Efficient and Concurrent Reliable Realization of the Secure Cryptographic SHA-3 Algorithm

Siavash Bayat-Sarmadi, *Member, IEEE,*
Mehran Mozaffari-Kermani, *Member, IEEE,*
and Arash Reyhani-Masoleh, *Member, IEEE*

***Abstract*—The secure hash algorithm (SHA)-3 has been selected in 2012 and will be used to provide security to any application which requires hashing, pseudo-random number generation, and integrity checking. This algorithm has been selected based on various benchmarks such as security, performance, and complexity. In this paper, in order to provide reliable architectures for this algorithm, an efficient concurrent error detection scheme for the selected SHA-3 algorithm, i.e., Keccak, is proposed. To the best of our knowledge, effective countermeasures for potential reliability issues in the hardware implementations of this algorithm have not been presented to date. In proposing the error detection approach, our aim is to have acceptable complexity and performance overheads while maintaining high error coverage. In this regard, we present a low-complexity recomputing with rotated operands-based scheme which is a step-forward toward reducing the hardware overhead of the proposed error detection approach. Moreover, we perform injection-based fault simulations and show that the error coverage of close to 100% is derived. Furthermore, we have designed the proposed scheme and through ASIC analysis, it is shown that acceptable complexity and performance overheads are reached. By utilizing the proposed high-performance concurrent error detection scheme, more reliable and robust hardware implementations for the newly-standardized SHA-3 are realized.**

***Index Terms*—Application-specific integrated circuit (ASIC), high performance, reliability, secure hash algorithm (SHA)-3, security.**

I. Introduction

The selection process for choosing a new secure cryptographic hash algorithm, i.e., secure hash algorithm (SHA)-3, was initiated by the National Institute of Standards and Technology (NIST) in 2007 to increase security and performance of hash functions. After three rounds of assessments, the winner algorithm for this standard has been chosen in 2012 as Keccak [1]. In this selection process, five SHA-3 finalists were assessed by the research community in terms of different aspects, such as software implementations, e.g., [2], hardware application-specific integrated circuit (ASIC) implementations [3]–[5], and field-programmable gate array (FPGA) implementations [5]–[7].

It is expected that Keccak, the winner of the SHA-3 competition, will provide confidentiality to various security-constrained applications such as the ones used for generating digital signatures and message authentication codes. For example, it can be utilized as an added integrity-checking measure for mobile *ad hoc* networks (MANETs) which lack physical layer security [8] or in conjunction with the efficient implementations for the point multiplication in elliptic curve cryptography (ECC). Integrity assurance of implantable and wearable medical devices, smart buildings, smart fabrics, Internet of nano-Things, and smart infrastructures as well as building automation systems, networked control systems, and wireless sensor networks can be provided by this recently-standardized secure hash algorithm as well [9].

Natural faults (defects) detection has been the center of many previous works [10]–[17]. Thus, reliability and fault immunity for the hardware implementations against natural defects need to be among the assessment aspects for the SHA-3 algorithm. To counteract the faults occurring in the cryptographic hardware systems, several error detection approaches have been presented to date, evaluating all of them is beyond the scope of this paper; refer to [18]–[20], for the error detection schemes for SHA-1 and SHA-2. Moreover, for the AES, several error detection schemes such as the ones presented in [21]–[23], aim at developing more robust hardware implementations for this algorithm. Furthermore, the authors in [15] and [24] proposed error detection methods for Grøstl, a SHA-3 finalist. It is also noted that error detection of the arithmetic operations utilized in the cryptographic hardware systems has gained attention in the literature [25]–[26]. However, to the best of our knowledge, mechanisms for making the selected SHA-3 algorithm immune against faults have not been reported to date.

We summarize our contributions in this paper as follows.

1) We consider an error detection approach based on the time-redundancy techniques, i.e., the recomputing with rotated operands (RERO) scheme. Through the RERO-based approach, low hardware overhead is added to the original designs, suitable for lightweight and low-power implementations.
2) To evaluate the error detection capability of the proposed scheme in response to transient and permanent faults, the proposed error detection structures are simulated. Through our simulations, we demonstrate that the proposed scheme reaches high error coverage.
3) Finally, the original SHA-3 and our proposed error detection scheme are synthesized using a 65-nm ASIC standard-cell library to obtain the area overheads and the performance metrics. Our synthesis results show acceptable overhead for the proposed technique, while achieving very high error coverage.

