Faster Coverage Convergence with Automatic Test Parameter Tuning in Constrained Random Verification

Qijing Huang, Hamid Shojaei, Fred Zyda, Azade Nazi,
Shobha Vasudevan, Sat Chatterjee, Richard Ho
Google Inc.
{hqjenny, hamids, zyda, azade, shovasu, schatter, riho}@google.com

Abstract

Constrained random verification (CRV) in industrial settings involves manual parameterized test generation, a costly and inefficient process. We formulate test parameter configuration as a blackbox optimization problem and we introduce Smart Regression Planner (SRP), an approach that automatically configures the tunable test parameters to better explore the input space and accelerate convergence towards coverage. The optimizer in SRP can drive the parameters update with two methods: a light-weight random search, and a Bayesian optimization technique that uses coverage from nightly regressions as feedback. Our experimental evaluation on open-source as well as larger industrial designs demonstrates that frequent perturbation and optimization of test parameters leads to higher coverage than the human baseline. Importantly, it converges to coverage milestones significantly faster than the human baseline. With high-level test parameter optimization, we introduce a problem space and an opportunity to achieve categorically higher coverage in industrial settings with very low overhead.

1 Introduction

Constrained random verification (CRV) is the de facto standard in industrial design verification. Central to this process is the design of an elaborate testbench that applies pseudorandom stimulus to the design-under-test (DUT) downstream. The testbench typically consists of *parameterized tests* that are manually crafted for verifying functionality. Each parameter acts as a high-level knob to control stimulus generation, and the testbench then generates a family of related stimuli based on these configurable parameters.

Coverage that determines the comprehensiveness of tests is recorded after each regression. Typically, coverage can be computed by taking the percentage of the tested code segments or variable values over a predefined set of target code segments and values. Coverage holes found in a nightly regression are addressed by changing parameter configurations, adding new parameters and/or new tests. This verification process is followed iteratively until 100% coverage is achieved.Test parameter configuration is critical to the coverage. In the RISCV verification platform [1], we find that random perturbation of the human-defined test parameters results in ~60% coverage difference between the best and worst configurations.

LATTE '21, April 15, 2021, Virtual, Earth

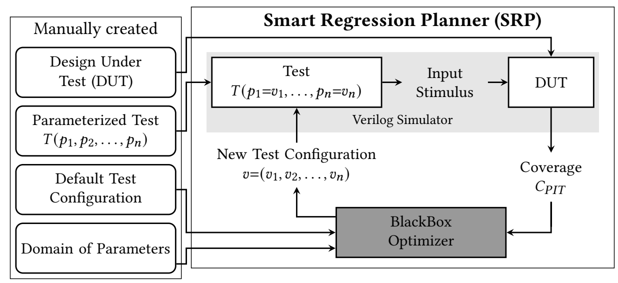

Figure 1: The Smart Regression Planner (SRP) framework.

Verification engineers rarely explore the large space of parameters systematically. In this paper, we investigate the value of *automatically configuring test parameters* towards increased coverage. We argue that fine-grained parameter tuning provides a unique opportunity for increasing functional and code coverage with *no* additional effort from the verification engineers beyond setting up the system.

We introduce Smart Regression Planner (SRP), an approach to configure high-level test parameters with the goal of quick coverage convergence. Traditional research on input stimulus generation is at the Boolean input level, instruction (or transaction) level, or at the constraint level [2, 3, 5]. Searching the space of input stimulus directly suffers from combinatorial explosion. In contrast, SRP works at a higher level of abstraction that naturally has much fewer configurable inputs (< 100); nonetheless it directly impacts verification coverage. The more tractable input dimensions allow the application of powerful optimization methods to this problem.

In SRP, we formulate test parameter configuration as a blackbox optimization problem with an objective to maximize coverage. We first employ a simple random search (random perturbation of parameters) to configure test parameters as a baseline. We then apply ML-based Bayesian optimization methods that can leverage coverage feedback from past regression tests and learn near optimal parameter configurations. Bayesian optimization [7] is agnostic to structure and flexible enough to adapt to changes in an evolving design. While random search relies purely on exploration, Bayesian optimization exploits learning through feedback. We also investigate use cases of (1) simultaneously minimizing runtime and maximizing coverage using multi-objective Bayesian optimization, and (2) transfer learning, or the ability to transfer learned heuristics from one set of parameters to another through design evolution.

2 Our Approach: Smart Regression Planner

Fig. 1 shows our SRP Framework for improving coverage closure in simulation. In our flow, the test uses the parameter configuration provided by the blackbox optimizer to generate inputs for the DUT.

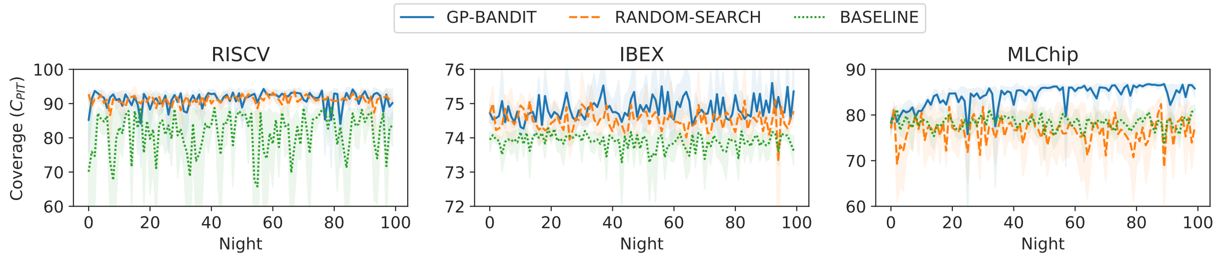

Figure 2: Each line represents the mean coverage across five random seeds and the shaded region shows the standard deviation across the five runs. More exploration leads to higher maximum point-in-time coverage (C_{PIT}). GP-Bandit+Baseline consistently achieves higher C_{PIT} than Baseline+Baseline on all designs over.

