Hardware Constructions for Error Detection in WG-29 Stream Cipher Benchmarked on FPGA

Jasmin Kaur, *Student Member, IEEE*, Alvaro Cintas Canto, *Member, IEEE*, Mehran Mozaffari Kermani, *Senior Member, IEEE*, and Reza Azarderakhsh, *Member, IEEE*

***Abstract*—WG-29 is a Welch-Gong (WG) stream cipher, implemented in $GF(2^{29})$ and an 11-stage LFSR, whose polynomial-basis (PB)-based architecture is utilized in diverse applications. This work, for the first time, presents low-cost normal signature, interleaved signature, and Hamming code-based error detection mechanisms for the hardware implementations of PB-based WG-29 stream cipher. The presented schemes are benchmarked on field-programmable gate array (FPGA) hardware platform using Kintex-7 and Spartan-7 FPGA families for area (< 40%), power (< 12%), and delay (< 10%) overheads. Using a faulty module to inject stuck-at single bit and multiple bit upsets, the error coverage for these presented schemes is evaluated via simulations performed in Xilinx Vivado for 80 000 faults and shown to be over 99.99%. The overhead and error simulation results for the presented schemes show that they provide high-error coverage with acceptable overheads to make hardware constructions of WG-29 more reliable. Other WG ciphers that have similar underlying primitives can also benefit from the presented work, with slight modifications, for secure hardware implementations.**

***Index Terms*—Error detection, field-programmable gate array (FPGA), linear feedback shift register (LFSR), polynomial basis (PB) multiplier, Welch-Gong (WG) cipher.**

I. Introduction

Stream ciphers are symmetric key cryptosystems that perform bit-by-bit encryption/decryption to provide confidentiality and integrity in Internet of Things (IoTs) devices, RFID tags, Bluetooth devices, network protocols, and long term evolution (LTE) security suite for secure communication. Welch-Gong (WG) ciphers (Fig. 1) are stream ciphers based on an l-stage linear feedback shift register (LFSR) and a WG-transformation function, both defined in the same finite field $GF(2^m)$, to generate a pseudo-random keystream [1], [2], [3], [4], [5]. The WG-29 [6], [7], [8] is a hardware-oriented WG stream cipher defined in the $GF(2^{29})$ with an 11-stage LFSR. Manufacturing or transient faults in the hardware implementations of WG-29 can be utilized for fault analysis [9], [10] that can compromise its security and reliability. As studied in a number of previous works, e.g., [11], [12], [13], [14], and [15], error detection is often used in cryptographic applications to enhance their reliability and security in hardware implementations. This article, for the first time as per authors' knowledge, proposes error detection mechanisms for the polynomial basis (PB)-based implementation of WG-29 stream cipher [7], [8]. Such schemes have also benefited the lightweight streamcipher WAGE, a successor of WG-29. We note that this article does not focus on error correction codes; however, this work also proposes Hamming codes scheme that can be utilized to correct errors up to a two bits. The presented work, with slight modifications, can also be used for secure hardware implementations of other WG ciphers that have similar underlying primitives.

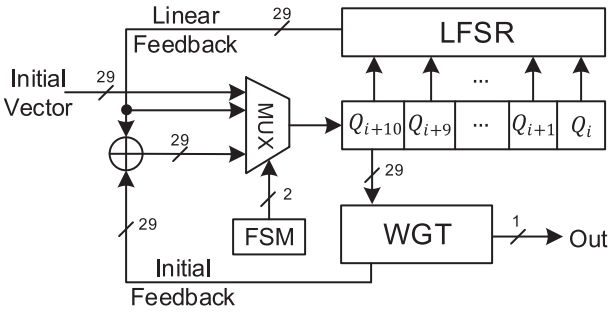

Fig. 1. General design of a WG-29 cipher [8].

We present the formulations for the low-cost normal and interleaved signatures schemes for the squaring matrices, the trace, and the trace of multiplication of two PB elements of WG-29 in hardware constructions of PB-based WG-29 to detect single/multiple bit faults. We also present the $(7, 4)$ Hamming codes for the complex PB multiplier of WG-29 [14], [15]. For the sake of brevity, the error coverage, through Vivado simulations, has been performed only for the stuck-at faults, but we expect similar error coverage for transient faults as well. The proposed architectures are benchmarked on field-programmable gate array (FPGA) hardware platform using Xilinx Kintex-7 and Spartan-7 FPGA families, for area, delay, and power overheads using Xilinx Vivado.

This article is organized as follows: in Section II, the functionality of WG-29 is described. In Section III, the proposed error detection schemes for the S-module, the trace functions and the PB multiplier are presented. In Section IV, the FPGA benchmarks followed by the error coverage assessments are given. Section V concludes the presented work.

