New Implementations of the WG Stream Cipher

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

***Abstract*— This paper presents two new hardware designs of the Welch–Gong (WG)−128 cipher, one for the multiple output WG (MOWG) version, and the other for the single output version WG based on type−II optimal normal basis representation. The proposed MOWG design uses signal reuse techniques to reduce hardware cost in the MOWG transformation, whereas it increases the speed by eliminating the inverters from the critical path. This is accomplished through reconstructing the key and initial vector loading algorithm and the feedback polynomial of the linear feedback shift register. The proposed WG design uses properties of the trace function to optimize the hardware cost in the WG transformation. The application-specific integrated circuit and field-programmable gate array implementations of the proposed designs show that their areas and power consumptions outperform the existing implementations of the WG cipher.**

***Index Terms*— Finite fields, linear feedback shift registers (LFSR), normal basis, optimal normal basis (ONB), pseudorandom key generators, stream ciphers, Welch–Gong (WG) transformation.**

I. Introduction

SYNCHRONOUS stream ciphers are lightweight symmetric-key cryptosystems. These ciphers encrypt a plain-text, or decrypt a cipher-text, by XORing the plain-text/cipher-text bit-by-bit with the generated key-stream bits. The key-stream bits are produced using a pseudorandom sequence generator (PRSG) and a seed (secret key). Stream ciphers are heavily used in wireless communication and restricted in resources applications such as 3GPP LTE-Advanced security suite [1], network protocols (Secure Socket Layer, Transport Layer Security, Wired Equivalent Privacy, and Wi-Fi Protected Access) [2], radio frequency identification (RFID) tags [3], and bluetooth [4], to name some.

Traditionally, many hardware-oriented stream ciphers have been built using linear feedback shift registers (LFSRs) and a filter/combiner Boolean function. However, the discovery of algebraic attacks made such a way of design insecure [5]–[8]. Many nonlinear feedback shift registers-based stream ciphers have been proposed in the eSTREAM stream cipher project [9], which have limited theoretical results about their randomness and cryptographic properties [3], and therefore, their security depends on the difficulty of analyzing the design itself [3], [10]. In addition, the arrival of the 4G mobile technology has triggered another initiative for new stream ciphers [11], [12]. The randomness of the keystreams generated by the 4G LTE cryptographic algorithms is, however, hard to analyze and, also, some weaknesses have been discovered [13]–[15].

The Welch–Gong (WG)$(29, 11)$ [29 corresponds to $GF(2^{29})$ and 11 is the length of the LFSR] is a stream cipher submitted to the hardware profile in phase 2 of the eSTREAM project [9]. It has been designed based on the WG transformations [16] to produce key bit-streams with mathematically proved randomness aspects. Such properties include balance, long period, ideal tuple distribution, large linear complexity, ideal two-level autocorrelation, cross correlation with an m-sequence has only three values, high nonlinearity, Boolean function with high algebraic degree, and 1-resilient [10], [17]–[19]. The revised version of the WG$(29, 11)$ [9], [10] does not suffer the chosen initial value (IV) attack [20], [21]. The number of key-stream bits per run is strictly less than the number of key-stream bits required to perform the attack introduced in [22]. In addition, the WG cipher is secure against algebraic attacks [10], [19]. Therefore, the WG$(29, 11)$ is secure and has the randomness properties that cannot be offered by other ciphers and, hence, it has a potential that the WG stream cipher will be adopted in practical applications.

Despite of its attractive randomness and cryptographic properties, few designs have been proposed for the hardware implementations of the WG$(29, 11)$. Gong and Nawaz [18] adopt a direct design using computation in the optimal normal basis (ONB), which requires seven multiplications and an inversion over $GF(2^{29})$. The inversion using Itoh–Tsujii algorithm requires $\left(\lfloor \log_2(28) \rfloor + H(28) - 1\right) = 4+3-1 = 6$ multiplications and 28 squarings in $GF(2^{29})$, where $H(28)$ denotes the Hamming weight of 28 [23]. Nawaz and Gong [10] replaced the inversion operation with a computation of the power $2^k - 1$ that requires four multiplications for $k = \lceil 29/3 \rceil = 10$ and reduced the other seven multiplications of the WG transformation in [18] by one through signal reuse. Krengel [24] uses a look-up based approach that uses 2^{29} bits of ROM. In Lam *et al.* [25], the authors propose a multiple-bit output version of the WG cipher, called multiple output WG (MOWG). The MOWG reduces the hardware cost through signal reuse by removing one multiplier from the WG permutation in [10], whereas it generates $d \leq 17$ output bits. Furthermore, [25] improves the hardware cost and throughput of the cipher through pipelining with reuse techniques. The keystream sequences generated by the MOWG cipher possess many of the WG keystream randomness properties [25].

In this paper, a novel method for computing the trace of a product of two field elements is presented, when the

Manuscript received October 22, 2012; revised February 8, 2013 and May 21, 2013; accepted August 12, 2013. Date of publication September 17, 2013; date of current version August 21, 2014. This work was supported in part by the Natural Sciences and Engineering Council Discovery and in part by the Discovery Accelerate Supplement Grants.

H. El-Razouk and A. Reyhani-Masoleh are with the Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada (e-mail: helrazou@uwo.ca; areyhani@uwo.ca).

G. Gong is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: ggong@uwaterloo.ca).

Digital Object Identifier 10.1109/TVLSI.2013.2280092

representation is the type-II ONB. In addition, two designs are proposed. One for the MOWG cipher and the other one for the WG cipher (that was initially proposed in [18]), demonstrated by application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) implementations. The proposed designs optimize the area by reducing the number of multiplications in the MOWG/WG transforms. This is done through signal reuse for the MOWG and through using the new trace properties for the WG. The ASIC and FPGA implementations of the proposed WG design show significant area and power consumption reductions and an improved speed compared with [10].

This paper is organized as follows. Section II defines the terms, notations, and gives a brief background about the MOWG/WG cipher. Sections III and IV presents the new hardware designs of the MOWG cipher and WG cipher, respectively. Results based on FPGA and ASIC implementations of the new designs are discussed in Section V. Section VI concludes this paper.

II. Preliminaries

This section defines the notations that will be used throughout this paper to describe the WG cipher and its operation. In addition, a brief introduction to the components and operation of this cipher is presented.

1) $GF(2)$, binary finite field with elements 0 and 1.
2) $GF(2^m)$, binary extension field with 2^m elements represented as m-bit binary vectors.
3) $Tr(Z) = Z + Z^2 + \cdots + Z^{2^{m-1}}$, the trace function from $GF(2^m) \longrightarrow GF(2)$.
4) If $\beta \in GF(2^m)$ and $\nabla = \left\{\beta^{2^0}, \ldots, \beta^{2^{m-1}}\right\}$ is a basis of $GF(2^m)$, then ∇ is a NB of $GF(2^m)$ over $GF(2)$.
5) Let $A = (a_0, \ldots, a_{m-1}) \in GF(2^m)$, and p is a positive integer, then, in NB.
   a) $A^{2^p} = A \gg p$, represents the right cyclic shift of the coordinates of A, with respect to NB, p-times.
   b) $A^{2^{-p}} = A \ll p$, represents the left cyclic shift of the coordinates of A, with respect to NB, p-times.
6) In NB, the addition of 1 to an element can be done by complementing the bits of that element.
7) The trace of any $GF(2^m)$ element $Z = \sum_{i=0}^{m-1} z_i \beta^{2^i}$ represented in NB is given by

$$Tr(Z) = \sum_{i=0}^{m-1} z_i \tag{1}$$

8) $\oplus$ represents the bit-wise addition operator (XOR) in $GF(2^m)$.
9) The inner product of two m-bit vectors, $A = (a_0, \ldots, a_{m-1})$ and $B = (b_0, \ldots, b_{m-1})$, is computed as $A \cdot B = \sum_{i=0}^{m-1} a_i b_i \in \{0, 1\}$.
10) $C(Z) = Z^l \oplus \sum_{i=0}^{l-1} C_i Z^i$, $C_i \in GF(2^m)$ is the characteristic polynomial of an l-stages LFSR over $GF(2^m)$, from which the recurrence relation is obtained as

$$A_{j+l} = \sum_{i=0}^{l-1} C_i A_{i+j} \tag{2}$$

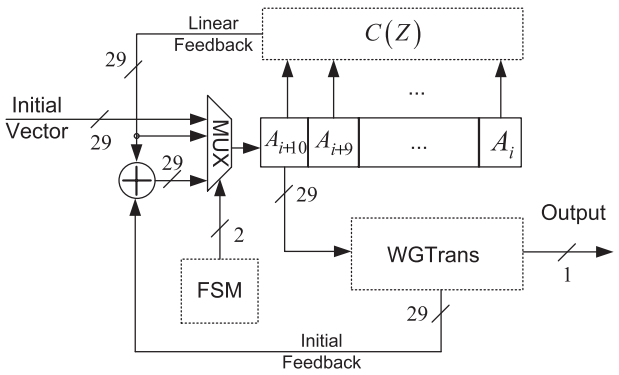

Fig. 1. WG generator [10], [18], [19], [25]. IV is the input during the loading phase. (linear feedback $\oplus$ initial feedback) is the input during the key initialization phase. Linear Feedback is the input throughout the PRSG phase.

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.

The architecture of the WG cipher is shown in Fig. 1. The LFSR feedback polynomial

$$C(Z) = Z^{11} \oplus Z^{10} \oplus Z^9 \oplus Z^6 \oplus Z^3 \oplus Z \oplus \beta \tag{3}$$

is a primitive polynomial of degree 11 over $GF(2^{29})$, where $\beta = \alpha^{464730077}$ is the generator of the ONB and α is a root of the defining polynomial of $GF(2^{29})$ given by [10]

$$g(Z) = Z^{29} \oplus Z^{28} \oplus Z^{24} \oplus Z^{21} \oplus Z^{20} \oplus Z^{19} \oplus Z^{18} \oplus Z^{17} \oplus Z^{14} \oplus Z^{12} \oplus Z^{11} \oplus Z^{10} \oplus Z^7 \oplus Z^6 \oplus Z^4 \oplus Z \oplus 1. \tag{4}$$

The output of the LFSR at A_{i+10} is filtered by an orthogonal 29-bit WG transformation $\left(GF(2^{29}) \longrightarrow GF(2)\right)$ given by

$$\text{WGTrans} = Tr\left(\text{WGPerm}\left(A_{i+10} \oplus 1\right)\right) \tag{5}$$

where

$$\begin{aligned} \text{WGPerm}(X) &= 1 \oplus X \oplus X^{r_1} \oplus X^{r_2} \oplus X^{r_3} \oplus X^{r_4} \\ &= \Big(1 \oplus X \oplus X^{2^k+1} \oplus X^{2^{2k}+(2^k+1)} \\ &\quad \oplus X^{2^k(2^k-1)+1} \oplus X^{2^{2k}+(2^k-1)}\Big) \end{aligned} \tag{6}$$

is the WG permutation, $r_1 = 2^k + 1$, $r_2 = 2^{2k} + 2^k + 1$, $r_3 = 2^{2k} - 2^k + 1$, $r_4 = 2^{2k} + 2^k - 1$, and $k = \lceil 29/3 \rceil$ [25]. This results in a binary key-stream of period $2^{319} - 1$ [10], [18].

