# Lightweight Architectures for Reliable and Fault Detection Simon and Speck Cryptographic Algorithms on FPGA

PRASHANT AHIR and MEHRAN MOZAFFARI-KERMANI, Rochester Institute of Technology
REZA AZARDERAKHSH, Florida Atlantic University

The widespread use of sensitive and constrained applications necessitates lightweight (low-power and low-area) algorithms developed for constrained nano-devices. However, nearly all of such algorithms are optimized for platform-based performance and may not be useful for diverse and flexible applications. The National Security Agency (NSA) has proposed two relatively recent families of lightweight ciphers, that is, Simon and Speck, designed as efficient ciphers on both hardware and software platforms. This article proposes concurrent error detection schemes to provide reliable architectures for these two families of lightweight block ciphers. The research work on analyzing the reliability of these algorithms and providing fault diagnosis approaches has not been undertaken to date to the best of our knowledge. The main aim of the proposed reliable architectures is to provide high error coverage while maintaining acceptable area and power consumption overheads. To achieve this, we propose a variant of recomputing with encoded operands. These low-complexity schemes are suited for low-resource applications such as sensitive, constrained implantable and wearable medical devices. We perform fault simulations for the proposed architectures by developing a fault model framework. The architectures are simulated and analyzed on recent field-programmable grate array (FPGA) platforms, and it is shown that the proposed schemes provide high error coverage. The proposed low-complexity concurrent error detection schemes are a step forward toward more reliable architectures for Simon and Speck algorithms in lightweight, secure applications.

CCS Concepts: • **Hardware** → *Very large scale integration design*; *Robustness;*

Additional Key Words and Phrases: Field-programmable gate array (FPGA), low complexity, reliability, Simon, Speck

**ACM Reference Format:**
Prashant Ahir, Mehran Mozaffari-Kermani, and Reza Azarderakhsh. 2017. Lightweight architectures for reliable and fault detection Simon and Speck cryptographic algorithms on FPGA. ACM Trans. Embed. Comput. Syst. 16, 4, Article 109 (May 2017), 17 pages.
DOI: http://dx.doi.org/10.1145/3055514

This material is based on work supported by the U.S. Army Research Laboratory under award no. W911NF-16-1-0204-(68023-CS). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon. This work has been also partly supported by the U.S. federal agency award 60NANB16D245 granted from the U.S. Department of Commerce, National Institute of Standards and Technology (NIST).
Authors' addresses: P. Ahir, Electrical and Microelectronic Department, Rochester Institute of Technology, Rochester, NY 14610; email: pa8238@rit.edu; M. Mozaffari-Kermani, Electrical and Microelectronic Department, Rochester Institute of Technology, Rochester, NY 14610; email: m.mozaffari@rit.edu; R. Azarderakhsh, Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431-0991; email: razarderakhsh@fau.edu.

## 1. INTRODUCTION

The need for lightweight cryptography has emerged due to the advancements of constrained devices, such as radio-frequency identification (RFID) tags, nano-sensor networks, and implantable and wearable medical devices. These utilize sensitive, low-power implementations over a very small chip area and consume a low amount of energy. The Advanced Encryption Standard (AES), the current symmetric-key cryptography standard, may not achieve the necessary constraints for area, power consumption, and energy, necessitating use of lightweight block ciphers. There have been prominent efforts to make the AES more compact; for example, a 128-bit AES was developed that expanded over an area of 2,400 gate equivalents [Moradi et al. 2011]. This has been a considerable reduction in area considering the AES algorithm. However, it is still a large overhead burden for highly constrained environments. Moreover, the AES cannot adapt to the varying level of security needed by different devices. Not all devices can spare area for 128-bit security. Consequently, it might waste chip area to encrypt 128-bit vectors where fewer bits need to be protected.

This motivation calls for lightweight security, and thus many lightweight block ciphers have been proposed to address these problems. However, some of these ciphers have been optimized for high performance on either hardware or software platforms. The ciphers KATAN and KTANTAN [Cannière et al. 2009] and PICCOLO [Shibutani et al. 2011] are all lightweight but are optimized to perform best on hardware platforms and might struggle to give good performance on software-based constrained devices. Similarly, for algorithms such as SEA [Standaert et al. 2006] and LED [Guo et al. 2011] ciphers, having small code size and memory make them more inclined toward software-based devices having a constrained instruction set.

Currently, the ISO 29192-2 standard specifies two lightweight block ciphers: CLEFIA, a 128-bit block cipher, and PRESENT, a 64-bit block cipher. CLEFIA could provide high security along with good hardware and software implementation capabilities. It had a proven highest hardware gate efficiency of 401 on 90nm technology [Beaulieu et al. 2015]. Moreover, it could perform on a wide range of processors at high speeds. Similarly, PRESENT has a compact design smaller than the AES. It was optimized for hardware implementations by using a single 4-bit S-box and had low power consumption and high chip efficiency.

The National Security Agency (NSA) has proposed two new lightweight block ciphers, Simon and Speck [Beaulieu et al. 2013], as alternatives to the aforementioned encryption systems being used for RFID tag readers. These ciphers have been submitted to ISO for inclusion in the ISO 29192-2 standard. They work better on small hardware devices, which have memory and processor constraints. In Beaulieu et al. [2015], application-specific integrated circuit (ASIC) implementation of Simon and Speck was performed on 90nm technology and had efficiencies of 2,130 and 1,307, respectively. They use simple nonlinear functions like AND and modular additions, which can be easily implemented on both hardware and software platforms, unlike PRESENT, which has been optimized only for hardware implementations. Moreover, Simon and Speck are families of ciphers, and each family has different ciphers based on the sizes of the blocks and encryption keys. This makes them flexible to be used with a wide variety of devices. This is our motivation for choosing Simon and Speck families of block ciphers above the other lightweight block ciphers.

In Biryukov et al. [2014] and Sun et al. [2014], these ciphers have been analyzed by attacking some of the rounds, and it is concluded that the ciphers provide acceptable security. Differential fault analysis (DFA) of these ciphers has been carried out in Tupsamudre et al. [2014]. The work has exploited the data leaking due to the AND operation in Simon to get the last round key. Similarly, in Speck, the modular addition has been proved to be the weak link giving out information to obtain the key. A proper fault detection technique needs to be in place to detect such cases and then respond to it by shutting down the device or deleting the secret key.

Concurrent error detection (CED) techniques have been widely used to architect reliable hardware for the AES and other cryptographic algorithms [Yen et al. 2006; Di Natale et al. 2009; Mozaffari-Kermani et al. 2010; Mozaffari-Kermani et al. 2013; Maistri et al. 2008; Guo et al. 2015; Yasin et al. 2015; Karaklajic et al. 2013; Mozaffari-Kermani et al. 2016; Mozaffari-Kermani et al. 2014; Bayat-Sarmadi et al. 2010; Mozaffari-Kermani et al. 2015; Mozaffari-Kermani et al. 2008]. It is well known that concurrent error detection techniques include a number of schemes, that is, hardware/information/time/hybrid redundancy. Hardware redundancy makes use of extra hardware to process the same input twice to match the two outputs; any mismatch will trigger the error flag. Information redundancy schemes have a number of variants, for example, parity codes [Karri et al. 2003] and robust codes [Karpovsky et al. 2004]. The time redundancy technique has a number of schemes, that is, recomputing with shifted operands (RESO) [Wu et al. 2006; Patel 1982], recomputing with rotated operands (RERO) [Li et al. 1992], and recomputing with permuted operands (REPO) [Guo et al. 2013]. The hybrid redundancy scheme is given in Karri et al. [2002], Satoh et al. [2008], and Rajendaran et al. [2010], where different improvements in the architecture have been proposed. The choice of the CED technique is completely dependent on the requirements in terms of overhead tolerance, security, and reliability.

