PRISM-1R

Implementation and Deployment Blueprint v0.2

Packet-programmable mixed-representation accelerator
Transformation plane + resolution plane + certificate-gated commit

---

Baseline	PRISM-1 Final Updated architecture specification
Revision	PRISM-1R physicalization and deployment blueprint
Deployment target	Golden model → CUDA → cycle simulator → FPGA → shuttle chiplet → reference chiplet → full package
Primary rule	Preserve structure. Execute locally. Refine only where evidence requires it.
Status	Engineering blueprint. Not fabricated silicon. All frequency, bandwidth, area and power values remain design targets until measured.

Komorebi P.

Implementation handoff draft v0.2

Contents

1 Purpose and Deployment Philosophy **4**

2 Reference Configuration **4**

3 Python-Generated Architecture Blueprint **4**

4 What Must Be Frozen Before Production RTL **7**

5 Dual-Plane Execution Model **7**

5.1 Transformation plane . . . 7

5.2 Resolution plane . . . 7

5.3 Generation guard . . . 8

6 Packet Streaming Multiprocessor Blueprint **8**

6.1 Command state machine . . . 8

7 Scratchpad and Memory System **8**

7.1 PSM scratchpad . . . 8

7.2 Bandwidth modeling point . . . 9

7.3 Hierarchy . . . 10

8 Chiplet Quadrant Organization **10**

9 Network-on-Chip Blueprint **10**

9.1 Residency states . . . 10

10 Certificate, Commit, and Replay **11**

10.1 Fused accumulation . . . 11

10.2 Hierarchical aggregation . . . 12

10.3 Commit predicate . . . 12

11 Packet IR, Compiler, and Runtime Contract **12**

11.1 Planner objective . . . 12

11.2 Software layers . . . 13

12 Command and Descriptor ABI **13**

12.1 Candidate 256-bit Packet Seed Command envelope . . . 13

12.2 Candidate packet descriptor packing . . . 14

12.3 Minimum descriptor classes . . . 14

12.4 Minimum first-generation commands . . . 15

13 Deployment Ladder and Exit Gates **15**

14 Stage 0: Golden Model **16**

15 Stage 1: CUDA Packet Backend **16**

16 Stage 2: Cycle-Aware Simulator **16**

17 Stage 3: FPGA PSM Blueprint **17**

17.1 Required blocks 17

17.2 FPGA acceptance tests 17

18 Stages 4-6: ASIC Development Path **17**

18.1 Stage 4: shuttle chiplet 17

18.2 Stage 5: reference chiplet 18

18.3 Stage 6: full package 18

19 Verification and Signoff Matrix **18**

20 Benchmark Suite and Required Metrics **18**

21 Deployment Runbook **19**

22 Failure Policy **19**

23 RTL Partition and Interface Blueprint **19**

23.1 Recommended ready/valid discipline 20

24 PSM Pipeline and Scheduling Contract **20**

24.1 Scoreboard entries 20

25 Scratchpad Bank Contract **20**

26 Formal Safety Invariants **21**

27 Boot, Discovery, and Runtime Bring-Up **21**

28 Telemetry and Performance-Counter Minimum Set **22**

29 Implementation Handoff Package **22**

30 Architecture Freeze Checklist **22**

31 First 30 Engineering Tasks **23**

32 Final Deployment Rule **23**

1. Purpose and Deployment Philosophy

PRISM-1R converts the existing PRISM architecture into an implementation program. The objective is not to freeze every circuit choice prematurely. The objective is to freeze the *semantic contracts* that software, simulation, FPGA and ASIC implementations must share, while keeping physical parameters adjustable until measurement and synthesis justify them.

The architecture is organized around one principle:

> Do not materialize, move, or schedule structure already declared inactive.

PRISM-1R adds a second principle:

> Do not execute at a resolution that current evidence has already invalidated.

The machine therefore chooses four quantities jointly:

$$(\text{representation},\ \text{resolution},\ \text{placement},\ \text{precision}).$$

A dense tensor remains a legal representation and fallback. It is not privileged as mandatory machine state.

Scope boundary. This document specifies interfaces, resource organization, deployment stages, verification obligations, prototype gates, and operating procedures. It does not claim completed production RTL, timing closure, physical signoff, fabricated silicon performance, or measured package power.

2. Reference Configuration

The reference package preserves the broad PRISM-1 scale while revising locality and control organization.

Dense peak remains secondary. The architecture succeeds only when mathematical structure removes enough arithmetic, storage, synchronization, or movement to justify specialized execution.

3. Python-Generated Architecture Blueprint

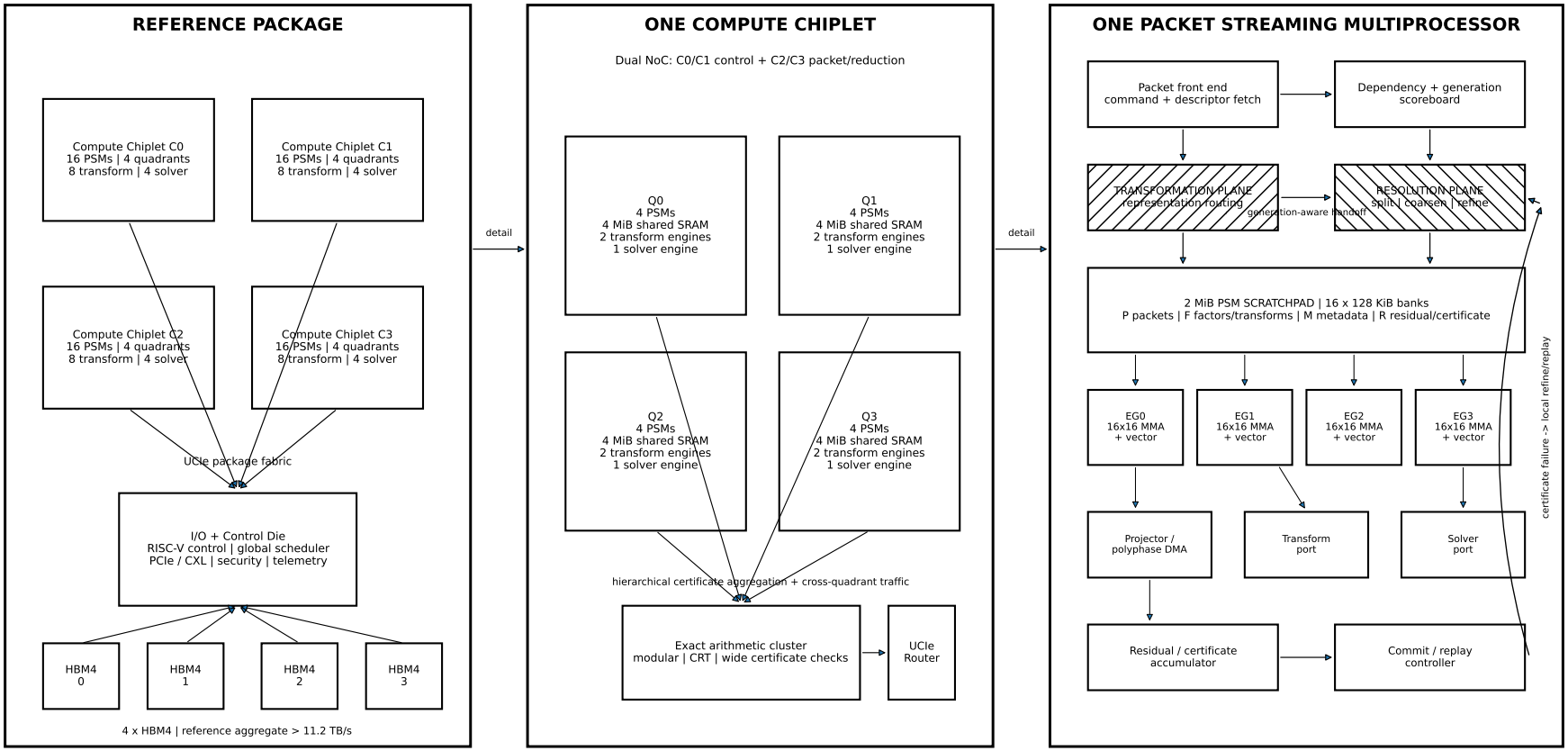
The following blueprint is generated programmatically from the same reference parameters used by this deployment guide. The figure is intended to function as an implementation map rather than a presentation illustration.

