Hardware Constructions for Error Detection in Lightweight Authenticated Cipher ASCON Benchmarked on FPGA

Jasmin Kaur, *Graduate Student Member, IEEE*, Mehran Mozaffari Kermani, *Senior Member, IEEE*, and Reza Azarderakhsh, *Member, IEEE*

***Abstract*—Lightweight cryptography plays a vital role in securing resource-constrained embedded systems such as deeply-embedded systems, RFID tags, sensor networks, and the Internet of nano-Things. ASCON is one of the finalists for the National Institute of Standards and Technology (NIST) lightweight cryptography standardization competition advanced to the final round in April 2021. It is designed to provide authenticated encryption with associated data (AEAD) and hashing functionalities in hardware and software implementations. ASCON's lightweight design utilizes a 320-bit permutation, bit-sliced into five 64-bit register words, providing 128-bit level security. This brief, for the first time, proposes error detection mechanisms for secure hardware implementations of ASCON. The proposed schemes, i.e., signature, interleaved signature, and cyclic redundancy check approaches are presented for both LUT and logic-based implementations of ASCON. The proposed error detection schemes are also benchmarked on two FPGA families (Spartan-7 and Kintex-7), achieving acceptable area, power, and delay overheads. The proposed mechanisms also provide a high error coverage (99.99%) shown via simulations performed for 640,000 injected faults. Hence, these approaches aim to make the respective ASCON architectures more reliable.**

***Index Terms*—ASCON, authenticated cipher, cyclic redundancy check (CRC), field-programmable gate array (FPGA), lightweight permutation.**

I. Introduction

AS THE Internet of Things (IoT) grows, especially the Internet of Nano-Things, and devices become smaller and heavily dependent on embedded systems, security often becomes an issue due to the compactness of such systems. Thus, lightweight ciphers can be prone to security attacks such as statistical fault analysis (SFAs) [1] or differential fault analysis (DFAs) [2].

As the National Institute of Standards and Technology (NIST) moves towards standardizing lightweight cryptography, it has approved many emerging efficient lightweight cipher algorithms such as [3]–[4]. ASCON [5], a NIST finalist for lightweight cryptography competition which was advanced in April 2021 [6], and a CAESAR competition winner [7], is a lightweight cipher suite that is designed to provide authenticated encryption with associated data (AEAD) and hashing functionalities. ASCON provides high security and robustness in implementations while having a low area, which is a requirement for resource-constrained devices.

The cryptographic properties of ASCON make it secure against various linear and differential cryptanalysis. However, a number of fault analyses such as statistical ineffective fault attacks (SIFA) [8], subset fault analysis (SSFA) [9], and fault intensity map analysis (FIMA) [10], have targeted the vulnerabilities of ASCON S-Box. These attacks rely on biased fault injections to recover the secret key by ciphertext analysis. Therefore, it is important to strengthen all 64 instances of the ASCON S-Box using error detection to make it secure for reliable hardware implementations. Moreover, in authenticated encryption schemes, both the authentication and encryption could be compromised due to natural faults, thus reducing their reliability. Error detection schemes [13]–[22] are often used for reliable cryptographic implementations through fault detection. In this brief, we propose error detection schemes for the non-linear S-Box of the ASCON's lightweight authenticated cipher variant ASCON-128. The contributions of this brief are as follows.

- To the best of the authors' knowledge, this is the first work that proposes logic gate-based and look-up table (LUT) -based error detection schemes for the hardware implementations of ASCON-128. The proposed schemes are implemented on the 5-bit S-Box in the nonlinear layer of ASCON-128 and aim to provide security.
- The presented schemes, signatures, interleaved signatures, and cyclic redundancy check (CRC-3) are tailored for detecting both permanent and transient faults (single, biased, or burst/adjacent faults) with acceptable overheads with their high error coverage shown via simulations.
- The proposed schemes are benchmarked on two field-programmable gate array (FPGA) hardware platforms to confirm the achieved objectives.

II. Preliminaries

ASCON [5] has the key, nonce, and tag of 128 bits each while the message length is either 64 bits (ASCON-128

Manuscript received November 30, 2021; accepted December 15, 2021. Date of publication December 17, 2021; date of current version March 28, 2022. This work was supported by the U.S. Department of Commerce through U.S. Federal Agency under Award 60NANB20D013, National Institute of Standards and Technology (NIST). This brief was recommended by Associate Editor B.-H. Gwee. *(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).