II. Preliminaries

Detailed specification of WG-29 (Fig. 1), its design parameters, and proof of the equations given below are described in [8]. Briefly, WG-29 is a bit-oriented sequence generator, where the $m = 29$ bits. The WGT is applied to the leftmost cell of the primitive LFSR of degree $l = 11$ over $GF(2^{29})$, which produces m-sequences of period $2^{m \cdot l} - 1$, i.e., $2^{319} - 1$ periods for $m = 29$ [1]. For WG-29, the field

Manuscript received 15 May 2023; accepted 16 November 2023. Date of publication 30 November 2023; date of current version 21 March 2024. This work was supported by the U.S. Department of Commerce, National Institute of Standards and Technology (NIST) through the U.S. Federal Agency under award 60NANB20D013. This article was recommended by Associate Editor J. L. Dworak. *(Corresponding author: Mehran Mozaffari Kermani.)*

Jasmin Kaur and Mehran Mozaffari Kermani are with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: jasmink1@usf.edu; mehran2@usf.edu).

Alvaro Cintas Canto is with the School of Technology and Innovation, Marymount University, Arlington, VA 22207 USA (e-mail: acintas@marymount.edu).

Reza Azarderakhsh is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: razarderakhsh@fau.edu).

Digital Object Identifier 10.1109/TCAD.2023.3338108

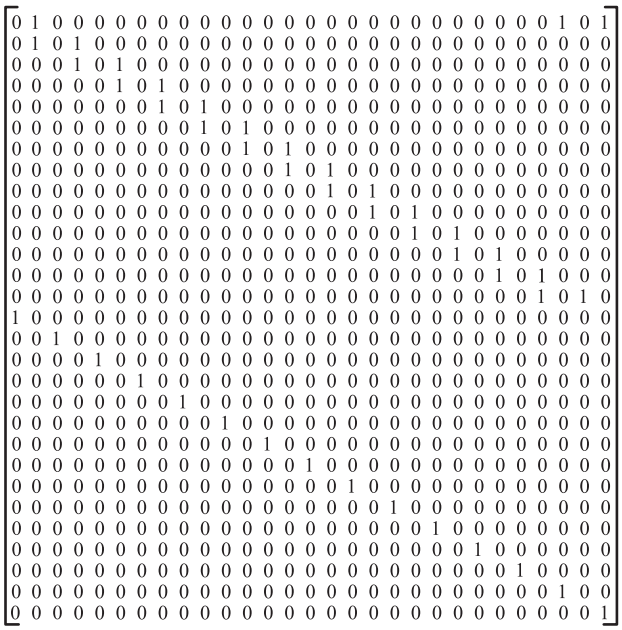

Fig. 2. S matrix for WG-29 [8].

polynomial $f(x) = x^{29} + x^2 + 1$ is used for lower-space complexity of squaring and multiplication operations. The LFSR based on primitive polynomial $P^{11} \oplus P^6 \oplus P^2 \oplus P \oplus x$ is used for WG-29, where x is the root of $f(x)$ and $x \in GF(2^{29})$. The squaring matrix S (Fig. 2) is a binary $m \times m$ matrix used to square a field element $U \in GF(2^{29})$ with respect to PB. Here, and throughout this article, the symbol $\oplus$ corresponds to the XOR operation. All the formulations follow the rules of modulo-2 arithmetic for finite fields.

From [8], the WG-29 permutation is given as $\text{WGP}(29) = 1 \oplus T \oplus T^{2^{10}+1} + T^{2^{20}+2^{10}+1} \oplus T^{2^{20}-2^{10}+1} \oplus T^{2^{20}+2^{10}-1}$, where $T = 1 \oplus L_{i+10}$ is the output of the LFSR with its least significant bit inverted. The WGT is obtained as $Tr(\text{WGP}(29))$. Thus, substituting $T^{2^{20}-2^{10}+1}$ with $(T^{2^{20}-2^{10}+1})^{2^{20}}$ in the equation above, WGT(29) in [8] becomes $\text{WGT}(29) = Tr[1 \oplus T \oplus T(T^{2^2})^{2^5}] + Tr[((T^{2^5})^{2^2})^{2^{10}}(T(T^{2^5})^{2^5} \oplus T^{2^{10}-1} \oplus (T^{2^{10}-1})^{2^{30}})]$, where $T^{2^{10}+1} = T(T^{2^5})^{2^5}$, $T^{2^{20}} = ((T^{2^5})^{2^5})^{2^{10}}$, and $T = [((T^{2^5+1})^{2+1})(Y^{2^5+1})^{2^4}][(T^{2^5+1})^{2+1}]^{2^2}$.

For $PB = \{1, x, \ldots, x^{27}, x^{28}\}$ of $GF(2^{29})$ over $GF(2)$ defined by $f(x)$, for an element $K = \sum_{i=0}^{28} x^i k_i$, the $Tr(K)$ [8] is derived as $Tr(K) = k_0\tau_0 + k_{27}\tau_{27}$.

For multiplication of two field elements $K = \sum_{i=0}^{28} x^i k_i$ and $O = \sum_{i=0}^{28} x^i o_i$, the $Tr(KO)$ [8] is calculated as $Tr(KO) = (k_0 + k_{27})o_0 + \sum_{j=1}^{25}(k_{27-j} + k_{29-j})o_j + (k_1 + k_{26})o_{28} + \sum_{j=26}^{27}(k_{27-j} + k_{29-j} + k_{54-j})o_j$.

