New Implementations of the WG Stream Cipher

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

Abstract

This paper presents two new hardware designs of the WG-128 cipher, one for the multiple output version (MOWG), and the other for the single output version (WG), based on type-II optimal normal basis (ONB) representation. The proposed MOWG design uses signal reuse techniques to reduce hardware cost in the MOWG transformation, while it increases the speed by eliminating the inverters from the critical path. This is accomplished through reconstructing the Key and Initial Vector (IV) loading Algorithm (KIA) and the feedback polynomial of the Linear Feedback Shift Register (LFSR). The proposed WG design uses properties of the trace function to optimize the hardware cost in the WG transformation. The ASIC and FPGA implementations of the proposed designs show that their areas and power consumptions outperform the existing implementations of the WG cipher.

1 Introduction

Synchronous bit-oriented stream ciphers are light weight symmetric-key cryptosystems which 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 Pseudo Random Sequence Generator (PRSG) and a seed (secret key). The seed changes for each run of the PRSG with each seed corresponds to a unique output sequence. Stream ciphers are heavily used in wireless communication and restricted resources applications such as 3GPP LTE-Advanced security suite [26], network protocols such as SSL, TLS, WEP and WPA [27], RFID tags [19], and bluetooth [4], to name some.

Traditionally, many hardware oriented stream ciphers have been built using LFSRs and Boolean functions with compact Algebraic Normal Forms (ANFs). However, the discovery of algebraic attacks made such a way of designing stream ciphers insecure [8, 9, 20, 6]. Many Non Linear Feedback Shift Registers (NLFSRs) based stream ciphers have been proposed in the eSTREAM stream cipher project [3], which have limited theoretical results about their randomness and cryptographic properties [19], and therefore their security depends on the difficulty of analyzing the design itself [22, 19].

The WG$(29, 11)$ (where 29 corresponds to $GF\left(2^{29}\right)$ and 11 corresponds to the number of stages in the LFSR), on the other hand, is a stream cipher submitted to the hardware profile in phase 2 of the eSTREAM project [3] which has been designed based on the WG transformations (Welch-Gong Transformation) [23] to produce key bit-streams with mathematically proved randomness aspects. Such properties include long period, ideal tuple distribution, large linear complexity, ideal two-level auto-correlation, cross correlation with an m-sequence has only three values, high non linearity, Boolean function with high algebraic degree and 1-resilient [14, 13, 22, 7]. The WG$(29, 11)$ is based on LFSR. This LFSR is composed of 11-stages with a linear feedback function over $GF\left(2^{29}\right)$. The revised version of the WG$(29, 11)$ [3] does not suffer the chosen IV (Initial Value) attack [28, 25]. Also, the number of key-stream bits which can be generated in a single key/IV pair is strictly less than the number of key-stream bits required to perform a linear span attack introduced in [28]. To defeat the algebraic attacks on LFSR based stream ciphers, the WG cipher relies on nonlinear Boolean functions with a large number of inputs (29-bits), a high degree and complex ANFs [22]. To overcome the implementation

complexity of the Boolean function, the WG cipher is designed in polynomial form instead of ANF [13, 22, 19]. Implementation of such functions requires multiplications and squaring over finite-fields, which can be done efficiently using ONB where the squaring operation is free of cost and multiplication complexity is practical [24].

Despite of its attractive randomness and cryptographic properties, few designs have been proposed for the hardware implementations of the WG$(29, 11)$. The authors of [13] adopt a direct design of the WG$(29, 11)$ using computation in the ONB, which requires seven multiplications and an inversion over $GF\left(2^{29}\right)$. An inversion in $GF\left(2^{29}\right)$ requires a number of multiplications which is logarithmic in 29. The inversion using Itoh-Tsujii algorithm requires $(\lfloor \log_2(28) \rfloor + H(28) - 1) = 4 + 3 - 1 = 6$ multiplications and 28 squarings in $GF\left(2^{29}\right)$, where $H(28)$ denotes the Hamming weight of 28 [5]. In [22], the authors replaced the inversion operation with a computation of the power $2^k - 1$ which requires 4 multiplications for $k = \left\lceil \frac{29}{3} \right\rceil = 10$ and reduced the other seven multiplications of the WG transformation in [13] by one through signal reuse. In [16], the author uses the interleaved approach with pre-computation [12] in order to enhance the speed of the cipher so that it outputs one bit per a clock cycle at the expense of requiring 2^{29} bits of ROM for hardware realization. In [17], the authors propose a multiple-bit output version of the WG cipher, called MOWG, which reduces the hardware cost through signal reuse by removing one multiplier from the WG transformation in [22], while it generates $d \leq 17$ output bits. Furthermore, [17] improves the hardware cost and throughput of the cipher through pipelining with reuse techniques. The keystream sequences generated by the MOWG cipher inherit many of the desirable randomness properties from the WG keystream sequences, such as large period, high linear complexity, high nonlinearity, and good autocorrelation properties [17].

In this paper, we propose two new designs, one for the multiple output WG (MOWG) cipher and the other one for the bit-oriented WG cipher which was initially proposed in [3], demonstrated by ASIC and FPGA implementations. The proposed designs optimize the area by reducing the number of multiplications in the MOWG/WG transformation. The ASIC implementation of the proposed WG design achieved an improvement of 40% in area, 39% in dynamic power consumption, and 17% in speed, while the FPGA implementation achieved an improvement of 37% in area and 51% in total power consumption, and 13% in speed, compared to [22].

The paper is organized as follows. In Section 2 we define the terms, notations, and give a brief background about the MOWG/WG cipher. Also, in this section we provide an overview about NB and ONB representations over $GF(2^m)$. Sections 3 and 4 presents our new hardware designs of the MOWG cipher and the WG cipher, respectively. Results based on FPGA and ASIC implementations of the new designs are discussed and compared to [22] in Section 5. Section 6 concludes the paper.

2 Preliminaries

This section defines the notations that will be used throughout this paper to describe the WG cipher and its operation:

- $GF(2)$, binary finite field with elements 0 and 1.
- $GF(2^m)$, binary extension field with 2^m elements. Each element in this field is represented as an m-bit binary vector.
- $Tr(Z) = Z + Z^2 + Z^{2^2} + \cdots + Z^{2^{m-1}}$, the trace function from $GF(2^m) \longrightarrow GF(2)$.
- Normal basis of $GF(2^m)$: Let β be an element of $GF(2^m)$ such that $\left\{\beta, \beta^{2^1}, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\right\}$ is a basis of $GF(2^m)$. Then $\left\{\beta, \beta^{2^1}, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\right\}$ is a normal basis of $GF(2^m)$ over $GF(2)$.
- Let $A = (a_0, a_1, \ldots, a_{m-1})$, $A \in GF\left(2^m\right)$, $a_i \in GF\left(2\right)$, and p is a positive integer, then

- $A^{2^p} = A \gg p = \left(a_{m-p}, \ldots, a_0, a_1, \ldots, a_{m-(p+1)}\right)$ represents the right cyclic shift of the coordinates of A, p-times.
- $A^{2^{-p}} = A \ll p = (a_p, \ldots, a_{m-1}, a_0, a_1, \ldots, a_{p-1})$ represents the left cyclic shift of the coordinates of A, p-times.

- If β is the generator of the normal basis then $\sum_{i=0}^{m-1} \beta^{2^i} = (1, 1, \ldots, 1) = 1 \in GF(2^m)$, i.e., the m-bit vector representing 1 is all ones vector. Therefore, addition of a field element, in normal basis representation, with 1 can be done by simply complementing the bits of that element.

- The trace of all the basis elements is one, i.e., $Tr\left(\beta^{2^i}\right) = 1$ where $0 \leq i \leq m-1$. Therefore, the trace of any element $Z = \sum_{i=0}^{m-1} z_i \beta^{2^i}$ represented in NB is given by

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

  i.e., the sum over $GF(2)$ of its coordinates with respect to the NB.

- $\oplus$ represents the bit-wise addition operator (XOR) in $GF(2^m)$.

- The inner product of two m-bit 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 $A \cdot B = a_0 b_0 + a_1 b_1 + \cdots + a_{m-1} b_{m-1} \in \{0, 1\}$.

- $C(Z) = Z^l \oplus \sum_{i=0}^{l-1} C_i Z^i$, $C_i \in GF(2^m)$ is the characteristic polynomial of and l-stages LFSR over $GF(2^m)$, from which the feedback recurrence relation can be derived as

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

  where $j \geq 0$, $A_i \in GF(2^m)$, and $(A_0, A_1, \ldots, A_{l-1})$ is the initial state of the LFSR.

