IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 July 2024, accepted 12 August 2024, date of publication 22 August 2024, date of current version 5 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3448313

RESEARCH ARTICLE

Self-Checking Hardware Design for Montgomery Exponentiation-Based Cryptography

MUHAMMAD ALI AKBAR[1], ABDULLATIF SHIKFA[2], BO WANG[1], (Senior Member, IEEE), AND AMINE BERMAK[1], (Fellow, IEEE)

[1]Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar
[2]Department of Data and Cyber Security, College of Computing and Information Technology, University of Doha for Science and Technology, Doha, Qatar

Corresponding author: Muhammad Ali Akbar (muakbar@hbku.edu.qa)

Open Access funding provided by the Qatar National Library. This work was supported by National Priorities Research Program (NPRP) under Grant NPRP13S-0212-200345 from the Qatar National Research Fund (a member of Qatar Foundation). The findings herein reflect the work and are solely the responsibility of the authors.

ABSTRACT Montgomery exponentiation is widely used for public-key-based cryptography systems. The current state-of-the-art designs for this algorithm are well-analyzed in terms of hardware overhead but are not investigated for faults caused by physical attacks. This paper presents a self-checking hardware design for the Montgomery Multiplier (MM), which can counter multiple faults simultaneously. The proposed 64-bit self-checking MM approach with a distributed fault prognosis mechanism requires only 43.5% area and 10.9% power overhead as compared to the non-self-checking design. Moreover, a novel self-checking parity prediction approach is proposed for carry save adder, which can be used in cases where it is used alone inside a loop.

INDEX TERMS Montgomery exponentiation, fault localization, self-checking adder, parity prediction.

I. INTRODUCTION

Modular exponentiation is widely used in public key cryptography systems [1], [2], in which a pair of public and private keys are used for secure communications. A repeated multiplication operation is required in modular exponentiation, followed by a division operation to provide modulus. In terms of hardware design, both multiplication and division are the most expensive operations because the product of two binary numbers will have a maximum size equal to the summation of the sizes of the two numbers. Suppose a and b are two integers having m and n bits, respectively, the result of their multiplication will be a maximum of m+n bits. It is not easy to handle such big numbers while keeping their partial products during the multiplications. The problem becomes more challenging if large exponent values need to be solved, as in the case of cryptography. Therefore, different algorithms like sliding window, k-ary methods etc, have been proposed to counter this problem, including the Montgomery algorithm, which has become prominent because of its hardware-friendly approach [3]. It simplifies and improves the processing speed of large key values, which is essentially required for system security [4], [5]. Therefore, it is considered as the main constituent part for the hardware architecture of public key cryptography like RSA [6], Diffie-Hellman [5], and other applications that require modular multiplication or exponentiation [7], [8], [9]. The resource efficient hardware architecture for Montgomery remains an active topic in the literature, with different optimizations at algorithmic and implementation levels as found in [5], [6], [7], [8]. However, these optimizations endanger the computational integrity of hardware design because of the resultant complexity which is directly related to system reliability.

The increasing complexity of systems on chip design with low power and heat exchange requirements makes the current digital system vulnerable to faults [10]. The thermal cycling, hardware aging, and other environmental conditions further raise the possibility of internally generated faults [11], [12]. In addition to these architectural concerns, the primary goal of cryptographic primitives is to ensure the security of the device. However, the said security can be compromised if the hardware architecture is designed without considering the physical attacks [13], [14], [15]. Among physical attacks,

The associate editor coordinating the review of this manuscript and approving it for publication was Tony Thomas.

fault injection-based side channel attacks cause serious concern for cryptographic devices because the attacker can potentially uncover the hidden security features (such as keys) regardless of theoretical and mathematically proven security of the primitives [16], [17]. The deliberate incursion of faults will force the crypto-processor to produce incorrect output, whose pattern can disclose the device's secrets, as investigated by [18]. These attacks can easily be achieved with slight modifications in the device parameters, such as, depleting the power supply, electromagnetic disturbances, clock tampering or environmental conditions, etc [19]. Hence, the reliability of crypto-systems cannot be ensured without achieving resistance against both intentional and unintentional faults.

Therefore, the need of self-checking hardware design for crypto-systems emerges with the earliest work reported in [20]. The challenging factor of self-checking hardware lies in the area or time overhead required by the fault prognosis process [11]. Therefore, the self-checking ability in critical parts will improve the overall reliability of the system [21]. Since the Montgomery Multiplier (MM) constitutes the major computational part of public key cryptography. Moreover, the repeated operations in the MM algorithm make it more prone to fault accumulation and propagation problems. A single erroneous bit at any stage of MM algorithm can entirely change the final output. Therefore, fault diagnosis in MM will have a significant impact on the overall reliability of the crypto-systems. However, to the best of our knowledge, the self-checking ability of MM hardware architecture has yet to be investigated.

This paper aims to introduce self-checking hardware design for the MM algorithm with distributed fault detection ability. We investigate each block used in constructing the MM algorithm with respect to their self-checking ability. As a result of our block-wise analysis, the proposed self-checking MM architecture can detect multiple faults simultaneously while effectively handling the fault propagation problems. Moreover, it can easily be adopted in the existing and emerging hardware architectures of the MM algorithm because the principle blocks remain the same irrespective of the optimization approach. The design of the MM algorithm with and without self-checking is implemented using Verilog HDL and synthesized using the synopsis design compiler. The proposed 64-bit self-checking MM architecture with multiple error-resilient features requires 43.5% area and 10.9% power overhead compared to the standard MM architecture. Moreover, a self-checking parity prediction approach for carry save adder (CSA) is proposed, which can be used primarily when the block is used inside a loop without carry propagate adder (CPA). In addition to this, the performance of different self-checking CSA is analyzed to select the optimal solution for MM.

The remaining parts of this paper are organized as follows. Sections II and III cover the related work and the MM algorithm with its standard hardware implementation. The proposed self-checking MM architecture is described in Section IV. Performance evaluation is presented in Section V, followed by a conclusion in Section VI.

II. RELATED WORK

The security of electronic devices is largely dependent on crypto algorithms, which often provide provably secure encryption or signature methods, assuming that the algorithm is well implemented and terminates properly. However, even the best theoretical algorithms would fail if their implementation had issues. One possible attack vector is to introduce faults deliberately to prevent the proper execution of the algorithm, which would produce incorrect results at least and might even reveal secrets at worst. Attacking hardware implementations by injecting faults is an active research field: a laser-based fault attack for identifying the vulnerable area in the chip before executing the attack is demonstrated by [22]. Another laser-based fault injection approach to skip instruction for disclosing the secret key is investigated by [23]. In the work of [24], a 128-bit key has been uncovered with less than seven injected faults on average. Detailed surveys of possible fault attacks on cryptosystems are presented in [25], [26], [27]. These works show different possible impacts of injecting such deliberate faults on cryptographic devices, hence outlining the need to come up with countermeasures to such critical attacks.

The most common approach to handle such faults is to use either the same or duplicated hardware to produce a copy of output at different or same periods [17]. In the work of [28], a double modular redundancy approach is used for fault detection by comparing the duplicated sum bits using a two-pair-two-rail checker. Besides the fault localization problem, the reported design requires additional area overhead for duplicating the summation block along with other checking circuitry. In order to reduce the hardware penalty, a parity prediction approach for self-checking carry select adder design is reported in [29]. The parity of the final sum bits is estimated using the parity of input operands and the intermediate carry bits. The fault is detected by comparing the estimated parity with the actual parity of sum bits. A similar concept of parity prediction is used by [30] to protect the sum bits of the carry-lookahead adder. The parity prediction-based self-checking designs can detect faults in even or odd number of bits without indicating the exact location of their occurrence. Furthermore, the absence of operational diversity raises concerns about common mode failure.

