Hardware Circuits and Systems Design for Post-Quantum Cryptography—A Tutorial Brief

Jiafeng Xie, *Senior Member, IEEE*, Wenfeng Zhao, *Senior Member, IEEE*, Hanho Lee, *Senior Member, IEEE*, Debapriya Basu Roy, *Member, IEEE*, and Xinmiao Zhang, *Senior Member, IEEE*

***Abstract*—Due to the increasing threats from possible large-scale quantum computers, post-quantum cryptography (PQC) has drawn significant attention from various communities recently. In particular, along with the National Institute of Standards and Technology (NIST) PQC standardization process, more works have gradually switched to the PQC hardware implementations. Following this trend, this tutorial brief, led by a group of experts in the field, aims to deliver a comprehensive tutorial on hardware circuits and systems design for PQC. After introducing primary arithmetic operations and algorithmic features of different PQC, we introduced related PQC hardware circuits and systems design techniques (from component to system levels). Future research and directions are also provided. This tutorial will provide useful information for the TCAS-II community and the broader Circuits and Systems Society.**

***Index Terms*—Arithmetic operation, hardware cryptographic design for PQC, circuits and systems design techniques.**

I. Introduction

ALONG with the rapid progress in building large-scale quantum computers, post-quantum cryptography (PQC) has gained substantial attention from various communities recently [1], [2], [3]. The National Institute of Standards and Technology (NIST) has already started the PQC standardization and selected algorithms for standardization last July [4]. On August 24, 2023, NIST released the Federal Information Processing Standards (FIPS) for selected three algorithms: KYBER, Dilithium, and SPHINCS$^+$ (standards for FALCON will be released later) [5]. Following this pace, more research has switched to hardware circuits and systems design for PQC.

Overview: Accompanying the innovations of the crypto-design community, standardization efforts from governmental institutions also become one of the major driving forces to facilitate PQC development. One such milestone was the NIST's announcement of the PQC standardization process in 2016, which led to the selection of four algorithms (July 2022) [6] and the release of FIPS [5]. Still, three algorithms are under consideration as NIST's fourth-round candidates [4]. To increase the diversity and reduce the risk of one algorithm being suddenly broken, NIST has also announced an additional round of digital signature scheme standardization [7].

Background of PQC Hardware Design: Typically, hardware design for PQC takes a longer development cycle than the software counterpart [3]. As a result, PQC hardware works usually fall behind the crypto community's research pace [8]. Overall, there are three types of hardware designs for PQC, i.e., high-level synthesis (HLS), hardware-software co-design, and full-hardware design, each with unique features over the others [3]. Hardware-software co-design involves simultaneous development of hardware and software, with the intention of optimizing the design through close collaboration between hardware and software teams. This approach allows for fast development period and deployment while taking the least advantage of parallel computation [9]. HLS allows designers to describe a hardware system at a high level of abstraction, typically using a high-level programming language like C, C++, or System C, while this approach also does not take full advantage of hardware processing capacity [10]. Standing from the perspective of the Circuits and Systems (CAS) community [11], hardware circuits and systems design for PQC refers to the designing of novel hardware structures for certain algorithmic operations/components within a specific algorithm and the overall scheme-level architectural building and data-flow optimization. More importantly, this approach allows full manipulation of basic circuit units and takes full advantage of hardware processing capacity [12].

As cryptographic hardware design traditionally belongs to one of the most important fields in the CAS community [11], this brief follows state-of-the-art research progress to present this tutorial. Specifically, (i) we mainly focus on the NIST-selected ones, as well as those promising schemes that are still under consideration; (ii) the focused hardware design techniques are mostly in the algorithm-to-architecture mapping and register-transfer-level (RTL) layers, by which the majority of research works are focusing currently; (iii) major components and representative PQC schemes are comprehensively covered (yet briefly) due to page limitations; (iv) hardware

Manuscript received 31 August 2023; revised 3 November 2023 and 3 December 2023; accepted 10 January 2024. Date of publication 24 January 2024; date of current version 5 March 2024. This work was supported by the Villanova University's Falvey Memorial Library Scholarship Open Access Reserve (SOAR) Fund. The work of Jiafeng Xie was supported in part by NIST under Grant 60NANB20D203, and in part by NSF under Grant SaTC-2020625. The work of Xinmiao Zhang was supported in part by NSF under Grant 2052641. This brief was recommended by Associate Editor Y. Ha. *(Corresponding author: Jiafeng Xie.)*

Jiafeng Xie is with the Department of Electrical and Computer Engineering, Villanova University, Villanova, PA 19085 USA (e-mail: jiafeng.xie@villanova.edu).

Wenfeng Zhao is with the Electrical and Computer Engineering Department, Binghamton University, Binghamton, NY 13902 USA (e-mail: wzhao@binghamton.edu).

Hanho Lee is with the Department of Electrical and Computer Engineering, Inha University, Incheon 22212, Republic of Korea (e-mail: hhlee@inha.ac.kr).

Debapriya Basu Roy is with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kanpur 208016, India (e-mail: dbroy@cse.iitk.ac.in).

Xinmiao Zhang is with the Department of Electrical and Computer Engineering, Ohio State University, Columbus, OH 43210 USA (e-mail: zhang.8952@osu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2024.3357836.

Digital Object Identifier 10.1109/TCSII.2024.3357836

security-related topics like side-channel attacks are slightly described as they are out of the scope of this tutorial.

The rest of this tutorial is arranged as follows. Section II focuses on the algorithmic and arithmetic aspects. Section III presents hardware circuits and systems design techniques at different layers. Section IV gives the remaining problems and future directions. Conclusions are drawn in Section V.

II. Brief Description of PQC Algorithmic Features & Arithmetic Operations

A. General PQC Schemes

Hardware implementations for PQC can be traced to the earlier works for lattice-based and code-based schemes. Lattice-based schemes mainly are based on Learning-with-Errors (LWE)/Ring-LWE and/or ring of polynomials of degree N (NTRU) problems [13], [14], [15], while code-based PQC relies on error-correcting codes to ensure its security [16]. Early papers of [17], [18], [19] presented efficient ways to implement lattice-based PQC and were followed by [20], [21], [22], [23], [24], [25], [26], [27]. Code-based hardware works were [28], [29] and recently [30], [31].