The MOWG cipher uses the same formulation presented in (5), however, without the trace. It outputs 17 concatenated bits arbitrarily selected from the 29 output bits of the WG permutation [25].

The WG/MOWG ciphers consist of three phases of operations: loading phase (11 cycles), key initialization phase (22 cycles), and running phase. The reader is referred to [10], [18], [19], and [25] for more details.

III. Optimized Hardware Design of the MOWG Cipher

This section presents a hardware design of the MOWG(29, 11, 17) cipher, where 29 corresponds to $GF(2^{29})$, 11 is the number of stages in the LFSR, and 17 is the number

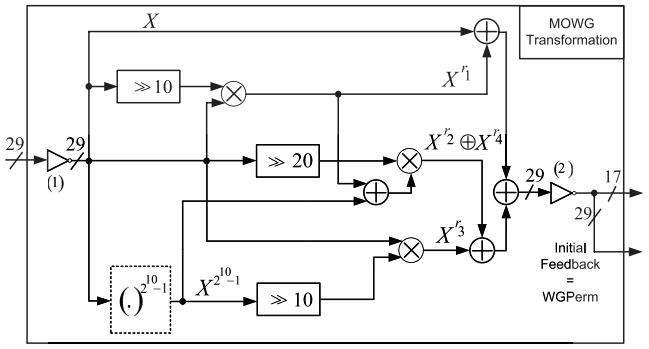

Fig. 2. Proposed MOWG transformation. $X = A_{i+10} \oplus 1$ is the bit-wise complement of the LFSR's output, $r_1 = 2^k+1$, $r_2 = 2^{2k}+2^k+1$, $r_3 = 2^{2k}-2^k+1$, $r_4 = 2^{2k} + 2^k - 1$, and $k = \left\lceil \frac{29}{3} \right\rceil = 10$.

of output bits. In this design, the MOWG transform uses seven multipliers, compared with eight multipliers in [25]. In addition, in an attempt to improve the overall speed of the cipher, the LFSR is reconstructed to remove the inverters from the critical paths during the PRSG phase/initialization phase. In what follows, the reduced area MOWG transform design is first introduced, followed by presenting the LFSR/key and initial vector loading algorithm (KIA) algorithm changes for speed improvement. Then, the architecture of the finite-state machine (FSM) is discussed, and the section ends up by deriving formulations for the space and time complexities.

A. Reducing the Hardware Complexity of the MOWG Transformation

The hardware cost of the MOWG cipher is dominated by its transform's field multipliers. Any decrease in the number of these multipliers would minimize the area of the overall cipher. This subsection presents the architecture of the MOWG transform, where the number of field multipliers is reduced by 1 through signal reuse, compared with those in [25].

The architecture of the proposed MOWG transform is shown in Fig. 2. Through taking $X^{2^{2k}}$ as a common factor of the exponent terms $2^{2k}+\left(2^k+1\right)$ and $2^{2k}+\left(2^k-1\right)$ in (6), this architecture can easily be obtained, where the WG permutation given by (6) is now computed as follows:

$$\text{WGPerm} = \Big(1 \oplus X \oplus X^{2^k+1} \oplus X^{2^k(2^k-1)+1} \oplus X^{2^{2k}}\left(X^{(2^k+1)} \oplus X^{(2^k-1)}\right)\Big). \quad (7)$$

In the MOWG$(29, 11, 17)$, $k = 10$ and, hence, the signal X^{2^k-1} requires four multiplications and four squaring operations (that is free of cost in ONB) [25]. In addition to the multiplication operations involved in computing the signal $X^{(2^k-1)}$, (7) requires three more multiplications to generate the signals X^{2^k+1}, $X^{2^k(2^k-1)+1}$, and $X^{2^{2k}}\left(X^{(2^k+1)} \oplus X^{(2^k-1)}\right)$. Therefore, the architecture of Fig. 2 requires a total of seven $GF(2^{29})$ multiplications. The inverter symbol denoted by (1) in this figure requires 29 NOT gates to generate $X = A_{i+10} \oplus 1$ from the LFSR's output signal A_{i+10}. The signal $X \oplus X^{r_1} \oplus X^{r_2} \oplus X^{r_3} \oplus X^{r_4}$ is obtained as the addition in $GF(2^{29})$ of X, $X^{r_1} = X^{2^k+1}$, $X^{r_2} \oplus X^{r_4} = X^{2^{2k}}\left(X^{(2^k+1)} \oplus X^{(2^k-1)}\right)$, and $X^{r_3} = X^{2^k(2^k-1)+1}$. The signals X^{2^k} and $X^{2^{2k}}$ are obtained by

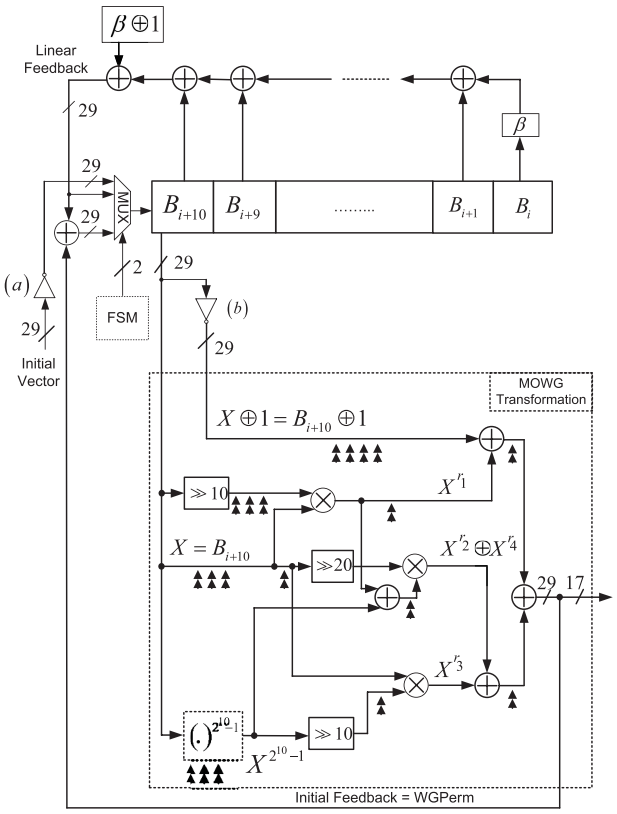

Fig. 3. Proposed design of the MOWG$(29, 11, 17)$ cipher. A double-headed arrow, under a component, corresponds to a 29-bit Register which is inserted for pipelining purposes (see Section V-B for more details).

right cyclic shifts of X, k, and $2k$ times, respectively. X^{2^k+1} is generated by multiplying X with X^{2^k} in $GF(2^{29})$. $X^{2^k(2^k-1)}$ is the right cyclic shift of $X^{(2^k-1)}$, k times, and $X^{2^k(2^k-1)+1}$ is generated by multiplying $X^{2^k(2^k-1)}$ with X in $GF(2^{29})$. In Fig. 2, the coordinates of the output of $X \oplus X^{r_1} \oplus X^{r_2} \oplus X^{r_3} \oplus X^{r_4}$ in $GF(2^{29})$ are complemented by the inverter symbol denoted by (2) to generate all 29 bits of the WGPerm function of (7), which forms the initial feedback. Seventeen bits of the WGPerm are the output of the MOWG in the run phase [25].

B. Improving the Critical Path of the MOWG Transform

The time delay through the MOWG transform dominates the delay of the overall cipher (Section III-D2). This subsection shows how to slightly reduce the delay through this transform. This is accomplished by removing inverter 1, and by reallocating inverter 2 away from the critical paths of the PRSG and key initialization phases. This reduces the delay of the critical path by an amount equivalent to the delay of two inverters. However, the MOWG transform delay is still the dominant because of the delays of five serially connected field multipliers. First, the required mathematical formulation is derived, then the resulting new architecture of the cipher is presented.

1) Formulation: During the key initialization and PRSG phases, inverter 1 in Fig. 2 generates the complement of A_{i+10}. Notice that this cell holds the feedback from the LFSR during the PRSG phase, and the bit-wise XOR of the LFSR feedback and the MOWG transform feedback

during the key initialization phase. Therefore, to remove inverter 1, it requires the direct storage of the complement of these values in both phases. In other words, it is required to reconstruct the LFSR such that it generates a sequence $\underline{B} = \{B_i = 1 \oplus A_i,\ 0 \le i \le 2^{319} - 1\}$, where $B_i \in GF(2^{29})$ and $\{A_i\}$ is the sequence generated by (3) over $GF(2^{29})$. Sequence $\underline{B}$ is referred to as the complement sequence of $\{A_i\}$. The following proposition shows how this is accomplished for an LFSR with a general feedback polynomial of degree l over $GF(2^m)$.

Proposition 1: Let $\underline{B}$ be the complement sequence of a sequence $\underline{A} = \{A_i,\ 0 \le i \le 2^{ml} - 1\}$, where $A_i \in GF(2^m)$ and $\underline{A}$ is generated by (2). Then, $\underline{B}$ is generated by the following recurrence relation:

$$B_{j+l} = \left(\sum_{i=0}^{l-1} C_i B_{i+j}\right) \oplus \left(\left(\sum_{i=0}^{l-1} C_i\right) \oplus 1\right) \tag{8}$$

where $j \ge 0$, and the initial state of $\underline{B}$ is $B_i = 1 \oplus A_i$, for $0 \le i \le l - 1$.