Table 1: PRISM-1R reference targets. Values are design targets, not measured silicon.

Resource	**Reference target**
Compute chiplets	4
Packet Streaming Multiprocessors (PSMs)	64 total
PSMs per chiplet	16
Quadrants per chiplet	4
Execution groups	256 total
Execution groups per PSM	4
Matrix array per execution group	16×16
Clock modeling point	1.5 GHz
Scratchpad per PSM	2 MiB
Physical scratchpad banks	16×128 KiB
Shared SRAM per chiplet	16 MiB
Total local + shared SRAM	192 MiB
Transform engines	32 total
Small-solver engines	16 total
Exact arithmetic clusters	4 total
HBM4 stacks	4
Reference HBM capacity	144 GiB total
Reference aggregate HBM bandwidth	> 11.2 TB/s
Package link class	UCIe 3.0 class
Host link class	PCIe 7.0 x16 class, optional CXL
Reference package power envelope	500 W

PRISM-1R ARCHITECTURE BLUEPRINT

Hierarchical implementation view: package -> compute chiplet -> packet streaming multiprocessor



Execution rule: jointly choose representation, resolution, placement and precision

Commit rule: publish only after certificate acceptance and generation validation

Preserve structure | execute locally | refine only where evidence requires it

4. What Must Be Frozen Before Production RTL

Freeze semantics before circuit details. A production RTL effort should not begin until the following objects have versioned, testable definitions.

1. Packet descriptor semantics.
2. Transfer-edge descriptor semantics.
3. Atlas and selector semantics.
4. Refinement-family and parent/child semantics.
5. Representation and output-contract semantics.
6. Dependency token and completion token semantics.
7. Generation-number invalidation semantics.
8. Precision, sensitivity, and error-budget semantics.
9. Certificate acceptance semantics.
10. Commit, replay, and fault-containment semantics.
11. Command-buffer ABI and completion-record ABI.
12. Packet IR canonical serialization.
13. CUDA/PRISM backend conformance requirements.

The following should remain parameterized until synthesis, FPGA measurement, or floorplanning provides evidence: scratchpad port width, final SRAM macro organization, issue width, exact NoC link width, number of virtual channels beyond the minimum classes, DVFS domains, exact root-solver microarchitecture, and hardened interface IP choices.

5. Dual-Plane Execution Model

PRISM-1R separates two control problems that were previously too implicit.

5.1 Transformation plane

The transformation plane performs computation within the current representation and resolution:

$$P_\alpha x \longmapsto P_\beta T P_\alpha x.$$

Typical operations include block matrix products, low-rank contractions, transform stages, Kronecker mode products, sparse-chain applications, modular operations, nonlinear memory kernels, reductions, and small solves.

5.2 Resolution plane

The resolution plane modifies the decomposition itself:

$$P_\alpha \longleftrightarrow \{P_{\alpha_1}, \ldots, P_{\alpha_k}\}.$$

It handles split, coarsen, local refinement, atlas changes, reconstruction changes, and replay boundaries.

5.3 Generation guard

Each mutable packet family carries a generation number g. A command records the generation it consumed. Refinement increments g for the affected family. A provisional result may commit only if

$$\boxed{g_{\text{consumed}} = g_{\text{current}}.}$$

This mechanism converts stale work after adaptive refinement into an explicit hardware predicate rather than a software convention.

6. Packet Streaming Multiprocessor Blueprint

One PSM is the independently scheduled execution unit. The recommended block composition is:

```
Packet front end
  -> dependency / generation scoreboard
  -> representation router -----------+
  -> resolution controller -----------|--- adaptive control
                                      |
16-bank 2 MiB scratchpad <------------+
  -> EG0: 16x16 MMA + vector
  -> EG1: 16x16 MMA + vector
  -> EG2: 16x16 MMA + vector
  -> EG3: 16x16 MMA + vector
  -> projector / polyphase DMA
  -> transform port
  -> solver port
  -> local reduction fabric
  -> residual / certificate accumulator
  -> commit / replay controller
```

Each execution group is intentionally modest. The PSM is not designed to maximize unstructured GEMM density. It is designed to keep the selected representation resident and avoid repeated conversion into a generic dense form.

6.1 Command state machine

A minimal command lifecycle is

$$\texttt{QUEUED} \to \texttt{READY} \to \texttt{DISPATCHED} \to \texttt{PROVISIONAL} \to \texttt{CERTIFIED} \to \texttt{COMMITTED}.$$

Failure paths are explicit:

$$\texttt{PROVISIONAL} \to \texttt{REFINE} \to \texttt{REPLAY}$$

or

$$\texttt{PROVISIONAL} \to \texttt{ABORT}.$$

Kernel completion and architectural visibility are therefore different events.

7. Scratchpad and Memory System

7.1 PSM scratchpad

Keep the 2 MiB capacity but replace the vague four-bank concept with 16 physical banks:

$$16 \times 128 \text{ KiB} = 2 \text{ MiB}.$$

Retain four logical allocator classes:

The classes are compiler-visible pools, not rigid physical partitions.