III. Proposed Error Detection Architectures for WG-29

This section presents low-cost formulations for the parity-based normal signature, interleaved signature, and $(7, 4)$ Hamming codes for error detection in hardware implementations of WG-29. The $(7, 4)$ Hamming code-based error detection scheme is utilized for the complex $GF(2^{29})$ PB multiplier for higher-error coverage. To determine if a bit fault has occurred in a module, the parity of the output (actual parity) is compared with the parity of the input (predicted parity) to trigger an error flag. For Hamming codes, the parity-check bits of the input and output vectors are compared for fault detection.

TABLE I
Formulations for the Normal ($\hat{\rho}_0$) and the Interleaved Signature ($\hat{\rho}_1$, $\hat{\rho}_2$) for the Squaring Matrices of PB-Based WG-29 Architecture (Section II)

Squaring Matrix	Signatures
S, S^{30}	$\hat{\rho}_0 = \{u_{28} \oplus u_{14} \oplus u_{13} \oplus u_{12} \oplus u_{11} \oplus u_{10} \oplus u_9 \oplus u_8 \oplus u_7 \oplus u_6 \oplus u_5 \oplus u_4 \oplus u_3 \oplus u_2 \oplus u_1 \oplus u_0\}$ $\hat{\rho}_1 = \{u_{14} \oplus u_{13} \oplus u_{12} \oplus u_{11} \oplus u_{10} \oplus u_9 \oplus u_8 \oplus u_7 \oplus u_6 \oplus u_5 \oplus u_4 \oplus u_3 \oplus u_2 \oplus u_1 \oplus u_0\}$ $\hat{\rho}_2 = \{u_{28}\}$
S^2	$\hat{\rho}_0 = \{u_{21} \oplus u_{14} \oplus u_7 \oplus u_6 \oplus u_5 \oplus u_4 \oplus u_3 \oplus u_2 \oplus u_1 \oplus u_0\}$ $\hat{\rho}_1 = \{u_{21} \oplus u_7 \oplus u_6 \oplus u_5 \oplus u_4 \oplus u_3 \oplus u_2 \oplus u_1 \oplus u_0\}$ $\hat{\rho}_2 = \{u_{14}\}$
S^4	$\hat{\rho}_0 = \{u_{27} \oplus u_{26} \oplus u_{18} \oplus u_{17} \oplus u_{16} \oplus u_{12} \oplus u_9 \oplus u_1 \oplus u_0\}$ $\hat{\rho}_1 = \{u_{27} \oplus u_{26} \oplus u_{16} \oplus u_{12} \oplus u_9 \oplus u_1 \oplus u_0\}$ $\hat{\rho}_2 = \{u_{18} \oplus u_{17}\}$
S^5	$\hat{\rho}_0 = \{u_{27} \oplus u_{23} \oplus u_{22} \oplus u_{19} \oplus u_{18} \oplus u_{15} \oplus u_{13} \oplus u_9 \oplus u_8 \oplus u_6 \oplus u_0\}$ $\hat{\rho}_1 = \{u_{27} \oplus u_{19} \oplus u_{18} \oplus u_{13} \oplus u_8 \oplus u_6 \oplus u_0\}$ $\hat{\rho}_2 = \{u_{23} \oplus u_{22} \oplus u_9\}$
S^{10}	$\hat{\rho}_0 = \{u_{28} \oplus u_{24} \oplus u_{21} \oplus u_{20} \oplus u_{18} \oplus u_{17} \oplus u_{15} \oplus u_{13} \oplus u_9 \oplus u_8 \oplus u_3 \oplus u_2 \oplus u_0\}$ $\hat{\rho}_1 = \{u_{28} \oplus u_{26} \oplus u_{19} \oplus u_{16} \oplus u_{15} \oplus u_{14} \oplus u_{13} \oplus u_{10} \oplus u_9 \oplus u_8 \oplus u_7 \oplus u_6 \oplus u_5 \oplus u_2 \oplus u_0\}$ $\hat{\rho}_2 = \{u_{26} \oplus u_{24} \oplus u_{21} \oplus u_{10} \oplus u_{19} \oplus u_{18} \oplus u_{17} \oplus u_{16} \oplus u_{14} \oplus u_{10} \oplus u_7 \oplus u_6 \oplus u_5 \oplus u_3\}$

A. Proposed Signature-Based Schemes for S

The S of WG-29 is a squaring matrix (Fig. 2) used to perform the squaring operations of elements in $GF(2^{29})$ [8]. As described in Section II, S is a binary $m \times m$ matrix whose elements are either "0" or "1." The *PB*-based implementation of WG-29 uses six chains of S, namely, S, S^2, S^4, S^5, S^{10}, and S^{30}. The square of an element $U^2 \in GF(2^{29})$ is computed as $Z = US = \sum_{i=0}^{28}\sum_{j=0}^{28} u_j S_{j,i}$, where $Z = \{z_{28}, z_{27}, \ldots, z_0\}$ and $U = \{u_{28}, u_{27}, \ldots, u_0\}$, respectively. Additionally, the square of exponentiation values $W = U^{2^e}$ is calculated as $W = US^e = \sum_{i=0}^{28}\sum_{j=0}^{28} u_j S^e_{j,i}$, where $W = \{w_{28}, w_{27}, \ldots, w_0\}$. Here, the matrix S^e represents the chains of squaring matrix S for $e \in \{1, 2, 4, 5, 10, 30\}$. The formulations for the normal and interleaved signatures of all the six aforementioned squaring matrices used are given in Table I. It is noted that $S^{30} = S$, and hence their formulations are the same.