2.1 Components and Operation of the WG Cipher

2.1.1 Components of the WG Cipher

The WG cipher is a key-stream generator composed of an 11-stages LFSR over $GF\left(2^{29}\right)$ and a 29-bit WG transformation, as shown in Figure 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 $GF\left(2^{29}\right)$, where $\beta = \alpha^{464730077}$ is the generator of the ONB and α is a root of the defining polynomial of $GF\left(2^{29}\right)$ given by [22]

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

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

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

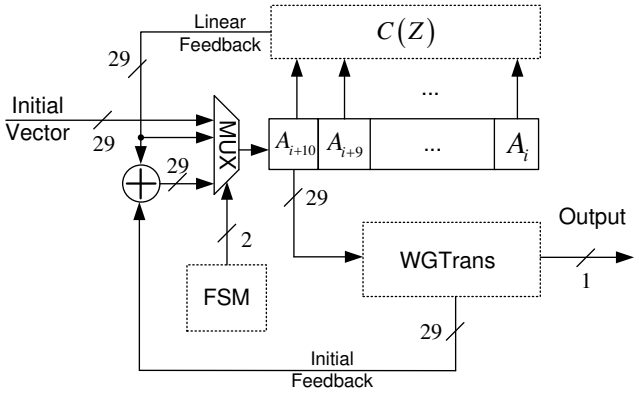

Figure 1: WG generator [13, 22, 17, 7]. Initial Vector is the input during the loading phase. (Linear Feedback $\oplus$ Initial Feedback) is the input during the key initialization phase. Linear Feedback is the input throughout the PRSG phase.

where

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

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 $k = \lceil \frac{29}{3} \rceil$ [17]. This results in a binary key-stream of period $2^{319}-1$ [13, 22]. Here we use the method from [10] for computing r_i's, the WG transformation's exponents. Although our WG transformation's exponents are different from the exponents used in [13, 22] (which are taken from [23]), the WG transformations are identical for both representations [7]. The MOWG cipher, on the other hand, uses the same formulation presented in (5), however, without the trace, and outputs 17 concatenated bits arbitrarily selected from the 29 bits of the WG permutation's output [17].

2.1.2 Operation of the WG Cipher

In Figure 1, the Finite State Machine (FSM) controls the three phases of the operations in the cipher, namely, load key and IV phase, key initialization phase, and running phase. The loading phase takes 11 clock cycles in which the secret key and the initial vector are loaded to the LFSR, one stage at a time. The key initialization phase takes 22 clock cycles, during which the input to the LFSR is the bit-wise XOR of the Linear Feedback from the LFSR with the Initial Feedback signal $WGPerm(A_{i+10} \oplus 1)$. The output of the key initialization phase serves as a session key for the running phase. By receiving the 34-th clock signal, the generator starts to generate its key-stream bits. While in the run phase, the only input to the LFSR is the Linear Feedback signal.

2.2 Multiplication in $GF(2^m)$

An element of $GF(2^m)$ is usually represented with respect to a basis such as polynomial basis (PB) or normal basis (NB). It is well known that a NB always exists in the field $GF(2^m)$ over $GF(2)$ [18]. By finding an element $\beta \in GF(2^m)$ such that $\left\{\beta, \beta^{2^1}, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\right\}$ is a basis of $GF(2^m)$ over

$GF(2)$, any element $A \in GF(2^m)$ can be represented as $A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = (a_0, a_1, \ldots, a_{m-1})$, where $a_i \in \{0,1\}$ is the i-th coordinate of A with respect to the NB.

Optimal Normal Basis (ONB) is a subclass of the NB class in which the multiplication operations have lowest possible complexity [21]. There are two types of ONB [11], namely type-I and type-II. A type-t ONB, $t \in \{1,2\}$, exists iff $p = tm+1$ is prime and $gcd\left(\frac{tm}{k}, m\right) = 1$, where k is the multiplicative order of 2 modulo p [15].

Let $A, B \in GF(2^m)$ where $A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$ and $B = \sum_{i=0}^{m-1} b_i \beta^{2^i}$ denote their representations with respect to the ONB, then the multiplication of A and B, i.e., $C = AB = \sum_{i=0}^{m-1} c_i \beta^{2^i}$ can be obtained using the multiplication formation presented in Corollary 1 of [24]. This algorithm requires m^2 AND gates and $\frac{3}{2}m(m-1)$ XOR gates, while it introduces a time delay of $T_A + (1 + \lceil \log_2(m) \rceil) T_X$, where T_A and T_X are the delays through an AND gate and an XOR gate, respectively.

For $m = 29$ which is used in this paper, there is a type-II ONB over $GF(2^{29})$.

3 Optimized Hardware Design of the MOWG Cipher

This section presents a new hardware design of the MOWG$(29, 11, 17)$ cipher, where 29 corresponds to $GF\left(2^{29}\right)$, 11 is the number of stages in the LFSR, and 17 is the number of output bits. In this design, the MOWG transformation uses $3 + N_{Mult}^{2^k-1}$ multipliers, compared to $4 + N_{Mult}^{2^k-1}$ multipliers in [17], where $k = \left\lceil \frac{29}{3} \right\rceil = 10$ and $N_{Mult}^{2^k-1} = 4$ is the number of multipliers required to compute the $\left(2^k - 1\right)'s$ power of the LFSR's output signal. Also, in the proposed design, the LFSR has been reconstructed in order to remove the inverters from the critical paths during the PRSG phase/key initialization phase, in an attempt to improve the overall speed of the cipher. In what follows, we first introduce the reduced area MOWG transformation design, followed by presenting the LFSR/KIA algorithm changes required for speed improvement. Then, we discuss the architecture of the finite state machine, and we end up this section by formulations of the space and the time complexities of the resulted design.

3.1 Reducing the Hardware Complexity of the MOWG Transformation

The hardware cost of the MOWG$(29, 11, 17)$ cipher is dominated by its transformation's finite field multipliers. Hence, any decrease in the number of multipliers of the transformation would minimize the area penalty of the overall cipher. This subsection shows how the hardware cost of the MOWG transformation can be reduced by 1 multiplier compared to [17]. In what follows, we first introduce the required mathematical formulation, followed by a presentation of the resulting MOWG transformation architecture.

3.1.1 Formulation

In Proposition 1 we derive a new formulation for the MOWG transformation which reduces the required finite field multipliers through proper reuse of common signals.

Proposition 1. *Let $X = A_{i+(l-1)} \oplus 1$ be the complement of the MOWG(m, l, d) LFSR's output, where the LFSR is of length l over $GF(2^m)$ and the output of the cipher has d-bits. Then, the m-bit WG permutation of a MOWG cipher given by (6) can be generated as follows*

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

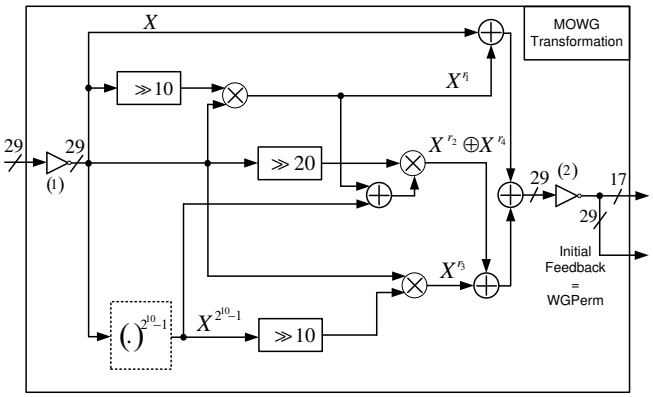

Figure 2: The realization of the MOWG transformation using (7). $X = A_{i+10} \oplus 1$ is the bit-wise complement of the LFSR's output, $r_1 = 2^k + 1$, $r_2 = 2^{2k} + 2^k + 1$, $r_3 = 2^{2k} - 2^k + 1$,$r_4 = 2^{2k} + 2^k - 1$, $k = \lceil \frac{29}{3} \rceil = 10$.

Where $k = \lceil \frac{m}{3} \rceil$.

Proof. By noticing that $X^{2^{2k}}$ is a common factor of the exponent terms $2^{2k} + \left(2^k + 1\right)$ and $2^{2k} + \left(2^k - 1\right)$ in equation (6), we complete the proof. □

It is clear that in addition to the multiplication operations involved in computing the signal $X^{\left(2^k-1\right)}$, (7) requires only three more multiplications to generate the signals X^{2^k+1}, $X^{2^k\left(2^k-1\right)+1}$, and $X^{2^{2k}}\left(X^{\left(2^k+1\right)} \oplus X^{\left(2^k-1\right)}\right)$.

