2026
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

From Specification to Closure: A Semi-Automated Coverage-Driven Verification Methodology for Cache Coherent Home Nodes

Kavin Rajendran, Anishmon Soosai, Sumit Dhamanwala, Kranthi Konganti, Shivaprasad Naranapura Chandrashekara Swamy

OPENEDGES

Agenda

1. Introduction
2. Design & Verification Context
3. Motivation
4. Problem Statement
5. Solution Overview
6. Methodology Process Flow
7. Scaling to Full Regression
8. Results & Insights
9. Conclusion

Introduction

Growing Complexity

- Scalability
- Hardware Coherent
- Multi Core
- Heterogeneous Compute

Critical Verification Gap

- Home Node uArch
- Complex protocol flows
- Cache State Transitions

Traditional Methods Fall Short

- Scoreboard & Assertions
- Don’t quantify tests
- System level interactions missed

Design & Verification Context

- Highly configurable NoC
- Various Topologies: Mesh, Ring, Crossbar etc
- Generated RTL with variable design parameters
- Dynamic Testbench

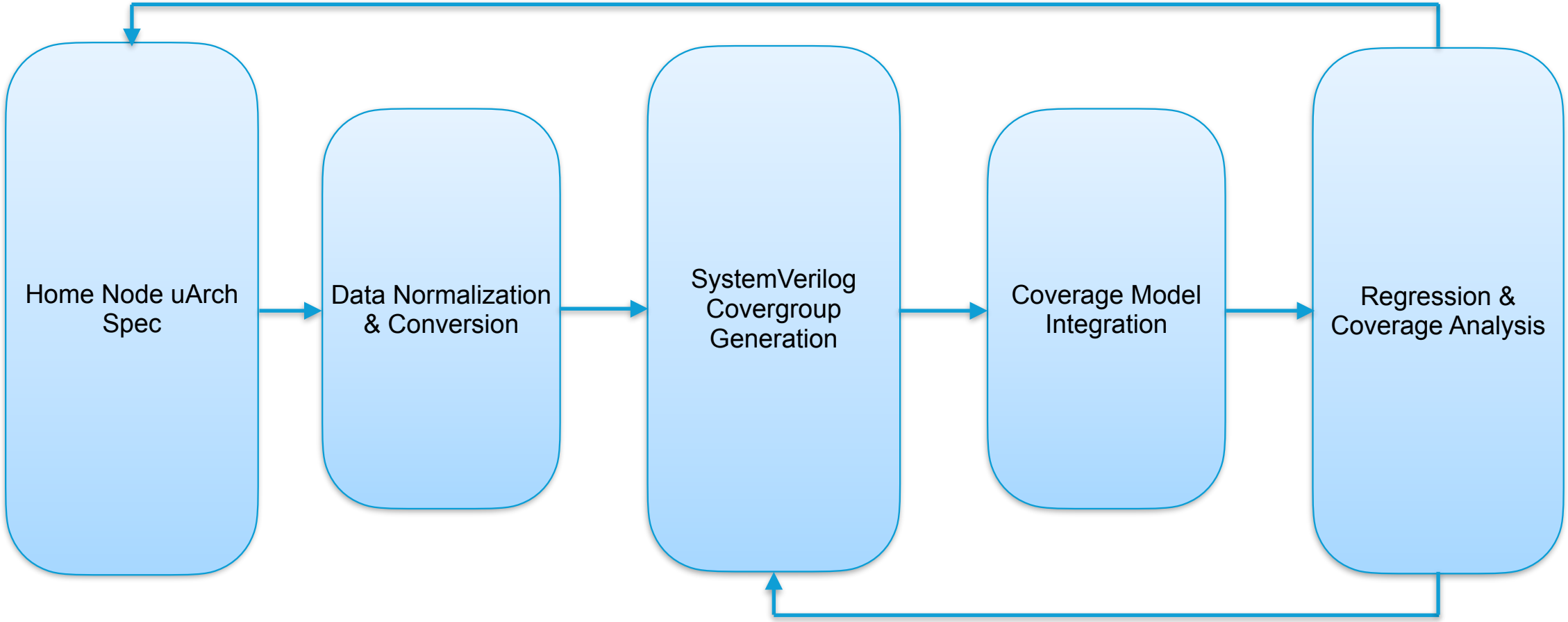
Motivation

