International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 6091-6108
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Pre-Silicon DFT Feedback Loops: Enhancing GPU Productisation Efficiency

Karan Lulla*

Senior Board Test Engineer, NVIDIA, SantaClara, CA, USA
*** Corresponding Author Email:** karanvijaylulla08@gmail.com - **ORCID:** 0009-0007-7491-4138

Article Info:

DOI: 10.22399/ijcesen.3778
Received : 27 June 2025
Accepted : 11 August 2025

Keywords

Pre-silicon DFT feedback loops,
GPU productisation,
Automatic Test Pattern Generation (ATPG),
Fault simulation,
Test-point insertion (TPI).

Abstract:

This paper describes a controlled pre-silicon Design-for-Test (DFT) feedback loop mechanism that allows faster GPU productisation with less test cost. DFT is redefined in a one-time metallic achievement into an Observe → Analyze → Decide → Act → Verify loop nested within CI/CD. A typical telemetry model allows combining ATPG coverage, fault-simulation results, timing and power constraints, diagnosis artifacts, and provenance to perform auditable automation. The three levers that are the primary focus of controller policies are constraint tuning, timing-aware test-point insertion, and selective pattern regeneration, which have been verified for verification gains using A/B comparisons to frozen baselines under quality gates. Screening gains are observed on typical GPU partition types (streaming-multiprocessor cluster, L2 cache slice, and HBM PHY wrappers): +1.6 to 2.8 percentage-point stuck-at and +1.0 to 2.2 percentage-point transition coverage; 22 to 38 percent reduced patterns; tens of percent tester time savings; and fewer suspect sets in diagnosis. Orchestration capabilities: Content-addressed storage, checkpointed compute, license-aware scheduling, keep throughput and reproducibility, dashboards expose Pareto tradeoffs and undetected-fault heatmaps to concentrate the compute. The limitations include the cell-aware run time, high X-density, multi-clock interactions, analog adjacency, seed sensitivity, and inter-tool naming drift, which are all alleviated by incremental engines, cache reuse, schema normalization, ECO-safe edit windows, power-aware X-filling, and controlled rollbacks. Future research will focus on multi-objective controllers, LBIST/MBIST, and in-field telemetry, collection of fabrication/test-floor data, cross-generation transfer learning, and open benchmarks to achieve sustained comparability and reproducibility across GPU families at scale.

1. Introduction

Recent GPUs have tens of billions of transistors, heavily pipelined execution units, and a multiported array of SRAMs. Design-for-test is emphasized at this architectural scale due to a drop off in controllability and observability as logic depth, clock gating, and power partitioning rise. Shift operations are made difficult by long scan chains and by shared clock domains, and asynchronous boundary mixes with clean launch and capture. Large limbs test limitations tend to jeopardize IR-drop and thermal margins, leading to restrictions on hastening convergence. The tester channel limitations limit choices in compression. Nasty schedules scale up the expense of every pattern. DFT complexity grows together with the number of cores and high-bandwidth memory added to the designs. There must be a feedback loop to convert telemetry to action that enhances coverage, minimizes pattern count, and rapid productisation without breaking performance constraints, power limits, or area.

The issue of minimizing the restriction of test cost subject to coverage, power, and schedule constraints of product release is handled. The solution has to increase stuck-at and transition coverage of sign-off levels, minimize pattern count without compromising the level of diagnostic resolution, comply with shift and capture power constraints, and reduce latency between telemetry and action. The DFT coverage is pre-silicon logic: logic scan insertion and compression, automatic test pattern generation, and fault simulation. Analog and mixed-signal DFT is reserved only for digital wrappers of PHYs and sensors. The direction puts the focus away from deterministic stimuli with conditional pseudo-random seeding, where it compacts favorably. Some of the assumptions are RTL freeze windows,

controlled ECO entry, sign-off requirements, repeatable runs due to version-controlled configurations, and vendor-independent integration. Content-addressed artifacts infrastructure and farm scheduling allow repeatable decision making and traceability.

A stuck-at fault causes a node to assume logic zero or one as a permanent value. Transition and path-delay faults represent timing faults identified through launch-capture sequences. Cell-aware testing enriches fault models with library defect mechanisms. Compression refers to integrated deterministic decompression, test compaction, and streaming that minimize tester memory and channels. The test-point insertion introduces the controllability or visibility at specified nets. The feedback loop represents an Observe Analyze Decide Act Verify cycle that is part of CI/CD. Telemetries are eaten, policies prioritize actions, and artifacts are recreated, and findings against quality gates are compared. The most critical metrics are stuck-at coverage (Csa), transition coverage (Ctr), pattern count (PC), test time (Ttest), and a proxy of DPPM.

This paper adds a concrete architecture of pre-silicon DFT feedback, a normalized data model of covering and constraining and the provenance thereof, controller policies to tune constraints and rank test-points, and quality gates that safeguard timing, power, and reproducibility. Anticipated deltas will be practical: one to three percent absolute coverage gain on challenging blocks using dedicated test points and more polished constraints; twenty to forty percent decrease in pattern count by use of selective compaction and adjusted masking; twenty to thirty percent decrease in tester time by reduction of pattern sets. Other effects are increased farm use of the mathematical model, reduced late ECOs, and more comprehensible engineering audit trails. As compared to ad-hoc flows, the feedback loop serves as a replacement by eliminating one-shot decision-making and replacing it with a measured iteration, allowing a data-driven trade-off of his position and objectives in performance and area.

Chapter 2 provides a review of contemporary work on big-DFT on SoCs and GPUs, characterization of ATPG progress and fault modeling, and the feedback-driven orchestration. Chapter 3 explains approaches, including the overview of the system, fault models, and pattern optimization, test-point and scan planning, and KPIs that drive decisions. In Chapter 4, the architecture of the feedback loop and the integration of the tool chain are stated, including telemetry, controllers, orchestration, and quality gates. Chapter 5 documents experiments of GPU blocks, baselines, coverage results, run time, scalability, and test cost. The results concerning productisation are interpreted in Chapter 6, trade-offs and constraints are analyzed, and portability is outlined. Chapter 7 suggests future developments of multi-objective controllers and post-silicon data fusion. Chapter 8 ends with recommendations.

2. Literature Review

2.1 DFT for Large SoCs/GPUs: Scan, Compression, LBIST/MBIST

Three realities converge to influence design-for-test (DFT) of modern graphics processing units (GPUs): extreme sequential depth, power-integrity constraints, test-application compliance, and Post-silicon validation timing pressures to shorten sign-off windows. Different genres of scan-based DFT embody the way that these pressures were incorporated into the standard methodology. Classical full-scan converted combinational to sequential q problems, inserting scan cells as needed into the design; however, the shift time and tester memory scaled with the number of flip-flops. Scan-compression architectures broke this linearity, compressors at the outputs, and decompressors at the inputs of scan paths [23]. Correlation is diffused through linear phase shifters, and excitation is enhanced, whereas response compactors are based on XOR delay pin counts and shift time. Compression results in stricter limits on non-known (X)-management and on compactor aliasing, and sign-off targets optimize the size of practical compactor matrices, bound mask allowables, and investigate aliasing budgets. Hierarchical scan-based DFT is signed off as shown in the figure above, replicated into the parent block, and integrated at the top level, where glue logic, memory testing, and interconnect needs are tied off. They support GPU realities of extreme depth and sequentiality, power-integrity demands, and short post-silicon schedules through this staged flow; they also support scan-compression tapes. Scan chains are fed with decompressors at ingress to a block, and responses are gathered out with compactors at egress, using linear phase shifters to diffuse correlation and compactors based on XOR to reduce shift time. Since compression constrains X-management and aliasing budgets, every hierarchy gate plans and optimizes compactor matrix sizes, masks, and coverage goals before signifying artifacts to the next hierarchy level.

The chain engineering grew up with compression. Timing-aware stitching places lock-up latches across domains, constrains the longest chain to

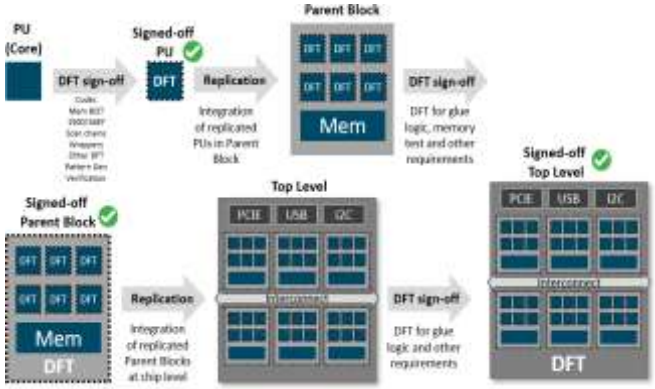

Figure 1: *Hierarchical scan-based DFT sign-off and block replication*

encompass the head of shift time, and aligns chains to routing resources to limit congestion and skew. With GPUs in the tens of chained millions of transistors, a dozen asynchronous or mesochronous clock domains operate in parallel; scan planning is then considered alongside floorplanning, and endpoint-to-endpoint routing of chains, clock trees, and power grids are co-optimized. Testing, designers will accept mission-mode-aware constraints (keeping isolation, retention, and power-gating intent), such that the test modes are an accurate model of real operation and do not have unrealistic controllability/observability.

