

CPM (Configurable Packet Modifier) Project

Verification Plan

Project Overview & Scope

DUT Summary

Configurable Packet Modifier (CPM): block with AXI-Stream-like input/output, programmable transformation (PASS, XOR, ADD, ROT) via RAL, mask/add_const params, and drop-by-opcode rule. Packets are accepted when valid && ready; output follows same handshake. Internal 2-slot pipeline with configurable latency per mode.

Verification Objectives

- Verify stream handshake protocol and interface compliance (stability under stall, bounded liveness).
- Validate RAL register map (CTRL, MODE, PARAMS, DROP_CFG, STATUS, COUNT_IN, COUNT_OUT, DROPPED_CNT) and reset behavior.
- Prove functional correctness of transformation (PASS/XOR/ADD/ROT) and drop rule using a reference model and scoreboard.
- Ensure end-of-test invariant: $COUNT_OUT + DROPPED_COUNT == COUNT_IN$, and no unexpected leftover expected packets.

Scope

- **In Scope:** Block-level functional verification, stream protocol (SVA), RAL-based configuration, constrained-random and directed traffic, functional coverage (MODE, OPCODE, MODE×OPCODE, drop, stall), scoreboard with reference model.
- **Out of Scope:** Gate-level or post-synthesis timing, power, electrical characteristics, system-level integration.

Requirements Extraction & RTM

Req ID	Requirement (from Spec)	Verification Method	Planned Artifact
R1	Packet accepted when in_valid && in_ready	Monitor / Scoreboard	cpm_in_monitor, scoreboard write_in
R2	When valid && !ready, valid and data must remain stable	Assertion	cpm_if.sv ASSERT_STABILITY_VALID, ASSERT_STABILITY_DATA
R3	Within 16 cycles of stall, either !valid or ready must hold	Assertion	cpm_if.sv ASSERT_LIVENESS_BOUND
R4	RAL reset restores default register state	Sequence	uvm_reg_hw_reset_seq, cpm_final_seq
R5	Drop rule: when drop_en and opcode==drop_opcode, no output	Scoreboard	cpm_scoreboard.sv (no push when dropped)
R6	MODE 0/1/2/3 apply PASS/XOR/ADD/ROT correctly	Scoreboard + Reference Model	cpm_scoreboard.sv predict_payload()
R7	COUNT_OUT + DROPPED_COUNT == COUNT_IN at end of test	Scoreboard check_phase	cpm_scoreboard.sv SCB_INVARIANT

Verification Strategy

Aspect	Plan
Stimulus Generation	Virtual sequence (cpm_final_seq): RAL reset, then phased config + traffic for PASS, XOR, ADD, ROT, plus stress and drop phases. Traffic uses constrained-random sequences (base_traffic_seq / coverage_traffic_seq via factory override) for coverage; directed config per mode.
Checking Mechanism	Transactional scoreboard with reference model (predict_payload from MODE/PARAMS). Compare observed output to expected queue; order-tolerant match by (id, opcode). End-of-test checks: counter invariant and queue empty (with optional +ALLOW_ONE_LEFTOVER for known RTL corner). SVA in cpm_if for protocol (stability, liveness).
Coverage Targets	Functional: MODE (100%), OPCODE (90%), MODE×OPCODE (80%), drop bin hit, stall bins hit (in/out). Implemented in cpm_coverage.sv (cpm_cg, stall_cg).
Regression / Automation	Single test (cpm_base_test) runs virtual sequence; PASS/FAIL from UVM report (0 UVM_ERROR, scoreboard mismatches). Run script: run_rtl.do (compile + simulate + coverage/assertion report).
Exit Criteria	Zero UVM_ERROR; scoreboard reports 0 mismatches (one allowed leftover with waiver); functional coverage targets met; RAL reset and end-of-test invariants checked. Assertion report (tool summary) and coverage report captured.

Functional Coverage Plan

- **MODE coverage (target 100%):**
Bins: PASS (0), XOR (1), ADD (2), ROT (3). All four modes exercised in cpm_final_seq.
- **OPCODE coverage (target 90%):**
Bins: low [0:3], mid [4:11], high [12:15]. Achieved via constrained-random traffic (coverage_traffic_seq).
- **MODE×OPCODE cross (target 80%):**
Cross of MODE and OPCODE bins; covered by phased traffic across modes with random opcodes.
- **Drop bin:**
At least one packet dropped (drop_en=1, opcode==drop_opcode). Dedicated drop phase in cpm_final_seq.
- **Stall bins:**
Input and output stall (valid && !ready) hit at least once. Achieved via +READY_PROB=80 and traffic volume.

Test Plan Table

Test Name	Type	Purpose	Traces to Req
cpm_base_test	Virtual (single run)	Full flow: RAL reset, config (PASS/XOR/ADD/ROT), traffic, stress, drop, readback. Ensures DUT alive, all modes and drop/stall coverage, scoreboard and invariant checks.	R1-R7

(One main test drives the full virtual sequence; phased coverage and scoreboard cover all requirements.)

Stimulus Planning

Sequence	Name Description	Parameters / Constraints
uvm_reg_hw_reset_seq	RAL reset	model = reg_block; run on reg_seqr.
cpm_config_seq	Write MODE, PARAMS, DROP_CFG via RAL	mode, mask, add_const, drop_en, drop_opcode; used per phase in cpm_final_seq.
cpm_final_seq	Virtual sequence: reset → PASS → XOR → ADD → ROT → stress → drop → readback	Phases with #500ns / #100ns delays; sets package globals after each config for scoreboard/coverage.
base_traffic_seq / coverage_traffic_seq	Packet traffic on input sequencer	num_packets; coverage_traffic_seq used via factory override for coverage bias.
stress_seq	High-volume traffic	Used in stress phase.
drop_seq	Traffic with drop_opcode to hit drop bin	Config drop_en, drop_opcode; traffic includes matching opcode.

Closure & Metrics

Metric	Target	Justification
UVM_ERROR	0	No functional or protocol errors acceptable for closure.
Scoreboard mismatches	0	One allowed “leftover” with +ALLOW_ONE_LEFTOVER=1 for known RTL corner; documented in bug tracker.
Functional coverage	MODE 100%, OPCODE 90%, MODE×OPCODE ≥80%; drop and stall bins hit	Aligns with project closure criteria.
Assertions	Liveness: 0 failures. Stability: documented where failures remain (TB/RTL).	Tool assertion report used as deliverable.
End-of-test invariants	COUNT_OUT + DROPPED_COUNT == COUNT_IN; queue empty (or 1 with waiver)	Enforced in scoreboard check_phase.

References

- Configurable-Packet-Modifier-CPM-Design-Specification-Version-1.0.pdf
- CPM-Final-Project-Verification-Requirements-and-Deliverables.pdf
- Bug tracker (Bug-Tracker-Asaf Kamber.xlsx) for RTL findings
- Reflection Report (Reflection_Report.pdf) for challenges, limitations, future work