FixSampZ: A Fixed-Point Hardware-Efficient Discrete Gaussian Sampler for Falcon Signatures

Pushpendra Singh*, Akshara K B†, Utsav Banerjee* and Navin Kashyap*
*Indian Institute of Science, Bengaluru, India †PSG College of Technology, Coimbatore, India
E-mail: {pushpendrasi, utsav, nkashyap}@iisc.ac.in, kbakshara1@gmail.com

***Abstract*—Falcon, a NIST-standardized post-quantum signature algorithm, offers compact signatures but relies on hardware-intensive floating-point arithmetic. Building on recent theoretical fixed-point formulations, we present a completely floating-point-free hardware architecture for SamplerZ, Falcon's primary signing bottleneck. By utilizing zero-cost hardware bit-sliced retagging of fixed-point data, which software avoids due to instruction overheads, we enforce tighter data formats at every stage boundary without area or latency penalties. Furthermore, executing floor functions as trivial two's-complement arithmetic right shifts eliminates the complex floor-prediction and redundant data-paths required by floating-point designs. Implemented on UltraScale+ FPGA, our fixed-point sampler exactly matches all 1,024 official test vectors and requires 38% fewer LUTs, 21% less FFs and 36% less DSPs compared to the floating-point baseline, with 21% fewer cycles at a higher clock frequency. We also demonstrate that substituting the baseline ChaCha20 PRNG with lightweight NIST-standardized Ascon-XOF128 achieves further 2-3× hardware resource savings with zero sampling latency overhead.**

***Index Terms*—Falcon, FN-DSA, post-quantum cryptography, lattice-based signatures, discrete Gaussian sampling, fixed-point arithmetic, Ascon-XOF128, hardware accelerator, FPGA.**

I. Introduction

Shor's quantum algorithm [1] and its derivatives pose a practical existential threat to classical public key cryptography, with exploitable vulnerabilities anticipated within 1-2 decades [2]. To mitigate this threat in the light of rapid advancements in quantum computing technologies, NIST formalized its first Post-Quantum Cryptography (PQC) standards [3]–[6] in August 2024, encompassing ML-KEM (CRYSTALS-Kyber) [7], ML-DSA (CRYSTALS-Dilithium) [8], SLH-DSA (SPHINCS+) [9] and FN-DSA (Falcon) [10]. While the first three have already been formally released as FIPS 203, FIPS 204 and FIPS 205 respectively, Falcon is slated for formalization as FIPS 206 by 2027 to complete this portfolio.

Characterized by compact signatures and rapid verification, Falcon is highly optimized for memory-constrained and time-sensitive environments, offsetting a marginally slower signature generation phase compared to ML-DSA. However, hardware acceleration of Falcon's signature generation presents significant architectural challenges. Its trapdoor sampler operates over real numbers, relying heavily on IEEE-754 double-precision arithmetic [11]. On FPGAs, standard floating-point hardware IPs are area-intensive, deeply pipelined, and enforce round-to-nearest policies. This strictly conflicts with Falcon's frequent requirement for exact floor operations (e.g., $\lfloor \mu \rfloor$ and $\lfloor x/\ln 2 \rfloor$). Because standard hardware IPs cannot natively compute a floor, designs are forced to calculate two adjacent integer candidates and duplicate downstream logic until the correct boundary resolves. This is evident in state-of-the-art hardware implementation and hardware-software co-designs [12]–[20].

Recently, De Almeida Braga et al. [21] have shown that Falcon does not need floating-point arithmetic at all. They have proposed a secure fixed-point formulation of the entire signing pipeline together with proven magnitude bounds for every intermediate value so that number formats can be assigned statically with overflow excluded by construction. Rényi divergence arguments [21] prove that fixed-point version of Falcon maintains the same theoretical security and correctness. While [21] has provided software reference implementations (available at https://github.com/fixed-point-fndsa), the impact of fixed-point transformations on hardware resource utilization and performance is yet to be evaluated. In this work, we answer this question with concrete hardware implementation results of the fixed-point sampler for Falcon-512 on UltraScale+ FPGA. We adapt the SamplerZ design from [16] to fixed-point arithmetic using bounds from [21]. We streamline the floor function execution directly as a two's complement arithmetic right shift, eliminating redundant prediction and selection data paths. Compared to state-of-the-art samplers, our implementation achieves significant hardware resource savings (up to 38%, 21% and 36% reduction respectively in LUT, FF and DSP counts) while maintaining correct functionality and similar latency. We also demonstrate that replacing the sampler's ChaCha20-based pseudo-random number generator (PRNG) [22] with the NIST lightweight cryptography standard Ascon-XOF128 [23] can provide further 2-3× reduction in hardware resource utilization without affecting accuracy and performance.

