



# Cognitive smoke testing

Nikhil Singla, Rohit Jindal, Pandithurai Sangaiyah  
Debarati Banerjee, Krish Desai

*Abstract- As SOC release cycle shrinks, prompt identification of design bugs is essential. Pre-submission checks and initial smoke testing are standard industry practices to prevent bad code from entering the main codebase. Current pre submission checks rely on static test lists which require manual updates and lack comprehensive coverage. This results in undetected design bugs being merged, which could have been caught by other tests which are not part of sanity tests. Early bug detection has always been worked upon to left shift the chip cycle. We're introducing 'cognitive smoke testing' which not only facilitates the early bug hunting but also reduces the compute demand for verification. This intelligent system automatically identifies RTL( Register transfer language) design changes and selects the precise tests needed to validate those changes. The main motive is to run the tests which are impacted by the changes rather than running a static list of tests to catch the bug even before it is merged in the database. Dynamically selecting the tests every time saves compute and license cost significantly. If the design changes require test bench modifications, provision is provided to embed those changes on top of design which ensures only sanitized design changes end up in the central codebase. This solution once implemented is autonomous and flags the bug in root itself. Cognitive smoke testing provides all the necessary hooks like priority, mandatory list, time limit, compute limit and bypass option to cater to user needs and ensure backward compatibility.*

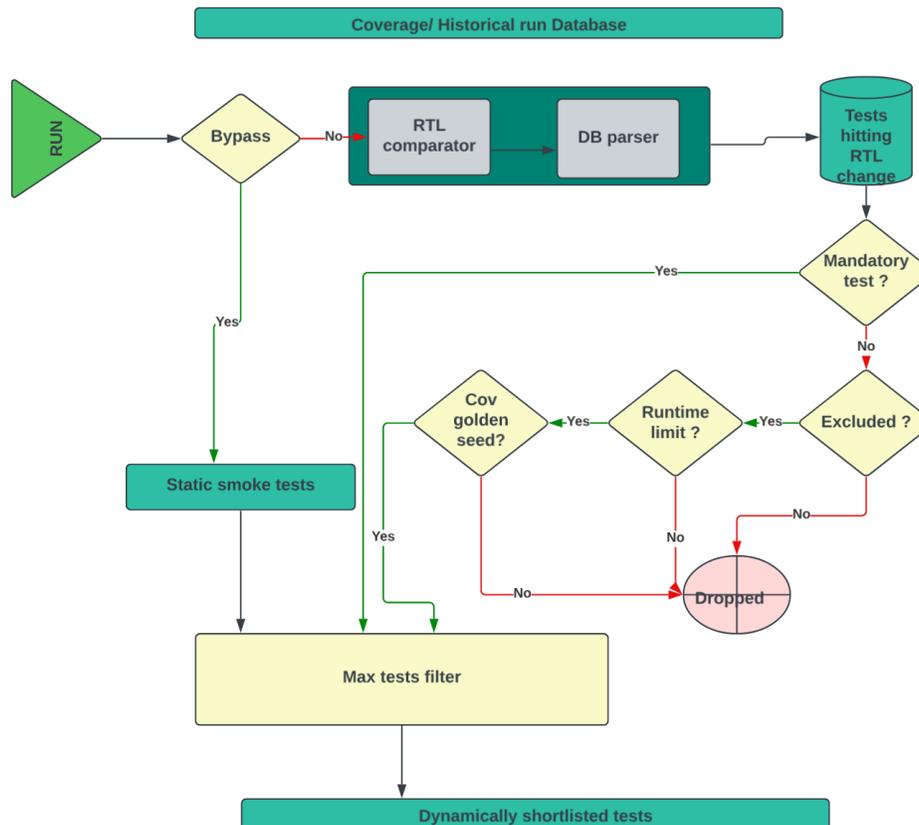
## I. INTRODUCTION

One of the major problem verification team faces is erroneous design changes, due to static smoke tests, which get stable with time and generally don't target the comprehensive design changes and keeping full test suite as sanity is not feasible. Debugging these wrong merges is very tricky as its iterative process to figure out which change broke the codebase. The idea behind smoke testing was to keep acting as a gatekeeper to validate the changes at root itself but static tests are not justifying the purpose. The proposed solution is smart to detect the changes done and figure out which tests contribute to the changes and run those in smoke testing rather than running the static ones. Also, most of the times its observed the changes are small and running a full list is not giving good ROI. As the cognitive smoke testing tool is dynamic, it reduces the compute and runtime for smaller changes significantly. Multiple knobs are provided to users like time limit for testing, priority of tests, coverage grading which gives a different dimension to smoke testing. The solution is architected keeping scalability as main focus. It supports all verification flavours functional ,UPF, GLS and is compatible with all the simulators.

Solution uses quick access database coverage to list down the potential tests ( targeting the design changes) and then multiple layers of filters like time limit, test count limit and test grading are used to select the final smoke test list which has the highest ROI of breaking if there is a design miss. Cognitive smoke testing is a real gatekeeper for maintaining a healthy codebase. Overall flow of the cognitive smoke testing solution is illustrated below

- Coverage database from historical runs along with the test list
  - Synopsys natively supports this with few extra switches
  - Cadence tool chain requires one extra correlation stage to be run on the coverage database to dump the test info
- Script to convert raw coverage to fast access database which lists the RTL nodes and tests covering those
- Design changes parser which lists the design changes and dumps in format compatible to fast access database

- Test list decoder selects the tests to run for max coverage of design changes . It houses multiple decision making elements which user can control as per user requirements
  - Time limit: user can set the maximum time all to complete the sanity, only tests fitting in that time limit will be shortlisted
  - Compute limit: memory and core limit can be capped
  - Mandatory list: user can mandate few tests to be shortlisted even if it's not targeting the design changes
  - Priority : coverage, runtime, memory footprint
  - Bypass: if bypassed, static list will be used
- Run with the shortlisted tests



## II. APPLICATION

A versatile Cognitive smoke testing solution is already deployed on multiple projects having a huge code base ( 10M code lines) . This is compatible with all version control systems like git, SVN. The nature of the solution makes it suitable for any code base not limited to just HDL languages. By making the tests dynamic, it significantly shifts the design issues by early detecting the issues which saves the tedious job of root cause analysis.



Currently the solution is built on git which has all the necessary hooks to port it to any other version control system. Also the APIs are coded in such a way that overhead for users to shift from native tool to proposed solution is bare minimum.

### III. RELATED WORK

Few EDA vendors are exploring change aware regression but the solutions are very heavy and tightly coupled with their tool chain. Most of the solutions are targeting tweaking the constraints to generate new test cases which complicates the implementation as the tool needs a lot of learning to correctly predict the modification and is error prone. The proposed solution is straightforward and targeting the area which is pain point for verification engineers without any overhead of integration. The flexibility of the solution makes it user friendly and keeps the testbench independent of the simulator, tool chain and version control tool.

### IV. RESULTS

The idea is proven on multiple testbenches ranging from IP to SOC. . It has helped multiple DV teams to catch bugs in the very initial stage which saves huge debug time.