Proof: By definition

$$B_{j+l} = A_{j+l} \oplus 1 \tag{9}$$

$j \ge 0$. Using (2) in (9), one gets $B_{j+l} = \sum_{i=0}^{l-1} C_i A_{i+j} \oplus 1$, and by noticing $2C_i = 0$ one obtains

$$\begin{aligned} B_{j+l} &= \sum_{i=0}^{l-1} C_i (A_{i+j} \oplus 1) \oplus \sum_{i=0}^{l-1} C_i \oplus 1 \\ &= \sum_{i=0}^{l-1} C_i B_{i+j} \oplus \sum_{i=0}^{l-1} C_i \oplus 1. \end{aligned}$$

Thus, the assertion is true. ∎

Through noticing that $X = 1 \oplus A_{i+10}$ in (7), then, from Proposition 1, one can see that X is B_{i+10}. Notice that the term $\left(\sum_{i=0}^{l-1} C_i\right) \oplus 1$ in (8) is a constant term. Hence, its addition in $GF(2^{29})$ is realized with a number of NOT gates equal to its Hamming weight. For the LFSR of the MOWG, replacing the coefficients of (3) in (8) gives $\left(\sum_{i=0}^{l-1} C_i\right) \oplus 1 = \beta \oplus 1$, which has a Hamming weight equal to 28.

Inverter 2, on the other hand, realizes the addition of the field element 1 in (7). Notice that this addition of the term 1 can be implemented in different ways. One way is to add it to one of the terms X, X^{r_1}, $X^{r_2} \oplus X^{r_4}$, or X^{r_3} before the summation of these terms. Doing so would reallocate inverter 2 from its current position. It is, however, required that this reallocation does not result in a delay higher than the current maximum delay of the MOWG transform. For this reason, the inverter is relocated to complement X before it is added to X^{r_1}. This is the path at the top of Fig. 2, which has the lowest delay with only two $GF(2^{29})$ adders between inverters 1 and 2.

2) Modified KIA Algorithm: Modifying the MOWGs LFSR according to (8) requires its left most stage to hold the complement of the IV during the loading phase. Therefore, it is required to complement the IV input before it is loaded to the modified LFSR. This can easily be implemented by inserting 29 inverters at the multiplexer's input that receives the IV in Fig. 1.

3) Architecture: Here, the overall proposed architecture of the MOWG(29, 11, 17) cipher is presented, as shown in Fig. 3. In this figure, the FSM controls the input to the LFSR for each phase of operation. In the same figure, because of the bit-wise complement operator denoted by (a), the LFSR receives the complemented IV during the loading phase. Hence, after 11 clock cycles, the initial state of this LFSR, $(B_0, B_1, \ldots, B_{10})$, is basically the complement of the initial state of the LFSR in Fig. 1, i.e., $B_i = A_i \oplus 1$, $0 \le i < 11$. When the key initialization phase starts, the bit-wise XOR of the initial feedback and linear feedback applies to the input of the LFSR. Note that the Linear Feedback in Fig. 3 is generated by (8), which is equivalent to $B_i = A_i \oplus 1$, $11 \le i < 33$ (complement of corresponding one in Fig. 1). However, the initial feedback signal in Fig. 3 has the same value as the one generated in Fig. 2. This means that the input to the LFSR during the key initialization phase in Fig. 3 is complemented with respect to the one in Fig. 1. Throughout the PRSG phase, the only input to the LFSR is the linear feedback signal $B_i = A_i \oplus 1$, $33 \le i < 2^{319} - 1$. This sets the MOWG transform of Fig. 3 to generate the same key-stream bits of Fig. 2. It is clear that the maximum delay of the MOWG transformation is reduced by an amount equivalent to the delay of two inverters, as compared with the one in Fig. 2. The revised LFSR in Fig. 3 has additional $H(\beta \oplus 1) = 28$ inverters, compared to Fig. 1. This is due to the new constant term $\beta \oplus 1$ in the feedback polynomial.

C. Finite State Machine

This subsection exposes the architecture of the FSM and describes how it schedules the input to the LFSR throughout the three phases of operation.

Fig. 4 shows the components of the FSM. The FSM has two inputs, namely clk and reset, 1-bit each, whereas there are two outputs denoted as op0 and op1. The reset input is pulled down before each run of the cipher. This forces the 11-bit one-hot counter to initialize to $(1, 0, \ldots, 0)$, i.e., output 0 is the only bit set to a high logic level. In addition, when the reset signal is low, the 2-bit binary counter resets its state to $(0, 0)$. Because of the 1-bit Register connected to the AND gate at the reset input of the 11-bit one-hot counter, this counter starts incrementing one clock cycle after the reset signal gets pulled up. This assures that the 11-bit one-hot counter returns to its initial state after 11 clock cycles. Then, it triggers the 2-bit binary counter to increment that starts the initialization phase. The output of the 2-bit binary counter controls the cipher's phase of operation. This is done by generating the op0 and op1 signals according to Table I.

The op0 and op1 signals select one of the three inputs of the multiplexer in Fig. 3 and connect it to the input of the LFSR, during each phase. It is noted that the loading phase takes 11 clock cycles, then starts the key initialization phase that takes 22 clock cycles, followed by the run phase. During the run phase, the clock inputs of the 11-bit one-hot counter and the 2-bit binary counter become idle.

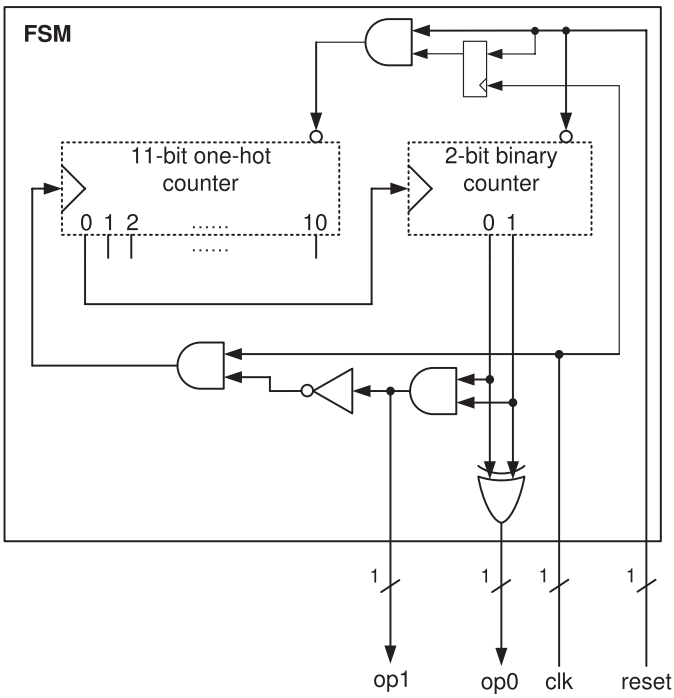

Fig. 4. FSM of the MOWG.

TABLE I
PHASE OF OPERATION IN THE PROPOSED MOWG AS A FUNCTION OF THE STATE OF THE 2-BIT BINARY COUNTER

2-bit counter		op1	op0	phase of operation
bit 1	bit 0			
0	0	0	0	Load Key and IV
1	0	0	1	Key Initialization
0	1	0	1	Key Initialization
1	1	1	0	Running Phase

D. Space and Time Complexities

This subsection provides the space and time complexities of the MOWG design in Fig. 3.

1) Space Complexity: The space complexity is evaluated in terms of number of gates in each component to obtain the overall hardware cost. Let N_R, N_A, N_X, N_O, and N_I denote the number of 1-bit Registers, AND gates, XOR gates, OR gates, and inverters, respectively.

a) MOWG transform: The transform dominates the hardware complexity of the MOWG design as it consists of seven field multipliers and four $GF(2^{29})$ adders. A $GF(2^{29})$ adder requires 29 XOR gates. Also, the multiplier in [26] is used for implementation, which has 841 AND gates and 1218 XOR gates. Therefore, the total hardware cost of the transformation is as listed in Table II.

b) Linear feedback shift register: The LFSR has 11-stages of 29-bit shift registers and a feedback polynomial. The feedback polynomial is composed of one field multiplier (with a constant),[1] five $GF(2^{29})$ additions, and $H(\beta \oplus 1) = 28$ inverters. Therefore, the hardware complexity of the LFSR is as listed in Table II.

[1]A multiplication with a constant can be further optimized so that it contains few XOR gates.

TABLE II
COUNT OF 1-BIT REGISTERS AND LOGIC GATES IN THE DIFFERENT COMPONENTS OF THE PROPOSED MOWG DESIGN

Component	N_R	N_A	N_X	N_O	N_I
MOWG Transform	-	5887	8642	-	-
LFSR	319	841	1363	-	28
FSM (Figure 4)	14	3	1	-	1
29-bit 4-to-1 MUX	-	174	-	87	2

c) 4-to-1 29-bit multiplexer: The 4-to-1 29-bit multiplexer is composed of a binary tree of three 2-to-1 29-bit multiplexers and two NOTs (selectors). Each 2-to-1 29-bit multiplexer is built from 29 parallel 2-to-1 1-bit multiplexers. A 2-to-1 one bit multiplexer consists of two AND gates and one OR gate. Therefore, the total cost of the 4-to-1 29-bit multiplexer is as listed in Table II.

d) Finite-state machine: From Fig. 4, there are three AND gates, one XOR gate, and one inverter in the FSM. The 11-bit one-hot counter is simply an 11-stages circular shift register with set/reset inputs having the output of the last shift register fed to the input of the first one. The 2-bit binary counter is built from two JK flip-flops (FF). The two inputs of the first FF are pulled to high logic and its output drives the two inputs of the second FF. Thus, one can find the total number of one-bit registers as

$$N_R = 11 + 2 + 1 = 14.$$

Table II lists the number of gates in the FSM.

In addition to the above-mentioned components, the MOWG cipher contains two 29-bit bit-wise complement operators (inverter symbol (a) and inverter symbol (b) in Fig. 3) and a $GF(2^{29})$ adder (computing the bit-wise XOR of initial feedback signal and the linear feedback signal). Let N_O^{MOWG}, N_I^{MOWG}, N_R^{MOWG}, N_A^{MOWG}, and N_X^{MOWG} denote the number of OR gates, Inverters, 1-bit Registers, AND gates, and XOR gates in the MOWG of Fig. 3, respectively. Therefore, by adding the corresponding number of gates in this $GF(2^{29})$ adder and in inverter symbols (a) and (b) to the number of gates in the FSM, the 4-to-1 29-bit multiplexer, the LFSR, and the MOWG transform (Table II) one obtains

$$N_O^{\text{MOWG}} = 87, \quad N_I^{\text{MOWG}} = 89, \quad N_R^{\text{MOWG}} = 333,$$
$$N_A^{\text{MOWG}} = 6905, \quad N_X^{\text{MOWG}} = 10035.$$

2) Time Complexity: Here, the formulation for the critical path delay of the MOWG cipher (Fig. 3) is derived. There are three critical paths in the MOWG.