II. Background

A. Discrete Gaussian Sampling in Falcon

Falcon is a post-quantum digital signature algorithm based on the Gentry-Peikert-Vaikuntanathan (GPV) hash-and-sign framework and Fast Fourier Sampling (FFS) defined over NTRU lattices [10]. In the signing phase, FFS recursively generates short vectors in the Fast Fourier Transform (FFT) domain. Each FFS leaf involves two discrete Gaussian samplings SamplerZ

This research work is part of the project titled "Development of Security Solutions that Meet the IEC 62351 Standards for Communication Protocols". This initiative is supported by the POWERGRID Center of Excellence in Cyber Security (PGCoE) at the Indian Institute of Science, Bangalore.

(μ_{L}, σ') and SamplerZ (μ_{R}, σ'), both with standard deviation σ' but different means μ_{L} and μ_{R} [10].

SamplerZ is Falcon's trapdoor sampler which draws integers from a discrete Gaussian distribution centered at an arbitrary mean μ with standard deviation σ', where $\sigma_{\min} \leq \sigma' \leq \sigma_{\max}$ [10]. SamplerZ consists of the following sub-functions: UniformBits, BaseSampler and BerExp. Here, UniformBits is used to sample uniformly random bit strings of desired length. BaseSampler draws a half-Gaussian candidate z_0 using a Reverse Cumulative Distribution Table (RCDT). A random sign bit b maps $z_0 \in \mathbb{Z}^+$ to $z \in \mathbb{Z}$ and BerExp performs rejection sampling based on a Bernoulli test on

$$x = \frac{(z-r)^2}{2\sigma'^2} - \frac{z_0^2}{2\sigma_{\max}^2} \tag{1}$$

where $r = \mu - \lfloor \mu \rfloor$. BerExp uses an ApproxExp subroutine to approximately compute an exponential by evaluating a degree-12 polynomial in Horner form with its coefficients fixed by the Falcon specification [10]. Notably, ApproxExp is already defined over 64-bit integers scaled by 2^{63}, but the remaining functions are traditionally floating-point in nature.

B. Related Work

Previous work on hardware implementation of Falcon's SamplerZ are all based on floating-point operations. [13]–[15] presented efficient hardware-software co-designs of Falcon by integrating dedicated floating-point arithmetic units with RISC-V and ARM micro-processors. [16] implemented a high-performance Falcon signature generation accelerator with optimized SamplerZ. [19] presented a low-latency sampler which evaluates two complete exponential approximation polynomials in parallel. Since the fixed-point version of Falcon [21] is very recent, to the best of our knowledge, we are the first to present its hardware implementation and analysis.

III. Fixed-Point Sampler Hardware Design

A. Fixed-Point Representation

Our fixed-point implementation of SamplerZ follows the formulation from [21]. A value a is represented by a k-bit two's complement integer X with $p = k - 1$ bits of signed mantissa and an integer magnitude tag m such that

$$a = X \cdot 2^{m-p} \text{ with } |a| < 2^m. \tag{2}$$

We use $k = 64$ and $p = 63$ throughout this work and denote the fixed-point format carrying tag m as Q$m.f$ with $f = p - m = 63 - m$. Therefore, Q0.63 holds values in $(-1, 1)$ and Q7.56 holds values in $(-128, 128)$. Addition and subtraction of two operands sharing m and p are exact. Multiplication produces a $2k$-bit product whose natural tag is the sum of input tags. Therefore, recovering a k-bit result requires a shift followed by rounding. Retagging [21] converts data formats — widening m requires rounding, while narrowing m is conditionally lossless. Each retag costs an explicit shift instruction in software, but can be realized using simple wiring in hardware.