3.1.2 Architecture

The architecture of the MOWG$(29, 11, 17)$ transformation which realizes the $WGPerm$ given in (7), is shown in Figure 2. In the MOWG$(29, 11, 17)$ we have $k = 10$ and, hence, the signal X^{2^k-1} requires 4 multiplications and 4 squaring operations (which is free of cost in ONB) [17]. Therefore, the architecture of Figure 2 requires a total of 7 $GF\left(2^{29}\right)$ 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 $GF(2^{29})$ (realized as a binary tree structure in order to minimize the corresponding time delay) of the signals X, $X^{r_1} = X^{2^k+1}$, $X^{r_2} \oplus X^{r_4} = X^{2^{2k}}\left(X^{\left(2^k+1\right)} \oplus X^{\left(2^k-1\right)}\right)$, and $X^{r_3} = X^{2^k\left(2^k-1\right)+1}$. The signals X^{2^k} and $X^{2^{2k}}$ are obtained by right cyclic shifts of X, k and $2k$ times, respectively. X^{2^k+1} is generated by multiplying X with X^{2^k} in $GF\left(2^{29}\right)$. $X^{2^k\left(2^k-1\right)}$ is the right cyclic shift of $X^{\left(2^k-1\right)}$, k times, and $X^{2^k\left(2^k-1\right)+1}$ is generated by multiplying $X^{2^k\left(2^k-1\right)}$ with X in $GF\left(2^{29}\right)$. In Figure 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) to be used by the Initial Feedback. Seventeen bits of the $WGPerm$ are the output of the MOWG in the running phase [17].

3.2 Improving the Critical Path Delay of the MOWG Transformation

The time delay through the transformation of the MOWG$(29, 11, 17)$ dominates the delay of the overall cipher (see Subsection 3.4.2). In this subsection we slightly reduce the delay through this transformation. This is accomplished by removing inverter (1), and by reallocating inverter (2), away from the critical

paths of both the PRSG phase and the key initialization phase. This reduces the delay of the critical path by an amount equivalent to the delay of two 1-bit inverters; however, the MOWG transformation delay is still dominating due to the delays of 5 serially connected field multipliers. We start by discussing the required mathematical formulation, then we present the resulting new architecture of the cipher.

3.2.1 Formulation

During the key initialization phase and the PRSG phase, inverter (1) in Figure 2 generates the complement of the content in cell $i+10$ of the LFSR. Notice that this cell holds the feedback from the LFSR during the PRSG phase, and the bit-wise XOR of the LFSR and the MOWG transformation 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 $\underline{B} = \{B_i = 1 \oplus A_i,\, 0 \leq i \leq 2^{319} - 1\}$, where $B_i \in GF\left(2^{29}\right)$ and $\{A_i\}$ is the sequence generated by the feedback polynomial in (3) over $GF\left(2^{29}\right)$. We refer to sequence $\underline{B}$ as the complement sequence of the sequence $\{A_i\}$. The following proposition shows how this can be accomplished for an LFSR with a general feedback polynomial of degree l over $GF\left(2^m\right)$.

Proposition 2. *Let $\underline{B}$ be the complement sequence of a sequence $\underline{A} = \{A_i,\, 0 \leq i \leq 2^{ml} - 1\}$, where $A_i \in GF\left(2^m\right)$ and $\underline{A}$ is generated by (2), then, $\underline{B}$ can be 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 $j \geq 0$, and the initial state of $\underline{B}$ is

$$B_i = 1 \oplus A_i,$$

for $0 \leq i \leq l-1$.

Proof. We have the following computation:

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

$j \geq 0$. Using (2) in (9) we get

$$B_{j+l} = \sum_{i=0}^{l-1} C_i A_{i+j} \oplus 1,$$

and by noticing $2C_i = 0$ we obtain

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

Thus the assertion is true. □

Proposition 3. *The WG permutation in (7) does not change if we set $X = B_{i+10}$.*

Proof. In (7), the term $1 \oplus X$ equals to the LFSR output in Figure 1, that is $1 \oplus X = A_{i+10}$. From Proposition 2 we have $A_{i+10} = B_{i+10} \oplus 1$, which completes the proof. □

Notice that the term $\left(\sum_{i=0}^{l-1} C_i\right) \oplus 1$ in (8) is a constant term and, hence, its addition in $GF\left(2^{29}\right)$ can be realized with a number of NOT gates equal to its hamming weight. For the LFSR of the MOWG$(29, 11, 17)$, replacing the coefficients of its feedback polynomial from (3) in (8) we get $\left(\sum_{i=0}^{l-1} C_i\right) \oplus 1 = \beta \oplus 1$ 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} in (7) prior to the summation of these terms. Doing so would reallocate inverter (2) from its current position. However, it is required that this reallocation does not result in making the delay in the new path higher than the current maximum delay of the MOWG transformation. For this reason, we choose to move the inverter to complement X before it is added to X^{r_1}. This path can be seen at the top of Figure 2 which has the lowest delay with only two $GF\left(2^{29}\right)$ adders between inverters (1) and (2).

3.2.2 Modified KIA Algorithm

Modifying the LFSR of MOWG$(29, 11, 17)$ as given in (8) requires stage 10 of the LFSR (left most stage) to hold the complement of the Initial Vector during the KIA loading phase. This imposes a corresponding change which should be applied to the KIA loading algorithm. Therefore, it is required to complement the Initial Vector input before it is loaded to the modified LFSR. This can easily be implemented by inserting 29 inverters at the multiplexer's input which receives the Initial Vector in Figure 1.

3.2.3 Architecture

Here we present the overall proposed architecture of the MOWG$(29, 11, 17)$ cipher, as shown in Figure 3. In this figure, the FSM controls the input to the LFSR for each phase of operation. In the same figure, due to the bit-wise complement operator denoted by (a), the LFSR receives the complemented Initial Vector signal during the loading phase and, 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 Figure 1, i.e. $B_i = A_i \oplus 1$, $0 \leq i < 10$. When the key initialization phase starts, the bit-wise XOR of the Initial Feedback signal and the Linear Feedback signal applies to the input of the LFSR. Note that the Linear Feedback signal in Figure 3 is generated by (8), which is equivalent to $B_i = A_i \oplus 1$, $11 \leq i < 33$, and therefore is the complement of corresponding one in Figure 1. However, according to Proposition 3, the Initial Feedback signal in Figure 3 has the same value as the one generated in Figure 2. This means that the input to the LFSR during the key initialization phase in Figure 3 is the complement of the input to the LFSR during the same phase in Figure 1. Throughout the PRSG phase, the only input to the LFSR is the Linear Feedback signal $B_i = A_i \oplus 1$, $33 \leq i < 2^{319} - 1$, which results in generating the same key-stream bits of Figure 2 at the output of the MOWG transformation of Figure 3, as can be seen from Proposition 3. 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 to the one in Figure 2. The revised architecture of Figure 3 has additional $H(\beta \oplus 1) = 28$ inverters, compared to Figure 1, where $H(\theta)$ denotes the hamming weight of a field element θ. This is due to the new constant term $\beta \oplus 1$ in the feedback polynomial of Figure 3.

3.3 The Finite State Machine

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

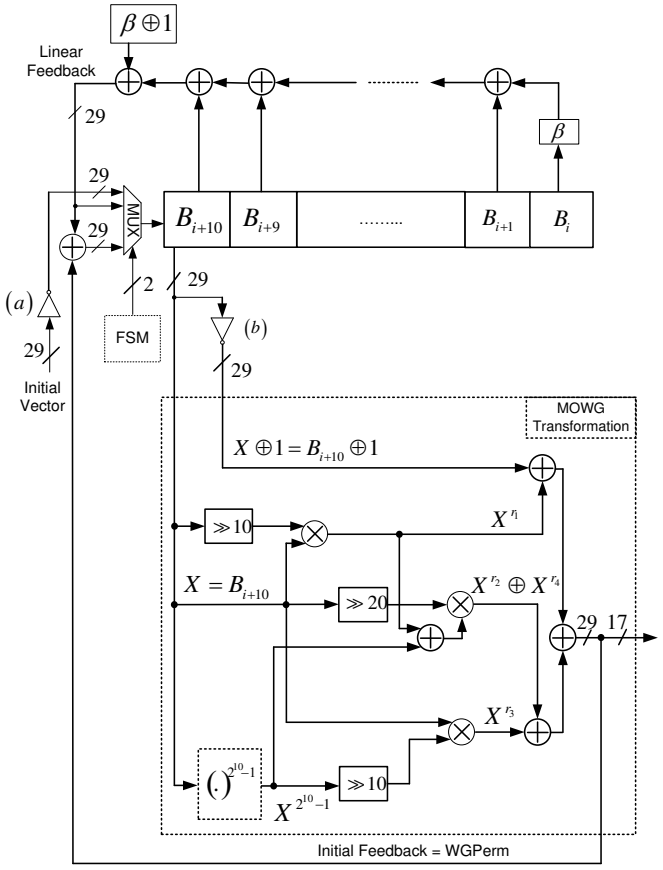

Figure 3: Proposed design of the MOWG$(29, 11, 17)$ cipher.