The organization of the paper is as follows. In Section II, preliminaries regarding the SHA-3 algorithm are provided. Section III presents our proposed error detection scheme. In Section IV, through fault-injection simulations, we evaluate the error coverage capabilities of the proposed scheme. Section V is presented to benchmark the overheads of the proposed scheme on ASIC utilizing a 65-nm standard-cell library. Finally, conclusions are made in Section VI.

Manuscript received October 6, 2013; revised January 1, 2014; accepted February 12, 2014. Date of current version June 16, 2014. This paper was recommended by Associate Editor C.-W. Wu.

S. Bayat-Sarmadi is with the Department of Computer Engineering, Sharif University of Technology, Tehran 11155–11365, Iran (e-mail: sbayat@sharif.edu).

M. Mozaffari-Kermani is with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: m.mozaffari@rit.edu).

A. Reyhani-Masoleh is with the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON N6A 3K7, Canada (e-mail: areyhani@uwo.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2307002

II. Preliminaries

In this section, we briefly explain the SHA-3 algorithm.

A. Keccak

The core of the Keccak algorithm is the permutation f which is repeatedly applied to a fixed-length state of $b = r + c$ bits, where r and c are bit rate and capacity, respectively. Higher values of r improve the speed whereas higher values of c correspond to higher security level. The input message is first padded to get a length multiple of r. Then, through five internal steps for each round, the absorbing phase is performed. Finally, the squeezing phase occurs in which the first r bits of the state are returned as the output block [1].

There are seven possible types for Keccak and for the sake of brevity, we focus on the recommended type, namely, Keccak-f[$r + c = 1600$] ($c = 1024$ and $r = 576$). In Keccak-f[1600], 1600 is the width of the underlying permutation (in bits). For this type of Keccak, the state consists of an array of 5×5 lanes, each of length $w = 64$ bits. The recommended number of rounds for Keccak-f[1600] is 24 [1].

Let us introduce the five internal steps of each of the 24 rounds of Keccak-f[1600] shown in Fig. 1. For all these steps, $0 \leq x, y \leq 4$. The first internal step is θ as follows:

$$\begin{aligned} C[x] &\leftarrow A[x, 0] \oplus A[x, 1] \oplus A[x, 2] \oplus A[x, 3] \oplus A[x, 4] \\ D[x] &\leftarrow C[x-1] \oplus rot(C[x+1], 1) \\ A[x, y] &\leftarrow A[x, y] \oplus D[x]. \end{aligned} \tag{1}$$

In (1), $A[x, y]$ denotes a particular lane in the state A, with C and D being internal states. Moreover, the operations on indices are bitwise XOR, modulo-5 addition and subtraction represented as $\oplus$, $+$, and $-$, respectively. Finally, $rot(C[x + 1], 1)$ is the bitwise cyclic shift operation, moving bit at position i into position $i + 1$ (modulo the lane size).

The next two steps are ρ and π as follows: $B[y, 2 \times x + 3 \times y] \leftarrow rot(A[x, y], r[x, y])$. We note that the constants $r[x, y]$ are the rotation offsets (refer to [1]).

The second to last step is χ, i.e., $A[x, y] \leftarrow B[x, y] \oplus (\overline{B[x + 1, y]}.B[x + 2, y])$. The overline and . symbols represent NOT and AND operations, respectively. Finally, in the last step, we have $A[0, 0] \leftarrow A[0, 0] \oplus RC$, where RC is the round constant specific for each of the 24 rounds of Keccak-f[1600]. Lastly, in the final squeezing phase, the first r bits of the state are returned as the output block.