B. NIST-Selected PQC Schemes

Among NIST-selected PQC, Kyber [32], Dilithium [33], and Falcon [34] belong to lattice-based PQC [6], while SPHINCS$^+$ is a hash-based scheme [35].

Kyber is a Module-Learning-with-Errors (MLWE)-based scheme [32]. The public-key encryption scheme (chosen plaintext attack, CCA-secure) is built and then the Fujisaki-Okamoto transform is used to obtain its ciphertext adaptive chosen ciphertext attack secure key-encapsulation mechanism (KEM) version. Besides, Kyber builds the Number Theoretic Transform (NTT) into its operations to obtain low-complexity computation. Its hardware works include [12], [36], [37], [38], [39], [40], [41].

Dilithium is a module lattice-based scheme based on the Fiat-Shamir paradigm [42]. Similar to Kyber, NTT is used to transfer Dilithium's major operations to NTT domain to lower the computation complexity. Hardware designs for Dilithium can be seen at [43], [44], [45], [46], [47], [48].

Falcon is also a NIST-selected lattice-based signature scheme, which is based on the short integer solution problem over NTRU lattices [34]. Compared with Dilithium, Falcon has shorter keys and signatures [42], but requires floating-point arithmetic. Hardware-implemented Falcon has not been reported except a few component works [49], [50].

SPHINCS$^+$ is a hash-based signature scheme, which was selected because of its workable signature scheme with solid security and is based on an entirely different assumptions than other selected signature schemes [42]. Again, hardware design for SPHINCS$^+$ is very rare (mostly the component [51], [52]).

C. NIST Round 4 PQC Submissions

Among the NIST round 4 submissions, BIKE (Bit Flipping Key Encapsulation) [53], Classic McEliece [54], and HQC (Hamming Quasi-Cyclic) [55] are all KEM-based code-based PQC. BIKE is based on binary linear quasi-cyclic moderate density parity check (QC-MDPC) codes [56], and is considered as having the most competitive performance among the non-lattice based KEMs [42]. Classic McEliece uses a binary Goppa code in the McEliece variant cryptosystem to obtain CCA security [42]. HQC is based on QC-MDPC codes and uses LWE-like scheme to build its protocol. The hardware designs for these schemes are relatively fewer, i.e., [57], [58], [59], [60], [61], [62], [63], [64], [65], [66].

NIST has also released another call for additional signature schemes [7], and 40 algorithms have been submitted. Due to their limited exposure time, very few hardware designs have been reported and we thus do not explicitly analyze them here.

D. Major Arithmetic Operations

Polynomial Multiplication: Polynomial multiplication is considered as one of the most important components in lattice-based PQC (LWE/Ring-LWE/variants [67], [68], [69] and NTRU [70], [71]). Hardware-implemented polynomial multiplications are mostly based on NTT [72], a complexity of $O(n\log n)$. The hardware design challenge lies in the data-flow optimization along with related resource usage tradeoffs. Subcomponents like point-wise multiplier and modular reduction unit are also critical to the overall efficiency.

Sparse polynomial multiplication is a special polynomial multiplication over $\mathbb{F}_2$, used in BIKE [53] and HQC [55]. Its design challenge comes from the super-large dimension yet a very small number of '1's, e.g., $n = 57,637$ with only 149 '1's in HQC [55]. Nevertheless, [63], [64], [66] have proposed strategies to implement it efficiently.

Sampler: Sampling for lattice-based PQC can be binomial distribution (Kyber [32]), uniform distribution (Dilithium [33]), or Gaussian distribution (Falcon [34]). Hardware binomial and uniform samplers are relatively easy [12], [44], but hardware sampler for Falcon is rare [73].

Sampling for code-based schemes like BIKE/HQC is a bit challenging due to the sparsity in the related polynomial multiplication. Effort like low failure probability method has been used to implement such sampler [63]. Recently, a fixed-weight sampler was proposed in [74] with zero failure rate.

Other Important Components: Falcon involves other special components, such as Falcon tree, floating-point arithmetic, and Fast Fourier Transform (FFT), whose dedicated hardware structure design needs significant efforts.

For the code-based schemes, polynomial inversions [53], encoder/decoder units [75], etc., are also involved.

SPHINCS$^+$ is based on an entirely different problem, and its hardware design has not been explored well in the literature. Though its original protocol (SPHINCS) was implemented with hardware [76], SPHINCS$^+$ has developed its unique algorithmic features [35] that previous design does not extend.

III. Brief Tutorial of Hardware Circuits and Systems Design Techniques for PQC

A. Component-Level Design Techniques

The efficiency of a PQC hardware accelerator is closely related to various functional blocks that execute specific arithmetic operations. We have listed several major operations and corresponding hardware designs here (see also Table I).

NTT/INTT Module: From the mathematical standpoint, NTT is the Discrete Fourier Transform that operates over finite field, For $B = \sum_{i=0}^{n-1} b_i x^i$, we have

$$f_i = \sum_{j=0}^{n-1} b_i \omega^{ij} \bmod q, \tag{1}$$

where ω is the twiddle factor [77]. Then $F = \sum_{i=0}^{n-1} f_i x^i$ is the corresponding polynomial in the frequency domain. Using

TABLE I
SOME REFERENCE WORKS ABOUT MAJOR COMPONENTS

Components	Reference Works
NTT/INTT	[12], [36], [37], [78]–[81]
Sparse Poly. Multiplier.	[62]–[64], [66]
Sampler	[12], [32]–[34], [44], [62], [65], [73], [74]

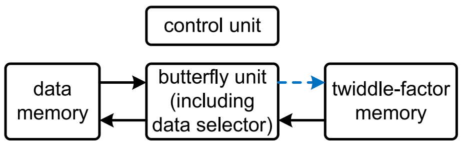

Fig. 1. General structure of the NTT/INTT module.

NTT transformation, the original polynomial multiplication becomes point-wise multiplications [77]. Of course, INTT is needed to transfer the result back to the integer domain, which follows the same format as (1) except that the twiddle factor is ω^{-1} and n^{-1} is also multiplied to each output result.

