International Journal of Innovative Solutions in Engineering
2026, Special Issue No. 1 (pages 98–109)
DOI: 10.47960/3029-3200.2026.S1.98

Review Article

OPEN ACCESS

Domain-Specific Accelerators for Post-Quantum Cryptography

Ratko Pilipović[1]

[1]Faculty of Computer and Information Science University of Ljubljana, Slovenia

Abstract
The advent of large-scale quantum computing threatens the public-key infrastructure that secures modern digital communication, as Shor's algorithm renders the integer-factorization and discrete-logarithm problems underlying RSA and elliptic-curve cryptography tractable. In response, the National Institute of Standards and Technology (NIST) has standardized a first generation of quantum-resistant digital signature algorithms, namely the lattice-based CRYSTALS-Dilithium (ML-DSA, FIPS 204) and Falcon (FN-DSA) and the hash-based SPHINCS+ (SLH-DSA, FIPS 205). These schemes impose far higher computational and memory costs than their classical counterparts, which motivates domain-specific hardware accelerators able to meet the performance and energy budgets of practical systems. This survey reviews domain-specific accelerators for post-quantum signatures, giving equal attention to the three standardized schemes. For each, we examine the underlying construction, its parameter sets, and, above all, the implementation challenges that shape its hardware, from the rejection sampling of Dilithium to the floating-point Gaussian sampling of Falcon and the hashing-bound datapath of SPHINCS+. We compare reported FPGA and ASIC accelerators for every scheme, present a general-purpose accelerator template, and outline the emerging schemes and open challenges that will guide future hardware design.

Article history
Received: 15.9.2026.
Revised: 5.10.2026.
Accepted: 7.10.2026.

Keywords
Post-Quantum Cryptography,
Digital Signatures,
Hardware Accelerators,
FPGA,
Falcon,
Dilithium.

1 Introduction

The security of contemporary digital communication rests on public-key cryptography, most notably the Rivest-Shamir-Adleman (RSA) cryptosystem and elliptic-curve cryptography (ECC), whose security derives from the presumed intractability of integer factorization and the discrete-logarithm problem. Shor's algorithm [1] dismantles both assumptions: a sufficiently large quantum computer would solve them in polynomial time, breaking the key-exchange and signature schemes that secure today's protocols. Even before such machines exist, the threat is immediate for long-lived data through 'harvest-now, decrypt-later' attacks.

Digital signatures, which provide authenticity, integrity, and non-repudiation, must therefore migrate to quantum-resistant foundations ahead of the arrival of cryptographically relevant quantum hardware.

Recognizing this risk, NIST initiated a post-quantum cryptography (PQC) standardization process in 2017 [2], announcing the first selected algorithms in 2022 and publishing the first standards in 2024. For digital signatures, three schemes were selected:

1. The lattice-based CRYSTALS-Dilithium, standardized as the Module-Lattice-Based Digital Signature Algorithm (ML-DSA) in FIPS 204 [3],

Contact Ratko Pilipović, ratko.pilipovic@fri.uni-lj.si, Faculty of Computer and Information Science University of Ljubljana
©2026 by the Author(s). Licensee IJISE by Faculty of Mechanical Engineering, Computing and Electrical Engineering, University of Mostar. This article is an open-access and distributed under the terms and conditions of the CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/)

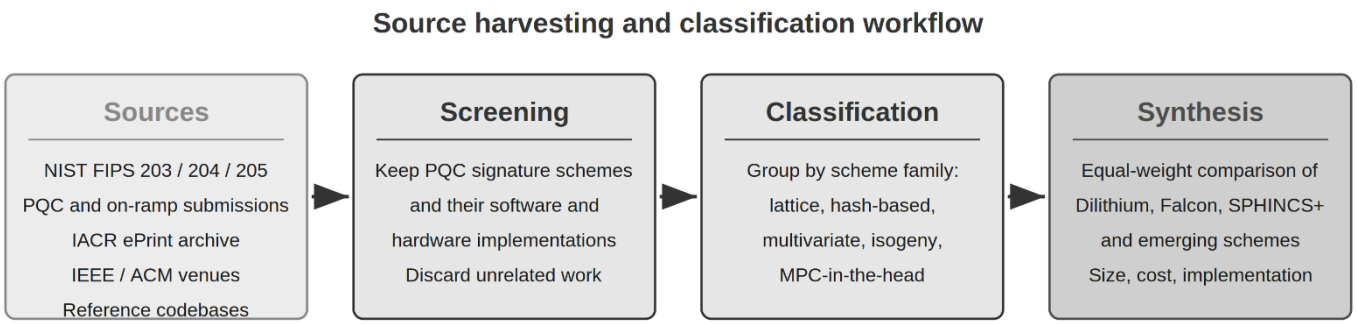


Figure 1 Source-harvesting and classification workflow used to collect and organize the literature surveyed in this work

2. The hash-based SPHINCS+, standardized as the Stateless Hash-Based Digital Signature Algorithm (SLH-DSA) in FIPS 205 [4], and
3. Falcon, to be standardized as the FFT-over-NTRU Digital Signature Algorithm (FN-DSA) in the forthcoming FIPS 206 [5].

Deliberately, these schemes rest on two distinct hardness families, namely structured lattices and cryptographic hash functions, so that a cryptanalytic advance against one does not compromise the whole portfolio.

