N E X U S

Nexus 1.0: A General-Purpose IVC Machine

nexus.xyz

Agenda

Part 1: (High Level) Intro to the Nexus 1.0 zkVM

Part 2: (Technical) The Nexus IVC Prover 1.0

zkVM

Can we reach 1 *trillion* CPU cycles proved / s?

Nexus 1.0 Whitepaper

nexus.xyz/whitepaper.pdf

Nexus 1.0: Enabling Verifiable Computation

Daniel Marin *,

Michel Abdalla, Paul Govereau, Jens Groth, Samuel Judson, Kristian Sosnin, Guru Vamsi Policharla, and Yinuo Zhang †

Nexus Labs
{daniel,michel,paul,jens,sam,kristian,vamsi,yinuo}@nexus.xyz

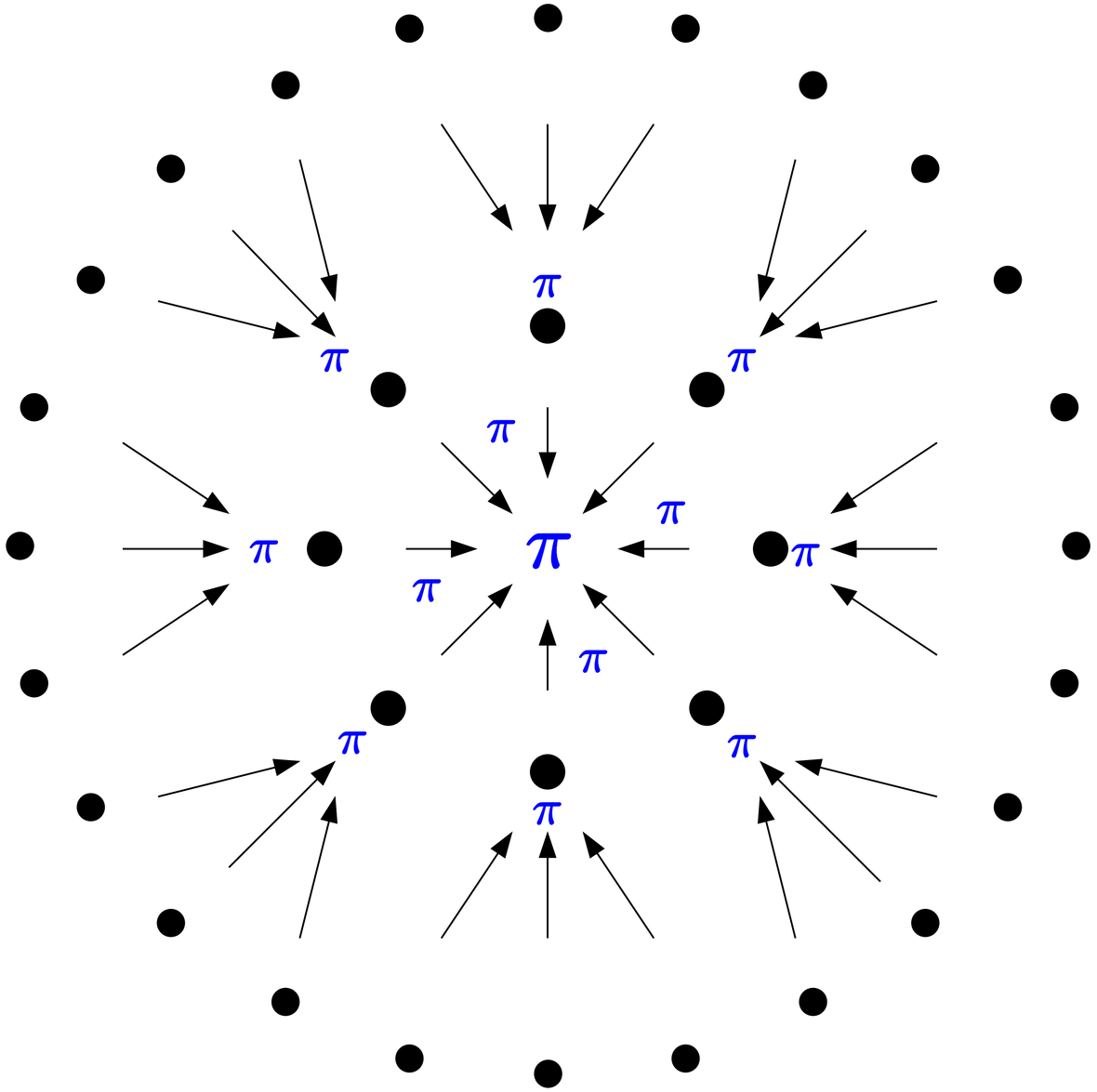
January, 2024

Abstract

We introduce the Nexus project, an endeavor to enable verifiable computation, at Internet scale. The world has come a long way since Turing introduced in 1936 a *universal computing machine*, a hypothetical machine capable of executing any computation. This concept is considered the origin of the general-purpose computer, and was used by von Neumann to introduce the von Neumann architecture, a physical instantiation of the Universal Turing Machine. This architecture now powers virtually all modern computers.

In this paper, we introduce the Nexus zkVM (zero-knowledge virtual machine), a machine capable of *proving* any computation. That is, the machine produces succinct zero-knowledge proofs of correct program execution for any stateful machine (e.g. RISC-V, EVM, Wasm) with any particular instruction

Can we unite the world's computers into a single world-scale zkVM?