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

Digital Object Identifier 10.1109/TCSII.2021.3136463

TABLE I
LUT REPRESENTATION OF THE NON-LINEAR 5-BIT S-BOX *SB* OF ASCON-128 IN HEXADECIMAL FORM

μ	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
SB	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

TABLE II
PROPOSED SIGNATURE ($\hat{p_0}$) FOR THE LUT-BASED APPROACH

μ	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$\hat{p_0}$	1	1	1	0	1	1	0	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	0	0	1	0	1	1	1	0	0	0

TABLE III
PROPOSED INTERLEAVED SIGNATURES ($\hat{p_1}, \hat{p_2}$) FOR OUR LUT-BASED APPROACH

μ	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$\hat{p_1}$	1	1	1	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1	0	0	0	0	1
$\hat{p_2}$	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	1	1	0	0	1

TABLE IV
PROPOSED CRC-3 SIGNATURES ($\hat{p_3}, \hat{p_4}, \hat{p_5}$) FOR OUR LUT-BASED APPROACH

μ	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$\hat{p_3}$	0	1	1	0	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
$\hat{p_4}$	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
$\hat{p_5}$	1	0	1	1	1	1	0	0	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1

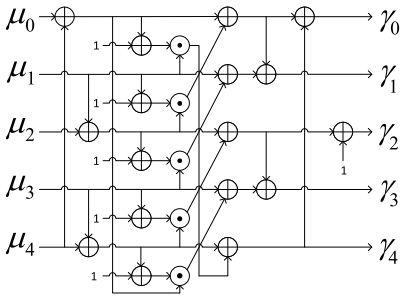

Fig. 1. The design of the 5-bit S-Box of ASCON [15]. Here, the symbols $\oplus$ and $\odot$ correspond to logic gates XOR and AND, respectively.

variant) or 128 bits (ASCON-128a variant). It uses a sponge-duplex mode [11] for its AEAD implementation and utilizes two 320-bit permutations, i.e., p^a and p^b, in its internal state *IS*, where a and b are the number of rounds. The two permutations are bit-sliced into five 64-bit register words μ_i, where $IS = \{\mu_0, \mu, \mu_2, \mu_3, \mu\}$. Both of the permutations iteratively apply a substitution-permutation network (SPN)-based round transformation (ρ) which is split into three parts: Adding round constants (ρ_c), applying substitution layer (ρ_s), and finally applying the linear layer (ρ_l) for diffusion. A full ASCON permutation consists of 12 rounds in total. The substitution layer uses a non-linear 5-bit S-Box (Fig. 1) and is chosen such that the one output bit is dependent on at least four of the input bits. Since the S-Box is realized using only Boolean functions, its design makes it super compact and lightweight for both FPGA and ASIC implementations. The S-Box is implemented on each bit-slice of the five registers, and updates the internal state *IS* with 64 parallel applications. Table I shows the hexadecimal representation of the 5-bit S-Box of ASCON-128.

III. PROPOSED ERROR DETECTION ARCHITECTURES

In this section, we present efficient and overhead-aware error detection schemes for the 5-bit S-Box of ASCON-128. The following parity based error detection schemes [12] have been extensively studied in the literature for various ciphers [13]–[15], but are proposed for ASCON-128 for the first time. We propose a) logic gate-based, and b) LUT-based variants of one-bit signature, interleaved signatures, and CRC-3 scheme to detect bit-errors in hardware implementations of ASCON-128. Traditionally, the state-of-the-art S-Boxes (including those of the AES) can be implemented on FPGAs effectively using LUTs because memory units (block memories or pipelined distributed memories) are abundant on FPGAs but are expensive on ASICs (memory macros and regular register implementations). The two aforementioned variants are well-known in cryptographic engineering, and thus we use them as the foundations for applying our proposed schemes.

A. Proposed Signature-Based Schemes for the S-Box

Here, the low-cost realization of logic gate-based formulations, and their LUT representations listed in Tables II, III, and IV are presented. The following remarks initiate our error detection scheme descriptions by presenting two sets of logic gate-based equations that have been formulated in this brief. In the remarks below, Logic-I (1) is the original design of the S-Box (Fig. 1) as described by the designers of ASCON [5], while Logic-II (2) represents another compact design of the S-Box derived from the mappings of the input and output vectors as listed in Table I. The formulations for the presented error detection schemes presented in (3), (5), and (7) are derived using (1) for the presented error detection schemes, while the formulations presented in (4), (6), and (8) are derived using (2), and mainly differ in the utilization of logical gates. All the presented equations are simplified for the sake of brevity. Logic-I, Logic-II, and LUT-based implementations are architecture oblivious and can be utilized in low-cost constructions of ASCON-128 in resource-constrained deeply-embedded systems applications. Throughout this brief, the symbols $\oplus$, $\vee$, and the *bar* represent bit-wise XOR, OR, and NOT operations, respectively.

