Optimizing System-on-Chip Verifications with Multi-Objective Genetic Evolutionary Algorithms

ADRIEL CHENG AND CHENG-CHEW LIM

School of Electrical and Electronic Engineering
The University of Adelaide
Adelaide, Australia, 5005

ABSTRACT. Verification of semiconductor chip designs is commonly driven by single goal orientated measures. With increasing design complexities, this approach is no longer effective. We enhance the effectiveness of coverage driven design verifications by applying multi-objective optimization techniques. The technique is based on genetic evolutionary algorithms. Difficulties with conflicting test objectives and selection of tests to achieve multiple verification goals in the genetic evolutionary framework are also addressed.

1. Introduction. Verification of hardware designs such as system-on-chips (SoC) is a common bottleneck in every chip design project [1]. The difficulty with design verification arises not only from increasing design complexities, but other verification goals and constraints imposed by testing of the design itself. Current goal orientated test strategies emphasize verifications that are typically driven by a single coverage measurement, which measure the progress and comprehensiveness of testing.

To enhance SoC verification, besides test coverage, other verification objectives should be *simultaneously* optimized with one another to drive verification procedures such as test generations. This paper proposes multi test objectives driven SoC verification. The ability to cater for multiple goals simultaneously within the same verification process is facilitated by optimal control of test generations that are based on multi-objective genetic evolutionary algorithms (GEA).

Our GEA test generation can be driven by several different criteria, each capturing specific verification goals. This exploits the parallelism available from multi-objective optimization; by amalgamating multiple test processes into a single GEA flow. Extending from conventional coverage driven verifications, using a feedback control strategy, the test generation can be directed by more than one coverage measure and other test objectives concurrently.

The success of multi-objective GEA verification depends on how objectives are incorporated into the test generation flow and the test platform. When multiple objectives are simultaneously targeted, a number of these objectives may be conflicting. For instance, in verification, coverage and test size objectives compete against one another during the GEA process. This is because higher coverage requires excitation of more design functions, which result in greater test size. Vice-versa, a smaller test size usually restricts the extent of testing on the SoC, giving lower coverage. Besides developing optimization control techniques for verification of hardware designs, another contribution of this paper is to propose and acquire solutions that address difficulties with conflicting objectives.

In multi-objective GEA test generations, tackling multiple conflicting objectives presents the most difficulties. Like [2,3,4], we consider multi-objective GEA hardware verification as an optimization problem, and we propose techniques for representing and manipulating multiple objectives in the GEA

optimization. Current strategies for handling multiple objectives are classified into two categories: aggregation and Pareto methods [5].

Aggregation combines the fitness values of multiple objectives together into a single fitness value. A range of aggregation methods has been proposed, each with varying success. For example, fitness values can be added based on differing objective priorities assigned by the user, or simply summed together and averaged out amongst the number of objectives. Jakob et al. [6] assigned specially devised weights to objectives and summed their fitness values accordingly. Goal attainment combines fitness values depending on how objectives satisfy certain goals [7]. Whilst it is easy to apply, the downside with aggregation is that certain objectives may dominate the GEA optimization process, leading to other objectives acquiring low fitness.

Pareto optimizing strategies for fitness assignment and GEA test selection was proposed by Goldberg et al. [8]. Their approach was to sort and rank all test solutions into different Pareto optimal subsets of tests, known as *fronts* [9] and then perform test selections. Extensions of Goldberg's method have continued. These include ranking the solutions within each front based on the number of other solution it dominates [10]. In [11], Pareto optimal solutions are employed for multi-objective optimization in the modelling of welfare economies. Like [11], the difficulty with large Pareto processing of objectives is tackled in [12]. We tackle Pareto optimization of large processes such as SoC verification, but difficulties with conflicting objectives are also examined.

Other schemes that extend Pareto techniques are the strength Pareto approach [13] or methods that combine tournament selections. The drawback with these methods is that much larger test populations are needed to provide a suitable sample set of solutions [14]. Our multi-objective GEA approach extends current methods by combining both aggregation and Pareto optimal methods to exploit their advantages and compensate for each other's shortcomings.

The remainder of the paper is as follows. The next section presents the problem statement and the multi test objectives driven verification technique. Section 3 describes how multiple conflicting test objectives are managed. The GEA test population selection method that optimizes multiple verification goals is described in section 4. Experimental results are presented in section 5, before the paper is concluded in section 6.

2. Overview of technique. The key to our multi test objectives driven verification technique is to encode the test generation as a multi-objective GEA process. Verification goals are posed as objectives of the test generation to be optimized concurrently according to the problem description below.

