Making the Case for a Keccak Instruction
Post-Quantum Cryptography on RVV

Markku-Juhani O. Saarinen, Tampere University
PQC Task Group Chair
`<markku-juhani.saarinen@tuni.fi>`

RISC-V Summit North America
Santa Clara -- October 23, 2024

Cryptography Extensions ("K")

Done: **Scalar Crypto** (Ratified 2021)**:** AES, SHA2, SM3, SM4, CMUL (GCM) with 32- and 64-bit **scalar registers**. + "Constant time" & Entropy Source.

Done: **Vector Crypto** (Ratified 2023)**:** AES, SHA2, SM3, SM4, GCM with **vector registers**: Make bulk crypto even faster with *parallel* AES-GCM etc.

-> many of these now In Linux Kernel, OpenSSL, going into Android Platform

Being worked on:

High Assurance Crypto TG (From late 2023)**:** "Full-rounds" AES allowing emission/power side-channel security. Key management features.

Post-Quantum Crypto TG (From late 2023)**:** What can we do to assist standard PQC algs (notably FIPS 203,204,205 - Kyber, Dilithium, SPHINCS+) ?

You all have heard about this --

Most online stuff is protected with TLS:

- Asymmetric key exchange for session keys.

- Authentication with certificates / signatures.

These now use cryptography breakable in polynomial time with a quantum computer.

Post-Quantum: Ongoing transition to use newer algorithms designed to resist quantum attacks.

August 13, 2024: Standards Came into Effect

Kyber (FIPS 203, ML-KEM) **--** *Key Establishment*
Replace or augment RSA, ECDH (e.g. X25519) for message encryption, key exchange (TLS).

Dilithium (FIPS 204, ML-DSA) **--** *Digital Signature*
Replace or augment RSA, ECDSA for Data or End-point Authentication, PKI Certificates.

https://csrc.nist.gov/news/2024/postquantum-cryptography-fips-approved

RWPQC / Toronto -- March 24, 2024. Picture from Bas Westerbaan.

How bad an extra hash can be?

By Sasha Frolov and Rafael Misoczki

- Key exchange is a (very) commonly performed operation at Meta
  - **Currently, ~0.05% of CPU cycles in Meta's data centers are spent doing X25519 key exchange**
  - We hope this data point is useful for making cost estimates while defining PQC standards specs
- This means
  - Deploying post-quantum key exchange has a non-negligible capacity cost
  - Apparently innocuous steps can cost hundreds of thousands or even millions of dollars a year
    - e.g. extra hashing steps, like hashing randomness or hashing parts of the transcript, which are being discussed as part of finalizing Kyber specification
    - Even if an extra step does not affect latency, the extra power usage/consumption of shared resources on highly parallel servers still has costs

Feedback? Write to sashafrolov@meta.com or rafam@meta.com.

Compute Impact (focusing on TLS with RVV)

- **Boot process** can use Dilithium, but **hash-based signatures** XMSS/LMS (SP 800-208) or SPHINCS+ (FIPS 205) are also often recommended. These are pretty slow algorithms, especially for signing. Not in TLS.

- **TLS** (or **QUIC**, **IPSEC**, **SSH**) key exchange latency affects user experience and overall power profile. Both **Kyber** and **Dilithium** will be used here.

- **Good news**: These lattice-based PQC algorithms are usually faster or roughly same speed as classical crypto. *But any speedup is welcome.*

Kyber & Dilithium on RVV

Optimized Kyber and Dilithium with RISC-V Vector Intrinsics / CLANG 20.

Benchmarked with SpacemiT X60 (VLEN=256), C908 (VLEN=128) silicon.