A power-sensitive test was essential because scan toggle is frequently much more common than functional switching. Low-toggle X-filling and weighted-random fills minimize concurrent switching, launch-on-shift capture distributes the activity, and constraint pruning prevents risky overlaps between test clocks. Flows target per-pattern toggle density, peak capture current, scan cell switching cap enforcement, and patterns that violate IR-drop, electromigration, or thermal limits, and place them in quarantine. Since GPUs focus on high-frequency cores and dense SRAMs, the controls allow avoidance of false failure at the silicon and lower pessimism during pre-silicon fault simulation. Built-in self-test (BIST) augments scan. Logic BIST (LBIST) tests combinational logic based on on-chip-generated pseudorandom patterns [14]. It can be tested at nearly full speed and with a few tests familiar to the tester, thus it sacrifices determinism to throughput and can be used to screen structure in the lab and test in the field. Memory BIST (MBIST) runs March-type algorithms to address stuck-at, transition, and coupling errors and orchestrate redundancy recovery of yield loss. MBIST is not optional in GPUs where register files, caches, and fabric buffers consume the majority of array space; however, deterministic logic ATPG and its associated structural coverage expansion, diagnosis resolution, and pattern compaction budgeting are the primary tools. The BIST system uses TPG, CUT, and RA blocks that coordinate the Logic BIST (LBIST) and Memory BIST (MBIST) operations as shown in Figure 2 below. The BIST controller drives tests, initiating and finishing tests, and generating pass/fail outcomes, using pseudorandom patterns and March-style algorithms, MBIST and LBIST.

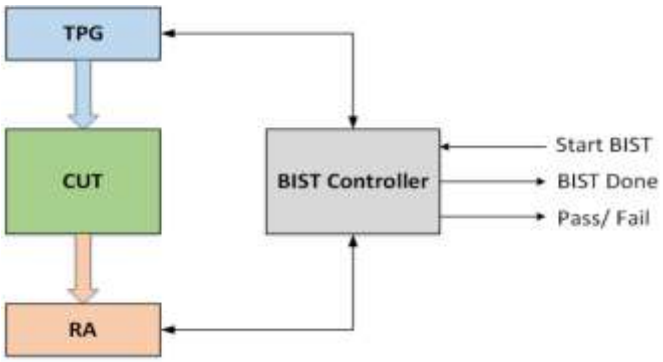

Figure 2: *An Overview of LBIST/MBIST block*

2.2 ATPG & Fault Simulation: Coverage Closure Techniques

Closure of that coverage on multi-billion-gate GPUs depends on the interaction between deterministic and pseudorandom pattern generation, compaction, unknowns treatment, and fault modeling. Deterministic ATPG addresses specific fault lists, enumerating the justification/propagation space with explicit mode constraints. Structural dominators, dynamic implication learning, conflict-directed

backtracking, and cautious branching heuristics give the technique scalability. Deterministic approaches observe launch/capture timing (launch-on-capture as opposed to launch-on-shift), respect power caps. They can selectively point to particular cones that are sources of residual undetected errors. Pseudorandom generation, in LBIST or weighted-random ATPG, explores large areas of the state space quickly, enhances path diversity, and can be used as a source of seeds, but can stall on reconvergent logic unless the logic is test-pointed (TPI) or subjected to deterministic top-up.

Compaction is constraining and enabling. Spatial compactors decrease the number of pins and time compactors can compress several capture cycles into signatures, although they cause an aliasing risk. The probability of aliasing is an analytic quantity when the compactor dimension, the expected X-masking, and a fault model are known; sign-off limits the number of bits masked in a pattern and within a pattern set. Since the excessive masking suffers the drawback of destroying observability, contemporary flows follow the masked-to-observed quotients and mark inconvenient schemes that are unlikely to lead to effective diagnosis. The method of pattern compaction (both static and dynamic) reduces the count of vectors through the combination of compatible assignments and cutting off dominated patterns without loss of detectability or diagnostic syndromes [4].

Unpredictable management is the chief of realism. Examples of X-sources are uninitialized memories, mixed-signal wrappers, clock-domain crossings, asynchronous resets, and power-gated islands. X-bounding, incomplete scan on recalcitrant sequential beings, wrapper logic to sensitize an unsteady interface, and test-mode clamps all guard against this by the designers. X-filling at the ATPG layer aims to take assignments that minimize X-propagation and enhance compaction subject to power constraints; structural untestability and environmental untestability (i.e., constraint conflicts), as well as tool artifacts, are reported at the infrastructure layer. Such a taxonomy makes it more possible to target interventions, e.g., TPI on poor observability, constraint relaxation on environmental untestability, or waiving on provably redundant faults.

With the scaling of the technology, fault modeling has become more complex. In addition to stuck-at and transition/path-delay, cell-aware test generates transistor-level defect libraries based on SPICE-tested standard cells, resulting in a better match to actual defect distributions at sophisticated nodes. Small-delay defect model and path sensitization look at marginal timing escapes pertinent to highly pipelined GPU processors [10, 30]. The price is a bigger fault universe and stress on ATPG and sims runtimes; most groups hence gate cell-aware usage on libraries/components where there has been proven value, or they run it as a late-stage incremental top-up. To maintain their cadence, fault simulation engines co-evolved in parallel and concurrent simulation to take advantage of word-level parallelism and event-driven activity, statistic fault dropping to terminate previously found faults, and incremental engines to reuse correlation paths between related patterns. These accelerations transform multi-day regressions to overnight cycles on a typical compute farm and generate low-granularity telemetry--per-fault detection counts, per-pattern toggle/capture metrics, and masked-bit distributions--that power the feedback loop.

2.3 Feedback Loops in EDA Flows and CI/CD for Hardware

Continuous integration has recast the way an organization views DFT as a single effort into a governed feedback loop. One useful loop consumes nightly coverage and fault-simulation telemetry, normalises logs across tools, and archives artefacts in content-addressed storage in such a way that any output can be revived off hashes of inputs, scripts, and version-specific tools. Automated quality gates determine promotions: a candidate pattern set can only proceed to the next stage when a configured delta has increased structural coverage, power metrics are contained within budget, compaction ratios are satisfied, and there are no timing-sensitive scan stitching regressions. Rollbacks are symmetric rollbacks: an artifact-dependent regression outside of tolerances is automatically rolled back to the previous good artifact [37]. CI dashboards display trendlines in coverage, count of patterns, ratios of mask bits, test-time forecasts, and compute utilization to allow teams to regulate license pools and farm schedules.

Security and governance are part of test artifacts since they expose internal design and potentially reflect sensitive microarchitectural information. DevSecOps concepts such as automated static reviews, dynamic testing, and software composition analysis are generalized into the hardware one, the policy-as-code gates, through which the promotion of unsafe patterns is prohibited, and traceability of every published pattern set can be traced back to its origin [8]. Practically, this becomes mandatory metadata (design revision, constraint bundle, tool versions, seeds), signature verification of artifacts, and change approvals by roles when the change that impacts observability paths and any logs shared externally are redacted. The result is an auditable, reproducible loop in which the pipeline enforces the

coverage growth, power safety, and IP hygiene at the same time, avoiding ad-hoc human review.

2.4 Data-Driven / Learning-Based Test Optimization

Data-driven approaches are gaining popularity as a method of making three everyday decisions that come to scale with designs: where to add test points, how to optimize ATPG constraints, and which patterns to cache within tester memory and timing budgets. In the case of TPI, attributes would be SCOAP-style controllability/observability, netlist-graph centrality (betweenness, eigenvector scores), reconvergence depth, the cardinalities of fan-in/fan-out, the estimated cone toggling due to random fill, historical detectability of faults arriving at a node, and closeness to known X-sources. Supervised ranking models can rank TPI candidates to maximize incremental coverage per area-power cost and can satisfy time guardrails. State records the current KPIs, coverage, the number of patterns, masked-bit budgets, and power peaks. In contrast, actions alter capture windows, X-mask budgets, clocking modes, and compaction parameters in sequence [18]. In their bandit and Bayesian optimization approaches, expensive evaluations are concentrated in high-leverage settings through the use of surrogate models trained on previous runs. Reinforcement learning can plan loop actions, such as whether to regenerate patterns, insert points, or relax a constraint, by carb quality gates and resource budgets.

Pattern ranking counteracts the tester limitation of discarding redundant vectors, but not the loss of detectability and diagnostic resolution. Isolated characteristics comprise the per-pattern contribution to coverage, overlapping with neighbors, and estimated torture, peak current, and diversity of diagnostic syndrome. Robust reductions are also attained in a two-stage scheme, fast static ranking augmented by focused fault simulations of the activated top-K candidates, with no re-simulation of the entire set. Whether the emerging machine-learning literature concerning large models can represent heterogeneous inputs and compute over them to provide value on targeted queries, it is a trend motivated to create multi-signal representations of DFT telemetry [32]. To do so in practice, a layered data model - raw artifacts, normalized to a consistent schema, represented as dense vectors to enable learning on graphs, counters, bitmaps, and logs with no custom feature engineering per tool.

