

ML-DSA-OSH: An Efficient, Open-Source Hardware Implementation of ML-DSA

Quinten Norga
COSIC, KU Leuven
Leuven, Belgium
quinten.norga@esat.kuleuven.be

Suparna Kundu
COSIC, KU Leuven
Leuven, Belgium
suparna.kundu@esat.kuleuven.be

Ingrid Verbauwhede
COSIC, KU Leuven
Leuven, Belgium
ingrid.verbauwhede@esat.kuleuven.be

Abstract—ML-DSA is a post-quantum lattice-based digital signature algorithm (DSA) that the National Institute of Standards and Technology (NIST) recently standardized as FIPS 204. Remarkably, there are only a handful of published hardware designs and no open-source hardware implementations of complete ML-DSA. In this work, we present an efficient open-source hardware (OSH) design of ML-DSA, based on a Dilithium implementation by Beckwith et al. (FPT 2021). We also discuss the required modifications for migrating existing CRYSTALS-Dilithium implementations to match FIPS 204. Through optimized instruction scheduling in the ML-DSA rejection loop, which enables the pre-computation of critical variables, the average signing latency is improved by 16 – 36%.

Index Terms—PQC, ML-DSA, FPGA, Open-Source Hardware

I. INTRODUCTION

Recently, NIST published Federal Information Processing Standard (FIPS) specifications for CRYSTALS-Dilithium (FIPS 204) and CRYSTALS-Kyber (FIPS 203) and renamed them ML-DSA [1] and ML-KEM [2], respectively. Although ML-DSA and Dilithium v3 share most of the primitive components, there are some differences between them. Only a handful of the Field Programmable Gate Array (FPGA) designs of ML-DSA are published, and almost no FPGA (RTL) implementations of complete ML-DSA are open-source.

A. Related Work

Open-source hardware implementations of Dilithium were presented by Beckwith et al. [3] and Land et al. [4]. Additionally, the CHIPS Alliance project has released their PQC IP core (Adam’s Bridge), containing an ML-DSA implementation [5] which suffers from a high memory (BRAM + FF) and area footprint. Alternatively, several designs were proposed for which the source code is not made available [6]–[9].

B. Contribution

We propose an efficient, ‘combined’ hardware implementation of the NIST DSA standard: ML-DSA (FIPS 204), which supports all ML-DSA operations (KeyGen, SigGen and SigVer) for all parameter sets (NIST II, III & V). The total utilization of our design is 54,942/29,309/29/16 LUTs/FFs/BRAMs/DSPs. Our source code (RTL) is available open-source¹. We reduce the latency (clock cycles, average)

for signature generation (NIST II, III & V) by 16% - 36% compared to prior work through optimized operation scheduling.

II. PRELIMINARIES

For a description ML-DSA, we refer to FIPS 204 [1]. All vectors and matrices are denoted by bold lowercase letters (e.g., \mathbf{v}) and bold upper case letters (e.g., \mathbf{A}), respectively. Additionally, a variable in its NTT form is denoted by a circumflex (e.g., $\hat{\mathbf{A}}$).

A. ML-DSA Signature Generation

The signature generation algorithm generates a signature σ from a secret key sk , message M and optional context string ctx (formatted to M'). A valid signature is generated through rejection sampling: a commitment \mathbf{w}_1 is computed, a pseudo-random challenge c is derived, and a response \mathbf{z} is computed. Together with a hint \mathbf{h} , they are used to compose a signature σ , which is released when it is valid.

B. Differences with CRYSTALS-Dilithium

This following list could serve as a starting point when upgrading an implementation of Dilithium to align with the NIST standardized ML-DSA:

- **Domain Separation:** to ensure independence between difference instantiations of seed expansion (KeyGen) and the context string ctx in M' (Sign).
- **Parameters and Variables:** to ensure the properties beyond strong unforgeability [10], a 64-byte hash of the public key (tr) [1, Alg. 6 (Line 9)] is added to the secret key sk . Also, for ML-DSA-65 and ML-DSA-87, the length of commitment hash \tilde{c} is increased to 384 and 512 bits, respectively [1, Alg.7 (Line 15)].
- **Hedged and Deterministic Signing:** FIPS 204 specifies a hedged and deterministic variant of the signing procedure [1, Alg. 2 (Line 5)]. The hedged variant requires the generation of a 32-byte random string rnd .