2.1 Problem formulation. In the context of semiconductor design verification, the multi-objective optimization problem is formulated as:

P1:
$$\max_{x \in X}\{f_l(x), f_t(x), f_c(x)\} \quad \text{and} \quad \min_{x \in X}\{f_s(x)\},$$
$$\text{subject to} \quad f_s(x) \le \mathrm{M} \quad \text{and} \quad f_l, f_t, f_c = (0,1)$$

where x is a test from X which represents the input solution 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 (i.e. binary signal) 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 realizable to hold a test for execution. In SoC verification, different types of coverage objectives measured between 0 to 1 are employed to examine the comprehensiveness of the testing process; whereby a value of 1indicates full coverage. □

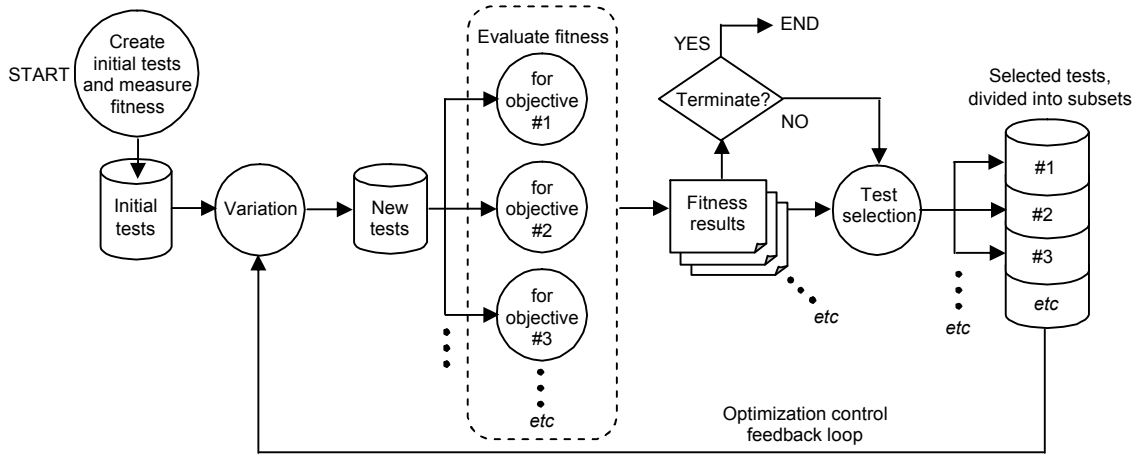

FIGURE 1. High-level multi-objective GEA test generation flow

To find a solution for P1, GEA was chosen. The test generation is encoded into the GEA process by (i) representing the test suite as the GEA population, (ii) assigning a test program to act as an individual chromosomal solution, and (iii) mapping the test building blocks to be the genome set that makes up each chromosome test individual.

The test building block genome consists of *snippets* of software functions [15]. Many different permutations of these snippets are selected and combined in a GEA manner to generate test programs for conducting SoC design verifications.

Figure 1 shows the high-level flow of the multi-objective GEA test generation. The GEA process begins by creating an initial population of tests and measuring their initial fitness performance for each objective. Variation is conducted using these initial tests to create new test variants. The variation operations carried out on individual tests are to add, subtract, mutate, and replace new test building blocks to existing tests. Recombination variation combines two or more existing tests to create new tests.

Once variation has created the next population of tests, their fitness is evaluated in the same way as the initial test population. Fitness evaluation is conducted concurrently for all objectives. Using fitness results, population selection chooses which tests to retain for the next evolution cycle.

Population selection is the most critical stage of the multi-objective GEA flow because it must interpret multiple objectives. The selection scheme establishes the selection criteria by which tests are retained to optimize all objectives simultaneously. Preserving a diverse population is essential for the GEA process to continue evolving tests that caters for all objectives in subsequent evolutions. In Section 3 and 4, we describe a novel selection scheme that classifies tests into different subsets according to conflicting objectives criteria.

Throughout the multi-objective GEA process, the optimization feedback loop (see Figure 1) allows for test cases with best fitness results to be fed back to the test variation phase. This controls generation of new tests that ensures required functionalities and critical areas of the SoC are continually verified.

In multi-objective GEA, the goal of the GEA process is to seek a set of optimized solutions that exhibit the best possible trade-off performance for all objectives. The successful creation of an optimized set of tests relies on tackling two sub-problems:

(i) SP1: how multiple conflicting objectives are encoded and managed, and

(ii) SP2: how diverse tests are selected during the test generation process.

3. Conflicting objectives.

3.1 Problem formulation: Consider two objective functions f_1 and f_2.

SP1: If $(f_1(x_1) > f_1(x_2) \rightarrow f_2(x_1) < f_2(x_2)) \vee (f_2(x_1) > f_2(x_2) \rightarrow f_1(x_1) < f_1(x_2))$, where x_1 and x_2 are arbitrary tests applied to maximise (or minimise) both f_1 and f_2 simultaneously, then f_1 and f_2 are said to be in conflict against each other.