2.5 Gaps and Open Challenges for GPU Productisation

However, there are a few barriers between the laboratory-level loops and production-level deployments at GPU scale, even though progress is ongoing. The first is scale as such. Tens of millions of flip-flops throughout netlists generate scan architectures involving thousands of chains, multiplying the compression ratio into large ratios of high-stress aliasing assumptions. By orders of magnitude, cell-aware fault lists widen the target space and take ATPG as well as simulation run times into hundreds of CPU-hours per iteration; lacking incremental simulation, checkpointing, and result caches, loop cadence becomes impossible to sustain. The other one is the fidelity of telemetry. Reports should robustly separate structural untestability, environmental conflict, and tool artifact; otherwise, controllers are optimizing the wrong things [1]. Tool normalization cannot be trivially addressed, as tools differ in naming, hierarchy, and formatting in synthesis, scan insertion, ATPG, and fault simulation; stable names and robust actionable mappers are needed to assign undetected faults to stable causes.

There is cross-interoperability. One loop blocks logic synthesis, scan insertion, ATPG, fault simulation, static timing, and place-and-route, each with semantically inconsistent artifacts. There may be little prospect of relating coverage deficiencies to particular nets or constraints without a persistent schema and dependable mapping of instance names between different compile tasks. Correlation services will need to resolve differences in renames and flattening; visualization should switch away from instance-level diagnostic results to layout-sensitive heatmaps so that physical concerns can be considered to inform TPI, re-stitching decisions. Four is governance. Artifacts of tests should be versioned, reproducible, and access-controlled [39]. Scan configurations and pattern sets, as well as BIST microcode, can leak valuable design information in an inappropriately handled system; they are also a high-value form of intellectual property requiring policy-based store and destroy policies. Operations telemetry security research focuses on the security implications of data streams themselves, leaving the system as an attack surface, data conveyed or staged without adequate guarantees; corresponding reasoning impels encryption, role-based access, and audited promotion gates to dataflow pipelines (Malik & Prashasti, 2023).

Coupling in organizations is difficult. Feedback loops intersect with RTL design, physical implementation, verification, silicon bring-up, and product engineering. When quality gates and budgets lack clear ownership, local optimizations can hurt global performance [35]. A timing team

may lock coverage growth by blocking corresponding scan re-stitch, or a DFT team may maximize compression at the cost of unintentionally adding ATE memory consumption. The hard tester limitations introduced by productisation are vector memory limits, loading/unloading latencies, sharing among multiple sites, and interface delays, which curtail what may be suggested by the loop. A realistic agenda has standardized, content-targeted artifacts surviving tool changes; incremental automated test program generation and simulation keyed by dependency graphs; diagnostic-sensitive compaction which maintains localization depth in high compression; and multi-objective controllers constrained to be utterly harmless in the sense that coverage gains never increase pattern count, power, or tester time past the budgets.

3. Methods and Techniques

3.1 System Overview: Signals, Artifacts, Decision Points

Implemented as a production pipeline, the pre-silicon DFT feedback loop consumes representations of the design. It produces measurable test artifacts that can be understood within large-scale, long-term constraints, then freely commits to its next course of action within the constraints of explicit guards. Synthesizable RTL and gate-level netlists (.v/.sv), static timing and test constraints (.sdc and .tcl), ATPG reports, fault-simulation (FSIM) logs, and power caps explicitly limiting peak shift and capture activity are inputs [34]. The pipeline normalizes logs and reports into a schema that includes design ID, netlist, constraint hashes, seeds, tool versions, and the compile flags to make runs reproducible and comparable across builds. The materialized artifacts are coverage databases indexed by fault model and cone, complete provenance (pattern IDs, source seeds, and mask budgets), pattern sets, scan-chain maps, and prioritized lists of test-point insertion (TPI) candidate sites. Decision points have to be positioned at the stable interfaces: tuning a constraint before an ATPG, editing TPI before scan stitching, and selectively regenerating patterns after diagnosis or FSIM regression. Orchestration work maintains entities to content-based storage and tags the checksums and timing waivers; promotion regulations specify minimum coverage delta and prohibit net negative slack. Telemetry is distributed in the form of time-series dashboards and a per-block Pareto chart, to enable engineers to audit the reason that the loop behaved in that particular way. These practices in governance and scaling resemble the stable practice of communication-system architecture, in which standardized interface, provenance, and quality gates are the precondition to safe automation [28].

3.2 Fault Models & Objectives

The loop demands a specific goal per block and per fault model. Typical targets are stuck-at coverage $C_{sa} \geq 98\%$ $C_{sa} \geq 98\%$ and transition/path-delay coverage $C_{tr} \geq 90\%$, with cell-aware tests selectively enabled as highlighted in Table 1 below. Cell-aware inclusion is supported by a policy dependent on library maturity and runtime budget: when defect susceptibility/field-return signatures are available on a per-cell basis and imply subsets of standard cells, only those libraries are enabled cell-awarely; otherwise, low-risk libraries are excluded to contain ATPG/FSIM effort. The objectives are stored as service-level objectives (SLOs) per block, maximum allowable pattern count, minimum scan shift under power limits, and launch/capture timing windows.

Table 1: *An Overview of Fault Models, Coverage Objectives, Risk Factors, and Actions for DFT*

Target Fault Model	Coverage Objective	Risk Factors	Action/Adjustment
Stuck-at	Csa ≥ 98%	Undetected faults, architectural criticality, observability	Adjust constraints (e.g., masks, timing)
Transition/Path-delay	Ctr ≥ 90%	Fan-out, gated clocks, multi-domain crossings	Insert TPIs or regenerate patterns
Cell-aware Tests	Selective based on cell maturity and runtime budget	Library maturity, field-return signatures, defect susceptibility	Insert TPIs or regenerate patterns

FSIM telemetry is accumulated per cycle by cone to identify undetected targets [2]. Deep logic, lots of fan-out, gated clocks, or crossings of many clock domains are highlighted as high-risk. A risk score is a composite of the undetected faults count, architectural criticality (e.g., coherence fabric, schedulers, command processors), and estimated observability. The controller can then select between

three possible courses of action, namely to adjust constraints (such as relax unknown masks, change at-speed timing) or insert TPIs to make the design more controlled/observed, or to regenerate targeted patterns with more effort in narrowly scoped cones. The statistical result is assessed objectively through seeds; termination criteria are achieved when statistical coverage gain is marginal to an hour, with its coverage per hour lower than a gradient, or when re-patterns would generate SLO violations approaching the risk-weighted, failed to be discovered faults within permissibility.

3.3 Pattern Optimization: Compaction, X-Filling, Power-Aware

Static compaction removes unnecessary patterns by retesting aggregate detection sets and extracting patterns that do not contribute any unique detections. Dynamic compaction, applied in ATPG, works by trying to increase the detect set of every pattern by solving to add further targets subject to mask and compressor constraints [36]. Compressed scan mixes aliasing and X propagation; the loop thus models channel aliasing directly, adds explicit per-channel unknown budgets, and ensures that the amount of X loaded by compaction does not exceed what the decompressor at response time can handle. X-filling has power and noise controls. Random fill seeks to maximize opportunistic detection and may increase shift activity; biased fill maximizes the use of toggling under low-cone activity; low-toggle fill minimizes IR-drop and supply droop at some cost to detection headroom. The controller chooses a fill strategy on a block-by-block basis by guessing which will run out first: power or the number of patterns. Once shift power approaches the level of the cap, loop favors low-toggle-fill and selectively increases ATPG effort on rigid cones to prevent a global count-inflation. Stop rules are data-driven: the marginal coverage gain δC/Δtime is tracked per action, and the loop halts compaction or regeneration when the derivative falls below a configured floor. This kind of adaptive choice design is compatible with principles of dynamic inference- prioritizing solutions based on intermediate cues and dedicating computer resources to a domain until that investment produces quantifiable payoffs [22].

3.4 Testability Enhancements: TPI & Scan Chain Strategies

The SCOAP (structural controllability and observability) measures that are calculated node by node are assigned to proposals and then summed per cone in the TPI proposals. Observability Candidates that optimize observability of profound reconvergent logic, or controllability of stubborn enables, are preferred, within timing and area budgets [12]. The loop supports observe points that gate internal nodes into compression channels, control points to gate seldom-activated signals, and hybrid points to re-time challenging nodes closer to scan nodes. Ability to fix each proposed point against Budgets of setup/hold, placement constraints, and test-mode isolation rules; in multi-clock areas, lock-up latches are used to bridge phase differences and extended detours—scan-chain optimization. Scan-chain strategies are co-optimized: The goal of length balancing is to reduce peak shift time by minimizing chromosome length, and at the same time, re-stitching reduces detours around long macros and crowded routing. The controller uses placement/timing hints to avoid creating critical paths, and is careful of ECO windows by grouping TPIs and chain edits into reviewable patches. Formal equivalence means that there can be no functional change with test mode disabled: static timing checks, scan-enable, and test clock timing. Any edits that are accepted are noted with justification, forecasted coverage lift, and rollback strategies so that subsequent cycles can automatically roll back lower-yielding changes.

