Post-Quantum Signatures on RISC-V with Hardware Acceleration

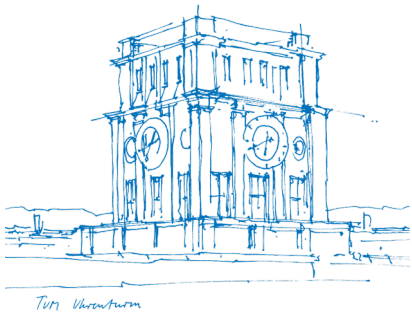
Patrick Karl[1], Jonas Schupp[1], Tim Fritzmann[1,2], Georg Sigl[1,3]

[1]Technical University of Munich
TUM School of Computation, Information and Technology
Chair of Security in Information Technology

[2]Now at Infineon Technologies

[3]Fraunhofer Institute for Applied and Integrated Security

February 27, 2023



Introduction

- NIST process to standardize quantum-secure crypto portfolio
  - Key-Encapsulation Mechanisms (KEMs)
  - Digital Signatures
- A lot of research for efficient implementations
  - Hardware or software
  - High-performance or resource-constrained
- Major focus on KEMs, less on signatures

$\rightarrow$ How would PQ-signatures benefit from accelerators proposed for KEMs?

Contribution

- Adapt accelerators to Dilithium and Falcon (verification only)
- Evaluation of accelerated RISC-V design
- Globalfoundries 22nm ASIC layout
- Exemplary usecase: TLS 1.3 handshake

TLS 1.3 in IoT scenario

- Handshake for mutual authentication
  - Requires signing and verification
  - → Dilithium as generic scheme
- Certificate verification
  - Requires verification only
  - → Falcon with small signatures
- Platform for Dilithium support with Falcon acceleration “for free”

Overview

CRYSTALS-Dilithium and Falcon

- Lattice-based schemes
  - Both selected by NIST for standardization
  - Both allow for fast polynomial arithmetic
- Dilithium overall efficient scheme
  - $|pk| + |\sigma| = 1,312B + 2,420B$ (NIST-2)
- Falcon small signatures, fast verification
  - Floating-point during signing
  - $|pk| + |\sigma| = 897B + 666B$ (NIST-1)
- Frequent operations:
  - Hashing and Random Number Generation using `shake256`
  - Polynomial multiplication using Number Theoretic Transform (NTT)

CRYSTALS-Dilithium and Falcon

- `shake256` from SHA3 standard
- NTT based polynomial arithmetic

→ Integrate HW acceleration for both operations

Algorithm Dilithium Verify

Require: Public Key pk, message M, signature $\sigma = (\tilde{c}, \boldsymbol{z}, \boldsymbol{h})$
Ensure: Accept or reject

1: $\boldsymbol{A} \leftarrow \text{ExpandA}(\rho)$
2: $\mu \leftarrow H(\, H(\rho \parallel \boldsymbol{t_1}) \parallel M)$
3: $c \leftarrow \text{SampleInBall}(\tilde{c})$
4: $\boldsymbol{w_1'} \leftarrow \text{UseHint}(\boldsymbol{h}, \boldsymbol{Az} - c\boldsymbol{t_1} \cdot 2^d)$
5: **if** $\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta$ AND $\tilde{c} = H(\mu \parallel \boldsymbol{w_1'})$ AND # of 1's in $\boldsymbol{h} \leq \omega$ **then**
6: **return** *accept*
7: **else**
8: **return** *reject*
9: **end if**

CRYSTALS-Dilithium and Falcon

- `shake256` from SHA3 standard
- NTT based polynomial arithmetic

→ Integrate HW acceleration for both operations

Algorithm Falcon Verify

Require: public key h, message M, $\sigma = (r, s_2)$
Ensure: Accept or reject

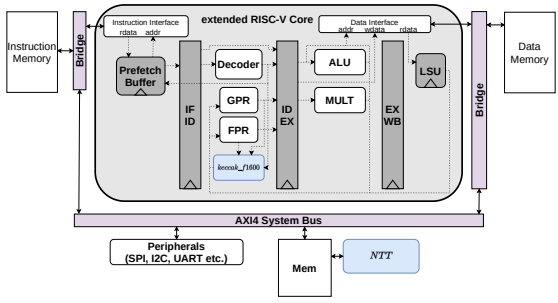
1: $c \leftarrow \texttt{HashToPoint}(r \parallel M)$
2: $s_1 \leftarrow c - s_2 h$
3: **if** $\| (s_1, s_2) \|^2 \leq \lfloor \beta^2 \rfloor$ **then**
4: *accept*
5: **else**
6: *reject*
7: **end if**

