RISC-V SUMMIT
NORTH AMERICA 2025

<- was 2025-10-23

This version 70% expanded.

PQCP Support for RISC-V Vector, Future Keccak ISE

Markku-Juhani O. Saarinen

Cryptography SIG Chair

2025-11-21: NTNU, Trondheim

Hello! 👋 I'm **Markku-Juhani Saarinen**.

- Cryptographer for soon 30 years (SSH Communications Security in 1997.)
- PhD RHUL (2009) about hashes. Started drifting into PQC around 2015.
- Academia and industry, most recently at PQShield from 2018 to 2024.
- **In the industry**: Pentests, then HW architecture and side-channel stuff.
- **Now:** Professor of Practice at Tampere University in Finland 🇫🇮.
- **RISC-V:** Cryptography SIG Chair, PQC TG Chair at RISC-V International.

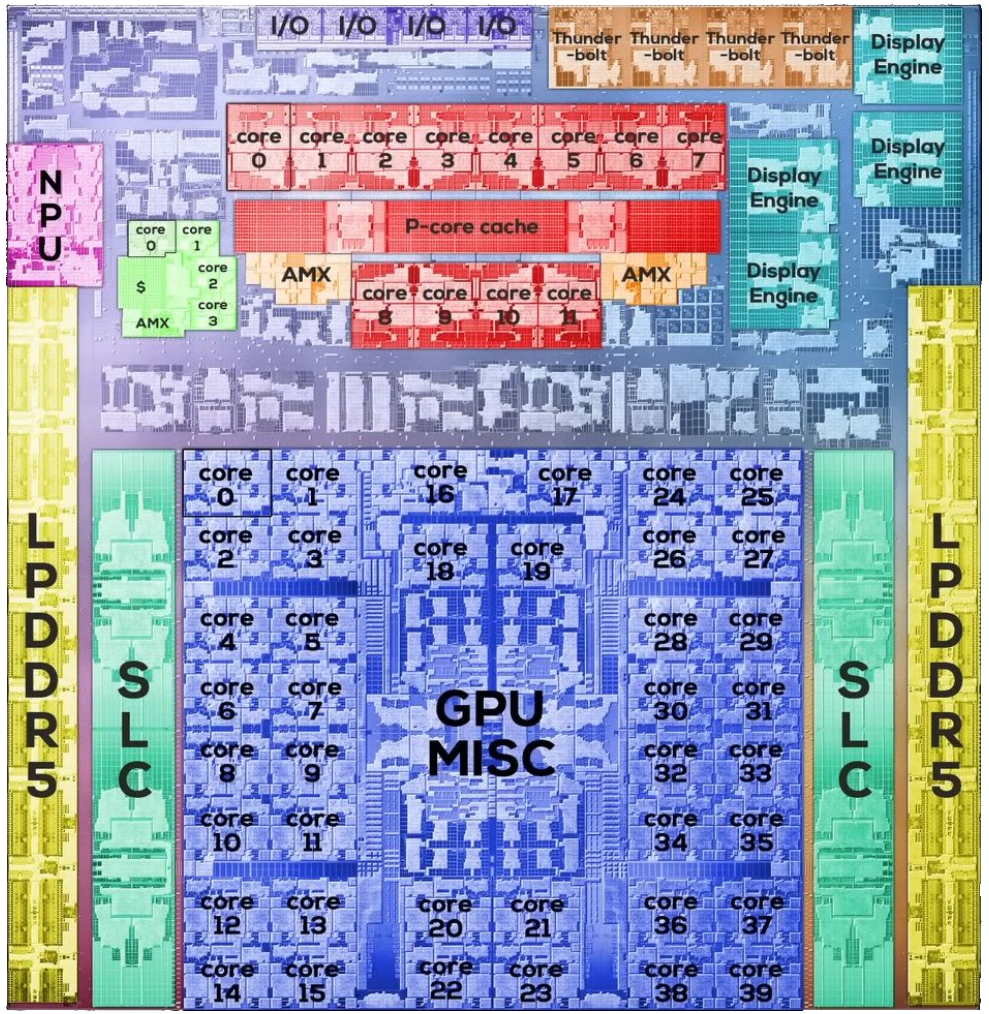
Focus: **Application Processor** Crypto Support

These cores run the “visible” OS:

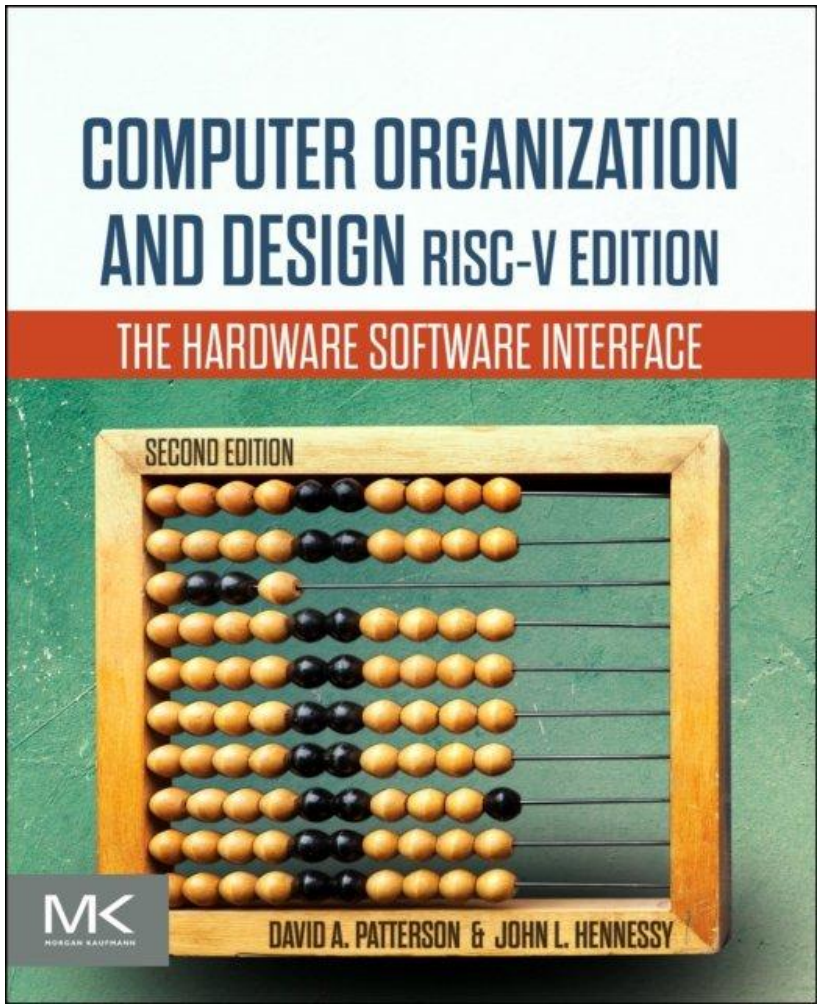
- In Kernel: IPSec, WireShark
- Sometimes also storage encryption
- Standard apps and libraries: TLS, QUIC
- OS tools, services: SSH, GnuPG
- User applications: Signal, WhatsApp, ..

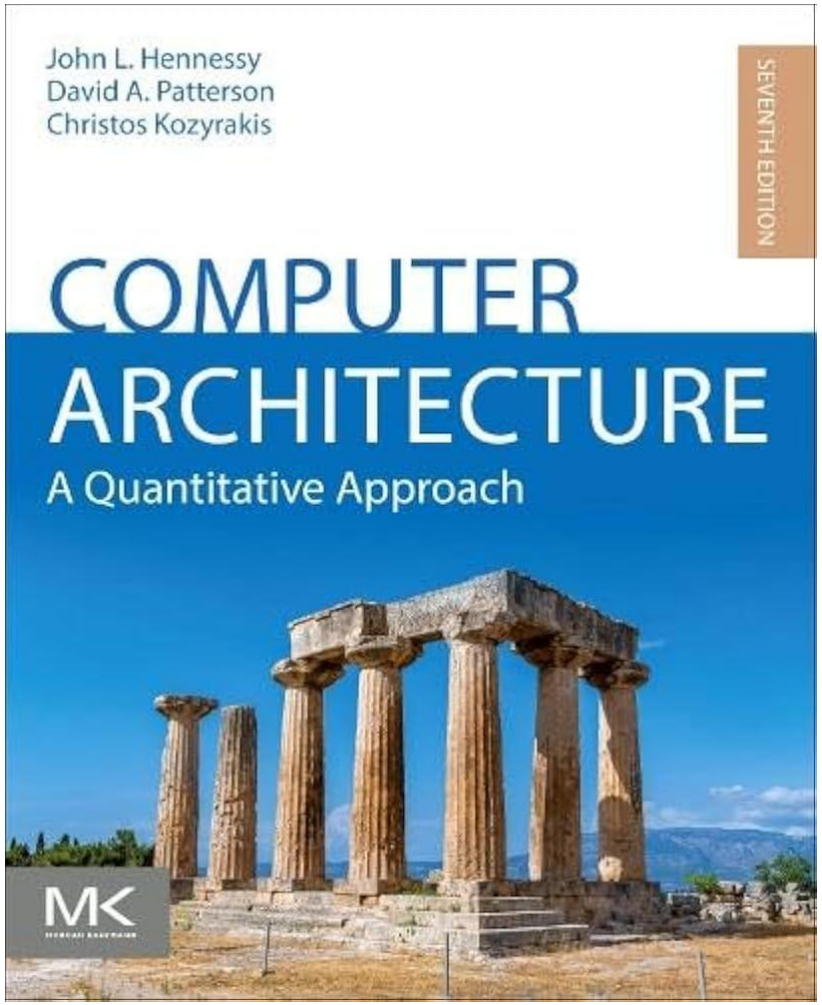
Crypto acceleration mostly done with **instruction set features** (for scalability).

Typically timing attack protection **only**.