3.5 Metrics & KPIs

KPIs make the loop accountable and enable principled trade-offs. Coverage is reported per model and per block as C=detected faults/total faults, with confidence intervals across seeds, effort levels, and library mixes. Coverage is also stratified by cone depth and safety criticality to separate shallow from deep gaps. Test time is approximated by $T_{\text{test}} \approx \sum_i (L_i/f_{\text{shift}}) + N_{\text{cap}} \cdot t_{\text{cap}}$, where L_i is scan-chain length i, f_{shift} is programmed shift frequency under power limits, N_{cap} is the number of capture cycles per pattern (including at-speed launches), and t_{cap} is capture dwell plus tester overhead. The cost proxy scales with ATE time: ATE\$ $\propto T_{\text{test}} \times$ units, letting finance translate Δ pattern count into budget impact. Secondary KPIs include ATPG/FSIM runtime, farm utilization, diagnostic resolution (median suspect count and localization depth), and flakiness rate—the fraction of patterns that intermittently fail under identical conditions. Dashboards display Pareto fronts of coverage versus pattern count, violin plots of per-block test time, and heat maps of undetected faults by cone. Quality gates enforce minimum Δ coverage for promotions, maximum allowed Δ pattern count, and zero timing violations for accepted edits [25]. These metrics close the loop by turning every action into measurable change against explicit targets.

***Table 2**: Key Metrics and KPIs for Pre-Silicon DFT Feedback Loop Evaluation*

Metric	Description	Formula/Calculation	Use/Impact	Visualization
Coverage	Reported per model and per block as C = detected faults/total faults, stratified by cone depth.	C = detected faults / total faults	Coverage is stratified by cone depth and safety criticality, showing shallow vs deep gaps.	Pareto front of coverage vs. pattern count, violin plots of test time
Test Time	Approximated by Ttest ≈ $\sum i$ (Li/fshift) + Ncap · $t$cap, where Li is scan-chain length, $f$shift is shift frequency.	Ttest ≈ $\sum i$ (Li/fshift) + Ncap · $t$cap	Test time impacts pattern count and tester utilization, scaling with scan-chain length.	Heat maps of undetected faults by cone, violin plots
Cost Proxy	ATE$ ∝ Ttest × units, translates Δ pattern count into budget impact.	ATE$ ∝ Ttest × units	Cost proxy translates pattern count change into financial impact.	Impact on budget from Δ pattern count
Secondary KPIs	ATPG/FSIM runtime, farm utilization, diagnostic resolution (median suspect count), flakiness rate.	ATPG/FSIM runtime, farm utilization, diagnostic resolution, flakiness rate	Supports analysis of test process efficiency and pattern reliability.	Farm utilization, runtime efficiency, diagnostic metrics
Quality Gates	Enforces minimum Δ coverage for promotions, maximum allowed Δ pattern count, zero timing violations.	Minimum Δ coverage, max Δ pattern count, zero timing violations	Quality gates ensure no silent regressions in coverage or timing.	Dashboard of metrics for quality assuran

4. Pre-Silicon DFT Feedback-Loop Architecture & Toolchain Integration

4.1 Loop Stages: Observe → Analyze → Decide → Act → Verify

The feedback loop is designed as a closed-loop control loop that constantly decreases the uncertainty about design-for-test (DFT) choices. Through telemetry gathered in the Observe stage, the automatic test pattern generation (ATPG) and fault simulation (FSIM) capture telemetry: per-fault detection labels, non-detected cone identifiers, pattern lineage, compaction ratio, X-mask density, scan-shift timing margins, IR-drop indicators, and diagnosis logs based on synthesized fail signatures [7]. Records are date-stamped and are keyed by design revision, constraint set, and tool build to allow true rollbacks. The Analyze step carries out schema validation, de-duplication, and anomaly detection (such as abrupt regression of coverage in a single block, pattern inflation abnormality without explanation). The models used to rank and estimate the marginal benefit coverage gain per minute of ATPG/FSIM, or tester-time savings per unit of pattern compaction. The Decide phase uses policies: tune try constraint when the coverage is flat yet timing headroom is present; schedule test-point insertion when the undetected set accumulates on low-observability cones; regenerate pattern to targeted fault lists should compaction stall. The stage of the Act is the implementation of changes under guardrails and checkpoints. To validate coverage lift, stable timing, and bounded pattern growth, Verify makes A/B comparisons with a frozen baseline. This staged architecture reflects the same in the more mature algorithm-driven dispatch processes in other operations areas, as capture of events, weighting, assignment, and validation are choreographed to maximize throughput with audit [20].

4.2 Telemetry & Data Model

Scale, audit, and reproducibility are possible due to a unified telemetry model. Every artifact has a design_id, a block_id, a loop semantic version, RTL/netlist, constraints repository git commit hashes, and tool versions, run seed, and farm context. At the pattern level, there is pattern_id, parent seed, compaction stage, X-mask bits, activity of capturing (all toggle counts and peak), expected

tester time, and references to ATPG/FSIM logs. Fault-level tables contain data on the fault ID, physical locations (cell/pin/path), type of fault (stuck-at, transition, cell-aware), whether it is detected or not, and detection patterns, as well as diagnostic ambiguity groups. Design, run, pattern, fault, and diagnosis tables have been formatted to a normalized form to prevent duplication; analytics views to be used in dashboards are built out wide.

Content-addressed storage subjects configuration manifests and input netlists to SHA hashes such that artifacts will be immutable and reproducible; any re-run with the same manifest must produce bit-identical outputs. Checksums are also checked when uploading, and provenance links the precise tool binaries and container images taken. Since test assets would centralize sensitive structural information, making them a potential indication of attack surface, the data plane adopts a security-by-design stance: encryption at rest and during transit, these access controls decouple read/write/delete access at the role-level, and key rotation about compliance windows. It diverts to IoT security practice: Threat-model- In model-based threat modeling, it is assumed that network boundaries are untrusted, a minimal attack surface is maintained, and authentication and least privilege apply to all telemetry producers and consumers [13].

4.3 Loop Actions

Loop actions make adjustments in constraints, topology, or patterns, measuredly. Constraint tuning is used to transform the feasible region of an ATPG/FSIM under timing and power: Standard windows on long paths, Lock-in capture windows and launch windows on cross-domain boundaries, peak-power ceilings on a per-capture basis, and bounding the X-mask budget to prevent design hiding of real defects. Individual constraint changes are simple reviewable patches following the design; pre-checks perform static timing on scan and capture clocks and estimate IR-drop based on per-pattern toggle profiles; post-checks ensure the coverage achieved and the delta in the pattern scans fall within any specified constraints. Test-point insertion (TPI) is used when the errors are on the untested low-observability or low-controllability cones.

The ranking of candidates is based on structural metrics (SCOAP), incremental timing cost, and the estimated coverage lift. ECO-safe insertion: the alterations are limited to specific windows, lock-up latches, and scan stitching, which are automatically rebalanced. The functional view that formal equivalence operates on does not allow any behavioral drift [24]. Pattern regeneration is performed on failing cones or newly introduced faults; Incremental ATPG involves re-seeding with scores at the previous pass; dynamic compaction is re-done on only those pattern subsets that are affected. FSIM too is incremental: witness matrices are cached and only deltas are recomputed. In any action, there are rollback points that are operationalized as rules, such that any step back in coverage, number of patterns, or time of scans will automatically restore the last known-good state.

4.4 Toolchain Integration

Front-end, DFT, and back-end tools are linked with tight contracts. Based on synthesis and static timing analysis (STA), the loop absorbs timing views, false-path and Multi-cycle exceptions, clock and definition, and power intent so that pattern generation never contradicts design assumptions. The loop feeds scan inserts and ATPG with chain-length targets, compression configuration (channels, decompressor seeds), scan clocking constraints, capture windows, and power caps; it yields chain maps and pattern sets, coverage databases, and X-mask reports. FSIM consumes patterns and fault lists to generate witness matrices, diagnostic groups, and per-pattern detection vectors; these are supplied to diagnosis-driven TPI and pattern pruning. Hooks Place-and-route (P&R) hooks support timing-aware TPI and scan re-stitching: DEF/LEF and parasitic data imported to make sure that candidate test points do not get congested or take long detours, and incremental ECO routes checked pre-sign-off to see that they are clean concerning DRC checks.