Figure 4 presents a block diagram showing the internal components of the FSM. The FSM has two inputs, namely clk and reset, each of which is a 1-bit signal, while there are two output signals denoted as op0 and op1. The reset signal 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 bit of position 0 is the only bit set to a high logic level. Also, when the reset signal is low, the 2-bit binary counter resets its state to $(0, 0)$. Due to 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 return to its initial state after 11 clock cycles where it triggers the 2-bit binary counter to increment and, hence, flag the start of the key 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 1. The op0 and op1 signals selects one of the three inputs of the multiplexer in Figure 3 to be connected 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 which 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.

3.4 Space and Time Complexities

In this subsection, we provide the space and time complexities of the MOWG design in Figure 3.

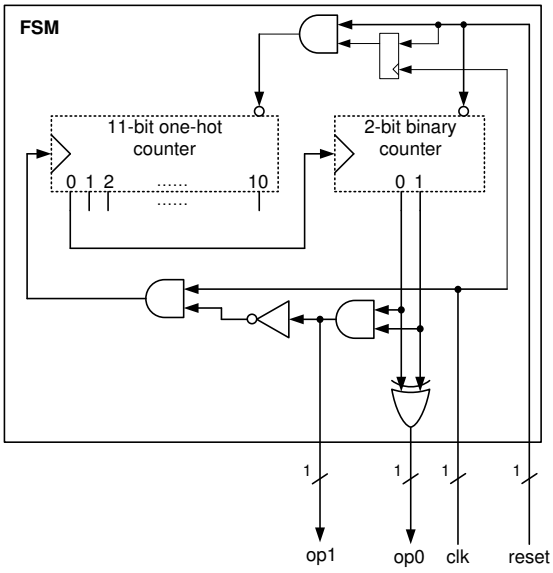

Figure 4: FSM of the MOWG.

Table 1: 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

3.4.1 Space Complexity

We evaluate the space complexity in terms of number of gates in each component, in order 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.

MOWG Transformation: The MOWG transformation dominates the hardware complexity of the MOWG design, as it consists of 7 field multipliers and four $GF\left(2^{29}\right)$ adders. A $GF\left(2^{29}\right)$ adder requires 29 XOR gates. Also, we deploy the multiplier in [24] in our implementation, which has 841 AND gates and 1218 XOR gates. Therefore, the total hardware cost of the WG transformation can be written as

$$\begin{cases} N_A = 7 \times 841 = 5887, \\ N_X = 7 \times 1218 + 4 \times 29 = 8642. \end{cases} \tag{10}$$

LFSR: The LFSR contains 11-stages of 29-bit shift registers, and a feedback polynomial. The feedback polynomial is composed of 1 field multiplier (with a constant)[1], 6 $GF\left(2^{29}\right)$ additions, and $H\left(\beta \oplus 1\right) =$

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

28 Inverters. Therefore, the hardware complexity of the LFSR can be summarized as follows

$$\begin{cases} N_R = 11 \times 29 = 319, \\ N_X = 29 \times 6 + 1218 = 1392, \\ N_I = 28, \\ N_A = 841. \end{cases} \tag{11}$$

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. 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, one OR gate, and an Inverter. Therefore, the total cost of the 4-to-1 29-bit multiplexer can be summarized as

$$\begin{cases} N_A = 3 \times 29 \times 2 = 174, \\ N_O = 3 \times 29 = 87, \\ N_I = 3 \times 29 = 87. \end{cases} \tag{12}$$

FSM: From Figure 4, the total number of AND, XOR and Inverter gates of the FSM are

$$\begin{cases} N_A = 3, \\ N_X = 1, \\ N_I = 1, \end{cases} \tag{13}$$

respectively. Notice that the 11-bit one-hot counter is simply composed of 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 register, while the 2-bit binary counter, on the other hand, is built from two JK Flip Flops with the two inputs of the first FF being pulled to high logic and the output of this FF drives the two inputs of the second FF. Thus, one can find the total number of one-bit registers as

$$N_R = 11 + 2 + 1 = 14. \tag{14}$$

Proposition 4. *The overall space complexity of the MOWG cipher in Figure 3 is as follows*

$$\begin{cases} N_O^{MOWG} = 87, \\ N_I^{MOWG} = 174, \\ N_R^{MOWG} = 333, \\ N_A^{MOWG} = 6905, \\ N_X^{MOWG} = 10064. \end{cases} \tag{15}$$

Where 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 Figure 3, respectively.

Proof. In addition to the FSM, the 4-to-1 29-bit multiplexer, the LFSR, and the MOWG transformation, the MOWG cipher contains two 29-bit bit-wise complement operators (inverter symbol (a) and inverter symbol (b) in Figure 3) and a $GF\left(2^{29}\right)$ adder (computing the bit-wise XOR of Initial Feedback signal and the Linear Feedback signal). Therefore, by adding the corresponding number of gates in this $GF\left(2^{29}\right)$ adder and in inverter symbols (a) and (b) to (14), (13), (12), (11), and (10) we get (15). □

3.4.2 Time Complexity

Here we derive the formulation for computing the critical path delay of the MOWG cipher in Figure 3. There are three critical paths in the MOWG:

- Critical path of the LFSR.
- Critical path along the MOWG transformation during the key initialization phase.
- Critical path along the MOWG transformation during the run phase.

The LFSR's path has one multiplication and five finite field additions (during key initialization phase). If the additions are implemented in a binary tree structure, this results in a time delay of

$$T_A + (1 + \lceil \log_2 (5) \rceil + \lceil \log_2 (29) \rceil) T_X = T_A + 9T_X, \tag{16}$$

where the delay through a finite field multiplier is $T_A + (1 + \lceil \log_2 (29) \rceil) T_X$ [24]. On the other hand, the delay through the two MOWG transformation's paths has 5 multipliers in series, which corresponds to a delay of

$$5 (T_A + 6T_X) = 5T_A + 30T_X. \tag{17}$$

From (16) and (17), we claim that the longest path of the MOWG cipher passes through its transformation.

Proposition 5. *The delay of the MOWG during the key initialization phase is the maximum delay and is as follows*

$$T_{KIPh} = 7T_A + 34T_X + T_R + 2T_O + 2T_I \tag{18}$$

where T_{KIPh} denotes the maximum time delay through the MOWG during the key initialization phase.

Proof. From Figure 3, the critical path of the proposed MOWG during the running phase includes the delays of a 29-bit Register, 5 field multipliers in series (from [24]), and 3 $GF\left(2^{29}\right)$ adders. This results in the delay stated in (19).

$$T_{RunPh} = 5T_A + 33T_X + T_R \tag{19}$$

where T_{RunPh} denotes the maximum time delay through the MOWG during the running phase. In the same figure, the critical path of the MOWG during the key initialization phase includes the delays of 4 $GF\left(2^{29}\right)$ adders, 5 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 2 2-to-1 1-bit multiplexers in series, which is equivalent to the sum of the delays through 2 AND gates, 2 OR gates, and 2 Inverters. Therefore, the delay of the MOWG during the key initialization phase follows (18). Comparing the two equations, it is clear that $T_{KIPh} > T_{RunPh}$. □

4 Low Complexity WG Cipher

In this section, we propose a new design of the WG$(29, 11)$ cipher presented in [13, 22].

The proposed WG$(29, 11)$ design offers a higher clock speed, and a better area and power consumption, in comparison to the MOWG$(29, 11, 17)$ design which has been proposed in the previous section. However, the MOWG design has a higher throughput and requires smaller number of clock cycles to conduct its key initialization phase, compared to the WG design.

The proposed WG design considers Figure 3 with an added trace to the output of the $WGPerm$ as a starting point for optimization. This section starts with presenting properties of the trace function

when the elements in $GF(2^m)$ are represented in ONB of type-II. It is assumed that m is selected such that $GF(2^m)$ has a type-II ONB over $GF(2)$ denoted by

$$\left\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\right\},$$

(which is true for $m = 29$), with β being the generator of the ONB (see Section (2.2)). The proposed WG design utilizes these properties in order to minimize the hardware complexity of the WG transformation. Note that the proposed design eliminates some signals which are required to generate the Initial Feedback signal, which is needed to conduct the key initialization phase of the cipher. Missing of the Initial Feedback signal is recovered by introducing a new serialized scheme to generate it. At the end of this section, the hardware and the time complexities of the new implementation are provided.