Overview

Design: SHAKE256

1. Loosely-coupled, connected to system-bus
   - Performant, easy integration
   - Data transfer might be bottleneck

2. Tightly-coupled, connected to registers
   - complex integration/register management
   - state can fully reside in registers using FPU



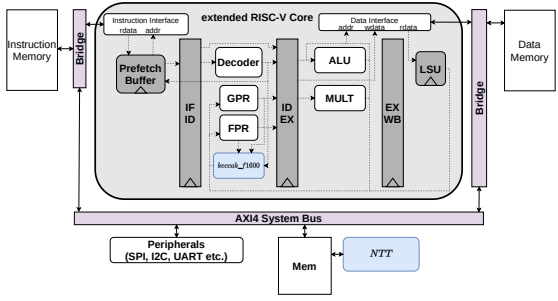
→ Tightly-coupled approach from [FSS20]

→ Single `Keccak` round connected to GPR and FPR

Design: NTT-based Polynomial Arithmetic

- Generic, configurable NTT accelerator proposed in [Fri+21]
  - Support for different NTT flavors
  - Modulus up to 39-bit

- Optimize it for our use-case:
  - Only 24-bit modulus required
  - Remove configuration options

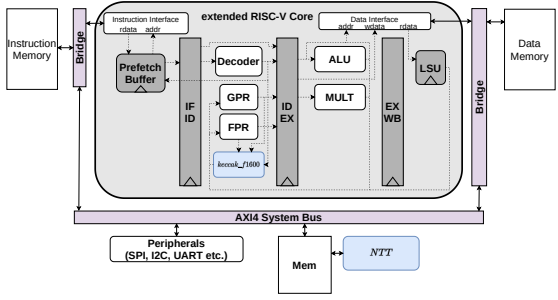


→ Loosely-coupled NTT

→ Computational intensity compensates for transfer overhead

System Overview

- RISC-V based PULPino microcontroller[1]
  - CV32e40p (RV32IMFC) [Gau+17]
  - `keccak_f1600` single RISC-V instruction



	LUTs	FFs	BRAMs	DSPs	kGE
base	15, 137	9, 943	48	6	143
acc.	22, 356	13, 181	54	13	244
Keccak	4, 782	1, 050	0	0	–
NTT [Fri+21]	2, 475	1, 940	9	7	–
NTT **(This)**	1, 402	1, 192	6	7	–

Table: Area overhead (Xilinx UltraScale+, GF 22nm)

[1] https://github.com/pulp-platform/pulpino

Overview

Results: Cycle Counts

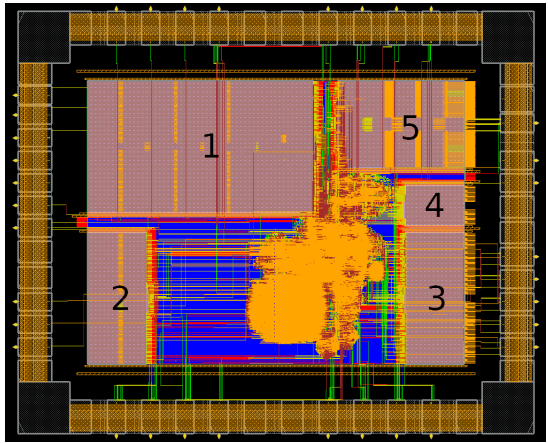
		Keygen	Sign	Verify
Dilithium-II	CVA6 SoC [Nan+21]	1,592,325	5,884,266	1,700,679
	base (this)	3,566,442	11,242,911	3,854,303
	acc. (this)	593,403	1,905,872	651,217
Falcon-512	base (this)	–	–	830,597
	acc. (this)	–	–	314,639

Table: Average cycle count for 100 iterations and a 59 B message.

- [Nan+21]: tightly-coupled NTT acceleration
- Frequency is not affected by acceleration ($\approx$ 150 MHz)
  - Cycle reduction directly transfers to latency

ASIC: Results

- Globalfoundries 22nm
  - $1,25mm \times 1,25mm$
  - 800 MHz (25°C, 0.8V core voltage)
- Size dominated by memories
  - 1, 2, 3: core data/instruction memory
  - 4, 5: NTT memories
- Energy savings up to a factor of $\times 14$ (Dilithium-V)