B. RERO Method

RERO [27] is a technique for concurrent error detection introduced for arithmetic units. As mentioned earlier, it is a redundancy-based technique. Suppose R and R^{-1} are n-bit rotations (or cyclic shifts) toward the least and most significant bits of a binary operand, respectively, where n is less than the size of the operand. Moreover, let x be the input to an arithmetic function f and $f(x)$ be its output in such a way that $R^{-1}(f(R(x))) = f(x)$. To apply the RERO method, we need to store the result of the $f(x)$ computation (first run) and compare it against the result of the $R^{-1}(f(R(x))$ computation (second run). If the results are different, it indicates an error alerted by the error indication flag.

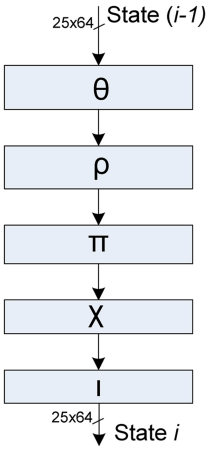

Fig. 1. Five internal steps in the Keccak algorithm.

III. Error Detection Approach

In this section, first we briefly describe the motivations for this paper, and then we present an efficient error detection scheme for the SHA-3 algorithm. We note that although we have considered the hash size of 512 (which is usually recommended and is the highest one), the proposed scheme is applicable to other hash sizes. In the next two sections, the presented scheme is benchmarked in terms of error coverage and performance metrics.

A. Motivations

SHA-3 is one of the important cryptographic tools which is being used for system and information security. This cryptographic tool can be used in assuring data integrity as changing one single bit in the input message can change about half of the output digest (denoted as avalanche effect). It also can help to achieve authenticity and nonrepudiation more efficiently as signing the entire input message or data is costly.

We can consider two reasons for having error detection for such an important cryptographic tool.

1) Digital circuits are prone to natural faults and this is the case for the hardware implementations of SHA-3 as well.
2) Malicious attacks such as those based on fault injections can target the availability of this algorithm which, in turn, result in malfunctioning of the integrity-checking process. This can induce much overhead to the system especially the constrained nodes such as those used in sensitive applications, e.g., security of implantable and wearable medical devices or constrained industrial setups.

Both of the above items can cause the integrity check to fail, introducing some levels of denial-of-service (DOS). Apparently, this is not desirable as it contradicts availability which is one of the features of a secure system. Also, regardless of the security point of view, such unreliable systems may need to restart or reinitialized for several times. Such behavior means more energy consumption and more cost.

Concurrent error detection can help to detect the errors as much as possible and stop the system or process gracefully to resolve the issue. We note that for malicious attacks, error

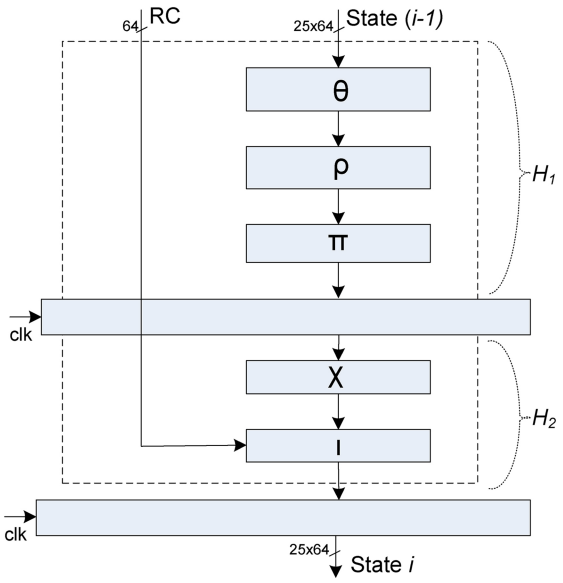

Fig. 2. Keccak algorithm modified for the presented RERO-based approach.

detection can reduce the choices of attacks and may not prevent them completely. However, inducing this difficulty is of much great importance and can be intensified by other security measures to have reliable architectures.

B. Proposed Concurrent Error Detection Scheme

We propose a RERO-based error detection method for Keccak-f[1600] which is also applicable to other variants of Keccak.