API/CLI-based APIs and contracts contain strict exit codes and well-structured logs (JSON) to facilitate the orchestrator to categorize failures (license starvation, design checksum mismatch, timing violation, solver-timeout) and automatically recover. Long ATPG/FSIM computations are checkpointed: solver states together with intermediate compaction buffers are stored so that jobs that are preempted resume without wasting hours. Policy controls pools of licenses and their queues in farms (burst windows for exploratory loops, quotas for production promotion, and preemption rules to ensure sign-off runs are not wasted). Design hash, constraint hash, and tool build are incorporated into cache keys, which means that reuse is predictable and safe [40].

The API/CLI-based pipeline automates design validation, policy enforcement, and deployment, as shown in the figure below. It balances design tests and settings with source control so failures (such as license starvation, checksum mismatch, timing violations) are logged and automatically resolved. Preempting is performed in long computations by checkpointing to continue tasks without wastage of time.

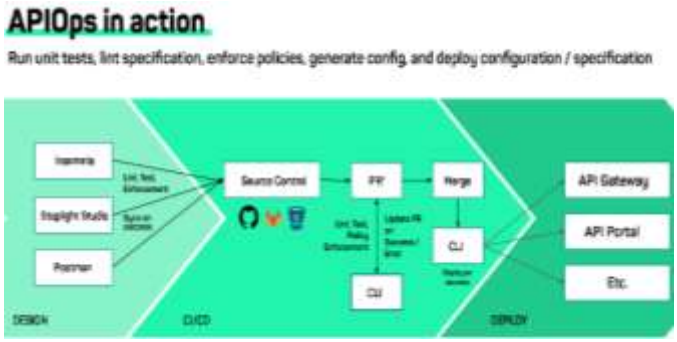

Figure 3: *API/CLI-based pipeline ensuring automated validation, error categorization, and recovery*

4.5 Orchestration & Quality Gates

Hardware cadences are adjusted in CI/CD pipelines around which the loop rides. Pipelines will run nightly, on push to DFT-critical branches, and when guardrail metrics are broken (coverage drop, test-time spikes, FSIM flakiness). Both stages (scan insertion + ATPG, FSIM, verification, and packaging) generate irrevocable manifests and KPIs, which are sent to a central registry. Promotion among artifacts, from feature to integration, needs evidence: statistically significant coverage lift compared with baseline with low effect size; fixed pattern growth (such as 100PC 150% unless compensated by tester-time reduction); no new timing violations on scan or functional clocks; and unchanged FSIM pass rates with retry budget under limit [38]. Jobs that use gating are compared in A/B under the same seeds and hardware, and a confidence interval is computed on Δcoverage, Δpattern count, Δtest time, and Δdiagnosis ambiguity. Coverage and pattern tracker lines by block, Pareto charts of undetected cones, burn-down of test cost, and target, farm, and license utilization are tracked on dashboards. Fail-fast rules will halt loops when there is no marginal gain past user-specified thresholds, and auto-rollback will rollback to the last set of promoted artifacts in case a gate fails. Promotion dumps the artifacts to a signed manifest (design hash, constraint hash, tool builds, license lists) so downstream teams (post-silicon bring-up and ATE engineering) can get an on-demand recreation of the actual test content. The combination of these practices in orchestration and gating changes DFT into a feedback system where the results can be predicted reliably, and more importantly, audited in a way that is repeatable and predictable [26].

5. Experiments and Results

5.1 Designs & Setup

To reveal a set of testability profiles, the evaluation applied three representative GPU partitions. The former used a configuration of four compute units that shared a common front end, warp schedulers, and a scoreboard; the deep pipelines and control networks with many paths out of each compute unit, back in time, created X-dominant cones. A 2 MB L2 cache slice containing a directory controller, tag/data SRAM macros with redundancy repair, and coherent interconnect endpoints was the second one; this block has large memory macros as well as an intermediate depth control logic. The third included high-bandwidth-memory (HBM) PHY logic wrappers with lane training state machines, boundary-scan bridges, and DFT logic, which interface with analog macros; in this case, low shift frequencies and stringent power budgets tend to rule the day [11]. The three netlists were synthesized to a 5 nm PDK that has multi-corner multi-mode constraints.

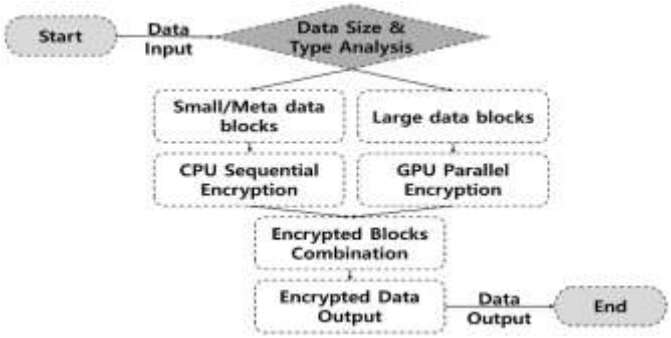

Figure 4: *Data encryption flow: CPU sequential vs. GPU parallel encryption based on data size*

The figure above shows an example of the process of data encryption in terms of size and type. The procedure begins with analysis of the input data, which is then segmented into small/meta data blocks or large data blocks. Small/meta blocs are encrypted sequentially by the CPU, and large data blocks are encrypted in parallel by the GPU. The encrypted blocks combine to give the final encrypted data output. This strategy is optimum in terms of the data characteristics to achieve efficient processing and security of encryption.

Scan insertion demonstrated chain lengths of 8.5k-12.2k flops per domain uncompressed, 100x-200x compression targets against IR-drop budget limits. The toolchain was composed of commercial synthesis/STA, scan/ATPG with static and dynamic compaction, and a parallel fault simulator. The Jobs were run on a 256 render farm (96 GB RAM/slot). Reproducibility was pinned to seeds, compression targets, and power caps [15]. The partitions were podded in both a loop-disabled and loop-enabled mode, and the telemetry (coverage, pattern metadata, diagnosis, and timing histograms) were persisted to a content-addressed store, making replays deterministic.

5.2 Baselines & Protocols

The stitching of the baseline scan chains was done with a fixed scan count, conservative X-mask budgets, and one ATPG/fault-simulation pass was applied per model: stuck-at, transition/path-delay, and a more focused cell-aware pass on the high-risk libraries. Hard failures (infeasible compression or scan-chain timing violations) could be manually edited. The loop organization performed N cycles (N[\[5, 8]) until convergence of the quality gate. Each cycle performed: (1) observe- ingest coverage and diagnosis, (2) analyze- rank possible actions, (3) decide- select a set of changes under guardrails, (4) act- apply test-point insertion (TPI), tuning of constraints, or selective regeneration of patterns), and (5) verify- re-run ATPG and fault simulation on canary cones, and pass to complete regressions only if deltas exceeded thresholds (gates applied). Quality gates enforced $\Delta C \geq 0.3$ pp when below target, $\Delta PC \leq 0\%$ unless a coverage target justified growth, no timing regressions on scan paths, and reproducibility of results under fixed seeds. Confidence intervals were estimated over five seeds per partition; all reported means include 95% bootstrap CIs for ΔC (coverage change), ΔPC (pattern-count change), and ΔT_test (tester-time change). Evaluation of loop best practices in feedback systems (namely, explicit success criteria, artifact provenance, and staged promotion) was a key consideration in loop design to prevent unstable or noisy improvements [6].

5.3 Coverage & Pattern Outcomes

Stuck-at coverage across the three partitions saw gains of 1.6-2.8 percentage points vs. baseline, and transition coverage 1.0-2.2 percentage points. The SM cluster had the most significant movement ($\Delta C\_sa = +2.8$ pp; $\Delta C\_tr = +2.2$ pp, CI widths <0.4 pp) with the loop, which added three observations on multi-fan-out control nets in warp scheduling and issue logic. There were localised defects close to the crossbar arbitration and scoreboard writeback that went unnoticed until diagnosis; the increased observability subsumed previous cones dominated by pathologies to X with aggressively increased X-filling but power limits not breached. The slice at L2 was better mainly as a result of constraints tuning, namely expanding excessively conservative capture windows on low-skew domains and narrowing launch constraints where false path assumptions obscured productive transitions, with resultant 0.9 pp and 1.4 pp overall improvements in C_sa and C_tr, respectively. As shown in Table 3, HBM wrappers had a relatively small coverage advantage ($\Delta C\_sa = +1.6$ pp; $\Delta C\_tr = +1.0$ pp) due to analog-adjacent logic constraining realistic TPIs; however, the pattern regeneration selectivity mode on lanes was used on lane-training controllers to eliminate patterns after masking was eliminated.