Remark 1: The following shows the formulations for the original 5-bit S-Box of ASCON-128 (Fig. 1) [5] presented as Logic I (1), for the input vectors $(\mu_0, \mu_1, \mu_2, \mu_3, \mu_4)$ and the output vectors $(\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4)$.

$$\begin{aligned}
\gamma_0 &= ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \oplus ((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)) \\
\gamma_1 &= (\mu_1 \oplus (\overline{(\mu_1 \oplus \mu_2)}\mu_3)) \oplus ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \\
\gamma_2 &= \overline{(\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)} \\
\gamma_3 &= (\mu_3 \oplus (\overline{(\mu_3 \oplus \mu_4)}(\mu_0 \oplus \mu_4))) \oplus ((\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)) \\
\gamma_4 &= ((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)). \qquad (1)
\end{aligned}$$

Remark 2: The following shows the formulation of an alternate design of the 5-bit S-box using the input to output mapping given in Table I presented as Logic II (2), for the input vectors $(\mu_0, \mu_1, \mu_2, \mu_3, \mu_4)$ and the output vectors $(\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4)$.

$$\begin{aligned}
\gamma_0 &= (\bar{\mu_0}\bar{\mu_1})(\mu_2 \oplus \mu_3) \vee \mu_1(\mu_3 \oplus \bar{\mu_4}) \vee (\mu_0\bar{\mu_1})(\mu_2 \oplus \bar{\mu_3}) \\
\gamma_1 &= (\bar{\mu_1} \oplus \mu_2)(\mu_1 \oplus \bar{\mu_3})(\mu_0 \oplus \mu_4) \vee (\mu_0 \oplus \bar{\mu_4})(\bar{\mu_1}\mu_3 \vee \mu_2\bar{\mu_3} \vee \mu_1\bar{\mu_2}) \\
\gamma_2 &= (\mu_1 \oplus \bar{\mu_2})(\bar{\mu_4} \vee \mu_3) \vee (\mu_1 \oplus \mu_2)(\bar{\mu_3}\mu_4) \\
\gamma_3 &= (\mu_1 \oplus \bar{\mu_2})(\mu_0 \vee (\mu_3 \oplus \mu_4)) \vee (\bar{\mu_0}(\mu_1 \oplus \mu_2)(\mu_3 \oplus \bar{\mu_4})) \\
\gamma_4 &= (\bar{\mu_1}(\mu_3 \oplus \mu_4)) \vee (\mu_1(\mu_0 \oplus \bar{\mu_3})). \qquad (2)
\end{aligned}$$

The following derives the logic-gate based formulations using Logic-I (1) and Logic-II (2) for the presented error detection schemes using the five bit input $(\mu_0, \mu_1, \mu_2, \mu_3, \mu_4)$ and output $(\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4)$ vectors of ASCON-128 S-Box.

1) Signature - The one-bit signature $\hat{p_0}$ is computed as modulo-2 addition of each bit of the output vector of the S-Box. The logic gate-based equations for this scheme are shown in (3) and (4), and its LUT-based implementation is shown in Table II.

$$\begin{aligned}
\hat{p_0} &= ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \oplus ((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)) \\
&\quad \oplus(\mu_1 \oplus (\overline{(\mu_1 \oplus \mu_2)}\mu_3)) \oplus ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \\
&\quad \oplus\overline{(\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)} \oplus ((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)) \\
&\quad \oplus(\mu_3 \oplus (\overline{(\mu_3 \oplus \mu_4)}(\mu_0 \oplus \mu_4))) \oplus ((\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)) \qquad (3) \\
\hat{p_0} &= (\bar{\mu_0}\bar{\mu_1}\bar{\mu_3}) \vee (\bar{\mu_2}\mu_3\bar{\mu_4}) \vee (\bar{\mu_0}\mu_2\mu_3\mu_4) \vee (\bar{\mu_1}\bar{\mu_3}\mu_4) \vee (\mu_0\bar{\mu_2}\mu_3) \\
&\quad \vee(\mu_0\mu_1\bar{\mu_3}\bar{\mu_4}). \qquad (4)
\end{aligned}$$