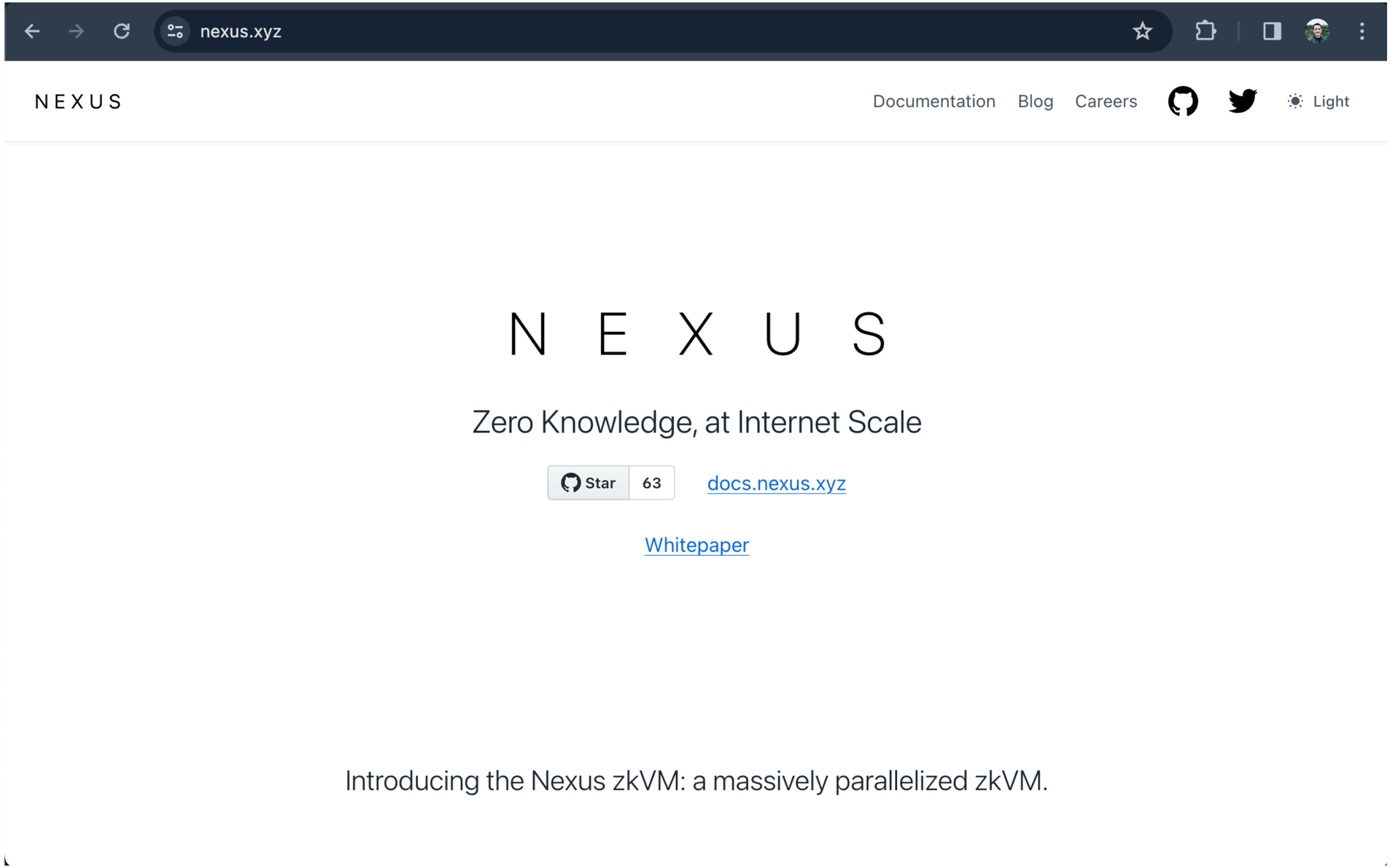


Introducing

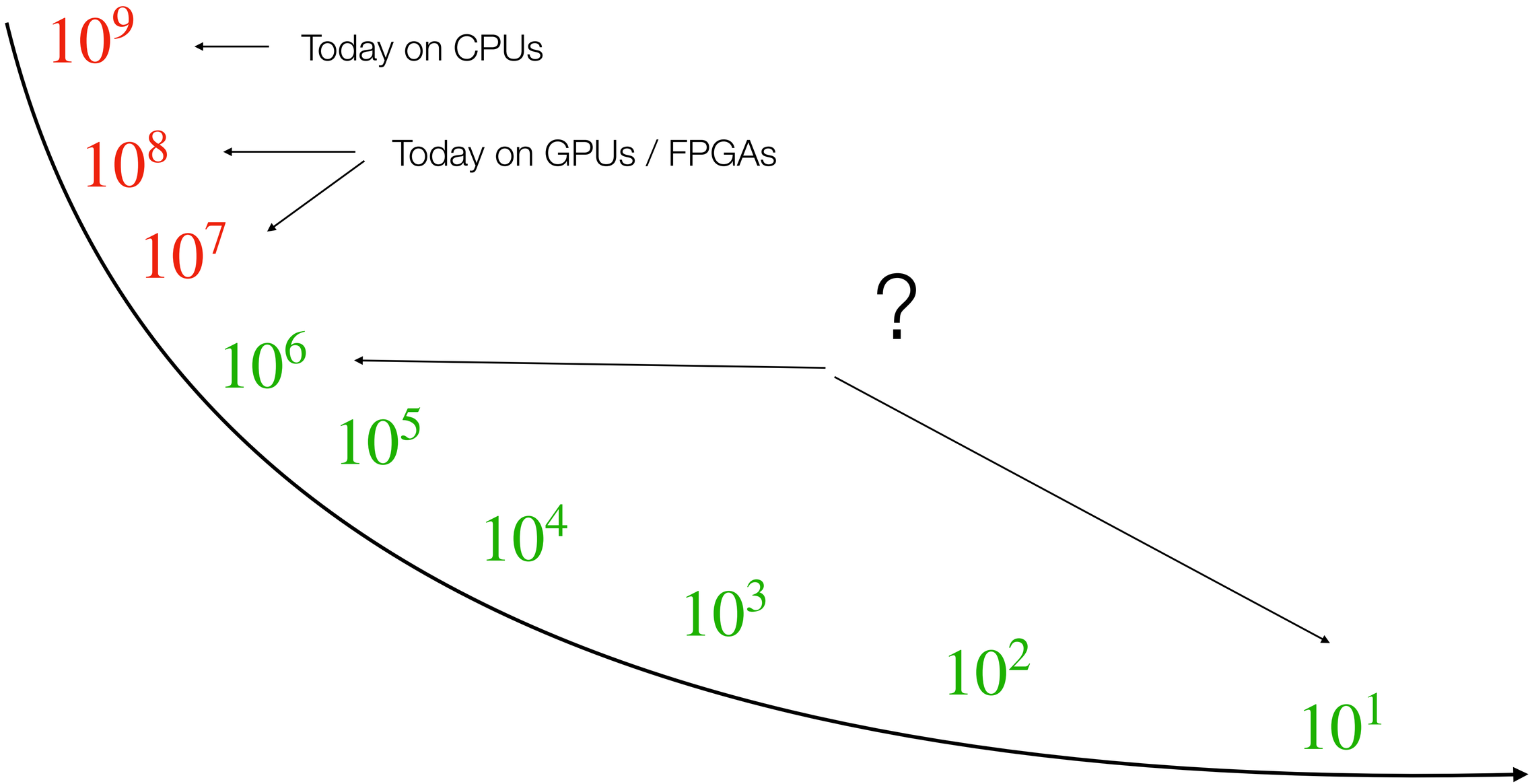
The Nexus zkVM

The Nexus Network

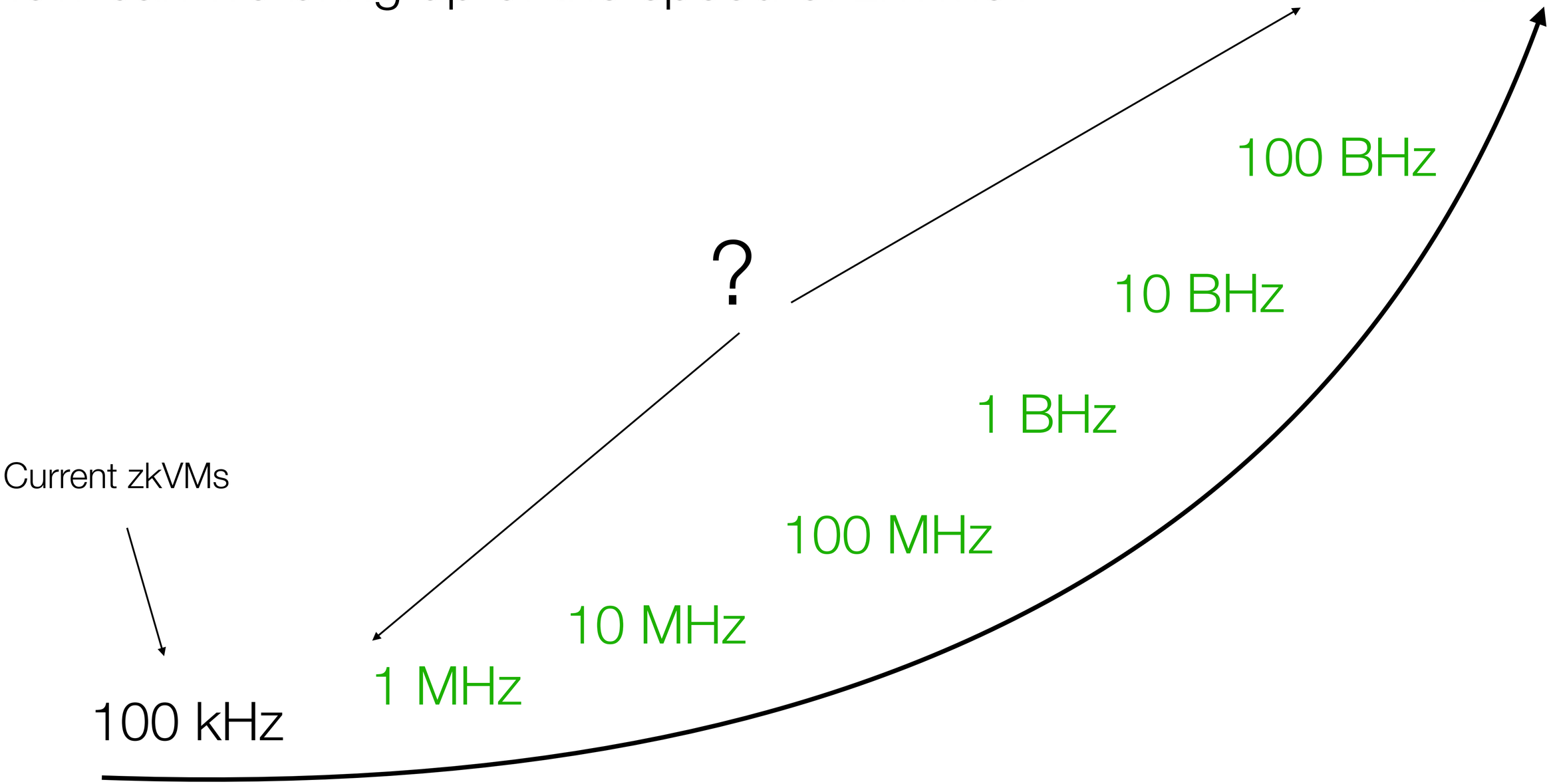
Open-sourced (MIT / Apache 2.0) 8 weeks ago: nexus.xyz



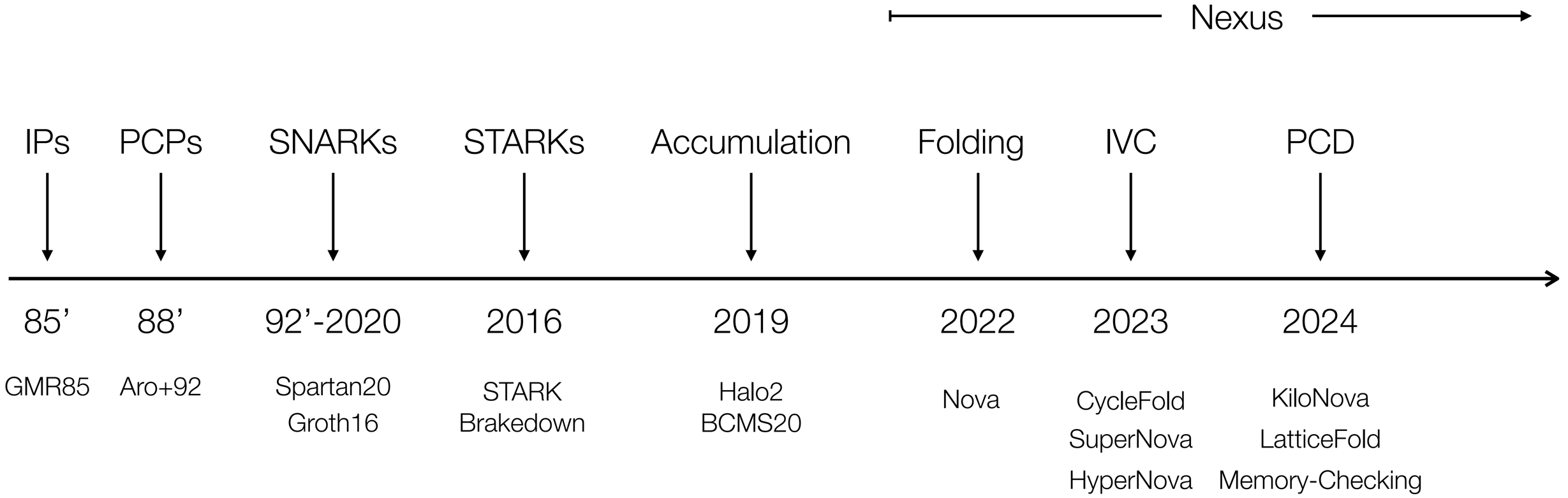
How can we bring down the cost of ZK?