To mitigate this issue, the code disjoint approach is introduced, in which separate hardware produces the encoded output [16], [17]. The comparison result between the actual and encoded output will determine the fault. A residue code-based modulo multiplier is reported by [16], with a modified compressor and modulo generator. The area overhead is reduced due to interconnected circuitry, however, it comes at the cost of fault propagation. The reported design offers limited fault coverage because the fault identification depends on the check base value. Moreover, the final comparison

requires output transformation, which causes additional delay overhead. These approaches can detect faults without indicating the exact location of their occurrence. However, fault localization is desirable for reliable hardware design because it can play a vital role in recovery process [31], [32].

Therefore, an improved code-based fault secure approach is reported by [17], which can limit the fault propagation problems. Besides the benefit of partial fault propagation, the approach suffers from area and performance penalty. In the work of [33], a hamming code approach is used to protect the data against faults during the transformation process. The reported design will increase the data size and can detect faults in even or odd number of bits, while the key generation, sharing, and scheduling part are not covered in the proposed study. In the work of [34], a parity prediction approach is adopted to secure the S-box. The reported approach is further improved by [35], where the parity approach is used so that the S-box is protected regardless of their composition style. However, the parity-based self-checking approach cannot be used in MM architecture because of the inter-connectivity issue, due to which the fault generated in one block can easily be propagated to other blocks. The fault propagation phenomenon can cause multiple-bit errors, which the parity approach cannot handle.

III. MONTGOMERY MULTIPLIER TOPOLOGY

The MM algorithm is frequently used in cryptographic applications to handle the large exponent value in secret key generation. Due to the application's sensitivity, we investigate the self-checking ability of the MM's hardware design without affecting the computational performance.

A. BACKGROUND

In the Montgomery approach for modular multiplication, the numbers will be transformed in Montgomery form such that a shifting operation replaces the expensive multiplication and division operation. The limitation of such an approach lies in the delay caused by the initial conversion of the number to the Montgomery form and then converting the result back to the original state. Therefore, MM is recommended for modular exponentiation, i.e., a^e mod n, because the intermediate results can be kept in the Montgomery form [36]. Different hardware approaches concerning the area and time efficiency for MM have been presented in the literature. In the work of [37], a scalable architecture for the MM algorithm is presented, whereas a pipelined architecture for high radix MM is presented in another work [5]. However, the Radix-2 architecture remains prominent for MM implementation due to its low complexity [38]. The main concern of Radix 2 architecture is the clock cycles, due to which the high radix architecture is shown to be 1.4 times faster than the Radix 2 approach [39]. Therefore, an energy-efficient Radix-2 algorithm with reduced clock cycles is presented by [40]. It also uses the clock gating approach to reduce further the power of most of the registers used in the design. In the subsequent work, an improved variant with a modified semi-carry save-based MM approach is presented by [41], where a detection circuit has been used with a single-level CSA for skipping the unnecessary addition operation by pre-computing the quotient values.

B. MM ALGORITHM

The MM algorithm mainly depends on a constant value R, whose value is chosen to be the power of 2. The selection of the power of two values will replace the multiplication and division process with a simple shift operation.

Suppose we have two numbers a, b, and we need to compute $y = ab \bmod N$, where N is a k-bit number. The value of R should be selected such that $R \geq 2^k$, and R is co-prime to n, i.e., $gcd(R, N) = 1$. Let R be equal to 2^k, and N is an odd number because the value of R is always even. The MM form of a and b with respect to R can be computed using (1) and (2). Further details of the MM algorithm can be found in [40].

$$A = aRmodN \tag{1}$$

Similarly,

$$B = bRmodN \tag{2}$$

C. HARDWARE IMPLEMENTATION

The hardware implementation of the MM algorithm can be achieved in different ways depending on the area, power, and time constraints. The most prominent one is the Radix-2 MM algorithm shown in the algorithm-1 [40], [42]. In contrast to the actual algorithm, it replaces multiplication or division operations with a single-bit shift operation. Also, it does not require the computation of the inverse of R or N value using an extended Euclidean algorithm. Moreover, the MM form of a and b for public key cryptography can be pre-computed because the values of a, b, and N are fixed and globally available. In contrast, the secret key remains changing during communication, which appears as an exponent. The central process needed for implementing the MM algorithm is a 3-operand addition operation, as observed from algorithm 1.

Algorithm 1 MM Algorithm for Modular Exponentiation Using Radix-2 [40]

Data: $A, B, N(modulus)$
Result: $S[k]$
$S[0] \leftarrow 0$;
for $i = 0$ *to* $k - 1$ **do**
  $q_i = (S[i]_0 + A_i \times B_0) mod 2$;
  $S[i+1] = (S[i] + A_i \times B + q_i \times N)/2$;
end for
if $S[k] \geq N$ **then**
  $S[k] = S[k] - N$
end

In algorithm 1, the q_i bit inside the For loop is required to make the value of $S[i+1]$ perfectly divisible by 2. The

Algorithm 2 Radix-2 MM Algorithm With 2-Layer of CSA [41]

Data: $A, B, N(modulus)$
Result: $S[k+2]$
$SS[0] = SC[0] = 0;$
for $i = 0$ *to* $k+1$ **do**
  $q_i = (SS[i]_0 + SC[i]_0 + A_i \times B_0) mod 2;$
  $(SS[i+1], SC[i+1] =$
  $(SS[i] + SC[i] + A_i \times B + q_i \times N)/2;$
end for
$S[k+2] = SS[k+2] + SC[k+2]$

last step of the comparison and subtraction operation causes additional hardware complexity to retain the residue within the limits. Walter in [43] resolves the issue by keeping the values of A, B, and S within the range of 0 to 2N. Moreover, the number of iterations and the value of R has been changed to $k+2$ and 2^{k+2}, respectively [40]. Although the complexity has been reduced significantly, the propagation delay for addition operation has increased. Therefore, a layered structure is used for the 3-operand summation operation to improve the algorithm's performance. The top layer is constructed using the CSA, while the last layer is the CPA. All the intermediate layers will be constructed using CSA, which means the CPA will be used as the last step to generate the final output. The number of CSA layers depends on the algorithm. The details of the MM algorithm with multiple layers of CSA are presented in [40], [41]. In all cases, an MM algorithm is mainly constructed using a combination of CSA and CPA, along with some logical operations.

In this research, we focus on the hardware architecture of the MM algorithm, which is constructed using 2-layers of CSA. The CPA is used outside the loop for computing the final output, as shown in algorithm 2. The carry propagation delay has been reduced, which is only required to calculate the final value of the modular exponentiation, as shown in Fig. 2.

IV. PROPOSED SELF-CHECKING MM ARCHITECTURE

The hardware implementation of algorithm-2 has shown that the main blocks for MM design are the CSA, CPA, and registers, which are used to store the intermediate values. The authenticity of data retrieved from the registers can be ensured using the parity bit approach, commonly adopted in computer architectures. In contrast, the self-checking approach for adder design varies with the type of adder. Therefore, each CSA and CPA is considered separately for selecting the optimized self-checking approach.