2) Interleaved Signature - The interleaved signatures $(\hat{p_1}, \hat{p_2})$ are parity-based signatures which are computed by modulo-2 addition of the interleaved bits, even and odd, respectively, of the output vector of the S-Box. The logic gate-based equations of this scheme are shown in (5) and (6), and its LUT-based implementation is given in Table III.

$$\begin{aligned}
\hat{p_1} &= ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \oplus ((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)) \\
&\quad \oplus\overline{(\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)} \oplus ((\mu_3 \oplus \mu_4) \oplus (\overline{(\overline{(\mu_0 \oplus \mu_4)})\mu_1})) \\
\hat{p_2} &= (\mu_1 \oplus (\overline{(\mu_1 \oplus \mu_2)}\mu_3)) \oplus ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \\
&\quad \oplus(\mu_3 \oplus (\overline{(\mu_3 \oplus \mu_4)}(\mu_0 \oplus \mu_4))) \oplus ((\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)) \qquad (5) \\
\hat{p_1} &= (\bar{\mu_0}\mu_1\bar{\mu_2}\mu_3\mu_4) \vee (\mu_0)(\bar{\mu_1}\mu_3\mu_4 \vee \mu_1\bar{\mu_2}\bar{\mu_3} \vee \bar{\mu_4} \vee \mu_2\mu_3\mu_4) \\
&\quad \vee(\bar{\mu_0})(\bar{\mu_4} \vee \bar{\mu_3})(\bar{\mu_1} \vee \mu_2) \\
\hat{p_2} &= (\bar{\mu_0})(\mu_2\mu_3 \vee \mu_1\mu_3 \vee \mu_1\mu_2) \vee (\mu_0\bar{\mu_1}\bar{\mu_2}\mu_3\bar{\mu_4}) \vee (\mu_2\mu_3\mu_4) \\
&\quad \vee(\mu_1\mu_2\bar{\mu_3}\bar{\mu_4}) \vee (\mu_0\mu_4)(\bar{\mu_1}\bar{\mu_3} \vee \mu_1\bar{\mu_2}). \qquad (6)
\end{aligned}$$

TABLE V
THE ERROR COVERAGE OF THE PROPOSED ERROR DETECTION SCHEMES FOR 640, 000 INJECTED FAULTS

Proposed Error Detection Scheme	Type of Fault	Faults Detected	Error Coverage
One-bit sig.	SBU	640,000	100%
	MBU	639,999	99.99984%
Interleaved sig.	SBU	640,000	100%
	MBU	639,999	99.99984%
CRC-3 scheme	SBU	640,000	100%
	MBU	639,999	99.99984%

3) CRC-3 Check - Reducing the output polynomial $f(x) = \gamma_0x^4 + \gamma_3x^3 + \gamma_2x^2 + \gamma_1x + \gamma_4$ using the CRC-3 reduction polynomial $g(x) = x^3 + x + 1$, we derive the CRC-3 equation as $(\gamma_0 + \gamma_2)x^2 + (\gamma_0 + \gamma_1 + \gamma_3)x^1 + (\gamma_1 + \gamma_4)x^0$, where the coefficient terms of x^0, x^1, and x^2 correspond to the CRC-3 signatures $\hat{p_3}, \hat{p_4}$, and $\hat{p_5}$, respectively. The logic gate-based equations for CRC-3 scheme are shown in (7) and (8), the LUT-based implementation is given in Table IV.