A simplified structure for the presented RERO-based error detection approach is shown in Fig. 2. As seen in this figure, a pipeline-register has been placed after the π step (compare Figs. 1 and 2). This location for placing the registers is chosen to break the timing path in to approximately equal halves. Let us denote the two halves of pipelined stages by H_1 and H_2. Now, we briefly explain the utilized method through this figure. In Fig. 2, the original input is first applied to the architecture in the first cycle. In the second cycle, while the second half of the circuit (H_2) executes this first input, the rotated variant of the first input is fed to the first half of the circuit (H_1). This trend is consecutively executed until the last rotated input is derived. It is worth mentioning that for second runs, we rotated each of 26 input words (25 state words and one RC word) by a number between 1 and 63, where each word has 64 bits. We also note that for detecting the errors, the outputs of the runs with the rotated-inputs are rotated back and compared against the original inputs. Any mismatch indicates an error.

We take advantage of subpipelining to reduce the throughput degradation of the proposed scheme. In this regard, the order of applying the inputs is managed so that we take advantage of concurrent executions. Although the added subpipelining registers slightly increase the induced hardware overhead, it is more preferable to use 100% time-redundancy schemes which introduce much more overall design overhead. Time-redundancy techniques inherently tend to increase the number of cycles needed for computations. This reduces

H_1	N_1	R_1	N_2	···	R_n	
H_2		N_1	R_1	···	N_n	R_n

Fig. 3. Pipelined scheduling for data path of the proposed schemes.

TABLE I
INJECTION OF TRANSIENT AND PERMANENT FAULTS FOR THE RERO-BASED SCHEME FOR 1 000 000 RANDOM INPUTS

Type of Stuck-at Fault	Transient Faults		Permanent Faults	
	No. per Input	Detection	No. per Input	Detection
Single bit	1664	100%	3328	100%
2 bits	500	99.99%	500	100%
3 bits		99.99%		100%
4 bits		99.99%		100%
5 bits		99.99%		100%
6 bits		100%		100%
7 bits		100%		100%
8 bits		100%		100%
Random No.		100%		100%

the throughput of the hardware implementations accordingly. Therefore, in our proposed approach, by introducing subpipelining, we increase the frequency of the clock to make sure the design throughput is close to the one for the original structure.

As mentioned, the throughput improvement approach is carried out through a subpipelining method to compensate for the inherent throughput reduction of the time-redundancy approaches. Then, processing of the inputs to the pipelined stages is scheduled to improve the throughput degradations of the error detection approach. Using this method, higher frequencies are achieved which compensate for the higher number of cycles needed for calculations. We have shown our scheduling in Fig. 3. As seen in this figure, the orders of cycles of normal (N_i) and rotated (R_i) operations are shown, where $1 \leq i \leq n$ and n is the number of cycles in the original (nonpipelined) approach. In this scheme, R_i and N_i are performed at the same cycle but in different pipelined stages. Then, in the next cycle, N_{i+1} and R_i are performed.

If we consider 48 cycles needed for the original pipelined architecture of Keccak-f[1600] to execute 24 rounds of five steps each, the number of cycles is intact when the RERO-based approach is utilized. This is because of the feedback structure of the rounds of Keccak-f[1600] (see Fig. 2). Consequently, as it is also shown later when we derive the overhead of the proposed approach through ASIC syntheses, low degradation in throughput is observed compared to the original Keccak-f[1600]. Finally, the proposed approach is capable of detecting both transient and permanent errors with very high coverage. Due to its low hardware overhead, the presented approach is suitable for the applications requiring high performance and low complexity.

IV. FAULT-INJECTION SIMULATIONS

To evaluate the error detection capability of the proposed scheme, we have performed injection-based fault simulations coded in C programming language. We need to select the number of bit-rotations for the simulations (has to be selected once at the very beginning of the second run and is fixed

TABLE II
ASIC SYNTHESES AND OVERHEAD ASSESSMENTS USING TSMC 65-NM STANDARD-CELL

Algorithm	Architecture	Area		Frequency	Throughput
		[μm^2]	kGE	[MHz]	[Gbps]
Keccak-f[1600] (512)	a. Original (with frequency constraint)	66,306	47.0	676	28.8
	b. Original (with loose constraint)	63,994	45.4	632	26.9
	RERO-based scheme	69,249 (w.r.t (a): 4.4%) (w.r.t (b): 7.6%)	49.1	1,192	25.4 (w.r.t (a): 11.8%) (w.r.t (b): 5.9%)

till the end of that run). Although our proposed approach is not restricted to this fixed number, for simulations, this has been selected as 32. In fact, one can use different options to either select a fixed rotation number for the entire design or to make the rotation number programmable for each time this scheme is used. Apparently, the second method needs a bit more hardware control and will result in a slightly more area overhead. In our work, we consider the fixed rotation approach (first one) as our goal in this paper is to describe the actual scheme, its capability and efficiency.