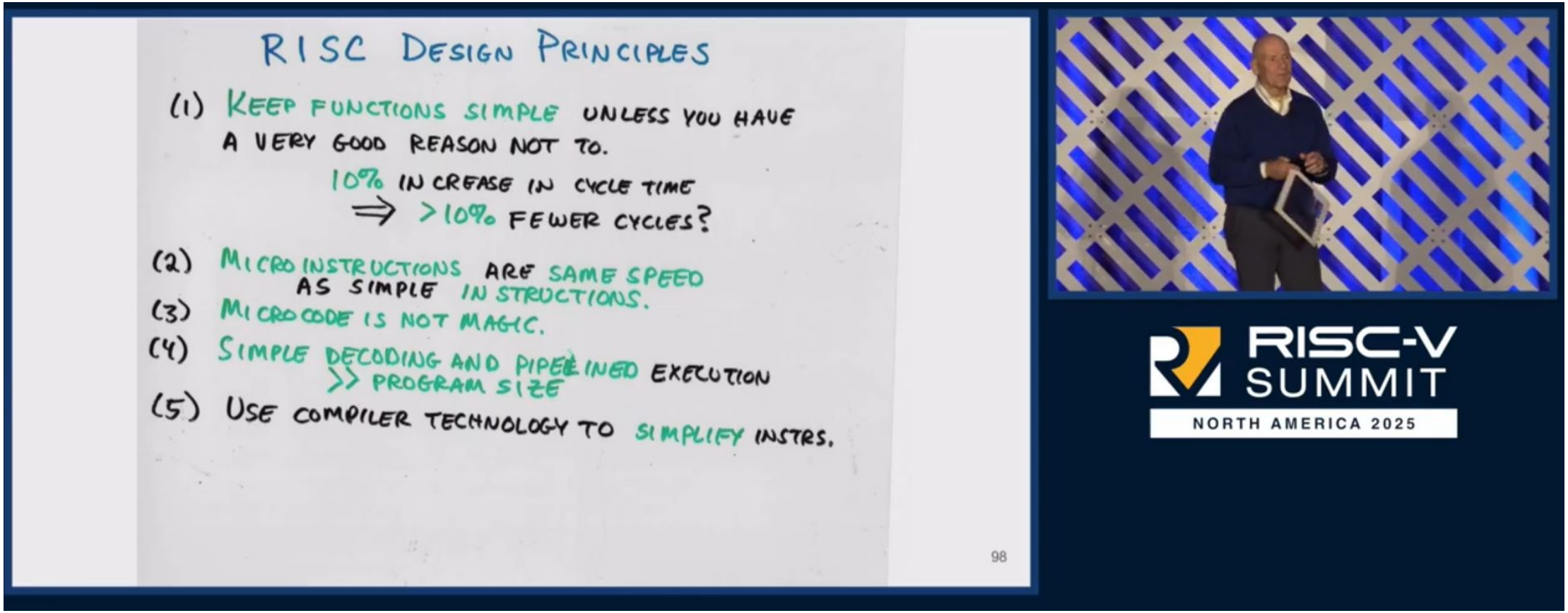


I read **“Hennessy & Patterson”** in 1995.. 👇





Patterson showing **1981** slides in **2025** .. 🙃



Instruction Set Architectures: Everything Old is New Again

RISC-V is a lot like SPARC, MIPS, PowerPC, PA-RISC, DEC Alpha.. A bit less like ARM, and much, *much* less like x86.

RISC-V Vector (v-extension) is a "real" vector architecture a la Cray-I (1976), not fixed-length SIMD (AVX2, Neon).

In RISC-V, almost everything is an extension (other major ISAs continuously expand too.) Two kinds of extensions:

- **Custom**: DIY instructions at custom opcodes.
- **Ratified**: Consistent specs, reserved opcodes, simulator, compiler, operating system, support, ...

RISC-V® Org (RISC-V International)

- Oversees ISA Development, Certification. 4500 Members in 70 Countries.
- Big open-source/technical standardization org: Committees galore.
- Now a ISO/IEC JTC1 PAS (*"Publicly Available Specification"*) Submitter.
- RISC-V has a Cryptography SIG. "Task Groups" create ISA specifications and are also under the Unprivileged ISA Committee and the Security "HC."

Some basic rules for new RISC-V instructions:

- Instructions need to demonstrate substantial, measurable advantages.
- Instructions need to fit into the big-picture RISC-V architecture.
- You need to *explicitly contribute* the instruction to the RISC-V ISA (membership.) This is to protect everyone against IP / patent problems.

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.