The schemes embody different design philosophies. Dilithium favors implementation simplicity at the cost of larger keys and signatures. Falcon minimizes the combined public-key and signature size through floating-point arithmetic and Gaussian sampling that are hard to implement securely. SPHINCS+ makes the most conservative assumption but produces the largest signatures. This survey presents the three side by side with equal treatment. Sections 3 to 5 examine Dilithium, Falcon, and SPHINCS+ in turn; Section 6 compares them, Section 7 surveys the emerging on-ramp candidates, and Section 8 discusses open challenges before Section 9 concludes. Figure 1 outlines the source-harvesting and classification approach used to prepare this survey.

2 Background

Post-quantum signatures rest on problems believed hard for both classical and quantum adversaries. The two lattice-based schemes draw their security from the difficulty of finding short vectors in high-dimensional lattices. The Shortest Vector Problem and the Learning With Errors (LWE) problem [8], with its ring and module variants, give a worst-case-to-average-case reducible foundation. The hash-based scheme relies only on the preimage and collision resistance of a hash function, properties only mildly weakened by Grover's quantum search and hence regarded as highly conservative.

Two paradigms turn these problems into signatures. In the Fiat-Shamir-with-aborts paradigm, an identification protocol is made non-interactive through a hash-derived challenge, and rejection sampling ('aborts') makes the signature independent of the secret key. Dilithium follows this route. In the hash-and-sign paradigm formalized by the Gentry-Peikert-Vaikuntanathan (GPV) framework [9], a message is hashed to a point and a lattice trapdoor samples a nearby point from a discrete Gaussian, leaking no information about the trapdoor. Falcon instantiates this construction over NTRU lattices. Hash-based signatures form a separate lineage, composing one-time and few-time signatures into a tree, of which SPHINCS+ is the stateless culmination. Fast polynomial multiplication via the Number Theoretic Transform (NTT), together with the floating-point Fast Fourier Transform (FFT) in Falcon, and the SHA-3/SHAKE hash family are computational primitives shared across the schemes. Figure 2 places the schemes covered in this survey within a broader taxonomy of post-quantum signature families.

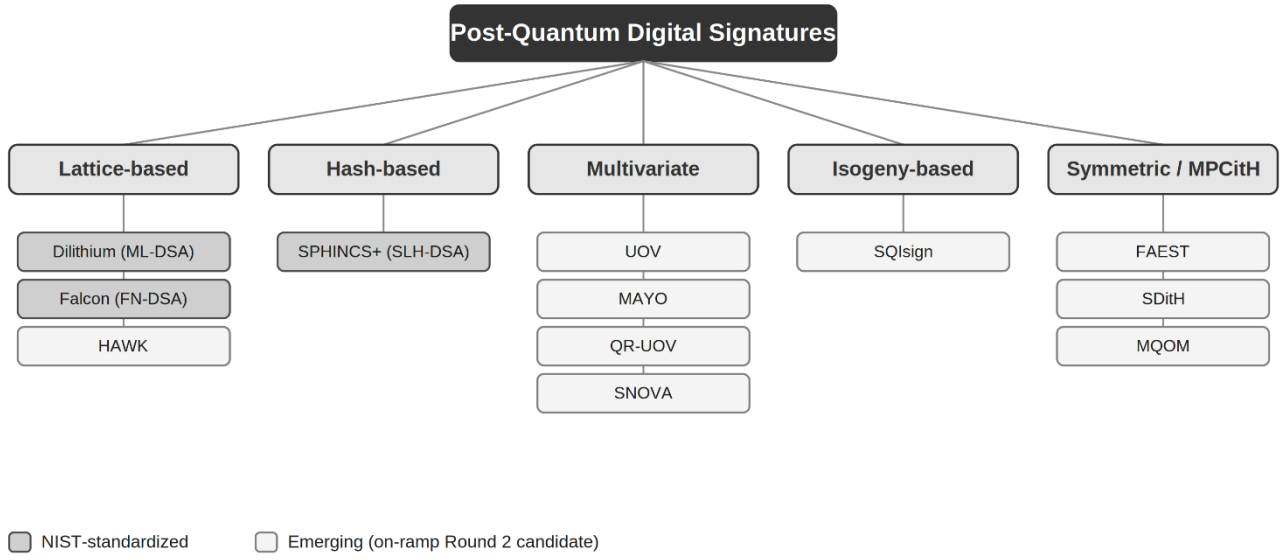


Figure 2 Taxonomy of post-quantum digital signature schemes by underlying hardness family, distinguishing the NIST-standardized schemes from the emerging on-ramp candidates

3 CRYSTALS-Dilithium (ML-DSA)

CRYSTALS-Dilithium [6], standardized as ML-DSA in FIPS 204 [3], is the primary NIST-recommended post-quantum signature scheme. Its security rests on the Module-LWE and Module-SIS problems over the ring $Z_q[x]/(x^256 + 1)$, with a 23-bit NTT-friendly prime modulus q = 8380417. Working over modules rather than plain rings lets a single implementation reach several security levels by changing only the module dimension, which simplifies both specification and hardware reuse.