A typical hardware module for NTT/INTT computation can be shown in Fig. 1, where the butterfly unit is determined by what type of computation strategy is used, i.e., CT (Cooley-Tukey) or GS (Gentleman-Sande) [82], [83]. Through the control unit's precise coordination, the twiddle factor is dynamically updated to be multiplied with the corresponding value from the data memory in the butterfly unit and then the result is sent back to the data memory. Note the twiddle factor memory can be ROMs/registers stored with pre-calculated values or RAMs/FIFOs along with related computational units to produce the twiddle factors dynamically (indicated by the dotted arrow). Besides, NTT and INTT can be combined as a unified unit to facilitate hardware resource saving [78]. We have summarized NTT hardware implementations into Table II, organized by storage type (in-place), pre-processing, and post-processing. In an in-place NTT, the storage unit size corresponds to that of the transformed polynomial, with some exceptions that involve additional buffers or duplicate memory. Pre-processing and post-processing primarily involve the bit-reverse operation. The necessity of these operations in the NTT architecture depends on the specific structure and address access pattern. Some implementations may require neither operation, either one, or both (Table II). There also exist advanced techniques that coefficient memory can be split into two parts, namely ping-pong memory, to accelerate data fetch/store for multiple processing elements [79], [80], [81].

Important cells inside the butterfly unit include data selector (determines which data from which memory is being processed), point-wise multiplier, and modular reduction cell. The point-wise multipliers can be implemented by DSP cores on the field-programmable gate array (FPGA) platform (but the strategy to reduce the complexity of point-wise multipliers, e.g., Karatsuba algorithm [84]). Concerning the modular reduction module, there exist commonly used Montgomery and Barrett reduction methods [85], [86]. Montgomery reduction and Barrett reduction represent of two classes of modular reduction algorithms, the left-to-right and right-to-left algorithms. In their most naive version, both algorithms have a runtime in $O(n^2)$ [87] where n is the bit-length of coefficients. Many recent works have been done to optimize the execution efficiency of both algorithms, including [88], [89], [90]. As Montgomery and Barrett reductions both require multiplication operations,

TABLE II
DESIGN FEATURE SUMMARY OF NTT ARCHITECTURES

Design	Scheme	In-place	Pre-processing	Post-processing
[12]	Kyber	Yes	No	Yes
[36]	Kyber	Yes	No	Yes
[37]	Kyber	Yes	No	No
[38]	Kyber	Yes	Yes	No
[39]	Kyber	Yes	No	Yes
[40]	Kyber/Dili.	Yes	No	Yes
[41]	Kyber	Yes	Yes	No
[92]	RLWE/SHE	No	Yes	No

Dili.: Dilithium. RLWE: Ring-LWE. SHE: Somewhat homomorphic encryption.

some other strategies were proposed to involve only additions and subtractions [78], [91].

Polynomial multiplication in KYBER and Dilithium are defined over the ring $\mathbb{Z}_q[x]/(x^n + 1)$ with $n = 256$ and $q = 3,329$ and $8,380,417$, respectively [32], [33]. It can also be defined as negative wrapped convolution, i.e., $\overline{B_j} = \sum_{i=0}^{n-1} \gamma^i \cdot {\omega_n}^{ij} \cdot b_i$, where $\overline{B}$ is the NTT transformed output. γ and ω are the $2n$-th and n-th root of unity, respectively $(\gamma = \sqrt{\omega})$. Similar to the general INTT, we have $A_j = \frac{1}{n}\gamma^{-j} \cdot \sum_{i=0}^{n-1} {\omega_n}^{-ij} \cdot \overline{b_i}$, where $\frac{1}{n}$ is carried out by incorporating a division by two operations at each step of the butterfly unit.

Even though polynomial multiplication in KYBER is equivalent to negative wrapped convolution, the $2n$-th root of unity does not exist for it. Therefore, KYBER uses a NTT variant (known as incomplete NTT), which generates 128 degree-2 polynomials to replace the original 256 degree-1 polynomial. Basically, the original 256 degree polynomial is decomposed into two 128 polynomials for NTT operations. When it comes to the polynomial multiplication stage, five multiplications are needed instead of the original one (INTT follows a similar rule). However, using Karatsuba method, we can reduce the required multiplication number to 4 [12]. Thus, the structure in Fig. 1 still applies to KYBER except with some adjustments.

Dilithium has a MLWE setup as KYBER, its NTT module and design techniques share a similar structure. Dilithium's q is larger than KYBER, but similar strategies like [78], [91] have been used to obtain efficient reduction [46].

Polynomial multiplication in FALCON can also be implemented by NTT, but few works have been done in this area.

Sparse Polynomial Multiplier: Sparse polynomial multiplication over $\mathbb{F}_2$ is critical to BIKE [53] and HQC [55]. Due to its special setting, traditional fast algorithms such as Karatsuba [93] are not as efficient as expected [62]. In fact, strategies based on schoolbook method obtain the best performance, as shown in [66]. Define $D = AC \bmod (x^n - 1)$, where $D = \sum_{i=0}^{n-1} d_i x^i$, $A = \sum_{i=0}^{n-1} a_i x^i$, and $C = \sum_{i=0}^{n-1} c_i x^i$ (d_i, a_i, and $c_i \in \{0, 1\}$ (A has only ω nonzero coefficients).

Then, we can have $[D]_{n\times 1} = [A]_{n\times n}[C]_{n\times 1}$, where $[A]_{n\times n}$ is a circulant matrix (the first column from left is the coefficients of A). For efficient realization, [66] suggested

$$[D]_{n\times 1} = \left[\overline{C}\right]_{n\times n}\left[\overline{A}\right]_{n\times 1}, \tag{2}$$

where one only needs to accumulate those nonzero values (of A) matched columns within the circulant matrix $[\overline{C}]$ ($[\overline{C}]$ now is a circulant matrix where the first left column contains the coefficients of C and $[\overline{A}]$ is a vector with all A's coefficients).

As shown in Fig. 2, the hardware-implemented sparse polynomial multiplier contains four major components. The two data memory cells inside the memory unit are responsible for feeding the coefficients of respective polynomials to the

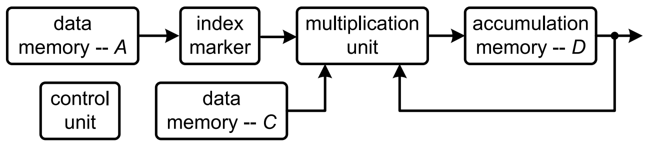

Fig. 2. Hardware architecture of sparse polynomial multiplier.

following components. Due to the sparsity that exists in polynomial A, an index marker (consisting of registers and a counter [66]) is needed to mark out the indices of those nonzero coefficients of A. The multiplication unit, which contains XOR gates, registers, and column ($[\overline{A}]$) segment calculator, executes the related column-based accumulation based on (2) along with the accumulation memory for D.