In the GEA optimization, such conflict objectives cause slow convergence to a solution. □

3.2. Partitioning conflicting objectives. The slow convergence of a GEA solution is due to one objective's fitness causing degradation of another objective. The method we propose to speed up the process is to partition the multiple objectives into subsets of objectives that explicitly conflict with one another.

This partitioning approach enables the aggregated fitness from each set of conflicting objectives to be considered directly, rather than taking on the entire set of multiple objectives all at once during the GEA process. It further reduces the likelihood of any one objective dominating another during optimization. Greater opportunities are also presented for objectives to be optimized from a Pareto optimal viewpoint, as described later in Section 4.

The partitioning of conflicting objectives into subsets is governed by their relationship with other objectives as described in sub-problem SP1. After identifying conflicting objectives, these objectives are grouped together whilst non-conflicting objectives are separated from each other. The process of segregating objectives into subsets is termed *specialisation*. Within each subset, objectives are considered specialised, to compete directly against other conflicting objectives in the same subset. Under the context of SoC verification, the definition for objectives specialisation is as follows.

Definition 3.1. Specialisation of objectives: A specialised subset of objectives contains objectives that are in conflict with one another. For multi-objective GEA test generation, the objectives are specialised into three conflicting objective subsets as follows: $L = \{f_l(x),\ f_s(x)\}$, $T = \{f_t(x),\ f_s(x)\}$, and $C = \{f_c(x),\ f_s(x)\}$, where f_1 and f_2 from sub-problem SP1 may be represented by any of the f_l, f_t, f_c, or f_s objectives fitness functions. □

Remarks:

(i) These objective subsets imply that the three coverage metric types for lines, toggles, and conditions do not conflict with one another, but each is in conflict with the test size objective. The three coverage metric types are not independent of one another, but optimizing one particular type of coverage does not adversely impinge on the fitness of other coverage objectives.

(ii) The slow convergence of optimal solutions for conflicting objectives in SP1 is improved by specialisation of objectives into different subsets. By grouping and only considering conflicting objectives within each subset, the GEA process gives higher priority toward optimizing these groupings of conflicting objectives quickly, before tackling the entire set of multiple objectives. Once these subsets of conflicting objectives are processed, the population selection phase (Section 4) can then proceed with selecting tests that perform best for each subset.

Note that the above definition and subsequent discussions in this paper are applicable in general to more than two conflicting objectives. In this paper, we focus on two conflicting objective functions at a time to simplify the discussion and illustration.

4. Multi-objective test selection sub-problem.

The second sub-problem to address when introducing multiples objectives into the GEA verification flow is how to select tests that optimizes multiple objectives simultaneously.

4.1 Problem formulation:

SP2: Given a test population of size n, find the set of solution points $\{x_1, \ldots, x_n\}$ from the solution space X that maximises (or minimises) the set of m multiple objective functions $f = (f_1, \ldots, f_m)$. □

Following on from the specialisation of conflicting objectives presented in Definition 3.1, we propose to conduct a three phase test population sort and selection using these subsets. The three phases are: (1) Pareto optimal sorting, (2) aggregate ranking, and (3) round-robin test selection.

Figure 2 shows the overall flow of multi-objective GEA test selection with the three phases embedded. We now describe the three phases.