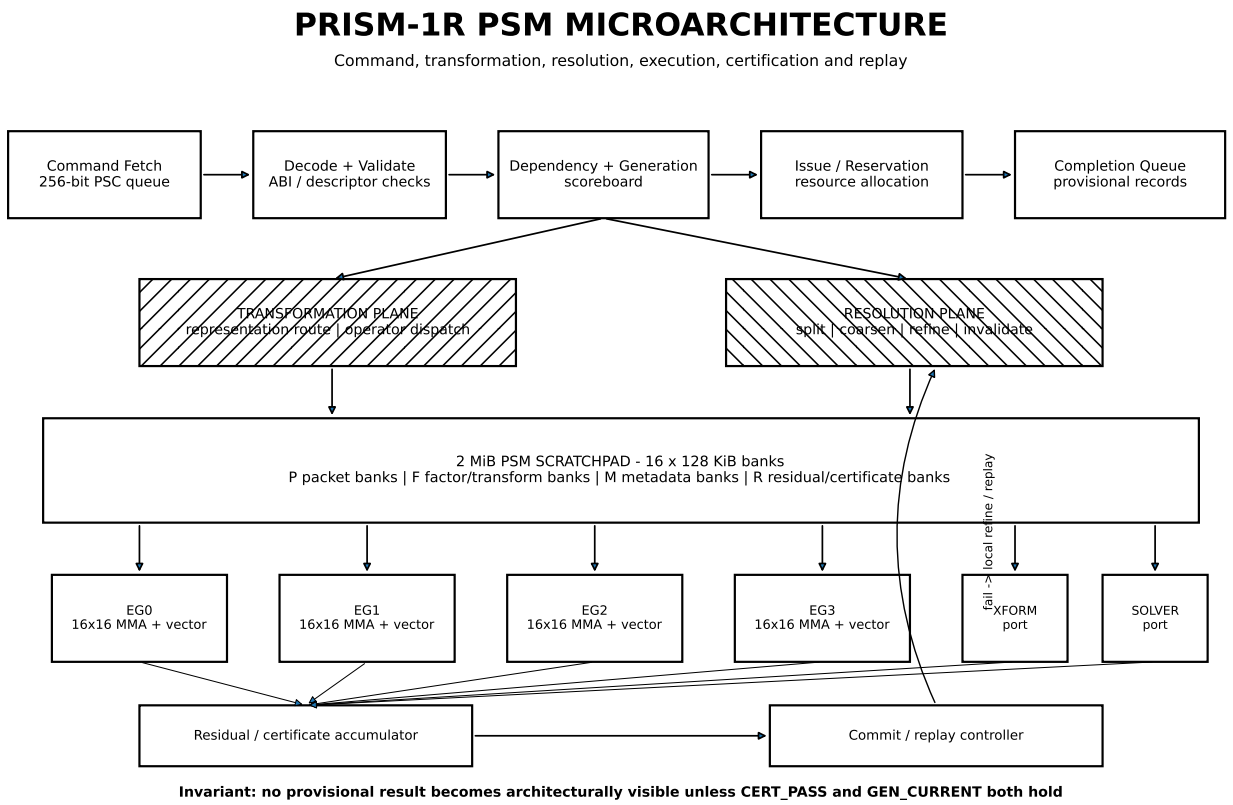


Figure 1: Implementation-oriented PSM microarchitecture. The transformation plane executes within the current decomposition; the resolution plane changes that decomposition and can trigger bounded replay.

Class	Name	Typical contents
P	Packet region	active blocks, packet outputs, local destination accumulators
F	Factor region	low-rank factors, transform stages, Kronecker factors, solver panels
M	Metadata region	descriptors, selectors, indices, dependency state, compact graph records
R	Residual region	residuals, norms, certificate state, replay records, refinement summaries

7.2 Bandwidth modeling point

If each of 16 banks supplies a 256-bit word per cycle, the aggregate raw bandwidth is

$$16 \times 32 \text{ B/cycle} = 512 \text{ B/cycle}.$$

At 1.5 GHz this is

$$\boxed{768 \text{ GB/s per PSM raw scratchpad bandwidth.}}$$

A simple FP16 feed estimate for one 16×16 array is 32 FP16 operands per active cycle, or 64 B/cycle. Four execution groups therefore require roughly

$$4 \times 64 = 256 \text{ B/cycle}$$

for direct operand feed before output traffic and reuse effects, corresponding to approximately 384 GB/s at 1.5 GHz. The proposed bank organization therefore begins with roughly 2x raw headroom against that simplified feed estimate. This is a modeling target, not a timing guarantee.

7.3 Hierarchy

- HBM: exact archive, cold factors, large graphs, model history, oversized packets.
- Chiplet shared SRAM: active packet graph, hot immutable factors, cross-PSM transform state.
- PSM scratchpad: active tiles, factors, solver panels, residuals, local metadata.
- Registers / array state: immediate execution state.

8. Chiplet Quadrant Organization

A compute chiplet is divided into four locality quadrants. Each quadrant contains:

$$\boxed{4 \text{ PSMs} + 4 \text{ MiB shared SRAM} + 2 \text{ transform engines} + 1 \text{ solver.}}$$

This exactly consumes the chiplet-level reference resources while reducing the distance between packet execution, transform stages, and small solves.

The placement objective is not merely minimum latency. It is minimum structured movement:

$$\min \sum_{e \in E} \text{bytes}(e) \times \text{hops}(e).$$

Hot factors should be replicated read-only where the replication cost is lower than repeated network movement. Mutable packet state should retain clear ownership.

9. Network-on-Chip Blueprint

PRISM-1R uses separate logical control and payload service, even if an implementation shares physical wires.

Table 2: Minimum traffic classes.

Class	Traffic	Service rule	Examples
C0	commit / fault	forward-progress guaranteed	certificate acceptance, dependency wakeup, invalidation, exception
C1	metadata	latency-oriented	descriptors, compact graph state, selector records
C2	payload	bandwidth-oriented	blocks, factors, transform streams, packet transfers
C3	reduction	bounded service	residual summaries, tree reductions, reconstruction contributions

C2 saturation must never prevent C0 progress. This is a correctness requirement, not only a performance preference.

An initial packet-link modeling point is 1024 bits at 1.5 GHz:

$$128 \text{ B/cycle} \times 1.5 \text{ GHz} = \boxed{192 \text{ GB/s per unidirectional link}}$$

before encoding, arbitration, flow control, ECC, and protocol overhead.