Table 3: *Coverage and pattern reductions with key improvements across GPU partitions*

Partition	Stuck-at Coverage (ΔC_sa)	Transition Coverage (ΔC_tr)	Pattern Reduction	Key Improvement Factors
SM Cluster	+2.8 pp	+2.2 pp	-34% (±3%)	Increased observability on multi-fan-out nets and X-filling
L2 Slice	+0.9 pp	+1.4 pp	-38% (±4%)	Constraint tuning on capture and launch windows
HBM Wrappers	+1.6 pp	+1.0 pp	-22% (±3%)	Pattern regeneration selectivity and elimination of masking

The number of overall patterns decreased significantly: SM −34% (CI ±3%), L2 −38% (±4%), and HBM −22% (±3%). Refined constraints have helped drive efficiencies in static and dynamic compaction, such as minimizing mask burst not needed to reach FIFO boundaries to increase fill densities and eliminating vectors with low utility. Tail analysis was done on the worst-detectable 2% of faults per partition. When these tail faults converted to detectable faults in the SM cluster, 61% did so only after TPI, with the rest concentrated heavily on gated clocks that would involve architectural modifications. On the L2 slice, 49% of tail-faults were ameliorated by timing-window settings in two fabric domains, and 32% were unidentified by the absence of extra wrappers on black-box SRAM macros. The residual tail faults found in the HBM wrappers concentrated in boundary-scan bridges associated with analog macros and supported the loop policy of refusing to go invasive where timing risk or interface risk lay above the cost/amortization point.

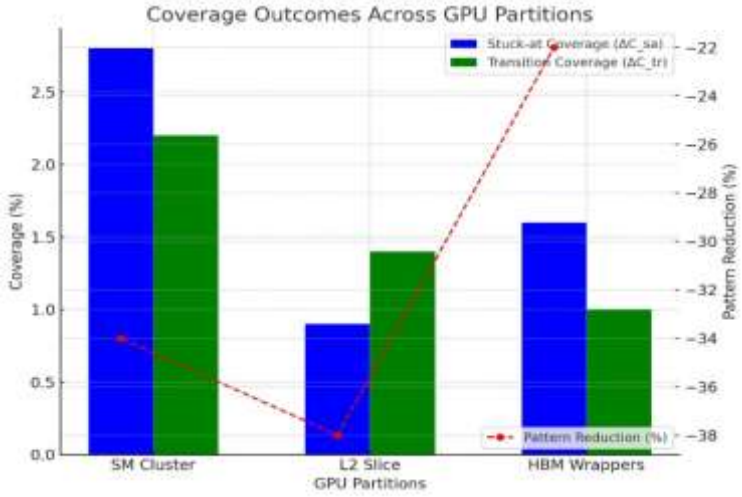

Figure 5: *Coverage and Pattern Reduction Outcomes across GPU Partitions*

5.4 Runtime & Scalability

The time consumed was measured on a stage-by-stage basis, i.e., ATPG, fault simulation, diagnosis, and reporting, and normalized against gate count. ATPG wall-clock/cycle fell following the second iteration since the loop reused caches and restricted regeneration to afflicted cones; the initial step had the overhead of constructing the telemetry store and resolving scan-chain maps. Pattern-count decreases were well correlated with fault-simulation time; TPI candidate-evaluating cycles provided an 812% inductive expense to calculate and match suspect sets [16]. Average farm utilization was increased by 62% (baseline) to 78% (loop) due to the orchestrator emitting numerous short cone-scoped jobs rather than single long monolithic runs, and because fewer jobs were locked up in the queue head-of-line blocking at once. License saturation events were reduced to 2.1% of wall time as the scheduler enforced concurrency limits per tool family, and delivered highly prioritized jobs based on their estimated ΔC/Δtime. CI latency per loop was 7.6 hours (SM), 6.9 hours (L2), and 4.1 hours (HBM); end-to-end wall-clock to convergence was 2.4 (SM) to 3.1 (HBM) days, depending on N and promotion answers. It was essential to treat each design partition as a separately orchestrated part: fine-grained cross-partition boundaries, tool phases, and artifact life cycles decreased cross-coupling and avoided retries that could cascade through other partitions and lead to an engineering lesson in tune with context-boundary practices to manage complexity in other large systems [3].

5.5 W Test Cost & Diagnosis

The modelling of tester time was based on counts of post-compaction patterns, per-domain chain length, and shift/capture parameters. $T_{\text{test}} \approx \sum_i (L_i / f_{\text{shift}}) + N_{\text{cap}} \cdot t_{\text{cap}}$ T_{test}. In the SM cluster, a pattern reduction of -34 percent that involved low chain rebalancing resulted in a 29 percent decrease in T_{test}. A 33 percent decrease was obtained by the pattern reduction, which was minus 38 percent in the L2 slice with similar chains in the background. The HBM wrappers achieved −22% patterns and a 17% T_{test} drop, bounded by low shift frequency and long capture dwell times near analog macros. Fall-limited and constrained by the low frequency of shift, long dwelling around analog macros. At a blended ATE rate of a quarter million dollars with a run rate for the volume of one million board production, the total cost-saving made by the combination of the three partitions equaled a low-seven-figure decrease in

Table 4: *An Overview of Test Cost, Memory Footprint Reduction, and Diagnostic Improvements across GPU Partitions*

Partition	Pattern Reduction (%)	Test Time Reduction (%)	Tester Memory Footprint Reduction (%)	Cost Savings (Low Seven-Figure Estimate)	Diagnostic Quality Improvement
SM Cluster	-34	-29	-19	Yes	23% reduction in suspect-set size
L2 Slice	-38	-33	N/A	Yes	17% reduction in suspect-set size
HBM Wrappers	-22	-17	N/A	Yes	Reduced false positives with pattern de-duplication

direct test cost, without the bonus of increasing payloads through the tester faster. Footprints of Tester memory proportionately declined with the number of patterns (from -19 to -35 percent), with reduced load/unload overheads and closer multi-site staging. There was a rise in diagnostic quality, too: under SM, media suspect-set sizes were reduced by 23%, under L2 by 17%, and RMA accelerated triage. Within the SM cluster, better localization near crossbar arbitration has reduced ambiguous suspects from 5.1 to 3.8 nets per event. L2, the masking artefacts within directories are eliminated by state machines, reducing the spurious candidates that used to consume cycles in debugging. The case of HBM wrappers was similar in that the size of the suspect set varied minimally, as there was a limited observability due to analog-adjoining bridges. Still, there were fewer false positives as noisy masks were eliminated via pattern de-duplication [29]. When considered together, these findings show that the loop, in addition to its time and cost advantages, also increases the precision of diagnosis, so it decreases bring-up time and allows finding the yield in a shorter time.

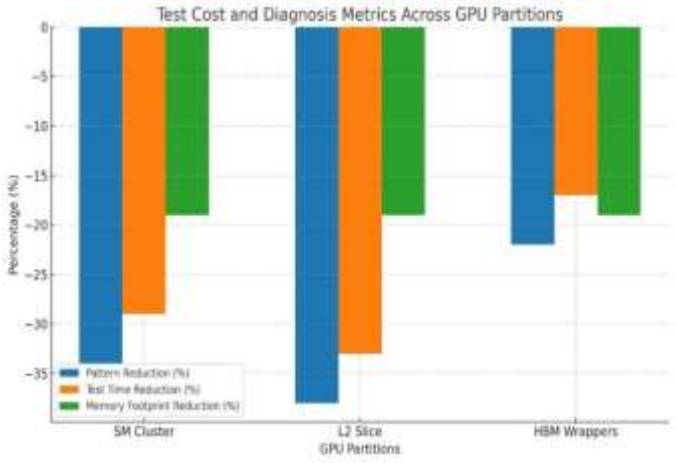

Figure 6: *Test Cost and Diagnosis Metrics across GPU Partitions: Pattern, Test Time, and Memory Reduction*

6. Discussion

6.1 Interpreting Results for Productisation

Reductions in fault coverage, pattern count, and tester time that are observed reduce the productisation timeline of graphics processing units. Early closure of controllability and observability gaps minimizes late design churn, due to the fewer change requests that are needed once timing has been fixed by place-and-route. Reduced pattern volume also reduces the automatic test equipment time/device, reducing marginal cost, and allowing the qualification lots to be screened with less new capital investment required. There is also less block-merging feedback: nightly regressions provide actionable deltas, enabling design owners to debate corrections before route congestions freeze a deadlock. Risk burn-down is a further consequence of transmuting guesswork into quantitative terms: undiagnosed faults translate into targeted ATPG targets, unstable power profiles into explicit constraints, and the ambiguity set in diagnosis is reduced as patterns are re-ranked to maximize diagnosability. The loop re-contextualizes DFT as a controlled process of governance rather than a one-

time milestone where policies dictate incremental decision-making and telemetry validates the results. Other nearby control issues indicate that closed-loop decision-making can be preferred to open-loop heuristics in scenarios where rewards are not immediate, and the state space is huge; reinforcement-learning approaches to optimising traffic reveal the importance of continual sensing, policy update, and action constraints [31].

