

Hardware Implementation of Falcon Post-Quantum Signature Scheme

Vid Smole

Supervisors: Ratko Pilipović, Sujoy Sinha Roy, Aikata

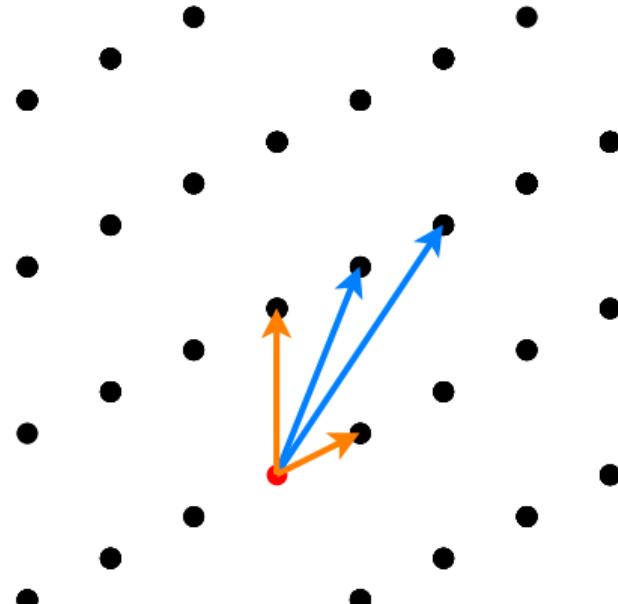
University of Ljubljana, Faculty of Computer and Information Science

Graz University of Technology

November 14, 2025

Introduction to Post-Quantum Cryptography

- ▶ Shor's algorithm breaks RSA, DSA, elliptic curve, etc. on a quantum computer
- ▶ Post-quantum alternatives needed soon
 - ▶ Slow transition
 - ▶ *Harvest now, decrypt later* attacks
- ▶ Post-quantum cryptographic algorithms
 - ▶ Lattice-based (Falcon)
 - ▶ Code-based
 - ▶ Hash-based



Example of a 2D lattice.

Falcon Signature Scheme

- ▶ Standardized by NIST
- ▶ Multiple security levels
 - ▶ Falcon-512
 - ▶ Falcon-1024
- ▶ Design goal: small public key and signature
- ▶ Hard to implement efficiently in hardware
 - ▶ Floating-point numbers
 - ▶ Recursive algorithms
- ▶ Top-level functions
 - ▶ Key-pair generation
 - ▶ Signature generation
 - ▶ Signature verification