B. Bounds for SamplerZ

We rely on the bound calculations from [21]. For Falcon-512, with $\sigma_{\min} = 1.277833697$ and $\sigma_{\max} = 1.8205$, we have: $z_0 \in [0, 18]$, $r \in [0, 1)$, $z \in \{-18, \ldots, 19\}$ and $|z - r| \leq 20$. Therefore, we have following bounds on x and $\lfloor x/\ln 2 \rfloor$:

$$0 \leq x \leq x_{\max} = \frac{19^2}{2\sigma_{\min}^2} - \frac{18^2}{2\sigma_{\max}^2} = 61.662034 < 62 \tag{3}$$

$$x/\ln 2 \leq x_{\max}/\ln 2 = 88.959 \Rightarrow s = \lfloor x/\ln 2 \rfloor \leq 88 \tag{4}$$

For Falcon-1024, $\sigma_{\min} = 1.298280334$ gives $x_{\max} = 58.207586$ and $\sigma_{\max}$ is 1.8205 for both parameter sets, a fact we exploit later in our implementation. We further note that $dss = 1/(2\sigma'^2) \in [0.1509, 0.3063]$ and $ccs = \sigma_{\min}/\sigma' \in [0.7019, 1]$ both satisfy $m = 0$. Two consequences are used repeatedly. First, Equation 3 bounds x below 2^6, so a tag of $m = 7$ carries it with a bit to spare. Second, Equation 4 gives $s \leq 88$, which is larger than 63.

C. Hardware Architecture

Figure 1 shows the top-level architecture of our fixed-point SamplerZ module for Falcon signing. It is an adaptation of the design from [16] (available open-source at https://github.com/YiOuyang1/FalconSign/tree/main/sampler) along with floating-point-to-fixed-point transformations and other hardware-specific optimizations. The PRNG and Refill_Control modules are used to realize the UniformBits functionality which supplies randomness, and they can be carried over directly from the floating-point design. Given the input μ and σ', the Pre_Sample module computes $\lfloor \mu \rfloor$, $r = \mu - \lfloor \mu \rfloor$, $dss = 1/(2\sigma'^2)$ and

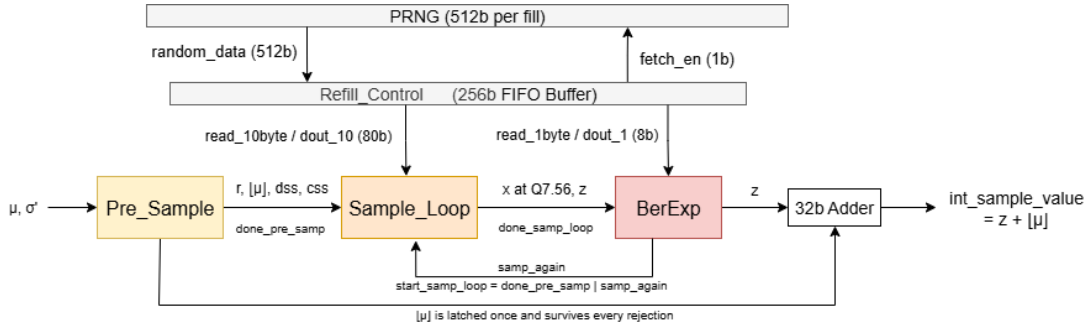

Fig. 1. Top-level block diagram of our fixed-point SamplerZ hardware.

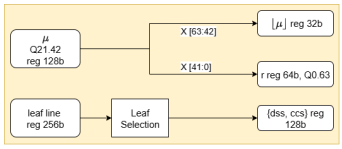

Fig. 2. Block diagram of our fixed-point Pre_Sample module.

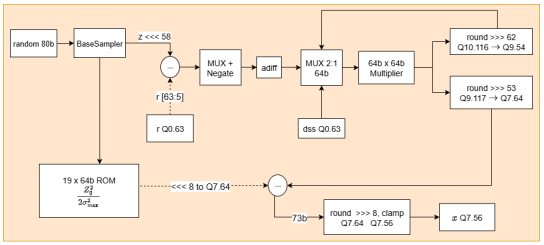

Fig. 3. Block diagram of our fixed-point Sample_Loop module.