9.1 Residency states

Avoid a CPU-style general coherence protocol. Use three architectural residency states:

owned one location may mutate the packet or representation state;

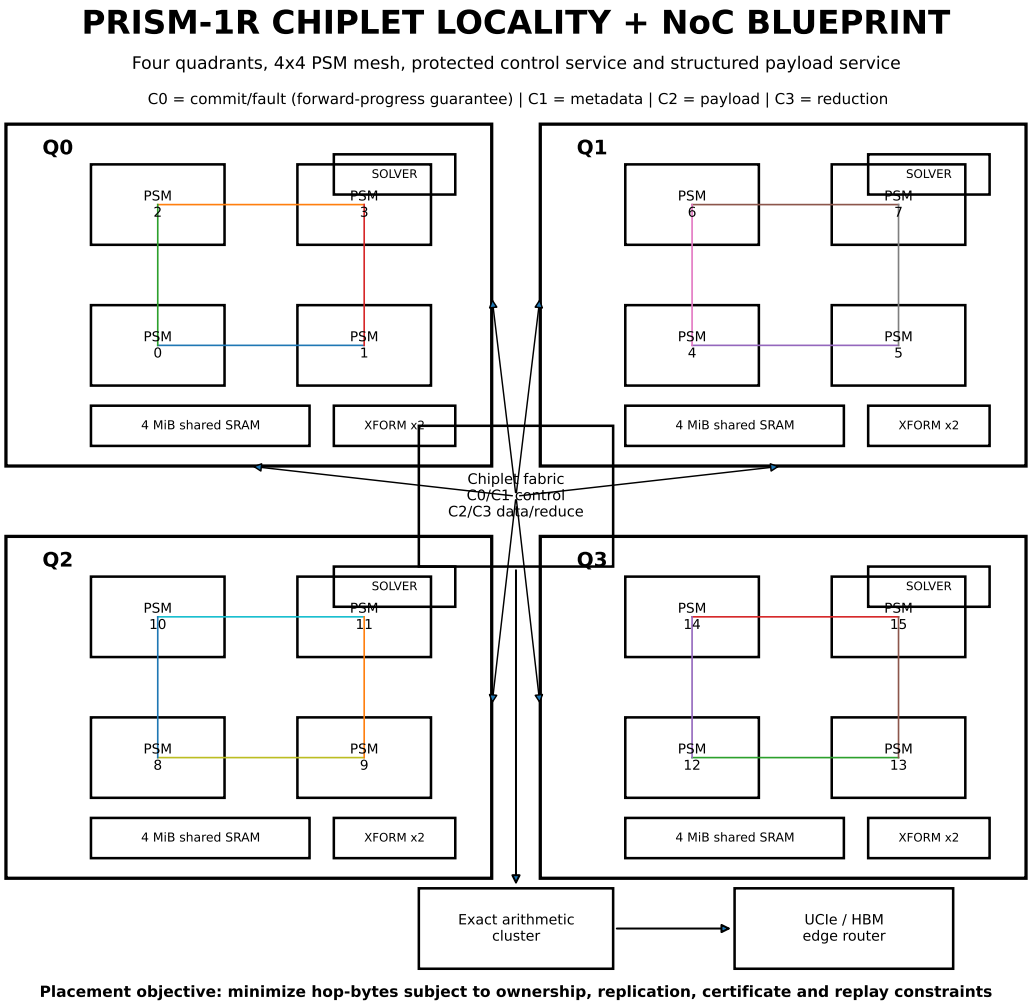


Figure 2: Chiplet locality and NoC organization. Quadrant locality is the default placement unit, while cross-quadrant traffic remains explicit and measurable.

replicated-read immutable factors, atlas data, or descriptors may exist at multiple locations;

streaming the payload has no persistent residency and is consumed in flight.

Generation invalidation handles stale read replicas associated with refinement.

10. Certificate, Commit, and Replay

Certification is a first-class data path. A kernel result is provisional until its declared acceptance rule passes.

10.1 Fused accumulation

Do not implement certification as a second full read of the result. Execution should tap quantities such as residual components, norms, energy partitions, trace summaries, rank margins, rounding bounds, and branch margins while the producing data is already live.

10.2 Hierarchical aggregation

Use the hierarchy

$$\text{execution group} \rightarrow \text{PSM} \rightarrow \text{quadrant} \rightarrow \text{chiplet} \rightarrow \text{exact cluster when required.}$$

For example, local squared residual summaries may reduce as

$$R_P^2 = \sum_i r_i^2$$

without transmitting the original packet tensor through the certificate network.

10.3 Commit predicate

A provisional result commits only if all required conditions hold:

$$\boxed{\texttt{CERT_PASS} \wedge \texttt{GEN_CURRENT} \wedge \texttt{NO_UNCORRECTABLE_FAULT} \wedge \texttt{DEPENDENCIES_VALID}.}$$

A failure should invalidate the smallest declared replay region unless the error is explicitly package-wide.

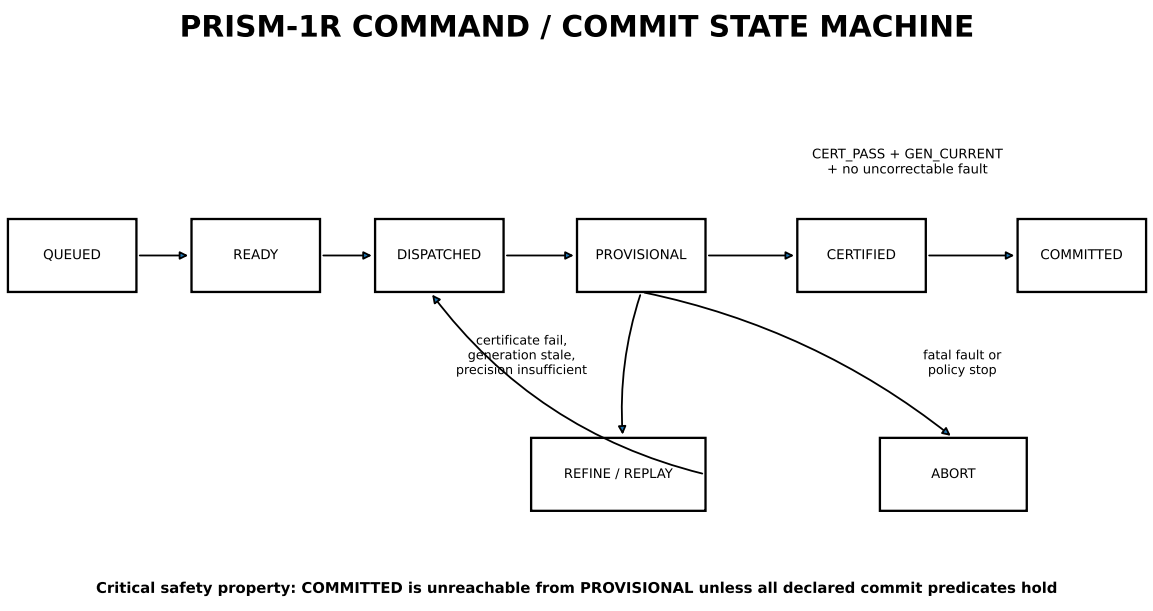


Figure 3: Command and commit state machine. The key safety property is that no direct path exists from a provisional result to committed state without the declared certificate and generation checks.

11. Packet IR, Compiler, and Runtime Contract

Packet IR is the stable software/hardware boundary. CUDA and PRISM backends consume the same typed graph and must preserve the same mathematical output contract.