Falcon Signature Generation

Goal: Find vectors s_1 and s_2 , such that $s_1 + s_2 h \equiv c \pmod{q}$ holds

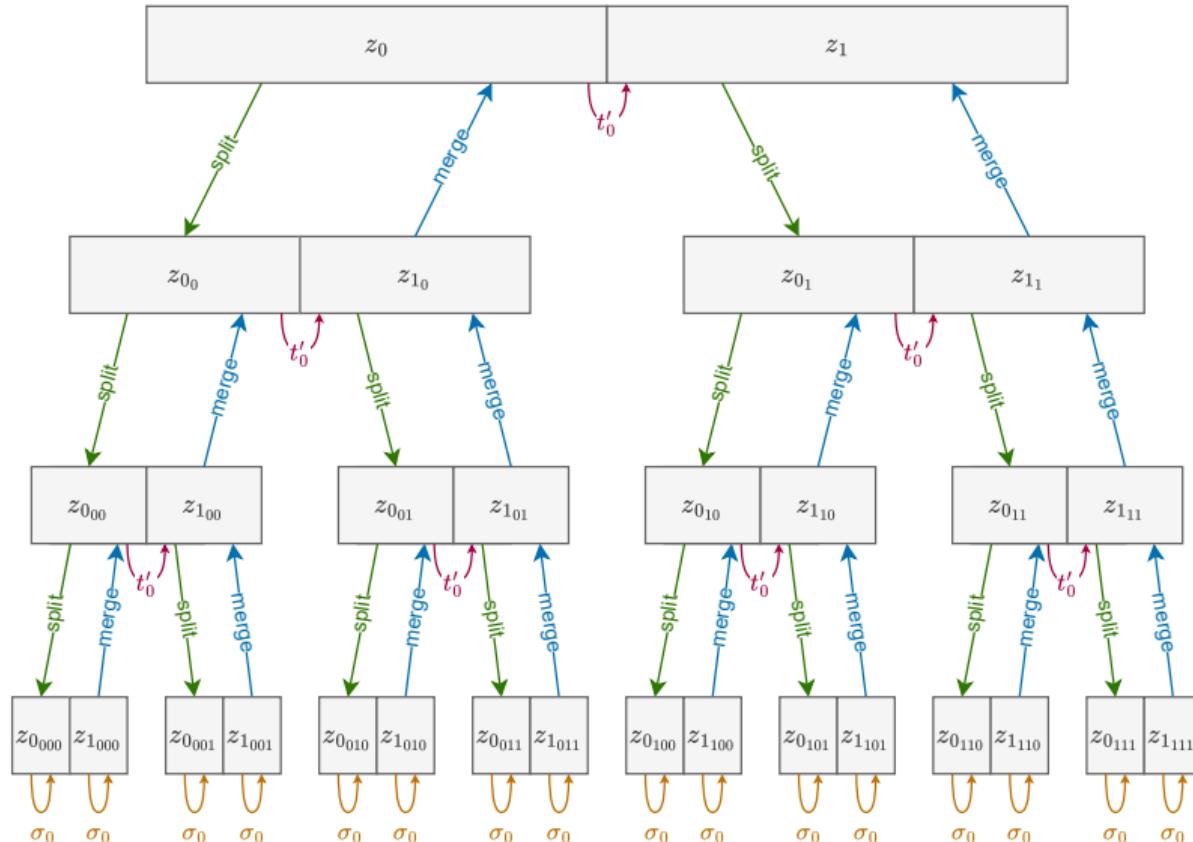
Algorithm $\text{Sign}(m, sk)$

Require: message m , secret key sk

Ensure: compressed signature sig

- 1: $c \leftarrow \text{HashToPoint}(m)$
 - 2: **do**
 - 3: $(s_1, s_2) \leftarrow \text{ffSampling}(c, sk)$
 - 4: **while** $\|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor$
 - 5: **return** $\text{Compress}(s_2)$
-

Falcon Signature Generation - ffSampling



Falcon Signature Verification

Goal: Compute $s_1 = c - s_2 h \bmod q$ and check that $\|(s_1, s_2)\|^2 \leq \lfloor \beta \rfloor^2$ holds

Algorithm Verify(m, sig, h)

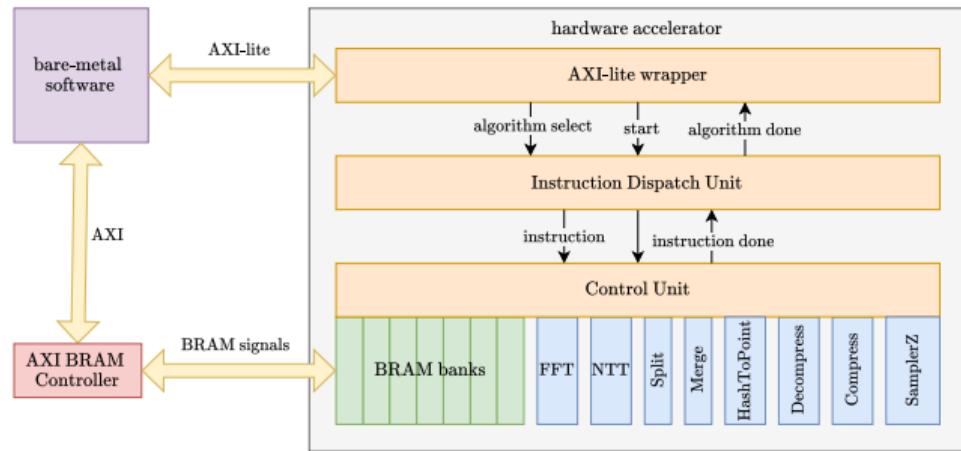
Require: message m , signature sig , public key h