How can we bring up of the speed of zkVMs?



History of Zero-Knowledge Proofs



(Read more on the Nexus Whitepaper)

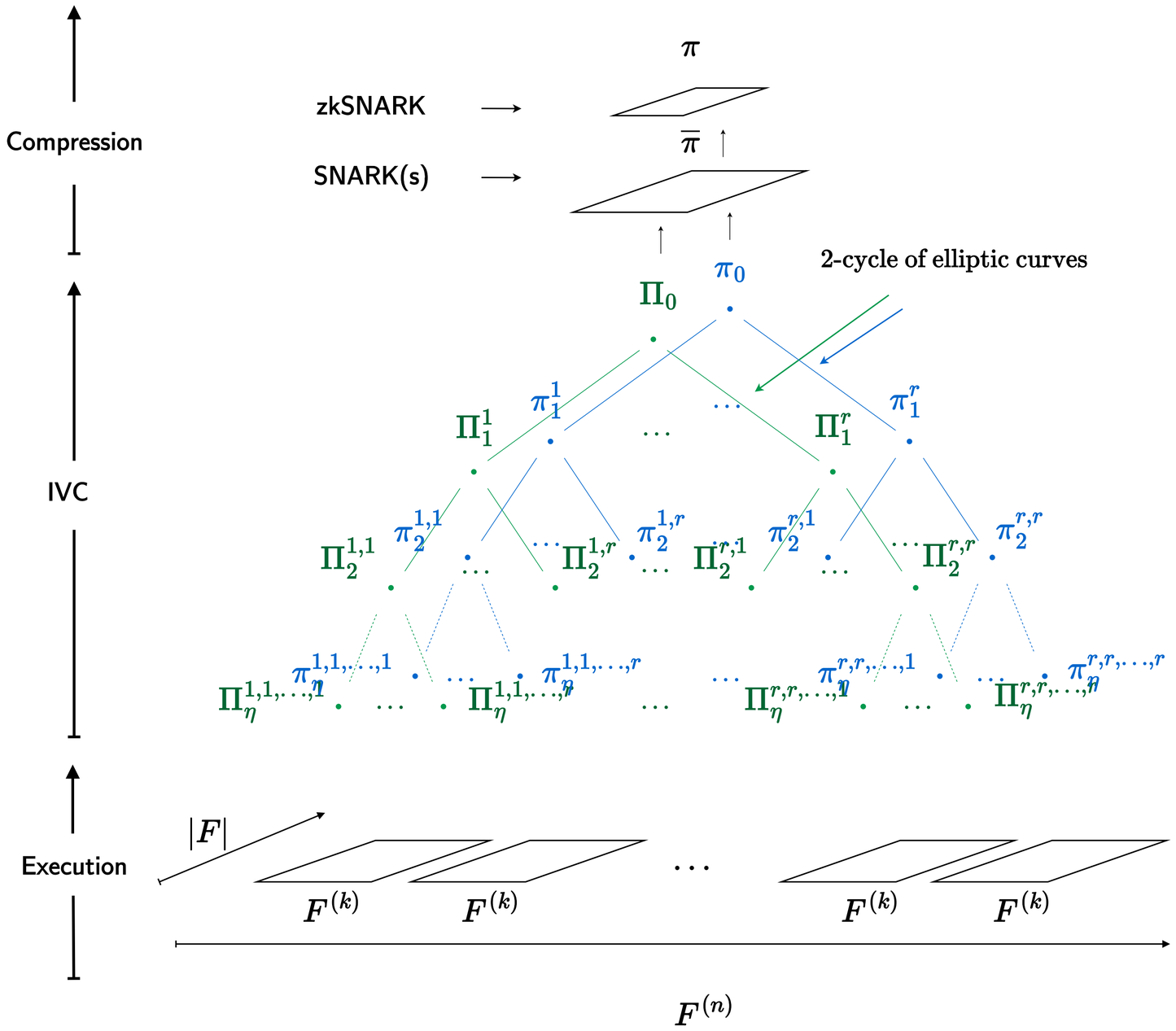
Reaching the next 1,000,000x in zkVM speed

requires totally new cryptography, GPUs are not enough

Going Beyond

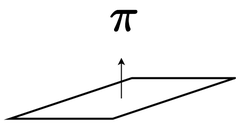
The Nexus zkVM

A zkVM designed to reach 1 trillion Hz

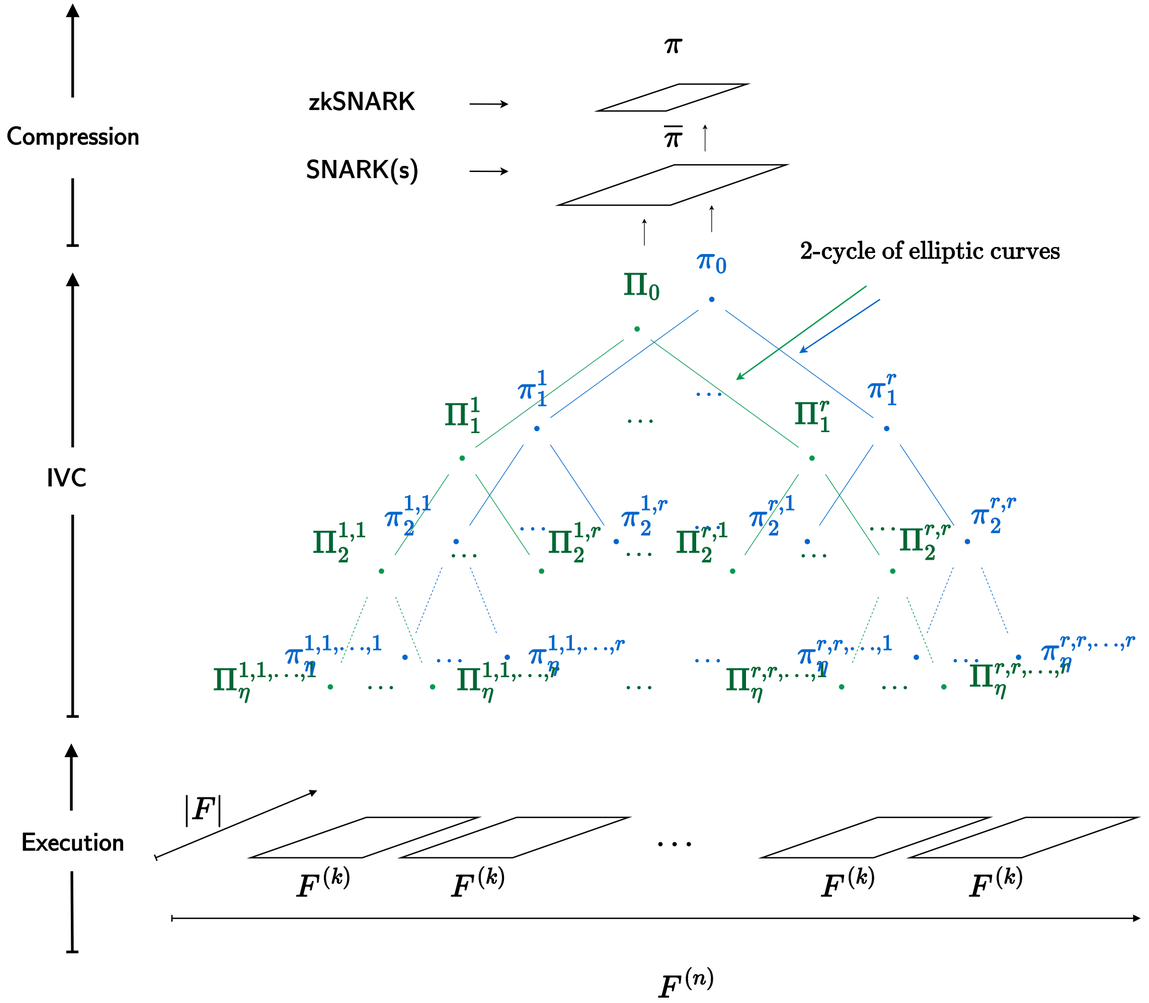


Traditional zkVM (2013-now)