11.1 Planner objective

The PRISM-1R planner should minimize a physicalized cost rather than arithmetic alone:

$$C(G) = \alpha F + \beta B_{\text{HBM}} + \gamma B_{\text{SRAM}} + \delta H_{\text{NoC}} + \eta L_{\text{sync}} + \kappa C_{\text{cert}} + \rho\, \mathbb{E}[C_{\text{replay}}].$$

Here F is arithmetic work, B_{HBM} and B_{SRAM} are bytes moved at each level, H_{NoC} is hop-bytes, L_{sync} is synchronization cost, C_{cert} is certificate cost, and the final term accounts for expected replay after adaptive refinement.

A decomposition is not beneficial merely because it has a structured name. The compiler must include a generic fallback leaf and negative controls.

11.2 Software layers

1. Mathematical reference kernels and conformance vectors.
2. Packet IR builder, validator, serializer, and graph visualizer.
3. Representation/resolution/placement/precision planner.
4. CUDA packet backend.
5. PRISM functional simulator.
6. PRISM cycle-aware simulator.
7. Runtime queues, allocator, residency manager, telemetry, and driver interface.
8. FPGA firmware and hardware driver.
9. ASIC firmware and package runtime.

12. Command and Descriptor ABI

Use a fixed command envelope with versioned payload schemas. Opcode semantics should remain stable across CUDA emulation, simulator, FPGA, and ASIC.

12.1 Candidate 256-bit Packet Seed Command envelope

The following packing is a *prototype ABI candidate*, not a frozen physical bit allocation. Its purpose is to prove that command identity, dependency identity, generation identity, and certificate identity fit comfortably in a compact fixed-width envelope.

Table 3: Candidate Packet Seed Command fields.

Field	Bits	Role
opcode	8	operation family
flags	8	deterministic mode, replay policy, privilege and compact modifiers
ABI version	8	command compatibility guard
format	8	arithmetic / payload format class
packet ID	32	primary packet identity
source descriptor	32	source descriptor-table index or handle
destination descriptor	32	destination descriptor-table index or handle
dependency token	24	readiness / wakeup dependency
generation	24	consumed packet-family generation
certificate ID	24	acceptance-rule record
payload or immediate	56	compact immediate, pointer fragment, shape class, or opcode-specific payload
Total	256	fixed command envelope

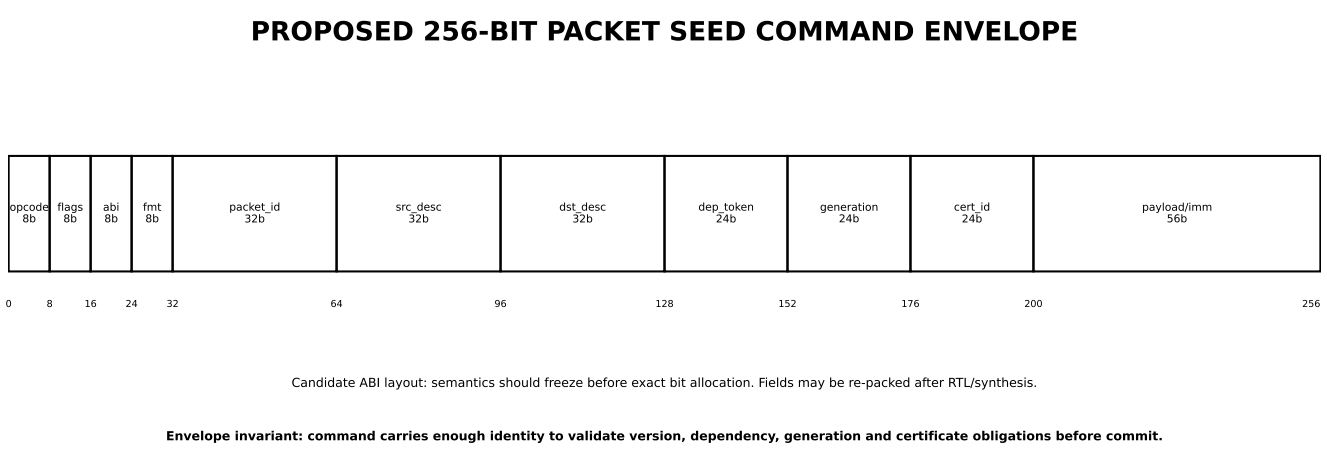


Figure 4: Proposed 256-bit command envelope. Semantic fields should freeze before the final bit positions do.

12.2 Candidate packet descriptor packing

A first 256-bit packet descriptor can carry the following information without external expansion for common cases:

packet ID / parent ID	$32 + 32$ bits
atlas ID / selector ID	$24 + 24$ bits
shape dimensions 0 and 1	$16 + 16$ bits
location or compact pointer	32 bits
precision / error-budget class	$8 + 8$ bits
generation	24 bits
flags	16 bits
reserved / extension class	24 bits
Total	256 bits

Larger ranks, higher-dimensional shapes, or wide pointers use extension records rather than silently changing the base record interpretation.

12.3 Minimum descriptor classes

- packet descriptor;
- transfer-edge descriptor;
- atlas descriptor;
- refinement-family descriptor;
- precision / sensitivity / error-budget record;
- dependency and completion token;
- generation record;
- residency record;
- completion record;
- certificate record.

12.4 Minimum first-generation commands

Command	Meaning
`PSELECT`	apply a packet selector without materializing a dense projector
`PSPLIT`	replace a parent packet with a declared child family
`PCOARSEN`	reconstruct a parent from a complete child family
`POLYPHASE`	extract, permute, or interleave residue / phase channels
`WAVELET_LIFT`	execute lifting predict/update/scale stages
`BLK_GEMM`	execute equal-shape block transfer batches
`BDLR_APPLY`	apply block-diagonal-plus-low-rank form without dense materialization
`HANKEL_BUILD`	stream moment packets into a compact Hankel system
`ROOT_PHASE`	perform small root / phase recovery sequence
`GRAPH_REFINE`	replace one parent task with a local child graph
`RESIDUAL`	form / accumulate a declared reconstruction residual
`COMMIT`	publish a provisional result after certificate and generation checks
`GENERIC`	execute a non-specialized compatibility leaf