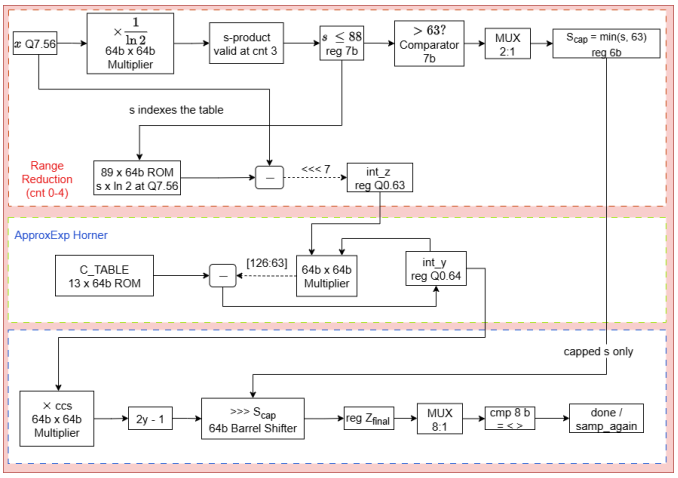

Fig. 4. Block diagram of our fixed-point BerExp module.

$ccs = \sigma_{\min}/\sigma'$. Next, the Sample_Loop module draws the half-Gaussian candidate using the RCDT and computes x as per Equation 1. Finally, the BerExp module performs rejection test on x. If x is accepted, then the sampler output is an integer value $z + \lfloor \mu \rfloor$, otherwise the Sample_Loop and BerExp are invoked again until a valid sample is found. Note that Pre_Sample is outside the rejection loop and is performed only once per pair of samples (for the FFS leafs μ_{L} and μ_{R}). Irrespective of the number of rejections, $\lfloor \mu \rfloor$ is latched once and simply re-used till a sample is accepted. Also, the two randomness ports are asymmetric. The Sample_Loop module draws 10 bytes per attempt for the base sampler, whereas BerExp draws 1 byte at a time and stops as soon as the comparison resolves. The Pre_Sample, Sample_Loop and BerExp modules are modified significantly for the optimized fixed-point conversion, as explained next.

1) Pre_Sample: In the fixed-point design shown in Figure 2, μ is formatted as Q21.42, allowing its integer and fractional parts to be easily extracted as disjoint bit fields: $\lfloor \mu \rfloor = \{\{10\{X_\mu[63]\}\}, X_\mu[63{:}42]\}$ and $r = \{0, X_\mu[41{:}0], 0^{21}\}$, where X_μ denotes the two's complement integervin the fixed-point representation of μ as per Equation 2. Slicing μ_{L} and μ_{R} from a single 128-bit input word captures all 4 outputs $\lfloor \mu_{\mathrm{L}} \rfloor$, $\lfloor \mu_{\mathrm{R}} \rfloor$, $r_{\mathrm{L}} = \mu_{\mathrm{L}} - \lfloor \mu_{\mathrm{L}} \rfloor$ and $r_{\mathrm{R}} = \mu_{\mathrm{R}} - \lfloor \mu_{\mathrm{R}} \rfloor$ in one clock cycle, eliminating the floating-point baseline's 2-cycle pipeline time-sharing. Additionally, pre-computing the per-leaf constants dss and ccs and providing them as inputs removes any shared multiplier dependency. Stripped of all multipliers and data-path adders, the Pre_Sample module reduces to a simple 4-bit state counter, cutting latency from 11 to 6 cycles. As a system-level trade-off, storing two $\{dss, ccs\}$ pairs instead of four $1/\sigma'$ values per 256-bit input word doubles the leaf storage and halves the caller's address stride.

2) Sample_Loop: The floating-point sample loop from [16] makes 5 requests to the shared multiplier. Since 2 of them compute $1/(2\sigma'^2)$ from $1/\sigma'$, they are skipped by pre-computing dss. The third computes $z_0^2/(2\sigma_{\max}^2)$. Since z_0 takes at most 19 values and $\sigma_{\max} = 1.8205$ for both Falcon parameter sets, this term is a constant table with no dependence on the security level, and we replace it with a 19 × 64-bit ROM. The remaining two are for a squaring and a multiplication, which we perform on a single local 64×64 multiplier used in two passes. As shown in Figure 3, the structural center of the fixed-point Sample_Loop is the feedback path from the pass-1 rounder back into the operand multiplexer. It allows a single 64×64 multiplier to serve both the squaring and the scaling by dss. Taking $|z - r|$ first makes both passes unsigned. The whole data-path is 1 multiplier, 3 subtractions, 4 multiplexers, a comparator bank and a distributed ROM. Both terms are formed at Q7.64, so the 73-bit subtraction is exact and the result is rounded exactly once at the output.