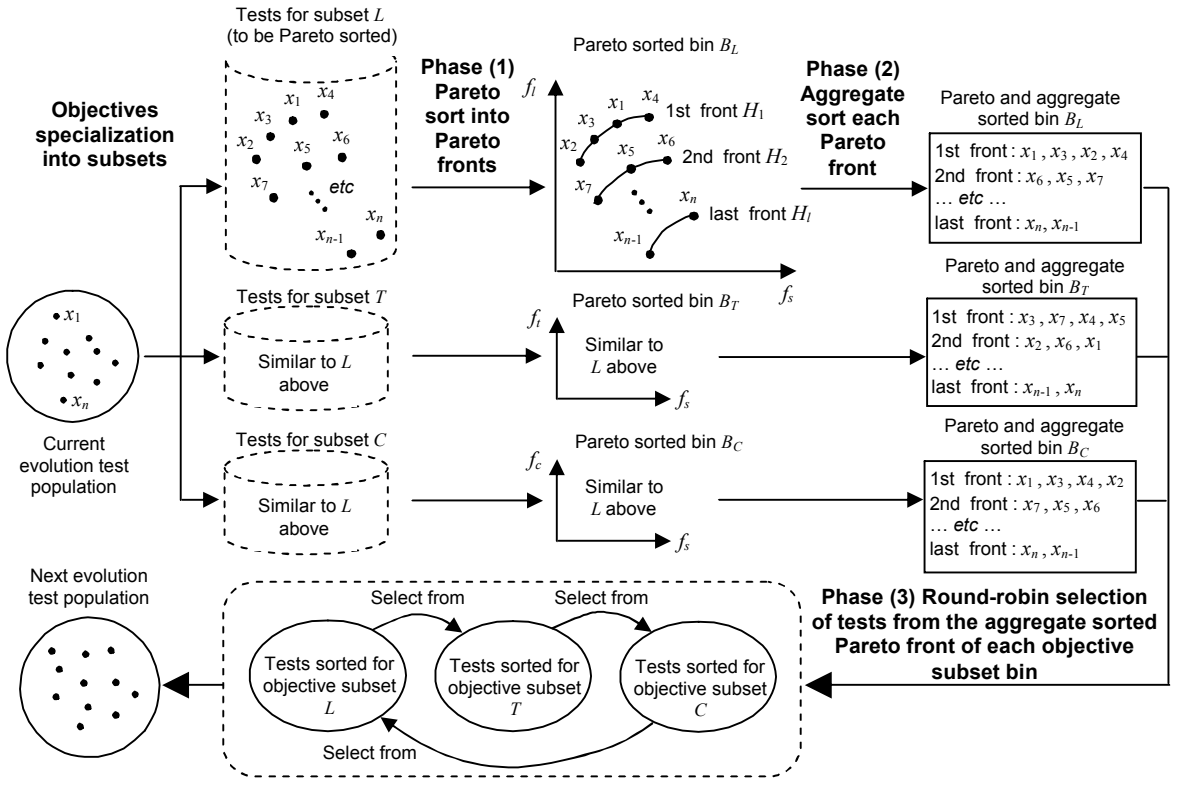

FIGURE 2. Three phase multi-objective GEA test population selection

4.2 Phase 1: Pareto optimal sorting. In the Pareto based method, the goal is to evolve GEA solutions into a Pareto optimal test population. A Pareto optimal population contains the set of tests which are the optimization solutions such that additional improvement in any one objective requires trade-off in performance of other objectives [9].

The set of Pareto optimal tests are considered non-*dominated*. Tests are non-dominated if they are not inferior to any other tests with respect to the conflicting objectives targeted. All Pareto optimal tests exhibit better performance in at least one objective, and attain at least the same performance for all other objectives. We now define Pareto optimality and Pareto optimal fronts in the context of multi-objective optimization.

Consider n solution points $x_1, \ldots, x_n$ in the solution space X, where the solution points in X is applied to maximise (or minimise) the set of m multiple objective functions $f = (f_1, \ldots, f_m)$.

Definition 4.1. Non-dominated set: A solution $x_1 \in X$ is said to dominate another solution $x_2 \in X$, denoted as $x_1 \succ x_2$, iff $\forall\, i \in \{1, \ldots, m\} : f_i(x_1) \geq f_i(x_2) \wedge \exists\, j \in \{1, \ldots, m\} : f_j(x_1) > f_j(x_2)$. □

Definition 4.2. Pareto optimality: A solution $x \in X$ is said to be *Pareto optimal* if there does not exist any other solution $x_k \in X$ such that x_k dominates x. That is, $x \succ x_k \; \forall \; k \in \{1, \ldots, n\} \wedge x \neq x_k$. □

Definition 4.3. Pareto solution mapping function: Let p: $P(X) \rightarrow P(X)$ be a mapping function that identifies the set of Pareto optimal solutions to form a Pareto front according to Definitions 4.1 and 4.2, where $P(X)$ is the power set of X.

Given $A \in P(X)$ and $B \in P(X)$ subsets, the function p is defined as $B = p(A)$ such that $\forall \; x \in B$ and $\forall \; y \in A \backslash B$, $x \succ y$. That is, p identifies and populates B with non-dominated Pareto optimal solutions from A. □

Definition 4.4. Pareto optimal fronts: Let $\langle H_1, H_2, \ldots, H_l \rangle$ be the ordered tuple of l hierarchical Pareto optimal fronts that can be formed from the set of solutions in X according to the multi-objectives function f. The *hierarchical set of Pareto fronts* are defined as $\langle H_1, H_2, \ldots, H_l \rangle$ such that $H_1 \cap H_2 \cap \ldots H_l = \varnothing \wedge H_1 = p(X) \wedge H_i = p(X \backslash H_{i-1})$ for $i = 2, ..., l$. □

The Pareto optimal fronts $H_1, H_2, \ldots, H_l$ contain disjoint set of solutions from X. The first Pareto optimal front H_1 contains the best solutions that dominate all other solutions in X. The second Pareto optimal front H_2 contains solutions which dominate other solutions not in H_1 and H_2. The last front H_l contains remaining solutions that do not dominate any other solutions, and are dominated by all other solutions in $(l–1)$ earlier Pareto fronts.

To explain, we consider a two-objective optimization process of maximising the coverage metric (f_1) against minimising the test size (f_2). Figure 3 shows the Pareto plot where each solid dot represents a solution point. The Pareto optimal solutions are located toward the top left region of the graph. By joining the Pareto optimal solution points together in a line, a wave-like front is formed (refer to Front 1 in Figure 3). This Pareto optimal front dominates all the other solutions to its lower right.