Dilithium follows the Fiat-Shamir-with-aborts paradigm. Signing samples a masking vector y, computes a commitment $w = Ay$, derives a challenge polynomial c by hashing the message with a compressed form of w, and forms the response $z = y + c*s1$, where $s1$ is part of the secret key. A rejection-sampling step restarts whenever z falls outside a prescribed range, guaranteeing that the released signature is statistically independent of the secret. Crucially, every operation uses integer arithmetic and uniform sampling. By deliberately avoiding discrete Gaussian sampling, Dilithium admits a constant-time, side-channel-resistantimplementation
comparatively straightforwardly, which is the main reason it is the safe default.

FIPS 204 defines three parameter sets, ML-DSA-44, ML-DSA-65, and ML-DSA-87, with public keys of 1312, 1952, and 2592 bytes and signatures of 2420, 3309, and 4627 bytes. These are larger than Falcon at comparable security but come with fast, balanced signing and verification, which makes Dilithium the recommended general-purpose choice.

Although Kyber is a key-encapsulation mechanism rather than a signature scheme, it is worth noting alongside Dilithium, since the two form the CRYSTALS suite and share almost all of their arithmetic. CRYSTALS-Kyber, standardized as the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) in FIPS 203 [27], rests on the same Module-LWE foundation over the ring $Z_q[x]/(x^256 + 1)$, differing mainly in the modulus (q = 3329 rather than 8380417) and in the module dimensions used per security level. Because both schemes rely on the Number Theoretic Transform, uniform sampling, and the SHA-3/SHAKE family, a single hardware datapath can serve both, and most Dilithium accelerators are in practice designed as joint Kyber-Dilithium engines. The unified KaLi architecture [10] and the industrial Adams Bridge design [11] both realize ML-KEM and ML-DSA on shared arithmetic and memory, amortizing area on a device that must perform key establishment as well as signing.

3.1 Implementation challenges

Although Dilithium is the most implementation-friendly of the three schemes, several practical obstacles remain. The most prominent is the rejection-sampling loop in signing. Because the response is discarded and recomputed whenever it leaves the prescribed range, the number of iterations is data-dependent and varies from signature to signature, with an average of several attempts. A hardware or embedded implementation must therefore tolerate a variable, unbounded latency, provide buffering for retries, and avoid stalls that would otherwise serialize the datapath. A second cost is hashing. Dilithium relies on the SHAKE extendable-output function to expand the public matrix A from a seed, to sample the masking vector, and to derive the challenge, so the Keccak permutation often dominates the overall cycle count and becomes the first target for acceleration. Expanding A on the fly rather than storing it saves considerable memory but shifts the burden further onto the hash unit, a classic area-versus-throughput trade-off.

On the arithmetic side, efficient designs need fast modular multipliers built around the NTT-friendly prime, using Montgomery or Barrett reduction, together with logic for bit-packing, coefficient decomposition, and the hint mechanism that compresses the signature.

From a security standpoint, Dilithium was deliberately designed so that the values exposed by rejection depend only on information that may be public, which keeps constant-time software comparatively easy to write. Protection against power and electromagnetic analysis is harder because masking the secret-dependent NTT and sampling increases area and latency and remains an active research topic. These considerations have nonetheless been met by mature accelerators, including the unified KaLi datapath shared with Kyber [10] and the industrial open-source Adams Bridge design [11], both of which exploit the scheme's NTT-centric, floating-point-free structure.

Table 1 compares representative FPGA implementations of Dilithium. The designs reveal a clear area-versus-latency trade-off. The high-throughput architecture of Ricci et al. [21] spends close to a thousand DSP blocks to drive signing below 100 microseconds, whereas the compact designs of Beckwith et al. [22] and Norga et al. [23] use only sixteen DSP blocks at the cost of longer signing times, and both support all parameter sets within a single datapath. The unified ASIC designs KaLi [10] and Adams Bridge [11] instead share their arithmetic with Kyber and ML-KEM, respectively, trading direct comparability for a smaller combined footprint when a device must run both a key-encapsulation and a signature scheme.

Table 1Representative CRYSTALS-Dilithium (ML-DSA) FPGA implementations at security level III (ML-DSA-65), each supporting key generation, signing, and verification. US+ denotes Xilinx UltraScale+, Freq is in MHz, and Sign and Verify latencies in us (average over rejection-sampling restarts). The unified ASIC designs KaLi [10] and Adams Bridge [11] are discussed in the text and omitted here as they are not directly comparable in FPGA metrics

Work	Platform	LUT	FF	DSP	BRAM	Freq	Sign	Verify
Ricci et al. [21]	FPGA (US+)	68461	86295	965	145	350	63	95
Beckwith et al. [22]	FPGA (US+)	53907	28435	16	29	256	193	39
Norga et al. [23]	FPGA (US+)	54942	29309	16	29	230	143	42

4 Falcon (FN-DSA)

Falcon [5], to be standardized as FN-DSA, is a lattice-based hash-and-sign scheme over NTRU lattices. Its defining advantage is compactness: it is engineered to minimize the sum of the public-key and signature sizes, a metric that dominates bandwidth in many protocols. The public key is a polynomial h over $Z_q[x]/(x^n + 1)$ with $q = 12289$, and the private key is a set of short polynomials (f, g, F, G) that form a trapdoor basis of the same lattice while satisfying the NTRU equation. Two parameter sets are standardized: Falcon-512 (security category 1) and Falcon-1024 (category 5), where n is the lattice dimension.