1) Critical path of the LFSR.
2) Critical path along the MOWG transformation during the key initialization phase.
3) Critical path along the MOWG transformation during the run phase.

The LFSR's path has one multiplication and five finite field additions. This results in a propagation delay of

$$T_A + \left(1 + \left\lceil \log_2 (6) \right\rceil + \left\lceil \log_2 (29) \right\rceil\right) T_X = T_A + 9T_X \quad (10)$$

where T_A and T_X denote the propagation delay of an AND and an XOR, respectively. The delay through a finite field multiplier is $T_A + \left(1 + \left\lceil \log_2 (29) \right\rceil\right) T_X$ [26]. On the other hand, the delays through the two MOWG transform paths have five multipliers in series, which corresponds to a delay of

$$5\,(T_A + 6T_X) = 5T_A + 30T_X. \quad (11)$$

From (10) and (11), it is clear that the longest path of the MOWG cipher passes through its transformation.

From Fig. 3, the critical path of the proposed MOWG during the run phase includes the delays of a 29-bit Register, five field multipliers in series, and three $GF(2^{29})$ adders. These results in the delay are stated as

$$T_{\text{RunPh}} = 5T_A + 33T_X + T_R \quad (12)$$

where T_{RunPh} denotes the maximum propagation delay through the MOWG during the run phase. In the same figure, the critical path of the MOWG during the key initialization phase includes the delays of four $GF(2^{29})$ adders, five field multipliers, a 29-bit Register, and a 4-to-1 29-bit multiplexer. Notice that the delay through the 4-to-1 29-bit multiplexer is equivalent to the delay through two 2-to-1 1-bit multiplexers in series. This is equivalent to the sum of the delays through two AND gates, two OR gates, and two inverters. Therefore, the delay of the MOWG during the key initialization phase is

$$T_{\text{KIPh}} = 7T_A + 34T_X + T_R + 2T_O + 2T_I. \quad (13)$$

Comparing (12) and (13), it is clear that $T_{\text{KIPh}} > T_{\text{RunPh}}$.

IV. Low Complexity WG Cipher

This section proposes a new design of the WG(29, 11). The proposed WG design considers Fig. 3 with an added trace to the output of the WGPerm as the starting point for optimization. Properties of the trace function when the elements of $GF(2^m)$ are represented in ONB of type-II (that exists for $m = 29$ [27]) are first introduced. The proposed WG design uses these properties to minimize the hardware complexity of its transform. Note that the proposed design eliminates some necessary signals for the generation of the initial feedback, which is required to conduct the key initialization phase of the cipher. Missing of the initial feedback signal is recovered by introducing a serialized scheme to generate it. At the end of this section, the hardware and the time complexities of the new implementation are provided.

A. Properties of the Trace Function for Type-II ONB

This section presents a method for computing the trace of a multiplication of two field elements when the representation is in the type-II ONB. In addition, two corollaries are deduced from the proposed method.

Fact 1 [28]: Let $\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$ be a type-II ONB in $GF(2^m)$. Then

$$Tr(\beta^{2^i}) = 1, \qquad i = 0, 1, \ldots, m-1$$

and

$$Tr(\beta^{2^i}\beta^{2^j}) = 0 \qquad \forall i \neq j; \quad i, j = 0, 1, \ldots, m-1.$$

In other words, a type-II ONB is a self-dual basis. Thus, Proposition 2 is achieved as follows.

Proposition 2: In a type-II ONB, the trace of the field multiplication of any two $GF(2^m)$ elements $A = (a_0, a_1, \ldots, a_{m-1})$ and $B = (b_0, b_1, \ldots, b_{m-1})$ is computed as the inner product of A and B as follows:

$$Tr\,(\text{AB}) = \sum_{i=0}^{m-1} a_i b_i. \quad (14)$$

Proof: The proof is completed by considering the following derivation:

$$\begin{aligned} Tr(\text{AB}) &= Tr\left(\sum_{i=0}^{m-1} a_i \beta^{2^i} \sum_{j=0}^{m-1} b_j \beta^{2^j}\right) \\ &= \sum_{0 \le i,j < m} a_i b_j Tr(\beta^{2^i + 2^j}) \\ &= \sum_{i=0}^{m-1} a_i b_i \end{aligned}$$

where the last result is obtained using Fact 1. ∎

Proposition 2 implies that the trace of a field multiplication of two elements represented in type-II ONB is easily implemented in hardware using m AND gates and $m-1$ XOR gates.

Corollary 1: In type-II ONB, the two relations below are valid for any two elements A and B in $GF(2^m)$

$$Tr\,(\text{AB}) = Tr\,((A \gg n)\,(B \gg n)) = \sum_{i=0}^{m-1} a_{i-n} b_{i-n} \quad (15)$$

and

$$Tr\,(AB) = Tr\,((A \ll n)\,(B \ll n)) = \sum_{i=0}^{m-1} a_{i+n} b_{i+n} \quad (16)$$

where n is a positive integer and the indices of a and b are computed modulo m.

Proof: Let A and B be any two elements in $GF(2^m)$ and n an arbitrary positive integer. It is well known that

$$Tr\left(X^{2^{\pm n}}\right) = Tr\,(X)^{2^{\pm n}} = Tr\,(X)$$

for any $X \in GF(2^m)$. Therefore, by replacing X with AB one obtains

$$Tr\,(\text{AB}) = Tr\left(A^{2^{\pm n}} B^{2^{\pm n}}\right). \quad (17)$$

Through using Proposition 2, the proof is completed by realizing that the squaring operation X^2 and the square root operation $X^{2^{-1}}$ are simply the right cyclic shift and the left cyclic shift of the coordinates of X with respect to the ONB, respectively. ∎

According to Corollary 1, the trace of the field multiplication of any two elements A and B, represented in type-II ONB, does not change if an n-bit cyclic shift (left or right) is applied to both elements in the same direction.

Corollary 2: Let C be a common factor of two or more $GF(2^m)$ elements AC, BC,..., etc, then, the following relation holds:

$$Tr(AC) + Tr(BC) + \cdots = \sum_{i=0}^{m-1} (a_i + b_i + \cdots) c_i. \quad (18)$$

Proof: Let A, B,..., etc, be any two or more arbitrary elements from the finite field $GF(2^m)$. Then

$$\begin{aligned} Tr(AC) + Tr(BC) + \cdots &= Tr((A \oplus B \oplus \cdots) C) \\ &= \sum_{i=0}^{m-1} (a_i + b_i + \cdots) c_i \end{aligned}$$

where the last result follows from Proposition 2, and $C \in GF(2^m)$. ■

B. Optimizing the WG Transform's Hardware for the Run Phase

Here, it is shown how Proposition 2 and Corollaries 1 and 2 are used to further reduce the number of field multiplications in the WG transform in Fig. 3 (with trace). Before proceeding, it is important to mention that by applying (14), one can generate the trace of the field multiplication of two elements A and B directly from A and B. However, the result of the multiplication operation, i.e., $C = AB$, will be lost. Therefore, it is important to apply (14) to the multiplication terms in (7), which are not used anywhere else. From Fig. 3, the two signals $X^{r_2} \oplus X^{r_4}$ and X^{r_3} are used only as inputs to the trace function (after they are bit-wise XORed), whereas the signal X^{r_1} is required in generating $X^{r_2} \oplus X^{r_4}$ (Section II for the values of $r_i's$). The first two signals are generated as follows:

$$\begin{cases} X^{r_2} \oplus X^{r_4} &= X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k - 1} \right) \\ X^{r_3} &= X X^{2^k (2^k - 1)}. \end{cases} \quad (19)$$

Therefore, applying the trace function to (19) one gets

$$\begin{cases} Tr(X^{r_2} \oplus X^{r_4}) &= Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k - 1} \right) \right) \\ Tr(X^{r_3}) &= Tr\left(X X^{2^k (2^k - 1)} \right). \end{cases} \quad (20)$$

Using (20), the WG transformation becomes

$$\begin{aligned} \text{WGTrans} = Tr\left(1 \oplus X \oplus X^{r_1}\right) &+ Tr\left(X X^{2^k (2^k - 1)} \right) \\ &+ Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k - 1} \right) \right). \end{aligned} \quad (21)$$

Applying a right cyclic shift of $2k$-stages to X and $X^{2^k (2^k - 1)}$ in the term $Tr\left(X X^{2^k (2^k - 1)} \right)$ of (21) does not change the value of the trace

$$Tr\left(X X^{2^k (2^k - 1)} \right) = Tr\left((X)^{2^{2k}} \left(X^{2^k (2^k - 1)} \right)^{2^{2k}} \right). \quad (22)$$

Using (22) in (21) gives

$$\begin{aligned} \text{WGTrans} = Tr\left(1 \oplus X \oplus X^{r_1}\right) &+ Tr\left(X^{2^{2k}} X^{3k (2^k - 1)} \right) \\ &+ Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k - 1} \right) \right). \end{aligned} \quad (23)$$

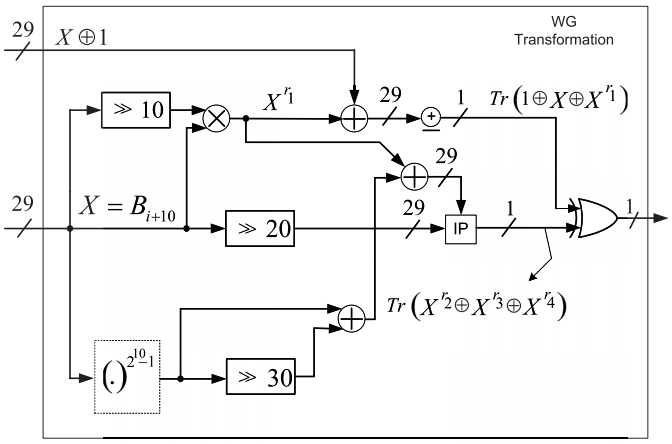

Fig. 5. Proposed design of the WG transformation. Block denoted by IP generates the inner product of the two 29-bit inputs (Section II), whereas $\underline{\oplus}$ adds the 29-bits at its input over $GF(2)$.

Taking $X^{2^{2k}}$ as a common factor in (23) one obtains

$$\begin{aligned} \text{WGTrans} = Tr\left(1 \oplus X \oplus X^{r_1}\right) \\ + Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k - 1} \oplus X^{3k (2^k - 1)} \right) \right). \end{aligned} \quad (24)$$

Notice that by applying Corollary 2 to (24), only one multiplication operation is required to generate $X^{r_1} = X^{2^k + 1}$ (excluding the generation of the signal $X^{2^k - 1}$). Fig. 5 shows the resulting architecture of the WG transform in (24). This architecture uses five field multipliers, i.e., four multipliers less than the WG transform presented in [10].