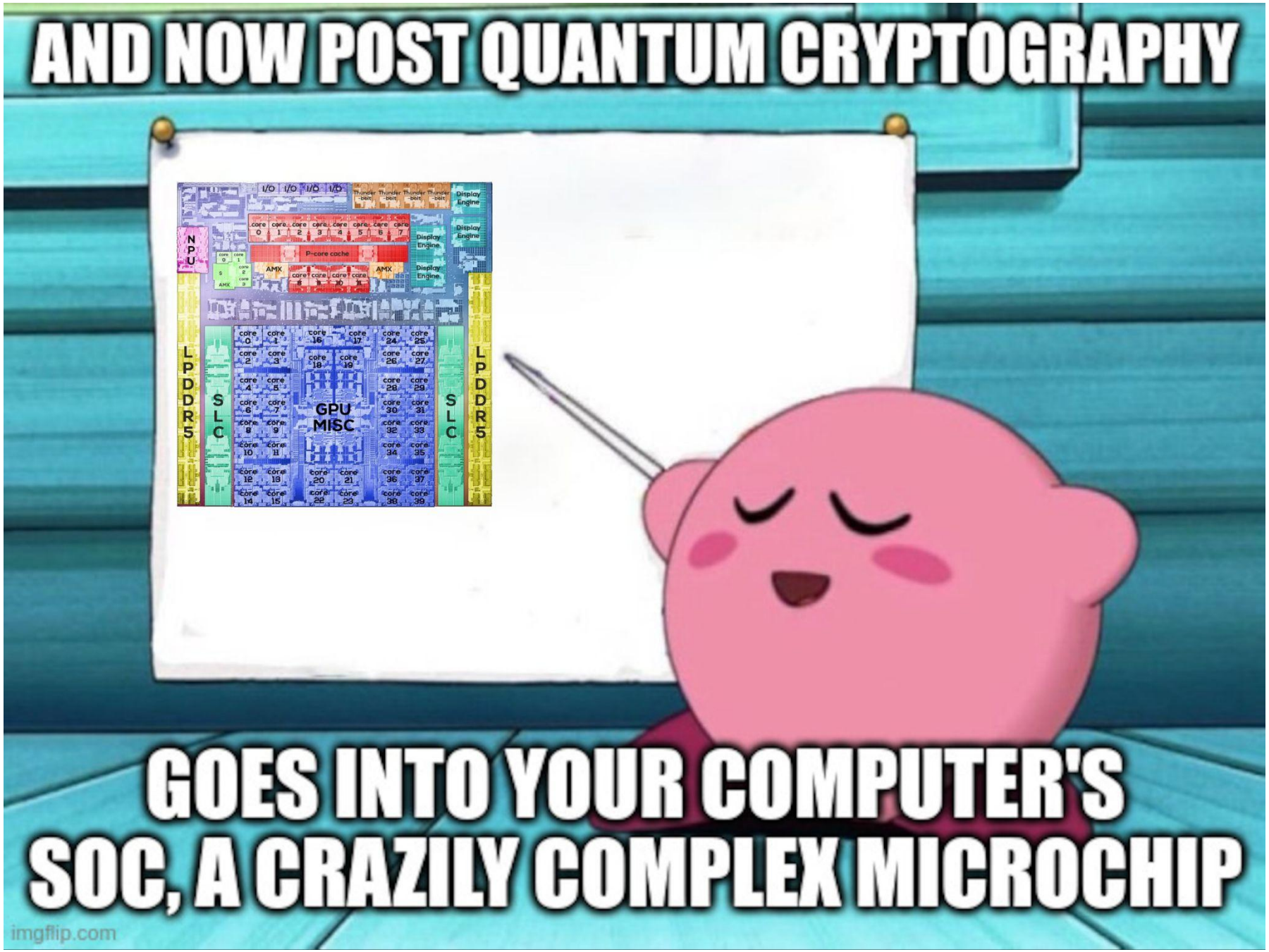
Done: Vector Crypto (Ratified 2023)**:** AES, SHA2, SM3, SM4, GCM with **vector registers**: Make bulk crypto even faster with *parallel* AES-GCM etc.

-> 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+) ?


AND NOW POST QUANTUM CRYPTOGRAPHY
I/O
Thunder
Display Engine
NPU
P-core cache
AMX
LPDDR5
SLC
GPU MISC
GOES INTO YOUR COMPUTER'S SOC, A CRAZILY COMPLEX MICROCHIP
imgflip.com

Reminder: **Crypto on a System-on-Chip (SoC)**

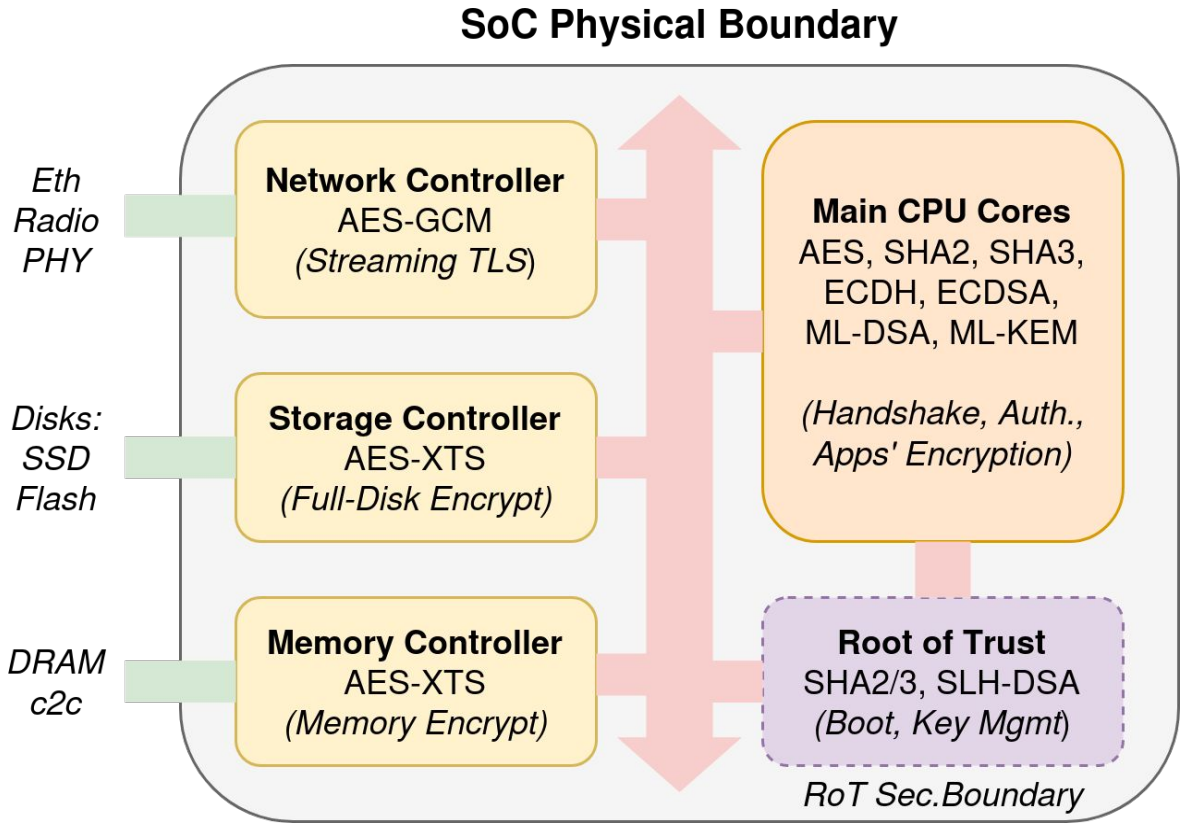
We do not use (or want to use) the main CPU for all cryptography!