Remark 1: Only the nonzero terms of the product Z are considered in the following equations.

1) *Signature:* The normal parity $\hat{\rho}_0$ of the S matrix for the input vector $\{u_{28}, u_{27}, \ldots, u_0\}$ is calculated as: $\hat{\rho}_0 = \sum_{i=0}^{28}\sum_{j=0}^{28} u_j S_{j,i}$.
2) *Interleaved Signature:* Similarly, the interleaved parity bits $(\hat{\rho}_1, \hat{\rho}_2)$ of the S matrix for the input $\{u_{28}, u_{27}, \ldots, u_0\}$ are: $\hat{\rho}_1 = \sum_{i=0}^{14}\sum_{j=0}^{28} u_j S_{j,2i}$ and $\hat{\rho}_2 = \sum_{i=1}^{14}\sum_{j=0}^{28} u_j S_{j,2i-1}$.

B. Proposed Signature-Based Schemes for the Trace Functions

The trace function in WG is used to map the trace of an element from $GF(2^{29}) \rightarrow GF(2)$. WG-29 uses two different trace vectors:

1) to get the trace of a single PB element and 2) to get the trace of the produce of two PB elements, respectively.

Remark 2: Since the output of both the trace functions is a single bit value and depends upon only nonzero terms, the normal and interleaved signatures of these functions can be directly computed from the trace equations given in Section II, and then compared to the respective trace function's output for error detection.

1) Trace Vector for Single PB Element: The trace $Tr(K)$ (Section II) of an element K of $GF(2^{29})$ with a row vector k is calculated as $Tr(K) = k\tau^T = \sum_{i=0}^{m-1} k_i \cdot \tau_i$, where $\tau = \{\tau_0, \ldots, \tau_{m-1}\}$ is a constant and unique vector such that $\tau^i = Tr(x^i)$. For the irreducible polynomial $f(x)$, the only two nonzero entries of τ are $\{\tau_0, \tau_{27}\}$. Therefore, the only terms considered for the trace value are k_0 and k_{27}.

a) Signature: For an input vector $K = \{k_{28}, k_{27}, \ldots, k_0\}$, the normal parity $\hat{\rho_3}$ is calculated as: $\hat{\rho_3} = \sum_{i=0}^{28} k_i \tau_i = \{k_0\tau_0 \oplus k_{27}\tau_{27}\}$.

b) Interleaved signature: For the input vector $K = \{k_0, k_1, \ldots, k_{28}\}$ and output $k\tau^T$, the 2-bit parity $(\hat{\rho_4}, \hat{\rho_5})$ is calculated as: $\hat{\rho_4} = \sum_{i=0}^{14} k_{2i}\tau_{2i} = \{k_0\tau_0 \oplus k_2\tau_2 \oplus \cdots \oplus k_{28}\tau_{28}\} = k_0\tau_0$ and $\hat{\rho_5} = \sum_{i=0}^{14} k_{2i-1}\tau_{2i-1} = \{k_1\tau_1 \oplus k_3\tau_3 \oplus \cdots \oplus k_{27}\tau_{27}\} = k_{27}\tau_{27}$.

2) Trace of the Multiplication of Two PB Elements: Detailed explanation of the calculation of the trace of multiplication of two PB elements $K, O \in GF(2^{29})$ is shown in [8]. The normal and interleaved signatures are derived directly from $Tr(K \cdot O)$.

a) Signature: The normal parity $\hat{\rho_6}$ is computed as the modulo-2 addition of all the terms of the $Tr(KO)$: $\hat{\rho_6} = (k_0 \oplus k_{27})o_0 \oplus \sum_{j=1}^{25}(k_{27-j} \oplus k_{29-j})o_j \oplus (k_1 \oplus k_{26})o_{28} \oplus \sum_{j=26}^{27}(k_{27-j} \oplus k_{29-j} \oplus k_{54-j})o_j$.

b) Interleaved signature: The interleaved parity $(\hat{\rho_7}, \hat{\rho_8})$ of $Tr(KO)$: $\hat{\rho_7} = \{(k_0 \oplus k_{27})o_0 \oplus (k_{25} \oplus k_{27})o_2 \oplus \cdots \oplus (k_3 \oplus k_5)o_{24} \oplus (k_1 \oplus k_{26})o_{28} \oplus (k_1 \oplus k_3 \oplus k_{28})o_{26}\}$ and $\hat{\rho_8} = \{(k_{26} \oplus k_{28})o_1 \oplus (k_{24} \oplus k_{26})o_3 \oplus \cdots \oplus (k_2 \oplus k_4)o_{25} \oplus (k_0 \oplus k_2 \oplus k_{27})o_{27}\}$.

