figure 11a. Figure 12 presents the proposed architecture for the serial computation of the WG-16 transform. In this architecture, the WGP16 is computed over 8 clock cycles (during initialization phase) while the WGT16 is computed over 6 clock cycles (run phase). The design uses only one field multiplier. The computations are accomplished according to Table VII.
Figure 12. Architecture of the serial implementation for the PB based design of the WG-16. In this architecture, $X = A_{16:31}$ and $Y = 1 \oplus X^{1057}$, $e_1 = 2^{11} + 1$, $e_2 = 2^{11} + 2^9 + 1$, $e_3 = -2^{11} + 2^9 + 1$, $e_4 = 2^{11} + 2^6 - 1$, and $s = 2^8 - 1$.

Table VII

<table>
<thead>
<tr>
<th>Step</th>
<th>Clock Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$X^2$</td>
</tr>
<tr>
<td>2</td>
<td>$X^{1057}$</td>
</tr>
<tr>
<td>3</td>
<td>$X^{1057}$</td>
</tr>
<tr>
<td>4</td>
<td>$X^{1057}$</td>
</tr>
<tr>
<td>5</td>
<td>$Y^2$</td>
</tr>
<tr>
<td>6</td>
<td>$Y^{4096}$</td>
</tr>
<tr>
<td>7</td>
<td>$Y^{4096}$</td>
</tr>
</tbody>
</table>

Table VIII

<table>
<thead>
<tr>
<th>Step</th>
<th>Clock Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$X^2$</td>
</tr>
<tr>
<td>2</td>
<td>$X^{1057}$</td>
</tr>
<tr>
<td>3</td>
<td>$X^{1057}$</td>
</tr>
<tr>
<td>4</td>
<td>$X^{1057}$</td>
</tr>
<tr>
<td>5</td>
<td>$Y^2$</td>
</tr>
<tr>
<td>6</td>
<td>$Y^{4096}$</td>
</tr>
<tr>
<td>7</td>
<td>$Y^{4096}$</td>
</tr>
</tbody>
</table>

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 8 clock cycles to generate the WGP16 signal. The LFSR is updated at the 9-th clock cycle. During the run phase, a stream bit is produced every 6 cycles. During these two phases, the multiplexers provide the inputs to the multiplier and adder. The multiplexers’ inputs are multiplexed through the selectors $m_0$ - $m_3$ during the clock cycles of the computations. The 4 registers are clocked as it is specified by the clocking table which is shown in Figure 12. The clocking of the different registers is enabled by means of clock enable signals (see Section IV-E2). In this design, the FSM’s signal lfsr_clk is required in order to clock the LFSR once every 1 clock cycle, 9 clock cycles, and 6 clock cycles, during loading, initialization, and run phases, respectively. This means that the initialization phase takes a total of $9 \times 64 = 576$ clock cycles. Moreover, an output enable signal EO is used to enable the keystream output every 6 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.

2) **FSM for the Serialized WG-16:** The FSM for the serialized WG-16 is a modified version of the one in Section III-D2. 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 Figure 12 in Table III, the initialization phase takes to 576 clock cycles. Each initialization round takes 9 clock cycles. The LFSR is clocked at the arrival of the 9-th clock cycle by means of the 9-bit 1-hot counter. During the run phase, the LFSR is clocked once every 6 clock cycles by means of the 6-bit counter. The clock enable signals which control the clocking of the registers in Figure 12, and the multiplexers’ selectors $m_0$, $m_1$, $m_2$, and $m_3$ are derived from the signal ph1, the outputs of the 9-bit 1-hot counter (during initialization), and outputs of the 6-bit 1-hot counter (during run phase), as it is shown in Figure 13. The cipher’s output is enabled once every 6 clock cycles during the run phase, through the 6-bit counter.

F. **Pipelined Implementation of the PB Based WG-16 Design**

1) **Architecture:** A pipelined version of the PB based implementation of the WG-16 transform is presented in Figures 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 IV-F2).

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 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 Figure 14 are controlled through the 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.

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 III-E2, 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 Figure 14 in Table III, 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 cthr0 and cthr1, which are generated by the 13-bit counter (Figure 15). Signal cthr0 is set during clock cycles 8 and 9, while signal cthr1 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 III. After 11 clock cycles from the 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 Figure 9a) during the run phase, as can be seen from Figure 15.

V. 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.

A. ASIC Implementations

In Table VIII, 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 65nm 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 VIII 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 [11] are reported in [16]. For the WG-16 which is presented in [22], the results are reported for post place and route.

B. Results and Comparisons

As shown in Table VIII, the space complexity for the standard PB based WG(29, 11) is reduced, w.r.t. the ones previously presented in [11], [16], and the normalized throughput is improved. While the proposed standard PB based WG(29, 11) design shows higher throughput compared to the one in [11], it reports a slightly lower throughput compared to the type-II ONB based design presented in [16]. The WG design 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 223-bits of ROM in addition to 9000 XORs and 319 registers are required, as can be seen from Table VIII. On the other hand, the space complexity of the proposed designs is based on the area
of the multiplier, which is quadratic in \( m \). For high speed applications, the throughput which is reported in Table VIII for the pipelined version of the proposed 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% increase in the space complexity. On the other hand, for area constrained applications, the serial version shows up to 59% decrease in the space complexity compared to the standard design, according to the results in Table VIII. This comes at the expense of reducing the throughput to the half.

In Table VIII, 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% decrease in the space complexity compared to the standard design. This comes at an expense of around 40% 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 7 times compared to the standard one. This comes at an expense of almost 33% increase in the space complexity.