A. CARRY SAVE ADDER (CSA)

CSA is the fastest adder design constructed using a chain of independent full adders to provide the output in sum and carry format. The output bits generated by CSA will be fed to CPA to generate the final sum and carry-out bits. The need for CPA at final output limits the use of CSA for adding more than

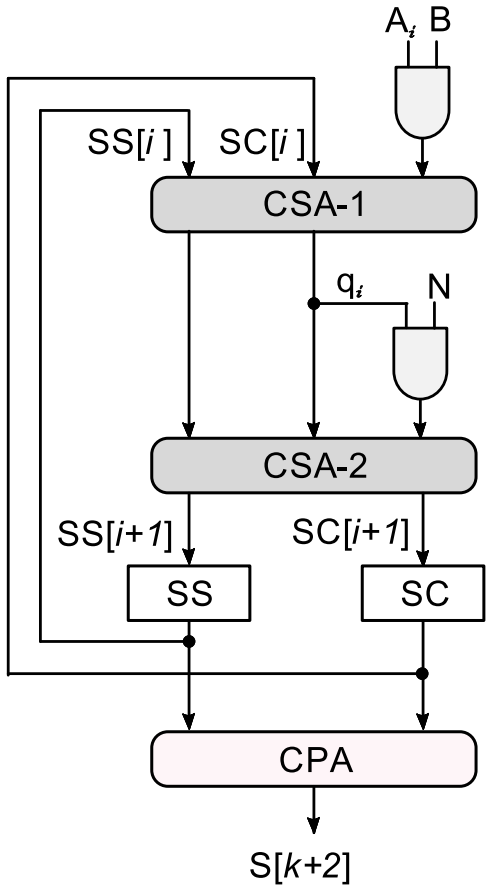


FIGURE 1. MM architecture using 2-Layer CSA.

3 bits. However, the prime advantage of CSA in self-checking is the absence of a carry propagation chain, which creates the fault propagation problem. In algorithm-2, a dual rail of CSA is used such that one operand for each of them is obtained from a group of AND gates, as shown in Fig. 2. The carry bits generated by the first CSA block are connected to the second CSA block after a single bit shift operation toward left because of which the first full adder of the second CSA block is replaced by half-adder.

Due to the limited scope of the generated fault, it is feasible to use any self-checking approach for CSA, including parity prediction, double-modular-redundancy (DMR), etc. In this research, we examine the hardware redundancy-based strategies for self-checking CSA. The fault coverage and performance evaluation for each self-checking approach are examined by considering the logical AND gates along with a pair of CSA.

1) SELF-CHECKING FULL ADDER (SFA) APPROACH

A self-checking full adder design with localized fault detection ability is presented in [31]. The reported design is based on the general operating principle of an adder, i.e., the sum and carry bit generated by the full adder will be equal if all the input bits are equal and vice versa. Therefore,

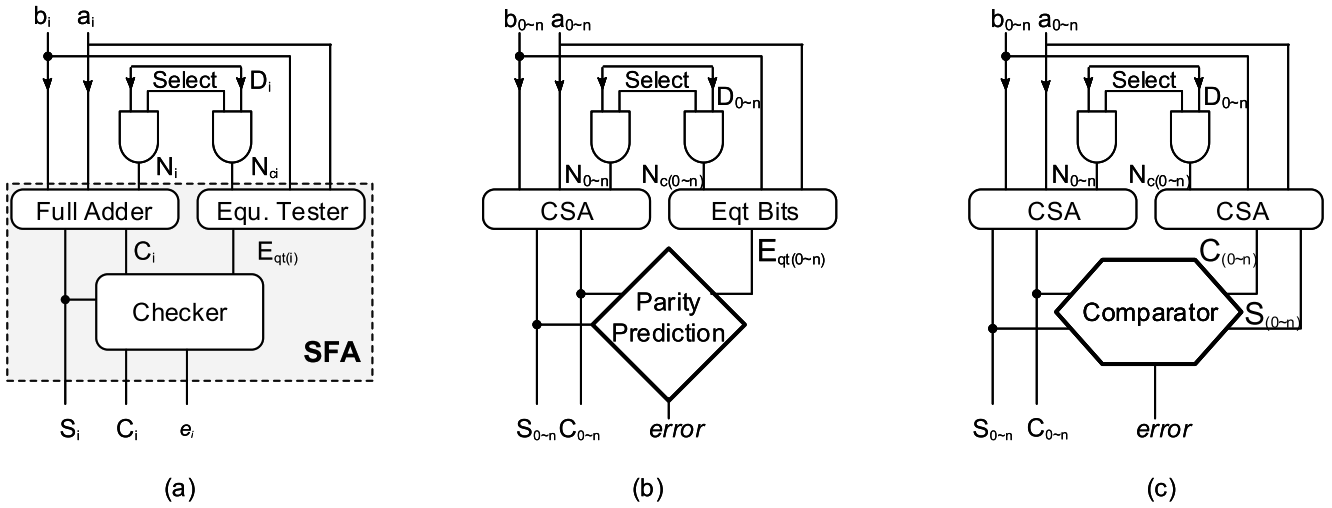


FIGURE 2. Self-checking CSA design using (a) SFA, (b) Parity prediction and (c) DMR.

an equivalence tester (Eqt) bit is required to construct the SFA, as depicted in Fig. 2 (a). The same Eqt bit concept can be used for self-checking half adder (SHA), which is used to compute the least significant bit for the second CSA. However, in the half adder, the sum and carry bits will complement each other if any or both input bits are logic-high. Therefore, the equation for computing the Eqt bit of SHA (Eqt_{SHA}) uses a single NOR operation. Both the SFA and SHA approaches can only detect the fault that occur in the internal architecture of the adder. The chain of independent full adders in CSA design can be replaced by SFA to obtain fault detection and localization. However, the AND gate connected to one of the operands of CSA is still prone to fault.

To overcome this problem, a self-checking AND gate is designed by using the duplex concept, such that the actual AND output is used to compute the sum and carry-out bit, whereas the duplicated output is used to compute the equivalence tester bit. Hence, the fault in the AND gate will be reflected in the output of SFA and can easily be identified using the checker, as shown in Fig. 2(a). The area overhead can be further reduced by using self-checking AND gate with dual output, as shown in [31]. Let A, B, and N be the three operands connected to the input port of CSA. The value of N is dependent on a select line; if the select line=0, then N=0 else, N is equal to the actual input operand D, as shown in Fig. 2(a). The sum and carry out (Cout) bit are generated using (3) and (4). The second AND gate produces a copy of N (N_c), which is used for computing the Eqt bit using (5) or (6), depending on the type of adder. The final error bit for both SFA and SHA is computed using (7).

$$Sum = A \oplus B \oplus N \tag{3}$$

$$C_{out} = A(B + N) + AN \tag{4}$$

$$Eqt_{SFA} = \overline{ABN_c + \overline{ABN_c}} \tag{5}$$

$$Eqt_{SHA} = \overline{A + N_c} \tag{6}$$

$$Error = Sum \odot C_{out} \oplus Eqt \tag{7}$$

2) PARITY PREDICTION APPROACH

The second approach utilized the concept of parity prediction (PP), which can effectively detect even or odd number of faults at a time. However, it is not recommended in highly interconnected designs because of fault propagation problem which can alter more than one bit at a time. In MM architecture, parity prediction is also advantageous for CSA because the last layer is connected to registers which are secured using parity bit. Hence, a parity prediction approach for CSA can be used to avoid the extra circuitry required to compute the parity bit for each register.

The standard principle of parity prediction for adders uses the parity of input and intermediate carry bits to estimate the parity of the sum bit. However, this principle is invalid for CSA alone because it produces partial sum and carry bits. If CPA is connected at the end to compute the final sum bits, then the standard parity concept can be applied to the CSA-CPA combination. However, in the case of MM, the CSA blocks are considered as standalone because they are used inside the loop such that each iteration uses partial sum and carry bits, while CPA is used to compute the final return value, as shown in algorithm-2. It is observed from Table. 1 that the parity of partial sum and carry bits by CSA block depends on the parity of the equivalence tester bits. If the parity of the equivalence tester is one, then the output parities of the CSA will not be equal and vice versa. This is because the Eqt bit is responsible for indicating the status of all input bits, and its value will be zero when all inputs are equal. Similarly, the sum and Cout bit will be equal when all inputs are the same and vice versa. The logical relationship between Eqt, the sum, and the Cout bit is shown in (8). It can be derived from (8) that the parity of Eqt is equal to the logical

TABLE 1. Partial truth table for CSA along with the input and output parity.