Following Remark 2, the $(\hat{\rho_7} \oplus \hat{\rho_8})$ is compared with the output of the $Tr(K)$ function (Section II) for the interleaved signature scheme.

C. Proposed Hamming Code-Based Scheme for the PB Multiplier

For the hardware implementation of WG-29 in this article, the PB multiplier from [14] is implemented, which uses *sum*, *pass-thru*, and *alpha* modules to multiply two PB elements in $GF(2^m)$. The multiplication $C = (A \cdot B) \bmod f(x)$ of $GF(2^m)$ two elements A and B in PB is performed as $C = \sum_{i=0}^{m-1} b_i \cdot ((Ax^i) \bmod f(x)) = \sum_{i=0}^{m-1} b_i \cdot M^i$ where, $b_i \in B$, $M^i = x \cdot M^{i-1} \bmod f(x)$ for $1 \le i \le m-1$, and $M^0 = A$ [14]. The *sum* module performs the finite field addition of two PB elements $A, B \in GF(2^m)$ as $\sum_{i=0}^{m-1}(b_i + a_i)x^i \bmod f(x)$; the *pass-thru* module multiplies an element $M^i \in GF(2^m)$ element with a $GF(2)$ element $b_i \in GF(2)$ as $b_i \cdot M^i \bmod f(x)$, where the output is M^i if $b_i = 1$, else 0 if $b_i = 0$ for $0 \le i \le m-1$; the alpha module multiplies a $GF(2^m)$ element A with the root x as $A(x) \cdot x = a_{m-1}x^m + \cdots + a_0 x \bmod f(x)$, where $x^m = f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \cdots + f_0 \bmod f(x)$ [14]. The formulated $(7, 4)$ Hamming codes for the 29-bit PB multiplier of WG-29 are given below.

Consider $\gamma(x)$ as a 29-bit input/output vector of the *sum*, *pass-thru*, or *alpha* modules detailed above. This vector is first padded with zeros to make it 32 bits, following a split into 4-bit blocks (1). Each of these 4-bit block is then multiplied with the last three columns of generator matrix G (2) to get $8 \cdot 3 = 24$ encoded parity bits

$$\begin{aligned}
\gamma_0' &= \gamma_0 + \gamma_1 x + \gamma_2 x^2 + \gamma_3 x^3 \bmod f(x)\\
\gamma_1' &= \gamma_4 x^4 + \gamma_5 x^5 + \gamma_6 x^6 + \gamma_7 x^7 \bmod f(x)\\
&\vdots\\
\gamma_7' &= \gamma_{24} x^{24} + \gamma_{25} x^{25} + \gamma_{26} x^{26} + \gamma_{27} x^{27} \bmod f(x)\\
\gamma_8' &= \gamma_{28} x^{28} + \gamma_{29} x^{29} + \gamma_{30} x^{30} + \gamma_{31} x^{31} \bmod f(x) \quad (1)
\end{aligned}$$

$$G = \begin{Bmatrix} 1\,0\,0\,0\,1\,1\,0\\ 0\,1\,0\,0\,1\,0\,1\\ 0\,0\,1\,0\,0\,1\,1\\ 0\,0\,0\,1\,1\,1\,1 \end{Bmatrix} \quad (2)$$