Signing follows the GPV framework [9]. The salted hash of the message is mapped to a target point, and a trapdoor sampler produces a short lattice vector near that target from a discrete Gaussian. Falcon performs this with fast Fourier sampling (ffSampling), a recursive procedure that traverses a precomputed binary 'Falcon tree', splitting and merging polynomials in the FFT domain and drawing integers with the SamplerZ Gaussian sampler at the leaves. A random salt ensures repeated signatures of the same message use different targets, preserving the security proof. Verification is comparatively cheap: the compressed signature is decompressed, multiplied with the public key modulo q using the NTT, and accepted only if the resulting vector's squared norm falls below a fixed bound.

This design yields very small artifacts, with public keys of 897 and 1793 bytes and variable-length signatures averaging about 666 and 1280 bytes, together with fast verification. That compactness, however, comes at the price of by far the most difficult implementation profile of the three standardized signatures.

4.1 Implementation challenges

Falcon's central difficulty is its reliance on floating-point arithmetic. The ffSampling procedure and the underlying FFT operate on double-precision complex numbers, whereas the other lattice signatures use only integers.

A floating-point unit is large and power-hungry in hardware, and, more subtly, the signing result must be reproducible across platforms so that the emitted signatures remain valid, which forces strict adherence to IEEE-754 rounding and makes portable, deterministic behavior hard to guarantee. Verification, by contrast, uses only integer NTT arithmetic and is far simpler, which is why the earliest hardware supported verification alone.

A second obstacle is the recursive structure of ffSampling. The algorithm walks a binary Falcon tree, splitting and merging polynomials level by level, a control flow that maps poorly onto fixed hardware datapaths and must be flattened into an iterative form before it can be pipelined, as early high-level-synthesis work demonstrated. The tree itself is bulky. In its pre-expanded form, it consumes on the order of tens of kilobytes for Falcon-1024, which strains on-chip memory and complicates the scheduling of split, merge, and sampling operations across memory banks. Key generation is more demanding still, since solving the NTRU equation involves big-integer arithmetic over a tower of fields, and is therefore normally performed off the device.

The discrete Gaussian sampler, SamplerZ, is the most delicate component from a security standpoint. It draws each coefficient from a distribution whose parameters change per coefficient, using an internal rejection loop, and it must run in constant time so that neither the rejection pattern nor the floating-point operations leak information about the secret basis. Constant-time floating-point is notoriously platform-dependent, so timing side-channels are a recurring concern and a driver of specialized sampler designs. Compounding these issues, the signature is compressed to a variable length, which is awkward for fixed-size hardware buffers and for constant-time encoding. As a result, Falcon has remained comparatively underexplored in hardware. Early work was software-oriented, and subsequent hardware efforts addressed isolated parts, most notably the signature-generation accelerator FalconSign [12] and dedicated Gaussian-sampler datapaths, while a recent unified architecture has

shown that signing and verification can share one datapath, narrowing a long-standing gap [15].

Table 2 compares reported Falcon hardware implementations at the highest security level. Because no prior design covers both signing and verification, a fully uniform comparison is not possible, yet the figures convey the wide spread of resource usage and latency across sign-only, verify-only, and unified designs, as well as the substantial area required once floating-point sampling is mapped to hardware.

Table 2 Reported Falcon hardware implementations at security level V (Falcon-1024). US+ denotes Xilinx UltraScale+, S7 the Series-7 family, Freq is in MHz, and Sign and Verify latencies in ms, with n/a marking a function the design does not implement

Work	Platform	Ops	LUT	FF	DSP	BRAM	Freq	Sign	Verify
Schmid [16]	HLS, US+	Both	58525	49989	197	51	187.5	8.736	1.255
FalconSign [12]	HW, US+	Sign	80496	46495	220	58	185	1.730	n/a
Pendyala [18]	HW, US+	Sign	88773	46945	686	58	185	0.863	n/a
RVCE-FAL [19]	HW/SW, US+	Both	18565	3210	17	0	83	37.740	n/a
Beckwith [17]	HW, US+	Verify	13729	6771	4	2	314	n/a	0.065
Nguyen [20]	HW/SW, S7	Verify	3690	2984	15	1	100	n/a	5.000
Smole [15]	HW, US+	Both	73934	38390	175	84	187.5	2.043	0.089

5 SPHINCS+ (SLH-DSA)

SPHINCS+ [7], standardized as SLH-DSA in FIPS 205 [4], departs entirely from lattices. It is a stateless hash-based scheme whose security reduces to standard properties of its underlying hash function, namely second-preimage and collision resistance, instantiated with SHAKE256 or SHA-2. Because these properties are only modestly affected by known quantum algorithms, SPHINCS+ offers the most conservative security guarantee among the standardized signatures, independent of any structured algebraic assumption, making it an attractive hedge against future lattice cryptanalysis.

SPHINCS+ composes several hash-based primitives. A Winternitz one-time signature (WOTS+) signs a single digest, a Forest of Random Subsets (FORS) provides a few-time signature for the randomized message hash, and a hypertree of Merkle trees in the XMSS style chains many one-time keys under one long-term public key. Statelessness, meaning that no signing state needs to be kept between signatures, which is essential for practical deployment, is achieved by pseudorandomly selecting the FORS index per message, so that accidental key reuse is overwhelmingly unlikely. The public key is remarkably small, only 32, 48, or 64 bytes at the three security levels.