Root of Trust (RoT): SoC-wide Platform Security. Isolated MCU + accelerators.

Disk or storage controller: E.g. AES-XTS.

Network Controller: Bulk symmetric encryption (e.g. TLS AES-GCM Frames).

Main CPU complex: TLS handshakes; asymmetric operations, X.509 parsin, cryptography in end-user applications.



Helicopter View: PQC for CPU and RoT

- **Boot process** can use Dilithium, but **hash-based signatures** XMSS/LMS (SP 800-208) or SPHINCS+ (FIPS 205) are also often recommended.

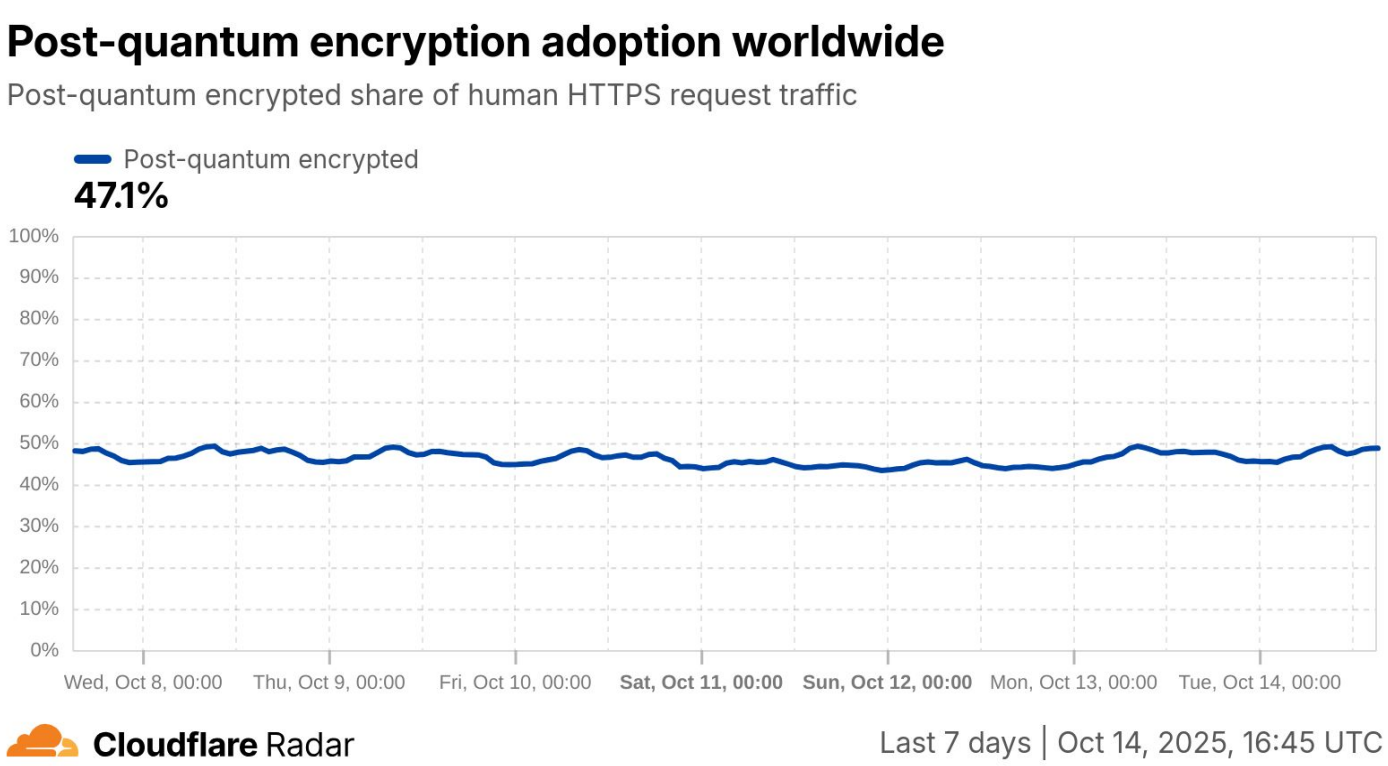
  Boot verification processing is often performed by the **Root of Trust** unit.