$$\begin{aligned}
\gamma_0' \cdot G \rightarrow \rho_0 &= \gamma_3 x^3 \oplus \gamma_1 x^1 \oplus \gamma_0\\
\rho_1 &= \gamma_3 x^3 \oplus \gamma_2 x^2 \oplus \gamma_0\\
\rho_2 &= \gamma_3 x^3 \oplus \gamma_2 x^2 \oplus \gamma_1 x\\
\gamma_1' \cdot G \rightarrow \rho_3 &= \gamma_7 x^7 \oplus \gamma_5 x^5 \oplus \gamma_4 x^4\\
\rho_4 &= \gamma_7 x^7 \oplus \gamma_6 x^6 \oplus \gamma_4 x^4\\
\rho_5 &= \gamma_7 x^7 \oplus \gamma_6 x^6 \oplus \gamma_5 x^5\\
\gamma_2' \cdot G \rightarrow \rho_6 &= \gamma_{11} x^{11} \oplus \gamma_9 x^9 \oplus \gamma_8 x^8\\
\rho_7 &= \gamma_{11} x^{11} \oplus \gamma_{10} x^{10} \oplus \gamma_8 x^8\\
\rho_8 &= \gamma_{11} x^{11} \oplus \gamma_{10} x^{10} \oplus \gamma_9 x^9\\
\gamma_3' \cdot G \rightarrow \rho_9 &= \gamma_{15} x^{15} \oplus \gamma_{13} x^{13} \oplus \gamma_{12} x^{12}\\
\rho_{10} &= \gamma_{15} x^{15} \oplus \gamma_{14} x^{14} \oplus \gamma_{12} x^{12}\\
\rho_{11} &= \gamma_{15} x^{15} \oplus \gamma_{14} x^{14} \oplus \gamma_{13} x^{13}\\
\gamma_4' \cdot G \rightarrow \rho_{12} &= \gamma_{19} x^{19} \oplus \gamma_{17} x^{17} \oplus \gamma_{16} x^{16}\\
\rho_{13} &= \gamma_{19} x^{19} \oplus \gamma_{18} x^{18} \oplus \gamma_{16} x^{16}\\
\rho_{14} &= \gamma_{19} x^{19} \oplus \gamma_{18} x^{18} \oplus \gamma_{17} x^{17}\\
\gamma_5' \cdot G \rightarrow \rho_{15} &= \gamma_{23} x^{23} \oplus \gamma_{21} x^{21} \oplus \gamma_{20} x^{20}\\
\rho_{16} &= \gamma_{23} x^{23} \oplus \gamma_{22} x^{22} \oplus \gamma_{20} x^{20}\\
\rho_{17} &= \gamma_{23} x^{23} \oplus \gamma_{22} x^{22} \oplus \gamma_{21} x^{21}\\
\gamma_6' \cdot G \rightarrow \rho_{18} &= \gamma_{27} x^{27} \oplus \gamma_{25} x^{25} \oplus \gamma_{24} x^{24}\\
\rho_{19} &= \gamma_{27} x^{27} \oplus \gamma_{26} x^{26} \oplus \gamma_{24} x^{24}\\
\rho_{20} &= \gamma_{27} x^{27} \oplus \gamma_{26} x^{26} \oplus \gamma_{25} x^{25}\\
\gamma_7' \cdot G \rightarrow \rho_{21} &= \gamma_{28} x^{28}\\
\rho_{22} &= \gamma_{28} x^{28}\\
\rho_{23} &= 0. \quad (3)
\end{aligned}$$

The predicted and actual parities of *sum*, *pass-thru*, or *alpha* modules (defined above) using the $(7, 4)$ Hamming codes are calculated by inserting the coefficients of the input and output vectors these modules into (3), respectively. The derivation of the all the Hamming code equations has been omitted for the sake of brevity.

IV. Error Coverage and FPGA Benchmark

This section presents the fault coverage and overhead results for the presented error detection schemes. The normal parity/interleaved parity schemes are adopted only for the squaring matrices and the trace functions while the $(7, 4)$ Hamming code is only adopted for the PB multiplier. Thus, in the hardware implementation of WG-29 presented here, the normal/interleaved signature schemes and the $(7, 4)$ Hamming code are combined for a higher-cumulative error coverage.

A. FPGA Benchmark and Overheads

The architecture of WG-29 [8], the PB multiplier [14], and the proposed error detection schemes are implemented using Verilog. The performance and implementation metrics for the error detection schemes for WG-29 are performed on devices xc7k160tfbg484-2L of the Xilinx Kintex-7 FPGA family and xc7s100fgga484-2 of Xilinx Spartan-7 FPGA family. Table II tabulates the overhead benchmarks for the combined the normal/interleaved signature and

TABLE II
BENCHMARK OF THE PROPOSED ERROR DETECTION SCHEMES ON KINTEX-7 AND SPARTAN-7 FPGA FAMILIES, (a) FPGA IMPLEMENTATION RESULTS FOR KINTEX-7 FPGA DEVICE XC7K160TFBG484-2L, (b) FPGA IMPLEMENTATION RESULTS FOR SPARTAN-7 FPGA DEVICE XC7S100FGGA484-2

(a)

WG-29 Architecture	Area (LUTs)	Power (mW)	Delay (ns)	Throughput ($Gbps$)	Efficiency ($Gbps/Area$)
Original	2,428	154	4.834	5.999	$2.471 \cdot 10^{-3}$
w/ Normal signature + Hamming code	3,298 (35.83%)	171 (11.04%)	5.038 (4.22%)	5.756 (-4.05%)	$1.745 \cdot 10^{-3}$ (-29.38%)
w/ Interleaved signature + Hamming code	3,305 (36.12%)	173 (12.34%)	5.152 (6.58%)	5.629 (-6.17%)	$1.703 \cdot 10^{-3}$ (-31.08%)

(b)

WG-29 Architecture	Area (LUTs)	Power (mW)	Delay (ns)	Throughput ($Gbps$)	Efficiency ($Gbps/Area$)
Original	2,463	170	5.172	5.607	$2.276 \cdot 10^{-3}$
w/ Normal signature + Hamming code	3,174 (28.87%)	189 (11.18%)	5.425 (4.89%)	5.346 (-4.65%)	$1.684 \cdot 10^{-3}$ (-26.01%)
w/ Interleaved signature + Hamming code	3,213 (30.45%)	189 (11.18%)	5.536 (7.04%)	5.238 (-6.56%)	$1.630 \cdot 10^{-3}$ (-28.39%)