3) BerExp: The floating-point baseline's inability to natively compute $s = \lfloor x/\ln 2 \rfloor$ necessitates a redundant shadow rail (a second 64×64 multiplier, accumulator and multiplexer) to evaluate 2 candidate polynomial states concurrently. In our fixed-point BerExp implementation shown in Figure 4, s is directly extracted as a bit field, entirely eliminating this shadow rail (dashed lines in Figures 3 and 4) and yielding our design's largest area saving. Furthermore, baseline multiplications by 2^{63} for integer conversion are reduced to zero-cost hard-wired retags. For example, shifting r_{be} left by 7 bits trivially converts it from Q7.56 to Q0.63. The $s \times \ln 2$ product is replaced by an 89-entry ROM indexed by the uncapped $s \leq 88$, whereas only the saturated $S_{\mathrm{cap}} = \min(s, 63)$ routes to the barrel shifter. While the core Horner approximation implementation remains bit-identical to the baseline, removing the shadow evaluation shortens the BerExp schedule from 48 down to 40 cycles.

D. Choice of PRNG

While previous hardware implementations of Falcon's sampler [12]–[20] have all used ChaCha20 [22] as its core PRNG, we also explore the effect of replacing it with Ascon-XOF128 [23] as a lightweight alternative. Independent of the PRNG, we maintain an identical interface, thus leaving the downstream sampler circuitry unaffected. After absorbing the input seed into its 320-bit internal state, Ascon squeezes uniformly

TABLE I
COMPARISON OF FIXED-POINT SAMPLERZ HARDWARE IMPLEMENTATIONS WITH DIFFERENT PRNGS BASED ON CHACHA20 AND ASCON-XOF128

SamplerZ Design	LUTs	FFs	DSPs	Max Freq (MHz)
with ChaCha20 PRNG	9,118	8,428	48	240
with Ascon-XOF128 PRNG ($u = 1$)	4,359	2,351	48	237
with Ascon-XOF128 PRNG ($u = 2$)	4,864	2,352	48	237
with Ascon-XOF128 PRNG ($u = 3$)	5,488	2,352	48	237
with Ascon-XOF128 PRNG ($u = 4$)	6,006	2,350	48	237
with Ascon-XOF128 PRNG ($u = 6$)	7,388	2,354	48	231

random outputs into the 512-bit buffer at the rate of 64 bits per permutation. The Ascon-p[12] permutation involves 12 iterations of its internal round function [23]. We explore round-based / partially-unrolled Ascon designs with unrolling factor $u \in \{1, 2, 3, 4, 6\}$ which compute u rounds per cycle and completes the 12-round permutation in $12/u$ cycles [24]. Note that this PRNG replacement is limited only to Falcon-512 since Ascon-XOF128 provides NIST Security Level I.

IV. IMPLEMENTATION RESULTS

We implement our fixed-point SamplerZ designs for Falcon-512 with Xilinx Vivado for the Zynq UltraScale+ ZCU104 evaluation board with the XCZU7EV-2 FPGA MPSoC device. The hardware resource utilization and maximum operating frequency for our designs with different PRNG configurations are shown in Table I. Clearly, using Ascon-XOF128 as the PRNG provides up to $2\times$ and $3.6\times$ savings in LUT and FF respectively. All our designs require only 48 DSP slices (12 each for the four 64×64 multipliers). Our fixed-point sampler perfectly matches all 1,024 test vectors from the Falcon specification [10], thus confirming structural correctness. However, quantization of μ and σ' to Q21.42 introduces a theoretical divergence from the floating-point reference. Bernoulli threshold perturbations are bounded near 2^{-42}, yielding a negligible 2^{-34} per-comparison error probability if the decision boundary falls within this narrow window.