Sampler: Sampling (and hashing) is another computational-intensive operation within these PQC schemes. Its hardware implementation is relatively simple for lattice-based KYBER, which uses binomial sampling to generate required noises. For instance, to generate values that lie within $[-2, 2]$, we simply need to obtain random numbers in [3, 2] and [1, 0] and then do a subtraction [12]. While for Dilithium, rejection sampling-based method is needed for hardware implementation [44] (following the software code of [33]). The rejection unit contains a coefficient generator that generates a random coefficient lying in [0, 0X7FFFFF] by adding 1, 2^8, and 2^{16} multiples of three consecutive bytes coming out of the Keccack unit and then taking the last 31 bits. The multiplications are completed by executing logic shifts. Then, the generated coefficient is sent to a comparator. If it is smaller than Q, the coefficient will be accepted; otherwise not. Meanwhile, a temporary storage unit is also needed to produce a stable and constant-timing output. FALCON's Gaussian sampling is a rather complicated process, and since there is only one pre-printed hardware-software co-design [73], we do not explicitly analyze it here.

Note apart from the pure sampler, a hashing unit (a Keccak core and a related wrapper) is needed according to the specific algorithmic operation of the PQC scheme (random number generator is built by the hash module with a given seed and Keccak, following the suggestion of [94]) [12], [44]. Mostly, the Keccak core is obtained from the open-accessed code developed by the Keccak team [95], while the wrapper design is based on the scheme's specific requirement and system-level setup (e.g., low-speed/high-speed).

Sampling also plays a critical role in code-based PQC like BIKE [53] and HQC [55], as they require fixed-weight sparse vectors used in algorithmic operations such as sparse polynomial multiplication. The major challenges are: (i) there exist cases that the generated indices (nonzero values) are duplicated, and the process to remove the duplication will cause non-constant time for the sampling; (ii) the sampling process is relatively long. As a result, most available hardware samplers have failure probabilities, though very small [65].

Recent work has shown that the sampler for the mentioned code-based schemes can be implemented with Fisher-Yates shuffling [96] to obtain constant-time and fixed-weight operation [66]. This type of sampler can be designed with three major components, namely a bit swapper, an index marker, and a control unit. First of all, a vector with length n and the first ω bits being '1' are pre-stored in the memory. When the sampling begins, the random number generator (Keccack) generates ω numbers (indices) ranging from 0 to n uniformly and stores them in the memory. Then, the bit swapper will swap one of the first ω bits with the bit corresponding to one of the generated indices at a time. After all the ω bits are swapped, the non-zero bits are distributed in the vector uniformly. Meanwhile, because the indices are generated uniformly and the bits are swapped one by one, it avoids the case where the sampling fails (there are more than one generated indices sharing the same value). The after-swapped vector is stored back in the memory. Finally, the generated vector is fed to the index marker, where the after-swapped indices will be examined and stored in the memory for future use. This sampling hardware architecture applies to all code-based schemes, including Classic McEliece [54].

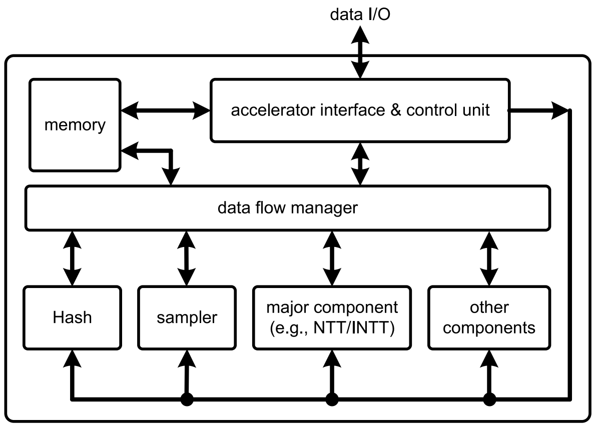
data I/O
memory
accelerator interface & control unit
data flow manager
Hash
sampler
major component (e.g., NTT/INTT)
other components

Fig. 3. The general architecture overview of a PQC accelerator.

Other Operations: Note that another expensive operation in Classic McEliece is the key generation [54], where a quasi-random binary matrix $\mathcal{H}$ needs to be reduced to its systematic form $(I_{n-k}|T)$ using Gaussian elimination, where I_{n-k} is a $(n-k) \times (n-k)$ identity matrix. Because $\mathcal{H}$ is not guaranteed to be systematicable, sometimes one needs to restart the reduction process and the key generation process. To address this problem, [57] proposed different versions of early abort systematizer to reduce computation time and resources, i.e., by detecting if the matrix is systematicable or not before actually executing the operation (see details in [57]).

For BIKE [53], polynomial inversion probably is another computational-intensive operation within its key generation. One recent work has proposed to use extended Euclidean algorithm to obtain efficient implementation [63]. The main idea behind this design was to initiate two polynomials where degree difference is 1 and then perform numbers of simple division steps with polynomials updated. Finally, one of the polynomials' coefficients are reversed to produce the inversion output. The detailed circuit design can be tracted at [63].

Encoder & decoder components play an important role in HQC, i.e., concatenated Reed-Muller and Reed-Solomon codes [55]. For HQC encoder, the message first goes into the Reed-Solomon encoder (consisting of a linear feedback shift register and multiple $\mathbb{F}_2$ multipliers). Then the encoded message is fed into the Reed-Muller encoder by performing matrix multiplication with the generator matrix G [55]. For the HQC decoder, the encoded message first goes to the Reed-Muller decoder (a modified version of Hadamard transform). Then the Reed-Solomon decoder uses syndrome decoding method to transfer the output of the Reed-Muller decoder into final output. Note the calculation of the roots of the syndrome equations is done by Additive FFT [55], [65].