In this article, motivated by the lightweight constructions of Simon and Speck, we propose CED schemes that have acceptable area and power overheads instead of being a burden for such constructions. To the best of our knowledge, research on developing reliable architectures for Simon and Speck have not been reported to date.

Our contributions in this article are summarized as follows:

—We use time redundancy concurrent error detection techniques and propose reliable hardware architectures for both Simon and Speck block ciphers. These schemes add acceptable overhead to the original designs, maintaining the lightweight property of the crypto-architectures.
—The proposed architectures are benchmarked for the ability to detect transient and permanent faults by performing fault injection simulations. The results of our error simulations show high error coverage for both of these block ciphers. The proposed fault detection schemes give high error coverage for Simon and Speck.
—Finally, we implement the architectures on two FPGA families to compare the performance and implementation metrics with the original Simon and Speck designs. The results show that the proposed designs have acceptable overheads with very high error coverage. The area, delay, and throughput overheads are acceptable for these two ciphers.

The rest of the article is organized as follows: In Section 2, preliminaries for Simon and Speck are provided. Section 3 is used as a motivating section to give details regarding various CED techniques and their shortfalls. Moreover, it presents our proposed design for reliable architectures. In Section 4, the fault injection simulations are performed to determine the error detection capabilities of the proposed architectures, and we benchmark our presented work by implementing our designs on FPGA. Finally, we present our conclusions in Section 5.

## 2. PRELIMINARIES

We present a brief description of Simon and Speck in what follows.

### 2.1. Simon

The Simon family has block ciphers for 10 distinct block and key sizes, which are generally written as Simon $2n/mn$ for a $2n$-bit block and $m$-word ($mn$-bit) key. For example, if the block size is 48 bits, then $n = 24$. If the word size is $m = 4$, then key is $m \times n = 4 \times 24 = 96$ bits, that is, $mn$ bits. The different sizes make the algorithm useful for a wide variety of constrained devices with different levels of security.

The round function is repeated to obtain a cipher-text and is a Feistel Map having two stages as follows: $R_k(x, y) = (y \oplus f(x) \oplus k, x)$, where $f(x) = (S_x.S_x^8) \oplus S_x^2$, and $k$ is the round key given by the key schedule. In this process, $\oplus$ denotes XOR, and for a given $j$, $S^j$ is the left circular shift (the nonlinearity is achieved here by rotating the same input by a different number of bits and then performing their AND operation).

### 2.2. Speck

The Speck family is represented, similar to Simon, as Speck 2n/mn. The round function is $R_k(x, y) = ((S_x^{-\alpha} + y) \oplus k, S_y^{\beta} \oplus (S_x^{-\alpha} + y) \oplus k)$. Here, if the block size is 32, then inputs are rotated by amounts $\alpha = 7$ and $\beta = 2$ (similarly, $\alpha = 8$ and $\beta = 3$ for others). The nonlinearity is obtained by using the modular addition, which favors a software platform over hardware.

There is always a bargain between efficiency and security depending on the application requirements. It is very difficult to achieve both at the same time. To obtain a high level of security, a very strong algorithm with large key is needed, but this increases the hardware overhead. Conversely, if efficiency is important, then we use a simple algorithm with a small key and run a large number of rounds. This would not have large hardware overhead, but the security obtained would not be very high. The Simon and Speck families with different key sizes for different block sizes attempt to give fairly good security, keeping hardware overhead to a low amount, nonetheless giving good efficiency.

## 3. PROPOSED RELIABLE SIMON AND SPECK

In this section, we present the motivations for our technique and also discuss the shortfalls and problems encountered by different CED techniques. Then, we present our proposed CED schemes for Simon and Speck.

### 3.1. Motivations

In this subsection and as motivations to our proposed work, we briefly present different CED techniques and some possible shortcomings with respect to lightweight applications. Full hardware redundancy techniques (e.g., partial or complete duplication) give good fault detection architectures; however, this is at the cost of high overhead. Therefore, such schemes cannot be used for lightweight algorithms.

*3.1.1. Signature-Based Diagnosis Approach.* The registers in the datapath are the key elements to propagate the errors. Hence, it is imperative that we detect the presence of faults in the datapath registers. Signatures (e.g., interleaved or single/multiple parity bits) can be efficiently used to represent the data held by the registers. As a case study, a parity-based CED scheme for Simon and Speck has been described in Figures 1 and 2. The general approach is to calculate the value for the parity bit based on the individual bits held by the register and then compare it by taking an XOR with the predicted parity bit value; then, any discrepancies witnessed raise the error indication flags.

The main disadvantage of the parity scheme is that the error coverage is almost only 50%. This is due to the fact that only odd number of faults can get detected with this method.

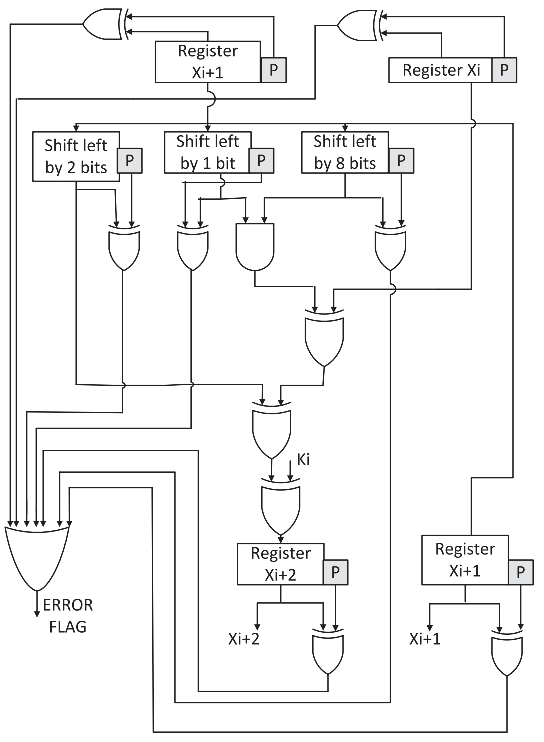

Fig. 1. Proposed signature-based CED scheme for Simon.

The Speck algorithm employs a modular adder in one of the steps to arbitrate the plaintext. To perform this addition, Figure 3 shows a 4-bit self-checking adder. It uses two 4-bit full adders to calculate addition results with input carry “0” and “1.” Then, according to actual input carry, the final output carry bit is selected. The self-checking action is performed by the two-pair two-rail checker as explained in Vasudevan et al. [2007] and Akbar et al. [2014]. However, as Speck and Simon are lightweight and are used in constrained applications, the aforementioned approach may not be suitable for error detection in our architecture.