In Fig. 5, the key stream bits are obtained by XORing $Tr(1 \oplus X \oplus X^{r_1})$ with $Tr(X^{r_2} \oplus X^{r_3} \oplus X^{r_4})$. $Tr(1 \oplus X \oplus X^{r_1})$ is the $GF(2)$ addition of the coordinates of $1 \oplus X \oplus X^{r_1}$ with respect to the ONB. On the other hand, notice that the signals X^{r_3} and $X^{r_2} \oplus X^{r_4}$ do not exist in the WG transform. This is because $Tr(X^{r_2} \oplus X^{r_3} \oplus X^{r_4})$ is generated directly from $X^{2^{2k}}$, X^{r_1}, $X^{2^k - 1}$, and $X^{3k (2^k - 1)}$ using an inner product operation, as it is stated in (24). This absence of the two signals X^{r_3} and $X^{r_2} \oplus X^{r_4}$ resulted in the elimination of the initial feedback signal. The next subsection proposes a recovery method for generating the initial feedback signal, which is only used in the key initialization phase.

C. Serializing the Computation of the Initial Feedback Signal

This section presents a method for the recovery of the Initial feedback signal through serialized computation. To accomplish the multiplication operations during this serial computation, the existing finite field multiplier that is used in generating the signal X^{r_1} in Fig. 5, is used. The proposed scheme generates the initial feedback signal by serially computing it over three consecutive clock cycles. Denote this complete round of the serialized initial feedback computation (three clock cycles) as an extended key initialization round. In addition, denote the single clock cycle version of this computation (as in the MOWG design) as a simple round. Therefore, with serialization, the entire key initialization phase requires

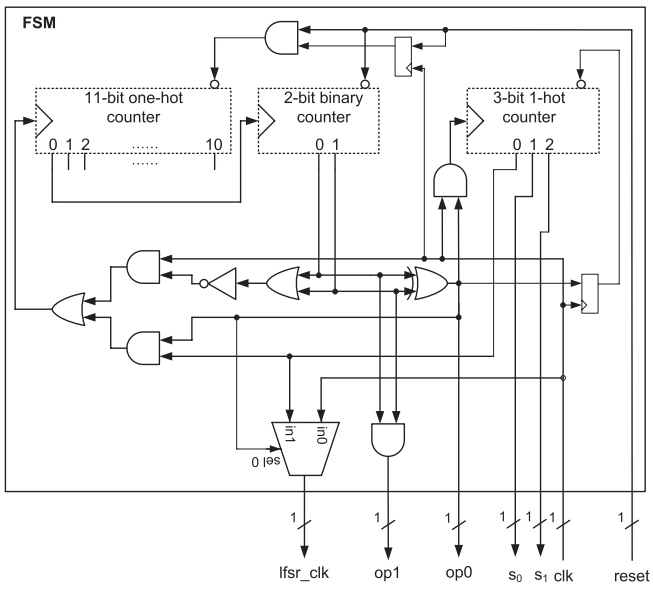

Fig. 6. Modified FSM after adding the new 3-bit one-hot counter.

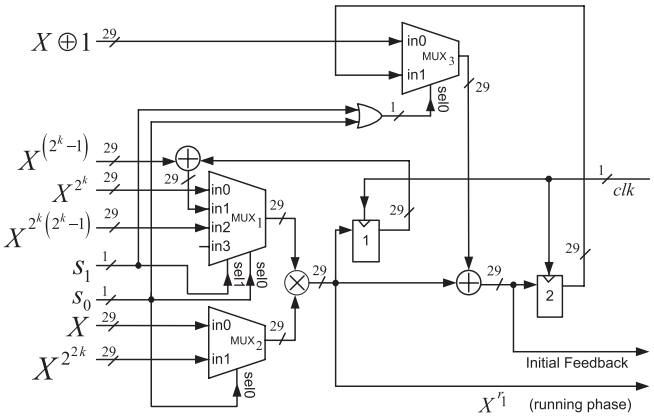

Fig. 7. Block diagram of the SKIM module. The initial feedback signal is connected to the LFSRs input multiplexer as shown in Fig. 1. X^{r_1} connectivity is shown in more details in Fig. 8.

$3 \times 22 = 66$ clock cycles instead of 22 clock cycles (that is, 22 extended rounds instead of 22 simple rounds). It is noted that this only affects the key initialization phase without increasing the number of cycles required for the run phase.

The expansion of the key initialization round from one to three clock cycles is established through the support of a new FSMs control signal, namely, lfsr_clk (Fig. 6). This signal controls the clock input of the LFSR and triggers it to shift once every three clock cycles. In addition, to compute the initial feedback signal over three stages, a new hardware module denoted as the serialized key initialization module (SKIM) will be introduced (Fig. 7). This module uses the available signals and the field multiplier that is used in the generation of X^{r_1}, in Fig. 5. This module schedules the proper inputs to the field multiplier in each stage of the serial computation through some multiplexers. The output of these multiplexers are controlled by two new signals generated by the FSM, namely, s_0 and s_1 (Fig. 6). The intermediate results, between two consecutive stages of the computation, are stored in internal 29-bit Registers of the SKIM module. In the following, the FSM changes required for the support of the serialization process are first introduced. Then, the architecture and operation of the SKIM module and its integration to the WG transform in Fig. 5 are discussed.

1) Architecture and Operation of the Modified FSM: Here, the new architecture and operation of the FSM are described. The architecture, which is shown in Fig. 6, generates the new set of control signals lfsr_clk, s_0, and s_1. These are required for the serial computation of the initial feedback signal. Before each run of the cipher, the FSM resets its 11-bit one-hot counter to $(1, 0, \ldots, 0)$ and its 2-bit binary counter to $(0, 0)$ (where the leftmost and rightmost bits, within the brackets, denote the lowest output bit and the highest output bit of the corresponding counter, respectively). This is done through pulling down the reset inputs. When the reset signal is released, the 2-bit binary counter becomes ready. At the same time, the 11-bit one-hot counter's reset input stays pulled down for an extra clock cycle. This is due to the 1-bit Register connected to the input of the AND gate that drives its reset input. This assures that the $(1, 0, \ldots, 0)$ state of the 11-bit one-hot counter consumes a clock cycle at the beginning of the loading phase. After 11 clock cycles, from the release of the reset signal, the 11-bit one-hot counter returns to the $(1, 0, \ldots, 0)$ state. At this point, it triggers the clock input of the 2-bit binary counter. The 2-bit binary counter changes its state to $(1, 0)$, triggering the start of the key initialization phase. Then, the clk signal starts triggering the clock input of the 3-bit one-hot counter. The counting will, however, start one clock cycle later, when the output of the 1-bit Register connected to the 3-bit one-hot counter's reset input pulls up. This in turn assures that the 3-bit one-hot counter consumes one clock cycle, before incrementing its initial state of $(1, 0, 0)$, at the start of the key initialization phase. During this phase, the first output bit of the 3-bit one-hot counter drives the clock input of the 11-bit one-hot counter. Therefore, it takes 33 clock cycles for the 11-bit one-hot counter to complete 11 counts. Hence, it takes 33 clock cycles for the 2-bit binary counter to increment. Therefore, it requires 66 clock cycles for the 2-bit binary counter to increment twice to start the running phase. When the running phase starts, with the 2-bit binary counter's state at $(1, 1)$, the 11-bit and the 3-bit one-hot counters stop counting, as their clock inputs become idle.

Notice that during the key initialization phase, the lfsr_clk is driven by the first output of the 3-bit one-hot counter. Hence, the LFSR shifts once every three clock cycles. The two signals s_0 and s_1 are derived from the 3-bit one-hot counter's output according to Table III. Notice that this table is realized without any additional hardware by setting s_0 to be the second output and s_1 to be the third output of the 3-bit one-hot counter, respectively. Therefore, (s_0, s_1) produces the three patterns of $(0, 0)$, $(1, 0)$, and $(0, 1)$ during the first, second, and third stages of an extended key initialization round, respectively. During the running phase, (s_0, s_1) will generate $(0, 0)$. The following shows how these patterns are used to accomplish the proper functionality in the key initialization phase as well as in the running phase.

TABLE III
SIGNALS s_0 AND s_1 AS A FUNCTION OF THE OUTPUT OF THE 3-BIT ONE-HOT COUNTER

3-bit one-hot counter			s_1	s_0
bit 2	bit 1	bit 0		
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0

2) Architecture and Operation of the SKIM: Here, the SKIM module, which performs the serialized computation of the initial feedback signal over an extended key initialization round (three clock cycles), is presented.

Fig. 7 is a block diagram describing the architecture of this module. During the extended key initialization round, the two signals s_0 and s_1 in Fig. 7 change values in each stage as mentioned in the previous section. These two signals control the outputs of the three multiplexers MUX_1, MUX_2, and MUX_3 according to Table IV.

In each stage of the extended key initialization round, the SKIM module computes a partial value of the initial feedback signal and stores it in Register 2 (Fig. 7).

During the first clock cycle, s_0 and s_1 are both at low logic levels. Hence, MUX_1, MUX_2, and MUX_3 generate the signals X^{2^k}, X, and $X \oplus 1$ at their outputs, respectively. The output of the multiplier becomes $X^{r_1} = X^{2^k+1}$ and that of the $GF(2^{29})$ adder is $X^{r_1} \oplus X \oplus 1$. Upon receiving a new clock signal, i.e., at the start of the second clock cycle, Register 1 and Register 2 update their states with the output signal of the multiplier and output of the $GF(2^{29})$ adder, respectively. In addition, X^{2^k-1} is stored in a 29-bit Register (see Fig. 8). At the same time, s_0 pulls up forcing the outputs of MUX_1, MUX_2, and MUX_3 to become $X^{r_1} \oplus X^{2^k-1}$, $X^{2^{2k}}$, and $X^{r_1} \oplus X \oplus 1$ (the state of Register 2 when the clock signal arrived), respectively. With these settings of the multiplexers and the registers, the multiplier output changes to $X^{r_2} \oplus X^{r_4} = X^{2^{2k}} \left(X^{r_1} \oplus X^{(2^k-1)} \right)$ and that of the $GF(2^{29})$ adder to $X^{r_4} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$, denoting Register 1's and Register 2's next states, respectively, when the third clock signal arrives. When the third clock cycle starts, s_0 changes to low logic level while s_1 changes to high logic level, which forces MUX_1, MUX_2, and MUX_3 to generate $X^{2^k(2^k-1)}$, X, and $X^{r_4} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$ at their outputs, respectively. The multiplier and the $GF(2^{29})$ adder outputs become $X^{r_3} = X^{2^k(2^k-1)+1}$ and $X^{r_4} \oplus X^{r_3} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$, respectively.