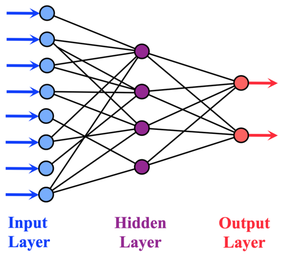
zkSNARK $\longrightarrow$ π



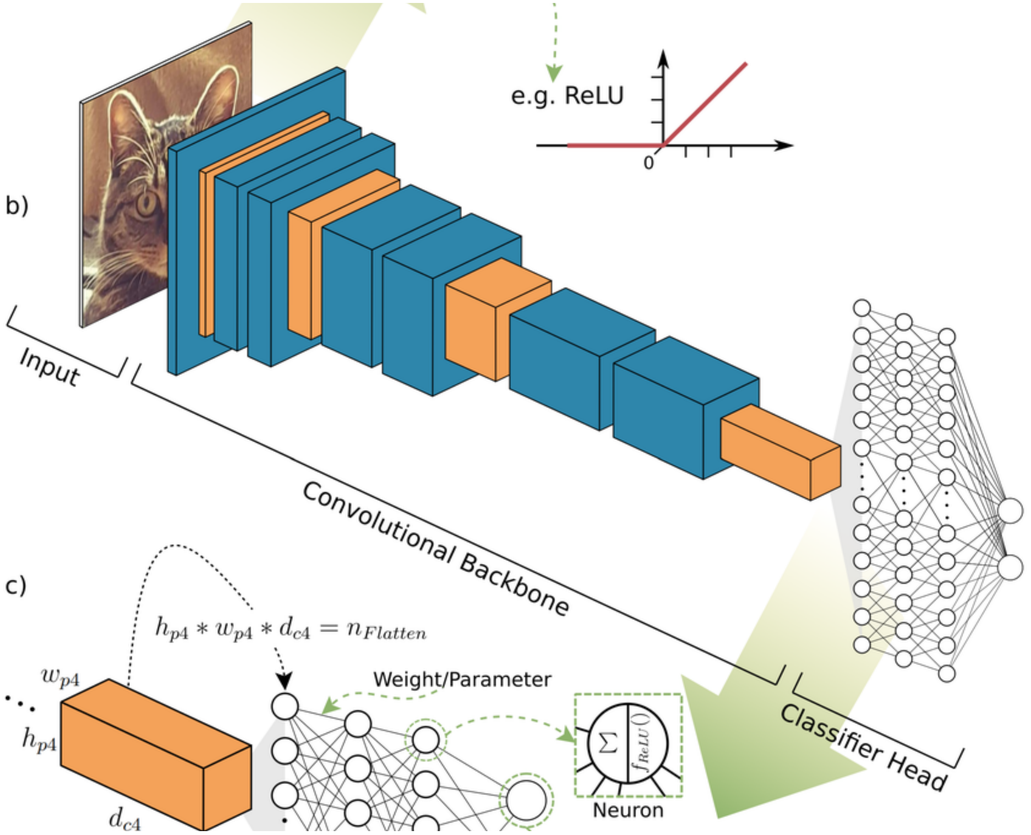
The Nexus zkVM (2024)



Simple neural network (1962)



Convolutional Neural Network (2012)



First from-the-ground-up open-source implementations of Folding Schemes

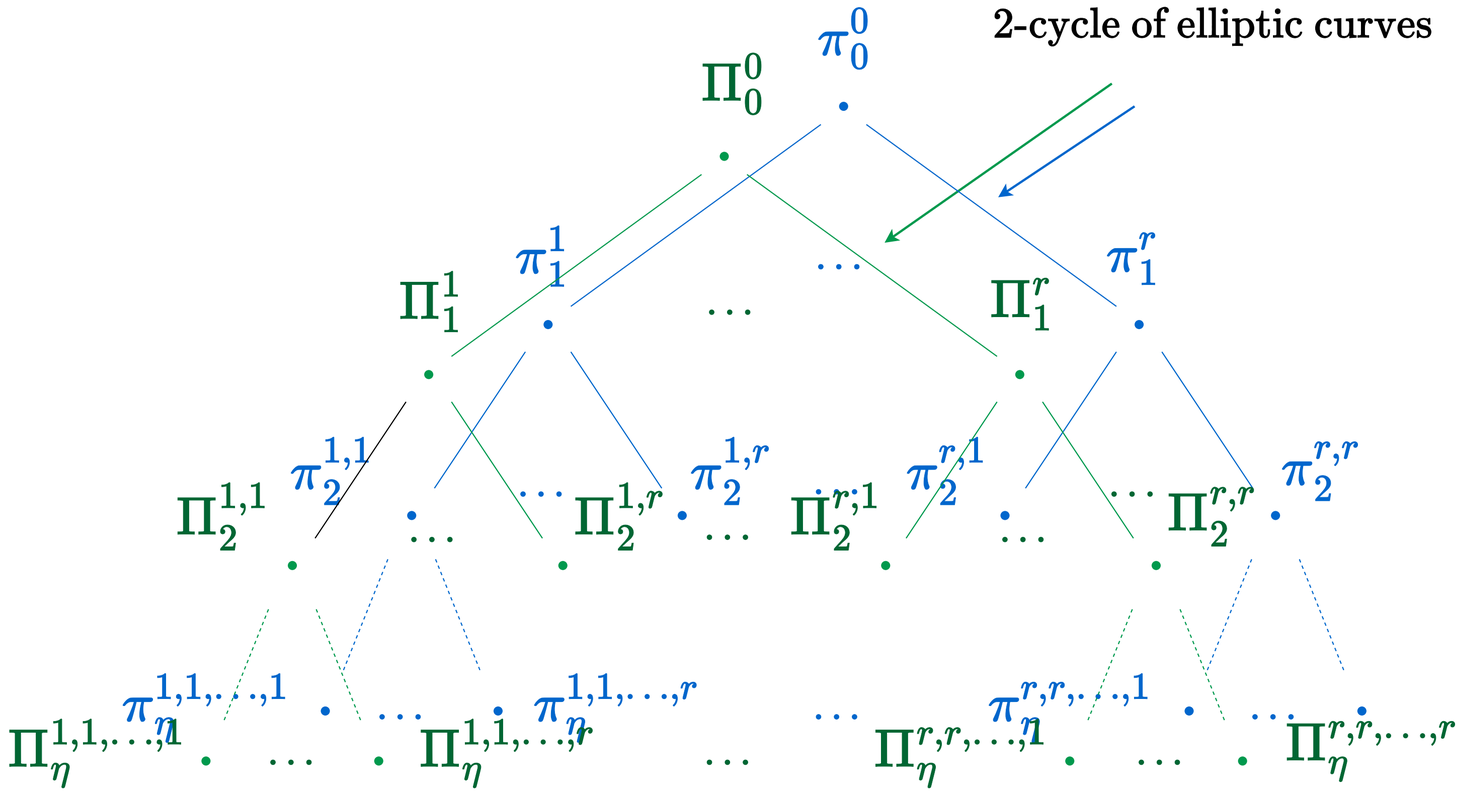
Nova, SuperNova, HyperNova, CycleFold

(modified) Spartan, Zeromorph, and more

https://github.com/nexus-xyz/nexus-zkvm

MIT License

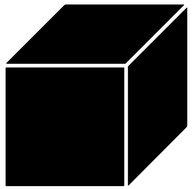
Distributed proof aggregation



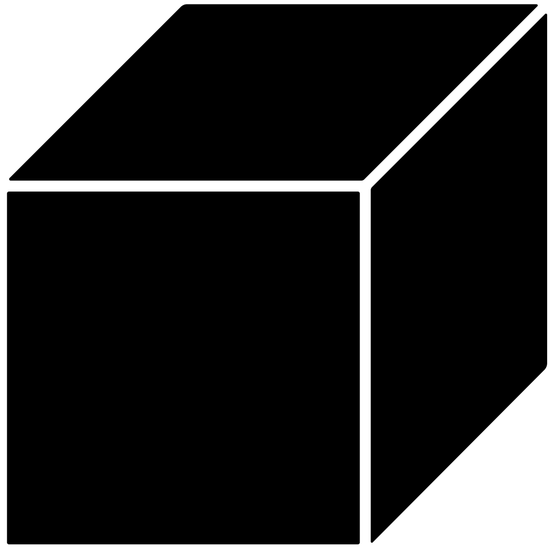
A tradeoff in proof size

SNARKs
(e.g. Groth16)

STARKs
(e.g. Aurora)