The point-in-time coverage C_{PIT} (a real number between 0 and 100%) is computed by the Verilog simulator that simulates the test and the DUT, and this value is fed to the blackbox optimizer.

The optimizer then generates a new value v_i for each test parameter p_i from its valid parameter domain which is fed back to the test. Test parameters can be numerical, categorical, ordinal, or Boolean. The test and the design are re-simulated with the new configuration and return the new C_{PIT}.

We formulate regression planning as an optimization problem. The goal is to find $v^* = \text{argmax}_v f(v)$, where v^* is test configuration that maximizes the C_{PIT}. Since the function f, that maps a test configuration v to C_{PIT} does not have any obvious structure that may be exploited for optimization (such as convexity or smoothness), it is natural to consider blackbox techniques. In SRP, we use Gaussian Process Bandits (GP-Bandit) [7] that models coverage results f as a Gaussian function of the test configuration. GP-Bandit uses Bayesian optimization technique, where it maintains a Gaussian prior $GP(\mu, \Sigma)$ over f and updates it with samples drawn from f to get a posterior that better approximates the objective. This algorithm performs exploration and exploitation technique to choose a test configuration.

3 Experimental Results

We evaluate SRP on two sets of designs: open-source (RISCV [1], IBEX [6]) as well as a larger industrial design MLChip. In RISCV, there are in total 15 ordinal test parameters, each with 10–40 categories. IBEX contains 31 test parameters. 16 of them are categorical with two classes while the rest are the same ordinal parameters from the RISCV testbench. In contrast to general-purpose RISCV/IBEX design, MLChip follows the CISC tradition for its custom instruction set architecture (ISA) design. The test parameters for MLChip includes many distribution specifications for the instructions and the test vector values. We evaluate the performance of SRP with respect to both Random-Search and GP-Bandit vs the Baseline which is the human-generated tests with fixed parameters.

3.0.1 SRP+Baseline Flow To benefit from both exploration introduced by SRP and exploitation from low-variance baseline setup with human-specified parameters, we propose a new use case for regression testing by running SRP in addition to the original Baseline flow. Instead of running Baseline multiple times with different random seeds every night in real CRV deployment, we propose to run GP-Bandit in SRP with Baseline.

We merge the coverage for the two runs and report it as the coverage C_{PIT} for every iteration. In Fig. 2, we see that this mode ensures that C_{PIT} **driven by GP-Bandit+Baseline for almost every night is higher than the Baseline+Baseline on all designs**. This makes the SRP+Baseline mode a highly attractive proposition for practical settings. Contrarily, Random_Search+Baseline does not provide any such assurance. Its exploration is quite expansive and frequently falls below the baseline. Note that the coverage does not reach 100% in the experiments as it is infeasible to run the number of tests till convergence every night.

3.0.2 Multi-Objective Optimization Optimizing for high coverage can sometimes lead to unacceptably high simulation runtimes. An engineer might want to trade off one for the other at different points in the verification phase. We explore multi-objective optimization (MO) in Bayesian optimization [4] to simultaneously minimize simulation runtime and maximize coverage. Our experiments shows that adding the multi-objective optimization leads to 1.18× **speedup in the mean runtime while achieving higher mean coverage over 200 nights.**

3.0.3 Transfer Learning in SRP A typical use case is the addition of new test parameters and design features as the design evolves. Instead of re-training the blackbox algorithms in this case, we investigate the ability to transfer learned heuristics and improve sample efficiency. In the transfer learning experiments, we held out 5 of the 11 parameters during initial optimization, then added them back to simulate new parameters being added to the test. Our study shows that the C_{PIT} **coverage with transfer learning starts higher and converges around 20 nights earlier** than the runs without transfer learning, showing the promise of this technique.

4 Future Work

In this work, we have formulated a verification problem capable of significant practical impact, at an abstraction level where scalability is a natural byproduct. We have found that algorithms like GP-Bandit that use coverage feedback can further improve the coverage with faster ramp up and less variance. This work opens up many research directions like the application of graybox/whitebox optimization techniques in the future for test parameter optimization, the feasibility of a continuous learning ML-based verification paradigm through the lifetime of the design, and the transfer of optimization heuristics to new designs. Given meaningful feedback, similar ML-driven optimizations can be applied to automate other hardware verification tasks, such as test case selection. A more direct objective to maximize would be the number of bugs detected.

References

[1] 2020. *SV/UVM based instruction generator for RISC-V processor verification.* https://github.com/google/riscv-dv

[2] Markus Braun, Shai Fine, and Avi Ziv. 2004. Enhancing the efficiency of Bayesian network based coverage directed test generation. In *Proceedings. Ninth IEEE International High-Level Design Validation and Test Workshop*. IEEE, 75–80.

[3] Shai Fine and Avi Ziv. 2003. Coverage directed test generation for functional verification using bayesian networks. In *Proceedings of the 40th annual Design Automation Conference (DAC)*. 286–291.

[4] Daniel Golovin et al. 2020. Random hypervolume scalarizations for provable multi-objective black box optimization. *arXiv preprint arXiv:2006.04655* (2020).

[5] Zdenek Kotásek et al. 2015. Automation and optimization of coverage-driven verification. In *2015 Euromicro Conference on Digital System Design*. IEEE, 87–94.

[6] Pasquale Davide Schiavone, Francesco Conti, Davide Rossi, Michael Gautschi, Antonio Pullini, Eric Flamand, and Luca Benini. 2017. Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 1–8.

[7] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995* (2009).