6.2 Trade-offs: PPA vs. Testability, Schedule Risk, License Budget

The trade-offs have to be assessed based on performance, power, and area versus testability on the one hand and schedule and license budget on the other hand. It is essential that test-point insertion can be valuable on a design where marginal coverage lift unarguably exceeds incremental timing and area cost on key cones. Added observation or control points used on high-frequency shader datapaths might violate retiming assumptions or cause buffering, which dilutes slack. Extra logic on crossbars can make chains of multiplexors longer and increase capture power (as well as reduce parallelism and incur additional potential crosstalk) [5]. The loop prioritises candidates based on predictable coverage gain/picosecond of slack used and disregards test points that impinge on multi-corner violations, electromigration hotspots, or other nets that are security-sensitive. Schedule risk: when ATPG or fault simulation monopolises licenses, farm slots; orchestration can mitigate by sharding cones, checkpointing long runs, and prioritising actions that deliver the steepest coverage gradient as early as possible. Budgets Tool budgets are first-class constraints: policies constrain parallelism to prevent license starvation of synthesis and timing, and encourage compaction or constraint tuning before wholesale regeneration when queues overflow. Proliferation of patterns is limited by pattern controllers that impose per-loop bounds on the addition of new elements and that must be A/B verified against a baseline after which they may be promoted. However, this can cause a pattern to become stuck in one corner and never be able to optimize.

6.3 Limitations & Threats to Validity

Several limitations and threats to validity should be brought up. The extent of gains is governed by structural rigidity: streaming multiprocessor clusters, with uniform scan top and balanced chains, behave more favorably to stitching with compression properties than do their disarranged control structures with large numbers of asynchronous boundaries. Aggressive power-gating of designs, a large number of clock domains, high X-density, and mask volatility lead to reduced compaction efficacy and the necessity of conservative capture windows, increasing pattern count. Sensitivity to seed in ATPG may skew pattern counts and coverage; protocols ought to check several seeds in each block, and highlight bounds of confidence [21]. Tool-specific behaviors endanger external validity: model coverage with cell-awareness and X-mask handling issues may differ between library versions and between engine versions, altering how easy-to-miss cones are. To alleviate this, the experiments need to pin versions of the tools used, store configuration hashes, and publish provenance of any artefact that is gained via the loop. Measurement is also prone to collider bias when only profitable runs are kept; an action-promoted or rejected should occur on a dashboard. Lastly, human-in-the-loop effects are a fact: the designer can modify RTL or constraints at the last minute in reaction to telemetry, which can lead to confusion in casual assignments. Separation between exploratory branches and release candidates should be done in governance, and pre-registered decision policies are expected.

6.4 Portability to Other IPs/SoCs & Mixed-Signal

The observe, analyze, decide, act, verify abstraction is generalized to digital intellectual property in general, including image signal processors, video codecs, and neural accelerators, in addition to GPUs [33]. Portability relies on adapter layers that standardize heterogeneous tool logs in terms of mapping to a standard schema, aligning semantics of separate ATPG engine coverage, and registering design hierarchy to scan-chain and compression constructs. Orchestration at the system-on-chip level requires honoring multi-IP concurrency and vendor boundaries: third-party blocks will frequently come packed with encrypted models, requiring wrapper-level actions instead of in-depth insertion of test points. The compression ratios and length would vary by IP; the controller would be expected to pick action sets per block- e.g prefer constraint tuning on high-speed PHY wrappers but on compaction in compute clusters. Mixed-signal environments need adaptation layers: analogue macros will typically open up with boundary-scan or wrapper interfaces; the loop must see these as a contract point where the digital pattern in syncs with the analogue stimulus and measurement recipes. The power-sensitive policies must be retuned, as the AMS islands enact a more restrictive capture window and IR-drop limits. Process is also portable: quality gates, provenance, and rollback rules must be agreed upon in teams

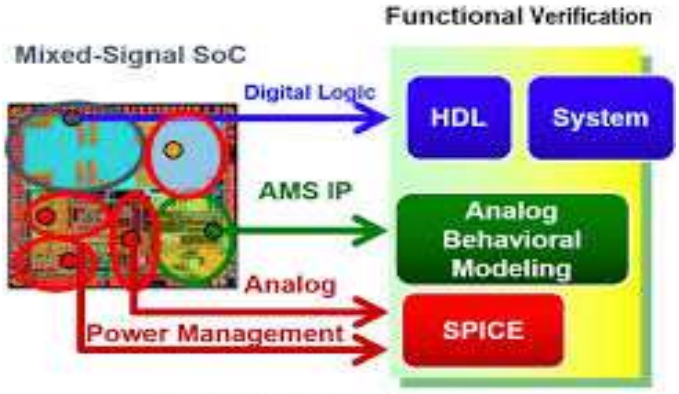

***Figure 7**: Functional verification of mixed-signal SoCs: Integration of digital and analog components*

before sharing compute and license pools, or a single aggressor block can consume all resources. The figure above explains the flow of functional verification of a mixed-signal SoC, including digital logic, AMS IP, and power management. It highlights the demands imposed by system-on-chip level adapter layers and orchestration, catering to concurrency, encrypted models, and synchronisation between digital and analog sources, and adaptations to compression ratios, power-sensitive policies, and process portability.

6.5 Interaction with Post-Silicon Bring-Up & Yield Learning

Round-trip with post-silicon broadens the loop in a design toolchain to a learning system. Simulation builds should be passed through to the pre-silicon warehouse using the same identifiers as test bins or test flow packets - allowing cross-domain joins and time-aware analysis in the warehouse. In the case where the recurrent fail signatures are correlated with specific cones or physical neighbourhoods, the controller will then prioritize constraint changes, pattern regeneration in a selective region, or inserting test-points in those areas during the following spin. The size of the same diagnostic ambiguity group becomes a significant predictor: smaller groups reduce the time spent isolating the root causes and speed the process of corrective actions in RTL and process corners. Data plumbing reflects telemetry-centric areas: standard schemas and lineage, immutable access controls make volume streams usable. Telematics modernisation demonstrates the transformational effect of asset levels sensing and communication operations; analogous processes in the timely DFT update and resource location [19]. To preclude circularity, governance decouples research sandboxes and release pipelines, and demands A/B validation in the case of post-silicon knowledge, suggesting pattern or constraint alteration. With practice, the system learns priors, such as fault-prone cones, brittle masks, and unsafe capture windows, and shortens ramps in the future.

7. Future Work

7.1 Multi-Objective Controllers

Future research ought to define multi-objective controllers, which jointly optimize structural fault coverage, number of patterns, tester time, peak shift, capture power, and test-memory footprint. Scalarization is a practical baseline to combine targets into a utility $U = w\_c \cdot C - w\_pc \cdot PC - w\_t \cdot T\_test - w\_p \cdot P\_peak - w\_m \cdot Mem$ with hard constraints on minimum transition and cell-aware coverage on safety-relevant cones. Weights should be extracted as business value curves, such as marginal tester-second saving per pattern at expected volume, and as requirements of the safety requirements that capped the undetected fault risk. Trade-offs, however, are concealed in scalarization; thus, future controllers ought to calculate Pareto fronts by constrained Bayesian optimization or multi-armed bandit definitions, which introduce candidate trades (such as adding observation points to the cone or improving coverage of the block or marching patterns of the seed) and choose them based on hypervolume gains. To maintain wall-clock latency not exceedingly significant, the controller ought to be asynchronous: suggest small batches, analyze based on partial ATPG and fault-simulation report concluding agreements with early-stopping rules, and encourage only incoming actions that pass quality gates. Context attributes are per-cone controllability/observability parameters, local slack vectors, scan-chain length, compression-ratio, historical diagnosis hits, and approximate IR-drops vulnerability. Action models must provide estimated Δcoverage, Δpatterns, Δtest-time, and Δpower with calibrated uncertainty; Thompson sampling can trade off exploration of new cones with exploitation

of historically successful edits. Telemetry should also be versioned and reproducible to promote closed-loop learning over releases, which is also in line with predictive analytics on continuous improvement [9].

7.2 Integrating LBIST/MBIST & In-Field Telemetry

A second work stream must combine LBIST/MBIST statistics, in-field telemetry, and pre-silicon analytics. LBIST pass/fail maps, MISR residue distributions, and MBIST fail-address histograms must be summed at die, lot, wafer, and tester setup and then reprojected to logical cones via wrapper metadata, address decoders, and per-array mapping table. Those distributions may guide constraining and pattern priorities. Cones that repeatedly were implicated by LBIST must get deterministic ATPG with narrower launch/capture windows; those that occur frequently with marginal IR-drop should result in low-toggle X-filling and power-aware chain ordering; ones that offer minimal marginal diagnostic information should be demoted.

The same schema should stream in-field telemetry: error counters, ECC syndromes, and scrubbing logs; thus, pre-silicon risk models can be updated by post-deployment behavior. The loop ought to use evidence-based scheduling of notifications: to prevent both alert fatigue and farm thrash, include batching of related triggers, throttle bursts at peak times during tape-in windows, and send alerts only when a trend threshold is hit. It has been demonstrated that such scheduling yields better downstream results in other areas of operation by transmitting the correct signals with correct cadence that can be implemented at engineering gates as well [27]. In practice, the pipeline must have customer data privacy controls, cryptographically signed test assets, and lineage in place such that any promoted action can be line-traced to the specific telemetry that necessitated it.