The price of this conservatism is signature size and signing cost. FIPS 205 defines parameter sets at three security levels, each in a 'small' (s) variant that minimizes signature size and a 'fast' (f) variant that minimizes signing time. Even the compact variants produce signatures of roughly 7856, 16224, and 29792 bytes, an order of magnitude larger than Dilithium or Falcon, and the fast variants are larger still.

5.1 Implementation challenges

The implementation profile of SPHINCS+ is the mirror image of Falcon. There is no floating point, no large modular arithmetic, and no rejection sampling, so the code is compact, and the constant-time question is comparatively simple. Instead, the dominant cost is hashing. A single signing operation evaluates tens of thousands of hash calls to build the Winternitz chains, the FORS trees, and the hypertree authentication paths, so overall performance is bound almost entirely by the throughput of the Keccak or SHA core. Hardware acceleration therefore concentrates on instantiating many hash cores in parallel and on pipelining the tree computations, since the numerous WOTS+ chains and FORS subtrees are largely independent and expose abundant parallelism [13]. The main scheduling difficulty is feeding those cores efficiently while respecting the sequential dependencies of each Merkle authentication path.

The second challenge is the sheer size of the signature. At roughly 8 to 50 kilobytes, it dwarfs the lattice schemes and stresses on-chip memory, external bandwidth, and any protocol that must transport or store it, so implementations often stream the signature rather than buffer it whole. Statelessness is a benefit for robustness, because it removes the catastrophic failure mode of one-time-key reuse after a state rollback, but it does not remove the threat of fault attacks on the hypertree, against which redundant recomputation or caching is used. Overall, SPHINCS+ trades speed and bandwidth for conservative security and implementation simplicity, making it well suited to firmware signing, certificate roots, and other settings where signatures are infrequent, and longevity matters more than size.

Table 3 Representative SPHINCS+ (SLH-DSA) FPGA implementations for the fast parameter set at security level I (SLH-DSA-128f). US+ denotes Xilinx UltraScale+, Freq is in MHz, and Sign and Verify latencies in ms. Amiet et al. [24] and Deshpande et al. [26] use SHAKE-256, whereas Berthet et al. [25] use SHA-256

Work	Platform	LUT	FF	DSP	BRAM	Freq	Sign	Verify
Amiet et al. [24]	FPGA (Artix-7)	47991	72505	1	11.5	500	1.01	0.16
Berthet et al. [25]	FPGA (US+)	6127	4933	0	0.5	150	64.30	2.50
Deshpande et al. [26]	FPGA (Artix-7)	10197	7357	0	8	150	19.30	1.91

Table 3 compares representative FPGA implementations of SPHINCS+. Because performance is bound by the hash core, the designs trade area against speed over a wide range. The high-throughput accelerator of Amiet et al. [24] instantiates large parallel hash cores to reach millisecond signing, at roughly 48,000 LUTs, whereas the compact coprocessor of Berthet et al. [25] fits in about 6,000 LUTs but signs more than an order of magnitude slower. The recent SPHINCSLET design of Deshpande et al. [26] targets the full standard across all parameter sets with a balanced area and latency. Across every design, the signing time is measured in milliseconds and the resource cost is dominated by hashing rather than by arithmetic, in sharp contrast to the lattice schemes.

6 Comparative Analysis

Table 4 summarizes the public-key and signature sizes of the three standardized schemes. The contrast is stark. Falcon offers by far the smallest combined footprint and the fastest verification, appealing for bandwidth-constrained protocols and verification-heavy workloads such as certificate validation, provided its implementation complexity is tolerable.

Dilithium sits in the middle on size but leads on implementation simplicity and balanced performance, which is why it is the recommended default. SPHINCS+ occupies the extreme of conservative security and minimal key size but incurs the largest signatures and the slowest signing.

Table 4 Public-key and signature sizes of the NIST-standardized post-quantum signatures. Falcon sizes marked * are variable-length averages, and SLH-DSA figures are for the small (s) variants

Scheme	Family / hard problem	Level	Public key (B)	Signature (B)
ML-DSA-44	Module-LWE/SIS	1	1312	2420
ML-DSA-65	Module-LWE/SIS	3	1952	3309
ML-DSA-87	Module-LWE/SIS	5	2592	4627
Falcon-512	NTRU (GPV)	1	897	~666*
Falcon-1024	NTRU (GPV)	5	1793	~1280*
SLH-DSA-128s	Hash-based	1	32	7856
SLH-DSA-192s	Hash-based	3	48	16224
SLH-DSA-256s	Hash-based	5	64	29792