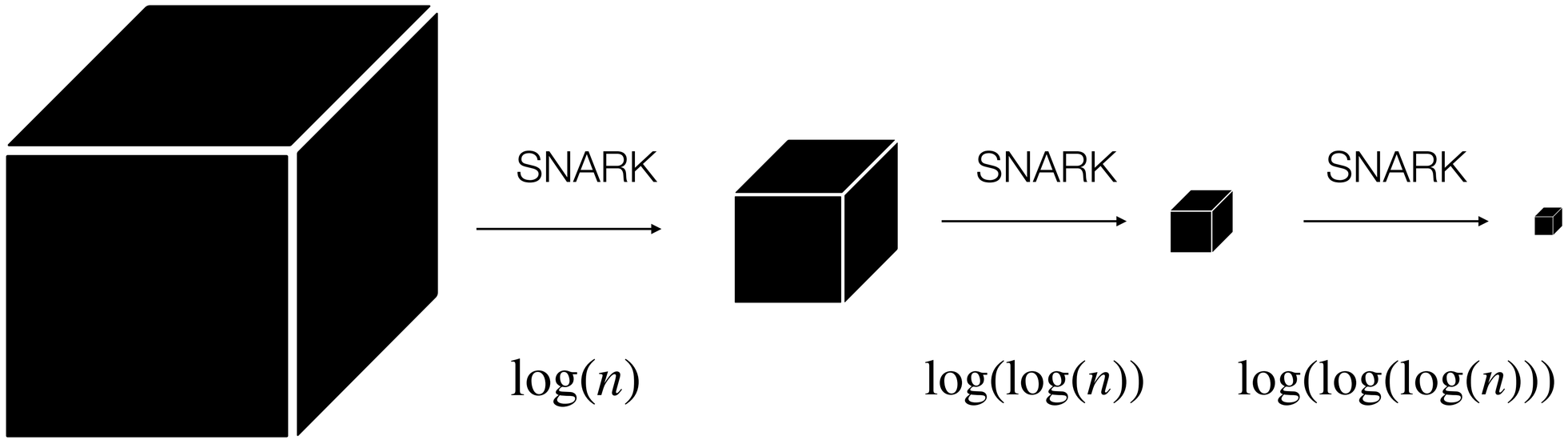


Folding
(e.g. HyperNova)



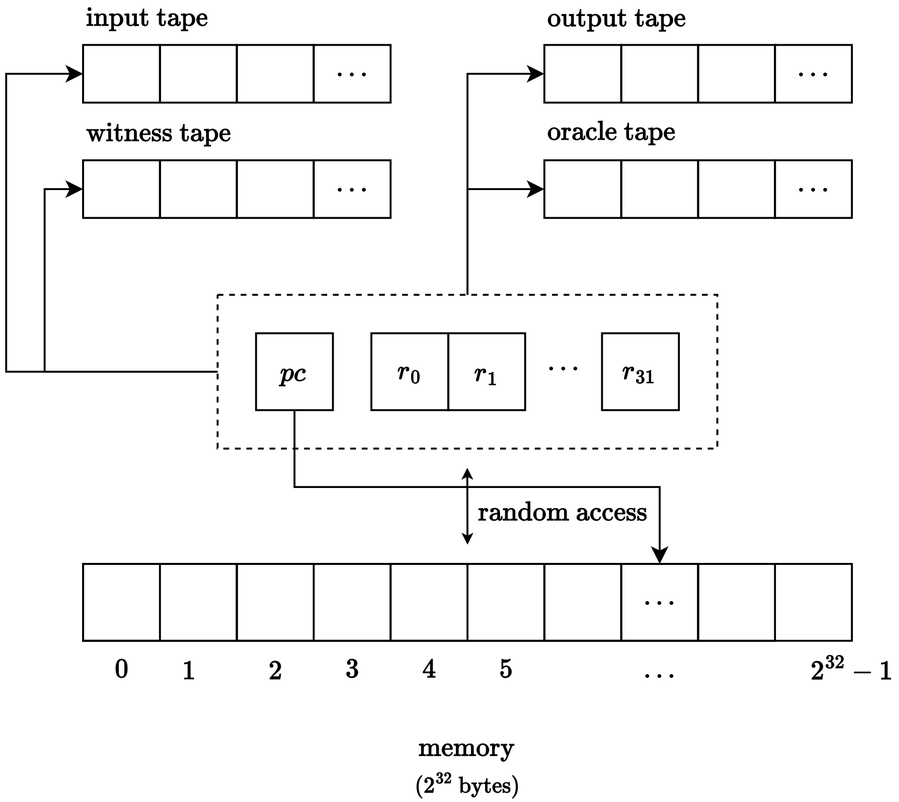
Nexus Proof Compression Sequence

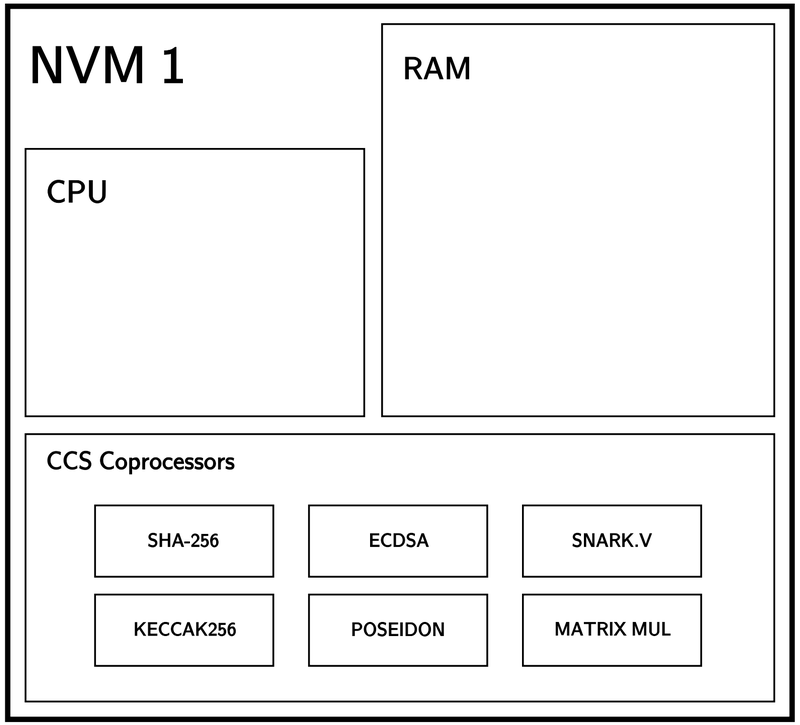
Nova Proof



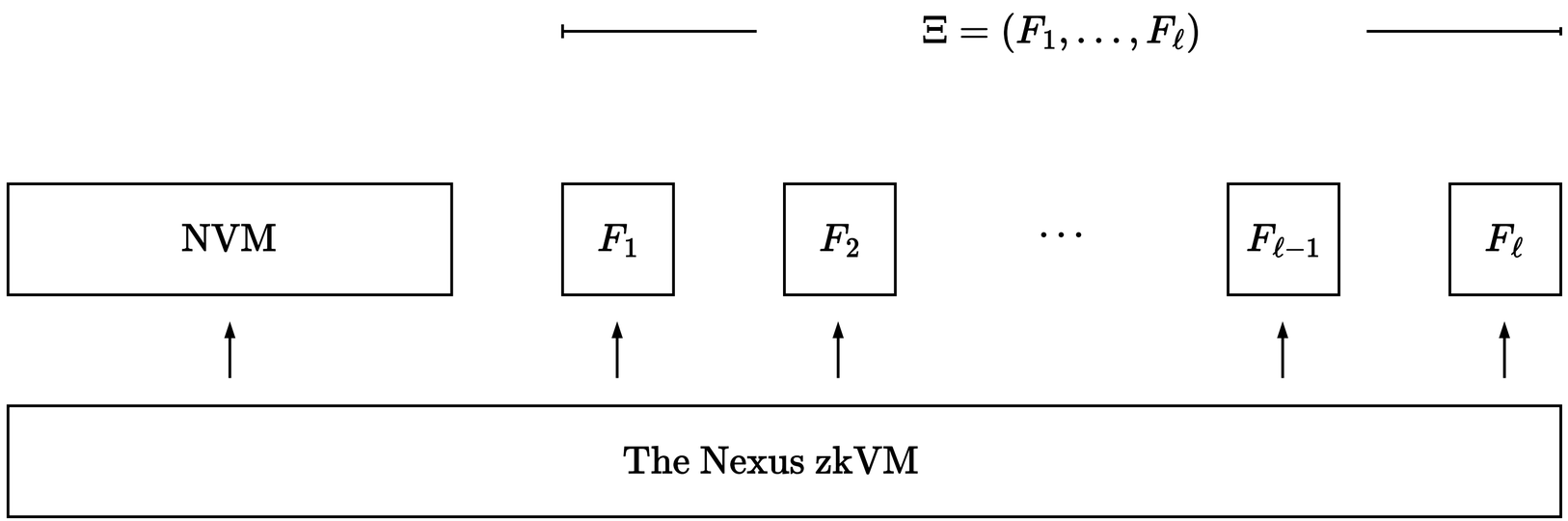
The Nexus Virtual Machine (NVM) 1.0

A prover-optimized CPU Architecture





The Nexus Precompile System



Custom accelerated instructions (e.g. SHA-256)