At the arrival of the fourth clock signal (the beginning of a new extended key initialization round) s_0 and s_1 both change back to low logic levels, the LFSR is clocked and latched with the result of the bit-wise XOR of the computed initial feedback signal ($X^{r_4} \oplus X^{r_3} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$) and the LFSRs linear feedback signal. At the arrival of the 67th clock signal, the LFSR would have been clocked 22 times and the running phase starts.

Throughout the run phase, both s_0 and s_1 stay at logic level 0; therefore, MUX_1 generates the signal X^{2^k} and MUX_2 generates the signal X. With these values, the multiplier generates X^{r_1} and the WG transform in Fig. 8 produces a stream bit for each cycle.

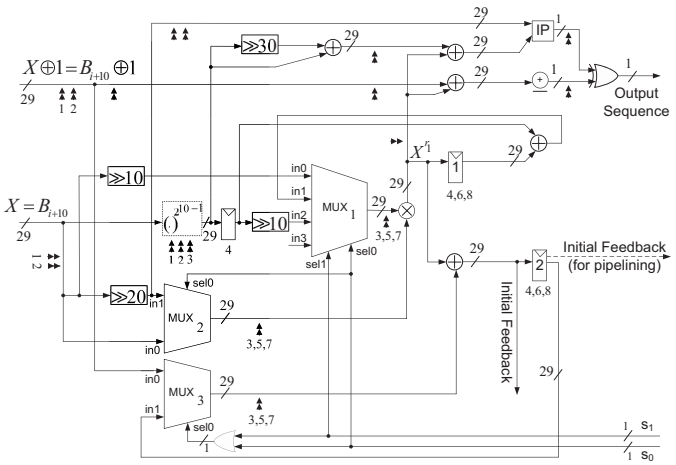

Fig. 8. Proposed WG transformation after integration with the SKIM module. Block denoted by IP generates the inner product of the two 29-bit inputs (Section II), whereas $\underline{\oplus}$ adds the 29-bits at its input over $GF(2)$. Double-headed arrows under a component (correspond to inserted registers) and the dotted arrow output (initial feedback), are used for pipelining (Section V-B). Numbers under a register specify the clocking of that register within the pipelined scheme, during initialization phase.

D. Space and Time Complexities

This section begins with presenting the hardware complexity of the proposed WG implementation, followed by the time complexity.

1) Space Complexity: The space complexity of the WG transform is reduced, whereas that of the WG's FSM is slightly increased, compared with the corresponding ones in the proposed MOWG. In what follows, the hardware complexities of the WG transform and its FSM are first summarized. Then, the overall hardware cost of the WG design is obtained.

a) WG transformation: The space complexity of the WG transform has been improved compared with the MOWG transform. This is mainly because the number of field multipliers in the WG transform is reduced by 2 with respect to that in the MOWG transform. On the other hand, compared with the MOWG transformation in Fig. 3, the design in Fig. 8 has the following additional components: 1) a $GF(2^{29})$ adder; 2) a 29-bit $GF(2)$ addition; 3) three 29-bit Registers; 4) an XOR gate; 5) an OR gate; 6) one 4-to-1 29-bit multiplexer; 7) two 2-to-1 29-bit multiplexers with 2 selector NOTs; and 8) an inner product. A 29-bit $GF(2)$ adder consists of 28 XOR gates. A 2-to-1 29-bit multiplexer consists of 29 parallel 2-to-1 1-bit multiplexers. The inner product has 29 AND gates and 28 XORs. Details about the hardware of the other components are listed in Section III-D1. Through adding the hardware of the additional components to the gate count in the MOWG transform (Table II), and then subtracting the hardware cost of two field multipliers, the total hardware cost of the proposed WG transform is obtained as listed in Table V.

b) Finite state machine: The FSM shown in Fig. 6 has additional two AND gates, two OR gates, a 2-to-1 1-bit multiplexer (with 1 selector NOT), 1-bit Register, and a 3-bit one-hot counter as compared with Fig. 4. Similar to

TABLE IV
MULTIPLEXERS OUTPUTS AND NEXT STATES OF REGISTER 1 AND REGISTER 2 AS A FUNCTION OF s_0 AND s_1 THROUGHOUT AN EXTENDED ROUND OF THE KEY INITIALIZATION PHASE (THREE CLOCK CYCLES)

Stage	s_0	s_1	Output			Next State	
			MUX_1	MUX_2	MUX_3	Register 1	Register 2
1	0	0	X^{2^k}	X	$X \oplus 1$	X^{r_1}	$X^{r_1} \oplus X \oplus 1$
2	0	1	$X^{r_1} \oplus X^{\left(2^k-1\right)}$	$X^{2^{2k}}$	$X^{r_1} \oplus X \oplus 1$	$X^{r_4} \oplus X^{r_2}$	$X^{r_4} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$
3	1	0	$X^{2^k\left(2^k-1\right)}$	X	$X^{r_4} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$	X^{r_3}	$X^{r_4} \oplus X^{r_3} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$

TABLE V
COUNT OF 1-BIT REGISTERS AND LOGIC GATES IN THE DIFFERENT COMPONENTS OF THE PROPOSED WG DESIGN

Component	N_R	N_A	N_X	N_O	N_I
WG Transform	87	4524	6292	146	4
FSM (Figure 6)	18	7	1	3	2

the 11-bit one-hot counter, the 3-bit one-hot counter is simply composed of a three stages circular shift register with set/reset inputs having the output of the last shift register fed to the input of the first register. Through adding the gates in the mentioned components to the number of gates of the FSM in Fig. 4 (Table II), the total hardware cost of the FSM in Fig. 6 is as shown in Table V.

The LFSR and the 4-to-1 MUX of the WG have same complexities as the ones in the MOWG (Table II). In addition, the WG design contains two 29-bit bit-wise complement operations [inverter symbol (a) and inverter symbol (b) in Fig. 3] and a $GF(2^{29})$ adder (computing the bit-wise XOR of initial feedback signal and the linear feedback signal). Let N_O^{WG}, N_I^{WG}, N_R^{WG}, N_A^{WG}, and N_X^{WG} denote the number of OR gates, inverters, 1-bit Registers, AND gates, and XOR gates in the proposed WG cipher, respectively. Therefore, through adding the corresponding number of gates in the $GF(2^{29})$ adder and in inverter symbols (a) and (b) to the number of gates in the 4-to-1 multiplexer, the LFSR (see Table II), and in the FSM, and the WG transform (Table V) one obtains

$$N_O^{\text{WG}} = 236, \quad N_I^{\text{WG}} = 94, \quad N_R^{\text{WG}} = 424,$$
$$N_A^{\text{WG}} = 5546, \quad N_X^{\text{WG}} = 7685.$$

2) Time Complexity: Here, the formulation for the critical path of the proposed WG design is derived. Notice that the LFSR delay in the WG is not a candidate for the critical path, because it still has less multipliers contributing to its delay, compared with the WG transform. In what follows, the formulation of the longest path during the key initialization phase is presented. After this, the running phase is proved to be the longest path of the cipher.

Let $T_{\text{clock}} \geq T_{\text{KIPh}}$ denotes the minimum clock period in the WG during the key initialization phase. During the three stages of an extended key initialization round, in order, the following three conditions hold:

$$T_{\text{clock}} \geq 24T_X + 4T_A + T_R \tag{25}$$
$$T_{\text{clock}} \geq 8T_X + 3T_A + T_R + 2T_O + 2T_I \tag{26}$$
$$T_{\text{clock}} \geq 8T_X + 5T_A + T_R + 4T_O + 4T_I \tag{27}$$

where the right hand sides in (25), (26), and (27) are simply the propagation delays during the first (generating X^{2^k-1}), second, and third stages of the extended key initialization round, respectively. It is clear that the right hand side of (25) is the largest, and hence, the longest path during the key initialization phase of the WG is

$$T_{\text{KIPh}} = 24T_X + 4T_A + T_R. \tag{28}$$

The delay of the longest path through the WG during the running phase is easily obtained by adding the delays of its components as follows:

$$T_{\text{RunPh}} = 32T_X + 5T_A + T_R. \tag{29}$$

From (28) and (29), the critical path of the cipher is (29).

V. RESULTS AND COMPARISONS

The following sections compare the proposed designs of the MOWG(29, 11, 17) and the WG(29, 11) ciphers with the corresponding previous implementations in [25], [10], and [24]. In addition, further optimizations and general applicability of the proposed algorithms are discussed.

A. Results from FPGA and ASIC Implementations

The proposed WG and MOWG designs, together with the WG in [10], have been realized using ASIC and FPGA implementations. The ASIC speed and area results are for the 65-nm CMOS technology based on Synopsys Design Compiler's estimate of area and clock speed before place-and-route with medium effort for optimizations. The power consumption readings have been conducted under 140-MHz frequency for all the designs. The FPGA designs have been synthesized using Xilinx Synthesis Tool [29]. The FPGA area and speed results are for Xilinx Virtex4 series FPGA device xc4vfx12sf363-10. All results are for post place-and-route and the power consumption results have been recorded for a frequency of 29 MHz for all the designs. The reported ASIC and FPGA results are listed in Tables VI and VII, respectively. Furthermore, theoretical results for the WG design in [24] are listed in Table VI. The WG-7, in the same table, is another member of the WG family based on an LFSR over $GF\left(2^7\right)$. In Tables VI and VII, the readings shown from the MOWG design in [25] were reported for the pipelined-with-reuse version of the transform. The following paragraphs analyze the reported results and compare the proposed WG and MOWG designs with the previous ones in the literature.