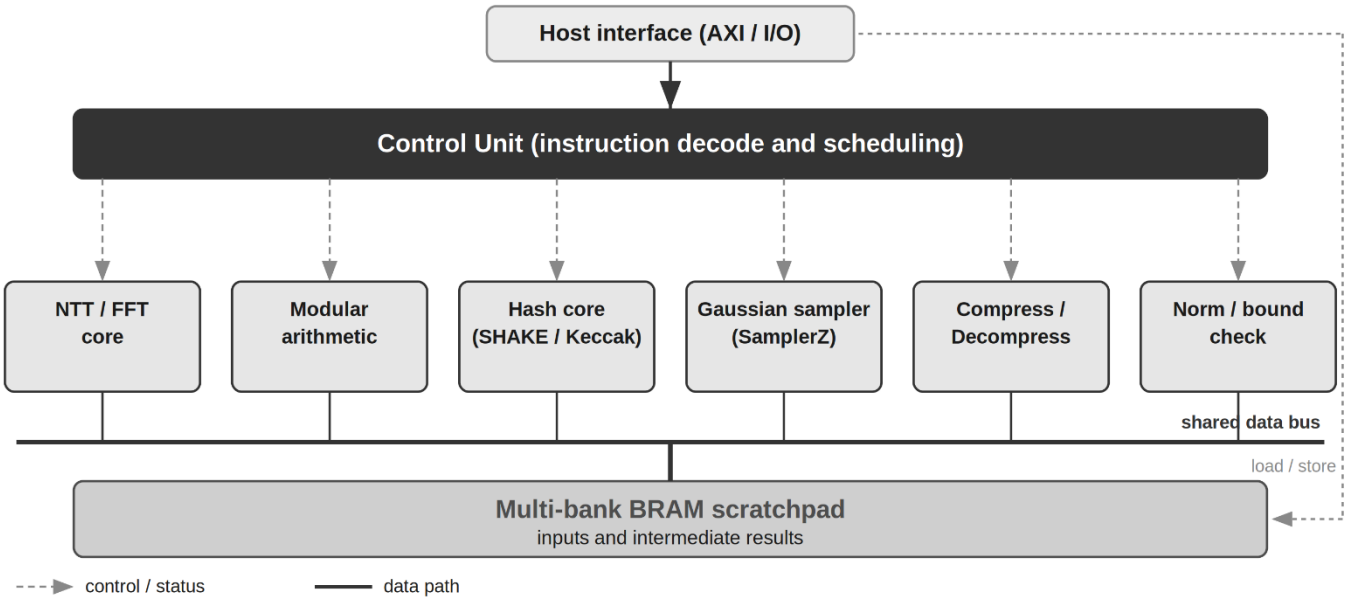


Figure 3 General-purpose post-quantum digital-signature accelerator. A control unit schedules dedicated modules that read and write a shared multi-bank BRAM scratchpad holding the inputs and intermediate results

Beyond raw sizes, the schemes differ in the arithmetic they demand. Dilithium and Falcon's verification path needs modular integer arithmetic accelerated by the NTT, Falcon's signing path additionally requires floating-point FFT and Gaussian sampling, and SPHINCS+ is dominated by hashing. A device supporting all three would thus need integer-NTT, floating-point, and high-throughput hashing datapaths, motivating the unified, reconfigurable accelerators now appearing in the literature [10], [11], [15]. Figure 3 sketches a general-purpose accelerator template of this kind, in which a control unit invokes a set of dedicated modules that exchange inputs and intermediate results through a shared multi-bank BRAM scratchpad, following the organization of the unified Falcon design [15].

A practical concern for such a shared design is memory arbitration. When several engines are active in the same cycle, for example, a Keccak hash core and the NTT butterfly units, their accesses to the scratchpad must not collide. The template resolves this at schedule time rather than at run time. The multi-bank BRAM is partitioned so that engines expected to run in parallel are mapped to disjoint banks, and the control unit issues a fixed instruction schedule that fixes, for every cycle, which engine reads or writes which bank. Producer and consumer stages are decoupled with ping-pong, or double-buffered, banks, so one engine fills a bank while another drains a second. Because the operand-to-bank assignment is determined at compile time, bank contention is avoided by construction, and no runtime stall or dynamic arbiter is needed, which keeps the control logic small and the datapath fully pipelined [10], [11], [15].

7 Emerging Schemes

To broaden its portfolio beyond structured lattices and hash functions, NIST launched a separate call for additional signatures, known as the 'on-ramp' process. The status report on the second round [14] advanced nine candidates for further study: FAEST, HAWK, MAYO, MQOM, QR-UOV, SDitH, SNOVA, SQIsign, and UOV. These span several families and are motivated by the desire for smaller signatures, faster verification, or hardness assumptions independent of lattices.

The multivariate-quadratic family is the most represented, with UOV and its descendants MAYO, QR-UOV, and SNOVA, which offer very short signatures at the cost of large public keys, a trade-off complementary to the lattice schemes. HAWK is a lattice hash-and-sign scheme in the spirit of Falcon that achieves similarly compact sizes while avoiding

floating-point arithmetic and Gaussian sampling, which could make it far more amenable to constant-time and hardware implementation. SQIsign is an isogeny-based scheme producing the smallest sizes of any candidate, albeit with high computational cost. Finally, FAEST, SDitH, and MQOM apply the 'MPC-in-the-head' methodology to symmetric-key, syndrome-decoding, and multivariate problems, offering conservative assumptions at moderate sizes and speeds. None is yet standardized, and their hardware landscape is still nascent.

