Multi-Objective Genetic Test Generation for Systems-on-Chip Hardware Verification

Adriel Cheng Cheng-Chew Lim

The University of Adelaide, Australia 5005

Abstract We propose a test generation method employing genetic evolutionary algorithms, and multi-objective optimising strategies that achieve Pareto optimality. The generated tests enable high coverage for the design under test and use low computational resources. The benefits of the technique are demonstrated by its application for system-on-chip verification.

Keywords multi-objective optimisation, Pareto optimality, genetic algorithm, test generation, hardware design verification

1 Introduction

Efficient test generation is the key to achieve high quality hardware design verification. In system-on-chip (SoC) hardware verification, the test generated must be comprehensive to exercise many SoC functions to expose bugs. The progress and comprehensiveness of testing is measured in terms of coverage from the design under test.

To enhance verification effectiveness, several coverage measures are employed to examine how the SoC is tested from different perspectives. Line coverage, toggle coverage and conditional coverage are some metrics used. High coverage results from multiple coverage metric implies more thoroughly verified SoCs overall.

Coverage facilitates effective fitness objectives in the genetic evolutionary algorithm (GEA) test generations. Previously, our GEA test generation was limited to a single objective only [Cheng and Lim (2007)]. Different coverage metrics were used to drive GEA test generation in separate evolutionary processes. In this paper, the multiple GAE processes, which are driven by specific verification goals, are amalgamated into a single test generation flow. In the test selection, we derive a technique that uses Pareto front sorting, aggregrate ranking and round-robin selection strategies. Rather than simply covering the hardware design test space extensively as in many existing methods, targeted test objectives or restrictions can be explicitly incorporated. The approach expands the applicability of the GEA test generation method to cater for wider range of verification challenges.

2 Problem Description

Test generation is an iterative cycle whereby tests are created based on components and verification results of previous test cycle. The goal of test generation is to create

efficient tests that exercise many SoC functions to expose bugs whilst using least number of SoC operations to reduce test sizes every cycle. More specifically, test generation must not only create tests that simultaneously maximise the hardware design's line coverage, toggle coverage and conditional coverage, but also ensure that the size of these tests are kept as small as possible. We formulate the multi-objective goal as an optimisation problem (P) as

$$\begin{aligned} \max_{x \in X} \quad & \{f_l(x), f_t(x), f_c(x)\} \quad \text{and} \quad \min_{x \in X} \quad \{f_s(x)\} \\ \text{subject to} \quad & f_s(x) \leq M, \end{aligned}$$

where x is a test from X which represents the input test space of the test generation, $f_l(x)$ is the line coverage fitness function of the SoC when exercised by the test x, $f_t(x)$ is the toggle coverage fitness function, $f_c(x)$ is the conditional coverage fitness function, $f_s(x)$ is the function that evaluates the size of the test, and M is the maximum size available to hold a test for execution.

3 Approach

We solve the multi-objective test generation problem as three sub-problems: (i) incorporation of multi-objective optimisation into the test generation flow and verification platform, (ii) handling of conflicting objectives when solving the multi-objective optimisation problem (P), and (iii) identifying a variety of tests that cater for different objectives, and retaining them for the next cycle to undergo the genetic and evolutionary variation.

3.1 Test generation and verification platform

We employ genetic and evolutionary algorithms (GEA) for generating tests. The GEA process offers two major attractions: It iteratively generates, in the same way as biological evolutionary processes, new tests by evolving the previous test suite over many cycles. It offers a natural way to optimise multiple objectives for successful evolution.

Figure 1 provides an overview of the multi-objective GEA test generation process. It allows the incorporation of multi-objective optimisation into the test generation flow and verification platform. The GEA population is represented by the test suite. The GEA process begins by creating an initial population of tests and measuring their initial fitness performance for each objective.

Using these initial tests, variation is conducted. The variation operations are carried out on individual tests. Once variation has created the next population of tests, their fitness is evaluated. Fitness evaluation is conducted concurrently for all the objectives to ensure efficiency and prevent bottlenecks.

Using fitness results, population selection chooses which tests to retain for the next evolution cycle. Population selection must interpret multiple objectives, and establish the selection criteria by which tests are retained to optimise all objectives simultaneously. Preserving a diverse population is essential [Tamaki *et al.* (1996)] for the GEA process to continue evolving tests that caters for all objectives in subsequent evolutions.

The termination assesses whether fitness improvements for any objectives are still possible. Otherwise, the GEA process stops.

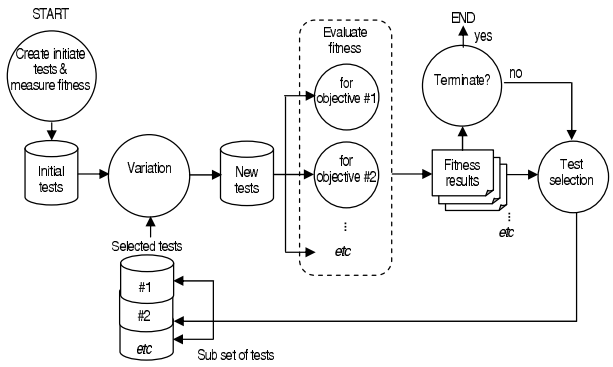

Figure 1: Multi-objective GEA test generation flow.

