

SYSTEM DESIGN FOR TEST USING A GENETICALLY BASED HIERARCHICAL ATPG SYSTEM

M. J. O'Dare and T. Arslan

Introduction

The continuous increase in the complexity of VLSI systems, is reflected in the numerous issues which must be considered by the system designer. In addition to the normal design constraints, the designer must consider testability enhancements which satisfy the needs of the test engineer. This paper describes a system-level hierarchical CAD test tool, which utilises the stochastic nature of genetic algorithms (GAs), expanding individual modules hierarchically and suggesting modifications to improve the testability. The GA is an artificial intelligence technique which has already been used to solve a number of problems in VLSI design and test, e.g. [1],[2] and [3], it is based on the natural evolutionary model [4]. Optimal solutions are produced by manipulation of a population of binary strings, governed by a selection mechanism biased towards producing a superior generation. Customarily the majority of the testing procedure takes place at the gate level of abstraction within the design hierarchy, posing a problem of test vector generation that is np-complete. The proposed system has therefore been developed by the authors for integration into the hierarchical design environment, establishing a link between the architectural and gate levels of abstraction during the fault simulation procedure. The CAD tool developed will prove useful to the designer at a higher level of the design stage. This simplifies the design task, in terms of test generation at the gate level. The system also incorporates heuristics that aid the design for test strategy by suggesting insertion of controllable test points at the architectural level of abstraction. During fault simulation single stuck-at-faults are detected by initially simulating the circuit at architectural level using high level primitives (adders, multiplexers, decoders etc.). A potential fault condition within a high level primitive module, is detected by dynamically expanding it to its gate level realisation during the fault simulation function.

1.1 Genetic Algorithm

The genetic algorithm is an artificial intelligence technique based and developed on the principles of evolutionary theory. A series of controlled operations on binary strings are performed, that emulate the biological evolutionary model. This procedure affords an effective search and optimisation algorithm, used in many real world problems [5].

A simple genetic algorithm is shown in Figure 1.1. Initially a population of binary strings are produced by the system. Where each member of the population is a potential test pattern for the circuit under test (CUT). Adopting the genetic terminology the test patterns are referred to as *chromosomes* [4]. A chromosomes length is equal to the number of primary inputs to the CUT.

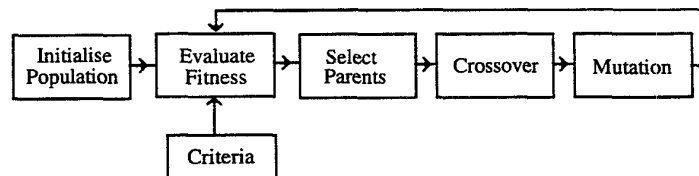


Fig 1.1 A Simple Genetic Algorithm

Each test pattern must now be evaluated for its *fitness* [4], a biologically derived term used to measure the proficiency of individuals in the quest for producing offspring that have a greater ability to solve a given problem than their parents. To evaluate the fitness of the chromosomes or patterns generated by the system, it is necessary to employ fault simulation to assess the effectiveness of every test pattern. In this work the single Stuck-at fault model [3] is used in the simulation procedure. In the initial population the fitness value is simply a measure of each patterns fault coverage. The fault-free response is then recorded for each test pattern. During simulation all internal nodes are driven to logical "1" and "0", whilst monitoring all primary outputs of the circuit for a change in state to that recorded for the circuit in the fault-free condition. If a change in state is observed the present fault condition is said to be detected and the fitness value assigned to the chromosome is incremented accordingly. Having assigned a fitness value to each member of the population based on the fault coverage, chromosomes will be selected to engage in crossover. The selection method adopted by the system is referred to as the Roulette Wheel method of parent selection, illustrated in Figure 1.2. Each chromosome may be visualised as occupying a segment of a roulette wheel. A chromosome with a relatively high fitness value would occupy a proportionally larger segment of the wheel when compared to a chromosome having a low fitness value. This method ensures that the probability of any chromosome being selected as a parent is directly proportional to its fitness value. In the example of Figure 1.2 the probability of each chromosome being selected as a parent is shown alongside the relevant segment.