A1	A2	B1	B2	N1	N2	S1	C1	S2	C2	Pa	Pb	Pn	Ps	Pc	Peqt
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1
0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	1
0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	1
0	0	0	1	0	1	0	0	0	1	0	1	1	0	1	1
0	0	0	1	1	0	1	0	1	0	0	1	1	0	0	0
0	0	0	1	1	1	1	0	0	1	0	1	0	1	1	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1

XOR between the parity of sum and Cout bit, as shown in (9). Hence, a novel parity prediction approach for CSA is designed, which can be used in cases where CSA is used repeatedly without a CPA layer, as in the case of the MM algorithm. Let P_s be the parity of sum bits; P_c be the parity of carry bits, and P_{eqt} be the parity of equivalence tester bit; then the error bit is calculated by using (10). The faults in AND gates are handled using the same approach presented for SFA, as shown in Fig. 2b.

$$E_{qt} = Sum \oplus C_{out} \tag{8}$$

$$P(E_{qt}) = P(Sum) \oplus P(C_{out}) \tag{9}$$

$$Error = P_s \oplus P_c \oplus P_{Eqt} \tag{10}$$

3) DOUBLE MODULAR REDUNDANCY (DMR) APPROACH

DMR is the most common self-checking approach in which duplicated hardware blocks generate the same or complementary output. The comparison result between these two independently working functional blocks indicates the fault. Besides the area overhead issue, this approach also suffered from fault propagation and common mode failure. These issues are generally uncommon for CSA because of the lack of inter-connectivity. Therefore, in this research, we also examine the DMR for CSA, as shown in Fig. 2(c).

B. CARRY PROPAGATE ADDER (CPA)

The basic architecture of CPA constitutes a chain of interconnected full adders where the sum of each full adder depends on the carry of the previous full adder. The existence of a carry propagation chain is the main issue for both standard and self-checking CPA designs. In terms of standard configuration, the carry chain creates the delay problem, which is why different CPA architectures have been introduced in the literature. The main aim of these architectures is to introduce parallelism either in sum or internal-carry generation. In terms of self-checking, the carry chain creates the possibility of fault propagation, due to which the exact location of the fault cannot be recognized.

In this research, we use the most recent architecture of CPA, which can perform 3-operand addition utilizing the concept of prefix adders [44]. The reported design is constructed using a single layer of CSA followed by a carry-prefix adder. However, we adopt the carry-prefix part because the CPA is used only once outside the LOOP statement. The architecture and block diagram remain the same as in [44]. The final sum bit is generated after receiving the carry bit of the previous stage, whereas all the internal carries are generated in parallel. In the case of MM design, the sum bits generated by the last CSA will be shifted right before connecting to CPA. Therefore, the least significant bit for CPA is computed using a half-adder block. All the remaining blocks require three inputs to generate the final sum bit. Although the design is complex compared to a simple ripple-carry adder, the basic principle remains the same, which helps select the self-checking approach.

Unlike the CSA, the parity prediction approach and DMR cannot be adopted for CPA due to the possibility of fault propagation. Therefore, we use the equivalence tester bit concept to design the self-checking CPA with the multi-error resilient feature. The least significant block is replaced with SHA. The remaining sum bits and their respective *Eqt* bits are computed using the sum and equivalence tester (SE) block, which uses the output bits of the last CSA block and the input carry bit (C_{i-1}) generated by the carry logic (CLL) block, as shown in Fig. 3(a). The SE block is also responsible for generating the final fault by comparing the internally generated sum and *Eqt* bit with the carry input of the next SE block (i.e., C_i). In the CLL block, the propagate and generate signals (i.e., P_i and G_i) are computed once for each bit position, whereas the computed values are shared internally for generating the intermediate carry bits using the internally generated carry (IGC) module in CLL. The logic diagram of P_i and G_i block is shown in Fig. 3b.

V. RESULTS AND BENCHMARK

This research proposes a self-checking MM architecture using self-checking CSA and CPA blocks. This section discusses the proposed self-checking MM architecture's area, power, and delay overhead, including CSA and CPA. The performance of each self-checking CSA and MM approach is evaluated by comparing it with the standard non-self-checking design. The designs are implemented using Verilog HDL and simulated using ModelSim. The synthesized results are obtained using the synopsis design compiler

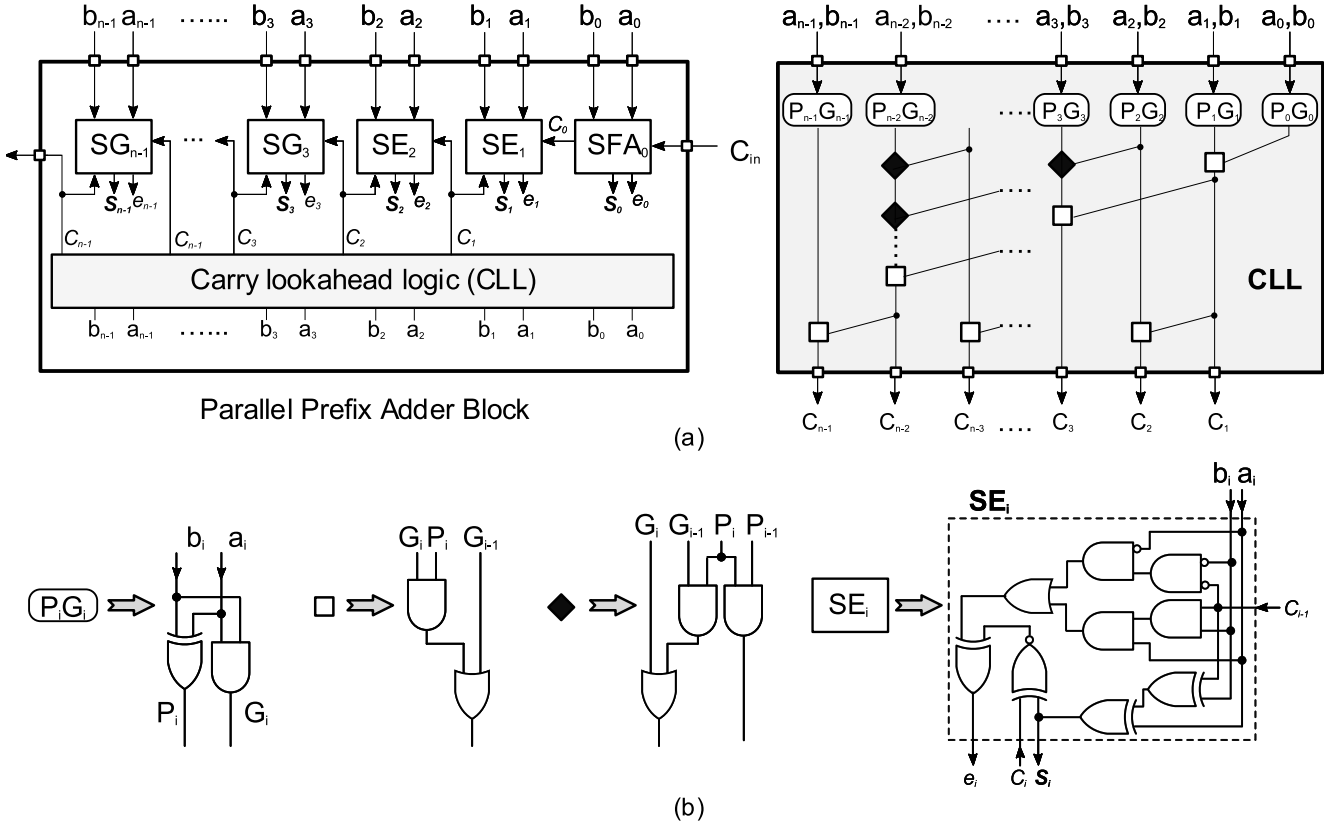


FIGURE 3. Self-checking parallel prefix (a) adder along with (b) logic gate diagram of internal blocks.

tool. A uniform Verilog-HDL coding style is adopted and synthesized with constant design constraints.