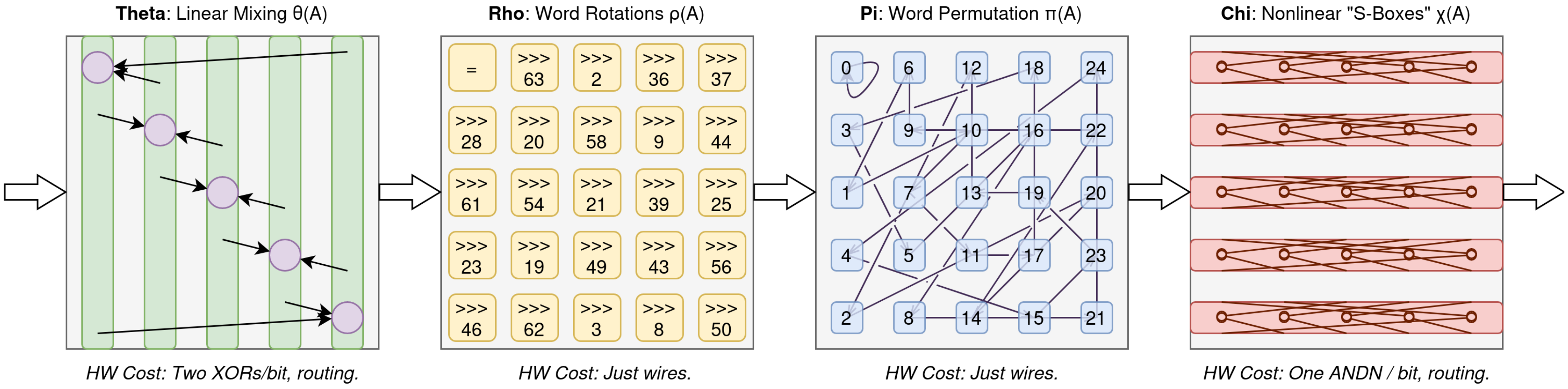
- Vector really helps (~5x speedup) with arithmetic parts (NTT) and somewhat with the bit packing and sampling too.
- Vector does **not** help SHA3/SHAKE much -- that becomes a bottleneck.

Another implementation:

Jipeng Zhang et al, *"Optimized Software Implementation of Keccak, Kyber, and Dilithium on RV{32,64}IM{B}{V}."* TCHES 2025/01.

Keccak f1600: Core of SHA3/SHAKE

- **SHA3** and **SHAKE** (FIPS 202) are "modes" of the 25×64=1600-bit **Keccak** permutation. 60-80% of Kyber, Dilithium cycles spent here.
- 24 Rounds. The rounds have an incredibly short critical path in hardware (fast hw!), but **vectorization is disappointing** (<2× scalar?)



Kyber-768 (ML-KEM) Key Exchange

	KeyGen()	Encaps()	Decaps()	TOTAL	Speedup
RV64GC	663,067	815,357	1,006,469	2,484,893	**1.00**
RV64GCV+ZBB	546,631	685,201	858,508	2,090,340	**1.19**
w. Intrinsics	223,400	239,714	262,241	725,355	**3.43**
w. Keccak Insn	49,363	61,632	84,120	195,115	**12.74**

Clang 20.0.0git -O3 with –march=rv64gc / rv64gcv_zbb_zvl256b

Lines 1-3 uses C language reference Keccak f1600; 4,038 instructions.

Line 4 uses SPIKE 1 cycle Keccak. In real-life in hardware ~100 cycles.

Keccak Instruction: Main PQC TG Proposal

Keccak state is awkward to fit into vector registers and architecture:

- Seemingly VLEN ≥ 256 is required (the max LMUL value is 8.)
- Element EEW = 64. Element group EGS = 32, LMUL = 2048 / VLEN:
  - VLEN = 256: LMUL = 8: A group of 8 vector registers of 256 bits.
  - VLEN = 512: LMUL = 4: A group of 4 vector registers of 512 bits.

Multi-round instruction (due to complexity of accessing 25 words):

```
vkeccak.vi vd, vs2, imm    # imm = 5-bit num rounds
```

