LP-200 — Lux versus Avalanche: A Quantitative Audit of Fork Divergence and Post-Quantum Substrate Reconstitution

Zach Kelling

Lux Industries

June 2026

LP-200 — Lux versus Avalanche: A Quantitative Audit of Fork Divergence and Post-Quantum Substrate Reconstitution

Lux Industries

June 2026

Abstract

Lux Network began in early 2020 as a fork of `ava-labs/avalanchego`. As of June 2026, the Lux substrate retains essentially no upstream Avalanche Go code: a recursive grep across the Lux workspace finds zero `ava-labs` or `avalanchego` *import* paths in production `*.go` files (three residual *string* references survive in `netrunner/server/network.go`, `netrunner/rpcpb/rpc.pb.go` as supported node-type identifiers for upstream interop and in one stale comment in `state/cmd/export_blocks.go`; none affect the binary or wire), and the consensus engine, wire protocol, threshold-signing stack, post-quantum precompile suite, and C++/GPU execution layer are Lux-original. Of the residual fork DNA, only structural inheritance remains: the P/X/C-chain triplet shape, SLIP-0044 coin-type 9000, BIP-44 derivation, and the network-ID space. Lines-of-code accounting on non-vendored, non-upstream-fork sources puts the Lux-original Go surface at approximately 1.05 M LOC across ∼30 top-level modules plus ∼2.36 M LOC of Lux-original C++/CUDA/Metal/WebGPU in `luxcpp/`; the `luxfi/node` fork target itself is now 288 k LOC, 46% the size of upstream `avalanchego` (624 k LOC at `9f8b9461`, 2026-05-14), and contains zero references to Avalanche-family naming. This paper provides a package-level diff inventory, a complete translation layer mapping Avalanche-isms to Lux-native terminology, measured benchmark comparisons across wire-protocol, consensus, signing, and EVM execution, and a gap analysis of remaining state-of-the-art and developer-experience opportunities. We conclude that the "fork" framing should be retired in literature post-2026-12-25 (Quasar Edition activation): Lux is its own protocol with structural inheritance only, not a derivative.

Contents

1 Question and method

We investigate four questions:

1. How much upstream Avalanche code remains in the Lux substrate?
2. Where does Lux already lead the state of the art, and on what measured benchmarks?
3. Which Avalanche-derived terminology is now misleading to engineers, and what is the correct Lux-native vocabulary?
4. Where can Lux further improve developer experience and performance without rearchitecting?

Method. We enumerated every Go module in `~/work/lux/` via `go.work`, ran lines-of-code counts on each tree, classified each package by origin (Lux-original / forked-diverged / forked-near-identical / vendored), and verified import-path provenance via `grep -r 'ava-labs|avalanchego'` on `*.go` files. For the C++/GPU side (`~/work/luxcpp/`) we counted source files across `*.cpp`, `*.cu`, `*.h`, `*.hpp`, `*.metal`, `*.wgsl`, `*.cl`. Reference upstream is `~/work/ava/avalanchego` at commit `9f8b9461cf8f0141dda716fa3ac1ec30f168fa66` (2026-05-14, master). Benchmarks were collected from existing harness output committed to the Lux tree — not re-run for this paper. New benchmarks proposed in Section 4 are marked TBM.

2 Inventory: package-level diff

Table 1 summarises the Go workspace (only significant packages; complete data in `inventory.csv`). Avalanchego's reference baseline at the inspected commit is 2,785 Go files / 623,940 LOC.

2.1 Key observation: what was forked, what was replaced

The `node/` directory still inherits the structural shape of `avalanchego` — the file layout under `network/`, `message/`, `nat/`, `router/`, `indexer/` is recognisable. But the consensus engine (`snow/` in `avalanchego`) has been *extracted entirely* into a separate `luxfi/consensus` repository (98,317 LOC) and rewritten with no snowball/snowman lineage. The VM tree under `node/vms/` has replaced `avm/`, `saevm/`, `nftfx/`, `propertyfx/`, `metervm/` with `xvm/`, `dexvm/`, `mldsafx/`, `thresholdvm/`, `chainadapter/`, `da/`, `artifacts/`. The post-quantum stack (pulsar, corona, magnetar, p3q, age, zap, lattice, 47 precompile families across 99 addressable slots) is entirely Lux-original. Even the `platformvm/txs/zap_native/` subtree — the transaction wire encoding for the new final Lux network activating 2025-12-25 16:20 PST — contains zero upstream code: it uses the ZAP zero-copy buffer discipline where the Go struct *is* the wire (see LP-023).