A. PERFORMANCE EVALUATION OF PROPOSED SELF-CHECKING CSA APPROACHES

The performance of each self-checking CSA is evaluated to determine the optimized solution for the final self-checking MM architecture. The area, power, and delay overhead, as compared to the standard CSA design, are shown in Table. 2. It should be noted that the discussion of comparison results for all standard and self-checking designs will only consider the 64-bit CSA architecture. In terms of area overhead, an SFA-based CSA requires 102.8% area overhead as compared to standard CSA design. However, the hardware penalty of parity and DMR-based CSA is 136.7% and 148.5% compared to standard CSA.

Unlike the area overhead of SFA-CSA, the power consumption is 59.6% more than the standard CSA. For the parity approach as well, the power consumption is not increased as much as the area, and the total power consumption in comparison to standard CSA is increased by 91.9%. However, in the case of DMR, the power consumption is as high as the area overhead and reaches 169.2% as compared to standard CSA. Hence, regarding area and power, SFA and parity-based CSA require fewer penalties than DMR.

Although the self-checking approaches do not affect the delay of the final sum and carry out, the delay for computing error might change depending on the approach. Therefore, the delay overhead needs to be observed before deciding on the optimized self-checking approach for CSA. In contrast to area and power consumption, DMR required the least delay overhead of 15.83%, which is 3.8% and 30.4% less than SFA and parity-based CSA. The area-delay product (ADP) and power-delay product (PDP) of each approach are shown in Fig. 4 (a) and (b). It is concluded that the SFA-based approach provides the least overhead in terms of area, power, ADP, and PDP as compared to the other two approaches. It can also be observed that DMR offers less ADP than the parity approach, whereas both DMR and parity require similar PDP overhead as compared to the standard CSA approach.

B. SYNTHESIS RESULTS FOR PROPOSED SELF-CHECKING MM ALGORITHM

MM architecture's hardware implementation mainly depends on CSA and CPA blocks. We analyze both these blocks independently to determine their respective optimized self-checking designs. The SFA-based self-checking approach is adopted for CSA because it requires optimal overhead compared to other methods. Similarly, an equivalence-tester-based self-checking approach is adopted for CPA because

TABLE 2. Area, Delay and Power consumption of standard CSA, SFA-CSA, PP-CSA and DMR-CSA.

Size(bit)	Self-checkingApproach	Area (um^2)			Delay(nS)	Power (mW)		
		no. of Cells	*no. of Net*	*Total*		*Internal*	*Switching*	*Total*
16	Standard CSA	162	353	2458.6	10.8	0.438	0.697	1.135
	SFA-CSA	322	641	4987.4	13.1	0.662	1.134	1.796
	PP-CSA	384	703	5790.9	18.4	1.109	1.053	2.162
	DMR-CSA	421	803	6012.6	12.6	1.222	1.926	3.148
32	Standard CSA	322	705	4917.2	11.2	0.873	1.394	2.267
	SFA-CSA	642	1281	9974.9	13.6	1.315	2.239	3.554
	PP-CSA	768	1407	11621.3	19.5	2.275	2.056	4.331
	DMR-CSA	868	1634	12222.8	13.1	2.203	3.755	5.958
64	Standard CSA	642	1409	9834.4	12	1.704	2.746	4.45
	SFA-CSA	1282	2561	19949.9	14.4	2.634	4.471	7.105
	PP-CSA	1536	2815	23282.2	19.9	4.542	3.999	8.541
	DMR-CSA	1732	3266	24445.7	13.9	4.418	7.566	11.984

it provides fault detection independent of the propagated carry. The self-checking MM architecture encapsulates these approaches, and the resultant overhead for 16-, 32-, and 64-bit MM architecture is shown in Table. 3. The larger bit-width of 128 or 256 can be accommodated using cascaded 32-bit or 64-bit MM architecture, which may introduce an insignificant delay overhead because the CSA will perform computation in parallel irrespective of the carry input from the previous stage. The only overhead that would occur is due to the cascaded CPA present outside the loop because it requires the Cout of the last block. The percentage value of area and power overhead will not change compared to the standard MM architecture.

The proposed 64-bit self-checking MM architecture requires 43.5% area overhead compared to the standard MM architecture. With less than 50% area overhead, the power consumption of the proposed 64-bit self-checking design is increased by only 10.9% with a delay overhead of 33.8%. However, the ADP and PDP for 64-bit self-checking MM architecture are increased by 92% and 48.4%, respectively, as shown in Fig. 5(a) and (b).

The self-checking architecture for MM has not been investigated to the best of our knowledge. In the work of [16], a residue code-based self-checking modulo multiplier is presented with an application in cryptography. However, the reported architecture does not have complete self-checking and fault localization because fault identification depends on the check base value. Moreover, the reported scheme requires additional delay to transform the final output for comparison. A detailed comparison with respect to area and fault coverage is presented in Table. 4. It can be observed that our proposed approach with fault detection and localization requires 73.6% less area overhead with 18.8% delay efficiency as compared to the residue code-based modulo multiplier in [16]. To further verify the efficiency of our proposed solution, we analyzed some recent articles as a case study. However, the problem is that some of them used FPGA resources for overhead computation. In order to make an impartial comparison, we also used Quartus to compute the FPGA resources for the 64-bit MM architecture before and after self-checking. Our proposed solution requires only 4.2% logic elements overhead, whereas the number of registers remain the same. In the work of [33], the Hamming code-based self-checking approach for block cipher requires an overhead of 16% LUT and 25.5% Slice registers. A detailed comparison between self-checking techniques for differential fault analysis has been presented in [45]. It can be observed from their analysis that most of the self-checking approaches for securing the cipher require a logic overhead of more than 10%. Moreover, a hybrid self-checking approach is adopted by [46] for hierarchical multipliers with area overhead increased by 71.1%. These reported research findings on self-checking multipliers or crypto-processors verify that our proposed solution provides minimum overhead.

C. FAULT COVERAGE

The fault injected into the system might not necessarily affect the final output due to the inherent fault-masking ability of the circuit. For example, if an input at the AND gate is logic low, the second input generated from faulty circuitry cannot affect the output. This research covers those faults which are reflected in the output.

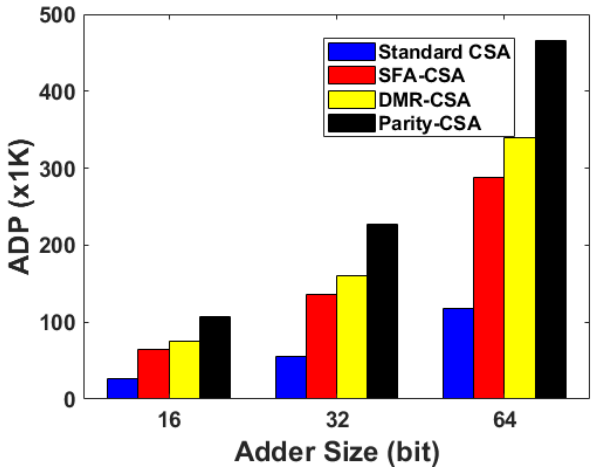
1) SELF-CHECKING CSA

In this research, the self-checking CSA is investigated using three different approaches: SFA, PP, and DMR. Each of them has its fault coverage and limitations.

