14th IEEE International On-Line Testing Symposium 2008

A Novel GA-Based High-Level Synthesis Technique to Enhance RT-Level Concurrent Testing

Naghmeh Karimi[1], Soheil Aminzadeh[2], Saeed Safari1 and Zainalabedin Navabi[1]
[1]Electrical and Computer Engineering Department, University of Tehran
[2]Computer Engineering Department, Sharif University of Technology
naghmeh@cad.ece.ut.ac.ir; aminzadeh@ce.sharif.edu; saeed@ut.ac.ir; navabi@ece.neu.edu

Abstract

This paper presents an efficient High-Level Synthesis (HLS) approach to improve RT-Level concurrent testing. The proposed method used for both fault detection and fault location. At first the available resources are used in their dead intervals to test active resources for fault detection, and then some changes are applied to the RT-Level controller to locate the faults. The fault detection step is based on a genetic algorithm (GA) search technique. This genetic algorithm is applied to the design after high level synthesis process to explore the test map. The proposed method has been evaluated based on dependability enhancement and area/latency overhead imposed to different benchmarks after applying our algorithm. The dependability has been considered in terms of fault coverage. The experimental result shows that applying our algorithm, the associated area overhead and performance penalty are negligible while the online fault coverage improvement is considerable.

1. Genetic Based Efficient Self-Testing Algorithm (GB-ESTA)

Here we describe the proposed method to improve the self-testability of the resulted RTL circuit. The proposed method produces a self-testable RTL circuit which is tested during its normal operation. The main goal of this method is to alter the original data-path and controller to find an optimum self-testable circuit with a minimum area/latency overhead.

1.1. Overview

At the first step, behavioral description of the input circuit is converted to an internal data structure shows the DFG of the circuit. After that the conventional HLS algorithms, i.e. scheduling and binding, are used to find an RTL implementation of the circuit that is not necessarily self-testable. Then GB-ESTA uses a deterministic modification algorithm to find a normal distribution of the idle modules in each working clock cycle of the circuit. This increases the possibility to find an idle module in each clock cycle to test a busy module of the same type. In the next step, GB-ESTA employs two GAs to find the effect of using extra modules (and extra clock cycles) on circuit parameters, e.g. area, latency and self-testability. Finally, this information is used to alter the data-path and controller of the circuit to improve the self-testability of the circuit with minimum area/latency overhead.

1.2. Conventional HLS

The input of high level synthesis tool is described as a behavioral HDL code. This behavioral code will be translated to a DFG intermediate format. Then, Force-Directed Scheduling (FDS) is used to find the execution time of each operation. Given an SDFG, binding assigns modules to perform operations and registers to store variables and generates the data-path of the circuit. To find the RTL implementation of the circuit an extra step, called controller synthesis, should be performed on Bound DFG to specify the control signals value at each clock cycle.

1.3. Algorithm Description

The proposed algorithm is implemented in two different versions: On-line Fault Detection version and On-line Fault Location/Correction version.

1.3.1. On-line Fault Detection: Here we will describe the algorithm developed for on-line fault detection. At first, a deterministic algorithm is run on SDFG to maximize the idle modules of all types at each clock cycle. We employed a deterministic algorithm to re-schedule the operational modules such that the numbers of idle modules at each clock cycles are maximized. In the next step this algorithm adds extra modules in such a way that there are two modules of the same type (one

978-0-7695-3264-6/08 $25.00 © 2008 IEEE
DOI 10.1109/IOLTS.2008.43

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 16:22:19 UTC from IEEE Xplore. Restrictions apply.

busy and one idle) at each clock cycle. Also in some cases it is required to add clock cycle(s) in which modules are idle and can be tested. To test them we may add an extra clock cycle in which both adders are idle and can be tested. Note that adding extra modules and extra clock cycles results in area and latency overhead respectively.

Then two GAs, named GA1 and GA2, are employed to find the test map of the circuit shows how to test all modules. GA1 tries to find a different test map for each module at each clock cycle and GA2 finds the best module test map resulted from GA1.

Now there is a test map for each module and this is the time to modify data-path and controller of the circuit. RTL modification part of GB-ESTA adds some hardware redundancies to the data-path and controller to apply the same inputs to modules A and B, if B is used to test A.

1.3.2. On-line Fault Location/Correction: In this scheme the fault detection is performed as before (described in section 1.3.1) and the RTL modification step is changed to provide fault location and correction. Here we add hardware redundancies assuring to have at least three instances of each module type. When an error reported due to the output mismatch of two modules (a busy and an idle one), the controller freezes the data-path and enters to a location/correction state. In this state controller applies the same inputs to the three modules of the same type and finds the faulty module using a sift unit. Sift unit also generates the correct output using a Triple Modular Redundancy (TMR) scheme. Then the controller goes back to a normal state and continues its works.

When an error is reported, we wish to apply the input that raises an error to all three modules at location/correction state. If the input and output variables of a module are mapped to the same register, we lose the input and correction phase can not be used correctly. So we should prevent that the input and output variables of a module are mapped to the same register and this results in more registers in RTL implementation of the circuit.

1.4. Reliability Improvement

Here we will show that the self-testable implementation provides more reliability in comparison of conventional implementation. Note that the self-testable implementation has only a 1-bit output, i.e. Error. So it may seem that it is harder to test it in comparison to the conventional implementation which has 3 8-bit outputs. We claim that all faults except the faults on register outputs are detectable in the self-testable implementation. This is due to the fact that when a fault occurred on a register output, it is applied to both (the busy and the idle) modules inputs and consequently they provide the same output and there is no way to detect the fault. For all other faults there is a test vector that activates the faults using operational modules.

2. Experimental Results

To evaluate our method, we have applied the proposed algorithm to several benchmarks. The selected benchmarks are cir1, DiffEq and a 6th order FIR filter [1]. We first applied our method to the behavioral description of each circuit for data-path and controller synthesis, obtaining an output in Verilog format. The output is then converted to a gate-level net-list called ISCAS. HITEC-PROOFS [2] is then used to evaluate fault coverage using 5000 test vectors. The area, latency and fault coverage are reported in Table 1. As shown in Table 1, due to the self-correcting mechanism the fault coverage for self-correctable implementation is not reported.

Note that the fault coverage reported in Table 1 for conventional and self-testable method shows the off-line and on-line testing fault coverage respectively. In other words, the on-line fault coverage of the conventional RTL is 0. So, our method results in a considerable improvement in on-line fault coverage.

3. References

[1] M. T. Lee, High-Level Test Synthesis of Digital VLSI Circuits, Artech House, 1997.

[2] T. Niermann, J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," Proceedings of the European Conference on Design Automation, pp. 214-218, 1992.

Table 1. Experimental Results

	Implementation	Area (#Gates)	Clock Freq (MHz)	Fault Coverage
Cir1	Conventional	3007	90.4	%99.8
	GB-ESTA (Detection)	3744	84.1	%90.2
	GB-ESTA (Correction)	6462	63.7	-
Diff Eq	Conventional	3553	86.9	%99.8
	GB-ESTA (Detection)	4694	83.4	%85.4
	GB-ESTA (Correction)	7901	60.2	-
FIR	Conventional	3468	82.6	%99.7
	GB-ESTA (Detection)	5360	80.8	%83.2
	GB-ESTA (Correction)	7974	56.3	-

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on June 14,2021 at 16:22:19 UTC from IEEE Xplore. Restrictions apply.