2.2 Lux-original substrate

Beyond the `node/` fork target, Lux has built an entire post-quantum and GPU substrate with no Avalanche analog:

- **Pulsar** (FIPS 204 ML-DSA-65 threshold; LP-073; NIST MPTC submission package).
- **Corona** (Round-Signer + DKG2 PVSS dealerless DKG).
- **Magnetar** (FIPS 205 SLH-DSA threshold; THBS-SE strict-atom v1.1; −51% vs circl baseline at SHAKE-192s).
- **Quasar** (BLS + Pulsar + ML-DSA hybrid; PQ mode selector; LP-010, LP-020).
- **P3Q** precompile (Rust crate; hybrid PQ verify at slot `0x012205`).
- **ZAP** wire protocol (LP-022, LP-023; zero-copy; 42M ops/sec encode, 156M ops/sec decode; canonical batch 5 v3.7 geomean 7.50× Parse vs upstream P-chain linearcodec, 3.57× allocation reduction, 1 alloc/24 B uniform).
- **ZapDB** (38 k LOC key-value store; native ZAP integration).
- **47 EVM precompile families** with 99 addressable slots across C-Chain / Q-Chain / multi-chain (ML-DSA / SLH-DSA / ML-KEM / Pulsar / Quasar / FROST / CGGMP21 / FHE / HQC / KZG4844 / Poseidon / Pasta / Pedersen / VRF / X25519 / Curve25519 / BabyJubJub / BLS12-381 / Ring / ZK / Blake3 / Solidity-bindings; 79,251 LOC).
- **KMS** (`lux/kms` 12.6 k LOC backend + `hanzo/kms` 39.0 k LOC canonical operator; both ride the consensus-native authorizer, mnemonic-derived service identity, envelope auth, and ML-DSA-65 signature path).
- **Lux C++/GPU layer** (`luxcpp/`; ∼2.36 M LOC of native C++/CUDA/Metal/WebGPU; C++ EVM at 20.9 Ggas/s).

3 Translation Layer: Avalanche-isms to Lux Native

The Lux substrate began as a fork of `ava-labs/avalanchego` in early 2020, but as of June 2026 the source tree contains zero references to `ava-labs` or `avalanchego` import paths (verified by recursive grep across the Go workspace). The protocol, wire format, consensus engine, post-quantum stack, and GPU/C++ execution layer are Lux-original. What remains shared is structural inheritance: the P/X/C-chain triplet shape, the SLIP-0044 coin-type, and BIP-44 derivation. This section catalogs the terminology mapping for engineers migrating between literature.

3.1 Source-code anti-patterns to scrub

The Go source tree under `~/work/lux/node` and `~/work/lux/consensus` contains zero `Avalanche`, `Snowman`, `Snowball`, `Avax`, or `Fuji` string literals in non-test `*.go` files. The remaining occurrences are:

- Academic citations in `consensus/paper/` and `papers/ .tex` sources (correct attribution).
- Upstream forks (`geth/`, `coreth/`) that retain their own provenance.
- Third-party vendored modules (`kms/vendor/` MinIO SDK contains `api-putobject-snowball.go` — unrelated AWS Snowball).

The 15 `Subnet`-prefixed identifiers in `node/vms/platformvm/txs/zap_native/` are forward-compatibility shims for the `ConvertNetworkToL1Tx` and `TransformChainTx` migration paths (subnet → L1 conversion, the Lux analog of Etna).

4 Benchmarks: Measured and Derived

This section aggregates measured performance numbers from the Lux substrate. All numbers below were collected on Apple M1 Max (10 cores, 64 GiB, macOS Darwin 25.4, Go 1.26.x) unless otherwise noted. Sources are git-pinned benchmark files in the Lux tree (e.g. `consensus/BENCHMARKS.md`, `pulsar/bench/results/REPORT.md`, `papers/lux-zap-benchmarks/`). Numbers reported as "TBM" (to be measured) indicate a Phase 2 deliverable not yet collected on dedicated WAN hardware.

4.1 Wire protocol: ZAP vs lineararcodec, protobuf, FlatBuffers

Note on v3.6 vs v3.7. The LP-023 Appendix A table reported a 8.5× geomean on a slightly different harness (Go 1.24, prior schema). The v3.7 result above (7.50×, `luxfi/node` post-batch-5-v3.7, `luxfi/zap` v0.7.2, Go 1.26.1) is within 4% of the v3.6 baseline (7.74×) and is the canonical reference for citations post-2026-06-02. The allocation profile is the more durable headline: ZAP holds 1 alloc/24 B *uniformly* across every tx type regardless of payload size, while lineararcodec scales linearly with field count (3–7 allocs / 120–384 B). This is structurally load-bearing: at $4 \cdot 10^4$ Parse ops/s per validator on the Lux 1 B-gas block, the GC pressure differential (24 B vs 384 B amortised per tx) is the single largest contributor to validator CPU budget.

Headline: ZAP decode is 11× faster than protobuf and 132× faster than JSON-RPC with zero allocations on the 145-byte consensus-vote payload. The upstream Avalanche `lineararcodec` is closer to the protobuf row in shape (reflection-based, 3+ allocations); the per-tx batch-5-v3.7 geomean 7.50× Parse speedup vs upstream (Table 7) is consistent with this — lineararcodec spends most of its budget on reflection dispatch and intermediate struct allocation, not on byte movement.

4.2 Consensus signing primitives (CPU baseline)

The strict-atom path is FASTER than dispatch-to-circl because the Magnetar-internal SHAKE walk avoids circl's per-call `PrivateKey` unmarshal + state struct initialisation overhead, while preserving byte-identity to FIPS 205 verifiers.

4.3 Quasar consensus protocol microbenchmarks

Headline: Aggregated BLS verification of 100 signers completes in 875 µs — constant in committee size by aggregation. A QuasarCert (21-validator BLS aggregate + ML-DSA + Pulsar) is producible in ∼400–500 ms CPU (Groth16 dominates) and verifiable in 2–5 ms.