Enabled by SuperNova-based non-uniform IVC

User Experience

Build your own zkVM

in Rust

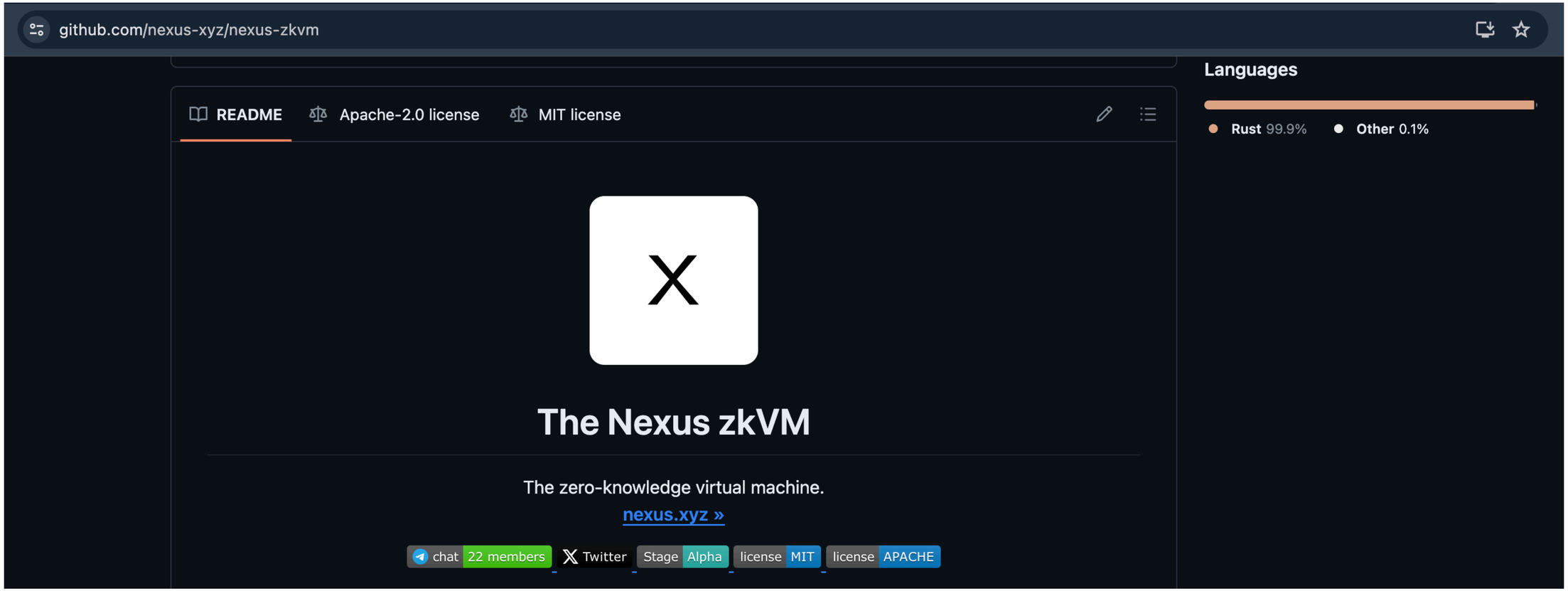
```
$ cargo nexus prove
$ cargo nexus verify
```

```rust
#![no_std]
#![no_main]

fn fib(n: u32) -> u32 {
    match n {
        0 => 0,
        1 => 1,
        _ => fib(n - 1) + fib(n - 2),
    }
}

#[nexus_rt::main]
fn main() {
    let n = 7;
    let result = fib(n);
    assert_eq!(result, 13);
}
```

Fully open-source and usable today



		Features
Nexus 1.0 zkVM	Today	NVM 1.0 Proof compression 1.0 IVC & PCD 1.0 Memory Subsystem 1.0 Compiler 1.0 Witness Extractor 1.0
Nexus 2.0 zkVM	Soon	Many expected upgrades

The Nexus IVC Prover 1.0

The Nexus IVC Prover 1.0: Recursion *without* SNARKs

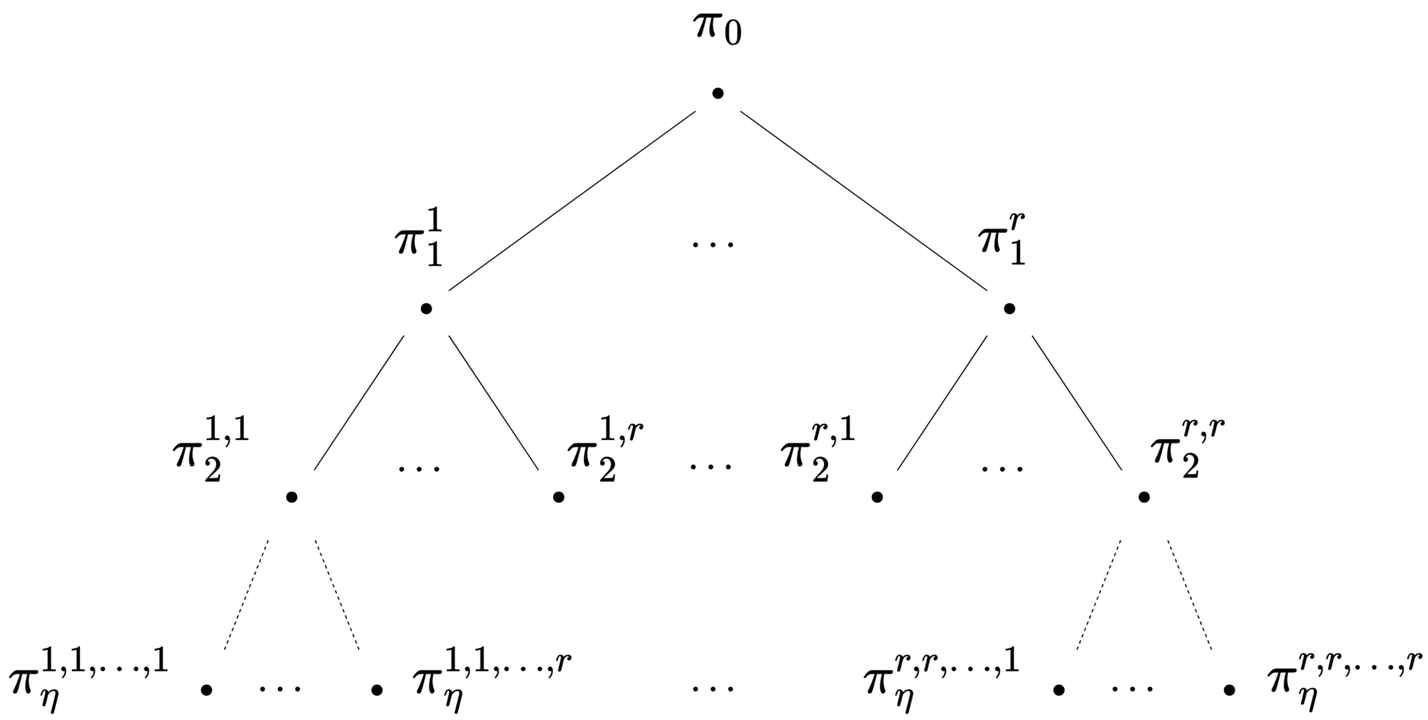


Figure 1: An r-ary proof accumulation tree of depth η, *without* SNARKs.

The Nexus IVC Prover 1.0: Recursion *without* SNARKs

Why:

- Very low recursion overhead
- Very low memory consumption
- Very fast proving: only 1 MSM

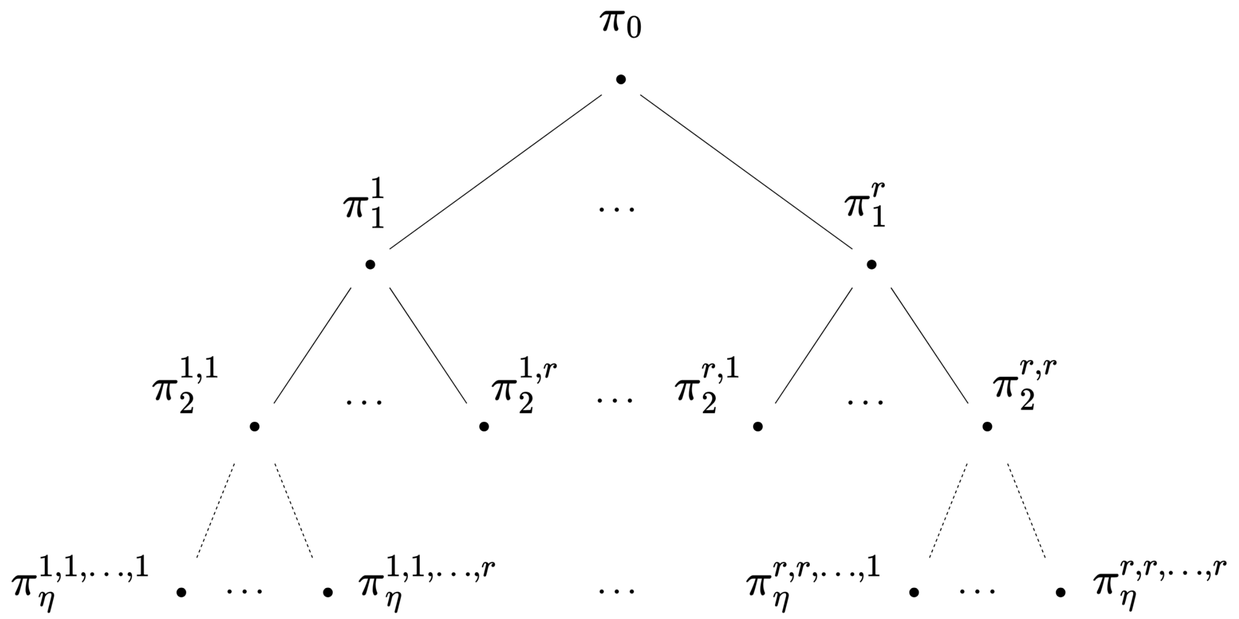
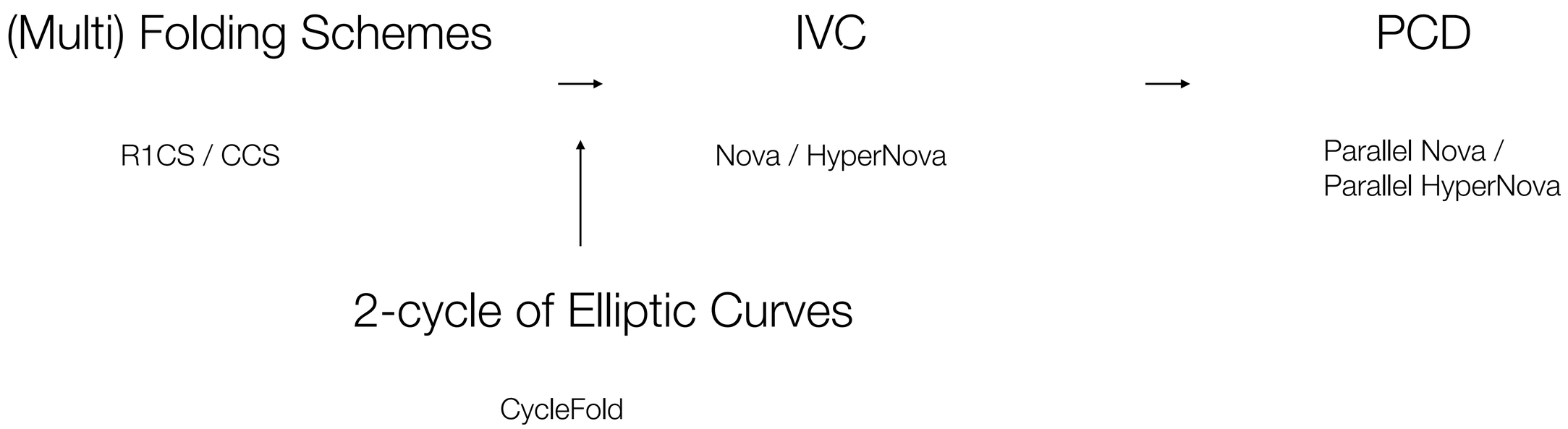