TABLE III
PERFORMANCE EVALUATION OF THE PROTECTED WG-29 WITH OTHER WG AND CRYPTOGRAPHIC CIPHERS

Architecture	Area	Throughput
PB WG-29	Table II	Table II
WG-29 [2]	4044 LUTs	1853 Mbps
MOWG-29 [3]	4184 LUTs	1853 Mbps
WG-16 WGT+LFSR $\mathbb{F}_{(2^4)^4}$ [4]	1558 LUTs	-
WG-8 (TF1) [5]	$\sim$ 5106 slices	209 Mbps
ZUC [17]	1147 slices	1216 Mbps
Snow3g [17]	3559 slices	3328 Mbps
WAGE (logic based) [18]-[19]	$\sim$ 940 LUTs	$\sim$ 11.024 Gbps

the $(7, 4)$ Hamming code for hardware implementation of WG-29. From Table II, it is noted that the area, power, and delay overheads for the combined error detection schemes are low for both the FPGA families. The throughput (*output bits/delay*) and the efficiency (*throughput/area*) of the proposed architectures are also listed in Table II. For the Kintex-7 FPGA family [Table II(a)], the overheads for the normal signature with the $(7, 4)$ Hamming code are 35.83% for area, 11.04% for power increases, and 4.22% for the delay. For the same FPGA family, the area, power and delay overheads for the interleaved signature scheme with the $(7, 4)$ Hamming code are 36.12%, 12.34%, and 6.58%, respectively. Similar overheads are observed for using Spartan-7 FPGA family [Table II(b)], showing that our presented error detection schemes have acceptable overheads across different FPGA families.

The performance of the error detection schemes, presented for the first time for the hardware implementations of WG-29, can be further evaluated by comparing to the implementations of other WG and cryptographic ciphers tabulated in Table III. Additionally, the presented combined implementation of the schemes performed equally better when compared with the 23.09% for area, 31.78% for delay, and overall negligible for power overheads of the $(7, 4)$ Hamming code scheme implemented for PB multiplier in [15]. Thus, our presented error detection schemes, with slight modifications, can be adopted for other WG and stream ciphers while having acceptable overheads.

B. Fault Model and Error Coverage

In hardware implementations of stream ciphers, manufacturing faults could occur as stuck-at faults, such as single-bit upsets (SBUs), single-byte double-bit upsets (SBDBUs), single-byte triple-bit upsets (SBTBUs), single-byte quadruple-bit upsets (SBQBUs), multiple-bit upsets (MBUs), and multiple byte upsets (MB). Malicious adversaries could leverage these manufacturing faults or inject similar transient faults to acquire the secret key. In [16], differential fault analysis strategy is applied to WG-29, where six randomly placed faults are injected into the internal state of the ciphers to recover secret key via analyzing fault distribution in faulty ciphertexts.

The presented normal signature scheme is able to fully detect SBUs. Interleaved signature and Hamming code are able to detect the practical SBTBUs, SBQBUs, and MBs. Hamming codes are also able to detect random MBUs, adjacent MBUs, double-bit upsets, and odd bit errors with high probability, and can also perform error correction up to two bits (omitted in this article). The normal and interleaved signature schemes are applied to the squaring matrices as well as the two trace functions of WG-29 for error detection at the output of these components, while the $(7, 4)$ Hamming code scheme is adopted for the complex PB multiplier for higher-error coverage. Using the error coverage formula $100 \cdot (1-(0.5)^{\hat{p}})\%$, where $\hat{p}$ is equal to the total number of error flags generated per scheme, the error coverage for each of the proposed scheme applied to WG-29 is equal to 99.80% for normal signature, 99.99% for interleaved signature, and close to 100% for the $(7, 4)$ Hamming codes as there are 12 protected components in total.

The schemes are combined in hardware implementation for better-coverage across WG-29 components. For the combined normal/interleaved signature with $(7, 4)$ Hamming code implementation, the error coverage for 80 000 injected faults is determined via simulations in Vivado version 2020.2. A stuck-at fault model (stuck-at 1 or stuck-at 0) is considered to simulate manufacturing faults where the SBUs/MBUs are inserted randomly at the outputs of squaring matrices, trace functions, sum, pass-thru, and alpha modules using a faulty Verilog module. Then the predicted and actual parities of the modules are compared and an error flag is triggered if the parities do not match to indicate the presence of faults. The simulation results showed that the cumulative error coverage for SBUs/MBUs using the combined schemes in the hardware implementation of WG-29 is over 99.99% (~100%). Thus, our presented error detection schemes achieve high-error coverage in the hardware implementation of WG-29.