ASIC: Comparison with ASICs

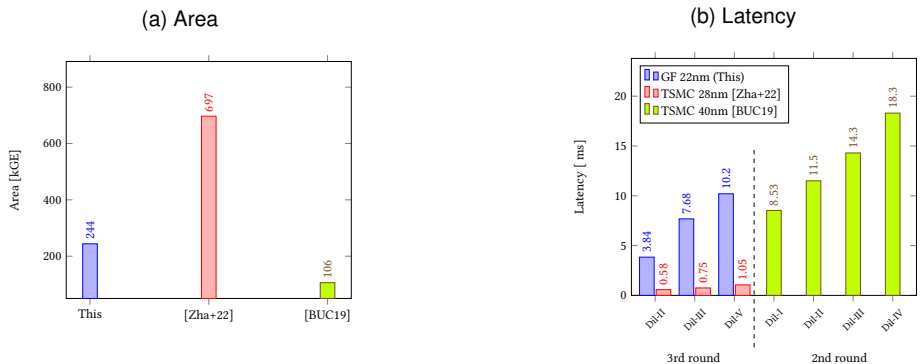


Figure: Comparison with the TSMC 28nm design of [Zha+22] and the TSMC 40nm design of [BUC19]

ASIC: Comparison with ASICs

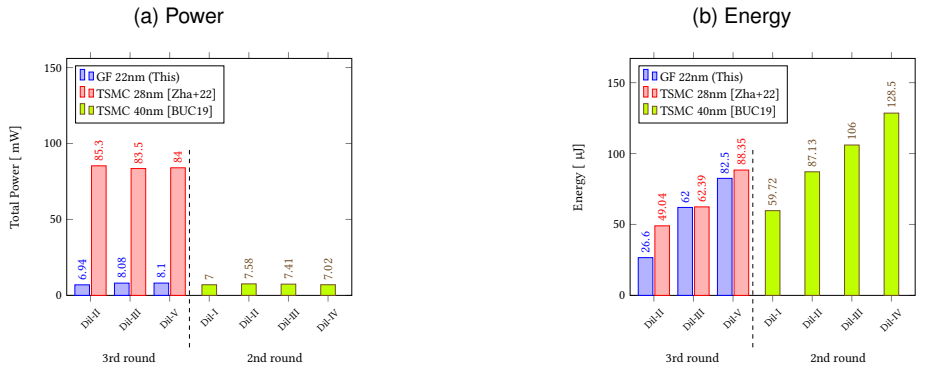


Figure: Comparison with the TSMC 28nm design of [Zha+22] and the TSMC 40nm design of [BUC19]

Comparison with TLS 1.3 evaluation in [Tas+21]

Design	Platform	Dilithium-II	Dilithium-V
[Nan+21]	FPGA at 100 MHz	15.9 / 58.8 / 17.0	50.1 / 133 / 51.3
This	ASIC at 180 MHz	3.30 / 10.6 / 3.62	9.92 / 24.2 / 10.3
This	ASIC at 800 MHz	0.74 / 2.38 / 0.81	2.23 / 5.45 / 2.31
[Zha+22]	ASIC at 540 MHz	0.08 / 0.32 / 0.17	0.18 / 0.58 / 0.30
		2048 bit RSA	secp2561r1 ECDSA
[Tas+21]	Cortex-M4 at 180 MHz	450 / 448 / 12.5	8.43 / 12.3 / 25.2

Table: Latency comparison for Dilithium keygen/sign/verify in ms

- Falcon verification for NIST-1 / NIST-5:
  - 180MHz: 1.72ms / 3.41ms
  - 800MHz: 0.39ms / 0.77ms

Overview

Conclusion

- Unified ASIC design targeting TLS 1.3 in IoT:
  - Dilithium as generic scheme
  - “Free” acceleration of Falcon verification on top
- Performance gain while energy consumption decreases
- Investigation of memory efficient implementations for future work

Thank you for your attention!

E-mail: patrick.karl@tum.de

Paper: https://doi.org/10.1145/3579092

References

[BUC19] U. Banerjee, T. S. Ukyab, and A. P. Chandrakasan. “Sapphire: A Configurable Crypto-Processor for Post-Quantum Lattice-based Protocols”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 17–61.

[Fri+21] T. Fritzmann et al. “Masked Accelerators and Instruction Set Extensions for Post-Quantum Cryptography”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), pp. 414–460.

[FSS20] T. Fritzmann, G. Sigl, and J. Sepúlveda. “RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography”. en. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* Volume 2020 (2020), Issue 4.

[Gau+17] M. Gautschi et al. “Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.10 (2017), pp. 2700–2713.

[Nan+21] P. Nannipieri et al. “A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms”. In: *IEEE Access* 9 (2021), pp. 150798–150808.

[Tas+21] G. Tasopoulos et al. *Performance Evaluation of Post-Quantum TLS 1.3 on Resource-Constrained Embedded Systems*. Cryptology ePrint Archive, Paper 2021/1553. `https://eprint.iacr.org/2021/1553`. 2021.

[Zha+22] Y. Zhao et al. “A High-Performance Domain-Specific Processor With Matrix Extension of RISC-V for Module-LWE Applications”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 69.7 (2022), pp. 2871–2884.