*3.1.2. Robust Protection Scheme.* Karpovsky et al. [2004] have proposed a robust protection scheme against DFA attacks. It is based on using nonlinear robust error-detecting codes with input as well as the computed output. The proposed design employs a counter to count the number of faults encountered by the device in its life-time, and once it reaches a predecided threshold value, the secret key is cleared by the device since it is assumed that, typically, it encounters a lower number of natural faults than those required by a practical DFA.

In this scheme, nonlinear codes are obtained using a cubic function. As shown in Figures 4 and 5, applying such methods to Simon and Speck can be considered; that is, two cubic functions are used at the input and output of each round function. The cubic function selection is based on the fact that it gives the best error coverage without requiring complicated hardware. A square function does not give a good error coverage, and the functions having powers higher than three result in much complicated hardware. Thus, the cubic function is a tradeoff between error coverage and complexity.

In these figures, the linear predictor generates a signature that is equivalent to the component-wise XOR of the output bytes of a round based on the block size. This

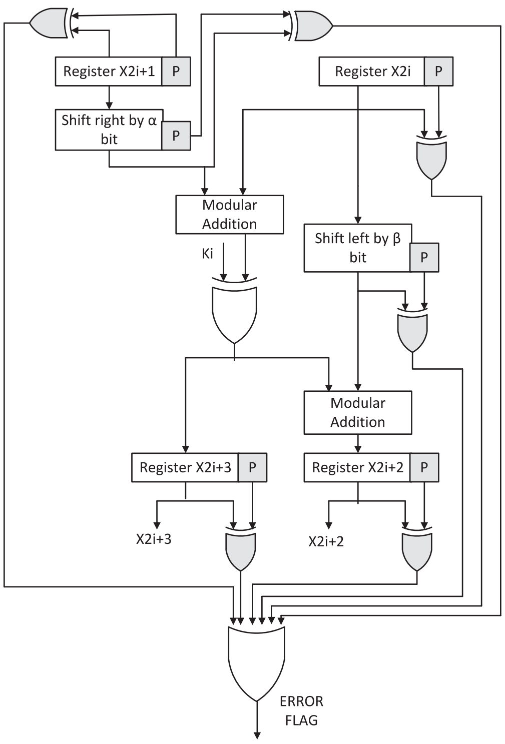

Fig. 2. Proposed signature-based CED scheme for Speck.

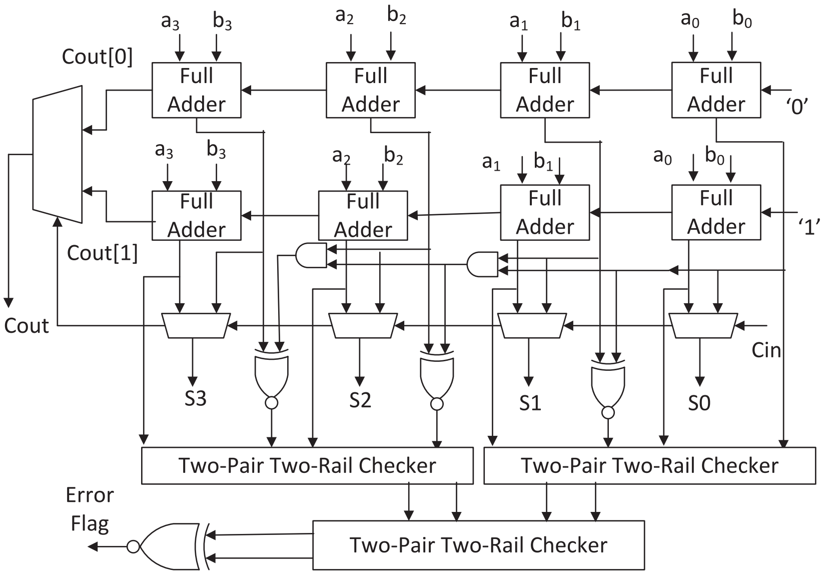
$a_3$ $b_3$ $a_2$ $b_2$ $a_1$ $b_1$ $a_0$ $b_0$
Cout[0]
Full Adder
'0'
'1'
Cout[1]
Cout
Cin
S3
S2
S1
S0
Two-Pair Two-Rail Checker
Two-Pair Two-Rail Checker
Two-Pair Two-Rail Checker
Error Flag

Fig. 3. Adopted self-checking adder used for modular addition of Speck.

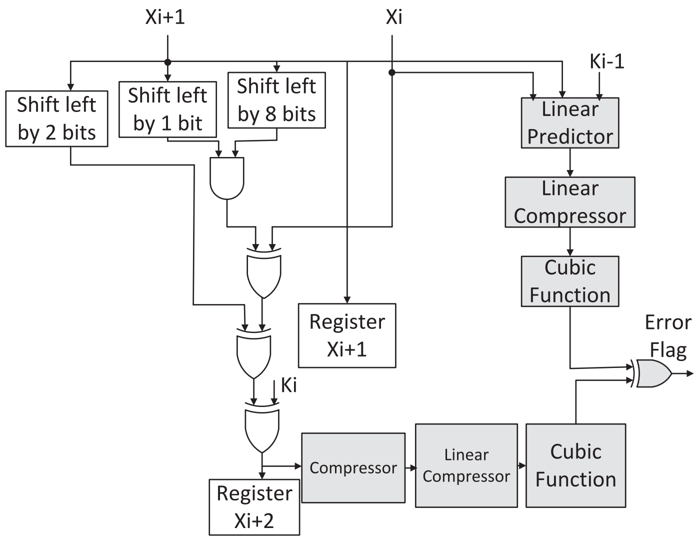

Fig. 4. Applicable robust protection scheme for Simon.

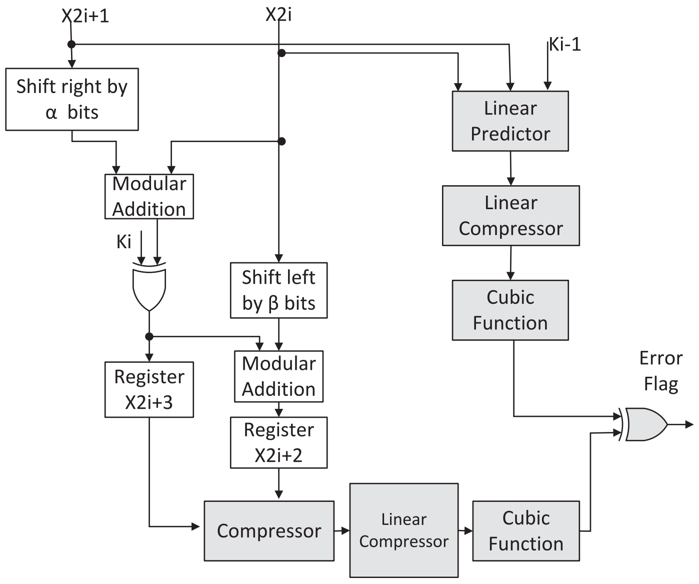

Fig. 5. Robust error detection scheme for Speck.

signature is then passed on to the cubic function. In cases where the size of the cubic function is less than that of the linear predictor, a compressor is used to compress the size of the predictor output so that it matches the size of the cubic function input. The cubic function with signature $r$ is dependent on primitive polynomial. For a 24-bit input, the signature of the cubic function can be chosen to be less than or greater than 24. The compressor is needed if $r$ is greater than 24. In order to reduce the complexity by not using the compressor, the primitive polynomial can be, for instance, $x^{20} + x^{17} + 1$ or $x^{16} + x^5 + x^3 + x^2 + 1$ for $r = 20$ or 16, respectively. The compressor shown in the design is to illustrate a generalized architecture incorporating all components of a robust scheme. This scheme provides protection for the encryptor and decryptor, as well as the key generation algorithm.