Because the rejection loop introduces variable latency, we calculate the expected cycle count as $E = \sum_i P_i \cdot \text{cycles}_i$. Using failure probabilities P_i from [19], we determine cycles_i through RTL simulation of the 1,024 test vectors. A single call requires $\text{cycles}_i \approx 50 + 44i$ cycles, yielding an expected $E = 82.32$ cycles. For a full 2-sample task integrated with the PRNG (either ChaCha20 or Ascon-XOF128) and measured across 20,000 simulated instances, the expected latency is $E = \sum_{i,j} P_i \cdot P_j \cdot \text{cycles}_{i,j} = 158.75$ cycles. A zero-rejection task averages 94 cycles, bounded by a 108-cycle worst case.

Table II compares our fixed-point hardware sampler with state-of-the-art implementations on UltraScale+ FPGA. Compared to the floating-point sampler baseline from FalconSign [16], our design requires 38% less LUTs, 21% less FFs and 36% less DSPs, and also operates at 240 MHz versus 185 MHz with 21% less cycle count. Although the floating-point sampler design from Bi-SamplerZ [19] has lower cycle count and latency, it has higher resource utilization, thus making our fixed-point design significantly better in terms of area-time product. Both [13] and [15] have low resource utilization but have significantly higher latency. Also, these are both hardware-software co-designs with a general-purpose micro-processor, thus making it difficult to performa direct comparison.

V. CONCLUSION

In this work, we have presented the first fixed-point hardware implementation of Falcon's discrete Gaussian sampler based on the theory proposed by [21]. Using zero-cost hardware retagging and bit-slice flooring, our fixed-point sampler eliminates all floating-point dependencies, redundant shadow evaluations and complex arbitration. Its data-path is 10 bits more precise than the baseline [16] and needs 38% fewer LUTs, 21% less FFs and 36% less DSPs, and requires 21% fewer cycles at a higher clock frequency. We further demonstrate that substituting the baseline ChaCha20 PRNG with a lightweight NIST-standardized Ascon-XOF128 PRNG achieves further 2-3$\times$ hardware resource savings with zero sampling latency overhead at NIST Security Level I. Although our demonstration is for the Falcon-512 parameters, our design can be easily extended to Falcon-1024 with minimal changes. Our design has been made publicly available on GitHub at [25].

Future work will explore further optimizations such as dual-data-path architecture [19], multiplier sharing and pipelining, as well as incorporation of side-channel countermeasures. Integration with full Falcon accelerators and RISC-V-based hardware-software co-design [13], [17] will also be pursued.

AI USAGE STATEMENT

Claude Opus 5.5 was used to generate the initial Ascon-XOF128 System Verilog RTL, which the authors extensively modified and verified against the NIST specifications. Gemini was used for language refinement. The authors take full responsibility for technical contents of this work, and confirm that no AI contributed to the research ideas, results or conclusions.

TABLE II
COMPARISON OF RESOURCE UTILIZATION AND TWO-SAMPLE PERFORMANCE WITH STATE-OF-THE-ART ON ULTRASCALE+ FPGA

Design	**H/W Resource Utilization LUT / FF / DSP**	**Max Freq (MHz)**	**No-Reject Cycles**	**No-Reject Latency**	**Expected Cycles**	**Expected Latency**
RVCE-FAL [13]	4,761 / 692 / 9	83	208	2.506 μs	—	—
AHS Base [15]	3,456 / 732 / 116	117	3,382	28.906 μs	—	—
FalconSign [16]	14,710 / 10,731 / 76	185	137	740.5 ns	230.83	1247.7 ns
Bi-SamplerZ [19]	14,327 / 10,841 / 85	150	59	393.3 ns	106.08	707.2 ns
This Work (with ChaCha20 PRNG)	**9,118 / 8,428 / 48**	**240**	**108**	**450 ns**	**183.14**	**763.08 ns**

REFERENCES

[1] P. W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *Annual Symposium on Foundations of Computer Science (FOCS)*, 1994, pp. 124–134.

[2] D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds., *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer, 2009.