B. System-Level Design Techniques

PQC hardware accelerator's system-level design efforts are not trivial even all the required components are well-designed [12], [44], [63]. In general, hardware PQC's system-level design (see Fig. 3) needs to take care of multiple aspects including memory setup, data flow management, performance, resource-oriented specific design, etc.

Memory: Memory setup is one of the most important efforts involved in system-level design [97]. Since memory has a pretty fixed read/write operation and limited data access width per cycle, the other hardware units need to handle the communication stream carefully to achieve high performance in a given context. Memory data flow usually includes proper connection to the accelerator interface & control unit, the data flow manager, and other major components. Architectures involving on-chip and off-chip communication also need to take care of extra design workloads, such as working with specific communication protocol provided by manufacturers. In addition, memory-based in-place operation architectures may result in high occupation of memory ports. Designing such architecture, e.g., in-place NTT architecture [12], requires extra design efforts to share memory with other components.

Data Flow Manager: The data flow manager is responsible for coordinating different components' working sequences (the specific PQC algorithmic operation determines the optimized processing order). Meanwhile, this data flow manager also moves data between different components according to the algorithm. The finite state machines (FSMs)' working statuses inside the data flow manager are determined through communicating with the control unit in the accelerator interface.

Performance: In most cases, performance directly determines the final success (or not) of a PQC accelerator, including resource usage, clock frequency, power consumption, etc. Strategies to improve performance include but are not limited to: (i) reducing major components' complexity by designing fine-tuned micro-architectures, e.g., NTT/INTT core can be further optimized by applying new memory access patterns [92]; (ii) careful arrangement of data flow within the accelerator to eliminate unnecessary latency cycles; (iii) minimizing the critical-path through optimizing the largest delay path within the accelerator (e.g., insert registers); (iv) share hardware resource among different components.

Application-Specific Design: For resource-abundant applications, a high-performance PQC accelerator can be designed to handle significant resource utilization effectively. Major components like NTT/INTT core can also contain highly parallel structures. For resource-constrained applications, a different design principle can be applied to fit limited resources, e.g., one can substitute the parallel structure with a serialized architecture. In both cases, the data flow manager and control unit need modifications accordingly.

C. Final Step: Testing and Validation

Hardware circuits and systems design also involves a final step: testing and validation. Software testbench is needed to verify the hardware-coded design. The actual efforts will be higher if on-board testing is involved. This final step, in general, requires tedious efforts (apart from technical elements).

IV. Needed Research and Future Directions

A. Needed Research

Hardware design for FALCON and SPHINCS$^+$ is desperately needed. It is expected that hardware design for FALCON will incur significant efforts, e.g., fast Fourier sampler, FALCON tree, etc. While one important component for SPHINCS$^+$ is the SPHINCS$^+$-Harak, which is an AES-based function [35]. As such AES has been widely studied [98], we hope a complete SPHINCS$^+$ accelerator can be released soon.

Another needed research is the secure implementation of PQC accelerators, i.e., side-channel attack resistant design. A recent report has revealed that existing hardware implemented PQC schemes are vulnerable to side-channel attacks [99]. Apart from power analysis, timing-based attacks, and fault attacks, electromagnetic-based and neural network assisted attacks nowadays are also widely investigated [100], [101]. We expect more efforts to be made for PQC.

NIST is also carrying out a new round of additional signature scheme standardization [7], where most submissions are based on problems other than the traditional lattice-based and/or code-based problems. The hardware designs for these schemes are valuable and may impact the future implementation standard after the NIST selection.

B. Applications & Future Directions

Real world PQC applications includes but are not limited to IoT security, cloud computing, blockchain security, and legal and notary services. Currently, Kyber has been standardized by NIST as PQC public-key encryption protocol. Dilithium and Falcon are standardized as PQC signature protocols. The next steps to integrate PQC protocols into specific applications include PQC hardware development [102], including hardware acceleration [8], integration with existing system [103], and cost-benefit analysis and usability enhancements [103], [104].

While current trend unveils novel progress on reconfigurable PQC circuits [105], [106], it is also expected that emerging hardware design techniques like in-memory computing [107], [108] can be explored to facilitate PQC hardware circuits and systems design. Meanwhile, we hope the mentioned hardware design techniques can also be extended to cryptosystems like fully homomorphic encryption [109].

V. Conclusion

This brief gives a brief yet comprehensive tutorial on different aspects related to hardware circuits and systems design for PQC schemes, standing on the recent advancements. We followed the NIST PQC standardization process to introduce algorithmic features, major arithmetic operations, and related component-level and system-level hardware design techniques. Further research directions and applications are also covered.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.
[2] "NIST post-quantum cryptography." nist.gov. 2024. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography
[3] J. Xie, K. Basu, K. Gaj, and U. Guin, "Special session: The recent advance in hardware implementation of post-quantum cryptography," in *Proc. 38th VTS*, 2020, pp. 1–10.
[4] "NIST post-quantum cryptography round 4 submissions." nist.gov. 2024. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions
[5] "Comments requested on three draft FIPS for post-quantum cryptography." nist.gov. Accessed: Aug. 24, 2023. [Online]. Available: https://csrc.nist.gov/news/2023/three-draft-fips-for-post-quantum-cryptography
[6] "NIST post-quantum cryptography selected algorithms 2022." nist.gov. 2024. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms-2022

[7] "Post-quantum cryptography: Digital signature schemes round 1 additional signatures." nist.gov. 2024. [Online]. Available: https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

[8] H. Li, Y. Tang, Z. Que, and J. Zhang, "FPGA accelerated post-quantum cryptography," *IEEE Trans. Nanotechnol.*, vol. 21, pp. 685–691, Oct. 2022.

[9] Z. Zhou, D. He, Z. Liu, M. Luo, and K.-K. R. Choo, "A software/hardware co-design of crystals-Dilithium signature scheme," *ACM Trans. Reconfig. Technol. Syst. (TRETS)*, vol. 14, no. 2, pp. 1–21, 2021.

[10] D. T. Nguyen, V. B. Dang, and K. Gaj, "High-level synthesis in implementing and benchmarking number theoretic transform in lattice-based post-quantum cryptography using software/hardware codesign," in *Proc. 16th Int. Symp. Appl. Reconfig. Comput. (ARC)*, Toledo, Spain, 2020, pp. 247–257.