- AMBA CHI: Highly concurrent, out-of-order flows across multiple channels
- Home Node is the coherency authority
- Verification space explodes combinatorially
- Evolving specification: Frequent spec changes requiring adaptive strategies
- VIP coverage: Protocol centric, limited uArch visibility
- Directed tests don’t scale & Random traffic lacks intent
- Hard to measure meaningful progress

Problem Statements

- Automate coverage development to easily absorb spec changes
- Reduce complexity, Increase maintainability
- Coverage model with impactful cross cover data

Solution Overview



Pillar 1 - Automation

- Python parses structured Home Node spec
- JSON-based opcode flow representation
- Generates Cover points + Crosses

Opcode	Current State	Lookup Result	Next State
ReadOnce	SF Lookup	Hit	Snoop
		Miss	SLC Lookup

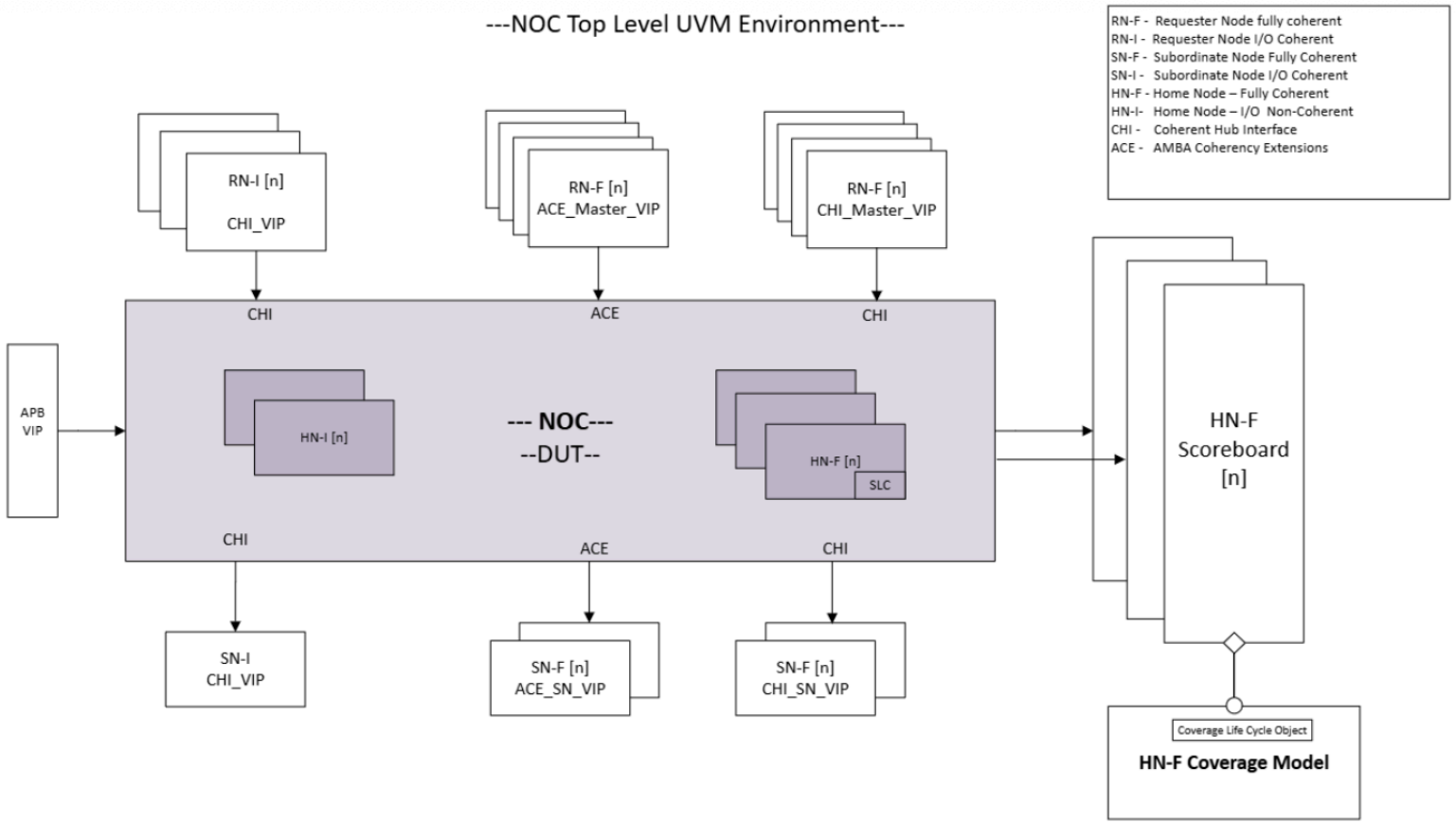
```
[
  {
    "Opcode": "ReadOnly",
    "Current State": "SF Lookup",
    "Lookup Result": "Hit",
    "Next State": "Snoop"
  },
  {
    "Opcode": "ReadOnly",
    "Current State": "SF Lookup",
    "Lookup Result": "Miss",
    "Next State": "SLC Lookup"
  }
]
```