[3] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on Post-Quantum Cryptography," National Institute of Standards and Technology Internal Report 8105, 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf

[4] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone, and Y.-K. Liu, "Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process," National Institute of Standards and Technology Internal Report 8240, 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf

[5] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, "Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process," National Institute of Standards and Technology Internal Report 8309, 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf

[6] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone, and Y.-K. Liu, "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process," National Institute of Standards and Technology Internal Report 8413, 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf

[7] NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS 203, 2024.

[8] ——, "Module-Lattice-Based Digital Signature Standard," FIPS 204, 2024.

[9] ——, "Stateless Hash-Based Digital Signature Standard," FIPS 205, 2024.

[10] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-Fourier Lattice-Based Compact Signatures over NTRU," Specification submitted for NIST Post-Quantum Cryptography Standardization, 2020. [Online]. Available: https://falcon-sign.info/falcon.pdf

[11] IEEE, "IEEE Standard for Floating-Point Arithmetic," no. 754-2019, pp. 1–84, 2019.

[12] M. Schmid, D. Amiet, J. Wendler, P. Zbinden, and T. Wei, "Falcon Takes Off—A Hardware Implementation of the Falcon Signature Scheme," Cryptology ePrint Archive, Paper 2023/1885, 2023. [Online]. Available: https://eprint.iacr.org/2023/1885

[13] 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 *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.

[14] Y. Lee, J. Youn, K. Nam, H. H. Jung, M. Cho, J. Na, J.-Y. Park, S. Jeon, B. G. Kang, H. Oh, and Y. Paek, "An Efficient Hardware/Software Co-Design for FALCON on Low-End Embedded Systems," *IEEE Access*, vol. 12, pp. 57 947–57 958, 2024.

[15] E. Karabulut and A. Aysu, "A Hardware-Software Co-Design for the Discrete Gaussian Sampling of FALCON Digital Signature," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2024, pp. 90–100.

[16] Y. Ouyang, Y. Zhu, W. Zhu, B. Yang, Z. Zhang, H. Wang, Q. Tao, M. Zhu, S. Wei, and L. Liu, "FalconSign: An Efficient and High-Throughput Hardware Architecture for Falcon Signature Generation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2025, no. 1, pp. 203–226, 2025.

[17] R. Shrivastava and U. Banerjee, "Hardware-Software Co-Design of Post-Quantum FALCON Digital Signature Scheme with RISC-V," in *IEEE International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*, 2025, pp. 121–127.

[18] S. Pendyala, R. Magesh, E. B. Kavun, and A. Aysu, "Outrunning the Millennium FALCON: Speed Records for FALCON on FPGAs," Cryptology ePrint Archive, Paper 2025/1490, 2025. [Online]. Available: https://eprint.iacr.org/2025/1490

[19] B. Zhao, G. Alsuhi, H. Saleh, and B. Mohammad, "Bi-SamplerZ: A Hardware-Efficient Gaussian Sampler Architecture for Quantum-Resistant Falcon Signatures," *arXiv preprint arXiv:2505.24509*, 2025.

[20] M. Wang, A. Li, T. Huang, Y. Wang, D. Liu, X. Zou, and J. Lu, "FALCON-Wings: A Fine-Grained Hardware Accelerator for Falcon Signature Generation with an Efficient Dual Channel Sampler and Versatile Computing Fabric," in *IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2026.

[21] D. D. A. Braga, P.-A. Fouque, B. Lachguel, and T. Prest, "Toward a Secure Fixed-Point Implementation of the Falcon Signature Scheme," in *Annual International Cryptology Conference (CRYPTO)*, 2026, pp. 166–197.

[22] D. J. Bernstein, "ChaCha, a Variant of Salsa20," 2008. [Online]. Available: https://cr.yp.to/chacha/chacha-20080120.pdf

[23] M. S. Turan, K. A. McKay, D. Chang, J. Kang, and J. Kelsey, "Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication SP 800-232, 2025.

[24] S. N. Shende and U. Banerjee, "The Fact of the MATTER: Efficient Hardware Accelerators for Wide-Block Memory Encryption," in *IEEE International Conference on Intelligent Computing and Systems at the Edge (ICEdge)*, vol. 1, 2025, pp. 1–7.

[25] P. Singh, "Hardware Implementation of Fixed-Point Discrete Gaussian Sampler for Falcon," 2026. [Online]. Available: https://github.com/Pushpendra-cell/fixsampz_rtl