Ensure: accept or reject

- 1: $c \leftarrow \text{HashToPoint}(m)$
 - 2: $s_2 \leftarrow \text{Decompress}(sig)$
 - 3: $s_1 \leftarrow c - s_2 \cdot h \bmod q$
 - 4: **return** $\|(s_1, s_2)\|^2 \leq \lfloor \beta \rfloor^2$
-

Hardware Implementation - Design Overview

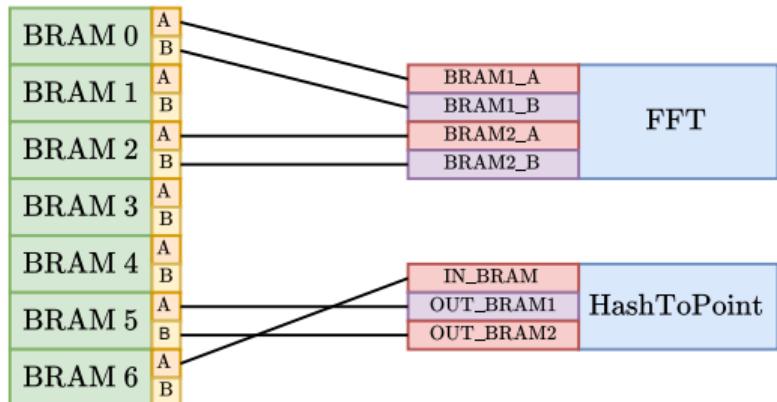
- ▶ Fully in hardware
- ▶ Software wrapper for easier use
- ▶ Memory-centric architecture
- ▶ Instruction-based execution flow



High level overview of proposed architecture.