3.2 Handling conflicting objectives

The challenge with the optimisation problem (P) arises from the presence of conflicting objectives. We handle conflicting objectives by grouping these objectives into different subsets. Objectives are examined to determine if they compete against one another. An objective is considered to compete against another objective if an increase in optimality for one objective leads to a decrease in optimality of the other objective. Specifically, if the GEA optimisation of an objective's fitness causes degradation in fitness of another objective, then both objectives are deemed to compete against one another.

In the test generation problem, we partition the conflicting objective subsets as

$$L = \{f_l(x), f_s(x)\},\ T = \{f_t(x), f_s(x)\},\ C = \{f_c(x), f_s(x)\}. \tag{1}$$

In practical terms, these subsets imply that the three coverage types do not conflict with one another, but each is in conflict with the test size objective. Thus, optimising one particular type of coverage does not directly impinge on the fitness of other coverage objectives. Optimising conflicting objectives such as coverage versus test size is beneficial for SoC verification. It aims for maximum bug detection via high coverage, and minimum testing time and resources through small test sizes.

3.3 Test and population selection

The goal of population selection is to identify a variety of tests that cater for different objectives, and retain them for the next evolution to undergo variation. Using such diverse selection of tests, the variation phase can then create new populations that optimise greater number of objectives concurrently.

With the conflicting objectives partitioned, the test population sort and the test selection proceeds in three phases, as follows.

3.3.1 Phase 1: Pareto optimal sorting

Using the current test population, for each objective subset, a hierarchy of Pareto optimal fronts are created [Pareto(1896)]. Each hierarchical front groups together tests that provide different objectives fitness values, but are considered equivalent. This is

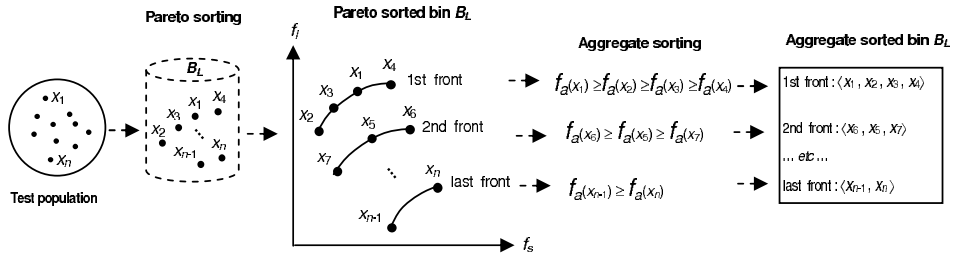

Figure 2: Pareto sorting and aggregate sort ranking for objective subset L.

because a test within a front that is superior in one objective does not provide improvement for any other objective fitness when compared to any other test within the same front.

Because the Pareto sort optimisation orders the test population into a hierarchical structure, each front captures a set of trade-off tests, separated into different levels. The first front contains tests that simultaneously optimise all objectives most effectively, whilst the last front is considered the worst optimised. Each hierarchy frontal set of tests achieve different levels of compromised fitness for conflicting objectives. This allows the test selection process to choose tests from different levels, thus promoting population diversity. After Pareto sorting, the tests within each front may be considered equivalent from a Pareto optimality perspective because tests from the same front do not dominate each other.

We produce multiple hierarchical sets of Pareto fronts and multiple orderings of tests from the same test population - one hierarchical set of Pareto optimal fronts per objective subset. We denote each set of Pareto optimal fronts as a Pareto sorted bin of tests. Each Pareto bin provides a different view of the test population, and reveals the progress of the optimisation for each set of conflicting objectives.

Figure 2 shows the Pareto optimising phase for subset L with its sorted bin B_L. The test selection makes use of the Pareto fronts derived from the Pareto sorted bin to ensure that the best tests for subset L are retained for future evolutions. The same procedure is applied to T and C.

The output from each of these sub-GEA Pareto processes will be combined later in the test selection phase to continue overall multi-objective optimisation in subsequent evolutions. To ensure the best tests from each Pareto front are selected, aggregation is employed to sort these frontal tests further.

3.3.2 Phase 2: Aggregate sort ranking

Aggregate ranking aims to combine the fitness of conflicting multiple objectives of each objective subset. Tests can then be sorted according to a common goal rather than any single objective. Aggregation is employed to sort tests within each Pareto optimal front.