It is noted that the reported space complexities, in Table VIII, for the proposed WG-16 are competitive to those for the SNOW 3G which are presented in [32] and [33]. Compared to ZUC [34], the proposed standard and the serial designs of the WG-16 offers lower area, while the pipelined version shows close values for both area and throughput. In addition, the reported results in Table VIII show that the different realizations of the proposed WG-16 stream cipher meet the LTE’s peak bit rate of 100 Mbps [36]. If even higher throughput is demanded, one can apply the unfolding technique which is presented in [37] to the proposed pipelined schemes of the WG(29, 11) and WG-16. In this technique, by implementing multiple transforms, the throughput will increase proportionally. Digit-level field multipliers [38], [39] can be considered if lower area is demanded; however, at the expense of adding more cycles for each multiplication.

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Basis</th>
<th>Transform Architecture</th>
<th>Multiplier</th>
<th>Technology</th>
<th>GE</th>
<th>Speed (MHz)</th>
<th>TP (MHz)</th>
<th>Normalized Throughput (MHz/Gate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNOW 3G [32]</td>
<td>PB</td>
<td>PB</td>
<td>-</td>
<td>65nm</td>
<td>1900</td>
<td>-</td>
<td>1000</td>
<td>55.88</td>
</tr>
<tr>
<td>SNOW 3G [33]</td>
<td>PB</td>
<td>PB</td>
<td>-</td>
<td>65nm</td>
<td>2916</td>
<td>249</td>
<td>900</td>
<td>315.97</td>
</tr>
<tr>
<td>ZUC [34]</td>
<td>PB</td>
<td>ONB</td>
<td>[35]</td>
<td>65nm</td>
<td>1000</td>
<td>-</td>
<td>1000</td>
<td>5.34</td>
</tr>
<tr>
<td>WG(29, 11) (15)</td>
<td>PB</td>
<td>Look-up Table (ROM)</td>
<td>-</td>
<td>-</td>
<td>319 Registers + 9000 XORs + 259 ROM bits</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WG(29, 11) (18)</td>
<td>ONB</td>
<td>Pipelined</td>
<td>[35]</td>
<td>65nm</td>
<td>1900</td>
<td>224</td>
<td>224</td>
<td>11.46</td>
</tr>
<tr>
<td>WG(29, 11) (This work, Figure 4a)</td>
<td>PB</td>
<td>Standard</td>
<td>[25]</td>
<td>65nm</td>
<td>17165</td>
<td>202</td>
<td>202</td>
<td>11.77</td>
</tr>
<tr>
<td>WG(29, 11) (This work, Figure 6)</td>
<td>PB</td>
<td>Pipelined</td>
<td>[25]</td>
<td>65nm</td>
<td>7050</td>
<td>610</td>
<td>101</td>
<td>14.32</td>
</tr>
<tr>
<td>WG(29, 11) (This work, Figure 8)</td>
<td>PB</td>
<td>Pipelined</td>
<td>[25]</td>
<td>65nm</td>
<td>21190</td>
<td>917</td>
<td>917</td>
<td>43.28</td>
</tr>
<tr>
<td>WG(16 [22]</td>
<td>PB</td>
<td>Pipelined /( \langle \overline{m}/4d \rangle )</td>
<td>-</td>
<td>65nm</td>
<td>12031</td>
<td>552</td>
<td>552</td>
<td>45.88</td>
</tr>
<tr>
<td>WG(16 [22]</td>
<td>PB</td>
<td>Pipelined /( \langle \overline{m}/4d \rangle )</td>
<td>-</td>
<td>65nm</td>
<td>12352</td>
<td>558</td>
<td>558</td>
<td>45.17</td>
</tr>
<tr>
<td>WG-16 (This work, Figure 11a)</td>
<td>PB</td>
<td>Standard</td>
<td>[25]</td>
<td>65nm</td>
<td>9103</td>
<td>189</td>
<td>189</td>
<td>20.76</td>
</tr>
<tr>
<td>WG-16 (This work, Figure 11a)</td>
<td>PB</td>
<td>Standard</td>
<td>[30]</td>
<td>65nm</td>
<td>8060</td>
<td>193</td>
<td>193</td>
<td>23.94</td>
</tr>
<tr>
<td>WG-16 (This work, Figure 14)</td>
<td>PB</td>
<td>Pipelined</td>
<td>[25]</td>
<td>65nm</td>
<td>11795</td>
<td>1149</td>
<td>1149</td>
<td>97.41</td>
</tr>
<tr>
<td>WG-16 (This work, Figure 14)</td>
<td>PB</td>
<td>Pipelined</td>
<td>[30]</td>
<td>65nm</td>
<td>10681</td>
<td>1370</td>
<td>1370</td>
<td>128.26</td>
</tr>
<tr>
<td>WG-16 (This work, Figure 12)</td>
<td>PB</td>
<td>Pipelined</td>
<td>[25]</td>
<td>65nm</td>
<td>3026</td>
<td>806</td>
<td>113</td>
<td>21.45</td>
</tr>
<tr>
<td>WG-16 (This work, Figure 12)</td>
<td>PB</td>
<td>Pipelined</td>
<td>[30]</td>
<td>65nm</td>
<td>5026</td>
<td>714</td>
<td>119</td>
<td>23.67</td>
</tr>
</tbody>
</table>

Table VIII

RESULTS OBTAINED FOR AREA AND SPEED FROM THE ASIC IMPLEMENTATIONS. GE DENOTES GATE EQUIVALENCE IN TERMS OF NUMBER OF NAND GATES (ESTIMATED AREA OF ONE NAND GATE IS 2.08 \( \mu \text{m}^2 \)). TP DENOTES THE THROUGHPUT.

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

VI. CONCLUSION
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 [40], [4], [5]. Specifically, the proposed WG-16 designs are promising for the 4G communications where the guaranteed randomness properties and security aspects are of significant importance.

REFERENCES