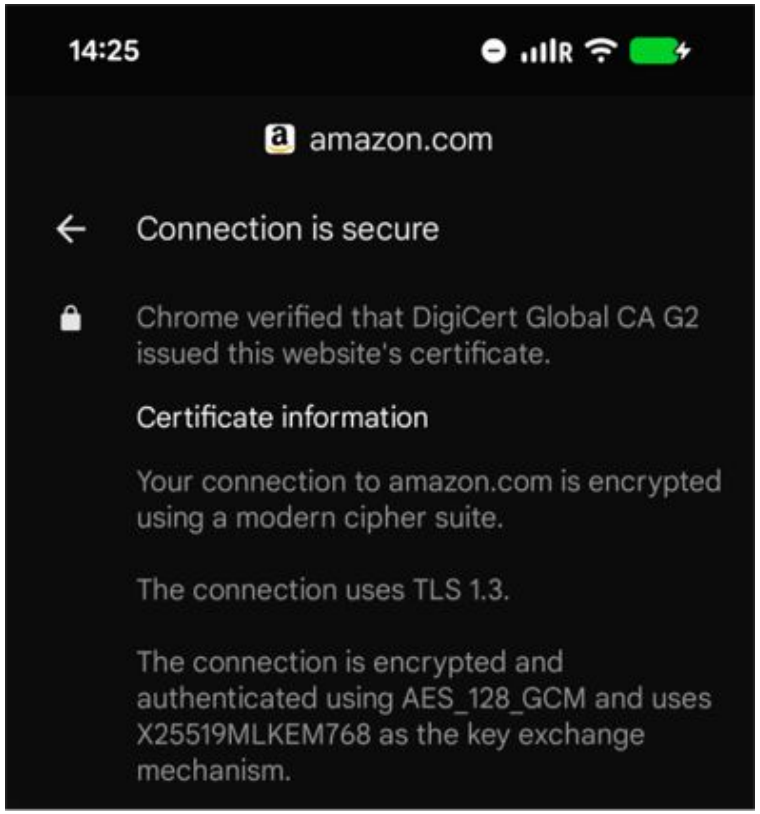
- **TLS** (or **QUIC**, **IPSEC**, **SSH**) key exchange latency affects user experience and overall power profile. Both **Kyber** and **Dilithium** will be used here (KEX+Auth.)

  This processing is usually done by the **Application Processor** units.

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

Reality Nov 2025: ~ 50% of non-bot HTTPS is PQC





- **Google** transitioned the Chrome browser and their services last year.
- **Cloudflare** (fronts a lot of big sites), **Apple**, **Amazon** have also transitioned.
- This is almost entirely **X25519MLKEM768** hybrid TLS 1.3 cipher suite.

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.

RVI PQC TG: Post-Quantum Cryptography

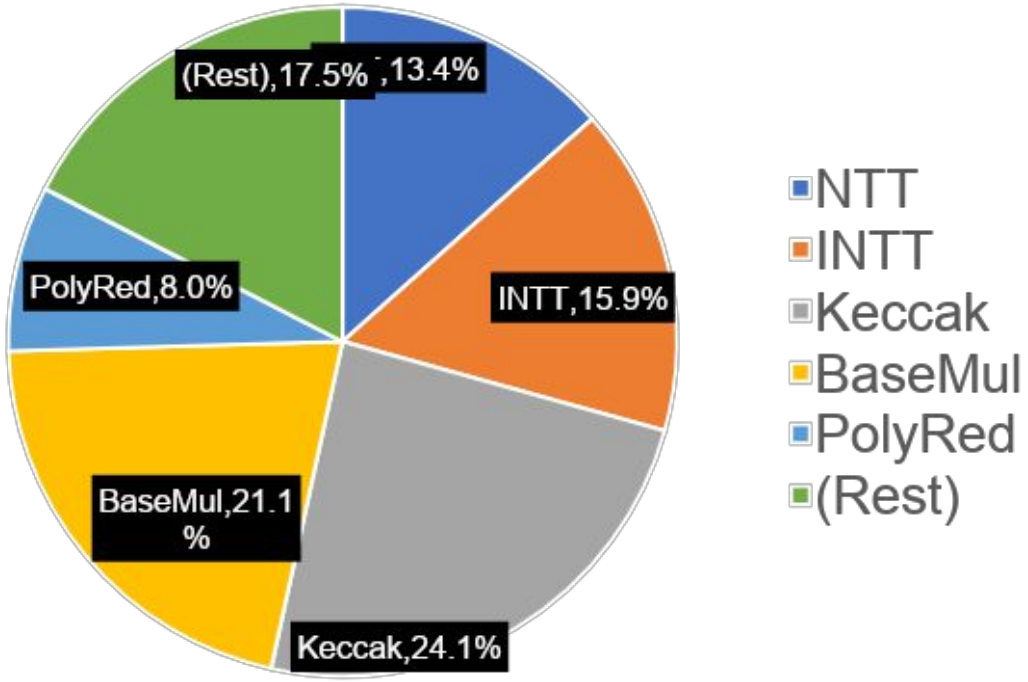
- **Kyber** (FIPS 202 ML-KEM) and **Dilithium** (FIPS 203 ML-DSA) are just as fast -- *or faster* -- than { RSA, ECDSA, ECDH } on current RVV CPUs.
- Due to flexibility required (e.g. hybrid crypto) and external interface complexities, asymmetric crypto is likely to remain in main CPUs.
- PQC TG evaluates **helper instructions**. To be considered, substantial perf advantages must be demonstrated, without too much cost.