$$\begin{aligned}
\hat{p_3} &= (\mu_1 \oplus (\overline{(\mu_1 \oplus \mu_2)}\mu_3)) \oplus ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \\
&\quad \vee((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)) \\
\hat{p_4} &= ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \oplus ((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)) \\
&\quad \vee(\mu_1 \oplus (\overline{(\mu_1 \oplus \mu_2)}\mu_3)) \oplus ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \\
&\quad \vee(\mu_3 \oplus (\overline{(\mu_3 \oplus \mu_4)}(\mu_0 \oplus \mu_4))) \oplus ((\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)) \\
\hat{p_5} &= ((\mu_0 \oplus \mu_4) \oplus (\bar{\mu_1}(\mu_1 \oplus \mu_2))) \oplus ((\mu_3 \oplus \mu_4) \oplus (\overline{(\mu_0 \oplus \mu_4)}\mu_1)) \\
&\quad \vee\overline{(\mu_2 \oplus \mu_1) \oplus (\bar{\mu_3}\mu_4)} \qquad (7) \\
\hat{p_3} &= ((\bar{\mu_3} \vee \mu_1\mu_2)\mu_4) \vee ((\bar{\mu_1}\bar{\mu_2} \vee \mu_3)\mu_0) \vee (\bar{\mu_1}\mu_3\bar{\mu_4}) \\
&\quad \vee(\bar{\mu_0}\mu_1\bar{\mu_2}\bar{\mu_4}) \vee (\bar{\mu_0}\mu_2\bar{\mu_3}) \\
\hat{p_4} &= ((\bar{\mu_1} \vee \mu_0)\mu_4) \vee (\mu_1\bar{\mu_3}\bar{\mu_4}) \vee (\mu_0\bar{\mu_1}\bar{\mu_2}) \vee (\mu_2\mu_3) \\
&\quad \vee((\mu_2 \vee \mu_3)\bar{\mu_0}) \\
\hat{p_5} &= ((\mu_1 \oplus \bar{\mu_2})\bar{\mu_4}) \vee ((\mu_1 \oplus \mu_3)\bar{\mu_2}) \vee (\bar{\mu_0}\bar{\mu_1}\mu_2\bar{\mu_3}) \\
&\quad \vee((\mu_4 \vee \mu_3)\mu_0\bar{\mu_1}) \vee (\mu_1\mu_3\mu_4). \qquad (8)
\end{aligned}$$

IV. ERROR COVERAGE AND FPGA BENCHMARK

The results for overheads and error coverage for the proposed error detection schemes are presented in this section.

A. Fault Model and Error Coverage

Our proposed signature-based error detection schemes detect several faults such as stuck-at faults, single-bit upset (SBUs), single-byte double-bit upsets (SBDBUs), single-byte triple-bit upsets (SBTBUs), single-byte quadruple-bit upsets (SBQBUs), other single byte (OSB) faults, and multiple byte (MB) faults. Specifically, SBUs and SBTBUs are detected fully through parities. Interleaved signatures help in detection of the SBTBUs, SBQBUs, OSBs, and MBs (that are more practical to mount than the costly single faults). Cyclic redundancy schemes such as CRC-3 can detect SBUs, random multi-bit upsets (MBUs), adjacent MBUs, double-bit upsets and odd errors with high probability.

As the 5-bit ASCON-128 S-Box is implemented 64 times in parallel, 64 error flags are produced in every permutation round of ASCON-128. The error flags (ef) generated for each of the proposed error detection schemes is either 1 (one-bit

TABLE VI
BENCHMARK OF THE PRESENTED SCHEMES FOR THE S-BOX OF ASCON-128 ON SPARTAN-7 AND KINTEX-7 FPGA FAMILIES

(a) FPGA implementation results for Spartan-7 FPGA device xc7s75fgga676-2

ASCON S-Box Architecture		Area ($Slice$)	Power (mW)	Delay (ns)	Throughput ($Gbps$)	Efficiency ($Gbps/Slices$)
Logic I - based	Original	371	99	9.630	6.646	$17.914 \cdot 10^{-3}$
	w/ One-bit	373(0.54%)	99	9.649(0.20%)	6.633(0.20%)	$17.783 \cdot 10^{-3}$(0.73%)
	w/Interleaved	380(2.43%)	99	9.655(0.26%)	6.629(0.26%)	$17.444 \cdot 10^{-3}$(2.62%)
	w/ CRC-3	407(9.70%)	99	10.079(4.66%)	6.350(4.4%)	$15.601 \cdot 10^{-3}$(12.91%)
Logic II - based	Original	375	99	9.602	6.665	$17.773 \cdot 10^{-3}$
	w/ One-bit	376(0.27%)	99	9.640(0.40%)	6.639(4.13%)	$17.656 \cdot 10^{-3}$(0.66%)
	w/Interleaved	377(0.53%)	99	9.754(1.58%)	6.561(1.56%)	$17.403 \cdot 10^{-3}$(2.08%)
	w/ CRC-3	380(1.33%)	100(1.01%)	9.835(2.43%)	6.507(2.37%)	$17.123 \cdot 10^{-3}$(3.66%)
LUT - based	Original	371	99	9.866	6.487	$17.485 \cdot 10^{-3}$
	w/ One-bit	372(0.27%)	100(1.01%)	9.926(0.61%)	6.448(0.60%)	$17.333 \cdot 10^{-3}$(0.86%)
	w/Interleaved	377(1.62%)	100(1.01%)	9.934(0.69%)	6.443(0.67%)	$17.090 \cdot 10^{-3}$(2.26%)
	w/ CRC-3	425(14.56%)	100(1.01%)	9.952(0.87%)	6.431(0.86%)	$15.131 \cdot 10^{-3}$(13.46%)