The aggregation of L is obtained by evaluating the function

$$f_a(x) = \frac{f_l(x) + \frac{M - f_s(x)}{M}}{2},$$

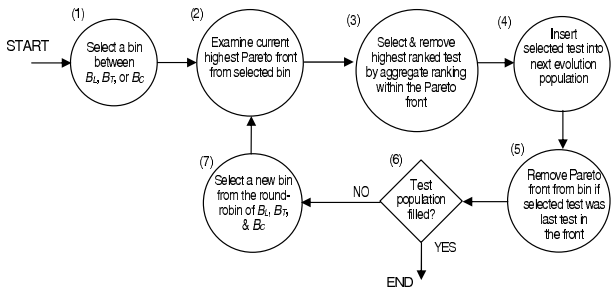

Figure 3: Test selection procedure.

which averages f_l and f_s and takes into account the size of the executable memory in the simulator or hardware tester. The aggregation for T or C is similarly obtained.

The aggregate ranking for subset L is shown in Figure 2. Every hierarchical Pareto front is aggregated sorted. Tests with a higher aggregate ranking will have higher likelihood of selection in the test selection phase described next.

3.3.3 Phase 3: Round-robin test selection

The Pareto and aggregate sorted bins of tests (B_L, B_T and B_C) are used to select tests for the next evolution population. Figure 3 shows the flow of the test selection process. We use the round-robin method to select a bin in Step 1 and Step 7 of the procedure.

Using this selection strategy, the highest aggregate ranked tests from the highest Pareto optimal frontal set is selected preferentially from each bin. Once a test is selected, it is removed from the Pareto front. In this way, the best trade-off tests as ordered by the sequence of Pareto optimal fronts are chosen. Also, given the aggregate rankings within each front, the best performing tests for all conflicting objectives are also chosen first.

By dividing objectives into different subsets and sorting tests within each bin according to various objectives, diversity in the test population is also enhanced. The tests selected will vary greatly from an objectives optimisation perspective. The selection method also promotes fairness. A test is selected from each bin every iteration using round-robin sharing amongst these bins. Tests are given equal selection priority from each of the objective subset bins.

4 Performance Evaluation

The multi-objective GEA test generation is evaluated against three test creation methods: (i) SAGETEG [Cheng (2009)], which employs single objective GEA test generation. (ii) µGP test generator [Corno *et al.* (2003)], an assembler instruction based test generator that also employs single objective genetic evolutionary method. (iii) Random constraint-biased test generation method. Each method is configured to perform similar verification on our target hardware design, the Nios SoC, using the same test composition library, if applicable, and similar test setup. We compare their coverage and test size results.

Table 1 shows the coverage results. Multi-objective GEA surpasses other test generation methods for line and toggle coverage. These improvements are attributed to the greater range of SoC design state storage values that arise from exercising other types

	Multi-Obj GEA	SAGETEG	µGP	Random
Line coverage %	99.5	98.9	97.5	89.9
Toggle coverage %	95.7	93.7	87.1	79.2
Conditional coverage %	83.3	83.0	72.4	69.3
Av. test size (bytes)	66,435	71,137	29,239	63,591

Table 1: Coverage and test size results.

of functionalities; which are also driven by the concurrent attainment of other coverage objectives such as conditional coverage during GEA.

Table 1 also gives the test size results. It shows that multi-objective GEA testing outperforms single objective GEA test generation of SAGETEG. However, average multi-objective test size is larger then individual line, toggle and conditional coverage test runs from µGP and random testing. This is not unexpected. µGP genome and test generation approach relies on assembler instruction programs, which are more space efficient than high level software based test programs of SAGETEG or multi-objective GEA. In the random approach, the test composition building blocks and test programs are created by stochastic means only, and do not increase in size significantly when it strives to attain higher coverage. On the other hand, GEA test programs evolve and grow incrementally using snippet genome in order to seek greater coverage.

Comparing with existing test generation methods, the multi-objective GEA scheme provides higher or equivalent coverage across all coverage metrics, and utilises much lower test sizes. The only downside observed was that processing multiple objectives causes lower coverage attainment rates. This is an area for potential improvements in the multi-objective GEA technique. The GEA selection and variation can be revised to more efficiently manage multiple objectives, and with parallelism, our multi-objective GEA process can be exploited to enhance the speed of the overall verification.

References

[Cheng and Lim (2007)] Cheng, A. and Lim, C. C., (2007), Test generation for system-on-chips using genetic algorithms, *7th Intl. Conf. on Optimization: Techniques and Applications* (ICOTA7), Kobe.

[Cheng (2009)] Cheng, A., (2009), *Verification of system-on-chips using coverage driven genetic evolutionary test techniques*, under preparation, Ph.D. Thesis, The University of Adelaide.

[Corno *et al.* (2003)] Corno, F., Cumani, G., Reorda, M. S. and Squillero, G., (2003), Exploiting auto-adaptive µGP for highly effective test programs generation, *ICES2003: 5th International Conference on Evolvable Systems*, Trondheim, pp. 262-273.

[Pareto(1896)] Pareto, V., (1896), *Cours D'Economie Politique*, vol. I & II, Lausanne.

[Tamaki *et al.* (1996)] Tamaki, H., Kita, H. and Kobayashi, S. (1996), Multi-objective optimization by genetic algorithms: A review, *Proc. IEEE International Conference on Evolutionary Computation*, Nagoya, pp. 517-522.