TABLE VI

RESULTS OBTAINED FROM ASIC IMPLEMENTATIONS (POSTSYNTHESIS) OF WG(29, 11)/MOWG(29, 11, 17). THE WG-7 RESULTS ARE FROM SOFTWARE IMPLEMENTATIONS PRESENTED IN [3]. KGATE IS THE AREA EQUIVALENCE IN TERMS OF NUMBER OF NAND GATES $\times 10^3$ [ESTIMATED AREA OF ONE NAND GATE IS 2.08 $(\mu\text{m})^2$]. THROUGHPUT IS THE # BITS PER CYCLE $\times$ SPEED (Mb/s $= 10^6$bit/s). Gbit $= 10^9$bit. THE RESULTS FOR THE WG(29, 11) HARDWARE IMPLEMENTATION PROPOSED BY [24] ARE BASED ON THEORETICAL ANALYSIS. EXP AND RET DENOTE THE DEPTH OF THE EXPRESSION AND RETURN STACKS

Cipher	Transform Architecture Type	Technology	Primary Optimization Target	# Clocks in Init. Phase	Bits/Cycle (Run Phase)	Area (KGate)	Latency (nsec)	Speed (MHz)	Throughput (Mbps)	Throughput Per Area (Kbps/Gate)	Dynamic Power (mW)	Energy (mJ/Gbit)
WG-7 @2MHz [3] (software)	Look-up Table	4-bit microcontroller MARC4 ATAM893 - D	-	10084	-	Code Lines = 1097, Exp/Ret = 7/4	-	-	0.098	-	-	-
WG-7 @8MHz [3] (software)	Look-up Table	8-bit microcontroller ATmega family	-	10074	-	SRAM = 0, Flash = 1100	-	-	0.28	-	-	-
WG [10]	Multiplier-based	CMOS 65nm	Area	22	1	33.2	6.94	144	144	4.34	7.28	50.6
WG [24]	Look-up Table (ROM)	-	-	-	1	319 Registers + 9000 XORs + 2^{29} ROM bits	-	-	-	-	-	-
MOWG [25]	Multiplier-based (Pipelined with Reuse)	CMOS 90nm	-	22	-	187 (Kμm^2)	-	1000	8500	45 (Kbps/μm^2)	-	-
WG (Figure 8)	Multiplier-based	CMOS 65nm	Area	66	1	19.9	4.45	224	224	11.2	4.45	19.8
MOWG (Figure 3)	Multiplier-based	CMOS 65nm	Area	22	17	26	6.62	151	2567	98.73	5.89	2.3

TABLE VII

RESULTS OBTAINED FROM FPGA IMPLEMENTATIONS (POSTPLACE AND ROUTE). THROUGHPUT IS THE # BITS PER CYCLE $\times$ SPEED (Mbps $= 10^6$bit/second). Gbit $= 10^9$bit

Cipher	Transform Architecture Type	Family	Synthesis Tool	Primary Optimization Target	# Clocks in Init. Phase	Bits/Cycle (Run Phase)	LUTs	Latency (nsec)	Speed (MHz)	Throughput (Mbps)	Throughput Per Area (Kbps/LUT)	Total Power (mW)	Energy (J/Gbit)
WG [10]	Multiplier-based	Virtex 4 (xc4vfx12sf363-10)	Xilinx XST	Area	22	1	6449	33.3	30	30	4.65	380	12.67
MOWG [25]	Multiplier-based (Pipelined with Reuse)	Stratix II (EP2S15F484C)	Mentor Graphics PrecisionRTL	-	22	-	4184	-	218	1853	443	-	-
WG (Figure 8)	Multiplier-based	Virtex 4 (xc4vfx12sf363-10)	Xilinx XST	Area	66	1	4044	29.4	34	34	8.41	187	5.5
MOWG (Figure 3)	Multiplier-based	Virtex 4 (xc4vfx12sf363-10)	Xilinx XST	Area	22	17	5512	28.6	35	595	108	342	0.57

The reported results show that the proposed WG takes longer to finish its initialization phase compared with the one in [10] (293 ns (ASIC)/1.94 ms (FPGA) in the proposed scheme compared with 152 ns (ASIC)/0.73 ms (FPGA) in [10]). This is not significant because initialization is executed only once per a run. The reported results also show that the proposed WG is superior to the one in [10] in terms of throughput, area, and power consumption. The proposed WG has lower latency, by 36% (ASIC) and 12% (FPGA), with respect to the one in [10]. In addition, accordingly, the speed/throughput of the proposed WG is increased by 55% (ASIC) and 13% (FPGA), compared with [10]. In addition, notice that the normalized throughput (proposed) is twice the one in [10]. This is due to the higher throughput and the significant reduction in area (area reduced by 40% for ASIC and by 37% for FPGA) of the proposed WG compared with the one in [10]. In addition, one can see that the proposed WG consumes less power (39% ASIC, 51% FPGA) and uses less than half the energy reported for [10].

The WG design in [24] requires 2^m ROM bits for a general WG over $GF(2^m)$. The area of the proposed WG is dominated by its field multipliers, which have space complexity quadratic in m. Specifically, for the WG(29, 11), 2^{29}-bits of ROM are required in [24] (in addition to 9000 XORs and 319 registers). There are no results in [24] about the running speed of the presented WG. According to a similar study on ROM- and multiplier-based MOWG designs by [25], ROM-based ASIC implementations are always larger and slower than using field multipliers, for $m > 11$.

The proposed MOWG design is expected to offer better area and speed compared to the one presented in [25]. The proposed MOWG has eight multipliers compared with nine in [25]. Therefore, its area is expected to be scaled down by a ratio close to $8/9$ with respect to the one in [25]. It is noted that the results from [25] are reported for the pipelined-with-reuse version of the transform. Applying pipeline-with-reuse techniques to the proposed MOWG would result in speed and area readings similar to the ones reported in [25]. For the nonpipelined and the pipelined (without reuse) versions, however, the proposed MOWG is expected to show lower area and a slightly higher speed/throughput, and lower latency, compared with the corresponding versions from [25]. This is due to the removed multiplier and the removed inverters from its critical path (Fig. 3). Notice that a 6-stage pipeline of the proposed MOWG offers 6-times the throughput that is reported for its nonpipelined version in Tables VI and VII (Section V-B). That is, almost double the throughput provided by the pipeline-with-reuse MOWG in [25].

The proposed WG offers higher clock speed, and better area and power consumption, compared with the proposed MOWG. The proposed MOWG has, however, higher throughput and better energy per bit. Most important, the WG has more good

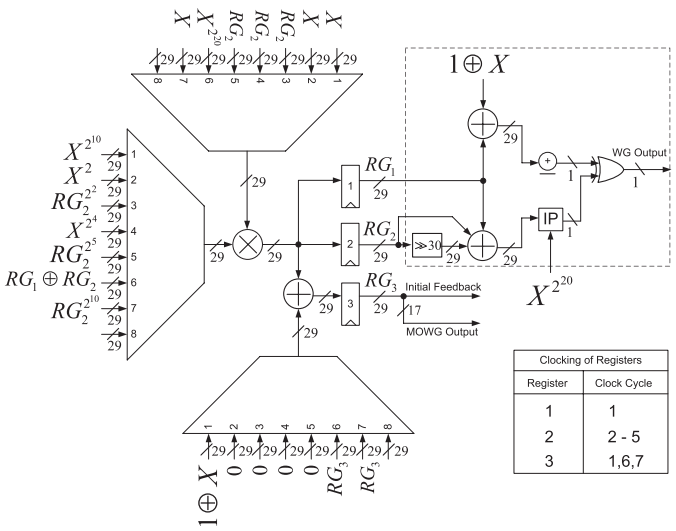

Fig. 9. Serial Implementation of MOWG/WG stream ciphers.

randomness properties than the MOWG cipher [10], [25]. Therefore, when security and randomness are critical for the application, the proposed WG design is preferred. If instead, throughput and area are the critical criteria for the application, then, the proposed WG design is superior for low area applications, whereas the proposed MOWG serves better for high speed applications. It is noted that one can apply serialization or pipelining to the WG/MOWG transforms for achieving lower area or higher throughput, if it is demanded by the application. This is discussed in the next section.

B. Discussion

This section discusses the serialization and pipeline techniques as further optimizations to the proposed WG and MOWG. In addition, the applicability of the proposed techniques to general MOWG/WG ciphers, in the NB, are considered.

For low throughput applications, smaller area can be achieved by serial computation of the MOWG/WG transforms. Fig. 9 shows how this is done using one multiplier. In this figure, the dotted square is used, only, for generating the WG stream bits. The rest of the diagram is common for MOWG and WG. The initialization round takes eight cycles for both transforms. During run phase of the MOWG, 17 output bits are generated every seven cycles. For the WG, a stream bit is produced every six cycles. The maximum propagation delay is equivalent to 17 levels of gate delays. Compared with 38 levels in (29) (WG) and 46 levels in (13) (MOWG), the clocks of the serial WG and MOWG are 2.2 and 2.7 times faster, respectively. Therefore, the throughput of the serial versions of the WG and the MOWG ciphers are almost $2/6$ and $3/7$ of the corresponding original ones in Figs. 3 and 8, respectively. The total gate counts for the serial versions of the transforms are 4155 (WG) and 4011 (MOWG). Compared with 11053 gates in the WG transform (Section IV-D1) and 14529 gates in the MOWG transform (Section III-D1), then, the area of the serial versions of the WG/MOWG transforms are almost $2/5$ and $2/7$ of their original architectures, respectively. If even lower area is demanded, a digit-level field multiplier [30], [31] can be deployed, adding more cycles for each multiplication.

The proposed schemes can achieve higher throughput through pipelined transforms. The LFSR should be reconstructed using the Galois-style feedback, or simply by placing the multiplication with β in between cells B_{i+1} and B_i. Otherwise, the LFSR's speed will constrain the pipelining. Fig. 3 shows how to achieve a 6-stage pipeline of the MOWG transform using 19 29-bit registers. The pipelined MOWG critical path has seven levels of logic gate delays. The corresponding throughput and run phase latency are $17/(T_A + 6T_X)$ and $6\,(T_A + 6T_X)$, respectively. Because (13) has 46 levels of logic gate delays, thus, the throughput of the pipelined MOWG is almost six times higher. Similarly, Fig. 8 shows a six-stage pipeline of the WG transform. From this figure, one can find the pipelined WG's latency and throughput as $6\,(T_A + 6T_X)$ and $1/(T_A + 6T_X)$, respectively [the latency during initialization is higher, i.e., $8\,(T_A + 6T_X)$]. Compared with the throughput that results from (29), this is almost five times higher. For even higher throughput, the unfolding technique presented in [32] can be deployed. Simply, the MOWG/WG LFSR is folded to generate n outputs ($2 \leq n \leq 11$) per a cycle. Hence, by implementing the same number of transforms, the throughput will be n-times higher at the expense of a proportional area increase.