(b) FPGA implementation results for Kintex-7 FPGA device xc7k160tifbg484-2L

ASCON S-Box Architecture		Area ($Slice$)	Power (mW)	Delay (ns)	Throughput ($Gbps$)	Efficiency ($Gbps/Slices$)
Logic I - based	Original	376	88	9.540	6.709	$17.843 \cdot 10^{-3}$
	w/ One-bit	381(1.33%)	89 (1.14%)	9.545(0.05%)	6.705(0.06%)	$17.598 \cdot 10^{-3}$(1.37%)
	w/Interleaved	384(2.13%)	89 (1.14%)	9.571(0.32%)	6.687(0.33%)	$17.414 \cdot 10^{-3}$(2.40%)
	w/ CRC-3	385(2.39%)	89 (1.14%)	9.692(1.60%)	6.603(1.58%)	$17.150 \cdot 10^{-3}$(3.88%)
Logic II - based	Original	378	89	9.726	6.580	$17.407 \cdot 10^{-3}$
	w/ One-bit	382(1.06%)	89	9.805(0.81%)	6.527(0.81%)	$17.086 \cdot 10^{-3}$(1.84%)
	w/Interleaved	385(1.85%)	89	9.996(2.78%)	6.403(2.69%)	$16.631 \cdot 10^{-3}$(4.46%)
	w/ CRC-3	422(11.64%)	89	10.174(4.60%)	6.291(4.39%)	$14.907 \cdot 10^{-3}$(14.36%)
LUT - based	Original	361	88	9.433	6.785	$18.795 \cdot 10^{-3}$
	w/ One-bit	363(0.55%)	89 (1.14%)	9.445 (0.13%)	6.776 (0.13%)	$18.667 \cdot 10^{-3}$(0.68%)
	w/Interleaved	372(3.04%)	89 (1.14%)	9.986 (5.86%)	6.409 (5.54%)	$17.228 \cdot 10^{-3}$(8.33%)
	w/ CRC-3	384(6.37%)	89 (1.14%)	10.026 (6.29%)	6.383 (5.92%)	$16.224 \cdot 10^{-3}$(13.68%)

signature), 2 (interleaved signatures), or 3 (CRC-3). The error coverage of the presented error detection schemes increases as the number of S-Boxes with error detection increases. For a single S-Box, the error coverage is ~50% for one-bit signature, ~75% for interleaved signature and ~85% for CRC-3 scheme. Since the error detection is added to all 64 S-Box instances in ASCON-128 permutation, the error coverage of the schemes can be calculated using the formula $100 \cdot (1-(0.5)^{signatures})\%$, where *signatures* denote the total number of signatures generated by an error detection scheme per S-Box, calculated as $signatures = ef \cdot 64$. Therefore, for the proposed schemes the error coverage for all 64 S-Boxes is as follows: a) One-bit signature scheme = $100 \cdot (1-(0.5)^{1 \cdot 64})\%$, b) Interleaved signatures scheme = $100 \cdot (1-(0.5)^{2 \cdot 64})\%$, and c) CRC-3 scheme = $100 \cdot (1-(0.5)^{3 \cdot 64})\%$. These are all very close to full error coverage of 100%, as shown by the results of Table V.