[11] "Society overview, the IEEE circuits and systems society." IEEE-CAS. 2023. [Online]. Available: https://ieee-cas.org/about

[12] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," in *Proc. IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021, pp. 328–356.

[13] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, 2009.

[14] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. 29th Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2010, pp. 1–23.

[15] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Proc. 3rd Int. Algorithm. Num. Theory Symp.*, 1998, pp. 267–288.

[16] R. Overbeck and N. Sendrier, "Code-based cryptography," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 95–145.

[17] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-LWE cryptoprocessor," in *Proc. 16th Int. Workshop Cryptogr. Hardw. Embed. Syst.*, 2014, pp. 371–391.

[18] J. Howe, C. Moore, M. ÓNeill, F. Regazzoni, T. Güneysu, and K. Beeden, "Lattice-based encryption over standard lattices in hardware," in *Proc. DAC*, 2016, pp. 1–6.

[19] A. Zalekian, M. Esmaeildoust, and A. Kaabi, "Efficient implementation of NTRU cryptography using residue number system," *Int. J. Comput. Appl.*, vol. 124, no. 7, pp. 33–37, 2015.

[20] C. P. Renteria-Mejia and J. Velasco-Medina, "High-throughput ring-LWE cryptoprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 2332–2345, Aug. 2017.

[21] Y. Zhang, C. Wang, D. E. S. Kundi, A. Khalid, M. ÓNeill, and W. Liu, "An efficient and parallel R-LWE cryptoprocessor," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 5, pp. 886–890, May 2020.

[22] H. Nejatollahi, F. Valencia, S. Banik, F. Regazzoni, R. Cammarota, and N. Dutt, "Synthesis of flexible accelerators for early adoption of ring-LWE post-quantum cryptography," *ACM Trans. Embed. Comput. Syst. (TECS)*, vol. 19, no. 2, pp. 1–17, 2020.

[23] B. J. Lucas et al., "Lightweight hardware implementation of binary ring-LWE PQC accelerator," *IEEE Comput. Archit. Lett.*, vol. 21, no. 1, pp. 17–20, Jan.–Jun. 2022.

[24] J. Xie, P. He, X. Wang, and J. L. Imana, "Efficient hardware implementation of finite field arithmetic $AB+C$ for binary ring-LWE based post-quantum cryptography," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1222–1228, Apr.–Jun. 2022.

[25] P. He, U. Guin, and J. Xie, "Novel low-complexity polynomial multiplication over hybrid fields for efficient implementation of binary ring-LWE post-quantum cryptography," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 2, pp. 383–394, Jun. 2021.

[26] P. Duong-Ngoc and H. Lee, "Configurable mixed-radix number theoretic transform architecture for lattice-based cryptography," *IEEE Access*, vol. 10, pp. 12732–12741, 2022.

[27] W. Tan, A. Wang, X. Zhang, Y. Lao, and K. K. Parhi, "High-speed VLSI architectures for modular polynomial multiplication via fast filtering and applications to lattice-based cryptography," *IEEE Trans. Comput.*, vol. 72, no. 9, pp. 2454–2466, Sep. 2023.

[28] M. Kratochvíl, "Implementation of cryptosystem based on error-correcting codes," B.S. thesis, Matematicko-fyzikální fakulta, Univerzita Karlova, Staré Mesto, Czechia, 2013.

[29] S. Heyse and T. Güneysu, "Towards one cycle per bit asymmetric encryption: Code-based cryptography on reconfigurable hardware," in *Proc. 14th Int. Workshop Cryptogr. Hardw. Embed. Syst.*, 2012, pp. 340–355.

[30] A. Galimberti, D. Galli, G. Montanaro, W. Fornaciari, and D. Zoni, "On the use of hardware accelerators in QC-MDPC code-based cryptography," in *Proc. 19th ACM Int. Conf. Comput. Front.*, 2022, pp. 193–194.

[31] C. M. Stuart and P. Deepthi, "FPGA implementation of highly secure, hardware-efficient QC-LDPC code–based nonlinear cryptosystem for wireless sensor networks," *Int. J. Commun. Syst.*, vol. 30, no. 10, 2017, Art. no. e3233.

[32] "CRYSTALS-Kyber." 2020. [Online]. Available: https://pq-crystals.org/kyber/index.shtml

[33] "CRYSTALS-Dilithium." 2021. [Online]. Available: https://pq-crystals.org/dilithium/index.shtml

[34] "FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU." falcon-sign. 2021. [Online]. Available: https://falcon-sign.info/

[35] "SPHINCS$^+$." sphincs. 2023. [Online]. Available: https://sphincs.org/

[36] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of CRYSTALS-KYBER PQC algorithm through resource reuse," *IEICE Electron. Exp.*, vol. 17, no. 17, pp. 20200234–20200234, 2020.

[37] V. B. Dang, K. Mohajerani, and K. Gaj, "High-speed hardware architectures and FPGA benchmarking of CRYSTALS-Kyber, NTRU, and saber," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 306–320, Feb. 2023.

[38] D. Soni and R. Karri, "Efficient hardware implementation of PQC primitives and PQC algorithms using high-level synthesis," in *Proc. ISVLSI*, 2021, pp. 296–301.

[39] Q. Zeng, Q. Li, B. Zhao, H. Jiao, and Y. Huang, "Hardware design and implementation of post-quantum cryptography Kyber," in *Proc. HPEC*, 2022, pp. 1–6.

[40] A. Aikata, A. C. Mert, M. Imran, S. Pagliarini, and S. S. Roy, "KaLi: A crystal for post-quantum security using Kyber and Dilithium," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 2, pp. 747–758, Feb. 2023.

[41] T. T. Nguyen, S. Kim, Y. Eom, and H. Lee, "Area-time efficient hardware architecture for CRYSTALS-Kyber," *Appl. Sci.*, vol. 12, no. 11, p. 5305, 2022.

[42] G. Alagic et al., "Status report on the third round of the NIST post-quantum cryptography standardization process," US Dept. Commerce, Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Rep. NIST IR 8413, 2022.

[43] X. Li, J. Lu, D. Liu, A. Li, S. Yang, and T. Huang, "A high speed post-quantum Crypto-processor for crystals-Dilithium," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 71, no. 1, pp. 435–439, Jan. 2024.