Figure 1: An r-ary proof accumulation tree of depth η, *without* SNARKs.

Nexus 1.0 IVC Prover



Nexus 1.0 IVC Prover: Folding CCS Instances

Advantage: We can prove R1CS, Plonkish and AIR!

4.8 Customizable Constraint Systems

4.8.1 The CCS Relation

Definition 4.17 (CCS [STW23a])**.** Let $\mathcal{R}_{\mathsf{CCS}}$ be the customizable constraint system relation (CCS), defined as follows. An $\mathcal{R}_{\mathsf{CCS}}$ structure

$$\mathsf{s}_{\mathsf{CCS}} = (m, n, N, \ell, t, q, d, [M_i]_{i=1}^{t}, [S_i]_{i=1}^{q}, [c_i]_{i=1}^{q})$$

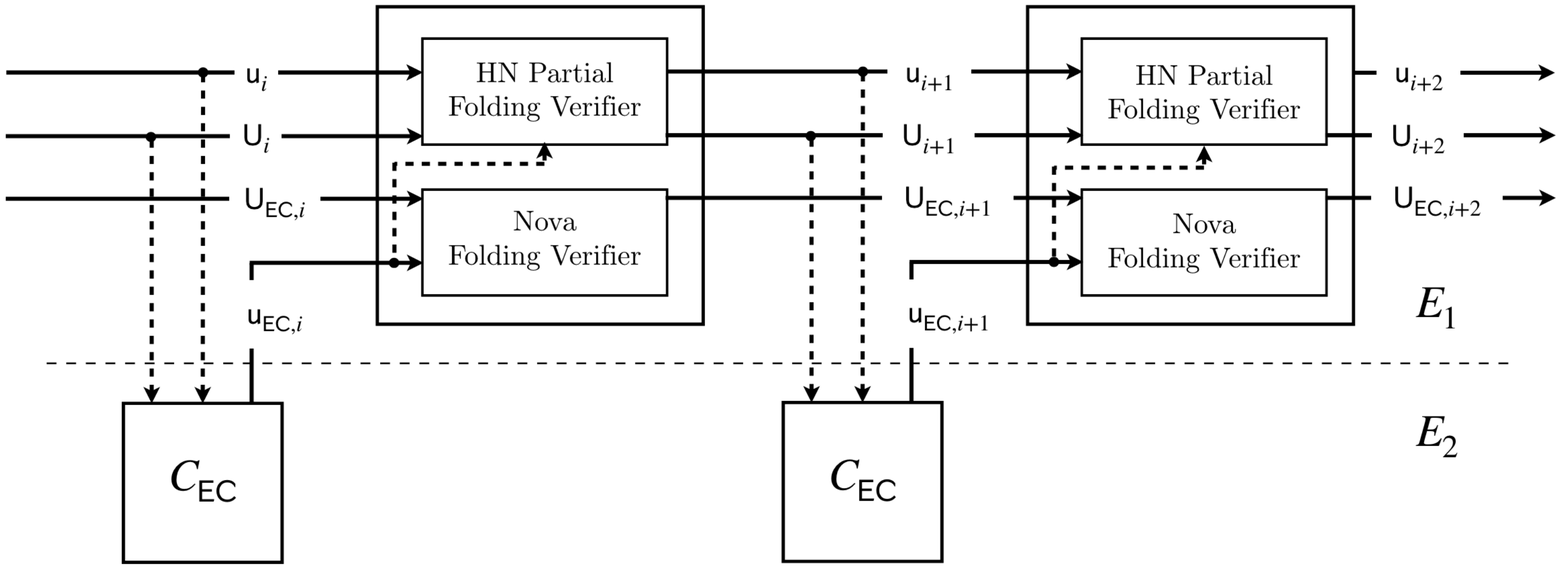
consists of:

- Size bounds $m, n, N, \ell, t, q, d \in \mathbb{N}$, where $n > \ell$.
- A sequence of t matrices $[M_1, \ldots, M_t]$ each in $\mathbb{F}^{m \times n}$, with at most $N = \Omega(\max(m, n))$ non-zero entries.
- A sequence of q multisets $[S_1, \ldots, S_q]$ each of cardinality at most d over the domain $\{1, \ldots, t\}$.
- A sequence of q constants $[c_1, \ldots, c_q]$ each in $\mathbb{F}$.

An $\mathcal{R}_{\mathsf{CCS}}$ instance x consists of public input $x \in \mathbb{F}^{\ell}$. An $\mathcal{R}_{\mathsf{CCS}}$ witness consists of a vector $w \in \mathbb{F}^{n-\ell-1}$. A CCS structure-instance tuple (s, x) is satisfied by a CCS witness w if

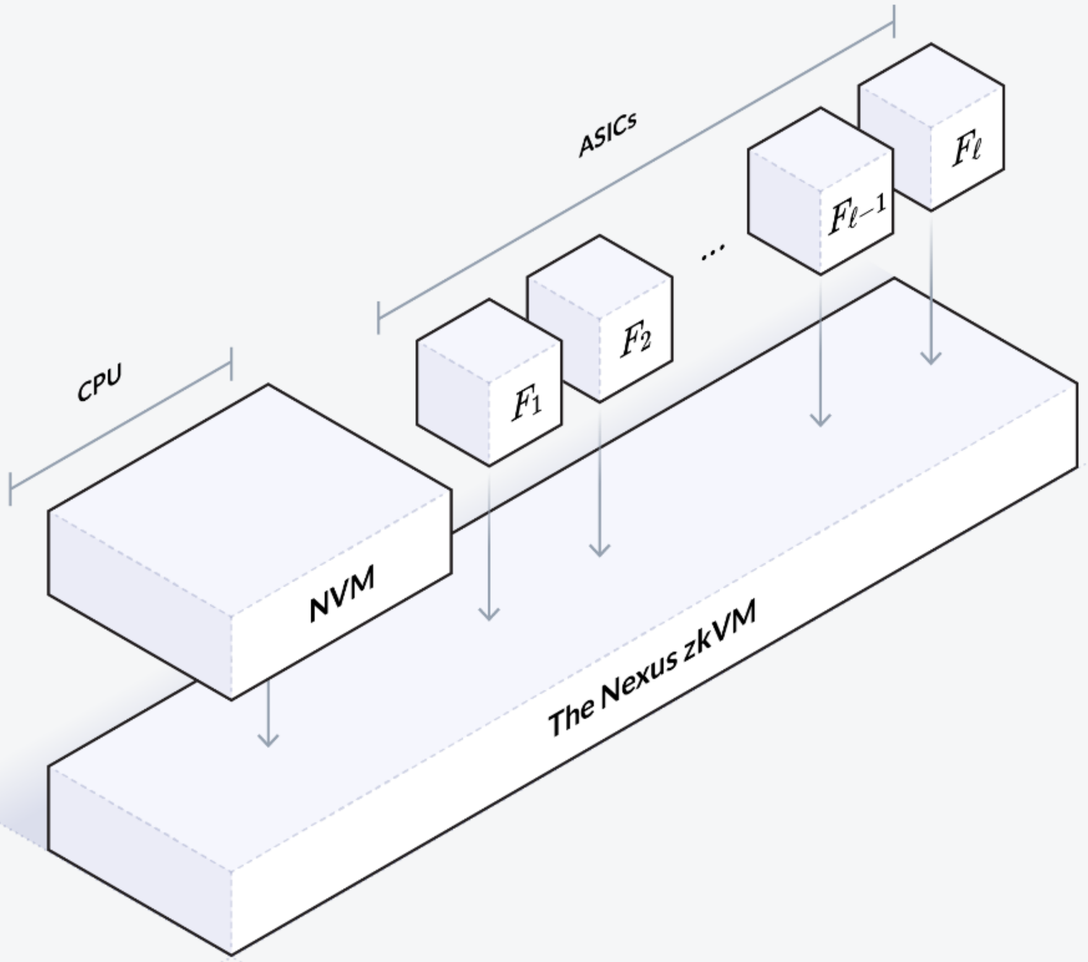
$$\sum_{i=1}^{q} c_i \cdot \bigcirc_{j \in S_i} M_j \cdot z = \mathbf{0}$$