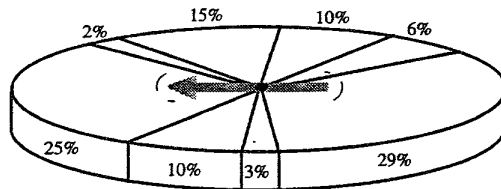


Figure 1.2 Roulette Wheel

Generally high calibre chromosomes will produce fitter offspring inheriting valuable characteristics from the parents. The mechanism employed that allows this to occur is *crossover* [4]. For this system two-point crossover is employed; firstly by random generation of two positions within the length of the parent chromosomes, secondly the information residing between the generated positions is exchanged between the parents, producing two children. The next stage in the main GA is the implementation of the *mutation* [4] operator, which introduces new information into the population by randomly changing the binary value of a gene within a chromosome. The probability of any bit being altered is 2.5%; a value chosen to satisfy the requirements of the test pattern generation system.

For this and subsequent generations the system employs a *global record table*, recording information obtained during fault simulation. The chromosome with the highest fault coverage in the first generation is entered into the test set. The stuck-at faults detected by this chromosome are recorded in the global record table, along with the total fault coverage achieved by the current test set. During fault simulation the global record table is used as a reference by another subsidiary function in the assignment of fitness values. For each detected fault a score is assigned to the chromosome's fitness, the value of which is based on the information held in the global record table. A fault that has previously been detected by a chromosome in the test set receives a relatively low score in comparison to the score awarded for detection of a previously undetected fault. The scores are dynamically adjusted in relation to a circuit's complexity.

2.1 Hierarchical Expansion

The ATPG system presented in this paper has been employed within a hierarchical framework [6], producing test patterns for circuits by simulating a module level realisation of a circuit. The modules are standard to cell libraries (e.g. adders, multiplexers, decoders etc.). Since the majority of test pattern generation takes place at the architectural level of abstraction, the design engineer has more control over issues such as - the Design For Test (DFT) issue. A diagrammatic representation of the hierarchical approach is shown in Figure 2.1.

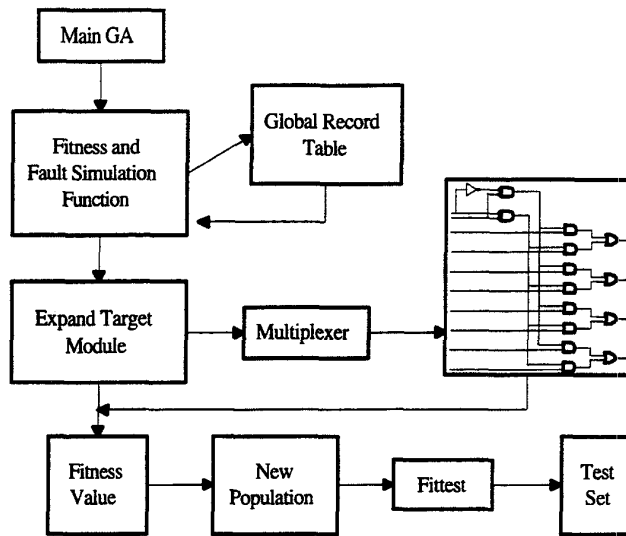


Figure 2.1 Hierarchical Test Approach

The fault simulation routine expands only one module at a time in order to detect a target fault within that module. In the example of Figure 2.1, a multiplexer is expanded in order to detect an internal fault, and simulation at gate level is performed for this module only, in a manner identical to that described in section 1.1. This procedure is obviously a great deal faster than performing gate level fault simulation for the whole circuit. A cost model for the speed-up is presented by Min *et al* in [7].

Some test patterns proved difficult to find for certain nodes within the circuit, for this reason a DFT heuristic has been incorporated to assist the system in its search for a compact test set.

3.1 DFT Heuristic

In the generation of test vectors for complex circuits certain faults have proved difficult to detect, investigation of such circuits by the authors prompted the use of a gate level heuristic. Generally for the circuits investigated there were two main reasons for undetected faults: firstly, the fault-free response required to detect a fault condition on a given node, had a low probability of occurrence, For example, in Figure 3.1, to detect the fault condition stuck-at-zero on a node *X* the fault-free response required on that node is logical one. If there are only 3 test vectors capable of producing this condition in a 20 input circuit, the probability of occurrence in a randomly generated test is $0.286 \times 10^{-3} \%$. The second was due to a low probability of propagating a path from a problem node to an observable primary output. To solve this problem the system suggests the placement of a limited number of test points within the circuit to improve testability. The heuristic monitors the progress of the GA, until no improvement in fault coverage for the circuit is made over several generations. Test points are then introduced into the circuit based on creating a propagation path to a primary output, that will allow detection of problem faults.