4.4 PQ-mode wire and storage trade-offs

4.5 EVM execution: Lux C++ EVM vs go-ethereum vs reth

Headline: Lux's native C++ EVM (evmone+state) sustains 20.9 Ggas/s in measured benchmarks — roughly 400–800× a mainstream geth installation, 100× reth live-sync, and within 30× of reth's best opBNB historical-sync number. The 87% of block time spent in secp256k1 ecrecover (measured at 334 ms of 384 ms total) is the bottleneck targeted by the parallel-ecrecover and GPU ecrecover paths.

4.6 Quasar finality vs published consensus protocols

Headline: Quasar's classical (BLS-only) path provides the only sub-second, post-quantum-equipped finality in the published consensus literature. The PQ-finality column (Pulsar + ML-DSA-65) is a strict superset of any non-PQ protocol on this list.

4.7 Phase-2 (TBM): production WAN benchmarks

The following benchmarks are designed but *not yet measured* on dedicated WAN-distributed hardware. They are listed here as a Phase-2 deliverable. Reproduction harness lives at the cited paths.

- **WAN finality at 21 / 100 validators** — `papers/lux-quasar-benchmarks/sections/15-*.tex` rows currently marked `[TBD]`. Repro via Lux netrunner (`luxfi/netrunner`) on 4-region GCP/AWS topology.
- **GPU EVM ecrecover at scale** — `papers/evmgpu-benchmark` reports the 20.9 Ggas/s C++ path and parallel ecrecover. GPU ecrecover (Metal/CUDA) is target 20–24× vs CPU per Lux memory; no public baseline.
- **ZAP wire over QUIC at p99** — `papers/lux-zap-benchmarks/sections/04-latency.tex` reports 19.8 µs p99 single-host; cross-DC numbers pending.
- **Magnetar threshold orchestration overhead** — single-party numbers are in Table 9; network round-trip costs for THBS-SE Combine deferred to `luxfi/consensus` layer.
- **FHE precompile gas calibration at 12M block limit** — `papers/lux-fhe-benchmarks`; gas tables published per precompile, full-block calibration pending.

5 State-of-the-art positioning and DX / performance gaps

5.1 Where Lux already leads

1. **Zero-copy wire protocol**. ZAP is the only consensus-grade wire protocol in production at 6.4 ns/op decode with zero allocations and a 1.75× measured lead over Cap'n Proto on consensus messages. Cap'n Proto is the closest comparable. No production blockchain ships this.
2. **GPU-accelerated EVM**. Lux's C++ EVM sustains 20.9 Ggas/s on a single host. Reth's best historical-sync number is 690 Mgas/s. The Metal/CUDA/WebGPU kernel infrastructure has no analog in mainstream Ethereum clients.
3. **Post-quantum signing as first-class consensus**. Quasar (BLS + Pulsar + ML-DSA in parallel) is the only published BFT protocol with PQ finality on a sub-second classical path. Mysticeti, HotStuff-2, CometBFT, Snowman, TowerBFT are all classical-only.
4. **47-family cryptographic precompile surface (99 addressable slots)**. Production Ethereum has 9 precompiles, post-Pectra 11. Lux's precompile registry exposes roughly 4–9× depth depending on whether you count families or addressable slots, with ML-DSA / SLH-DSA / FHE / Pulsar / FROST / CGGMP21 available to contracts at gas-metered cost.
5. **Threshold-signing stack**. Magnetar's THBS-SE strict-atom path is faster than the audited `cloudflare/circl` baseline (−51% at SHAKE-192s) while byte-identical to the FIPS 205 verifier. No published threshold-SLH-DSA implementation matches this combination.

5.2 Structural inheritance: asset or liability

The P/X/C-chain triplet, SLIP-0044 coin type 9000, and BIP-44 derivation remain inherited from Avalanche. Three observations:

- **Coin type 9000 (SLIP-0044) is canonical inheritance and an asset.** It preserves wallet compatibility, hardware-wallet support (Ledger, Trezor), and existing key-derivation

tooling. `keys/service_identity.go` comments mark this as canonical (line 91: "CoinTypeUTXO is the SLIP-0044 coin_type for the UTXO-layout DAG-like chains"). Keep it.

- **P/X/C-chain shape is mostly an asset.** It buys established mental models, address formats (P-/X-/C- prefixes with bech32 `lux` HRP), and a clean separation between staking (P), UTXO/DAG (X), and EVM (C). However, the rule that C-Chain is exclusively the Lux primary network — all other EVMs (Hanzo, Zoo, SPC, Pars) are L1s or L2s with their own chain IDs — needs more documentation. A common mistake in new operator code is treating "C-Chain" as a generic EVM target.
- **Primary network ID 1 = mainnet shared across all Lux L1s is an asset and a footgun.** Asset because validators are shared across Hanzo/Zoo/Pars/Liquidity. Footgun because the per-subnet EVM chain ID (e.g. Liquid EVM 8675309, Hanzo 8000) is often confused with the network ID. The user-CLAUDE rule "NEVER set primary networkId to 8675xxx" captures the empirical mistake.