This method gives 100% error coverage; however, the hardware overhead is almost 50%, which may not be acceptable considering the lightweight applications of Simon and Speck. Thus, this scheme may not be ideal to be used for protection of Simon and Speck.

### 3.2. Proposed Error Detection Schemes

So far, we have explained problems with usage of various fault diagnosis schemes, such as higher overheads in case of hardware redundancy and robust codes or lower error detection rate in case of parity schemes. Therefore, we select a protection scheme that will provide close to 100% error coverage at suitable area and power overheads. The proposed scheme, as explained in the following sections, has a high error detection rate at acceptable performance metric overheads.

In this section, we propose concurrent error detection schemes that are applicable to both Simon and Speck.

In addition to the schemes used in this article, the RESO approach can also be used for error detection. In RESO-$k$, in the recomputation step, the inputs are shifted left by $k$ bits. Now, usually the leftmost $k$ bits, on shifting, will get lost. If we are to store them, we will need to house an $n + k$ bit register. This will, in turn, create needs for all the subsequent registers and computations to be of $n + k$ bit length; that is, the adders and datapath registers will be of $n + k$ bits. Due to this, the recomputation step will take more cycles to produce the output. This latency will only increase with higher values of $k$. To house the increased size, more chip area will be consumed, resulting in increased complexity. Due to these drawbacks, we do not propose RESO as a comparatively good error detection approach.

For the sake of brevity, we discuss only the error detection of the encryption operation. We note that the decryption can be protected through the proposed approaches as well.

We propose RERO for both Simon and Speck. Based on the methods of processing the data, we propose two types of architectures, that is, iterative and pipelined architectures. A Simon block cipher having a $2n$-bit block of plaintext made up of two $n$-bit words $X_{i+1}$ and $X_i$ is passed as input, as shown in Figure 6. Each of the input blocks (plaintext blocks) is operated upon twice. A multiplexer controls the passage of the normal and recomputed plaintext. During the first run, the operands are passed in their normal state. As can be seen in this figure, the Feistel stepping of the Simon round function operates on the plaintext. The output generated is stored in a register for a later comparison. During the second run, the multiplexer selects the recomputed operand to be passed on to Simon. The recomputed plaintext is obtained by rotating the input by a constant value of $a$ bits. Each word of the input block-cipher is rotated by same amount of $a$ bits toward right or left. Similarly, the key $K_i$ is also recomputed by rotating it by same amount in the same direction as the plaintext. The Feistel stepping function's output is the recomputed output. This output is then rotated in an inverse direction by $a$ bits. The output thus obtained is compared with the output calculated originally in the first run. These two are then XORed to check their equality and the error indication flag is raised if they are not equal.

Let us take the example of Simon48/96. Each of the 24-bit words of the input is transformed to $[Xi + 1_{23}\ ..\ Xi + 1_{j+1}\ Xi + 1_j\ ..\ Xi + 1_0]$ and $[Xi_{23}\ ..\ Xi_{j+1}\ Xi_j\ ..\ Xi_0]$. During the second run, we rotate left by $j$ bits ($j$ is an integer such that $j = 0$ to 23), which makes the input as $[Xi+1_j\ ..Xi+1_0\,Xi+1_{23}\ ..Xi+1_{j+1}]$ and $[Xi_j\ ..Xi_0\,Xi_{23}\ ..Xi_{j+1}]$. The outputs, $Xi + 2$ and $Xi + 1$, are then fed back as inputs to the next round of the function. Thus, we *iterate* the input through the round function repeatedly to get a final secure ciphertext. A multiplexer selects between the main plaintext and the ciphertext

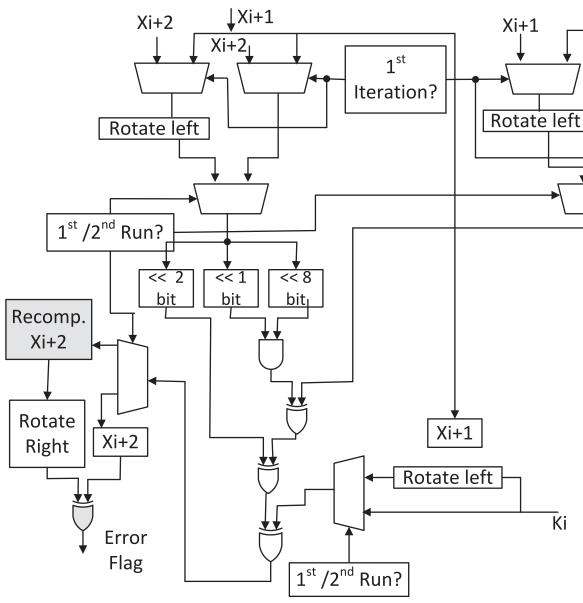

Fig. 6. Proposed error detection approach for Simon.

generated by the previous round. Each cipher family is iterated through a predecided number of times. Simon48/96 is run through the round function 36 times to get the final output.

For the Speck algorithm, as shown in Figure 7, a similar methodology is followed where we compare recomputed and original outputs. Inequalities will raise the error indication flags.

An important component in Speck is the modular adder. We use the self-checking adder and modify it to make it work on rotated operands for Speck48/96.

Consider Figure 8, which shows the proposed modified self-checking adder. The main reason behind modifying the normal self-checking adder is to ensure that the carry generated by addition of $b_{23}$ and $a_{23}$ does not affect the $z_0$ bit after rotation, and after rotation, correct carry goes into addition of bits $b_{j+1}$ and $a_{j+1}$.