We have used the C-model of the SHA-3 algorithm available in [28]. The faults that we consider are stuck_at_0 and stuck_at_1. Both single-bit and multiple-bit faults were injected at the input state of the algorithm as well as the round constant (RC) which is being used in the last internal step. We refer to the locations that fault were injected as fault locations. The number of fault locations for a single round is 1664(= 25 × 64 + 64). The reason for performing injection at these locations is to achieve some speed up as a complete injection on all internal nets is very time consuming. Moreover, we consider the transient faults to occur at both runs independently. Therefore, the fault locations for transient faults are twice of the above numbers. However, for permanent faults, each fault can occur at the same location for both runs with the same polarity. Therefore, the number of fault locations for this type is half of that for the transient one.

For single-bit fault-injection, we generated 1 000 000 random inputs and for each random input, both the stuck_at_0 and the stuck_at_1 faults were injected at the above mentioned locations. For multiple-bit fault-injection simulations, similarly, 1 000 000 random inputs were generated and 500 multiple-bit faults were injected per each random input. For each multiple-bit fault-injection, say n-bit fault-injection, n random locations from the above mentioned locations were chosen and the stuck_at_0 or the stuck_at_1 faults were randomly injected.

Let the number of all the faults, detected faults, and masked faults be denoted as AF, DF, and MF, respectively. The number of faults that can manifest into errors, i.e., in case of no fault protection, they can produce erroneous result at the output of the circuit, is $AF - MF$. Then, the percentage of error detection is calculated as $\frac{\mathrm{DF}}{\mathrm{AF - MF}}\%$.

The error detection results for transient fault-injections into all functions/circuits using RERO-based method are presented in Table I. As shown in the table, all the injected single-bit transient faults can be detected in the RERO-based scheme. Multiple-bit faults can also be detected with very high probability for transient faults. Moreover, the table shows that the faults with higher weight (with more number of bits) can be detected with higher rate in such a way that all the injected faults with random number of bits can be detected. Based on our experiments, all the permanent injected faults are detected no matter what the number of injected faults is. The error detection capability for permanent multiple-bit faults is slightly higher than that for the transient ones. The reason can be explained by an example. A 2-bit transient fault (with same polarity for both bits) may not be detected if one fault occurs at the first run and in the bit, say, 0, of a word and the second one occurs at the second run and in the bit 32 of the same word. Occurrence of such faults for permanent type is less probable.

Finally, the evaluation of the error detection capability is based on fault-injection based simulation. The number of locations of possible fault injections for transient and permanent faults are reported in Table I. Also, the simulation is repeated for 1 000 000 random inputs. Hence, the coverage is reported according to the result of the simulation and is not actually always 100% (see Table I). This is a simulation-based benchmark and is not as accurate as math models; however, it is typically considered as a fair measure to evaluate the error detection capabilities of schemes.

V. SYNTHESES ON ASIC AND OVERHEAD DERIVATIONS

This section presents the results of our ASIC syntheses performed for SHA-3 algorithm to benchmark the overheads induced. We note that ASIC is chosen based on the resources available to us (library and tools) and because our presented schemes are not dependent on the hardware platform, similar overheads are expected if FPGAs are utilized for the implementations. Through these ASIC syntheses, the overheads in terms of hardware and timing are derived. We have used the TSMC 65-nm standard-cell library [29] for the presented post-synthesis results. VHDL has been used as the design entry for the original and the error detection structures to the Synopsys Design Compiler [30]. The VHDL code for original design is from [28].