The GEA procedure aims to steer the Pareto optimal front toward the top left portion of the graph. The best compromise solutions amongst the two objectives are displayed along the line forming the best Pareto front 1.

Besides one Pareto optimal front, multiple Pareto optimal fronts in a hierarchical order can also be plotted (i.e., Fronts 2 to 4 in Figure 3). By excluding the current best Pareto optimal set of solutions (i.e., in Front 1), the remaining subset of solutions can be re-examined to identify another new Pareto optimal set (i.e., Front 2) as per Definitions 4.1 to 4.4. This then acts as the next lower hierarchical Pareto optimal front.

Further Pareto optimal set of solutions and fronts can be identified by repeatedly excluding the previous best Pareto optimal solutions and applying Definitions 4.1 to 4.4. This provides a series of Pareto optimal fronts as shown in Figure 3, whereby the first front represents the best Pareto optimal set. Identifying the series of Pareto optimized fronts facilitates a sorting process for the population, which is described in the next sub-section.

With Pareto optimization, certain characteristics are desired when creating the first (best) Pareto optimal front. These characteristics are:

(i) The Pareto front for each GEA evolution should expand towards the upper left region as more evolutions are conducted to optimize the population.

(ii) The larger the expansion achieved by the Pareto front, the more diverse the population will be. Diversity is important because it provides greater sample of solutions to choose from.

In Figure 2, the test population is ordered to produce different Pareto fronts depending on which objectives were used to perform Pareto sorting. A test may perform well for a set of conflicting objectives but not effectively for another set of objectives. For example, tests x_1, x_2, x_3 and x_4 perform well for the L and C objective subsets, and are placed within the first front of B_L and B_C (where B_* holds a *bin* of tests). However, x_1 and x_2 do not optimize the T objectives as effectively, and are placed in the second Pareto front of B_T.

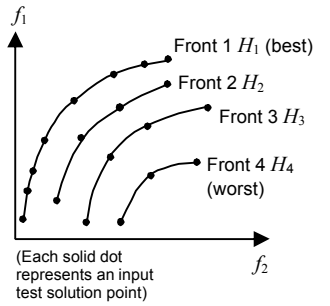

FIGURE 3. Example Pareto plot

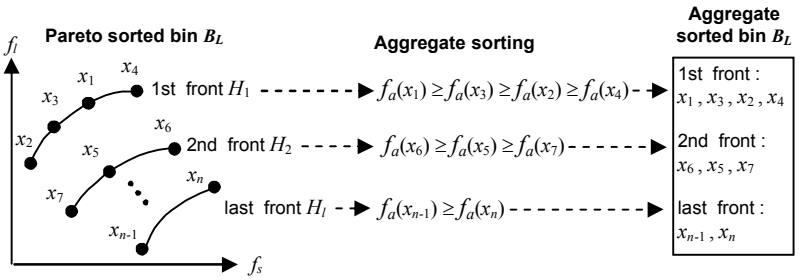

FIGURE 4. Aggregate sort example for objective subset L

The test selection makes use of the Pareto fronts from each Pareto sorted bin to ensure best tests that cater for each objective subset are retained for further optimization in future evolutions. The output from each of these sub-GEA Pareto processes will be combined later in the test selection phase 3 to continue overall multi-objective optimization in subsequent evolutions.

By Pareto optimizing the test population into a hierarchical structure, each front within a bin captures a set of trade-off tests, separated into different levels. Each hierarchical frontal set of tests achieves different levels of compromised fitness for conflicting objectives. This allows the test selection process to choose tests from different levels, thus promoting population diversity.

4.3 Phase 2: Aggregate sort ranking. Aggregate ranking combines the fitness functions of multiple objectives into one function. In maximising coverage and minimising test sizes, the function for a test individual $x \in X$ is chosen to be

$$f_a(x) = f_m(x) + \frac{\mathrm{M} - f_s(x)}{\mathrm{M}} \tag{2}$$

where f_m is either the line, toggle or conditional coverage fitness function, f_s is the test size fitness function, and M is the maximum test size allowed. The set X is the input set of possible tests created by multi-objective GEA.

With (2), the test size objective fitness $f_s(x)$ undergoes a transformation such that a smaller test size provides greater contribution to the aggregated fitness value $f_a(x)$. For example, figure 4 shows the sorting of tests from aggregate ranking for the subset L. The tests in every hierarchical Pareto front are ordered so that each test is given a ranking according to their aggregated fitness objective value calculated from (2). The larger the aggregated fitness value, the higher the ranking assigned to the test. Tests assigned with a higher aggregate ranking will have higher likelihood of selection in the test selection phase described next.

4.4 Phase 3: Round-robin test selection for next population. In the final phase, the GEA test selection process makes use of the Pareto and aggregate sorted bins of tests to select tests for the next evolution population. Tests are selected in a round-robin manner, cycling between the three sorted B_L, B_T and B_C bins of tests.