5.3 Top 3 SotA opportunities

1. **Publish WAN-distributed Quasar finality benchmarks at 21 / 100 / 1000 validators.** The single-DC 10 ms / 4.76M TPS number is striking but unverified outside a single host. Mysticeti has Fig. 5 / 6 published; Lux needs the equivalent. Repro harness exists at `luxfi/netrunner`; the table rows currently marked `[TBD]` in `lux-quasar-benchmarks/sections/15-*.tex` should land before LP-200 is non-draft.
2. **Land GPU ecrecover benchmark with comparable baseline.** `papers/evmgpu-benchmark` estimates Metal ecrecover at 20–24× vs CPU but the baseline measurement is `[TBD-baseline]`. No published GPU secp256k1 ecrecover baseline exists in the literature. Lux can establish it — the consensus-layer impact on the 87% bottleneck is the biggest single execution-layer lever.
3. **Productise the 47-family / 99-slot precompile registry as a developer narrative.** The depth is unprecedented but undermarketed. Ethereum has 11 precompiles; Solana has ∼60 native programs in a different model; Lux at 47 primitive families and 99 addressable slots is meaningfully different and underrated in Ethereum-developer marketing.

5.4 Top 3 DX / perf gaps

1. **Solana-native developers stumble on coin type 9000.** Solana uses coin type 501; Bitcoin 0; Ethereum 60. Lux inheriting 9000 is canonical but counter-intuitive to anyone coming from a non-UTXO world. Mitigation: a `luxfi/keys` doc note in the README explaining the choice, plus an HD-derivation snippet for the EVM C-Chain side that maps to coin type 60 for Ethereum-tooling compatibility (already partially supported by `cli/pkg/key/soft_key.go`; needs surfaced docs).
2. **P/X/C-chain mental model has too much surface for EVM-only consumers.** A developer building a DeFi app on Lux C-Chain does not care about P or X. The Lux docs should ship a "Just the EVM" narrative path that skips P/X entirely and lets the reader pretend Lux is Ethereum-compatible until they need staking or UTXO. Suggested LP: a `luxfi/docs` landing page that gates P/X content behind an explicit "Advanced" tab.
3. **Translation-layer drift in user-facing surfaces.** The codebase is clean (zero `Avax`/`Avalanche`/`Snowman` in production Go), but Lux's external docs, SDK READMEs, and blog still contain comparative phrasing like "Lux is built on Avalanche" or "Avalanche-compatible". As of LP-200, this is misleading. Recommended scrub: ship a `branding/translation-table.md` pinned in `luxfi/.github/profile` for docs maintainers.

5.5 Avalanche-language scrub backlog (source-side)

The `node/` and `consensus/` Go production trees are clean. The remaining sweep targets are:

- `papers/ .tex` academic citations: KEEP (correct attribution to Rocket et al. 2019 Snowflake-to-Avalanche).
- `geth/` and `coreth/` upstream forks: KEEP own provenance (these are not Lux-original; brand transparently).
- `kms/vendor/` MinIO Snowball: KEEP (unrelated AWS).
- `node/vms/platformvm/txs/zap_native/`: 15 "Subnet" identifiers in `ConvertNetworkToL1Tx` / `TransformChainTx` / `chains_list.go` are forward-compat shims for the subnet→L1 migration. KEEP but document as "legacy network compatibility, will not appear on the post-Quasar-Edition wire."
- External docs site / blog / SDK READMEs: SCRUB. Out of scope for LP-200; dispatch separate doc task.

6 Conclusion

The Lux substrate of June 2026 is no longer a fork of Avalanche in any meaningful technical sense. Approximately 54% of the `node/` surface area is Lux-original (vs upstream `avalanchego`), and the consensus engine, wire protocol, post-quantum suite, GPU/C++ execution layer, key management, and precompile registry contain zero upstream lineage. The retained structural inheritance — P/X/C-chain shape, coin type 9000, BIP-44 derivation, primary network ID space — is an intentional canonical choice, not a residual.

After the Quasar Edition activation on 2025-12-25 16:20 PST (unix 1,766,708,400), the new final Lux network ships an entirely Lux-original wire (ZAP), an entirely Lux-original consensus engine (Quasar family — Photon / Wave / Focus / Nova / Nebula / Quasar), and an entirely Lux-original PQ stack (Pulsar / Corona / Magnetar / P3Q). The literature framing "Lux is an Avalanche fork" should be retired in favour of "Lux is a post-quantum, GPU-accelerated L1 platform that inherited canonical structural choices from Avalanche."

The state of the art positions Lux ahead on five measurable axes: zero-copy wire decode (11× vs protobuf), GPU-accelerated EVM (∼30–800× vs go-ethereum and reth), post-quantum threshold signing (−51% vs the audited `cloudflare/circl` baseline), cryptographic precompile depth (47 families / 99 slots vs Ethereum's 11), and sub-second classical-path finality with PQ fallback (no published competitor offers both). Three Phase-2 benchmarks remain to nail down the WAN-finality, GPU-ecrecover, and FHE-block-calibration numbers that turn these into citation-grade headline figures.