We have presented the results of our syntheses in Table II. In this table, for the original SHA-3 algorithm, Keccak, and the proposed error detection scheme, the areas (in terms of μm^2), maximum working frequencies (in terms of MHz), and throughputs (in terms of Gb/s) have been obtained. In order to make the area results meaningful when switching technologies, we have provided the NAND-gate equivalency (kilo gate equivalent, kGE). This is performed using the area of a NAND gate in the utilized TSMC 65-nm CMOS library which is $1.41\mu m^2$. Additionally, the area and throughput degradations are presented in parentheses to benchmark the proposed error detection schemes. We note that for the original

algorithm, the area is dominated by the steps θ (54%) and χ (45%), while three other steps constitute around 1%.

We have synthesized the original Keccak-f[1600] and its error detection approach (we need 25 x 64 additional flip-flops for this architecture). Two different experiments have been performed for deriving the area and timing overheads. In the first one, a period constraint has been imposed to achieve the working frequency of more than 650 MHz for the original algorithm (although this is applied to the fault detection architecture as well, it does not affect its area due to the high-speed nature of this architecture). This constraint allows higher performance for the original structure at the expense of higher area (suitable for high-speed applications). Moreover, another experiment has been conducted by loosening the period constraint which has resulted in lower working frequency for the original architecture and lower area. A variant of Keccak-f[1600] is considered which operates one round in each cycle (the total 5-step number of rounds is 24) to derive the hashed output. As seen in Table II (in the parentheses), specifically, for the constrained architectures, the area overhead and throughput degradation for the RERO-based scheme for Keccak-f[1600] are 4.4% and 11.8%, respectively; whereas, for the cases that these are loosened, the overheads are 7.6% and 5.9%, respectively.

VI. Conclusion

We have presented a time-redundancy scheme for error detection of the recently-standardized secure cryptographic SHA-3 algorithm, i.e., Keccak. The proposed approach is based on the low hardware overhead RERO-based approach. We have also applied subpipelining to overcome the inherent throughput degradation of this time-redundancy approach. Through fault-injection analysis, it has been shown that the error coverage is 100% for multiple random fault-injections. Moreover, through ASIC synthesis, we have shown that the area and throughput degradations for the RERO-based approaches are 4.4% and 11.8%, respectively. Thus, this approach is suitable for resource-constrained applications. Based on the reliability requirements and available resources, one may utilize the proposed error detection scheme for making the hardware implementations of secure cryptographic SHA-3 algorithm more reliable.

References

[1] *Keccak Hash Function*, NIST (National Institute of Standards and Technology), (2014, Mar.) [Online]. Available: http://csrc.nist.gov/groups/ST/hash/sha-3

[2] D.-J. Bernstein and T. Lange. (2012). The new SHA-3 software shootout. *e-Print* [Online]. Available: http://eprint.iacr.org/2012/004.pdf

[3] S. Tillich, M. Feldhofer, M. Kirschbaum, T. Plos, J.-M. Schmidt, and A. Szekely, "Uniform evaluation of hardware implementations of the round-two SHA-3 candidates," in *Proc. Conf. SHA-3 Candidate*, pp. 1–16, 2010.

[4] X. Guo, S. Huang, L. Nazhandali, and P. Schaumont, "Fair and comprehensive performance evaluation of 14 second round SHA-3 ASIC implementations," in *Proc. Conf. SHA-3 Candidate*, pp. 1–13, Aug. 2010.

[5] M. Knežević *et al.*, "Fair and consistent hardware evaluation of fourteen round two SHA-3 candidates," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 5, pp. 827–840, May 2012.

[6] E. Homsirikamol, M. Rogawski, and K. Gaj, "Throughput vs. area trade-offs in high-speed architectures of five round 3 SHA-3 candidates implemented using Xilinx and Altera FPGAs," in *Proc. Workshop Cryptograph. Hardw. Embedded Syst.*, 2011, pp. 491–506.

[7] K. Latif, M. Rao, A. Aziz, and A. Mahboob, "Efficient hardware implementations and hardware performance evaluation of SHA-3 finalists," in *Proc. Conf. SHA-3 Candidate*, pp. 1–14, Mar. 2012.

[8] E. M. Shakshuki, N. Kang, and T. R. Sheltami, "EAACK–A secure intrusion detection system for MANETs," *IEEE Trans. Ind. Electron.*, vol. 60, no. 3, pp. 1089–1098, Mar. 2013.