[44] C. Zhao et al., "A compact and high-performance hardware architecture for crystals-Dilithium," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2022, no. 1, pp. 270–295, 2021.

[45] L. Beckwith, D. T. Nguyen, and K. Gaj, "High-performance hardware implementation of crystals-Dilithium," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, 2021, pp. 1–10.

[46] N. Gupta, A. Jati, A. Chattopadhyay, and G. Jha, "Lightweight hardware accelerator for post-quantum digital signature CRYSTALS-Dilithium," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 8, pp. 3234–3243, Aug. 2023.

[47] S. Ricci et al., "Implementing crystals-Dilithium signature scheme on FPGAs," in *Proc. 16th Int. Conf. Availab., Rel. Secur.*, 2021, pp. 1–11.

[48] G. Mao et al., "High-performance and configurable SW/HW co-design of post-quantum signature CRYSTALS-Dilithium," *ACM Trans. Reconfig. Technol. Syst.*, vol. 16, no. 3, pp. 1–28, 2023.

[49] L. Beckwith, D. T. Nguyen, and K. Gaj, "Hardware accelerators for digital signature algorithms Dilithium and FALCON," *IEEE Design Test*, early access, Aug. 14, 2023, doi: 10.1109/MDAT.2023.3305156.

[50] S. Coulon, P. He, T. Bao, and J. Xie, "Efficient hardware RNS decomposition for post-quantum signature scheme FALCON," in *Proc. 57th Asilomar Conf. Signals, Syst., Comput.*, 2023, pp. 1–8.

[51] D. Amiet, L. Leuenberger, A. Curiger, and P. Zbinden, "FPGA-based SPHINCS+ implementations: Mind the glitch," in *Proc. 23rd Euromicro Conf. Digit. Syst. Des. (DSD)*, 2020, pp. 229–237.

[52] Y. Dai, Y. Song, J. Tian, and Z. Wang, "High-throughput hardware implementation for Haraka in SPHINCS+," in *Proc. 24th ISQED*, 2023, pp. 1–6.

[53] (Bike Suite, Singapore). *BIKE—Bit Flipping Key Encapsulation*. 2021. [Online]. Available: https://bikesuite.org/

[54] "Classic McEliece." 2019. [Online]. Available: https://classic.mceliece.org/

[55] "HQC (Hamming Quasi-Cyclic)." 2020. [Online]. Available: https://pqc-hqc.org/

[56] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. Barreto, "MDPC-McEliece: New McEliece variants from moderate density parity-check codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pp. 2069–2073.

[57] P.-J. Chen et al., "Complete and improved FPGA implementation of classic McEliece," Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2022/412, 2022.

[58] S. Chen, H. Lin, W. Huang, and Y. Huang, "Hardware design and implementation of classic McEliece post-quantum cryptosystem based on FPGA," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, 2022, pp. 1–6.

[59] Y. Zhu et al., "Mckeycutter: A high-throughput key generator of classic McEliece on hardware," in *Proc. NIST 4th Standard. Conf.*, 2022, pp. 1–22.

[60] V. Kostalabros, J. Ribes-González, O. Farrás, M. Moretó, and C. Hernandez, "HLS-based hw/sw co-design of the post-quantum classic McEliece cryptosystem," in *Proc. FPL*, 2021, pp. 52–59.

[61] K. Basu et al., "NIST post-quantum cryptography-a hardware evaluation study," Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2002/122, 2002.

[62] J. Richter-Brockmann, J. Monom, and T. Güneysu, "Folding BIKE: Scalable hardware implementation for reconfigurable devices," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1204–1215, May 2021.

[63] J. Richter-Brockmann, M.-S. Chen, S. Ghosh, and T. Güneysu, "Racing BIKE: Improved polynomial multiplication and inversion in hardware," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2022, no. 1, pp. 557–588, 2021.

[64] J. Hu, W. Wang, R. C. C. Cheung, and H. Wang, "Optimized polynomial multiplier over commutative rings on FPGAs: A case study on BIKE," in *Proc. ICFPT*, 2019, pp. 231–234.

[65] S. Deshpande et al., "Fast and efficient hardware implementation of HQC," Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2022/1183, 2022.

[66] Y. Tu, P. He, Ç. K. Koç, and J. Xie, "LEAP: Lightweight and efficient accelerator for sparse polynomial multiplication of HQC," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 31, no. 6, pp. 892–896, Jun. 2023.

[67] J. Xie, P. He, and W. Wen, "Efficient implementation of finite field arithmetic for binary ring-LWE post-quantum cryptography through a novel lookup-table-like method," in *Proc. 58th DAC*, 2021, pp. 1279–1284.

[68] S. Khan, W.-K. Lee, A. Khalid, A. Majeed, and S. O. Hwang, "Area-Optimized constant-time hardware implementation for polynomial multiplication," *IEEE Embed. Syst. Lett.*, vol. 15, no. 1, pp. 5–8, Mar. 2023.

[69] X. Zhang and K. K. Parhi, "Reduced-complexity modular polynomial multiplication for R-LWE cryptosystems," in *Proc. ICASSP*, 2021, pp. 7853–7857.

[70] E. Carter, P. He, and J. Xie, "High-performance polynomial multiplication hardware accelerators for KEM saber and NTRU," Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2022/628, 2022.

[71] P. He et al., "HPMA-NTRU: High-performance polynomial multiplication accelerator for NTRU," in *Proc. IEEE DFT*, 2022, pp. 1–6.

[72] D. D. Chen et al., "High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems," *IEEE Trans. Circuits Syst. I*, vol. 62, no. 1, pp. 157–166, Jan. 2015.

[73] E. Karabulut and A. Aysu, "A hardware-software co-design for the discrete gaussian sampling of FALCON digital signature," Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2023/908, 2023.

[74] P. He, Y. Tu, and J. Xie, "LOCS: LOw-latency and ConStant-timing implementation of fixed-weight sampler for HQC," in *Proc. ISCAS*, 2023, pp. 1–5.

[75] X. Zhang and Z. Xie, "Sparsity-aware medium-density parity-check decoder for McEliece Cryptosystems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 9, pp. 3343–3347, Sep. 2023.