III. PROPOSED ARCHITECTURE AND OPTIMIZATIONS

Our implementation is modified from the high-speed CRYSTALS-Dilithium design proposed by Beckwith et al. [3]. We integrate all modifications presented in Section II-B, as well as increase the robustness of the design to support the signing of a wider range of message lengths.

¹<https://github.com/KULeuven-COSIC/ML-DSA-OSH>

TABLE I
OVERVIEW AND COMPARISON OF PQC DSA HARDWARE IMPLEMENTATIONS (XILINX VIVADO 2021.1).

Scheme	Works	Platform	Latency [cc & μ s]			Area [LUT/FF/DSP/BRAM] (or mm^2 for ASIC)
			KeyGen	SigGen	SigVer	
Dilithium-II (v3.1)	[4]	A7 @ 163 MHz	18,761 115	76,613 470	19,687 121	27,433/10,681/45/15
	[8]	A7 @ 114.6 MHz	5,251 46	33,503 293	4,936 43	20,621/8,104/16/8
Dilithium-III (v3.1)	[4]	A7 @ 145 MHz	33,102 228	123,218 850	32,050 221	30,900/11,372/45/21
	[8]	A7 @ 109 MHz	7,871 72	53,577 491	7,610 70	23,378/9,079/8/20
Dilithium-V (v3.1)	[4]	A7 @ 140 MHz	50,982 363	145,192 1,042	52,712 377	44,653/13,814/45/31
	[8]	A7 @ 114.3 MHz	12,483 109	72,179 632	11,825 103	28,791/11,338/24/8
ML-DSA-87	[7]	US+ @ 391 MHz	63,200 161	113,900 291	67,900 173	13,975/6,845/35/4
	[5]	5nm @ 600 MHz	15,600 26	106,400 177	18,800 31	0.0366
Dilithium (v3.1)	[3]	US+ @ 256 MHz	4,875/8,291/14,037 19/32/55	29,876/49,437/55,070 117/193/215	6,582/9,724/13,642 26/38/53	53,907/28,435/16/29
	[6]	A7 @ 96.9 MHz	4,172/5,851/8,765 43/60/90	28,091/44,706/48,996 290/461/506	4,422/6,181/9,039 46/64/93	29,998/10,366/11/10
ML-DSA	Ours	US+ @ 230 MHz	4,872/8,291/14,033 21/36/61	24,283/32,800/37,592 106/143/160	6,637/9,730/14,642 29/42/64	54,942/29,309/16/29

A. Arithmetic Operators

The arithmetic operator modules are used to compute all polynomial arithmetic: addition, subtraction, and Number Theoretic Transform (NTT)-based multiplication. The modular multiplier uses Barrett reduction, requiring only shifts and additions in hardware. The 2×2 butterfly units support both Cooley-Tukey and Gentlemen-Sande operations for NTT and INTT operations, as proposed in [3], [11].

B. Polynomial Sampler and Hashing

Being a costly operation in lattice-based crypto, the authors in [3] propose to instantiate three Keccak cores for the polynomial sampling and hashing. Two units are dedicated to sampling the public matrix \mathbf{A} , and the third is for all other sampling and hashing operations. Sufficient memory (BRAM) is instantiated to ensure that the public matrix can be fully stored, minimizing stalling during the signing procedure.

C. Operation Scheduling

ML-DSA-OSH follows a centralized processor architecture, in which most of the operation scheduling is controlled through a single state machine. We schedule the operations such that the arithmetic units are maximally occupied (minimal stalling) and consume the data (inputs) as soon as possible.

The authors of [3] divide the signature generation into three stages: a pre-computation phase (stage 1) and two rejection loop stages (stage 2-3). The pre-computation phase decodes and transforms the secret key to the NTT domain, the first rejection loop stage generates the commitment \mathbf{w} , and the second computes the response \mathbf{z} and hint \mathbf{h} . We optimize the total latency of the signature generation by ensuring the the polynomial multiplication between public matrix \mathbf{A} and \mathbf{y} (stage 2) starts as soon as both components have been generated in stage 1. Previously, for the medium and high-

security level, the architecture would unnecessarily stall when the NTT computation of \mathbf{y} exceeded the sampling of \mathbf{A} .