4.5 Summary. Using the Pareto based test selection strategy, the highest aggregate ranked tests from the best Pareto optimal frontal set is selected preferentially from each bin. Therefore, the best trade-off tests as ordered by the sequence of Pareto optimal fronts are chosen. Given the aggregate rankings within each front, the best performing tests for all conflicting objectives are chosen first. Low performing tests for any

objectives will not be selected from any bin, and is released from the GEA process.

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 optimization perspective. The resultant test population will be effective for optimizing the entire set of objectives.

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.

The goal of the Pareto and aggregation phase is to rank tests so test selection can be conducted according to different subsets and priority of certain objectives. It addresses a common issue (described in [16]) whereby ranking skews tests to focus on certain conflicting objectives only, but ignores other objectives. Our approach alleviates this problem by identifying subsets of conflicting objectives and managing these smaller groups of objectives individually. This reduces the possibility of objectives being overlooked for optimization if they were all considered together at once.

By employing round-robin selection to select tests from the Pareto fronts of each objective subset, diversity of tests that caters for both conflicting and non-conflicting objectives is maintained. Round-robin selection also reduces the likelihood of one objective dominating another.

5. Experiments. The test generation method was implemented as a test generation tool. It is integrated into our verification platform [15] to create tests for the Nios SoC [17], an industry-standard programmable chip.

In order to calibrate the verification platform and select test generation operating parameters, an analytical technique based on Markov chain modelling of the GEA test creation procedure was undertaken. Our Markov analysis method [18] was employed specifically to model the GEA variation flow of the test generation. Test generation parameters were chosen to maximise the likelihood of desired test creation characteristics

Our experiments are presented in two sections. Pareto and test diversity characteristics of the multi-objective GEA test generation process are examined in Section 5.1. Section 5.2 assesses the performance of our multi-objective test generation against other verification schemes.

5.1 Multi-objective GEA test generation characteristics

5.1.1 Pareto characteristics. Figure 5 shows the Pareto front plot of the toggle coverage and test size objective subset from our test generation process. The best Pareto fronts from each evolution are plotted for toggle coverage achieved against test sizes. Best Pareto front plots show the fitness trade-offs possible between coverage and conflicting test size objectives. While our discussion focuses on toggle coverage in Figure 5, we emphasize that similar plots and characteristics are prevalent for the line and conditional coverage with test sizes objective subsets.

Figure 5 shows that tests achieve coverage enhancements whilst trying to contain test size. For instance, during early to mid evolutions, the best Pareto fronts rise upwards along the y-axis showing coverage gains, but smaller drifts to the right along the x-axis indicate test sizes did not increase rapidly.

This pattern continues up till the middle stages of the evolution process, because opportunities for optimization and uncovering of new test space are higher with low test size penalty. Toward later evolutions, optimizations to attain further coverage improvements whilst containing test sizes become increasingly difficult. Hence, the vertical y-axis gap between consecutive Pareto fronts becomes smaller.

Eventually, the GEA process reaches a plateau whereby no further coverage enhancement is possible using the current test building block genome and test population.

5.1.2 Test diversity characteristics. Pareto front length, slope and curvature characteristics at every evolution are all indicators of the types and range of tests that make up the test population. The larger the length and slope of the Pareto fronts, the more diverse the tests are.

In Figure 5 and similar plots for the other objective subsets, Pareto fronts are generally of sufficient size throughout the GEA process for a diverse test population. The large expansion of the best Pareto front at each evolution indicates tests are diverse to exercise different types of SoC functions.

Besides Pareto front size, the Pareto front curvature characteristics reveal how effective the GEA process optimizes objectives. A high curvature implies the GEA process is optimizing all objectives to find the best trade-off between coverage and test size fitness. Ideally, if the GEA process is to optimize multiple objectives effectively, the Pareto front curvature should also be convex toward the top left region of the plot. As demonstrated in Figure 5, between evolutions 20 to 25, the Pareto front whose curvature is obvious and its peak aimed at this region indicates tests exists along the Pareto front for which the highest coverage and lowest test size are being actively evolved. From evolution 25 onwards, as test diversity starts to fall, the resultant test suite is unable to achieve the same rate of optimization for all coverage and test size multi-objectives.

The Pareto front characteristics and test diversity results show our test generation method conducted multi-objective Pareto optimization as required. The SoC was verified with tests that actively maximised coverage metrics whilst test sizes and verification resources were minimised.

5.2 Multi-objective test generation comparison. In this section, the multi-objective GEA test generation is evaluated against four test creation techniques; two are based on single-objective schemes, and the other two are common methods employed in the verification industry. The techniques to evaluate against are: (i) SAGETEG [19], a single-objective GEA test generator; (ii) the µGP test generator [20], an assembler instruction based test generator that also employs single-objective GEA; (iii) random constraint-biased test generations [15]; and (iv) applications driven test creations that is conducted manually.