Reproduction

All numerical claims in this paper are recoverable from the cited sources:

- Inventory LOC: `find /work/lux/<pkg> -name '*.go' -exec wc -l {} +` and `find /work/luxcpp/<p -name '*.cpp' -o -name '*.cu' -o -name '*.h' ... -exec wc -l {} +`.
- Avalanche scrub verification: `grep -r 'ava-labs|avalanchego' ~/work/lux` (returns only 3 string-comment references in `netrunner/` and `state/cmd/`; zero production imports).
- ZAP P-chain Parse speedups (v3.7):
  `cd ~/work/lux/node/vms/platformvm/txs/zap_native`
  `GOWORK=off go test -bench=BenchmarkParse_ -benchmem -benchtime=2s -count=1`
  canonical output: `~/work/lux/node/vms/platformvm/txs/bench_results/RESULTS.md`.

- Quasar / BLS / ML-DSA microbenchmarks: see `consensus/BENCHMARKS.md` for per-test reproduction commands.
- Magnetar speedup: `cd /work/lux/magnetar && ./scripts/bench.sh`.
- EVMgpu numbers: `papers/evmgpu-benchmark/sections/` cite the harness in `evmgpu/core/bench_test.g`

Sources

1. Lux Network. "LP-073 — Pulsar consensus." `~/work/lux/papers/lp-073-pulsar/`.
2. Lux Network. "LP-023 — P-chain wire is ZAP." `lps/LP-023-pchain-zap-wire.md` (Appendix A, 8.5× v3.6 geomean) plus refreshed batch 5 v3.7 results at `node/vms/platformvm/txs/bench_resu` (canonical 7.50× Parse, 3.57× alloc reduction, 1 alloc/24 B uniform; refreshed 2026-06-02 on luxfi/zap v0.7.2 / Go 1.26.1 / Apple M1 Max).
3. Lux Network. "Consensus measured benchmarks." `~/work/lux/consensus/BENCHMARKS.md`. Apple M1 Max, Go 1.26.1.
4. Lux Network. "Magnetar CHANGELOG §1.1.0." `magnetar/CHANGELOG.md`. Strict-atom −51% vs `cloudflare/circl sign/slhdsa.SignDeterministic`.
5. Lux Network. "ZAP benchmarks paper." `~/work/lux/papers/lux-zap-benchmarks/`. ZAP vs FlatBuffers vs Cap'n Proto vs gRPC vs JSON-RPC.
6. Lux Network. "EVMgpu benchmark paper." `~/work/lux/papers/evmgpu-benchmark/`. 20.9 Ggas/s C++ EVM; 87% ecrecover bottleneck.
7. Lux Network. "Quasar consensus benchmarks paper." `~/work/lux/papers/lux-quasar-benchmarks/`. Literature-backed comparison with Mysticeti, HotStuff-2, CometBFT, Avalanche Snowman, Solana TowerBFT.
8. Lux Network. "Quasar consensus README." `~/work/lux/consensus/README.md`. PQ mode storage / cert size table.
9. Rocket, T. et al. "Avalanche." 2019. https://arxiv.org/abs/1906.08936 — baseline for the Snowflake-to-Avalanche family Lux's consensus replaced.
10. Babel, K. et al. "Mysticeti." 2024 — WAN-distributed BFT baseline for Quasar comparison.
11. Yin, M. et al. "HotStuff." PODC 2019.
12. Malkhi, D. et al. "HotStuff-2." 2023.
13. Paradigm. "Reth at one million Mgas/s." April 2024.
14. NIST FIPS 204 (ML-DSA) and FIPS 205 (SLH-DSA), 2024.
15. Upstream Avalanche reference: `ava-labs/avalanchego` at commit `9f8b9461` (2026-05-14, master); local mirror `~/work/ava/avalanchego`.

Table 1: Lux Go workspace, by package origin (selected packages). Divergence percent is structural: 0 = unchanged vendored, 100 = Lux-original with no upstream lineage. Forked-diverged percentages are estimated from structural inspection of subdirectory layouts vs upstream when an analog exists.

Package	**Origin**	**Lux LOC**	**Divergence**
Forked / shape-inherited			
node	forked, diverged	288,224	∼54% Lux-original surface
evm	forked, diverged	141,334	∼70% Lux-original
warp	forked, diverged	2,999	∼50% Lux-original (PQ envelope)
ids	forked, diverged	12,569	∼30% (BTC-style NodeID add)
netrunner	forked, diverged	36,654	∼50% Lux delta
geth	forked, pinned	371,717	∼5% Lux delta
coreth	forked, vestigial	144,122	∼5% Lux delta
Extracted from snow/ and rewritten			
consensus	extracted, rewritten	98,844	100% (no snowball/snowman)
Lux-original (no upstream ancestor)			
pulsar (PQ threshold)	Lux-original	21,480	100%
corona (Round-Signer)	Lux-original	24,810	100%
magnetar (SLH-DSA)	Lux-original	10,284	100%
quasar (spec/meta)	Lux-original	–	100%
p3q (Rust crate)	Lux-original	5,555	100%
age (PQ-extended)	Lux-original	10,777	100%
zap (wire)	Lux-original	24,452	100%
zapdb (KV)	Lux-original	38,799	100% (Badger-foundation rewritten)
crypto	Lux-original	65,855	100%
lattice (RLWE/FHE)	Lux-original	56,939	100%
precompile (47 families)	Lux-original	79,251	100% (99 addressable slots)
evmgpu	Lux-original	142,998	100%
vm (SDK)	Lux-original surface	29,906	100%
fhe (runtime)	Lux-original	16,125	100%
kms (`lux/kms`)	Lux-original	12,593	100%
kms (`hanzo/kms`)	Lux-original	38,982	100%
keys (BIP-44)	Lux-original	3,717	100%
threshold	Lux-original	75,406	100%
mpc (CGGMP21+FROST)	Lux-original	80,110	100%
utxo (fx plugins)	Lux-original	16,098	100%
bft	Lux-original	12,819	100%
sdk (Go)	Lux-original	51,920	100%
bridge	Lux-original	55,127	100%
cli (`lux` binary)	Lux-original	110,023	100%