Evaluation metric:

- What matters: **End-to-end latency** (µs/op) - when instantiated in a typical application (most often a TLS stack; server or client.)

Reference (non-vector) Kyber Histogram

Reference Kyber-768: 2.26M Insn
KG 600k + Enc 734k + Dec 921k



Compute in NIST Lattice Crypto:

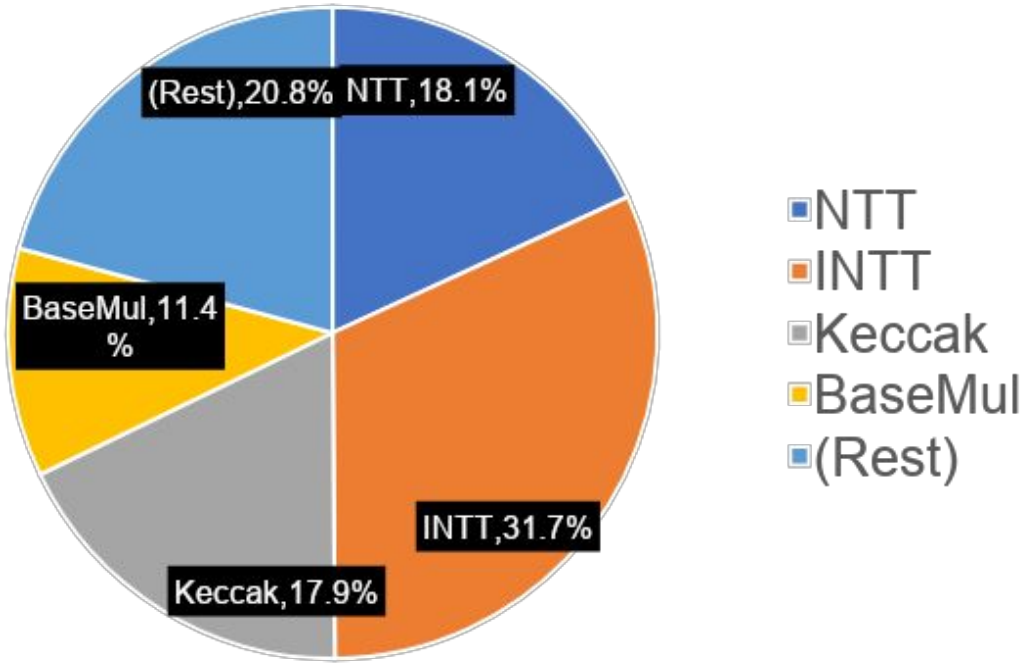
- **Keccak** i.e. SHA3/SHAKE operations. Can be even >50% of overall cycles.
- **Number Theoretic Transforms.** Vectorizable functions (256 x 16/32.)
- **Other polynomial arithmetic.** Mostly integer vectors; shifts, adds, sub.
- **Samplers** (rejection and CBD), rounding, "packing" (serialize).

Instret (with vlen:128,elen:64) - LLVM 18 snapshot, Oct 2023. `-Ofast -march=rv64gcv_zbb (zvk)`