IV. PERFORMANCE EVALUATION

We now present the performance results of our ML-DSA hardware (FPGA) implementation which support key generation, signature generation and signature verification (NIST II, III & V) in Table I. We verify the functional correctness of our implementation using the NIST ACVP testvectors.

As expected, the modifications presented in Section II-B results in a minor ($\pm 1\%$) increase in area overhead, compared to Dilithium (v3.1) [3]. Compared to the low-area design by Zhao et al. [6], our designs require $1.83/2.82/1.45/2.9 \times$ more LUT/FF/DSP/BRAM. For medium security parameters, our design requires $36.05/142.61/63.66 \mu$ s for key generation, signing, and verification, respectively. Interestingly, our modifications and improvements result in a reduction of 16% - 36% in (average) clock cycles for the signature generation operation. A contributing factor is ensuring the multiplication $\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{y})$ starts as soon as both multiplicands are computed.

V. CONCLUSION

In this work, we presented an efficient, open-source hardware implementation (ML-DSA-OSH) of the new NIST PQC standard ML-DSA, directly enabling different avenues of future research. Our design is highly flexible, supporting all three security levels and digital signature routines. Our improved signature generation instruction scheduling results in signature generation latency of $106/143/160 \mu$ s for NIST II - V variants.

ACKNOWLEDGMENT

This work was partially supported by Horizon 2020 ERC Advanced Grant (101020005 Belfort), Cybersecurity Research Program Flanders (CRPF) with reference number VOEWICS02, Belgian-QCI (3E230370) (see beqci.eu) and Intel Corporation.

REFERENCES

- [1] NIST, “FIPS 204 Module-Lattice-Based Digital Signature Standard,” Online. Accessed 10th February, 2025, 2024. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf>
- [2] —, “Module-Lattice-Based Key-Encapsulation Mechanism Standard,” Online. Accessed 10th April, 2025, 2024. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf>
- [3] L. Beckwith, D. T. Nguyen, and K. Gaj, “High-performance hardware implementation of crystals-dilithium,” in *2021 International Conference on Field-Programmable Technology (ICFPT)*, 2021, pp. 1–10.
- [4] 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.
- [5] CHIPS Alliance, “Adam’s bridge,” Online. Accessed 18th February, 2025. [Online]. Available: <https://github.com/chipsalliance/adams-bridge>
- [6] C. Zhao, N. Zhang, H. Wang, B. Yang, W. Zhu, Z. Li, M. Zhu, S. Yin, S. Wei, and L. Liu, “A compact and high-performance hardware architecture for crystals-dilithium,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 1, p. 270–295, Nov. 2021. [Online]. Available: <https://tches.iacr.org/index.php/TCHES/article/view/9297>
- [7] N. Gupta, A. Jati, A. Chattopadhyay, and G. Jha, “Lightweight hardware accelerator for post-quantum digital signature crystals-dilithium,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 8, pp. 3234–3243, 2023.
- [8] Z. Wu, R. Chen, Y. Wang, Q. Wang, and W. Peng, “An efficient hardware implementation of crystal-dilithium on fpga,” in *Information Security and Privacy*, T. Zhu and Y. Li, Eds. Singapore: Springer Nature Singapore, 2024, pp. 64–83.
- [9] K. Raj, P. Ravi, T. K. Chia, and A. Chattopadhyay, “Improved ML-DSA hardware implementation with first order masking countermeasure,” *Cryptology ePrint Archive*, Paper 2024/1817, 2024. [Online]. Available: <https://eprint.iacr.org/2024/1817>
- [10] C. Cremers, S. Düzlülü, R. Fiedler, M. Fischlin, and C. Janson, “Buffing signature schemes beyond unforgeability and the case of post-quantum signatures,” in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1696–1714.
- [11] D. T. Nguyen, V. B. Dang, and K. Gaj, “A high-level synthesis approach to the software/hardware codesign of ntt-based post-quantum cryptography algorithms,” in *2019 International Conference on Field-Programmable Technology (ICFPT)*, 2019, pp. 371–374.