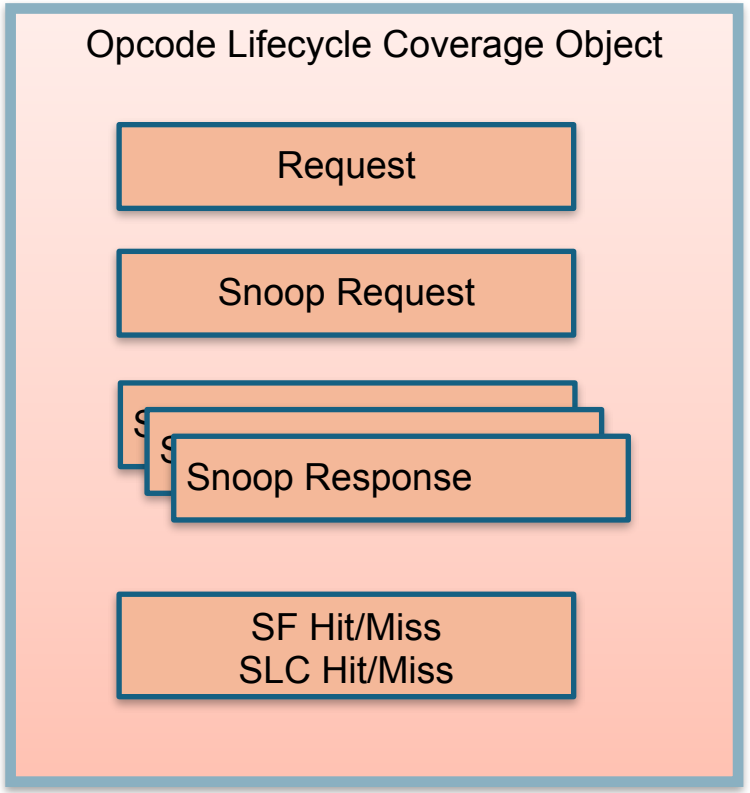
```
covergroup cg_ReadOnce;
  option.per_instance = 1;
  option.goal = 100;

  cp_SLC_LkUp: coverpoint slc_lkup_status_fcov {
        bins ValidCom[]        = { MISS ,HIT};
    }
```