13. Deployment Ladder and Exit Gates

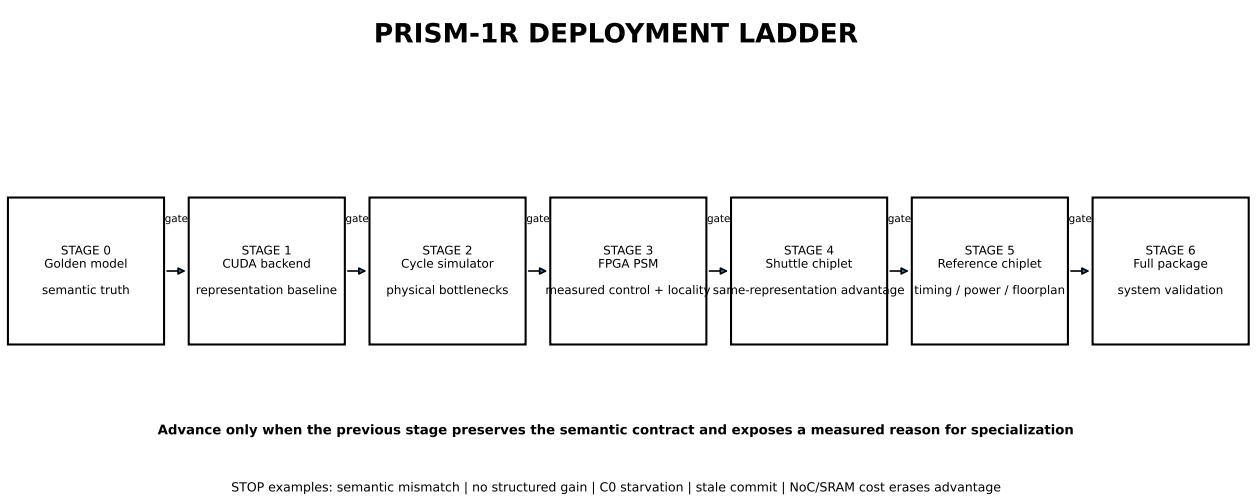


Figure 5: Deployment ladder. Each stage exists to remove a different class of uncertainty before more expensive hardware is attempted.

No stage advances because later hardware is attractive. It advances only when the preceding stage proves the contracts and exposes a measurable reason to continue.

Stage	Artifact	Purpose	Exit gate	Stop condition
0	Golden model	Freeze Packet IR semantics, reference kernels, certificates, replay, deterministic traces	All reference vectors pass; no ambiguous descriptor semantics	semantic mismatch
1	CUDA packet backend	Prove the calculus runs on existing hardware and establish realistic baselines	Structured routes reduce bytes, work, or end-to-end cost on declared workload classes	only naming structure, no real reduction
2	Cycle simulator	Model banks, queueing, NoC, contention, solver / transform occupancy, replay	No deadlock; calibrated bottleneck model; C0 forward progress under C2 saturation	model cannot reproduce measured microbenchmarks

Stage	Artifact	Purpose	Exit gate	Stop condition
3	FPGA PSM	Measure command path, DMA, bank conflicts, generation guard, certificates, local replay	ABI conformance and measured execution of representative packet graphs	semantics change to fit FPGA
4	Shuttle chiplet	Measure locality, hop-bytes, factor residency, transform-to-solver handoff	Stable same-representation advantage over optimized packet GPU for target classes	NoC / SRAM cost erases advantage
5	Reference chiplet	Close chiplet timing, power, SRAM macros, network widths, DVFS	Full chiplet passes functional, timing, power and reliability gates	physical closure requires removing core semantics
6	Full package	Validate multi-chiplet placement, HBM, package fabric, thermals and system software	Package-level performance and reliability goals achieved	package overhead dominates structured gain

14. Stage 0: Golden Model

The golden model is the authoritative semantic implementation. It must be deliberately slow if necessary. Its purpose is correctness and traceability.

Required outputs for every graph execution:

- canonical Packet IR graph hash;
- packet and transfer descriptors after validation;
- chosen representation and resolution;
- expected output contract;
- arithmetic count and byte-count estimates;
- dependency and generation trace;
- provisional outputs;
- certificate inputs and acceptance result;
- replay/refinement events;
- final committed outputs.

The golden model should be the source of conformance vectors used by every later backend.

15. Stage 1: CUDA Packet Backend

The CUDA stage establishes the most important baseline: how far the representation itself can go before custom hardware contributes anything.

Each experiment should separate:

$$S_{\text{representation}}, \quad S_{\text{backend}}, \quad S_{\text{architecture}}, \quad S_{\text{amortized}}.$$

Never report a combined ratio without preserving those components.

Measure grouped block kernels, asynchronous transfers, factor residency, transform libraries, sparse kernels, Woodbury paths, packetwise mixed precision, graph replay, and negative controls.

CUDA gate. Custom hardware is justified only for bottlenecks that remain after the packet representation is already implemented competently on an existing GPU.

16. Stage 2: Cycle-Aware Simulator

The simulator must add physical constraints absent from the golden model:

- 16-bank scratchpad conflicts and queueing;
- execution-group issue / completion timing;
- transform and solver contention;
- C0-C3 network arbitration;

- multicast and tree reductions;
- shared-SRAM residency and eviction;
- dependency wakeups;
- certificate latency;
- generation mismatch and local replay;
- HBM request scheduling;
- thermal / DVFS hooks for later calibration.

The simulator should produce cycle-by-cycle counters and a summarized plan report. Do not silently hide bank conflicts or network queues inside an "effective latency" constant.

17. Stage 3: FPGA PSM Blueprint

The first FPGA target implements one logical PSM. Resource scaling is permitted, semantic scaling is not.

17.1 Required blocks

- command queue and descriptor fetch;
- dependency + generation scoreboard;
- 16-bank or proportionally banked BRAM/URAM scratchpad;
- four logical execution groups or time-multiplexed equivalents;
- projector/polyphase DMA path;
- one transform path;
- one small-solver path;
- residual / certificate accumulator;
- commit / replay controller;
- performance-counter block;
- fault injection for descriptor, generation, and certificate errors.

17.2 FPGA acceptance tests

1. Execute a block-diagonal workload without forming a dense matrix.
2. Execute BDLR apply with resident factors.
3. Execute a transform $\rightarrow$ small solve $\rightarrow$ reconstruction chain without host round-trip.
4. Trigger a certificate failure and verify local replay.
5. Trigger a generation change while work is provisional and prove stale commit is impossible.
6. Saturate payload traffic and demonstrate forward progress of C0 commit/dependency traffic.
7. Compare all committed outputs with golden-model vectors.

18. Stages 4-6: ASIC Development Path

18.1 Stage 4: shuttle chiplet

Use 4-8 PSMs and at least two quadrants. The key measurements are not dense FLOP/s. Measure:

- local versus remote packet access;
- hop-byte distribution;
- factor reuse per load;
- transform-to-solver handoff latency;
- solver occupancy;
- refinement invalidation radius;
- certificate bandwidth;
- energy per committed output.

18.2 Stage 5: reference chiplet

Scale to 16 PSMs and four quadrants. Freeze SRAM macro organization, final physical NoC width, clock domains, DVFS, and floorplan only after evidence from synthesis and the shuttle-scale measurements.

18.3 Stage 6: full package

Integrate four compute chiplets, I/O/control die, HBM, UCIe-class package links, host I/O, system firmware, telemetry, fault containment, and thermal control. The full package is a final integration stage, not the first place to discover whether packet locality works.