During the first run, the input operands are appended with bit-@ (a stuck-at-0 bit) at the most significant bit position of both the operands. Therefore, an $(n + 1)$-bit adder is needed to operate on these operands. The effect of this bit-@ is such that no matter what carry-out actually gets generated by the bit-$(n - 1)$, the bit-@ will always be “0.” Now, during the second run, after rotating the input, we ensure that the adder operands have bit-@ between bit-0 and bit-23. The addition of the bits-23 ($b_{23}$ and $a_{23}$) will generate a carry-out that does not affect the bits-0 ($b_0$ and $a_0$) addition result due to the presence of bit-@ between them. This enables a correct addition result before and after the rotation. Moreover, as can be seen in the figure, the carry-out generated by the last bits is connected as a carry-in to the first bits. This is again to ensure correct addition of the bits $j + 1$ and $j$ during both runs. At the output $z$, the bit-@ is removed from the result; that is, for the first run, let $p$ be the output $[@\, b_{23} \ldots b_{j+1}\, b_j \ldots b_0] + [@\, a_{23} \ldots a_{j+1}\, a_j \ldots a_0] = [p_{23} \ldots p_{j+1}\, p_j \ldots p_0]$, and for the second run, let $q$ be the output $[b_j \ldots b_0\, @\, b_{23} \ldots b_{j+1}] + [a_j \ldots a_0\, @\, a_{23} \ldots a_{j+1}] = [q_{23} \ldots q_{j+1}\, q_j \ldots q_0]$.

In this iterative approach, we let the entire input pass through the hardware before passing the next input. This reduces the throughput since hardware is not being used

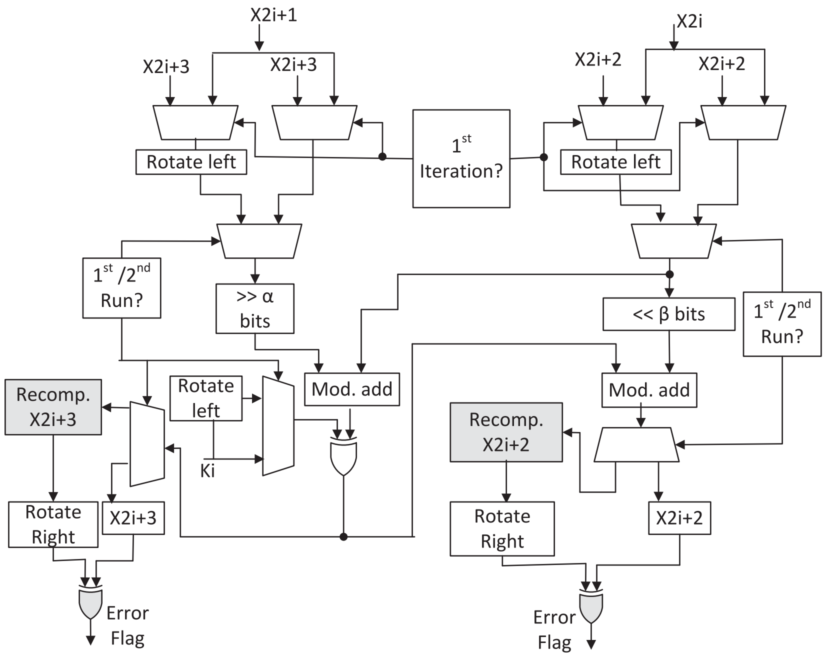

Fig. 7. Proposed error detection approach for the Speck algorithm.

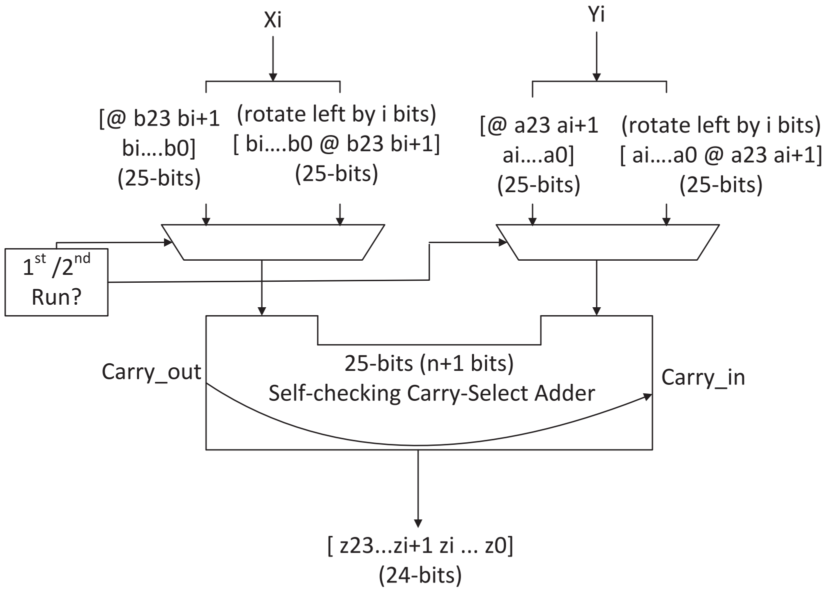
Xi
Yi
[@ b23 bi+1 bi….b0] (25-bits)
(rotate left by i bits) [ bi….b0 @ b23 bi+1] (25-bits)
[@ a23 ai+1 ai….a0] (25-bits)
(rotate left by i bits) [ ai….a0 @ a23 ai+1] (25-bits)
1st /2nd Run?
Carry_out
25-bits (n+1 bits) Self-checking Carry-Select Adder
Carry_in
[ z23…zi+1 zi … z0] (24-bits)

Fig. 8. Modular adder operation for the Speck algorithm in the proposed scheme.

at its fullest and it takes more cycles to run the input through a single round. We, alternatively, propose a pipelined architecture to improve such throughput degradations. Subpipelining can be performed to alleviate this problem. Suppose $n$ pipeline registers have been placed to subpipeline the structures to break the timing path to approximately equal segments. Let us denote the segments of pipelined stages by $\exists_n$. The original input is first applied to the architecture, and in the second cycle, while the second segment of the circuit executes the first input, the second input or the encoded variant of the first input is fed to the first half of the circuit (this depends on the objectives, i.e., reliability vs. getting the results first). This trend is consecutively executed for the normal and rotated operands. Such approach ensures that lower degradation in throughput at the expense of more area overhead is achieved.

## 4. ERROR INJECTION SIMULATIONS AND IMPLEMENTATIONS THROUGH FPGA

In what follows, we present the results of our error simulations and FPGA implementations benchmark.

### 4.1. Error Simulations

The proposed fault detection architectures have been simulated after injecting faults. The proposed architectures have the capability of detecting both permanent and transient faults (this covers both natural and malicious faults). We have performed the transient fault injection simulation in two stages. First, we have injected the faults in just the first round and observed the efficiency. Then, we have injected the faults in just the second round. The approach that has been followed for the proposed fault diagnosis schemes is to inject faults and then observe the error indication flags. For simulations, Verilog HDL has been used. We have considered all the subblocks of the original architecture, that is, the adders, XOR, AND, and OR gates, to induce faults by flipping one or more bits and then inspecting the generated outputs. We have considered a particular fault scenario and applied different inputs to assert a subset of entries while injecting faults. We then observe all the errors that get detected for all the inputs. The fault model used to test the proposed architectures is created using an external feedback linear-feedback shift register (LFSR) to generate pseudo-random fault vectors that can flip random bits in the output of the gates and at random intervals. The LFSRs used here are 8-bit registers with the polynomial $x^8 + 1$ for maximum taps. This is achieved using multiplexers whose select signal is driven using an LFSR, thus randomizing the selection of faulty bits (coming from another LFSR) and correct bits, that is, the actual results. We have employed an LFSR to randomize the position and value of the injected fault for both the intermediate and pipeline registers.