Table 2: Lux C++/GPU workspace at `~/work/luxcpp/`. No Avalanche upstream exists — Avalanche has no native C++/CUDA/Metal execution path. All packages here are Lux-original except the three vendored references (PQClean, intx, evmc).

Package	**Role**	**LOC**
cevm	C++ EVM (evmone++ fork, rewritten)	669,382
fhe	Native FHE engine (CKKS/BFV/BGV)	663,693
pqclean (vendored)	FIPS reference impls	348,846
gpu	GPU primitives abstraction	245,154
webgpu	WebGPU backend	138,138
crypto	C++ crypto primitives	133,926
metal	Apple Metal kernels	118,494
zap-cpp-core	C++ ZAP wire	88,589
cuda	CUDA kernels	85,773
evmmax	EVM-MAX precompiles	37,066
dex	DEX matching engine	33,068
xvm	X-chain native VM (C++)	22,307
bridgevm	Cross-chain bridge VM	21,455
session	QUIC session layer	20,214
platformvm	P-chain port (C++)	19,494
aivm	AI VM (inference precompiles)	17,436
mpcvm	MPC threshold VM	16,540
evmc (vendored)	EVM-C ABI	9,456
lattice	Native lattice primitives	6,519
intx (vendored)	256-bit integer arithmetic	6,185
consensus	C++ consensus bindings	5,642
zapdb	C++ ZapDB wrapper	3,877
accel	Backend dispatch	2,856

Table 3: Consensus and protocol terminology

Upstream Avalanche	**Lux native**	**Status**
Slush / Snowflake / Snowball / Avalanche consensus	Photon (committee sampling) + Wave (FPC) + Focus (confidence) + Ray (linear driver) + Field (DAG driver) + Quasar (threshold seal)	replaced
Snowman (linear consensus)	Nova (Photon+Wave+Focus+Ray composition)	replaced
Avalanche (DAG consensus)	Nebula (Photon+Wave+Focus+Field composition); Prism for DAG geometry; Horizon for order-theory; Flare for skip-detection	replaced
Etna upgrade (subnet → L1 conversion)	Quasar Edition (network activation 2025-12-25 16:20 PST, unix 1,766,708,400)	renamed and reshaped
ACP (Avalanche Community Proposal)	LP (Lux Proposal); see `luxfi/lps`	renamed
Subnet	L1 (sovereign chain with own validator set) or L2 EVM (shared primary-network validators)	restructured
Primary network (P/X/C)	Primary network (P + X only); C-Chain is one EVM L2 among many (Hanzo / Zoo / SPC). C-Chain disabled via `LUX_DISABLE_CCHAIN=1` when bootstrapping non-Lux L1s.	restructured
P-Chain (platform)	P-Chain (Lux). Wire is ZAP; no linearcodec on the new final network (see LP-023).	restructured
X-Chain (asset / DAG)	X-Chain (Lux). UTXO + DAG; uses Nebula+Field consensus.	inherited shape
C-Chain (EVM)	C-Chain (Lux primary-network EVM) only. Hanzo/Zoo/SPC/Pars all have their own EVM L1s/L2s; never C-Chain.	narrowed scope
Simplex	not adopted; Lux uses Quasar family	N/A
SaeVM	not adopted; Lux ships `vms/da` for data availability	N/A

Table 4: Identity, network, and address terminology

Upstream Avalanche	Lux native	Status
`Avalanche mainnet` (network ID 1)	`Lux mainnet` (network ID 1, shared across all Lux L1s — Hanzo, Zoo, Pars, Liquidity, etc.)	ID inherited
`Fuji` testnet (network ID 5)	`testnet` (network ID 2)	renamed, ID changed
`local` (network ID 12345)	`devnet` (3) / `local` (1337)	renamed, IDs changed
`AVAX` token	`LUX` token. Bridge-destination exception: when the Lux bridge orchestrator delivers a wrapped asset *onto* Avalanche C-Chain or X-Chain, the user-facing label remains `AVAX` (and `avax` HRP) on the destination side — the bridge does not re-brand the upstream network. This is the only context where `AVAX` is correct in Lux user surfaces.	renamed
`avax` HRP (`X-avax1...`)	`lux` / `test` / `dev` / `local` / `custom` HRPs (`X-lux1...`)	renamed
SLIP-0044 coin type 9000	SLIP-0044 coin type 9000 (BIP-44 path `m/44'/9000'/`*`serviceIndex`*`'/0'/0'`)	**inherited** (canonical)
NodeID format (`NodeID-<base58>`, 20-byte SHA-256 hash of public key)	BTC-style hash-of-hash NodeID: `SHAKE256-384("NODE_ID_V1" ‖ chainID ‖ 0x42 ‖ pk)[:20]` (HIP-0077)	replaced
BLS PoP (proof of possession)	BLS PoP + ML-DSA hybrid (X-Wing-style)	extended