Pillar 2 - Lifecycle tracking

- Per-channel monitoring
- Scoreboard-driven lifecycle construction
- Supports 1 to 1 or Many to 1 mapping

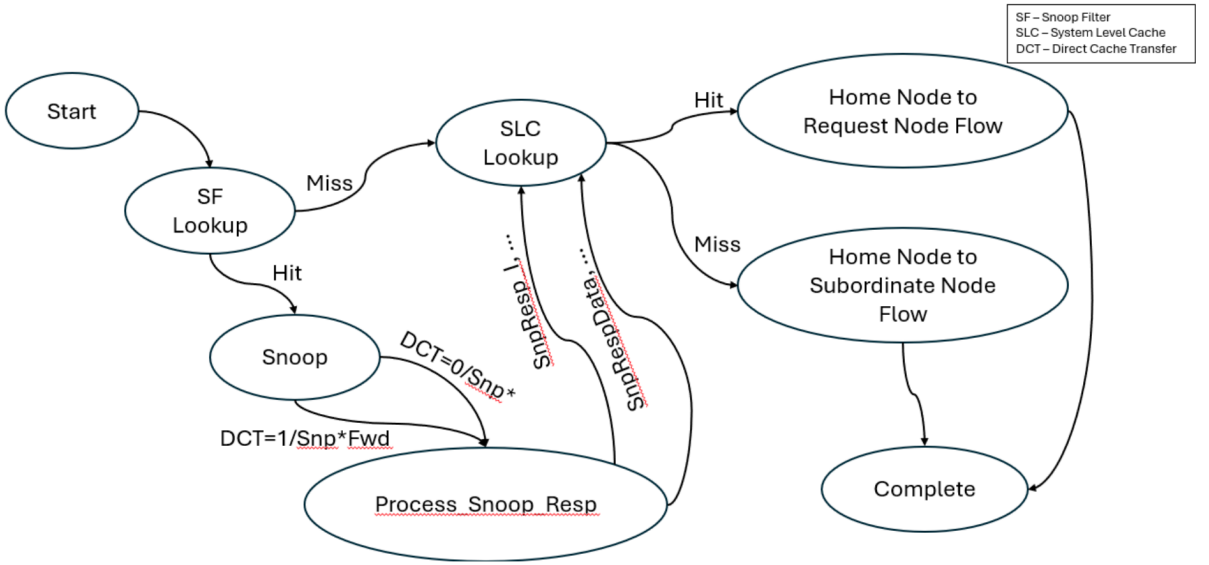




Pillar 3 - Coverage Model

- Opcode x Snoop outcome coverage
- Response vs forwarded states
- Internal HA metadata integration
- Cross Example: { Req x SF Hit/Miss x SnpReq x SnpResp }

Opcode	Snoop Request	Snoop Response Opcode	Response States	Forward States
ReadOnce	SnpOnce	SnpResp	I, UC, SC	NA
		SnpRespData	I, UC, SC, UD, SD	NA
		SnpRespDataPtl	I_PD, UD	NA
	SnpOnceFwd	SnpResp	I, UC, SC	NA
		SnpResp_<RS>_Fwded_<FS>	I, UC, SC, UD, SD	I
		SnpRespData_<RS>_Fwded_<FS>	I_PD, SC_PD	I