As discussed in the previous sections, for the RERO scheme, we pass normal input in the first round and then pass the rotated one in the second round. Thus, each of Simon and Speck requires two runs to detect the presence of faults. The Simon block cipher has a combination of AND and XOR gates. We select 4 random bits from each of these gates and inject faults in them. The Speck block cipher has two modular adders in its architecture as well. A similar approach is followed to induce faults here; that is, we select any 4 bits from each of the adders and the gates and flip them using the LFSRs. Thus, a total of 12 different faults are induced in each Simon and Speck. In addition to this multiple random fault model and to assess other potential scenarios, we also test our architecture for 2/3/4-bit fault models. Overall, 100,000 faults are injected in each cipher and the error indication flag is observed. A counter is set to count the number of faults detected. We have also tested our proposed architectures using the single-bit fault model (as the most typical natural fault model) and the single-byte fault model (as the most practical fault model for fault attacks). For both of these cases, we get 100% error coverage. It is observed that for Simon and Speck we get very close

Table I. Zynq-7000 FPGA Implementation Results for Simon Block Cipher

| Metric | Simon | Simon-RERO | Overhead |
|---|---|---|---|
| Power (W) | 0.239 | ∼0.239 | Negligible |
| Delay (ns) | 5.448 | 5.607 | 2.919% |
| Area | 73 | 95 | 30.137% |
| Throughput (Gbps) | 0.245 | (0.238)[1] | (2.836%) |

[1]One stage subpipelined.

Table II. Virtex-7 FPGA Implementation Results for Simon Block Cipher

| Metric | Simon | Simon-RERO | Overhead |
|---|---|---|---|
| Power (W) | 0.248 | ∼0.248 | Negligible |
| Delay (ns) | 4.415 | 4.562 | 3.330% |
| Area | 73 | 95 | 30.137% |
| Throughput (Gbps) | 0.302 | (0.292)[1] | (3.4%) |

[1]One stage subpipelined.

to 100% error coverage for multiple faults. The reason that we cannot detect a very small percentage of faults is that we inject faults in the original architecture and the comparison XOR gate (final XOR gate, which compares the first/second round results) at the same time (this is in analogy with predicted/actual signature comparisons). Such additional circuitry can be hardened, for example, through fault tolerance techniques such as triple modular redundancy.

Next, we describe the results obtained after implementing our proposed architectures on Xilinx FPGA families [Xilinx 2017].

## 4.2. Implementations on FPGAs

This section presents the overhead incurred while applying the proposed error detection schemes on FPGA platforms. We would like to emphasize that the presented results are independent of the platform or FPGA family, and similar results are expected on other hardware platforms. The implementations on FPGAs have been performed on the Xilinx Zynq-7000 family (xc7k70tfbg484-1Q) and Xilinx Virtex-7 (xc7k70tfbg484-1Q) using the Xilinx Vivado 2014.4 Design Suite. In order to get the overheads, we compare the implementation results obtained from the original Simon and Speck architecture with the proposed error detection architectures.

The implementations have been performed for Simon48/96 and Speck48/96 block ciphers. The Simon cipher has to make 36 runs to give a final ciphertext. Similarly, for Speck, it has to be run 23 times. According to the RERO approach, during each run, the input needs to be passed for two rounds in order to detect an error. This degrades the overall throughput. Nevertheless, we can alleviate this as discussed before using subpipelining.

Each of the two ciphers has a control unit that directs the passage of normal/rotated operands to the main block cipher module. The control unit then receives the normal/recomputed outputs at the end of each round and sets/resets the error indication flags.

The overhead calculations are shown in Tables I to III.

The results are in conformity with our expectations for lightweight applications. The original Simon architecture, being made up of combinational logic, has a small slice area occupancy. The XOR and OR gates, responsible for the setting of the error indication flag, occupy a considerable number of LUTs and hence the area overhead

Table III. Xilinx Zynq-7000 FPGA Implementation for Speck Block Cipher

| Metric | Speck | Overhead RERO | Overhead Parity | Overhead RESO |
|---|---|---|---|---|
| Power (W) | 0.251 | Negligible | 0.75% | 1.99% |
| Delay (ns) | 2.445 | 10.18% | 20.80% | 18.80% |
| Area | 471 | 12.10% | 14.44% | 12.95% |
| Throughput (Gbps) | 0.854 | (9.24%)[1] | 16.20% | 15.92% |

[1]One stage subpipelined.

Table IV. Utilization of the Cubic Function for Different Signature Sizes [Karpovsky et al. 2004]

| Size of the Cubic Signature | Primitive Polynomial | Cube Size (Slices) |
|---|---|---|
| 8 | $x^8 + x^4 + x^3 + x + 1$ | 28 |
| 16 | $x^{16} + x^5 + x^3 + x^2 + 1$ | 150 |
| 20 | $x^{20} + x^{17} + 1$ | 202 |
| 24 | $x^{24} + x^7 + x^2 + x + 1$ | 368 |
| 28 | $x^{28} + x^3 + 1$ | 349 |
| 29 | $x^{29} + x^2 + 1$ | 359 |
| 31 | $x^{31} + x^3 + 1$ | 452 |
| 32 | $x^{32} + x^{22} + x^2 + x + 1$ | 747 |

goes to roughly 30%. This can be seen as a tradeoff for this scheme, but considering that other viable error detection schemes consume more area, we can consider that this is an acceptable area overhead that is always incurred if the block cipher is to be given close to 100% error coverage. The implementation, mapping, and placement settings are all the same for all the architectures in Vivado to have a fair comparison.

We also present the results for two other cases for Speck as examples, that is, the results of the approach with no subpipelines and the results with two stages of subpipelines. In the former case, where there is no intermediate subpipeline architecture, the area overhead is lower at the expense of higher delay and degradation in throughput. Moreover, in the latter case, for the two-stage subpipelined structure, we get higher area but lower critical path delay and higher frequency, with lower degradation in throughput. For the former case, and for Speck, the area would be 515 slices (9.3% overhead, which is lower than the one-stage subpipelined variant), and the power consumption and throughput overheads are 0.28% and around 100%, respectively, where the latter is not practically acceptable. For the latter case, we get the area overhead of 14.4%, which is higher than what we get in Table III, but that would alleviate the throughput degradation compared to that of Table III (here, the power consumption overhead is still negligible but the throughput degradation is 4%). We note that the choice is up to the designers, and none of these, without having an objective in mind in terms of overhead tolerance and the required error coverage, can be treated as an optimal design.

For comparisons, as a case study, we have implemented the signature-based odd parity scheme for our Speck cipher (refer to Table III). The disadvantage of such a scheme is that it can only detect an odd number of fault injections. Thus, this reduces the fault detection efficiency to just 50%. Contrary to this scheme, the proposed fault detection architecture has a 100% error coverage to detect both odd and even faults.

In the robust protection scheme, as shown in Figures 4 and 5, a predictor and cubic function are used in the extended network to convert the input plaintext and key into a signature. The cubic function is used in this extended network as well as the error detection network at the output. The cubic function for a given signature size "r" is based on a primitive polynomial. Table IV [Karpovsky et al. 2004] gives a list of primitive polynomials based on the signature values and their corresponding utilization on a Xilinx Virtex-E FPGA.