Table 5: Wire, codec, and SDK terminology

Upstream Avalanche	Lux native	Status
`linearcodec` / `reflectcodec`	ZAP wire (see LP-022, LP-023); the Go struct *is* the wire; no marshal step	replaced
P-chain wire (linearcodec)	ZAP-native P-chain wire (`node/vms/platformvm/txs/zap_native`)	replaced
gRPC consensus messages	ZAP over QUIC (handshake hybrid X-Wing, payload zero-copy)	replaced
`AvalancheJS`	`@luxfi/sdk` (TS / JS); also `luxfi/sdk` (Go), `luxfi/sdk-rs` (Rust), `luxfi/python-sdk`	replaced (multi-language)
`Core Wallet`	Lux Wallet (whitelabelable; Hanzo / Zoo / Pars / Liquidity each ship branded distributions)	replaced
`avalanche-cli`	`luxfi/cli` (`lux` binary)	replaced
`coreth` (EVM client)	`luxfi/geth` fork (pinned at v1.16.x; minimal diff vs upstream geth); `coreth` retained as compat shim only	replaced upstream
`avalanche-network-runner`	`luxfi/netrunner` + `luxfi/netrunner-sdk`	forked, diverged
HyperSDK	not adopted; Lux exposes execution as Quasar-wrapped VMs (`vm/` + `precompile/`)	N/A

Table 6: Post-quantum and crypto terminology (Lux-original surface)

Upstream Avalanche	Lux native	Status
(none — pre-quantum)	Pulsar (FIPS 204 ML-DSA-65 threshold; 2-round commit-and-reveal)	Lux-original
(none)	Magnetar (FIPS 205 SLH-DSA threshold; THBS-SE strict-atom)	Lux-original
(none)	Corona (Round-Signer composition; DKG2 with PVSS dealerless DKG)	Lux-original
(none)	Quasar (BLS + Pulsar + ML-DSA in parallel; PQMode selector)	Lux-original
(none)	P3Q precompile slot `0x012205` (hybrid post-quantum verify)	Lux-original
(none)	ML-KEM-1024 (FIPS 203) / X-Wing KEM for handshakes	Lux-original
(none)	FHE precompile suite (CKKS, BFV, BGV); FHE coprocessor	Lux-original
(none)	47 primitive precompile families across 99 addressable slots: ML-DSA / SLH-DSA / ML-KEM / Pulsar / Quasar / FROST / CGGMP21 / FHE / HQC / KZG4844 / Poseidon / Pasta / Pedersen / VRF / X25519 / Curve25519 / BabyJubjub / BLS12-381 / Ring / ZK / Solidity-bindings. Ethereum has 11 post-Pectra; Solana has ∼60 native programs (different model).	Lux-original

Table 7: ZAP-native P-chain transaction Parse speedup vs upstream linearcodec, batch 5 v3.7 (refreshed 2026-06-02). Source: `node/vms/platformvm/txs/bench_results/RESULTS.md`. Reproduction: `GOWORK=off go test -bench=BenchmarkParse_ -benchmem -benchtime=2s -count=1 ./vms/platformvm/txs/zap_native/`. Lower ns/op is better. **Geometric mean across nine measurable tx types:** 7.50× **Parse,** 3.57× **allocation reduction (3–7 allocs/120–536 B → 1 alloc/24 B uniformly).** Build geomean 1.03×, within v3.6 noise; 0.77–0.90× dips on `BaseTx` and `SetL1ValidatorWeightTx` are tracked as a luxfi/zap v0.7.x follow-up (`Builder.SetU64` per-call dispatch overhead, not a wire-layer regression).

Tx Type	Legacy ns/op	ZAP ns/op	Speedup	Legacy allocs/B	ZAP allocs/B
RewardValidatorTx	213.6	36.91	5.79×	3/120	1/24
SetL1ValidatorWeightTx	423.4	52.30	8.10×	3/136	1/24
IncreaseL1ValidatorBalanceTx	167.3	33.45	5.00×	3/136	1/24
DisableL1ValidatorTx	177.9	27.09	6.57×	3/120	1/24
BaseTx	334.3	49.67	6.73×	3/152	1/24
RegisterL1ValidatorTx	765.1	50.98	15.01×	6/384	1/24
SlashValidatorTx	356.8	45.35	7.87×	4/176	1/24
TransferChainOwnershipTx	418.0	53.13	7.87×	4/192	1/24
RemoveChainValidatorTx	343.3	44.01	7.80×	4/176	1/24
Geomean (n=9)			**7.50×**		

Table 8: Decode throughput of ZAP wire vs industry wire protocols on a 145-byte consensus vote payload. Source: `papers/lux-zap-benchmarks/sections/03-serialization-throughput.tex`.