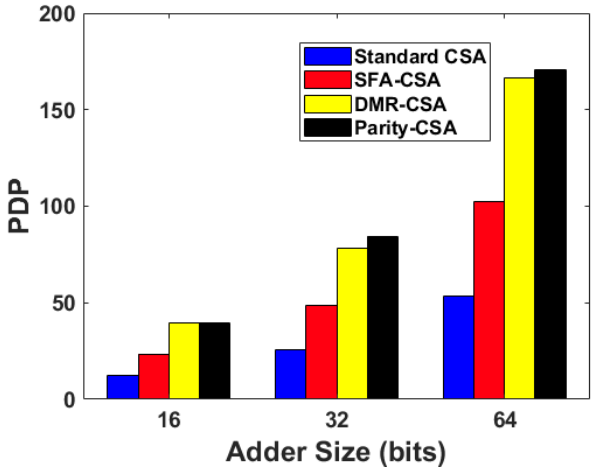
The AND gate connected to one of the CSA's operands is constructed using DMR so that each output of DMR engages in the computation of the rivalry submodules. The fault in the AND gate causes a mismatch in the output of the rivalry submodules, which the checker can detect. Therefore, the

TABLE 3. Area, Delay and Power consumption of standard MM and proposed self-checking MM architecture.

Size(bit)	Architecture	Area (um^2)			Delay(nS)	Power (mW)		
		no. of Cells	*no. of Net*	*Total*		*Internal*	*Switching*	*Total*
16	Standard MM	576	1164	10692.8	8.27	0.1966	0.0055	0.2020
	Proposed self-checking MM	787	1618	14541	11.31	0.2323	0.0252	0.2576
32	Standard MM	1088	2285	18887.5	8.87	0.3444	0.0106	0.3551
	Proposed self-checking MM	1507	3187	26634.4	11.98	0.4311	0.0604	0.4916
64	Standard MM	2172	4671	35669.8	9.47	0.8250	0.1197	0.9447
	Proposed self-checking MM	3007	6459	51203	12.67	0.8867	0.1614	1.0482



(a)



(b)

FIGURE 4. (a) ADP and (b) PDP of standard and proposed self-checking CSA approaches.

scope of the checker in any of the approaches is not limited to identify the fault within the adder circuitry but to also reflect the fault occurring at the AND gate associated with one of the input terminals. The use of a checker to identify the faults in the AND gate minimizes the need for comparators, which is required in a duplex system for fault identification. This approach can detect 50% of faults in AND gates, whereas it is possible to use a comparator to design a complete self-checking AND gate.

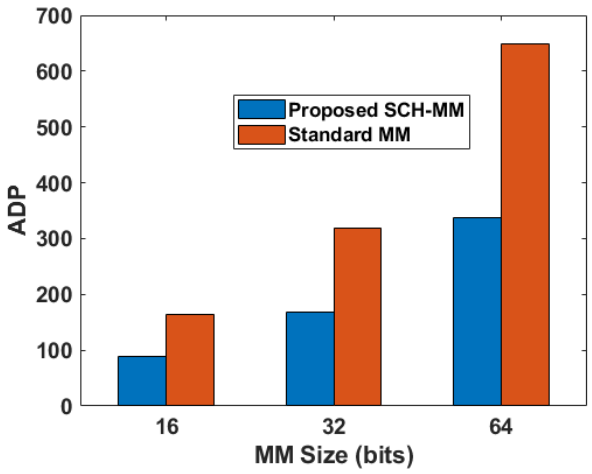
TABLE 4. Comparison between self-checking modulo multiplier and our proposed approach.

Parameters	[16]	Proposed Approach
Technology	0.18um CMOS	
Topology	Modulo Multiplier	MM
Area (um^2)	194624	51203
Delay(ns)	15.62	12.67
Fault Localization	Not Possible	Possible
Fault Type	Single	Multiple
Checker	Centralized	Distributed

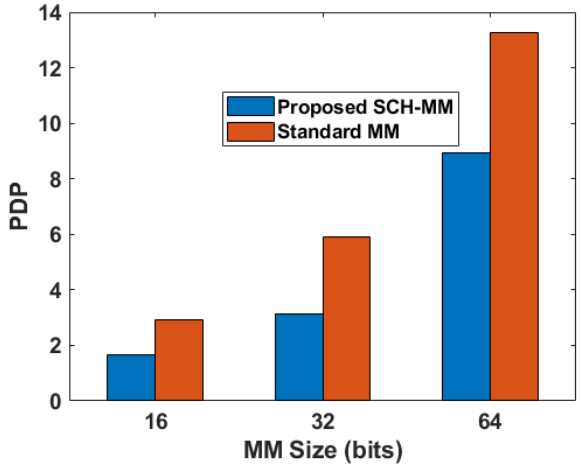
The SFA-based self-checking CSA can identify any fault with the condition that each SFA has a single fault at a time. Moreover, it can localize the fault due to a distributed self-checking mechanism. In contrast, the fault diagnosis in the PP approach is limited to either an even or odd number of erroneous bits and cannot be localized. Both SFA and PP possess diversity and are protected against common mode failure. In contrast, DMR can detect the fault without localization; hence, the scheme fails with common mode failure due to the absence of diversity. Constructing the DMR with built-in diversity is possible, but it may cause additional overhead or performance degradation.

2) SELF-CHECKING MM ARCHITECTURE

The MM architecture is constructed using SFA-based CSA and self-checking carry prefix adder. Similar to CSA, CPA can detect all kinds of faults with the condition that a single SE and IGC (responsible for generating C_{in} for the neighboring SE) have a single fault at a time. In the presented self-checking architecture, the faulty carry bit generated by the IGC will be detected in the first place; hence, the fault will not create a false alarm for other SE-IGC pairs. Therefore, the CLL architecture is designed using shared logic, which is not recommended for other self-checking architectures because of fault propagation. The fault localization ability



(a)



(b)

FIGURE 5. (a) ADP and (b) PDP of standard and proposed self-checking MM approach.

in the proposed architecture can significantly improve the post-error decision capability of crypto-processors. In case of an unknown fault location, the system may trash all the resources for security concerns. With the proposed solution, it is possible to adopt the most suitable solution based on the scenario because it can detect the fault and locate the exact position of its occurrence. For example, the system can re-execute the instructions to check whether the fault occurs at the same location; if so, the system may replace the faulty module to bring back the device's normal operation.

3) LIMITATIONS

The fault coverage is limited to the faults injected at the logic level of the MM architecture, which is possible with active physical attacks. However, it is possible to apply the proposed solution in existing countermeasures for passive side-channel and timing attacks like digital serial [47] or high-radix MM [48]. The corrupted input bits caused by the faults occurring in the memory unit are not considered in this research. Each register is assumed to be self-checking with a check bits approach like parity bit or cycle redundancy code. Moreover, the fault will be detected with the condition that each SFA in the CSA and SE-IGC pair in CPA have a single fault at a time. However, this assumption is valid because the area of SFA and SE-unit along with the respective carry bit generator is negligible compared to the overall size of the multiplier.

VI. CONCLUSION

In this research, we proposed a self-checking hardware architecture for the MM algorithm, which can handle multiple faults simultaneously because of the distributed fault detection mechanism. In addition, a novel parity prediction approach is proposed, which can be used in cases where CSA blocks are used repeatedly in a loop without the CPA layer. The performance of the MM algorithm is mainly dependent on CSA and CPA blocks. Therefore, we analyzed each individually to construct an optimized self-checking approach for MM.

Due to the absence of a carry propagation chain, we examined three different fault prognosis approaches for CSA. It is observed that the SFA-based self-checking CSA approach provides 18.3% and 14.3% area efficiency as compared to the DMR and PP approaches. In terms of power, the SFA approach is 40.71% and 16.8% more efficient than DMR and PP. As a result of the analysis, the SFA-based self-checking CSA approach is used for MM architecture because it provides the optimal area and power overhead of 102.8% and 59.6% compared to standard CSA design. In contrast to CSA, the carry chain in CPA limits the self-checking approach due to fault propagation. Therefore, an equivalence tester-based self-checking is used for the carry-prefix adder design. The proposed approach can detect multiple faults simultaneously, and the checking mechanism is independent of the corrupted input carry bit.