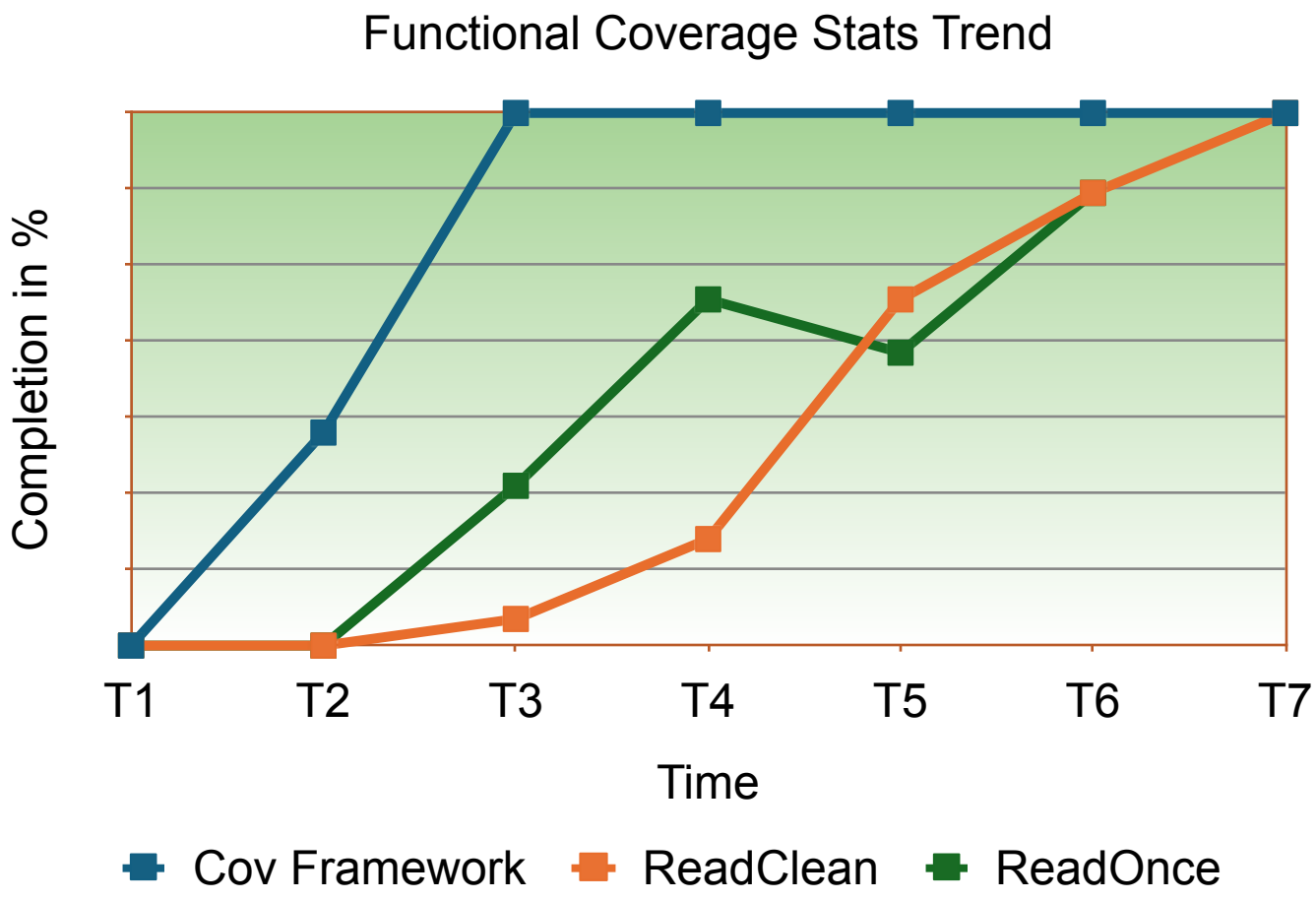


Scaling to full regression

- Bring up & validation: Automation scripts, Life cycle object construction and Coverage sampling logic.
- Directed tests on a smaller network.
- Baseline coverage numbers established.
- Deploy on daily/weekly regressions

Results & Insights

- ✓ Helped identify unseen snoop responses
- ✓ Exposed uArch bugs, CHI compliance issues
- ✓ Recover from spec changes in hours
- ✓ Focus on ‘High ROI’ stimulus with constraints control
- ✓ Preserve test bench integrity via coverage trends



Conclusion

- Transformed coverage from late stage checklist
- Used as a proactive, continuous metric
- Served as a resource planning tool
- Smarter use of regression cycles
- Deeper observability, Faster Closure