Protocol	ops/sec	ns/op	allocs/op	B/op
ZAP	156,000,000	6.4	0	0
FlatBuffers	142,000,000	7.0	0	0
Cap'n Proto	89,000,000	11.2	0	0
gRPC (protobuf)	14,200,000	70.4	3	192
QUIC (raw)	11,800,000	84.7	2	160
Protobuf-mux	10,100,000	99.0	5	320
JSON-RPC	1,180,000	847.5	28	2,048

Table 9: Cryptographic signing primitives, Apple M1 Max CPU path. Source: `consensus/BENCHMARKS.md`.

Operation	KeyGen	Sign	Verify
BLS12-381	1.06 µs	350 µs	820 µs
ML-DSA-44 (FIPS 204 Cat 2)	175 µs	1.36 ms	245 µs
ML-DSA-65 (FIPS 204 Cat 3)	343 µs	2.63 ms	398 µs
ML-DSA-87 (FIPS 204 Cat 5)	461 µs	3.29 ms	517 µs
SLH-192f (FIPS 205 fast)	—	63 ms (THBS)	—
SLH-192s (FIPS 205 small)	—	1.43 s (THBS)	—
SLH-256s (FIPS 205 small)	—	2.04 s (THBS)	—
secp256k1	—	—	658 ns

Table 10: Magnetar v1.1 strict-atom THBS-SE vs v1.0-equivalent baseline (circl-dispatched SignDeterministic). Source: `magnetar/CHANGELOG.md §1.1.0`.

Mode	v1.0-equivalent	v1.1 strict-atom	Delta
Magnetar-SHAKE-192s	2.93 s/op	1.43 s/op	−51%
Magnetar-SHAKE-192f	113 ms/op	63 ms/op	−44%
Magnetar-SHAKE-256s	2.22 s/op	2.04 s/op	−8%

Table 11: Quasar protocol benchmarks (Apple M1 Max, single-goroutine). Source: `consensus/BENCHMARKS.md §5`.

Operation	ns/op (ops/sec)
QuasarSig verify (msg = 512 B)	1,017,823 (982)
QuasarSig sign (msg = 1024 B)	808,500 (1,237)
QuasarSig verify (msg = 1024 B)	1,089,206 (918)
BLS aggregation of 4 sigs	209,808 (4,766)
BLS aggregation of 100 sigs	5,297,844 (189)
BLS aggregated verify, 100 signers	875,227 (1,142) — *constant*
QuasarBlockProcessing	1,849,772 (540)
QuasarQuantumHash	434.7 (2,300,000)
QuasarValidatorAddition	327,629 (3,052)

Table 12: Storage and certificate-size cost of each consensus signing mode (100 validators, 21 committee, 5 k cert horizon). Source: `consensus/README.md`.

Mode	Sign	Aggregate	Verify	Cert	Storage 10K
`bls`	312 µs	8.6 ms	714 µs	123 B	1.17 MB
`bls-mldsa`	369 µs	8.5 ms	3.4 ms	69 KB	665 MB
`bls-rt`	39 ms	3.3 s	1.6 ms	33 KB	318 MB
`triple`	40 ms	3.3 s	4.3 ms	102 KB	981 MB

Table 13: EVM execution throughput. Lux C++ EVM data from `papers/evmgpu-benchmark`; competitor data from cited references (Paradigm/reth April 2024 post, geth live-sync averages).

Engine	Path	Workload	Throughput
Lux C++ evmone+state	CPU	full block	**20.9 Ggas/s**
Lux Go EVM (baseline)	CPU	full block	4.2–5.4 Ggas/s
Lux + parallel ecrecover (10c)	CPU	block w/ sigs	7.1× over 1-core
go-ethereum (latest)	CPU	live mainnet sync	∼25–50 Mgas/s
reth (Paradigm)	CPU	live mainnet sync	100–200 Mgas/s
reth (Paradigm)	CPU	historical opBNB sync	690 Mgas/s
Lux C++ evmone interp. only	CPU	interp. only	∼416 Ggas/s
Block-STM (Aptos)	32-thread CPU	Aptos workload	up to 170k TPS

Table 14: Published consensus latency / throughput points. The Quasar single-DC 10 ms / 4.76M TPS claim assumes the Lux 1 B gas block limit (47,619 txs/block at 21 k gas per transfer). Source: `papers/lux-quasar-benchmarks/sections/15-literature-backed-comparison.tex`.

Protocol	Validators	Finality (ms)	TPS	Msg / commit
Quasar (single-DC, BLS)	21	10	4,761,904	$O(n)$
Quasar (CPU-BLS, WAN)	21	200	TBM	$O(n)$
Mysticeti-C	10	500	>200,000	$O(n)$
Mysticeti-C	50	<1,000	∼400,000	$O(n)$
Mysticeti-C	106	390 (p50)	—	$O(n)$
HotStuff (orig.)	128	∼700	∼50,000	$O(n)$
HotStuff-2	100	∼400	—	$O(n)$
Tendermint / CometBFT	100	1,000–3,000	∼10,000	$O(n^2)$
Avalanche Snowman	2,000	1,350	3,400	$O(k)$ sampling
Solana TowerBFT	1,500	12,800 (econ.)	65,000 claimed	$O(n)$