4.1 Properties of the Trace Function for Type-II ONB

This subsection presents a new method for computing the trace of a multiplication of two field elements when the representation is a type-II ONB representation. Also, two corollaries are deduced from the proposed method.

Fact 1. *[21] Let $\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$ be a type-II ONB in $GF(2^m)$. Then*

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

and

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

In other words, a type-II ONB is a self-dual basis. Thus we have the following property.

Proposition 6. *In a type-II ONB, the trace of the field multiplication of any two elements $A = (a_0, a_1, \ldots, a_{m-1})$ and $B = (b_0, b_1, \ldots, b_{m-1})$ in $GF(2^m)$ can be computed as the inner product of A and B, that is:*

$$Tr(AB) = \sum_{i=0}^{m-1} a_i b_i. \tag{20}$$

Proof. We have the following derivation:

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

where the last result is obtained using Fact 1. □

Proposition 6 implies that the trace of a field multiplication of two elements represented in type-II ONB can be easily implemented in hardware using m 2-input AND gates and $m-1$ 2-input XOR gates.

Corollary 1. *In type-II optimal normal basis, the two relations below are valid for any two elements A and B in $GF(2^m)$*

$$Tr(AB) = Tr((A \gg n)(B \gg n)) = \sum_{i=0}^{m-1} a_{i-n} b_{i-n}, \tag{21}$$

and

$$Tr(AB) = Tr((A \ll n)(B \ll n)) = \sum_{i=0}^{m-1} a_{i+n} b_{i+n}, \tag{22}$$

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 we get

$$Tr(AB) = Tr\left(A^{2^{\pm n}} B^{2^{\pm n}}\right). \tag{23}$$

By using Proposition 6, the proof is completed by realizing that the squaring operation X^2, and the square root operation $X^{2^{-1}}$, in ONB representation, are simply the right cyclic shift and the left cyclic shift of the coordinates of X, 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. *In type-II optimal normal basis, the following relation holds for any three or more elements A, B, C, ..., and D in $GF(2^m)$:*

$$\begin{aligned} Tr(AD) + Tr(BD) + Tr(CD) + \cdots & \\ = \sum_{i=0}^{m-1} (a_i + b_i + c_i + \cdots) d_i. & \end{aligned} \tag{24}$$

Proof. Let A, B, C, ..., and D be any three or more arbitrary elements from the finite field $GF(2^m)$. We have the following derivation:

$$\begin{aligned} Tr(AD) + Tr(BD) + Tr(CD) + \cdots & \\ &= Tr(AD \oplus BD \oplus CD \oplus \cdots) \\ &= Tr((A \oplus B \oplus C \oplus \cdots) D) \\ &= \sum_{i=0}^{m-1} (a_i + b_i + c_i + \cdots) d_i, \end{aligned}$$

where the last result follows from Proposition 6. □

4.2 Optimizing the WG Transformation's Hardware for the Running Phase

Here, we show how Proposition 6 and Corollaries 1 and 2 can be used to further reduce the number of field multiplications in the WG transformation in Figure 3 (with trace). Before proceeding, it is important to mention that by applying (20) of Proposition 6, one can directly 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, a careful application of the proposed method requires first to apply the trace function to the individual signals of the WG permutation in (7) in such away it becomes possible to apply (20) only to the traces of multiplication terms which are not used anywhere else. As it is clear from Figure 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), while the signal X^{r_1} is required in generating $X^{r_2} \oplus X^{r_4}$(see (6) in Section 2.1.1 for the values of $r_i's$). The first two signals are generated as follows

$$\begin{cases} X^{r_2} \oplus X^{r_4} = X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k-1} \right), \\ X^{r_3} = X X^{2^k\left(2^k-1\right)}. \end{cases} \tag{25}$$

Therefore, applying the trace function to (25) we get

$$\begin{cases} Tr\left(X^{r_2} \oplus X^{r_4}\right) = Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k-1} \right) \right), \\ Tr\left(X^{r_3}\right) = Tr\left(X X^{2^k\left(2^k-1\right)} \right). \end{cases} \tag{26}$$

Using (26), the WG transformation becomes

$$\begin{aligned} WGTrans =& Tr\left(1 \oplus X \oplus X^{r_1}\right) + Tr\left(X X^{2^k\left(2^k-1\right)} \right) \\ &+ Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k-1} \right) \right). \end{aligned} \tag{27}$$

Applying a right cyclic shift of $2k$-stages to X and $X^{2^k\left(2^k-1\right)}$ in the term $Tr\left(X X^{2^k\left(2^k-1\right)} \right)$ of (27) does not change the value of the trace, i.e.

$$Tr\left(X X^{2^k\left(2^k-1\right)} \right) = Tr\left((X)^{2^{2k}} \left(X^{2^k\left(2^k-1\right)} \right)^{2^{2k}} \right). \tag{28}$$

Using (28) in (27) gives

$$\begin{aligned} WGTrans =& Tr\left(1 \oplus X \oplus X^{r_1}\right) + Tr\left(X^{2^{2k}} X^{2^{3k}\left(2^k-1\right)} \right) \\ &+ Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k-1} \right) \right). \end{aligned} \tag{29}$$

Taking $X^{2^{2k}}$ as a common factor in (29) we get

$$\begin{aligned} WGTrans =& Tr\left(1 \oplus X \oplus X^{r_1}\right) + \\ & Tr\left(X^{2^{2k}} \left(X^{r_1} \oplus X^{2^k-1} \oplus X^{2^{3k}\left(2^k-1\right)} \right) \right). \end{aligned} \tag{30}$$

Notice that by applying Corollary 2 to (30), 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}). Figure 5 captures the resulting architecture of the WG transformation after incorporating the modifications in (30) to the WG transformation

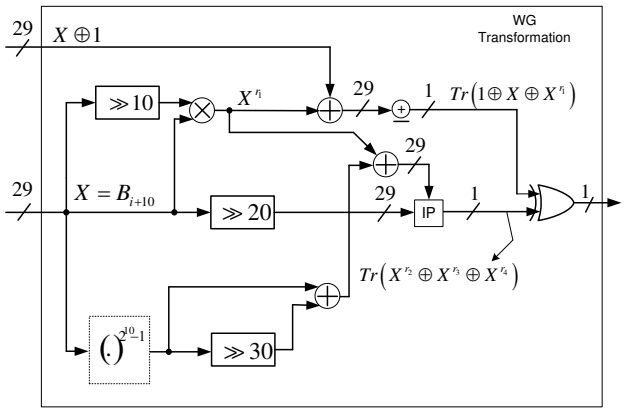

Figure 5: The proposed design of the WG transformation. The block denoted by "IP" generates the inner product of the two 29-bit inputs (see Section 2), while $\underline{\oplus}$ adds the 29-bits at its input over $GF(2)$.

in Figure 3 (with trace). This architecture uses $1 + N_{Mult}^{2^k-1}$ field multipliers, where $N_{Mult}^{2^k-1} = 4$ for $k = 10$ (see Section 3), i.e., four multipliers less than the WG transformation presented in [22].

In Figure 5, the key stream bits are obtained by XORing $Tr(1 \oplus X \oplus X^{r_1})$ with $Tr(X^{r_2} \oplus X^{r_3} \oplus X^{r_4})$. $Tr(1 \oplus X \oplus X^{r_1})$ has been generated in the same way as obtained in Figure 3. 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 transformation because $Tr(X^{r_2} \oplus X^{r_3} \oplus X^{r_4})$ is generated directly from the signals $X^{2^{2k}}$, X^{r_1}, X^{2^k-1}, and $X^{2^{3k}(2^k-1)}$ using an inner product operation in place of the field multipliers, as it is stated in (30). 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.

4.3 Serializing the Computation of the Initial Feedback Signal

This subsection presents a new method for the recovery of the Initial Feedback signal through serialized computation. To accomplish the multiplication operations, which are required in this computation, we use the existing finite field multiplier which is used in generating the signal X^{r_1} in Figure 5. The proposed scheme generates the Initial Feedback signal by serially computing it over three consecutive clock cycles. We denote this complete round of the serialized Initial Feedback computation (three clock cycles) as an "extended key initialization round", while a single clock cycle version of this computation (as in the MOWG design) is denoted as a "simple round". Therefore, with serialization, the entire key initialization phase requires $3 \times 22 = 66$ 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 clock cycles required for the running phase.