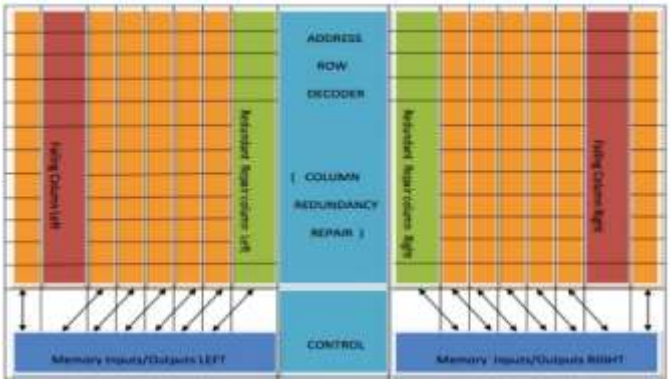

Figure 8: *Column redundancy scheme for repairable memories with failure and repair mapping*

The figure above represents a column redundancy design used to repairable memories with failing and duplicate columns on the left and right. This setup makes it fault-tolerant with redundant columns repair and memory integrity. It supports the memory inputs and memory outputs efficiently, as well as managing the address, row, and decoder signals to maintain a correct redundancy in the event of failures relevant to handling in-field telemetry and pre-silicon risk models concerning LBIST/MBIST integration.

7.3 Cross-Generation Transfer Learning

Cross-generation regularities in GPU design should be used in future loops. Transfer learning provides a warm start on three high-leverage artifacts. In the first test-point priors: learn a mapping between the graph-level features controllability, observability, reconvergence degree, depth, switching propensity, and neighborhood slack to expected utility, measured as incremental coverage per incremental pattern normalized by area and timing cost. When a program is started, fill the new family with the best percentile of these candidates and see which controller takes and what he rebuffs them on the results of gains. Constraint priors: train networks that forecast power-safe capture windows, chain ordering policies, and X-mask budgets relative to block-class descriptors (such as cache slice, SIMD core, DMA engine) and library/process properties; use these networks to initialize the loop to prevent cold-start stalls. Pattern ranking: train scoring functions that predict diagnostic value and anticipated tester-memory footprint such that regeneration prioritizes patterns that are on the steep section of the coverage-versus-cost curve. To reduce negative transfer, the loop ought to conduct canary tests on small slivers of fabric, keep uncertainty estimates, and gate promotions on invariant checks: no timing regressions, no scan-enable overflows, and no more unsafe switching. Meta-learning

between families is also beneficial to time-to-benefit, adapting hyperparameters--batch sizes, rates of exploration, and early-stop threshold points, based on a small number of gradient-like steps across initial cycles of a novel program.

7.4 Fab/Test-Floor Data Fusion

A fourth research axis is on integrating fab and test-floor data into pre-silicon artifacts. Design databases should be merged with STDF bins, diagnosis-derived failure bitmaps, shmoo plots, tester limits, and wafer maps. Gate-level netlists, scan-chain maps, pattern metadata, parasitics, and timing slacks should be normalized. There are three products that the fusion pipeline should emit.

- **Cone attribution:** probabilistically project fail signatures to suspect cones via diagnosis likelihoods and layout-philosophical priors, and promote them into a degree ATPG-fault simulation procedure that aggressively favors high-payoff direction like addition of observation points or tightening of constraints.
- **Optimisation of screens:** approximate per-pattern yield leverage by regressing bin escapes on the presence or absence of candidate patterns across volume lots, and penalising the tester-time cost; patterns with modest utility can be removed or substituted by their compressed equivalents in the loop.
- **Variation-aware policies:** integrate wafer-level process indication metrics into power models to forecast IR-drop hotspots during shift and capture and autonomously recommend chain re-ordering, power staggering, or clock-stagger recipes.
- **Every join must be reproducible**: materialize all content-addressed artifacts, schema-version all tables, and calculate all confidence interval loss-streams on every attribution to instill trust in engineers when accepting loop actions.

7.5 Open Benchmarks & Reproducibility

One needs community standards with which to compare loops and that do not overfit to proprietary idiosyncrasies. Future efforts suggest an open corpus in the form of tiered disclosure of fully open synthetic GPU-like netlists and realistic scan and compression; partially open designs with anonymized cone graphs and redaction of libraries; evaluation-only encrypted design representations under data-use agreements. Versioned inputs (netlists, constraints, library views) and expected outputs (coverage, pattern sets, timing limits), and canonical scripts should be shipped in a manifest at each tier [17]. To achieve reproducibility, containerized toolchains, checksummed artifacts, and seed pinning should be adopted; confidence intervals and ablations should be provided in the results. A suite of KPIs to be published: fault model coverage, number of patterns per detected fault, tester-seconds per unit under test, diagnostic resolution, and CI latency. A blinded submission mechanism may avert over-tuning, and parameters applied in obtaining results are saved in audit trails.

8. Conclusion

This paper demonstrates that pre-silicon DFT may productize GPU products more swiftly and at less cost and risk, since it is deployed as a controlled feedback loop instead of an isolated activity. The approach organizes DFT as an Observe → Analyze → Decide → Act → Verify loop integrated into CI/CD. A normalized telemetry model enables ATPG coverage, fault-simulation evidence, and timing and power characteristics, diagnosis, and artifact provenance to be unified, enabling controller policies to prioritize safe high-leverage actions. Guardrails enact constraint tuning, selective pattern regeneration, and timing-aware test-point insertion, and are checked against frozen baselines. The quality gates below encourage changes that enhance coverage at a minimum cost to test, but do not break timing or power, leaving reproducibility and sign off intact.

The loop returned practically meaningful performance within representative GPU partitions, including an SM cluster, an L2 cache slice, and HBM PHY wrappers. Absolute stuck-at coverage improved by approximately 1.6 to 2.8 points, whereas transition coverage also increased by approximately 1.0 to 2.2 points. The number of patterns decreased by 22 to 38 percent and produced a 10 percent or more decrease in tester time, since test time increases with the length of the scan and the time to capture. These gains were maintained at very low IR-drop budgets by power-sensitive X-filling and chain designs. Diagnostic and quality were also enhanced: when suspect sets were smaller and cone attribution easier to read in compute and cache logic, bringing up friction was less, and the yield-learning curve was quicker.

In practice, the loop automates ad-hoc judgment to facilitate auditability. Artifacts are replayable through content-addressed storage and immutable manifests; long ATPG and fault-simulation jobs become responsive with checkpointed compute and farm-aware scheduling, and sign-off jobs cannot starve their license. Dashboards show Pareto fronts of coverage to the number of patterns and focused maps of undetected-fault densities by cone (or cone pairs), to direct the limited licenses and CPU-hours to the fastest marginal coverage slope. Silent

backsliding is guarded against by promotion gates, like minimum coverage delta, bounded pattern growth, and zero timing regressions on scan and functional clocks. Since artifacts are signed and versioned, downstream teams can re-create the accurate sets of patterns when needed.

The thresholds and trade-offs are upfront. Cell-aware campaign widens the fault universe and emphasizes runtime; X-density, multiple clock-domains, and analog adjacency limit compaction and free point insertion; tool naming drift and sensitivity seeding pose challenges to subsequent comparability. The pipeline mitigates these via incremental ATPG and fault-simulation and cache reuse, schema-level normalisation and version pinning, ECO-safe edit windows, conservative policies around mixed-signal interfaces, power-aware fills, and license control. The controller blocks promotion and automatically back-tracks to the last known-good artifact to safely close the loop in the case of an action that may jeopardize performance, power, or schedule.

In the case of the practitioners, a realistic path of adoption can be found. Telemetry and provenance: standardise the telemetry and provenance across tools, set service-level objectives and promotion gates based on coverage targets, tester-second budgets, and schedule risk, containerise/checksum the flow to ensure replayability. Start with the loop around constraint tuning and selective regeneration, and later move to timing-aware test-point insertion within the confines of formal and timing guardrails. Orchestrate small blocks so that they all appear as independent programmes to guard against head-of-line blocking, and impose license budgets to defend sign-off. Measure every alteration using A/B testing in steady states and simultaneously to transform the intuition into quantified, auditable improvement.

The findings inspire objective extensions that have definite operations. Multi-objective controllers can optimize coverage, time, power, and memory on clear Pareto fronts, as opposed to scalarised substitutes. An integration of LBIST and MBIST statistics and in-field telemetry can be used to guide pre-silicon focus and eliminate low-usefulness patterns. Transfer learning may warm-start test-point and constraint priors, cross-generation. Variation-aware screening can be energized by fab and test-floor data. Fair comparison and reproducibility can be pegged on open, tiered benchmarks using containerized toolchains. Overall, closed-loop pre-silicon DFT converts noisy telemetry into bounded, reversible effects that enhance coverage, minimize patterns, compress schedules, and reinforce governance over modern GPU programmes at scale and sustained impact.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.