Using simulations, the actual error coverage of the proposed schemes is determined for 640,000 injected errors and tabulated in Table V. A stuck-at fault model is considered for error coverage, and faults are injected using VHDL in the hardware implementation of ASCON-128. For example, for SBUs, any single bit in the S-Box output vector is kept stuck-at 0, while for MBUs, random bits of the S-Box output vector is kept stuck-at 0 using AND operation. In our fault model, we assume that all the stuck-at faults are injected at the output of every S-Box during the execution phase. The parities of the input and the output vectors of the S-Boxes are computed and compared to give the error flag value as either "0" (in case of no error) or "1" (when an error gets detected), which is then used to signal the presence of an error. We only focus upon including error detection capabilities in the architecture of ASCON-128. This is because fault attacks are transient in nature but the attacks are noticed by the observers regardless. Hence, we note that in cryptographic applications, error correction is often not intended concerning natural and transient faults. Moreover, since the presented schemes are only implemented on the S-Box, they would not work for erroneous input to S-Boxes. For a fault models which consider erroneous input, other components of the cipher need to be protected as well to achieve a fully-protected cipher [20], as the protection for previous transformations can detect erroneous inputs. The aforementioned extension is straightforward as the other transformations are linear and easy to protect.

Fault analysis techniques such as SIFA [8], SSFA [9], and FIMA [10], utilize various fault injection and analysis techniques to target the 5-bit S-Box of ASCON. Authors of [8] mount SIFA on ASCON by using double-fault injection and key division strategies. The faults are injected in two selected instances of the S-Boxes in the last permutation round of the finalization stage to recover the secret key by analyzing the tag values resulting from fault injections. Authors of [9] mount SSFA on ASCON-128 by using a combination of correlation attack and subset cryptanalysis. The proposed attack assumes that the attacker can inject bit-reset faults (injected as 1-bit, 1-byte, multiple bytes, etc.) to recover the secret key. FIMA [10], combines different statistical fault analysis techniques to recover the secret key by comparing faulty ciphertexts to correct ciphertexts under ineffective fault injections and data dependent intensity profiles. Since FIMA combines the properties of SIFA and SSFA, the faults are injected similarly as the aforementioned techniques for fault analysis. Our proposed error detection schemes provide high error coverage for a number of bit upsets and therefore provide significant protection against fault attacks listed above.

B. FPGA Benchmark and Overheads

The overhead benchmarks of the proposed schemes, implemented for the S-Box of ASCON-128, are presented in Table VI. The FPGA implementations are performed on devices xc7s75fgga676-2 of Xilinx Spartan-7 FPGA family and xc7k160tifbg484-2L of Xilinx Kintex-7 FPGA family, for a thorough evaluation. Xilinx Vivado version 2018.3 has been used for performance and implementation metrics derivations. The results of our implementations for the logic gate-based and LUT-based variants are tabulated in Table VI and have been performed using VHDL by modifying the ASCON-128 RTL code [5].

From Table VI, we notice that the overheads for the proposed error detection schemes, when implemented on ASCON-128, are low for both FPGA families. The overheads for the presented schemes on Spartan-7 FPGA family (Table VIa) are less than 10% for Logic I architecture, less than 5% for Logic II architecture, and less than 15% for LUT-based architecture. The overhead results for the presented schemes on Kintex-7 FPGA (Table VIb) are less than 4% for Logic I architecture, less than 15% for Logic II architecture, and less than 14% for LUT-based architecture. Across the presented architectures for the proposed schemes in Table VI, a gradual increase is observed in the area and delay overheads, while the power overheads are almost negligible (under 2%). Moreover, the overhead results for presented schemes using Logic II and LUT-based architectures, when compared to the overheads of Logic I, are marginal. Therefore, our presented schemes have low overheads across different FPGA families for all the presented architectures.

We note that this is the first work to present error detection for ASCON-128; however, the overheads of similar works performed on other cryptographic ciphers can be compared for evaluation. In [21], the authors propose fault detection schemes for block cipher SIMON, which incur 30.14% and 3.33% area and delay overhead, respectively. Moreover, in [22], the area and throughput overheads are 14.33% for the parity prediction scheme of a well-known stream cipher ChaCha. Thus, our error detection schemes achieve acceptable overhead with more than 99.99% error coverage as shown by the tabulated results.

V. Conclusion

This brief proposes error detection schemes for all the 64 5-bit S-Boxes of the lightweight cipher suite ASCON for the first time, implemented on ASCON-128 variant. The signature and cyclic redundancy-based error detection schemes have been derived and implemented using both logic gate-based and LUT-based techniques. The error simulations of the derived one-bit signature, interleaved signature, and CRC-3 schemes are performed in the Vivado design suite and show a high error coverage of more than 99.999% for most SBUs and MBUs. The design implementation is performed for Xilinx Spartan-7 and Kintex-7 FPGA families in the Vivado design suite and the overheads results for both the FPGAs are less than 10% for Logic-I, 15% for Logic-II, and 15% for LUT-based architectures. Hence, the simulation and implementation results show that the schemes provide high error coverage with acceptable architecture overheads across different FPGA families for resource-constrained applications. Therefore, the architecture of ASCON cipher suit variants can be made more secure through the proposed error detection schemes.