Nexus 1.0 IVC Prover: CycleFold for 2-cycle of curves



Kothapalli & Setty, 2022

Nexus 1.0 IVC Prover: SuperNova “Co-processors”



Nexus Precompile System 1.0

Nexus 1.0 IVC Prover: Parallel HyperNova for PCD

5.7 Parallel HyperNova

Construction 5.7 (Parallel HyperNova)**.** Let $\mathsf{NIMFS} = (\mathsf{G}, \mathsf{K}, \mathsf{P}, \mathsf{V})$ be the non-interactive multi-folding scheme for $(\mathcal{R}_{\mathsf{LCCCS}}, \mathcal{R}_{\mathsf{CCCS}}, \mathsf{compat}, \mu, \nu)$ from [ZZD23]. We construct a PCD scheme from HyperNova for r-ary tree transcripts.

A message z is a tuple $(n, z^{(\ell)}, z^{(r)})$ attesting to $F^{(n)}(z^{(\ell)}) = z^{(r)}$. We define the trivial message as $\perp = (0, \perp, \perp)$. We define T to be an r-ary tree, where each node $v = (n, z^{(\ell)}, z^{(r)}) \in V(\mathsf{T})$ is labeled by its outgoing message.

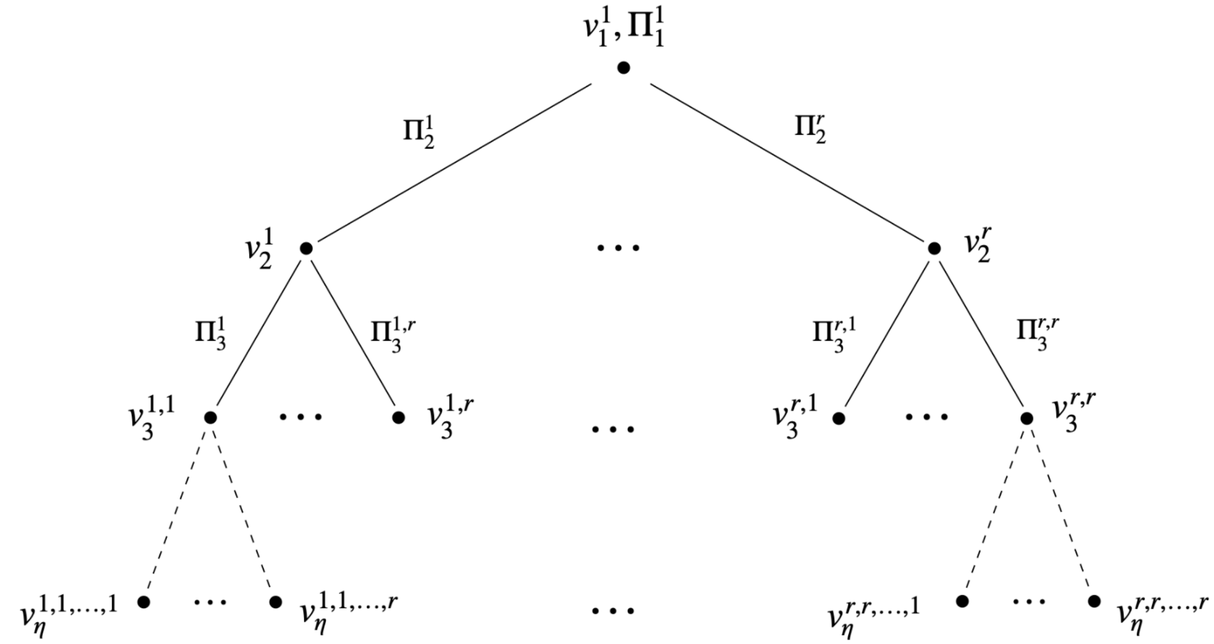
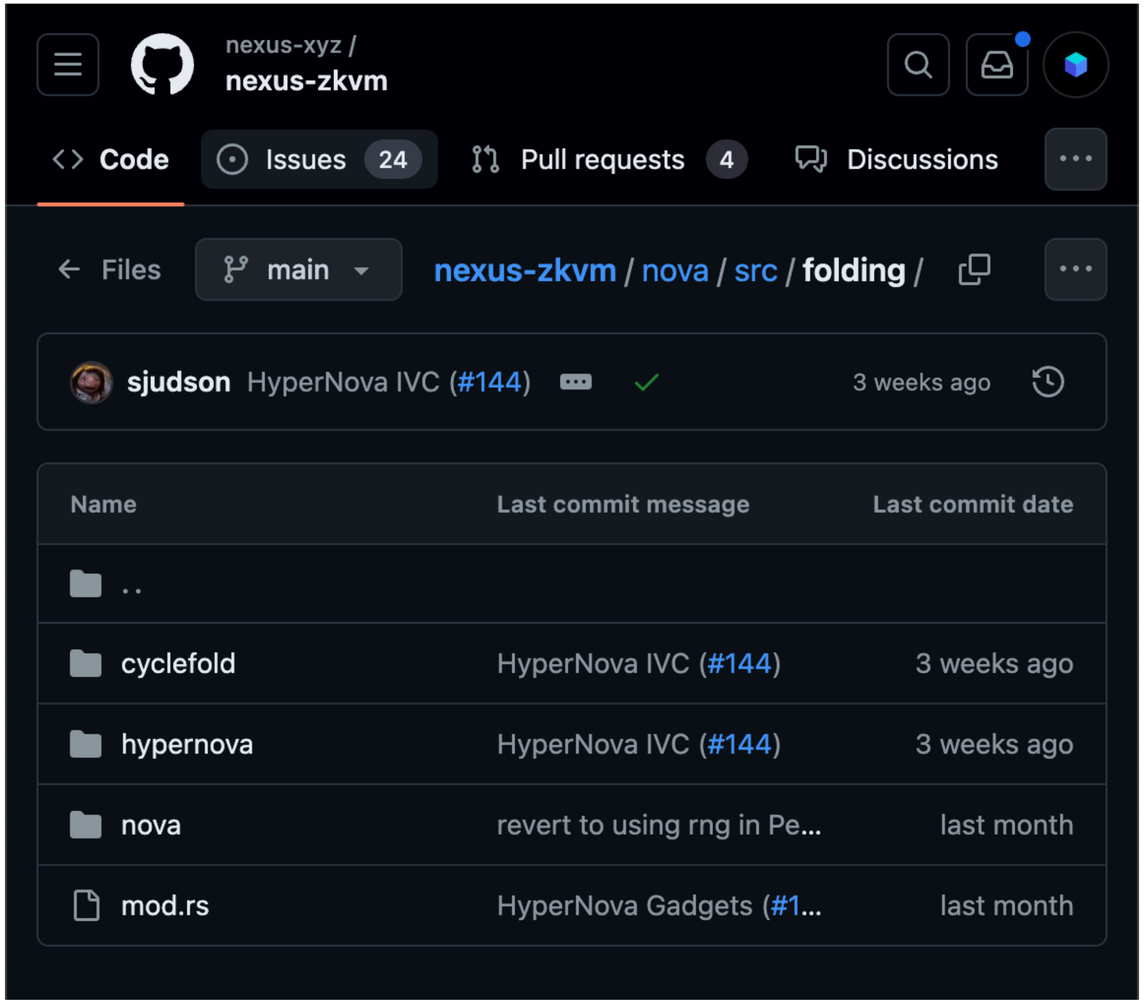


Figure 8: An r-ary tree transcript of a PCD scheme for a parallel HyperNova construction.

Nexus 1.0 IVC Prover: Nova + CycleFold + HyperNova

All open-source: MIT License

github.com/nexus-xyz



Nexus 1.0 IVC Prover

Upsides:

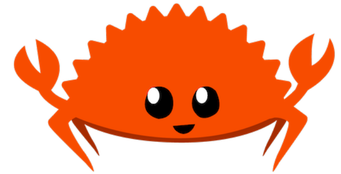
- Fast recursion (~10k rec. overhead)
- Low memory overhead
- Prove CCS instances (any of R1CS, Plonkish or AIR)
- Compatible with other SNARKs or STARKs
- SuperNova: IVC-compatible precompiles
- Highly parallelizable

Downsides:

- Two-cycle of Elliptic Curves (complex engineering)
- Big fields
- Pedersen commitments require a CRS
- High communication overhead

Nexus 2.0?

Contribute to the Nexus 1.0 zkVM today!



https://github.com/nexus-xyz

Fully open source, all Rust, contributor-friendly, MIT License

nexus.xyz