8 Open Challenges and Future Directions

Beyond the per-scheme obstacles discussed above, several cross-cutting challenges remain. Foremost is secure and efficient implementation. Falcon's floating-point Gaussian sampling is hard to render constant-time, and side-channel-resistant implementations of all three schemes are an active research area, with masking and fault-attack countermeasures adding significant area and latency. Hardware support is also uneven. Dilithium and, increasingly, SPHINCS+ enjoy mature accelerators, whereas complete Falcon hardware has only recently been demonstrated, which motivates the unified, reconfigurable accelerators that reconcile the schemes' very different arithmetic. At the protocol level, the large SPHINCS+ signatures stress transport protocols and certificate chains, and hybrid classical and post-quantum deployment is needed throughout the transition period. Continued cryptanalysis of the underlying assumptions, together with the maturation of the on-ramp candidates, will ultimately shape how the signature portfolio evolves.

A first cross-cutting concern is the architectural cost of side-channel countermeasures. Protecting the lattice schemes against power and electromagnetic analysis relies on masking, which splits every sensitive variable into several random shares, one more than the masking order. Linear operations such as the NTT simply replicate across the shares and grow roughly linearly in area and time, but the non-linear steps around them, including sampling, rejection, and the arithmetic-to-Boolean conversions, require share interactions whose cost grows quadratically with the masking order, together with a heavy demand for fresh randomness. In hardware, this becomes a quadratic blow-up in both area and latency at higher orders, and the randomness generator and mask-refresh logic often dominate the protected datapath, as reported for masked Kyber and Dilithium [29], [31]. A practical accelerator must therefore budget for this overhead rather than treat masking as a fixed-cost add-on.

A second concern is fault injection. Because signing in Dilithium is gated by a rejection step that tests the response against a fixed norm bound, a single well-placed fault that skips or corrupts that check can release a signature that should have been rejected, and such out-of-bound signatures can leak information about the secret key. Deterministic lattice signatures are especially exposed, because re-running the same input with and without a fault yields a differential that recovers secret coefficients [28], and loop-abort faults that cut short the sampling or the Fiat-Shamir and hash-and-sign computation have been shown to break several lattice schemes [30]. Faults aimed at signature encoding or compression can likewise bypass correctness checks. Countermeasures such as verify-after-sign, redundant or double computation, and consistency checks on the bound raise assurance but add area and latency, so fault resistance, like masking, is an explicit design axis for a secure accelerator rather than an afterthought.

9 Conclusion

The transition to post-quantum digital signatures is well under way, anchored by three standardized schemes that deliberately span two independent hardness assumptions. Dilithium provides a simple, implementation-friendly default. Falcon offers unmatched compactness at the price of implementation complexity. SPHINCS+ supplies conservative, hash-only security at the cost of large signatures. No single scheme dominates, and the right choice depends on the target application's constraints, including bandwidth, latency, code and hardware complexity, and the desired conservatism of the security assumption. As NIST's additional-signature process matures and hardware support broadens, the portfolio will continue to diversify, but

the three schemes surveyed here will form the backbone of quantum-resistant authentication for the foreseeable future.

Acknowledgments: This research was supported by the Slovenian Research and Innovation Agency (ARIS) under Grant P2-0359 (National research program Pervasive Computing).

Conflicts of Interest: The author reports there are no competing interests to declare.

Data availability statement: The data presented in this study are available on request from the corresponding author.

References

[1] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, doi: 10.1137/S0097539795293172.

[2] L. Chen et al., "Report on post-quantum cryptography," NIST, Gaithersburg, MD, USA, NISTIR 8105, 2016.

[3] National Institute of Standards and Technology, "Module-lattice-based digital signature standard," FIPS 204, 2024.

[4] National Institute of Standards and Technology, "Stateless hash-based digital signature standard," FIPS 205, 2024.

[5] P.-A. Fouque et al., "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," Submission to the NIST PQC standardization process, 2020.

[6] S. Bai et al., "CRYSTALS-Dilithium: Algorithm specifications and supporting documentation," Submission to the NIST PQC standardization process, 2021.

[7] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, “The SPHINCS⁺ Signature Framework,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London United Kingdom: ACM, Nov. 2019, pp. 2129–2146. doi: 10.1145/3319535.3363229.

[8] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *J. ACM*, vol. 56, no. 6, pp. 1–40, Sep. 2009, doi: 10.1145/1568318.1568324.

[9] C. Gentry, C. Peikert, and V. Vaikuntanathan, “Trapdoors for hard lattices and new cryptographic constructions,” in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, Victoria British Columbia Canada: ACM, May 2008, pp. 197–206. doi: 10.1145/1374376.1374407.

[10] 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*, vol. 70, no. 2, pp. 747–758, Feb. 2023, doi: 10.1109/TCSI.2022.3219555.

[11] M. Bisheh-Niasar, E. Karabulut, K. Upadhyayula, M. Norris, and B. Pillilli, "Adams Bridge accelerator: Bridging the post-quantum transition," IACR Cryptol. ePrint Arch., 2026/256.

[12] Y. Ouyang *et al.*, “FalconSign: An Efficient and High-Throughput Hardware Architecture for Falcon Signature Generation,” *TCHES*, vol. 2025, no. 1, pp. 203–226, Dec. 2024, doi: 10.46586/tches.v2025.i1.203-226.

[13] P. Karl, J. Schupp, and G. Sigl, “Performance and Communication Cost of Hardware Accelerators for Hashing in Post-Quantum Cryptography,” *ACM Trans. Embed. Comput. Syst.*, vol. 24, no. 5, pp. 1–31, Sep. 2025, doi: 10.1145/3676965.