The expansion of the key initialization round from 1 to 3 clock cycles can be established through the support of a new FSM's control signal, namely, lfsr_clk (Figure 6), which controls the clock input of the LFSR and triggers it to shift once every three clock cycles. Also, in order to compute the Initial Feedback signal on three stages, a new hardware module denoted as the Serialized Key Initialization Module (SKIM) will be introduced (Figure 7). This module uses the available signals and the field multiplier which is used in the generation of X^{r_1}, in Figure 5. This module schedules the proper inputs to the field multiplier in each stage of the serial computation by means of some multiplexers. The output of these multiplexers are controlled by two new signals generated by the FSM, namely, s_0 and s_1 (Figure 6). The intermediate results, between two consecutive stages of the computation, are stored in internal

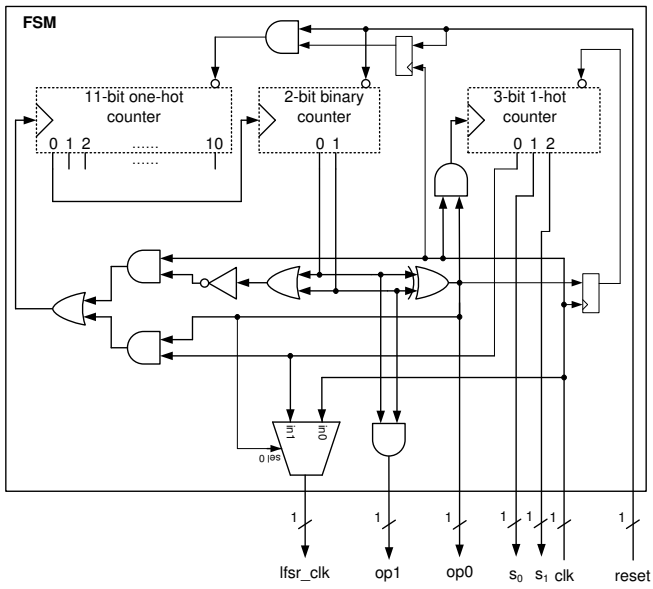

Figure 6: Modified FSM after adding the new 3-bit one-hot counter.

29-bit Registers of the SKIM module. In the following, we first introduce the FSM changes required for the support of the serialization process, followed by discussing the architecture and operation of the SKIM module and its integration to the WG transformation of the WG design in Figure 5.

4.3.1 Architecture and Operation of the Modified FSM

Here we describe the new architecture and operation of the FSM. The architecture, which is shown in Figure 6, generates the new set of control signals lfsr_clk, s_0, and s_1, which 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 bit and the rightmost bit, within the brackets, denote the lowest output bit and the highest output bit of the corresponding counter, respectively) by means of pulling down their reset inputs. When the reset signal is released, the 2-bit binary counter becomes ready, while the 11-bit one-hot counter's reset input stays pulled down for an extra clock cycle due to the 1-bit Register connected to the input of the AND gate which 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, before incrementing under the effect of the clk signal. 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, and hence 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. At this point, the clk signal starts triggering the clock input of the 3-bit one-hot counter; however, the counting will 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 cycles, 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, and 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 in order to start the running phase. When the running phase starts, with the 2-bit binary counter's state being $(1, 1)$, the 11-bit and

Table 2: 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

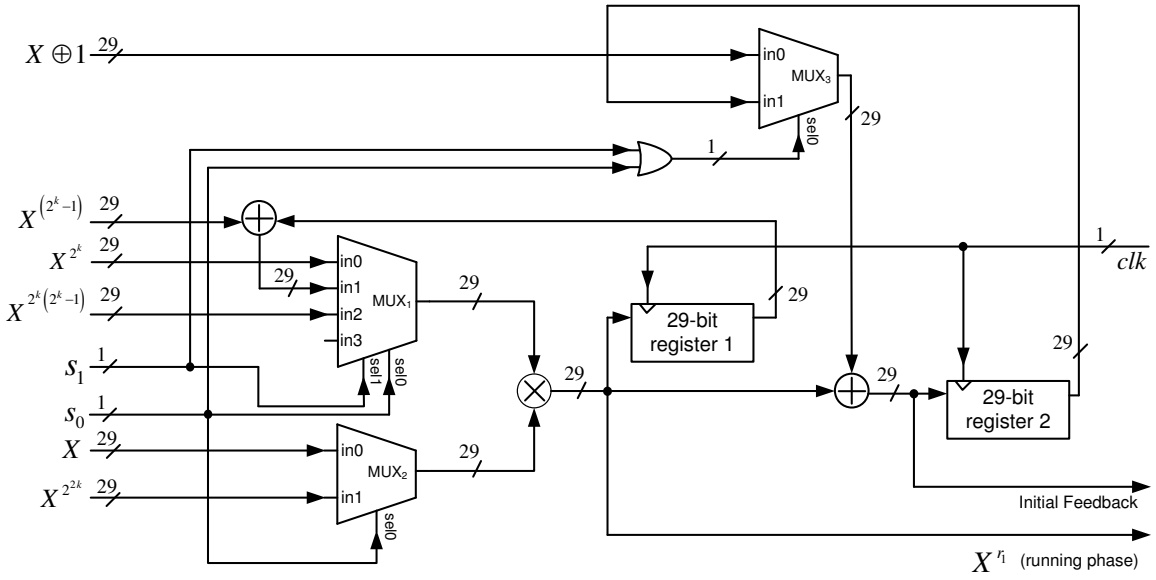

Figure 7: Block diagram of the SKIM module. The Initial Feedback signal is connected to the LFSR's input multiplexer as shown in Figure 1. X^{r_1} connectivity is shown in more details in Figure 8.

the 3-bit one-hot counters stop counting, as their clock inputs become idle at the low logic level.

Notice that during the key initialization phase, the lfsr_clk is driven by the first output of the 3-bit one-hot counter, which pulls high every three clock cycles and, hence, causing the LFSR to shift 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 2. 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 stage, the second, and the third stage of an extended key initialization round, respectively. During the running phase, (s_0, s_1) will generate $(0, 0)$. In the following we show how these patterns are used to accomplish the proper functionality in the key initialization phase as well as in the running phase.

4.3.2 Architecture and Operation of the Serialized Key Initialization Module

Here we introduce the SKIM module, which performs the serialized computation of the Initial Feedback signal over an extended key initialization round (three clock cycles).

Figure 7 is a block diagram describing the architecture of this module. During the key initialization phase, the two signals s_0 and s_1 in Figure 7 change values in each stage of the extended key initialization round as mentioned in the previous sub-subsection. These two signals control the outputs of the three

Table 3: 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 (3 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	$X^{r_1} \oplus X^{(2^k-1)}$	$X^{2^{2k}}$	$X^{r_1} \oplus X \oplus 1$	$X^{r_4} \oplus X^{r_2}$	$X^{r_4} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$
3	1	0	$X^{2^k(2^k-1)}$	X	$X^{r_4} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$	X^{r_3}	$X^{r_4} \oplus X^{r_3} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$

multiplexers MUX_1, MUX_2, and MUX_3 according to Table 3.

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 (see Figure 7).

During the first clock cycle, s_0 and s_1 are both at low logic levels, and 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 $GF\left(2^{29}\right)$ 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 the output of the $GF\left(2^{29}\right)$ adder, respectively, and 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 $GF\left(2^{29}\right)$ 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 $GF\left(2^{29}\right)$ adder outputs become $X^{r_3} = X^{(2^k+1)(2^k-1)}$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 ($X^{r_4} \oplus X^{r_3} \oplus X^{r_2} \oplus X^{r_1} \oplus X \oplus 1$) with the LFSR's Linear Feedback signal. At the arrival of the 67-th clock signal, the LFSR would have been clocked 22 times and the running phase starts.

Figure 8 presents the block diagram of the WG transformation of the proposed WG integrated with the SKIM module. Notice that, as we mentioned in 4.3.1, throughout the running phase, both s_0 and s_1 stay at logic level 0; therefore MUX_1 generates the signal X^{2^k}, MUX_2 generates the signal X, and MUX_3 generates the signal $X \oplus 1$. With these values of s_0, s_1, and the three multiplexers outputs, Figure 8 is fully configured to function exactly the same as the WG transformation in Figure 5, producing a new key-stream bit per each clock cycle in the running phase.