References

[1] X. Zhang, X. Feng, and D. Lin "Fault attack on the authenticated cipher ACORN v2," in *Proc. Secur. Commun. Netw.*, 2017, pp. 1–16.

[2] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *J. Cryptograph. Eng.*, vol. 5, no. 3, pp. 153–169, 2015.

[3] M. S. Turan, K. A. McKay, Ç. Çalık, D. Chang, and L. Bassham, "Status report on the first round of the NIST lightweight cryptography standardization process," U.S. Dept. Commerce, Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, NIST Interagency/Internal Rep. 8268, 2019.

[4] I. Dinur and G. Leurent, "Preface to volume 2020, special issue on designs for the NIST lightweight standardisation process," in *Proc. IACR Trans. Symmetric Cryptol.*, 2020, pp. 1–4.

[5] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, *ASCON v1. 2.*, Sub. NIST Lightweight Cryptography Stand, Process, Gaithersburg, MD, USA 2019. [Online]. Available: https://ascon.iaik.tugraz.at/specification.html

[6] Accessed: Nov. 2021. [Online]. Available: https://csrc.nist.gov/Projects/lightweight-cryptography/finalists

[7] Accessed: Nov. 2021. [Online]. Available: https://competitions.cr.yp.to/caesar-submissions.html

[8] K. Ramezanpour, P. Ampadu, and W. Diehl, "A statistical fault analysis methodology for the ASCON authenticated cipher," in *Proc. IEEE Int. Symp. Hardw. Orient. Secur. Trust (HOST)*, 2019, pp. 41–50.

[9] P. Joshi and B. Mazumdar, "SSFA: Subset fault analysis of ASCON-128 authenticated cipher," *J. Microelectron. Rel.*, vol. 123, Aug. 2021, Art. no. 114155.

[10] K. Ramezanpour, P. Ampadu, and W. Diehl, "FIMA: Fault intensity map analysis," in *Proc. Int. Workshop Constr. Side-Channel Anal. Secure Design*, 2019, pp. 63–79.

[11] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Duplexing the sponge: Single-pass authenticated encryption and other applications," in *Proc. Sel. Areas Cryptograph. (SAC)*, 2011, pp. 320–337.

[12] S. J. Johnson, *Introducing Low-Density Parity-Check Codes*, vol. 1. Callaghan, NSW, Australia: Univ. Newcastle, 2006.

[13] M. M. Kermani and A. Reyhani-Masoleh, "Reliable hardware architectures for the third-round SHA-3 finalist Grostl benchmarked on FPGA platform," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (DFT)*, 2011, pp. 325–331.

[14] S. Subramanian, M. Mozaffari-Kermani, R. Azarderakhsh, and M. Nojoumian, "Reliable hardware architectures for cryptographic block ciphers LED and HIGHT," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1750–1758, Oct. 2017.

[15] M. Mozaffari-Kermani, R. Azarderakhsh, A. Sarker, and A. Jalali, "Efficient and reliable error detection architectures of Hash-Counter-Hash tweakable enciphering schemes," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 2, pp. 1–54, 2018.

[16] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security analysis of logic encryption against the most effective side-channel attack: DPA," in *Proc. Defect Fault Tolerance VLSI Syst.*, 2015, pp. 97–102.

[17] B. Liu and R. Sandhu, "Fingerprint-based detection and diagnosis of malicious programs in hardware," *IEEE Trans. Rel.*, vol. 64, no. 3, pp. 1068–1077, Sep. 2015.

[18] 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. 2010.

[19] K. N. Dang and X.-T. Tran, "Parity-based ECC and mechanism for detecting and correcting soft errors in on-chip communication," in *Proc. IEEE Int. Symp. Embedded Multicore/Many-core Syst. Chip (MCSoC)*, 2018, pp. 154–161.

[20] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "A parity code based fault detection for an implementation of the advanced encryption standard," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2002, pp. 51–59.

[21] P. Ahir, M. Mozaffari Kermani, and R. Azarderakhsh, "Lightweight architectures for reliable and fault detection Simon and Speck cryptographic algorithms on FPGA," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 4, pp. 1–17, 2017.

[22] 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.