[9] M. Mozaffari-Kermani, M. Zhang, A. Raghunathan, and N. K. Jha, "Emerging frontiers in embedded security," in *Proc. Conf. VLSI Design*, Jan. 2013, pp. 203–208.

[10] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection schemes of fault based side-channel cryptanalysis of symmetric block ciphers," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1509–1517, Dec. 2002.

[11] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Parity-based fault detection architecture of S-box for advanced encryption standard," in *Proc. IEEE Int. Symp. Defect Fault-Tolerance VLSI Syst.*, 2006, pp. 572–580.

[12] M. Mozaffari-Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5925–5932, Dec. 2013.

[13] M. Mozaffari-Kermani, R. Azarderakhsh, C.-Y. Lee, and S. Bayat-Sarmadi, "Reliable concurrent error detection architectures for extended Euclidean-based division over GF(2^m)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, DOI: 10.1109/TVLSI.2013.2260570.

[14] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Fault detection structures of the S-boxes and the inverse S-boxes for the advanced encryption standard," *J. Electron. Test.*, vol. 25, nos. 4–5, pp. 225–245, 2009.

[15] X. Guo and R. Karri, "Recomputing with permuted operands: A concurrent error detection approach," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1595–1608, Oct. 2013.

[16] D. R. Avresky, S. J. Geoghegan, and Y. Varoglu, "Evaluation of software-implemented fault-tolerance (SIFT) approach in gracefully degradable multi-computer systems," *IEEE Trans. Rel.*, vol. 55, no. 3, pp. 451–457, Sep. 2006.

[17] S. Bayat-Sarmadi and M. A. Hasan,"On concurrent detection of errors in polynomial basis multiplication," *IEEE Trans. VLSI*, vol. 15, no. 4, pp. 413–426, Apr. 2007.

[18] I. Ahmad and A. Das, "Analysis and detection of errors in implementation of SHA-512 algorithms on FPGAs," *Comput. J.*, vol 50, no. 6, pp. 728–738, 2007.

[19] M. Juliato, C. Gebotys, and R. Elbaz, "Efficient fault tolerant SHA-2 hash functions for space applications," in *Proc. IEEE Conf. Aerosp.*, 2009, pp. 1–16.

[20] M. Bahramali, J. Jiang, and A. Reyhani-Masoleh, "A fault detection scheme for the FPGA implementation of SHA-1 and SHA-512 round computations," *J. Electron. Test.*, vol. 27, no. 4, pp. 517–530, 2011.

[21] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 608–622, May 2010.

[22] A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "High-performance concurrent error detection scheme for AES hardware," in *Proc. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2008, pp. 100–112.

[23] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A low-power high-performance concurrent fault detection approach for the composite field S-Box and inverse S-Box," *IEEE Trans. Comput.*, vol. 60, no. 9, pp. 1327–1340, Sep. 2011.

[24] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Reliable hardware architectures for the third-round SHA-3 finalist Grøstl benchmarked on FPGA platform," in *Proc. Conf. Defect Fault Tolerance VLSI Syst.*, Oct. 2011, pp. 325–331.

[25] S. Bayat-Sarmadi and M. Hasan, "Concurrent error detection in finite-field arithmetic operations using pipelined and systolic architectures," *IEEE Trans. Comput.*, vol. 58, no. 11, pp. 1553–1567, Nov. 2009.

[26] N. Joshi, K. Wu, J. Sundararajan, and R. Karri, "Concurrent error detection for involutional functions with applications in fault-tolerant cryptographic hardware design," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1163–1169, Jun. 2006.

[27] J. Li and E. E. Swartzlander, "Concurrent error detection in ALUs by recomputing with rotated operands," in *Proc. Conf. Defect Fault Tolerance VLSI Syst.*, Oct. 1992, pp. 109–116.

[28] (2014, Mar.). Keccak Website [Online]. Available: http://keccak.noekeon.org/

[29] (2014, Mar.). Taiwan Semiconductor Manufacturing Company [Online]. Available: http://www.tsmc.com/

[30] (2014, Mar.). Synopsys [Online]. Available: http://www.synopsys.com/