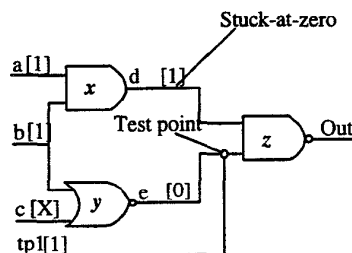


Figure 3.1 Example Circuit

4.1 Results

The system has been successfully applied to a number of the ISCAS 1985 benchmark circuits, and favourable results were obtained in comparison to other research in the same area. For example, when the system was applied to the ISCAS C-432 circuit 23 test patterns were produced covering 97.704% of the circuits single stuck-at-faults. Ayari *et al* present a system in [8] that produces 52 test patterns that detect 97.48% of the faults.

The fault coverage is improved even further by implementation of the DFT heuristic, which the authors have implemented for the insertion of test points, with a number of the ISCAS combinational benchmark circuits. The system identifies positions in a circuit where a test point would aid fault detection, based on information gleaned during the fault simulation routine. The system records the state of all nodes within the circuit in the global record table during simulation, a test point may then be placed on a node that has a low probability of the required signal state being present, as discussed in section 3.1. The graph in Figure 4.1 is an illustration of the effect of the DFT heuristic employed to insert test points within the C432 ISCAS benchmark circuit. The fault coverage improved to 99.25% with a test set of 30 patterns.

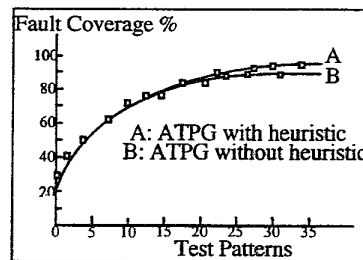


Figure 4.1 Fault Coverage Improvement

The hierarchical approach referred to in section 2.1, improves the speed of the system by dealing with the circuit at a modular level. The actual speed-up achieved is based on the number of gates contained within a module, and the number of potential fault conditions for each gate; a full derivation of the formulae required to calculate the cost of hierarchical test pattern generation based on speed-up is presented in [7]. The authors have applied hierarchical test pattern generation to several combinational circuits of varying complexity. The circuits were constructed of modules such as: adders, decoders, parity generators etc., consisting of approximately 10 to more than 1000 gates per module, and circuits consisting of between approximately 3 and 12 modules.

5.1 Conclusion

In this paper a genetic algorithm is used for test pattern generation within a hierarchical framework. The genetic algorithm is aided by a heuristic which suggests possible test points by tracking the genetic evolutionary process. The use of such test points aids the system in its search for a compact set of test patterns by speeding-up the search specially at the earlier stages of the genetic evolution.

References:

- [1] Buttitta, P. Orlando, F. Sorbello, and G. Vassallo, "Monreale: A new genetic algorithm for the solution of the channel routing problem," Proc. IEEE, CH3001-5, pp. 462-466, 1991.
- [2] Hegde, and B. Ashmore, "A feasibility study of genetic placement," Texas Instrum. Technol. J., vol. 9, pp. 72-82, 1992.

- [3] M. J. O'Dare and T. Arslan, "Generating test patterns for VLSI circuits using a genetic algorithm," IEE Electronics Letters, Vol 30, No. 10, pp. 778-779, May 1994.
- [4] Holland, "Adaptation in Natural and Artificial Systems," Univ. of Michigan Press, Ann Arbor, 1975.
- [5] L. Davis "Handbook of genetic algorithms," Van Nostrand Reinhold, New York, 1991.
- [6] D. Calhoun, and F. Brglez, "A Framework and Method for Hierarchical Test Generation," IEEE Trans. on Computer-Aided Design, vol. 11, pp. 45-67, January 1992.
- [7] Min, H. A. Luh, and W. A. Rogers, "Hierarchical Test Pattern Generation: A Cost Model and Implementation," IEEE Trans. Computer-Aided Design, vol. 12, pp. 1029-1038, July 1993.
- [8] Ayari, and B. Kaminska, "A New Dynamic Test Vector Compaction for Automatic Test Pattern Generation," IEEE Trans. Computer-Aided Design, vol, 13, pp. 353-358, March 1994.