Notice that (7) is a general form of the WG permutation (for any MOWG(m, l, d)). Because squarings are cyclic shifts in the NB, then, only the architecture of the power $2^k - 1$ will vary for different values of $k = \lceil \frac{m}{3} \rceil$. Through having the WGPerm, the MOWG transform is just a proper selection of d bits from the WGPerm [25]. In addition, notice that the compliment LFSR in (8) is general for any $GF\,(2^m)$. Similarly, except for the power $2^k - 1$, (24) is general for any WG(m, l). However, (14) is only applicable to $GF\,(2^m)$ where self-dual NB exist. Therefore, if there is not self-dual NB [33], the inner product that is used to compute $Tr\left(X^{2^{2k}}\left(X^{r_1} \oplus X^{2^k-1} \oplus X^{2^{3k}(2^k-1)}\right)\right)$ in Figs. 5 and 8 should be replaced with a field multiplication followed by a trace.

It is interesting to investigate the WG implementation in the PB. It is known that the PB offers area efficient multipliers, compared with the NB representation. There is, however, a penalty because of the additional space and propagation delay introduce by the squaring operations.

VI. Conclusion

Two new designs for the MOWG$(29, 11, 17)$ and the WG$(29, 11)$ ciphers have been proposed. As compared with the MOWG presented in [25], the proposed MOWG reduces the number of field multipliers in the transform by one through signal reuse. In addition, it increases the speed by eliminating two inverters delay from the critical path. This is accomplished by reconstructing the KIA and feedback polynomial of the LFSR. The proposed WG is an optimization of the proposed MOWG with trace (WG version).

It is obtained through using the new properties of the trace function for type-II ONB, accompanied with serialized computation of the initial feedback signal during key initialization phase.

The proposed designs have been implemented on ASIC and FPGA. The ASIC implementations show that the proposed WG implementation achieves better results compared with [10] for area, speed, and power consumption. The WG improves the power consumption by a 39% reduction, area by a 40% reduction, and speed by an increase of 55%. Similarly, the FPGA implementations show that the proposed WG achieves better results for area, speed, and power consumption compared with [10]. The power consumption is reduced by 51%, the area is reduced by 37%, and the speed is increased by 13%.

Based on these results, the proposed implementations of the MOWG(29, 11, 17) cipher and the WG(29, 11) cipher are promising candidates for high speed and limited resources platforms, respectively, where throughput, area, and power consumption are of critical importance and the guaranteed randomness properties are required.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their constructive comments and Canadian Microelectronics Corporation Microsystems for providing the required infrastructure and CAD tools that have been used in this paper.

REFERENCES

[1] S. Sen Gupta, A. Chattopadhyay, and A. Khalid, "HiPAcc-LTE: An integrated high performance accelerator for 3GPP LTE stream ciphers," in *Proc. 12th Int. Conf. Cryptol. India*, 2011, pp. 196–215.

[2] 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.

[3] 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.

[4] Bluetooth Special Interest Group. (2010, Jun.). *Adopted Bluetooth Core Specifications, Core Version 4.0*, Kirkland, WA, USA [Online]. Available: https://www.bluetooth.org/

[5] N. Courtois, "Fast algebraic attacks on stream ciphers with linear feedback," in *Proc. Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 2729. New York, NY, USA: Springer-Verlag, 2003, pp. 176–194.

[6] N. Courtois, "Algebraic attacks on combiners with memory and several outputs," in *Information Security and Cryptology—ICISC* (Lecture Notes in Computer Science), vol. 3506, C.-S. Park and S. Chee, Eds. New York, NY, USA: Springer-Verlag, 2005, pp. 3–20.

[7] W. Meier, E. Pasalic, and C. Carlet, "Algebraic attacks and decomposition of Boolean functions," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 3027, C. Cachin and J. Camenisch, Eds. New York, NY, USA: Springer-Verlag, 2004, pp. 474–491.

[8] F. Armknecht. (2004). *On the Existence of Low-Degree Equations for Algebraic Attacks* [Online]. Available: http://eprint.iacr.org/

[9] (2005). *eSTREAM—The ECRYPT Stream Cipher Project* [Online]. Available: http://www.ecrypt.eu.org/stream/

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

[11] *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*, 3rd Generation Partnership Project (3GPP), France, Jun. 2011, [Online]. Available: http://www.3gpp.org/

[12] *3rd Generation Partnership Project; Long Term Evaluation Release 10 and Beyond (LTE-Advanced); Proposed to ITU at 3GPP TSG RAN Meeting*, 3rd Generation Partnership Project (3GPP), France, 2009, [Online]. Available: http://www.3gpp.org/.

[13] T. Wu and G. Gong, "The weakness of integrity protection for LTE," in *Proc. 6th ACM Conf. Security Privacy Wireless Mobile Netw.*, Apr. 2013, pp. 79–88.

[14] H. Wu, T. Huang, P. Nguyen, H. Wang, and S. Ling, "Differential attacks against stream cipher ZUC," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 7658, X. Wang and K. Sako, Eds. Berlin Heidelberg, Germany: Springer-Verlag, 2012, pp. 262–277.

[15] A. Biryukov, D. Priemuth-Schmid, and B. Zhang, "Differential resynchronization attacks on reduced round SNOW $3G^{\oplus}$," in *e-Business and Telecommunications* (Communications in Computer and Information Science), vol. 222, M. Obaidat, G. Tsihrintzis, and J. Filipe, Eds. Berlin Heidelberg, Germany: Springer-Verlag, 2012, pp. 147–157.

[16] J.-S. No, S. Golomb, G. Gong, H.-K. Lee, and P. Gaal, "New binary pseudo-random sequences of period $2^n - 1$ with ideal autocorrelation," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 814–817, Mar. 1998.

[17] 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.

[18] G. Gong and Y. Nawaz. (2005). *The WG Stream Cipher* [Online]. Available: http://www.ecrypt.eu.org/stream/wgp2.html

[19] L. Chen and G. Gong, *Communication System Security*. London, U.K.: Chapman & Hall, 2012.

[20] H. Wu and B. Preneel, "Resynchronization Attacks on WG and LEX," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 4047, M. Robshaw, Ed. New York, NY, USA: Springer-Verlag, 2006, pp. 422–432.

[21] A. Mirzaei, M. Dakhilalian, and M. Modarres-Hashemi, "An Improved Attack on WG Stream Cipher," *Int. J. Comput. Sci. Netw. Security*, vol. 10, no. 4, pp. 45–52, Apr. 2010 [Online]. Available: http://paper.ijcsns.org/07_book/201004/20100408.pdf

[22] S. Ronjom and T. Helleseth. (2007). *Attacking the Filter Generator Over $GF(2^m)$* [Online]. Available: http://www.ecrypt.eu.org/stream/papersdir/2007/011.pdf

[23] A. Hariri and A. Reyhani-Masoleh, "Digit-level semi-systolic and systolic structures for the shifted polynomial basis multiplication over binary extension fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 11, pp. 2125–2129, Nov. 2011.

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

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

[26] 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.

[27] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Security*, vol. 1, no. 1, pp. 36–63, 2001.

[28] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson, "Optimal normal bases in $GF(p^n)$," *Discrete Appl. Math.*, vol. 22, no. 2, pp. 149–161, Feb. 1989.

[29] (2011). Xilinx, Inc., *Xilinx ISE Design Suite 13.3*, San Jose, CA, USA [Online]. Available: http://www.xilinx.com/

[30] A. Reyhani-Masoleh and M. A. Hasan, "Efficient digit-serial normal basis multipliers over binary extension fields," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 575–592, Aug. 2004 [Online]. Available: http://doi.acm.org/10.1145/1015047.1015053

[31] 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.

[32] C. Cheng and K. Parhi, "High-speed parallel CRC implementation based on unfolding, pipelining, and retiming," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 10, pp. 1017–1021, Oct. 2006.

[33] D. W. Ash, I. F. Blake, and S. A. Vanstone, "Low complexity normal bases," *Discrete Appl. Math.*, vol. 25, no. 3, pp. 191–210, 1989.

Hayssam El-Razouk received the B.Eng. degree in electrical and computer engineering from Beirut Arab University, Beirut, Lebanon, in 2002, with the first rank, and the M.Esc. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2006.

He was a Software Engineer with RedIron Technologies, London, ON, Canada, from 2006 to 2011. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Western University, London, ON, Canada.

Arash Reyhani-Masoleh (M'13) received the B.Sc. degree in electrical and electronic engineering from Iran University of Science and Technology, Tehran, Iran, in 1989, the M.Sc. degree in electrical and electronic engineering from the University of Tehran, Tehran, Iran, in 1991, both with the first rank, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2001.

He was with the Department of Electrical Engineering, Iran University of Science and Technology, from 1991 to 1997. From June 2001 to September 2004, he was with the Centre for Applied Cryptographic Research, University of Waterloo, where he was awarded a Natural Sciences and Engineering Research Council of Canada (NSERC) Post-Doctoral Fellowship in 2002. In October 2004, he was with the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada, where he is currently a Tenured Associate Professor. His current research interests include algorithms and VLSI architectures for computations in finite fields, fault-tolerant computing, and error-control coding.

He has been awarded a NSERC Discovery Accelerator Supplement in 2010. Currently, he serves as an Associate Editor for Integration, the VLSI Journal (Elsevier). He is a member of the IEEE Computer Society.

Guang Gong received the B.S. degree in mathematics in 1981, the M.S. degree in applied mathematics in 1985, and the Ph.D. degree in electrical engineering in 1990, from Universities in China. She received a Post-Doctoral Fellowship from the Fondazione Ugo Bordoni, Rome, Italy, and spent the following year there.

She was an Associate Professor with the University of Electrical Science and Technology of China, Sichuan, China, after returning from Italy. From 1995 to 1998, she was with several internationally recognized, outstanding coding experts and cryptographers, including Dr. Solomon W. Golomb, at the University of Southern California, Los Angeles, CA, USA. She was with the University of Waterloo, Waterloo, ON, 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. She has authored or co-authored more than 200 technical papers and two books, one co-authored with Dr. Golomb, entitled "Signal Design for Good Correlation for Wireless Communication, Cryptography and Radar," published by Cambridge Press in 2005, and the other coauthored with Dr. Lidong Chen, "Communication System Security," published by CRC 2012. Her current research interests are in the areas of sequence design, cryptography, and communication security.

Dr. Gong served as Associate Editor 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. She has 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, ON, Canada, in 2001, NSERC Discovery Accelerator Supplement Award, 2009, Canada, and ON Research Fund - Research Excellence Award, 2010, Canada, Best Paper Award of IEEE ICC 2012.