Dilithium: Vectors (mod 8380417) + Keccak

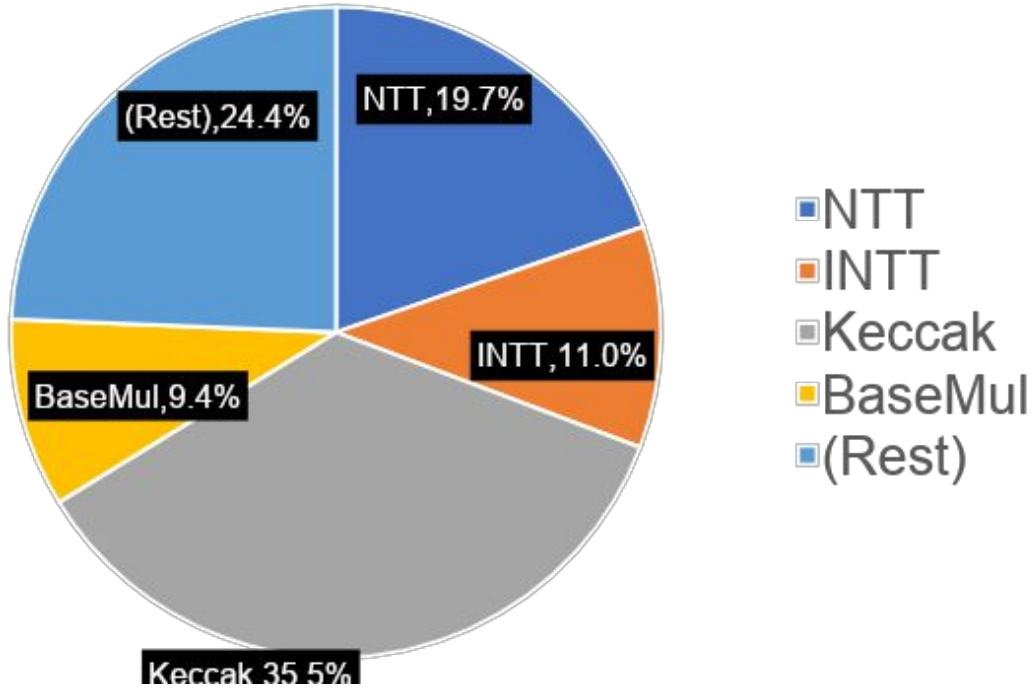
ML-DSA-44 Sign: Avg 4.60M Insn

ML-DSA-87 Sign: Avg 8.37M Insn



ML-DSA-44 Verify: 1.16M Insn

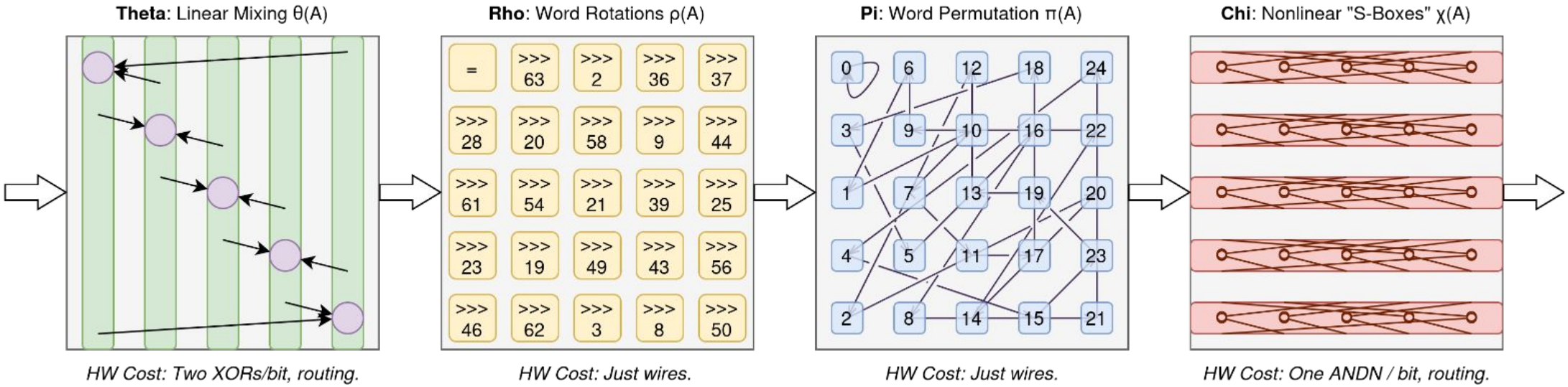
ML-DSA-87 Verify: 3.09M Insn



Instret (with vlen:128,elen:64) - LLVM 18 snapshot, Oct 2023. `-Ofast -march=rv64gcv_zbb (zvk)`

PQC Task Group: Keccak instruction proposal

- Implementations can spend >50% of cycles on the Keccak f1600 permutation used by SHA3 / SHAKE, which vectorizes poorly.
- The main PQC TG proposal remains a Keccak instruction: Doubles not only ML-KEM but also ML-DSA (signing) speed.



RISC-V Vector Register File Organization

Hardware constraints:

- **RISC-V Vector Extension** defines has 32 vector registers (V0 .. V31).
- Each vector register is VLEN ∈ { 128, 256, 512.., *up to 65536* } bits.

Dynamically configured (vsetvli/vsetivli/vsetvl):

- Register are be divided to elements of SEW ∈ { 8, 16, 32, 64 } bits.
- Hence each register is viewed as having VLEN / SEW elements.
- LMUL ∈ { 1, 2, 4, 8 } registers can be grouped into a register group.
- Single instruction can operate on VL ≤ LMUL * VLEN / SEW elements..

1600-bit Keccak state vs Vector Register File

Instruction: **`vkeccak.vi vd, vs2, uimm`**

Destination register group, source register group, immediate = nr. rounds

VLEN=512, LMUL=4:	v0 *..v3*	v4 *..v7*	v8 *..v11*	v12 *..v15*
	v16 *..v19*	v20 *..v23*	v24 *..v27*	v28 *..v31*
VLEN=256, LMUL=8:	v0 *..v6*	v8 *..v14*	v16 *..v22*	v24 *..v30*
VLEN=128? Perhaps Ignore LMUL!			v0 *..v12*	v16 *..v28*

Expected Cycle counts..

This is not what typical vector processor register files are organized for !

Simplistic “add-on” Keccak instruction implementation (~40k GE ?):

1. Get 1600 bits from a vector register group (3 cycles ?)
2. Run the Keccak permutation – 24 cycles.
3. Put the 1600 bits back into a vector register group (3 cycles ?)

With luck the whole operation is perhaps 30 cycles?

PQC TG: Keccak Instruction Conclusion

Architecturally unusual:

- Performs the 1600-bit Keccak permutation with 1 instruction.
- Needs to read/write a register group containing 25 × 64-bit words.
- But provides incredible speedup, from ~2000 cycles down to ~30.
- Much faster than "partial" SHA3 instructions on ARM ISA.

Current status:

- Has spike, benchmarks (for a while now). But no HW PoC yet.
- Spec + Internal review scheduled before the end of the year.

PQCP: Post-Quantum Code Package

(END OF TALK)

The third part – on the PQCP, referred to in the talk title – was not actually covered in the lecture due to time constraints.

RISC-V SUMMIT

NORTH AMERICA 2025