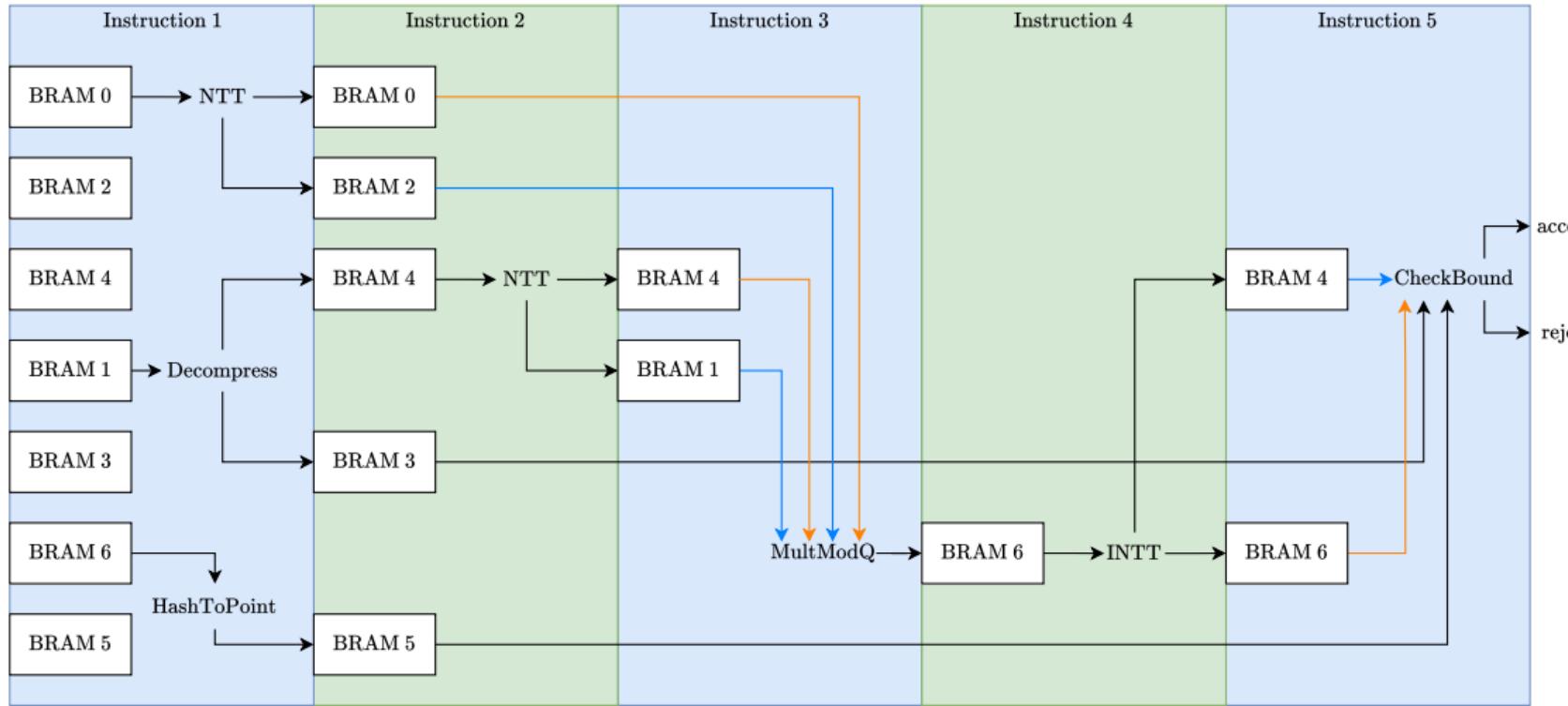
Hardware Implementation - Instructions

- ▶ Instruction width: **70 bits**
- ▶ Instructions encode:
 - ▶ Which modules to run
 - ▶ Module configuration
 - ▶ BRAM bank connections
 - ▶ Memory addresses
- ▶ Total instruction count:
 - ▶ Verify: **5**
 - ▶ Sign: **4615 / 9223**



Example connections between modules and BRAM banks.

Hardware Implementation - Scheduling



Results

Toolchain and Platform:

- ▶ Vivado 2023.1
- ▶ Zynq UltraScale+ (PYNQ-ZU)
- ▶ Maximum frequency: **125 MHz**

FPGA Resource usage

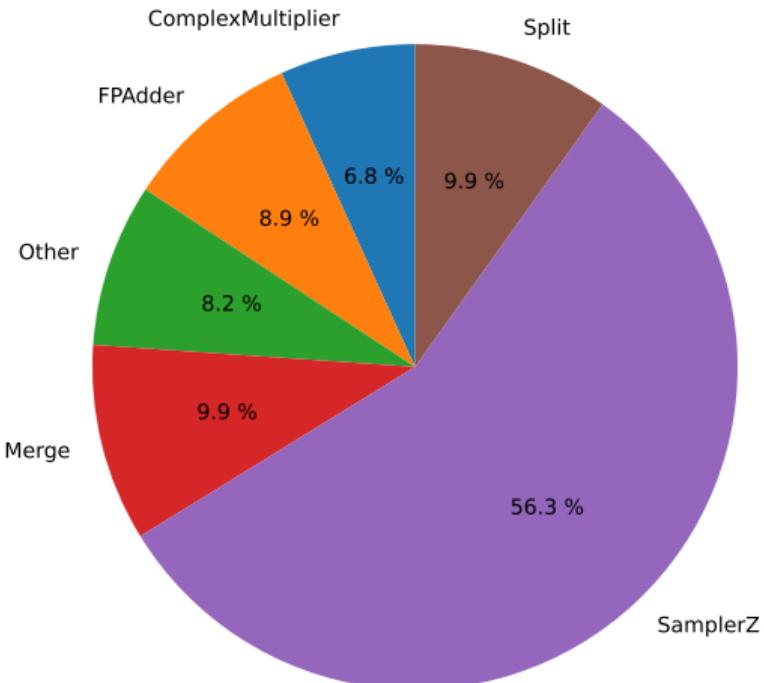
- ▶ LUTs: **74k**
- ▶ Flip-Flops: **38k**
- ▶ DSPs: **175**
- ▶ BRAMs: **69.5 / 84**

Performance (Clock cycles)

- ▶ Signing: **190k / 383k**
- ▶ Verification: **7.8k / 16.6k**

Future Work

- ▶ Key-pair generation
- ▶ On-the-fly instruction generation for ffSampling
- ▶ Better implementation of SamplerZ
- ▶ Streaming support for input messages



Module running time for signature generation.

Conclusion

- ▶ First hardware implementation supporting both Falcon signing and verification
- ▶ Performance and resource usage similar to prior implementations of only one operation
- ▶ Source code publicly available