[76] D. Amiet, A. Curiger, and P. Zbinden, "FPGA-based accelerator for post-quantum signature scheme SPHINCS-256," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2018, no. 1, pp. 18–39, 2018.

[77] J. M. Pollard, "The fast fourier transform in a finite field," *Math. Comput.*, vol. 25, no. 114, pp. 365–374, 1971.

[78] F. Yaman, A. C. Mert, E. Öztürk, and E. Savas, "A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme," in *Proc. DATE*, 2021, pp. 1020–1025.

[79] N. Zhang et al., "NTTU: An area-efficient low-power NTT-uncoupled architecture for NTT-based multiplication," *IEEE Trans. Comput.*, vol. 69, no. 4, pp. 520–533, Apr. 2020.

[80] C. Zhang et al., "Towards efficient hardware implementation of NTT for kyber on FPGAs," in *Proc. ISCAS*, 2021, pp. 1–5.

[81] Y. Su, B.-L. Yang, C. Yang, Z.-P. Yang, and Y.-W. Liu, "A highly unified reconfigurable multicore architecture to speed up NTT/INTT for homomorphic polynomial multiplication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 8, pp. 993–1006, Aug. 2022.

[82] E. Chu and A. George, *Inside The FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*. Boca Raton, FL, USA: CRC Press, 1999.

[83] W. M. Gentleman and G. Sande, "Fast fourier transforms: For fun and profit," in *Proc. Fall Joint Comput. Conf.*, 1966, pp. 563–578.

[84] Z. Huai, J. Zhou, and X. Zhang, "Efficient hardware implementation architectures for long integer modular multiplication over general Solinas prime," *J. Sig. Process. Syst.*, vol. 94, no. 10, pp. 1067–1082, Aug. 2022.

[85] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, no. 170, pp. 519–521, 1985.

[86] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in *Proc. Conf. Theory Appl. Cryptogr. Techn.*, 1986, pp. 311–323.

[87] W. Hasenplaugh, G. Gaubatz, and V. Gopal, "Fast modular reduction," in *Proc. 18th ARITH*, 2007, pp. 225–229.

[88] M. Knezevic, L. Batina, and I. Verbauwhede, "Modular reduction without precomputational phase," in *Proc. ISCAS*, 2009, pp. 1389–1392.

[89] M. Knezevic, F. Vercauteren, and I. Verbauwhede, "Faster interleaved modular multiplication based on Barrett and montgomery reduction methods," *IEEE Trans. Comput.*, vol. 59, no. 12, pp. 1715–1721, Dec. 2010.

[90] S. Li and Z. Gu, "Lazy reduction and multi-precision division based on modular reductions," in *Proc. APCCAS*, 2018, pp. 407–410.

[91] N. Zhang, B. Yang, C. Chen, S. Yin, S. Wei, and L. Liu, "Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 2, pp. 49–72, 2020.

[92] W. Guo and S. Li, "Highly-efficient hardware architecture for CRYSTALS-Kyber with a novel conflict-free memory access pattern," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 11, pp. 4505–4515, Nov. 2023.

[93] A. Karatsuba, "Multiplication of multidigit numbers on automata," in *Sov. Phys. Doklady*, vol. 7, pp. 595–596, 1963.

[94] "SHA-3 standard: Permutation-based hash and extendable-output functions," Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Rep. FIPS 202. [Online]. Available: https://csrc.nist.gov/pubs/fips/202/final

[95] Keccak Team. "Keccak in VHDL: High-speed core." [Online]. Available: https://keccak.team/hardware.html

[96] R. A. Fisher and F. Yates, *Statistical Tables for Biological, Agricultural and Medical Research*. Royal Oak, MI, USA: Hafner Publ. Co., 1953.

[97] P. He, C.-Y. Lee, and J. Xie, "Compact coprocessor for KEM saber: Novel scalable matrix originated processing," in *Proc. NIST 3rd Stand. Conf.*, 2021, pp. 1–16.

[98] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 9, pp. 957–967, Sep. 2004.

[99] Y. Ji, R. Wang, K. Ngo, E. Dubrova, and L. Backlund, "A side-channel attack on a hardware implementation of CRYSTALS-Kyber," in *Proc. IEEE ETS*, 2023, pp. 1–5.

[100] M. Alam et al., "Neural network-based inherently fault-tolerant hardware cryptographic primitives without explicit redundancy checks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 1, pp. 1–30, 2020.

[101] S. Saha, D. Jap, D. Basu Roy, A. Chakraborty, S. Bhasin, and D. Mukhopadhyay, "A framework to counter statistical ineffective fault analysis of block ciphers using domain transformation and error correction," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1905–1919, 2019.

[102] D. Dione et al., "Hardware security for IoT in the quantum era: Survey and challenges," *J. Inf. Sec.*, vol. 14, no. 4, pp. 227–249, 2023.

[103] D. Joseph et al., "Transitioning organizations to post-quantum cryptography," *Nature*, vol. 605, no. 7909, pp. 237–243, 2022.

[104] S. Paul, "On the transition to post-quantum cryptography in the Industrial Internet of Things," Ph.D. dissertation, FachbereichInformatik Sicherheitinder Informationstechnik(SIT), Technische Universität Darmstadt, Darmstad, Germany, 2022.

[105] Y. Zhu et al., "A 28nm 48KOPS 3.4 μJ/op agile Crypto-processor for post-quantum cryptography on multi-mathematical problems," in *Proc. ISSCC*, vol. 65, 2022, pp. 514–516.

[106] U. Banerjee, A. Pathak, and A. P. Chandrakasan, "2.3 an energy-efficient configurable lattice cryptography processor for the quantum-secure Internet of Things," in *Proc. ISSCC*, 2019, pp. 46–48.

[107] A. Dervay and W. Zhao, "CIMulator: A computing in memory emulator framework," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 10, pp. 4183–4187, Oct. 2022.

[108] J. Chen, W. Zhao, Y. Wang, Y. Shu, W. Jiang, and Y. Ha, "A reliable 8T SRAM for high-speed searching and logic-in-memory operations," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 30, no. 6, pp. 769–780, Jun. 2022.

[109] S. Kurniawan, P. Duong-Ngoc, and H. Lee, "Configurable memory-based NTT architecture for homomorphic encryption," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 10, pp. 3942–3946, Oct. 2023.