19. Verification and Signoff Matrix

Domain	Obligation	Primary method	Pass condition
Semantic	reconstruction, transform/synthesis identity, output contracts	golden model + property tests	no semantic mismatch
Control	dependency tokens, generation invalidation, replay boundary	formal assertions + randomized traces	no stale commit, no deadlock
Arithmetic	mixed precision, exact modes, rank/phase decisions	bit-accurate models	declared certificate bound satisfied
Memory	bank conflicts, allocation, ECC/parity, double buffering	cycle sim + FPGA counters	stress targets met
NoC	traffic classes, multicast, reduction, congestion	adversarial traffic tests	C0 forward progress, no protocol deadlock
Compiler	representation/resolution/placement decisions	differential testing on small graphs	plan never changes semantics
Reliability	malformed commands, corrupted descriptors, replay storms	fault injection	deterministic containment and reporting
Physical	timing, power, thermal, IR drop, CDC/reset	EDA signoff	all process/package limits satisfied

20. Benchmark Suite and Required Metrics

Every benchmark report should expose at least the following quantities.

Metric	Definition / purpose
Bytes eliminated	explicit-array route bytes divided by packet-route bytes
Work eliminated	dense/reference arithmetic divided by executed arithmetic
Packet occupancy	useful execution-group cycles divided by available cycles
Residency reuse	number of factor/atlas uses per memory load
NoC cost	hop-bytes and utilization by C0-C3 traffic class
Refinement locality	remote state invalidated per local refinement event
Certificate overhead	certificate cycles and bytes divided by total execution
Replay cost	useful work discarded or repeated after refinement
Energy	joules per committed output
Fallback quality	fraction of negative controls correctly routed away from specialization

Representative workload families should include structured linear operators, BDLR, regular block sparse graphs, transform-domain operators, unknown-edge reconstruction, small-solver-heavy chains, packet Bayesian portfolios, exact modular/residue workloads, and deliberately unstructured negative controls.

21. Deployment Runbook

1. **Build.** Compile the reference kernels, Packet IR tools, planner, CUDA backend, simulator, firmware, driver, and target image from one tagged release.
2. **Conformance.** Run descriptor schema tests and golden vectors before allowing hardware execution.
3. **Discover.** Query target capabilities: ABI version, PSM count, bank topology, formats, transform/solver inventory, link width, and exact-mode support.
4. **Preload.** Place immutable hot atlases and factors where reuse is expected.
5. **Plan.** Compile representation, resolution, placement, precision, and certificate plan. Emit a human-readable plan report.
6. **Execute.** Submit versioned command buffers and record queue, bank, NoC, execution, certificate, and replay telemetry.
7. **Validate.** Compare committed outputs against exact expectations or reference tolerances. Never consume provisional outputs.
8. **Profile.** Attribute bottlenecks separately to HBM, scratchpad banks, NoC, transforms, solvers, certificates, or replay.
9. **Promote.** Make a specialized route default only after repeatable wins and passing negative controls.
10. **Regress.** Re-run the canonical benchmark and fault-injection suites for every compiler, firmware, RTL, or physical-parameter change.

22. Failure Policy

The deployment process should fail visibly rather than convert uncertainty into optimistic defaults.

Failure	Required response
Unknown descriptor version	reject before execution
Uncorrectable descriptor corruption	abort affected command and invalidate dependent provisional work
Generation mismatch	prevent commit; replay smallest declared region
Certificate failure	refine, increase precision, choose alternate representation, or abort according to policy
C0 starvation	architecture bug; test must fail
Planner chooses unsupported feature	lower to legal fallback or reject; never reinterpret silently
Negative control forced into specialization	compiler quality failure
Backend disagrees with golden semantics	release-blocking error
Timing/power closure requires semantic weakening	stop physical implementation and redesign rather than changing the contract silently

23. RTL Partition and Interface Blueprint

The first RTL release should be deliberately narrow. The objective is to make the safety and scheduling boundaries executable before attempting high-throughput kernels.

```
rtl/
  include/
    prism_types_pkg.sv
  core/
    prism_command_frontend.sv
    prism_descriptor_cache.sv
    prism_dependency_scoreboard.sv
    prism_generation_guard.sv
    prism_representation_router.sv
    prism_resolution_controller.sv
```

```
  prism_scratchpad_banks.sv
  prism_eg_cluster.sv
  prism_transform_port.sv
  prism_solver_port.sv
  prism_certificate_accumulator.sv
  prism_commit_controller.sv
  prism_psm_top.sv
noc/
  prism_control_router.sv
  prism_packet_router.sv
  prism_multicast_unit.sv
  prism_reduction_unit.sv
tb/
  tb_commit_semantics.sv
  tb_generation_race.sv
  tb_c0_forward_progress.sv
  tb_local_replay.sv
```

The accompanying handoff package includes boundary skeletons for the generation guard, commit controller, a PSM top-level shell, and a smoke-test bench. These are semantic scaffolds, not production RTL.

23.1 Recommended ready/valid discipline

All command, descriptor, and data interfaces should use explicit backpressure. A sender may not infer acceptance from elapsed cycles. For every channel:

$$\texttt{transfer} = \texttt{valid} \wedge \texttt{ready}.$$

Metadata and payload channels should carry a transaction identity sufficient to detect mismatched completion records in simulation and debug builds.

24. PSM Pipeline and Scheduling Contract

A practical first implementation can expose the following logical pipeline. Physical retiming may alter cycle counts but must preserve the boundaries.

1. **Fetch**: read fixed-width command envelope.
2. **Validate**: check ABI, opcode, descriptor bounds, privilege, and basic format legality.
3. **Resolve**: fetch packet, transfer, generation, and certificate descriptors.
4. **Ready**: dependency scoreboard confirms all required inputs are eligible.
5. **Reserve**: allocate scratchpad destination, execution route, and completion entry.
6. **Execute**: issue transformation or resolution work to the chosen engine.
7. **Provisional completion**: capture output identity, bytes, cycles, residual summary, and consumed generation.
8. **Commit resolution**: certificate + generation + fault + dependency predicate chooses commit, replay, or abort.

The scheduler may overlap commands aggressively, but it may not elide stages 3, 7, or 8. Those stages carry the semantic obligations that distinguish completion from commitment.

24.1 Scoreboard entries

At minimum, an in-flight entry should retain command ID, packet ID, consumed generation, dependency token, destination reservation, certificate ID, replay-region ID, engine assignment, and current state. Debug builds should additionally retain timestamps for every state transition.

25. Scratchpad Bank Contract

The 16 physical banks should be exposed to the allocator through a bank-mapping function rather than a fixed contiguous modulo mapping. Polyphase and strided packet accesses can otherwise create systematic conflict patterns.

A prototype mapping may use a configurable XOR swizzle,

$$\text{bank}(w) = \left(w_{3:0} \oplus w_{7:4} \oplus h(P)\right) \bmod 16,$$

where w is the local word index and $h(P)$ is a four-bit packet/phase hash. This is an implementation policy, not part of the mathematical contract, and should be replaced if FPGA counters show worse behavior for target transforms.