Each test generation method was configured to perform verification on the Nios SoC using the same test building block library. We evaluate and compare the coverage, test size, and testing time results for all methods, as tabulated in Table 1.

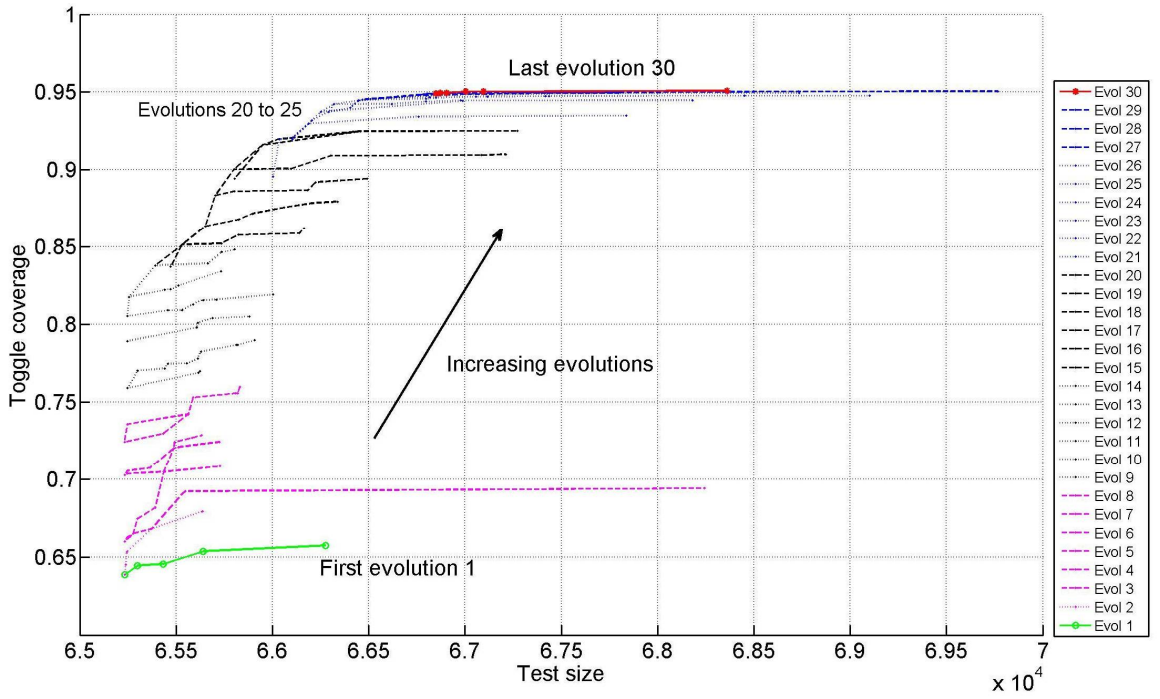

FIGURE 5. Best Pareto front from every evolution (toggle coverage vs. size)

In terms of line and toggle coverage, multi-objective GEA clearly surpasses other test generation methods, whilst conditional coverage shows lower improvement. Conditional coverage is the most difficult metric to maximise, and the limitations in enhancing conditional coverage further is due to the available test building block genome, not our test generation method. Despite this, multi-objective GEA testing acquires up to 17% better conditional coverage than other test generations.

Comparing test program size results, multi-objective GEA testing outperforms single-objective GEA test generation of SAGETEG and µGP. Overall, multi-objective test size per averaged test is approximately 7% smaller than SAGETEG and 23% smaller than µGP. This demonstrates multi-objective test generation achieves greater or equivalent coverage using smaller tests.

In the random approach however, random test programs are created by stochastic means only, and do not increase in size significantly when it strives to attain coverage, hence their size is slightly lower than multi-objective GEA testing. On the other hand, GEA test programs evolve and grow incrementally using test building block genome in order to seek greater coverage, but without significant size penalty.

Examining the time results in Table 1, multi-objective GEA test generation is efficient with regards to test execution and overall verification time. Multi-objective GEA completes at least 40% quicker, achieving superior or equivalent coverage, when compared with SAGETEG, µGP and random methods. For manual application testing, the elapsed testing time comparison is not applicable because creating tests manually will naturally be longer than automation.

In summary, the coverage, test size and time results demonstrate the effectiveness and advantages of our multi-objective GEA approach, especially when test size resources constraints exists in a verification project.