The proposed self-checking MM architecture is designed using the SFA-based CSA and equivalence tester-based CPA blocks. The proposed design can detect multiple faults simultaneously with an area overhead of 43.5% compared to the standard MM design. The power consumption of the proposed self-checking architecture is only 10.9% higher than the traditional MM architecture. The fault diagnosis process operated in parallel with the operation of standard MM architecture and did not affect the latency of the final output bits. However, the dependency of error computation on MM output creates a 33.8% delay overhead for generating the final error output.

ACKNOWLEDGMENT

Open Access funding provided by the Qatar National Library. This work was supported by NPRP under Grant NPRP13S-0212-200345 from the Qatar National Research Fund (a member of Qatar Foundation). The findings herein reflect the work and are solely the responsibility of the authors.

REFERENCES

[1] S. Vollala and N. Ramasubramanian, "Energy efficient modular exponentiation for public-key cryptography based on bit forwarding techniques," *Inf. Process. Lett.*, vol. 119, pp. 25–38, Mar. 2017.

[2] N. Nedjah, L. M. Mourelle, S. Raposo, and M. Santana, "Massively parallel modular exponentiation method and its implementation in software and hardware for high-performance cryptographic systems," *IET Comput. Digit. Techn.*, vol. 6, no. 5, pp. 290–301, Sep. 2012.

[3] A. A. H. Abd-Elkader, M. Rashdan, E.-S.-A. M. Hasaneen, and H. F. A. Hamed, "Efficient implementation of Montgomery modular multiplier on FPGA," *Comput. Electr. Eng.*, vol. 97, Jan. 2022, Art. no. 107585.

[4] K. V. Reddy, C. S. Singh, V. Desalphine, and D. Selvakumar, "A low latency Montgomery modular exponentiation," *Proc. Comput. Sci.*, vol. 171, pp. 800–809, Jan. 2020.

[5] A. Krikun and A. Levina, "Parallelized Montgomery exponentiation in GF (2^k) for Diffie–Hellman key exchange protocol," *Eng. Lett.*, vol. 29, no. 2, pp. 645–649, 2021.

[6] Y. Zhao and C. Zhao, "FPGA-based hardware architecture for Montgomery modular exponentiation algorithm," in *Proc. Int. Conf. Intell. Manuf. Ind. Big Data (ICIMIBD)*, Dec. 2022, pp. 77–81.

[7] A. A. H. Abd-Elkader, M. Rashdan, E. A. M. Hasaneen, and H. F. A. Hamed, "FPGA-based optimized design of Montgomery modular multiplier," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 6, pp. 2137–2141, Jun. 2021.

[8] T. Abirami, S. Saravanan, A. Rajeshkumar, and K. M. Santhosh, "FPGA-based optimized design of Montgomery modular multiplier using Karatsuba algorithm," in *Proc. 2nd Int. Conf. Electron. Renew. Syst. (ICEARS)*, Mar. 2023, pp. 131–135.

[9] M. Huang, K. Gaj, and T. El-Ghazawi, "New hardware architectures for Montgomery modular multiplication algorithm," *IEEE Trans. Comput.*, vol. 60, no. 7, pp. 923–936, Jul. 2011.

[10] B. Yuce, P. Schaumont, and M. Witteman, "Fault attacks on secure embedded software: Threats, design, and evaluation," *J. Hardw. Syst. Secur.*, vol. 2, no. 2, pp. 111–130, Jun. 2018.

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

[12] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, H.-J. Wunderlich, and J. Henkel, "Aging resilience and fault tolerance in runtime reconfigurable architectures," *IEEE Trans. Comput.*, vol. 66, no. 6, pp. 957–970, Jun. 2017.

[13] M. Nagata, T. Miki, and N. Miura, "Physical attack protection techniques for IC chip level hardware security," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 1, pp. 5–14, Jan. 2022.

[14] M. M. Sravani and S. A. Durai, "Attacks on cryptosystems implemented via VLSI: A review," *J. Inf. Secur. Appl.*, vol. 60, Aug. 2021, Art. no. 102861.

[15] C. Shepherd, K. Markantonakis, N. van Heijningen, D. Aboulkassimi, C. Gaine, T. Heckmann, and D. Naccache, "Physical fault injection and side-channel attacks on mobile devices: A comprehensive analysis," *Comput. Secur.*, vol. 111, Dec. 2021, Art. no. 102471.

[16] W. Hong, R. Modugu, and M. Choi, "Efficient online self-checking modulo 2^{n+1} multiplier design," *IEEE Trans. Comput.*, vol. 60, no. 9, pp. 1354–1365, Sep. 2011.

[17] A. Aghaie, A. Moradi, S. Rasoolzadeh, A. R. Shahmirzadi, F. Schellenberg, and T. Schneider, "Impeccable circuits," *IEEE Trans. Comput.*, vol. 69, no. 3, pp. 361–376, Mar. 2020.

[18] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of eliminating errors in cryptographic computations," *J. Cryptol.*, vol. 14, no. 2, pp. 101–119, Mar. 2001.

[19] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proc. IEEE*, vol. 100, no. 11, pp. 3056–3076, Nov. 2012.

[20] C. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults (extended abstract)," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1997, pp. 37–51.

[21] A. Mukherjee and A. S. Dhar, "Real-time fault-tolerance with hot-standby topology for conditional sum adder," *Microelectron. Rel.*, vol. 55, nos. 3–4, pp. 704–712, Feb. 2015.

[22] E. Trichina and R. Korkikyan, "Multi fault laser attacks on protected CRT-RSA," in *Proc. Workshop Fault Diagnosis Tolerance Cryptography*, Aug. 2010, pp. 75–86.

[23] J. Breier, D. Jap, and C.-N. Chen, "Laser profiling for the back-side fault attacks: With a practical laser skip instruction attack on AES," in *Proc. 1st ACM Workshop Cyber-Physical Syst. Secur.*, Apr. 2015, pp. 99–103.

[24] L. Song and L. Hu, "Differential fault attack on the PRINCE block cipher," in *Proc. Int. Workshop Lightweight Cryptogr. Secur. Privacy (CSP)*, 2013, pp. 43–54.

[25] C. Giraud and H. Thiebeauld, "A survey on fault attacks," in *Smart Card Research and Advanced Applications VI*. Berlin, Germany: Springer, 2004, pp. 159–176.

[26] A. Baksi, S. Bhasin, J. Breier, D. Jap, and D. Saha, "A survey on fault attacks on symmetric key cryptosystems," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–34, 2022.

[27] J. Breier and X. Hou, "How practical are fault injection attacks, really?" *IEEE Access*, vol. 10, pp. 113122–113130, 2022.

[28] S. Weidling, E. S. Sogomonyan, and M. Goessel, "Error correction of transient errors in a sum-bit duplicated adder by error detection," in *Proc. Euromicro Conf. Digit. Syst. Design*, Sep. 2013, pp. 855–862.

[29] N. Kito and N. Takagi, "Concurrent error detectable carry select adder with easy testability," *IEEE Trans. Comput.*, vol. 68, no. 7, pp. 1105–1110, Jul. 2019.

[30] M. Valinataj, "Fault-tolerant carry look-ahead adder architectures robust to multiple simultaneous errors," *Microelectron. Rel.*, vol. 55, no. 12, pp. 2845–2857, Dec. 2015.