V. CONCLUSION

This article proposes normal signature, interleaved signature, and the $(7, 4)$ Hamming code-based error detection schemes for the squaring matrices, trace functions, and the PB multiplier of the stream cipher WG-29 for the first time. The derived normal signature,

interleaved signature, and $(7, 4)$ Hamming codes are capable of detecting both SBUs and MBUs with high-error coverage, hence providing measures against both manufacturing and maliciously injected faults. The performance benchmarks of the proposed schemes on the Xilinx Kintex-7 and Xilinx Spartan-7 FPGA families along with error coverage simulations are conducted using Xilinx Vivado 2020.2. The FPGA overheads of the proposed protected WG-29 are between 28.87% to 36.12% for area, 11.04% to 12.34% for power, and between 4.22% to 7.04% for the delay across two Xilinx FPGA families: 1) Kintex-7 and 2) Spartan-7, with an error coverage close to 100%. The results of the performed benchmarks are further evaluated by comparing them to other state of the art WG and cryptographic ciphers, consequently exhibiting that the presented schemes achieve high-error coverage with sufficient overheads. Additionally, the presented schemes can be modified for hardware implementations of other cryptographic ciphers with similar underlying primitives, such as other WG-based ciphers. Thus, the hardware constructions of WG-29 are made more reliable against manufacturing and transient faults with the presented error detection schemes.

REFERENCES

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

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

[3] C. H. Lam, M. Aagaard, and G. Gong, "Hardware implementations of multi-output Welch-Gong ciphers," Dept. Elecr. Comput. Eng., Univ. Waterloo, Waterloo, ON, Canada, Rep. CACR2011-01, 2009.

[4] X. Fan, N. Zidaric, M. Aagaard, and G. Gong, "Efficient hardware implementation of the stream cipher WG-16 with composite field arithmetic," in *Proc. Inter. Workshop Trust. Embedded Devices (TrustED)*, 2013, pp. 21–34.

[5] G. Yang, X. Fan, M. Aagaard, and G. Gong, "Design space exploration of the lightweight stream cipher WG-8 for FPGAs and ASICs," in *Proc. Workshop Embed. Syst. Sec. (WESS)*, vol. 8, 2013, pp. 1–10.

[6] G. Gong and Y. Nawaz. "The WG stream cipher." eSTREAM, ECRYPT Stream Cipher Project. 2005. [Online]. Available: http://www.ecrypt.eu.org/stream/wgp2.html

[7] M. Sattarov, "Hardware implementations of the lightweight Welch-Gong stream cipher family using polynomial bases," M.S. thesis, Dept. Electr. Comput. Eng., Univ. Waterloo, Waterloo, ON, Canada, 2019.

[8] H. El-Razouk, A. Reyhani-Masoleh, and G. Gong, "New hardware implementations of WG(29,11) and WG-16 stream ciphers using polynomial basis," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 2020–2035, Jul. 2015.

[9] S. Saha, A. Bag, D. B. Roy, S. Patranabis, and D. Mukhopadhyay, "Fault template attacks on block ciphers exploiting fault propagation," in *Proc. Int. Conf. Theory App. Crypto. Techn.*, 2020, pp. 612–643.

[10] S. Saha and D. Mukhopadhyay, "Transform without encode is not sufficient for SIFA and FTA security: A case study," in *Proc. Int. Workshop Construct. Side-Channel Anal. Secure Design*, 2021, pp. 85–104.

[11] S. Patranabis, A. Chakraborty, D. Mukhopadhyay, and P. P. Chakrabarti, "Fault space transformation: A generic approach to counter differential fault analysis and differential fault intensity analysis on AES-like block ciphers," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 1092–1102, 2017.

[12] S. Bauer, S. Rass, and P. Schartner, "Generic parity-based concurrent error detection for lightweight ARX ciphers," *IEEE Access*, vol. 8, pp. 142016–142025, 2020.

[13] C. Y. Lee, P. K. Meher, and J. C. Patra, "Concurrent error detection in bit-serial normal basis multiplication over $GF(2^m)$ using multiple parity prediction schemes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1234–1238, Aug. 2009.

[14] A. Reyhani-Masoleh and M. A. Hasan, "Error detection in polynomial basis multipliers over binary extension fields," in *Proc. CHES*, 2002, pp. 515–528.

[15] A. C. Canto, M. Mozaffari-Kermani, and R. Azarderakhsh, "Reliable CRC-based error detection constructions for finite field multipliers with applications in cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 1, pp. 232–236, Jan. 2021.

[16] M. A. Orumiehchiha, S. Rostami, E. Shakour, and J. Pieprzyk, "A differential fault attack on the WG family of stream ciphers," *J. Cryptogr. Eng.*, vol. 10, pp. 189–195, Mar. 2020.

[17] P. Kitsos, N. Sklavos, G. Provelengios, and A. N. Skodras, "FPGA-based performance analysis of stream ciphers ZUC, Snow3g, Grain V1, Mickey V2, Trivium and E0," *J. Microprocess. Microsyst.*, vol. 37, no. 2, pp. 235–245, 2013.

[18] M. Aagaard, R. AlTawy, G. Gong, K. Mandal, R. Rohit, and N. Zidaric. "WAGE: An authenticated encryption algorithm." Sep. 2019. [Online]. Available: https://uwaterloo.ca/communications-security-lab/lwc/wage

[19] J. Kaur, A. Sarker, M. M. Kermani, and R. Azarderakhsh, "Hardware constructions for error detection in lightweight Welch-Gong (WG)-oriented streamcipher WAGE benchmarked on FPGA," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1208–1215, Apr.–Jun. 2022.