Bank arbitration must report conflicts to performance counters. The simulator and FPGA must agree on the definition of a bank-conflict cycle before architectural performance claims are made.

26. Formal Safety Invariants

The following properties should be encoded as assertions in the RTL and independently checked in the reference model.

I1: stale-commit exclusion
: If the consumed generation differs from the current generation, `commit` is false.

I2: certificate gating
: If a required certificate is unfinished or failed, `commit` is false.

I3: fault containment
: An uncorrectable protected-state fault cannot produce a committed result.

I4: dependency validity
: A completion token cannot release downstream committed work before the producing command commits.

I5: replay locality
: A retryable failure invalidates only the declared replay region and its dependent provisional descendants.

I6: control forward progress
: Under finite injection and a responsive destination, C0 traffic cannot be permanently starved by C2/C3 traffic.

I7: deterministic mode
: When deterministic mode is requested, reduction topology, rounding mode, and replay decisions are reproducible from the same command and state trace.

I8: backend semantic equivalence
: CUDA, golden-model, FPGA, and ASIC executions of the same legal Packet IR program satisfy the same declared output contract.

For I1 the hardware-level implication is simply

$$(g_{\text{consumed}} \neq g_{\text{current}}) \Rightarrow \neg \texttt{commit}.$$

The implementation package includes a minimal executable reference test for this invariant.

27. Boot, Discovery, and Runtime Bring-Up

A target should not accept general packet programs immediately after reset. Recommended bring-up order:

1. reset all PSMs, queues, routers, certificate accumulators, and generation tables;
2. verify ROM / firmware image and ABI compatibility;
3. scrub or initialize descriptor SRAM and protected control memories;
4. enumerate PSM, transform, solver, exact-arithmetic, SRAM-bank, HBM, and link capabilities;
5. run a local self-test for scratchpad banks, command queue, generation guard, and commit controller;
6. run a cross-quadrant C0/C2 traffic smoke test;
7. publish the capability block to the host runtime;
8. accept only Packet IR plans whose required feature set is a subset of the discovered capabilities.

A failed generation-guard or commit-controller self-test is a hard device-disable condition. A failed transform or solver self-test may permit a degraded capability profile if the runtime can route those operations elsewhere without changing semantics.

28. Telemetry and Performance-Counter Minimum Set

Every PSM should expose counters sufficient to distinguish a poor mathematical plan from a poor hardware mapping. Minimum counters:

- commands fetched, issued, completed, committed, replayed, and aborted;
- cycles stalled on dependency, destination allocation, scratchpad banks, NoC, transform, solver, certificate, and HBM;
- scratchpad bytes read/written by P/F/M/R logical class;
- bank-conflict cycles by bank and access class;
- NoC flits or bytes by C0-C3 class and hop count;
- factor / atlas reuse count;
- generation mismatch count;
- certificate failure count by reason;
- replayed cycles and bytes;
- deterministic-mode invocations;
- corrected and uncorrectable protected-state errors.

The planner calibration loop should consume these counters directly rather than tuning against wall-clock time alone.

29. Implementation Handoff Package

The v0.2 package accompanies this PDF with:

- Python-generated PDF, SVG, and PNG engineering blueprints;
- the Python figure generator;
- SystemVerilog semantic boundary skeletons for generation and commit logic;
- a minimal Python semantic reference model;
- executable smoke tests for valid commit, stale generation replay, and fatal-fault abort;
- a manifest and status boundary distinguishing implemented scaffolding from future production RTL.

The supplied Python semantic tests pass in the build environment. No SystemVerilog compiler was available in that environment, so the RTL skeleton is *not* claimed to have passed HDL compilation or synthesis.

30. Architecture Freeze Checklist

A PRISM-1R implementation is ready to enter production RTL only when all items below are checked.

☐ Packet / transfer / atlas / refinement / generation descriptor schemas are versioned.
☐ Packet IR canonical serialization is stable.
☐ Output contracts are explicit and tested.
☐ Commit and replay state machines are formally specified.
☐ Generation invalidation rules are formally specified.
☐ Minimum C0-C3 traffic guarantees are specified.
☐ Scratchpad allocator contract and bank-conflict behavior are specified.
☐ Supported arithmetic formats and rounding rules are fixed.
☐ Exact arithmetic obligations are bounded and specified.
☐ Transform and small-solver minimum operation set is fixed.
☐ CUDA and PRISM conformance vectors pass.
☐ Golden model is reproducible from a tagged release.

☐ Cycle simulator reports all modeled queues and conflicts.
☐ FPGA prototype demonstrates stale-commit prevention.
☐ FPGA prototype demonstrates certificate-triggered local replay.
☐ Negative controls route correctly.
☐ Benchmark reports separate representation, backend, architecture, and amortized effects.
☐ Fault-injection suite passes.

31. First 30 Engineering Tasks

No.	Task
1	Freeze Packet IR schema version 1.
2	Freeze packet, transfer, atlas, refinement, generation, and certificate descriptors.
3	Implement canonical binary/JSON serialization and schema validator.
4	Build deterministic golden Packet IR interpreter.
5	Define canonical execution trace format.
6	Implement generation guard in golden model.
7	Implement certificate state machine in golden model.
8	Implement local replay region semantics.
9	Port representative block-diagonal and BDLR graphs to canonical Packet IR.
10	Port transform-to-small-solver reconstruction chain.
11	Implement CUDA grouped block backend.
12	Implement CUDA factor-resident BDLR backend.
13	Implement CUDA transform/solver chain with graph replay.
14	Add structured and unstructured negative controls.
15	Define benchmark result schema separating gain types.
16	Implement PSM functional simulator.
17	Add 16-bank scratchpad timing model.
18	Add C0-C3 NoC queue model.
19	Add quadrant placement and hop-byte accounting.
20	Add transform/solver occupancy model.
21	Add hierarchical certificate timing model.
22	Add replay-cost estimator.
23	Implement joint representation/resolution/placement cost planner.
24	Write SystemVerilog command and descriptor interfaces.
25	Implement FPGA scoreboard and generation guard.
26	Implement FPGA scratchpad bank controller.
27	Implement FPGA certificate/commit path.
28	Implement fault injection and stale-generation testbench.
29	Run first end-to-end FPGA conformance workload.
30	Freeze shuttle-chiplet experiment matrix from measured FPGA bottlenecks.

32. Final Deployment Rule

The deployment program should preserve one strict ordering:

$$\boxed{\text{mathematical contract} \rightarrow \text{software proof of execution} \rightarrow \text{measured bottleneck} \rightarrow \text{hardware specialization.}}$$

PRISM-1R is justified only where the representation has already removed unnecessary work and where dedicated hardware measurably improves the remaining structured computation.

The target machine is therefore not a GPU with extra packet opcodes. It is a processor in which representation, resolution, locality, uncertainty, and commit state are all architectural information.

End of PRISM-1R Implementation and Deployment Blueprint v0.2