For a 48-bit block cipher, the round input will be 24 bits long. Thus, we can select a primitive polynomial with signature $r \leq 24$. As can be seen in Table IV, the primitive polynomial with $r = 24$ utilizes around 368 slices. It is possible to use a lower signature polynomial to reduce the slice utilization, say, $r = 20$ or $r = 16$; however, in that case, the 24-bit output from the predictor will have to be compressed using a linear compressor adding to the slice utilization.

A relative comparison of the overhead incurred by just the cubic function on Simon and Speck block ciphers shows that with 30 and 111 slices for Simon and Speck, respectively, a 24-bit cube size in extended network and error detection output is 368, which shows very high overhead, not suitable for lightweight applications. As can be seen, the overhead estimates are high for a lightweight cipher compared to the overheads incurred by the proposed architectures.

We have also implemented the RESO-2 fault detection scheme for the Speck block cipher for one FPGA family. The metrics are given in Table III. As can be seen, the overheads are within the acceptable limits with 100% error coverage. The Speck cipher involves circular shift operations on the plaintext. In the RESO approach, we increase the size of registers to accommodate the sifted bits and, hence, append zeros at the MSB positions of the plaintext registers. As a result, the ciphertext obtained is different than the one encrypted using normal-size registers. In order to get the correct ciphertext, additional logic will need to be employed, which will prove detrimental to the overhead statistics.

Dofe et al. [2015] utilize three methods to detect faults in Simon, among which the modular redundancy method and the reversed method have high overheads to be practical for lightweight applications. The third method in Dofe et al. [2015] through compensation logic has the area overhead of 65% as well. Mozaffari Kermani et al. [2015] present the fault diagnosis of a Pomaranch cipher. They used a bit-interleaved scheme for error detection. We compare the overheads of Pomaranch with the proposed scheme. The area and throughput overheads for Pomaranch are 21% and 12%, respectively. The proposed schemes have area and throughput overheads of 30% and 10% for Simon and 11% and 6% for Speck, respectively. Since the architecture of Pomaranch and the presented fault detection scheme are a lot different than the proposed method, the differences in the overheads are reasonably justified. The proposed fault detection methods can be applied to Pomaranch and other ciphers as well to obtain approximately closer overheads.

### 4.3. Differential Fault Analysis (DFA)

The proposed methods, being for reliability, can deal with permanent and transient faults. Even though the proposed methods make a potential DFA attack more difficult to mount, they may not completely thwart such attacks. In this section, we present previous DFA attacks on Simon and Speck families and make additional modifications to our proposed architecture to go toward making such attacks more difficult.

The work in Tupsamudre et al. [2014], Takahashi et al. [2015], and Vásquez et al. [2015] presents three DFA attacks on the Simon family. The authors used data leaked by an AND operation to deduce the secret key. In the case of Speck, the modular addition can be used by the attackers to gain knowledge of the secret key. After analyzing the block cipher, they concluded that injecting faults in each round will not help them get the secret key. Tupsamudre et al. [2014] and Takahashi et al. [2015] have demonstrated that by injecting a bit-flip fault at the input of the penultimate round (or ante penultimate round in the case of Vásquez et al. [2015]), they can deduce the value of at most 2 bits of the penultimate input. Thus, in turn, they can find out the value of the secret key used in the last round. The main difference in the three papers is the number of fault injections required to get all the bits of the secret key.

This DFA attack can potentially bypass the proposed RERO error detection scheme (please also refer to Barenghi et al. [2012]). Therefore, we make a small architectural addition to our proposed scheme in order to detect such types of DFA attacks. Since the fault injections are made at the input of a round, we compare the input subcipher in each round (starting from the second round) with that generated in the previous round. Any discrepancies will be indicated by the error indication flag. Should the attacker try to inject faults in the subcipher in the previous round itself, the previously proposed RERO scheme will detect such an attack. Thus, the RERO and the suggested addition should be able to protect Simon and Speck against permanent and transient faults and make the DFA attacks presented in Tupsamudre et al. [2014], Takahashi et al. [2015], and Vásquez et al. [2015] more difficult; however, we do not claim that it will be able to detect all types of DFA attacks. The method that we have employed involves comparing every round input with the previously generated round cipher-text and the overhead incurred is negligible. We have modified our architecture to protect against these DFA techniques. However, for new DFA attacks, the proposed architecture will have to be modified.

The signature-based diagnosis approach, which uses linear codes that can (always) detect random errors of small multiplicity (and can never detect some other errors), is diverse from an architecture based on robust codes that can detect (with probability) any error. These two solutions have two different goals; the first gives reliability and the second gives hardware security (against DFA).

Finally, note that according to Karpovsky et al. [2004], the linear compressor can make the code not robust anymore. Furthermore, this compressor is not required at all since cubic function can be designed for any vector length. In the context of its hardware overhead, there are high rate robust codes [Tomashevich et al. 2014] that have lower hardware complexity [Neumeier et al. 2015].

## 5. CONCLUSIONS

This article proposes reliable and efficient error detection architectures for the block ciphers Simon and Speck. The proposed schemes are optimized for low-area and low-power applications since Simon and Speck are among the lightweight block ciphers. We propose diagnosis approaches for inner subblocks of these ciphers and present an approach for alleviating the throughput overheads. The simulation results show that the proposed error detection schemes can detect close to 100% of the injected faults. We have also implemented our proposed architectures on the Xilinx Zynq-7000 FPGA family. The implementation results show that the power, area, and delay overheads incurred by the proposed architectures are acceptable. Therefore, the proposed architectures for Simon and Speck block ciphers can be reliably and efficiently used and further tailored by customizing the architectures based on the requirements in terms of reliability, security, and overhead tolerance.

## REFERENCES

M. A. Akbar and J. A. Lee. 2014. Comments on self-checking carry-select adder design based on two-rail encoding. *IEEE Trans. Circuits Syst. I*, 61, 7, 2212–2214.

R. Beaulieu, D. Shors, J. Smith, S. T. Clark, B. Weeks, and L. Wingers. 2015. Simon and Speck: Block ciphers for the internet of things. In *Proc. Cryptology ePrint Archive*, Report 2015/585.

R. Beaulieu, D. Shors, J. Smith, S. T. Clark, B. Weeks, and L. Wingers. 2013. The Simon and Speck families of block ciphers. In *Proc. Cryptology ePrint Archive*, Report 2013/404.

A. Biryukov, A. Roy, and V. Velichkov. 2014. Differential analysis of block ciphers Simon and Speck. In *Proc. Fast Software Encryption*, 546–570.

S. Bayat-Sarmadi, M. Mozaffari Kermani, and A. Reyhani-Masoleh. 2014. Efficient and concurrent reliable realization of the secure cryptographic SHA-3 algorithm. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 33, 7, 1105–1109.

A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. 2012. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proc. IEEE*, 100, 11, 3056–3076.

C. D. Cannière, O. Dunkelman, and M. Knezevic. 2009. KATAN & KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *Proc. Cryptographic Hardware and Embedded Systems*, 272–288.