[31] M. A. Akbar, B. Wang, and A. Bermak, "Self-repairing hybrid adder with hot-standby topology using fault-localization," *IEEE Access*, vol. 8, pp. 150051–150058, 2020.

[32] J. Gu, Y. Wang, C. Hu, and Z. Luo, "A novel method for predicting fault labels of roller bearing by generalized Laplacian matrix," *IEEE Access*, vol. 9, pp. 14330–14339, 2021.

[33] F. E. Potestad-Ordónez, E. Tena-Sánchez, R. Chaves, M. Valencia-Barrero, A. J. Acosta-Jiménez, and C. J. Jiménez-Fernández, "Hamming-code based fault detection design methodology for block ciphers," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[34] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight high-performance fault detection scheme for the advanced encryption standard using composite fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 1, pp. 85–91, Jan. 2011.

[35] 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.

[36] C. K. Koc, T. Acar, and B. S. Kaliski, "Analyzing and comparing Montgomery multiplication algorithms," *IEEE Micro*, vol. 16, no. 3, pp. 26–33, Jun. 1996.

[37] A. F. Tenca and C. K. Koc, "A scalable architecture for modular multiplication based on montgomery's algorithm," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1215–1221, Sep. 2003.

[38] J. C. Néto, A. F. Tenca, and W. V. Ruggiero, "A parallel and uniform k-partition method for Montgomery multiplication," *IEEE Trans. Comput.*, vol. 63, no. 9, pp. 2122–2133, Sep. 2014.

[39] F. Bernard, "Scalable hardware implementing high-radix Montgomery multiplication algorithm," *J. Syst. Archit.*, vol. 53, nos. 2–3, pp. 117–126, Feb. 2007.

[40] S.-R. Kuang, J.-P. Wang, K.-C. Chang, and H.-W. Hsu, "Energy-efficient high-throughput Montgomery modular multipliers for RSA cryptosystems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 1999–2009, Nov. 2013.

[41] S.-R. Kuang, K.-Y. Wu, and R.-Y. Lu, "Low-cost high-performance VLSI architecture for Montgomery modular multiplication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 434–443, Feb. 2016.

[42] K. Manochehri and S. Pourmozafari, "Modified Radix-2 Montgomery modular multiplication to make it faster and simpler," in *Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC)*, 2005, pp. 598–602.

[43] C. D. Walter, "Montgomery exponentiation needs no final subtractions," *Electron. Lett.*, vol. 35, no. 21, pp. 1831–1832, 1999.

[44] A. K. Panda, R. Palisetty, and K. C. Ray, "High-speed area-efficient VLSI architecture of three-operand binary adder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 11, pp. 3944–3953, Nov. 2020.

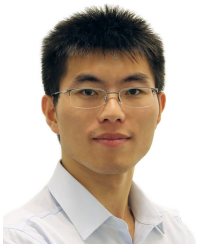
[45] F. E. Potestad-Ordóñez, E. Tena-Sánchez, A. J. Acosta-Jiménez, C. J. Jiménez-Fernández, and R. Chaves, "Hardware countermeasures benchmarking against fault attacks," *Appl. Sci.*, vol. 12, no. 5, p. 2443, Feb. 2022.

[46] M. Valinataj and A. Jantsch, "Hierarchical multipliers: A framework for high-speed multiple error detecting architectures," *Microelectron. J.*, vol. 125, Jul. 2022, Art. no. 105459.
[47] S. S. Erdem, T. Yanik, and A. Çelebi, "A general digit-serial architecture for Montgomery modular multiplication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1658–1668, May 2017.
[48] A. Miyamoto, N. Homma, T. Aoki, and A. Satoh, "Systematic design of RSA processors based on high-radix Montgomery multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 7, pp. 1136–1146, Jul. 2011.

MUHAMMAD ALI AKBAR received the B.Eng. degree (Hons.) in electronic engineering from NED University, Karachi, Pakistan, in 2011, and the M.Sc. degree in computer engineering from Chosun University, South Korea, in 2014. He is currently pursuing the Ph.D. degree in computer engineering with Hamad Bin Khalifa University, Qatar.

In 2014, he joined Qatar University as a Research Assistant to work on different research project related to hardware design and machine learning applications. His main research interests include fault localization, self-reliable systems, sensor design, and artificial intelligence.

ABDULLATIF SHIKFA is currently the Cyber Security Department Head of the College of Computing and Information Technology (CCIT), University of Doha for Science and Technology (UDST). He played a key role in the transformation of UDST and was a major Contributor to the launch of the new bachelor programs with CCIT. Before joining UDST, he held both research and industry positions, including a Scientific Advisor and the Deputy Head of the Security Research Department, Bell Laboratories, the Technical Project Manager and a Security Expert with Thales, and a Research Assistant Professor with Qatar University. His research interest includes cyber security, with a particular emphasis on applied cryptography.



BO WANG (Senior Member, IEEE) received the B.Eng. degree (Hons.) in electrical engineering from Zhejiang University, Hangzhou, China, in 2010, and the M.Phil. and Ph.D. degrees in electronic and computer engineering from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2012 and 2015, respectively.

In 2015, he joined HKUST as a Postdoctoral Researcher and was a Visiting Research Scientist with Massachusetts Institute of Technology, in 2016. In 2017, he joined Hamad Bin Khalifa University as a Founding Faculty Member. He is currently an Assistant Professor. His research interests include analog mixed-signal circuits, sensor and sensor interfaces, and heterogeneous integrated systems for in vitro/vivo health monitoring. He was a recipient of the IEEE ASP-DAC Best Design Award, in 2016. He serves as a Technical Committee Member for the IEEE CAS Committee on Sensory Systems and an Associate Editor for the IEEE Sensors Journal.

AMINE BERMAK (Fellow, IEEE) received the master's and Ph.D. degrees in microelectronics and microsystems from Paul Sabatier University, Toulouse, France, in 1994 and 1998, respectively.

He joined the ECE Department, The Hong Kong University of Science and Technology (HKUST), where he held all academic ranks and subsequently promoted to a Full Professor. He was also the ECE Associate Head of Research and Postgraduate Studies. Currently, he is with Hamad Bin Khalifa University, Qatar Foundation, Qatar, holding a Full Professor appointment and the Associate Dean. He taught 25 different courses at the undergraduate and post-graduate levels. He has published over 350 articles in journals, book chapters and conference proceedings, and designed over 30 chips. He has graduated 25 Ph.D. and 20 master's students.

Prof. Bermak is a fellow of IEEE Distinguished Lecturer. He is recognized as the world-leading author in the sensors area and the inventor of time-domain sensing. For his excellence and outstanding contribution to teaching, he was nominated for the 2013 Hong Kong UGC Best Teacher Award (for all HK universities). He was a recipient of the 2011 University Michael G. Gale Medal for distinguished teaching (Highest University-Wide Teaching Award). He was also a two-time recipient of the "Engineering School Teaching Excellence Award" from HKUST, in 2004 and 2009, respectively. He has received six distinguished awards, including the "Best University Design Contest Award" at ASP Design Automation Conference (DAC), in Macau 2016, "Best Student Paper Award" at IEEE International Symposium on Circuits and Systems ISCAS 2010, the 2004 "IEEE Chester Sall Award" from IEEE Consumer Electronics Society, the IEEE Service Award from IEEE Computer Society, and the "Best Paper Award" at the 2005 International Workshop on System-On-Chip for Real-Time Applications. He has served on the editorial board for IEEE Transactions on Very Large Scale Integration (VLSI) Systems and the *Sensors* journal. He is also serving on the editorial board for IEEE Transactions on Biomedical Circuits and Systems and the IEEE Transactions on Electron Devices (IEEE TED). He is also an Editor of *Scientific Report* (Nature).

Open Access funding provided by 'Qatar National Library' within the CRUI CARE Agreement