4.4 Space and Time Complexities

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

4.4.1 Space Complexity

Compared to the MOWG, in Figure 3, the hardware changes in the proposed WG have been applied to the FSM and the WG transformation. In what follows, we summarize the new hardware complexity

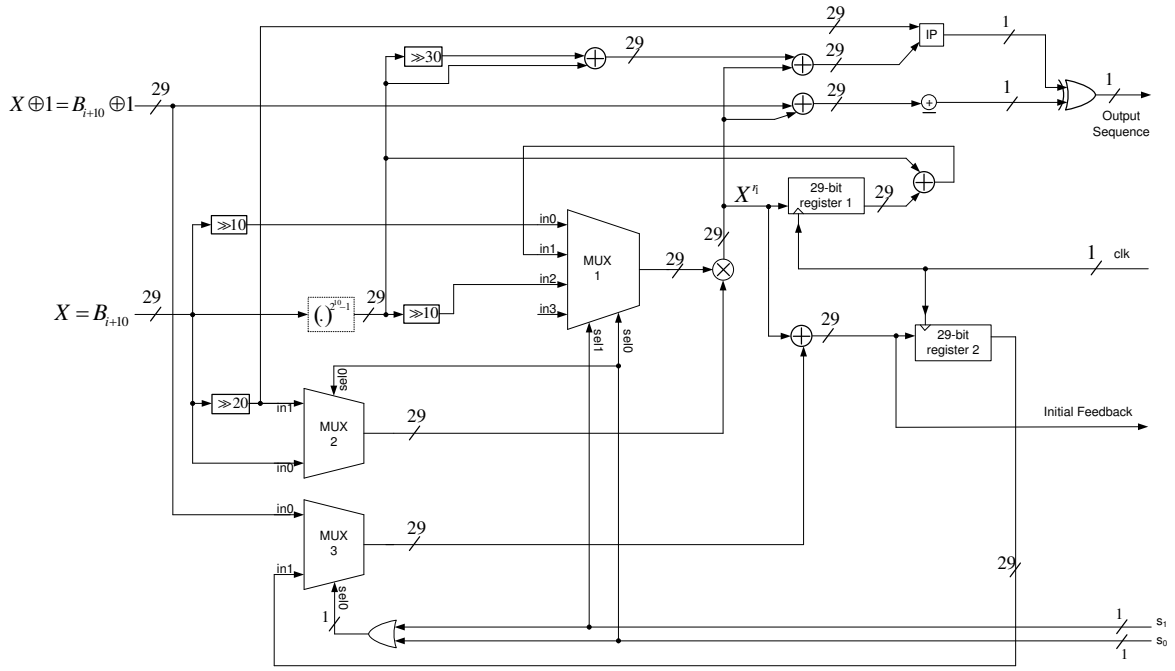

Figure 8: The proposed WG transformation after integration with the SKIM module. The block denoted by "IP" generates the inner product of the two 29-bit inputs (see Section 2), while $\underline{\oplus}$ adds the 29-bits at its input over $GF(2)$.

of these two components in the WG implementation, as a result of the modifications discussed in Subsection 4.3, followed by determining overall hardware cost of the design.

WG Transformation: Compared to the MOWG transformation in Figure 3, the design in Figure 8 has two less multipliers, while it has the following additional components

- A $GF\left(2^{29}\right)$ adder.
- Two 29-bit $GF(2)$ additions.
- Two 29-bit Registers.
- An XOR gate.
- An OR gate.
- One 4-to-1 29-bit multiplexer.
- Two 2-to-1 29-bit multiplexers.
- 2-input bit-wise AND operation, with each input of a 29-bit width.

A 29-bit $GF(2)$ adder consists of 28 XOR gates (in tree structure for maximum speed). A 2-to-1 29-bit multiplexer consists of 29 parallel 2-to-1 1-bit multiplexers. The bit-wise AND operator has 29 AND gates in parallel. Refer to Subsection 3.4.1 for details about the hardware of the other components listed above. By adding the hardware of the components in the list above to (10), and then subtracting

the hardware cost of two field multipliers, the total hardware cost of the WG transformation in Figure 8 can be written as

$$\begin{cases} N_A = 11 \times 29 + 5887 - 2 \times 841 = 4524, \\ N_X = 29 + 2 \times 28 + 1 + 8642 - 2 \times 1218 \\ \quad\; = 6292, \\ N_O = 1 + 5 \times 29 = 146, \\ N_I = 5 \times 29 = 145, \\ N_R = 2 \times 29 = 58. \end{cases} \tag{31}$$

FSM: The FSM depicted in Figure 6 requires the addition of two AND gates, two OR gates, a 2-to-1 1-bit multiplexer, 1-bit Register, and a 3-bit one-hot counter, as compared to Figure 4. Similar to 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. By adding the gates in the mentioned components to(13) and (14), the total hardware cost of the FSM in Figure 6 is as follows

$$\begin{cases} N_R = 1 + 3 + 14 = 18, \\ N_A = 2 + 2 + 3 = 7, \\ N_X = 1, \\ N_O = 2 + 1 = 3, \\ N_I = 1 + 1 = 2. \end{cases} \tag{32}$$

Proposition 7. *The overall space complexity of the proposed WG cipher is*

$$\begin{cases} N_O^{WG} = 233, \\ N_I^{WG} = 320, \\ N_R^{WG} = 395, \\ N_A^{WG} = 5546, \\ N_X^{WG} = 7714. \end{cases} \tag{33}$$

Where 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.

Proof. In addition to the FSM, the 4-to-1 29-bit multiplexer, the LFSR, and the WG transformation, the WG design contains two 29-bit bit-wise complement operations (inverter symbol (a) and inverter symbol (b) in Figure 3) and a $GF\left(2^{29}\right)$ adder (computing the bit-wise XOR of Initial Feedback signal and the Linear Feedback signal). Therefore, by adding the corresponding number of gates in the $GF\left(2^{29}\right)$ adder and in inverter symbols (a) and (b) to the values in (12), (11), (32), and (31) we obtain (33). □

4.4.2 Time Complexity

Here we derive the formulation for the critical path of the proposed WG design. Notice that the LFSR delay in the WG is not a candidate for the critical path, since it still has significantly less multipliers contributing to its delay, compared to the WG transformation. In what follows, the formulation of the longest path during the key initialization phase is presented. After this we prove that the running phase is the longest path of the cipher.

Proposition 8. *The longest path during the key initialization phase of the WG is given by*

$$T_{KIPh} = 16T_X + \frac{7}{2}T_A + \frac{1}{2}T_R + T_O + T_I. \tag{34}$$

Proof. Let $T_{clock} \geq T_{KIPh}$ 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

$$T_{clock} \geq \Big(7T_X + 3T_A + T_R + 2T_O + 2T_I\Big) \tag{35}$$

$$T_{clock} \geq \frac{1}{2}\Big(32T_X + 7T_A + T_R + 2T_O + 2T_I\Big) \tag{36}$$

$$T_{clock} \geq \frac{1}{3}\Big(32T_X + 9T_A + T_R + 4T_O + 4T_I\Big) \tag{37}$$

where the right hand side in (35), (36), and (37) is simply the delay during the first stage, the second stage, and the third stage of the extended key initialization round, respectively. By subtracting the right hand sides of each pair from (35), (36), and (37) it is easy to show that the right hand side of (36) is the largest, which completes the proof. □

Proposition 9. *The longest delay through the WG during the running phase given by*

$$T_{RunPh} = 32T_X + 5T_A + T_R, \tag{38}$$

is the maximum delay of the cipher.

Proof. (38) can be easily obtained by adding the delays of the components on the longest path of the running phase of the WG. From (34) and (38) we have

$$T_{RunPh} - T_{KIPh} = 15T_X + \frac{3}{2}T_A + \frac{1}{2}T_R - T_O - T_I,$$

which is obviously greater than zero. This establishes the assertion. □

5 Results and Comparisons

This section presents a comparison between the proposed designs of the MOWG$(29, 11, 17)$ cipher and the WG$(29, 11)$ cipher with the previous implementations in [17] and [22], respectively. Also, speed, area, and power consumption results are conducted based on both FPGA and ASIC implementations.

5.1 Design Parameters of the MOWG/WG Ciphers

We use the same parameters of the WG$(29, 11)$ in [22] and MOWG$(29, 11, 17)$ in [17] as follows:

- The defining polynomial, $g(X)$, of $GF\left(2^{29}\right)$ is given in (4), where α is a root of $g(X)$.
- The field elements are represented in type-II ONB with the generator of the basis $\beta = \alpha^{464730077}$.
- The characteristic polynomial of the LFSR, $C(X)$, is of degree 11 and is given in (3).
- $k = \left\lceil \frac{29}{3} \right\rceil = 10$.

Table 4: Results obtained for area, speed, and power consumption from the ASIC implementations. GE denotes Gate Equivalence in terms of number of NAND gates (estimated area of one NAND gate is $2.08\,(\mu m)^2$). Relative GE, relative speed, and relative power are all with respect to [22] for the WG designs.