J. Dofe, C. Reed, N. Zhang, and Q. Yu. 2015. Fault-tolerant methods for a new lightweight cipher Simon. In *Proc. Int. Symp. Quality Electronic Design*, 460–464.

G. Di Natale, M. Doulcier, M. L. Flottes, and B. Rouzeyre. 2009. A reliable architecture for parallel implementations of the advanced encryption standard. *J. Electron. Test. Theory Appl.*, 25, 4, 269–278.

J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. 2011. The LED block cipher. In *Proc. Cryptographic Hardware and Embedded Systems*, 326–341.

X. Guo and R. Karri. 2013. Recomputing with permuted operands: A concurrent error detection approach. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 32, 10, 1595–1608.

X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri. 2015. Security analysis of concurrent error detection against differential fault analysis. *J. Cryptographic Eng.* 5, 3, 153–169.

D. Karaklajic, J.-M. Schmidt, and I. Verbauwhede. 2013. Hardware designer's guide to fault attacks. *IEEE Trans. Very Large Scale Integration (VLSI) Syst.* 21, 12, 2295–2306.

R. Karri, G. Kuznetsov, and M. Goessel. 2003. Parity-based concurrent error detection of substitution-permutation network block ciphers. In *Proc. Cryptographic Hardware and Embedded Systems*, 113–124.

M. Karpovsky, K. J. Kulikowski, and A. Taubin. 2004. Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In *Proc. Dependable Systems and Networks*, 93–101.

R. Karri, K. Wu, P. Mishra, and Y. Kim. 2002. Concurrent error detection schemes of fault based side-channel cryptanalysis of symmetric block ciphers. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 21, 12, 1509–1517.

M. Karpovsky and A. Taubin. New class of nonlinear systematic error detecting codes. *IEEE Trans. Inf. Theory*, 50, 8, 1818–1819.

J. Li and E. E. Swartzlander. 1992. Concurrent error detection in ALUs by recomputing with rotated operands. In *Proc. Defect and Fault Tolerance in VLSI Systems*, 109–116.

M. Mozaffari-Kermani and A. Reyhani-Masoleh. 2010. Concurrent structure independent fault detection schemes for the advanced encryption standard. *IEEE Trans. Comput.*, 59, 5, 608–622.

M. Mozaffari-Kermani and R. Azarderakhsh. 2013. Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA. *IEEE Trans. Ind. Electron.*, 60, 12, 5925–5932.

P. Maistri and R. Leveugle. 2008. Double-data-rate computation as a countermeasure against fault analysis. *IEEE Trans. Comput.*, 57, 11, 1528–1539.

A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. 2011. Pushing the limits: A very compact and a threshold implementation of AES. In *Proc. Advances in Cryptology*, 69–88.

M. Mozaffari-Kermani, R. Azarderakhsh, and A. Aghaie. 2016. Fault detection architectures for post-quantum cryptographic stateless hash-based secure signatures benchmarked on ASIC. *ACM Trans. Embedded Comput. Syst.*, 16, 2, 59:1–19.

M. Mozaffari-Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi. 2014. Fault-resilient lightweight cryptographic block ciphers for secure embedded systems. *IEEE Embedded Syst.*, 6, 4, 89–92.

M. Mozaffari-Kermani and R. Azarderakhsh. 2015. Reliable hash trees for post-quantum stateless cryptographic hash-based signatures. In *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT'15)*, 103–108.

M. Mozaffari-Kermani and A. Reyhani-Masoleh. 2008. A lightweight concurrent fault detection scheme for the AES S-boxes using normal basis. In *Proc. LNCS Cryptographic Hardware and Embedded Systems (CHES'08)*, 113–129.

M. Mozaffari-Kermani, R. Azarderakhsh, and A. Aghaie. 2015. Reliable and error detection architectures of Pomaranch for false-alarm-sensitive cryptographic applications. *IEEE Trans. VLSI Syst.* 23, 12, 2804–2812.

Y. Neumeier, Y. Pesso, and O. Keren. 2015. Efficient implementation of punctured parallel finite field multipliers. *IEEE Trans. Circuits Syst. I: Regular Papers*, 62, 9, 2260–2267.

J. H. Patel and L. Y. Fung. 1982. Concurrent error detection in ALUs by recomputing with shifted operands. *IEEE Trans. Comput.*, C-31, 7, 589–595.

J. Rajendran, H. Borad, S. Mantravadi, and R. Karri. 2010. SLICED: Slide based concurrent error detection technique for symmetric block cipher. In *Proc. Hardware-Oriented Security and Trust (HOST'10)*, 70–75.

F. X. Standaert, G. Piret, N. Gershenfeld, and J. J. Quisquater. 2006. SEA: A scalable encryption algorithm for small embedded applications. In *Proc. Smart Card Research and Advanced Applications*, 222–236.

S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. 2014. Automatic security evaluation and (related-key) differential characteristic search: Application to Simon, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In *Proc. Advances in Cryptology*, 158–178.

A. Satoh, T. Sugawara, N. Homma, and T. Aoki. 2008. High-performance concurrent error detection scheme for AES hardware. In *Proc. Cryptographic Hardware and Embedded Systems (CHES'08)*, 100–112.

K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. 2011. Piccolo: An ultra-lightweight blockcipher. In *Proc. Cryptographic Hardware and Embedded Systems*, 342–357.

H. Tupsamudre, S. Bisht, and D. Mukhopadhyay. 2014. Differential fault analysis on the families of Simon and Speck ciphers. In *Proc. Fault Diagnosis and Tolerance in Cryptography*, 40–48.

J. Takahashi and T. Fukunaga. 2015. Fault analysis on Simon family of lightweight block ciphers. In *Proc. Information Security and Cryptology*, 175–189.

V. Tomashevich, Y. Neumeier, R. Kumar, O. Keren, and I. Polian. 2014. Protecting cryptographic hardware against malicious attacks by nonlinear robust codes. In *Proc. Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 40–45.

D. P. Vasudevan, P. K. Lala, and J. P. Parkerson. 2007. Self-checking carry select adder design based on two-rail encoding. *IEEE Trans. Circuits Syst. I*, 54, 12, 2696–2705.

J. C. G. Vasquez, F. Borges, R. Portugal, and P. Lara. 2015. An efficient one-bit model for differential fault analysis on Simon family. In *Proc. Fault Diagnosis and Tolerance in Cryptography*, 61–70.

K. Wu and R. Karri. 2006. Algorithm-level recomputing with shifted operands-a register transfer level concurrent error detection technique. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 25, 3, 413–422.

C. H. Yen and B. F. Wu. 2006. Simple error detection methods for hardware implementation of advanced encryption standard. *IEEE Trans. Comput.*, 55, 6, 720–731.

M. Yasin, B. Mazumdar, S. Subidh Ali, and O. Sinanoglu. 2015. Security analysis of logic encryption against the most effective side-channel attack: DPA. In *Proc. DFTS*, 97–102.

Xilinx FPGA Families. 2017. www.Xilinx.com.

Received July 2016; revised February 2017; accepted February 2017