[14] National Institute of Standards and Technology, "Status report on the second round of the additional digital signature schemes for the NIST post-quantum cryptography standardization process," NISTIR 8610, 2026.

[15] V. Smole, A. Aikata, R. Pilipovic, and S. S. Roy, “Giving Falcon Its Wings: Fast Hardware Acceleration for Signing and Verification Using FN-DSA,” *IEEE Trans. Circuits Syst. I*, vol. 73, no. 9, pp. 6101–6114, Sep. 2026, doi: 10.1109/TCSI.2026.3705146.

[16] M. Schmid, D. Amiet, J. Wendler, P. Zbinden, and T. Wei, "Falcon takes off: A hardware implementation of the Falcon signature scheme," IACR Cryptol. ePrint Arch., 2023.

[17] L. Beckwith, D. T. Nguyen, and K. Gaj, “High-Performance Hardware Implementation of CRYSTALS-Dilithium,” in *2021 International Conference on Field-Programmable Technology (ICFPT)*, Auckland, New Zealand: IEEE, Dec. 2021, pp. 1–10. doi: 10.1109/ICFPT52863.2021.9609917.

[18] S. Pendyala, R. Magesh, E. B. Kavun, and A. Aysu, "Accelerating FALCON: Speed records for FALCON's SamplerZ on Xilinx FPGAs," IACR Cryptol. ePrint Arch., 2025/1490.

[19] X. Yu, Y. Sun, Y. Zhao, H. Kuang, and J. Han, “RVCE-FAL: A RISC-V Scalar-Vector Custom Extension for Faster FALCON Digital Signature,” in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Valencia, Spain: IEEE, Mar. 2024, pp. 1–6. doi: 10.23919/DATE58400.2024.10546713.

[20] T.-T. Nguyen, D.-D. Nguyen, T.-T. Dao, and N.-Q. Luc, “Implementation Efficiency of Falcon Digital Signature Scheme on Arty-7 XC7A35T Board,” *Electronics*, vol. 14, no. 22, p. 4504, Nov. 2025, doi: 10.3390/electronics14224504.

[21] S. Ricci *et al.*, “Implementing CRYSTALS-Dilithium Signature Scheme on FPGAs,” in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, Vienna Austria: ACM, Aug. 2021, pp. 1–11. doi: 10.1145/3465481.3465756.

[22] G. Land, P. Sasdrich, and T. Güneysu, “A Hard Crystal - Implementing Dilithium on Reconfigurable Hardware,” in *Smart Card Research and Advanced Applications*, V. Grosso and T. Pöppelmann, Eds., Cham: Springer International Publishing, 2022, pp. 210–230. doi: 10.1007/978-3-030-97348-3_12.

[23] Q. Norga, S. Kundu, and I. Verbauwhede, “ML-DSA-OSH: An Efficient, Open-Source Hardware Implementation of ML-DSA,” in *2026 Design, Automation & Test in Europe Conference (DATE)*, Verona, Italy: IEEE, Apr. 2026, pp. 1–3. doi: 10.23919/DATE69613.2026.11539720.

[24] D. Amiet, L. Leuenberger, A. Curiger, and P. Zbinden, "FPGA-based SPHINCS+ implementations: Mind the glitch," in Proc. IEEE Int. Symp. Hardware Oriented Security and Trust (HOST), 2020.

[25] Q. Berthet, A. Upegui, L. Gantel, A. Duc, and G. Traverso, "An area-efficient SPHINCS+ post-quantum signature coprocessor," in Proc. IEEE Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW), 2021.

[26] S. Deshpande, Y. Lee, C. Karakuzu, J. Szefer, and Y. Paek, “SPHINCSLET: An Area-Efficient Accelerator for the Full SPHINCS+ Digital Signature Algorithm,” *ACM Trans. Embed. Comput. Syst.*, vol. 24, no. 5, pp. 1–19, Sep. 2025, doi: 10.1145/3728469.

[27] National Institute of Standards and Technology, "Module-lattice-based key-encapsulation mechanism standard," FIPS 203, 2024.

[28] L. Groot Bruinderink and P. Pessl, “Differential Fault Attacks on Deterministic Lattice Signatures,” *TCHES*, pp. 21–43, Aug. 2018, doi: 10.46586/tches.v2018.i3.21-43.

[29] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. Van Vredendaal, "Masking Kyber: First- and Higher-Order Implementations," *TCHES*, pp. 173–214, Aug. 2021, doi: 10.46586/tches.v2021.i4.173-214.

[30] T. Espitau, P.-A. Fouque, B. Gérard, and M. Tibouchi, "Loop-Abort Faults on Lattice-Based Fiat-Shamir and Hash-and-Sign Signatures," in *Selected Areas in Cryptography – SAC 2016*, vol. 10532, R. Avanzi and H. Heys, Eds., Cham: Springer International Publishing, 2017, pp. 140–158. doi: 10.1007/978-3-319-69453-5_8.

[31] V. Migliore, B. Gérard, M. Tibouchi, and P.-A. Fouque, "Masking Dilithium: Efficient Implementation and Side-Channel Evaluation," in *Applied Cryptography and Network Security*, vol. 11464, R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, Eds., Cham: Springer International Publishing, 2019, pp. 344–362. doi: 10.1007/978-3-030-21568-2_17.