TABLE 1. Coverage, test size and time results

	Multi-objective GEA	SAGETEG	µGP	Random	Manual tests
Line coverage %	99.5	98.9	97.5	89.9	66.6
Toggle coverage %	95.7	93.7	87.1	79.2	48.8
Conditional coverage %	83.3	83.0	72.4	69.3	65.6
Average test size (bytes)	66,435	71,137	86,318	63,591	N/A
Test execution CPU time (s)	307,336	752,884	751,023	600,455	N/A

6. Conclusions. This paper presented a multi test objectives driven verification technique using genetic evolutionary algorithms and multi-objective optimizations. The key innovation with the multi-objective GEA method is realised by combining Pareto front sorting, aggregate ranking, and multi-phase divide-and-conquer GEA selection strategies. We demonstrated the multi-objective GEA test technique on the Nios SoC to maximise multiple coverage metrics and simultaneously minimise sizes of test programs executed. When compared with four other test generation methods, the multi-objective GEA scheme provided higher coverage across all coverage metrics, and utilised lower test sizes and resources.

REFERENCES

[1] J. Bergeron, *Writing Testbenches using SystemVerilog*, 1st ed. New York: Springer Science + Business Media, 2006.

[2] T. F. Liang and H. W. Cheng, "Multi-objective aggregate production planning decisions using two-phase fuzzy goal programming method," *Journal of Industrial and Management Optimization*, vol. 7, No. 2, American Institute of Mathematical Sciences (AIMS), pp. 365-383, 2011.

[3] T. Ray and R. Sarker, "EA for solving combined machine layout and job assignment problems," *Journal of Industrial and Management Optimization*, vol. 4, No. 3, American Institute of Mathematical Sciences (AIMS), pp. 631-646, 2008.

[4] E. Sanchez and G. Squillero, "Evolutionary Techniques Applied to Hardware Optimization Problems: Test and Verification of Advanced Processors," *Chapter in Studies on Computational Intelligence*, Vol 66, Advances in Evolutionary Computing for System Design, edited by Lakhmi C. Jain, Vasile Palade and Dipti Srinivasan, Springer publisher, 2007, pp. 83-106.

[5] C. A. C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Journal of Knowledge and Information Systems*, vol. 1, pp. 269-308, 1999.

[6] W. Jakob, M. Gorges-Schleuter, and C. Blume, "Application of genetic algorithms to task planning and learning," *Parallel Problem Solving from Nature, 2nd Workshop, Lecture Notes in Computer Science*, pp. 291-300, 1992.

[7] P. B. Wilson and M. D. Macleod, "Low implementation cost IIR digital filter design using genetic algorithms," *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, vol. 1, pp. 41-48, 1993.

[8] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Massachusetts: Addison-Wesley, 1989.

[9] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, pp. 221-248, 1984.

[10] C. M. Fonseca and P. J. Flemming, "Genetic algorithms for multi-objective optimization: formulation, discussion, and generalization," in *Proc. 5th Int. Conf. on Genetic Algorithms*: Morgan Kaufmann, 1993, pp. 416-423.

[11] T. Bao and B. Mordukhovich, "Refined necessary conditions in multi-objective optimization with applications to microeconomic modeling," *Discrete and Continuous Dynamical Systems*, vol. 31, No. 4, American Institute of Mathematical Sciences (AIMS), pp. 1069-1096, 2011.

[12] H. Bonnel and N. S. Pham, "Non-smooth optimization over the (weakly or properly) pareto set of a linear-quadratic multi-objective control problem: explicit optimality conditions," *Journal of Industrial and Management Optimization*, vol. 7, No. 4, American Institute of Mathematical Sciences (AIMS), pp. 789-809, 2011.

[13] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, pp. 257-271, 1999.

[14] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A Niched Pareto genetic algorithm for multiobjective optimization," *IEEE World Congress on Computational Computation*, vol. 1, pp. 82-87, 1994.

[15] A. Cheng, A. Parashkevov, and C. C. Lim, "A software test program generator for verifying system-on-chips," in *10th IEEE Int. High Level Design Validation and Test Workshop (HLDVT'05)*. Napa Valley, CA, 2005, pp. 79-86.

[16] H. Tamaki, H. Kita, and S. Kobayashi, "Multi-objective optimization by genetic algorithms: a review," in *Proc. IEEE Int. Conference on Evolutionary Computation*. Nagoya, Japan, 1996, pp. 517-522.

[17] Altera Inc, "Nios II hardware development tutorial". Available: http://www.altera.com/literature/tt/tt_nios2_hardware_tutorial.pdf.

[18] A. Cheng and C. C. Lim, "Markov modeling and parameterization of genetic evolutionary test generation," *Journal of Global Optimization*. Vol. 51, No. 4, 2011, pp.743-751.

[19] A. Cheng, C.C. Lim, Y. Sun, H. He, Z. Zhou, and T. Lei, "Using genetic evolutionary software application testing to verify a DSP SoC," in 4th IEEE Int. Workshop on Electronic Design, Test & Applications. Hong Kong: IEEE Computer Society, 2008, pp. 20-25.

[20] E. Sanchez, M. Schillaci, and G. Squillero, *Evolutionary Optimization: the μGP toolkit*, 1st ed. New York: Springer Science + Business Media, 2011.