CMOS 65nm						
Implementation	GE		Speed		Dynamic Power	
	Total	Relative	MHz	Relative	mW	Relative
WG [22]	33180	100 %	144	100 %	7.28	100 %
WG (Figure 5)	19892	60 %	169	117 %	4.45	61 %
MOWG (Figure 3)	26261	-	151	-	5.89	-

Table 5: Results for area, speed, and power consumption analysis of FPGA implementations. Relative LUTs, relative speed, and relative power are all with respect to [22] for the WG designs.

Virtex4 (xc4vfx12sf363-10)						
Implementation	LUTs		Speed		Total Power	
	Total	Relative	MHz	Relative	mW	Relative
WG [22]	6449	100 %	30	100 %	380	100 %
WG (Figure 5)	4044	63 %	34	113 %	187	49 %
MOWG (Figure 3)	5512	-	35	-	342	-

5.2 Results from FPGA and ASIC Implementations

Our ASIC implementations provide speed and area results for the 65nm CMOS technology with medium effort for optimizations using Synopsys Design Compiler [1]. The results are based on Design Compiler's estimate of area and clock speed prior to place-and-route. The power consumption readings were conducted under 140 MHz frequency for all the designs. Our FPGA area and speed results are for Xilinx Virtex4 series FPGA device xc4vfx12sf363-10 using Xilinx Synthesis Tool (XST) for synthesis [2]. All results are for post place-and-route and the power consumption results were recorded for a frequency of 29 MHz for all the designs.

For MOWG$(29, 11, 17)$, the authors of [17] conducted implementations using 90nm CMOS technology for ASIC and using Altera Stratix II series device EP2S15F484C for FPGA. Also, the results in [17] are not reported for the original design without pipelining. The results are reported only for the pipelined-with-reuse version. The reader is referred to [17] for these results, here we discuss the differences of [17] against the proposed design in this paper, in general.

Tables 4 and 5 presents the area, speed, and power consumption results for ASIC and FPGA implementations, respectively, while Table 6 presents the required number of clock cycles in the key initialization phase for the different designs.

It is clear that the proposed WG design significantly reduces area and power consumption while it improves the speed, compared to [22]. Similar to the WG design presented in [22], the proposed WG generates a stream bit per each clock cycle during the running phase; however, it requires three times the number of clock cycles in [22] for conducting its key initialization phase.

The proposed MOWG consumes the same number of clock cycles as in [17] to perform the key initialization process. The proposed MOWG provides the same speed (slightly improved by removing the delays of two inverters) and throughput, as in the corresponding design in [17], however, at a lower

Table 6: Comparison of number of clock cycles in key initialization phase

Implementation	# of clocks in key initialization phase
WG [22]	22
MOWG (Figure 3)	
WG (Figure 5)	66

area complexity due to the one extra multiplier in [17].

6 Conclusion

We have proposed two new designs for the MOWG$(29, 11, 17)$ and the WG$(29, 11)$ ciphers. As compared to the MOWG presented in [17], the proposed MOWG reduces the number of finite field multipliers in the MOWG transformation by one through signal reuse, while it increases the speed by eliminating two inverters delay from the critical path. This is accomplished by reconstructing the key/IV loading algorithm and the feedback polynomial of the LFSR. The proposed WG is an optimization of the proposed MOWG with trace (WG version) which is obtained through using properties of the trace function for type-II ONB, accompanied with serialization of the computation of the Initial Feedback signal during key initialization phase.

We have implemented the proposed designs on ASIC and FPGA. The ASIC implementations show that the proposed WG implementation achieves better results compared to [22] 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 17%. Similarly, the FPGA implementations show that the proposed WG achieves better results for area, speed, and power consumption compared to [22], where 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 limited resources platforms where area and power consumption are of critical importance and the guaranteed randomness properties are required.

Acknowledgments

The authors would like to thank Canadian Microelectronics Corporation (CMC) Microsystems for providing the required infrastructure and CAD tools that have been used in this work.

References

[1] Synopsys. `http://www.synopsys.com/`.

[2] Xilinx. `http://www.xilinx.com/`.

[3] eSTREAM - The ECRYPT Stream Cipher Project, 2005.

[4] Adopted Bluetooth Core Specifications, Core Version 4.0. Bluetooth Special Interest Group, June 2010. `https://www.bluetooth.org/`.

[5] B. Ansari and M. Hasan. High-Performance Architecture of Elliptic Curve Scalar Multiplication. *IEEE Trans. Comput.*, 57(11):1443 –1453, nov. 2008.

[6] F. Armknecht. On the Existence of Low-degree Equations for Algebraic Attacks. Cryptology ePrint Archive, Report 2004/185, 2004. `http://eprint.iacr.org/`.

[7] L. Chen and G. Gong. *Communication System Security*. Chapman and Hall - CRC Press, 2012.

[8] N. Courtois. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In *Proc. Advances in Cryptology - CRYPTO'2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, 2003.

[9] N. Courtois. Algebraic Attacks on Combiners with Memory and Several Outputs. In C.-s. Park and S. Chee, editors, *Information Security and Cryptology - ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 68–72. Springer-Verlag, 2005.

[10] J. Dillon and H. Dobbertin. New Cyclic Difference Sets with Singer Parameters. *Finite Fields and Their Applications*, 10(3):342 – 389, 2004.

[11] S. Gao and H. W. Lenstra. Optimal Normal Bases. *Designs, Codes, and Cryptography*, 2:315–323, 1992.

[12] S. W. Golomb and G. Gong. *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*. Cambridge University Press, New York, NY, USA, 2004.

[13] G. Gong and Y. Nawaz. The WG Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/033, 2005. `http://www.ecrypt.eu.org/stream/wgp2.html`.

[14] G. Gong and A. Youssef. Cryptographic Properties of the Welch-Gong Transformation Sequence Generators. *IEEE Trans. Inf. Theory*, 48(11):2837 – 2846, Nov. 2002.

[15] D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int'l J. Information Security*, 1(1):36–63, 2001.

[16] E. Krengel. Fast WG Stream Cipher. In *IEEE Region 8 Int. Conf. on Computational Technologies in Elect. and Electron. Eng., 2008. SIBIRCON 2008.*, pages 31 –35, Jul. 2008.

[17] C. Lam, M. Aagaard, and G. Gong. Hardware Implementations of Multi-output Welch-Gong Ciphers. Technical Report CACR 2011-01, University of Waterloo, Waterloo, ON, Canada, 2009.

[18] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge University Press, New York, NY, USA, 1986.

[19] Y. Luo, Q. Chai, G. Gong, and X. Lai. A Lightweight Stream Cipher WG-7 for RFID Encryption and Authentication. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1 –6, Dec. 2010.

[20] W. Meier, E. Pasalic, and C. Carlet. Algebraic Attacks and Decomposition of Boolean Functions. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer-Verlag, 2004.

[21] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson. Optimal Normal Bases in $GF(p^n)$. *Discrete Applied Math.*, 22(2):149–161, Feb. 1989.

[22] Y. Nawaz and G. Gong. WG: A Family of Stream Ciphers with Designed Randomness Properties. *Inf. Sci.*, 178(7):1903 – 1916, 2008.

[23] J.-S. No, S. Golomb, G. Gong, H.-K. Lee, and P. Gaal. Binary Pseudorandom Sequences of Period $2^n - 1$ with Ideal Autocorrelation. *IEEE Trans. Inf. Theory*, 44(2):814 –817, Mar. 1998.

[24] A. Reyhani-Masoleh and M. Hasan. A New Construction of Massey-Omura Parallel Multiplier Over $GF(2^m)$. *IEEE Trans. Comput.*, 51(5):511 –520, May. 2002.

[25] S. Ronjom, G. Gong, and T. Helleseth. On Attacks on Filtering Generators Using Linear Subspace Structures. In S. Golomb, G. Gong, T. Helleseth, and H.-Y. Song, editors, *Sequences, Subsequences, and Consequences*, volume 4893 of *Lecture Notes in Computer Science*, pages 204–217. Springer-Verlag, 2007.

[26] S. Sen Gupta, A. Chattopadhyay, and A. Khalid. HiPAcc-LTE: An Integrated High Performance Accelerator for 3GPP LTE Stream Ciphers. In *Proceedings of the 12th international conference on Cryptology in India*, INDOCRYPT'11, pages 196–215, Berlin, Heidelberg, 2011. Springer-Verlag.

[27] S. Sen Gupta, A. Chattopadhyay, K. Sinha, S. Maitra, and B. Sinha. High Performance Hardware Implementation for RC4 Stream Cipher. *IEEE Trans. Comput.*, to appear.

[28] H. Wu and B. Preneel. Resynchronization Attacks on WG and LEX. In M. Robshaw, editor, *Fast Software Encryption*, volume 4047 of *Lecture Notes in Computer Science*, pages 422–432. Springer-Verlag, 2006.