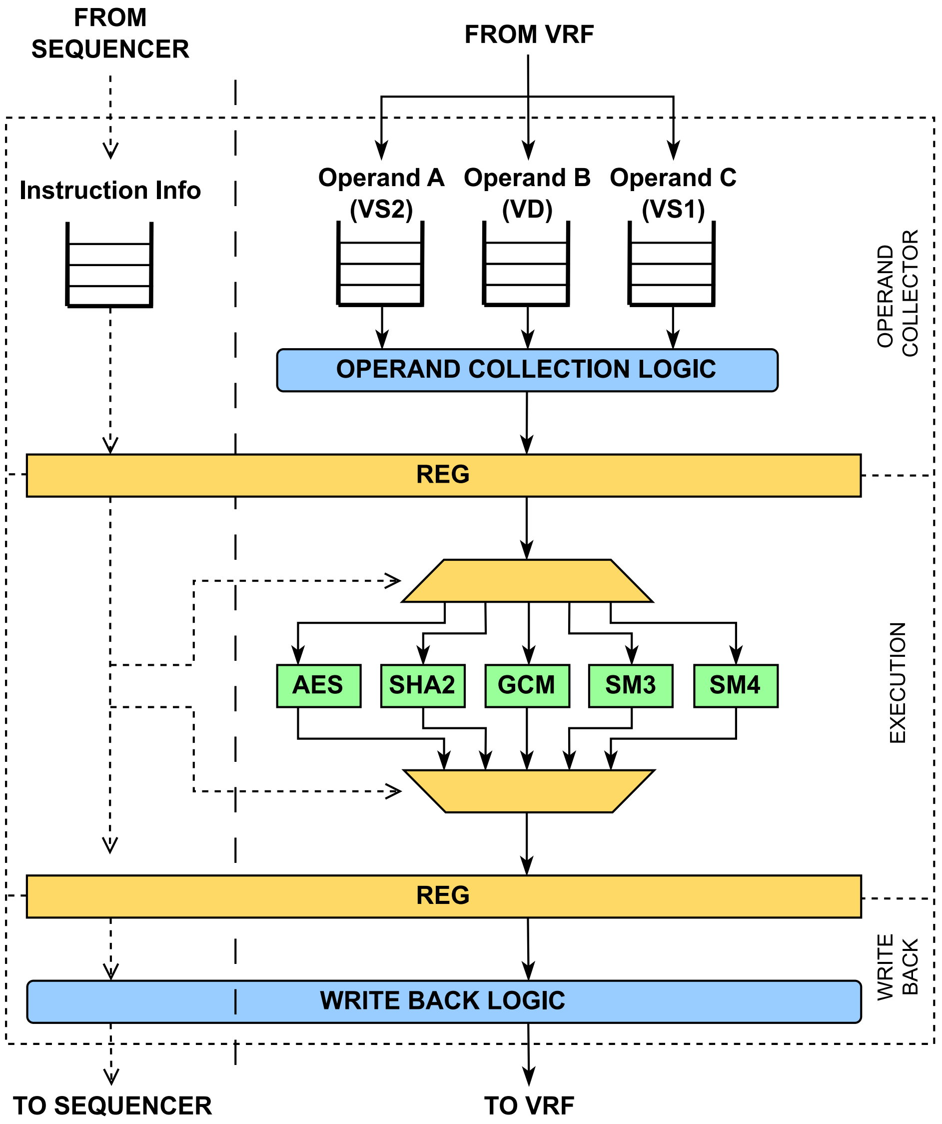
Computes 24 rounds of Keccak-p[1600,24] permutation with imm=24.

Microarchitecture Notes

In **"Marian"** we extended the PULP Ara2 vector unit with Zvk crypto instructions.

We used a 256-bit "operand collector" because of the VRF structure. VRF is split into lanes. Each lane only has 64-bit segments of each reg; need to combine.

Keccak would need a 1600-bit collector.
But would probably still be worthwhile!



Consider a combined Shuffle / Keccak unit?

- Vector 1.0 implementors already have to consider instructions such as "vrgather" that will permute bytes across LMUL*VLEN bits.
- **Example:** VLEN=256, LMUL=8 vrgather permutes 2048 bits / 256 bytes.
- Post-Quantum Crypto uses a lot of NTT; need fast butterflies (like FFT.)
- Available silicon (X60 core, VLEN=256) requires $4*(LMUL)^2$ cycles:

```
__riscv_vrgather_vv_u8m1(): 4 cycles   (LMUL=1)
__riscv_vrgather_vv_u8m2(): 16 cycles  (LMUL=2)
__riscv_vrgather_vv_u8m4(): 64 cycles  (LMUL=4)
__riscv_vrgather_vv_u8m8(): 256 cycles (LMUL=8)
```

- **Keccak could be in a fast shuffle/slide unit with large holding registers.**

SPHINCS$^+$: Impact on FIPS 205 SLH-DSA

FIPS 205 SLH-DSA *"Stateless Hash-Based Digital Signature Standard"* (a.k.a. SPHINCS$^+$) has two parameter instantiations, SHA2 and SHAKE.

SLH-DSA-SHAKE is made at least 10 times faster by **vkeccak.vi**.

Note that holding the Keccak state in vector registers allows "padding template" forming and Winternitz iteration (https://ia.cr/2024/367).

Similar speedup for SHAKE variants of LMS & XMSS in SP 800-208.

Conclusions: PQC TG Recommendation

Keccak instruction seems like a winner, giving significant speedups for all PQC algorithms. NTT can also be considered, but RVV already helps a lot with that.

This 1 instruction is replacing thousands of instructions. Core f1600 permutation is 24 cycles. Together with operand collection + writeback can still be under 100.

Hardware note: Permutation alone is about 40kGE + "operand collector" logic.

PQC speedup of Keccak Insn. on RVV ~2-3x. Quite easy to integrate into software.

Microarchitecturally awkward but saves device battery / $$$ in data centre.