PROPERTY-BASED TESTABILITY ANALYSIS FOR HIERARCHICAL RTL DESIGNS

Yiorgos Makris & Alex Orailoğlu

Reliable Systems Synthesis Lab – CSE Department
University of California San Diego
9500 Gilman Drive 0114, La Jolla, CA 92122, USA

ABSTRACT

We present an analysis methodology that identifies testability bottlenecks in RTL designs, based on the concept of *transparency properties*. We discuss a hierarchical test generation methodology, wherein test is locally generated for each module and subsequently translated into global design applicable test. We introduce the notion of transparency properties for capturing test translation related behavior of modules, without reasoning on the complete functionality of the design. A recursive search algorithm that combines properties into test justification and propagation paths and reveals the reachability bottlenecks for each module in the design is subsequently devised. An ATPG-based experimental setup validates that the proposed methodology identifies accurately the test translation bottlenecks in the hierarchical design.

1. INTRODUCTION

Continuous silicon-manufacturing technology improvements have enabled the realization of extremely large and complex designs that have by far outpaced the capacity of the EDA tools to handle them as monolithic entities. Significant effort has been consequently invested in devising hierarchical approaches for accommodating current and future design and test needs. Hierarchical test generation, as depicted in Fig. (1), comprises a promising and viable alternative to the slow and complex global design gate-level test generation process. Local test can be quickly and efficiently generated for each module of a hierarchical design and consequently translated to global test, applicable at the complete design boundary.

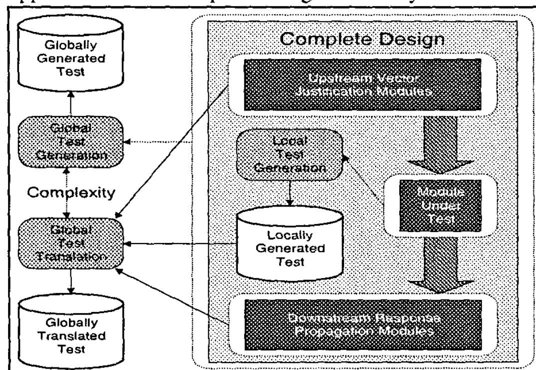


Fig. (1): Hierarchical Test Generation Framework

Despite the fact that such approaches are independent of the local test generation mechanism and the targeted fault models, their success has been heavily dependent on the efficacy of the test translation process. Test translation approaches that attempt to explore exhaustively the complete functional space of the upstream vector justification and downstream response propagation logic, are doomed by the same complexity barrier as global design test generation. Test transparency related behavior has been alternatively employed for the surrounding modules, while translating local into global test. Although such approaches make translation a fast and trivial process when transparency exists, partial transparency or lack of transparency, even for a single module, proves catastrophic for the translation process, resulting in very low fault coverage. Defining the transparency notion becomes therefore a crucial task, constituting the trade-off mechanism between the complexity and the efficiency of the local-to-global test translation process. The role of testability analysis is to identify the test translation bottlenecks in the design, in order to provide valuable information that can be used for guiding the translation process and for incorporating judicious DFT enhancements in the design, as discussed in [10].

Section 2 briefly reviews previous work in the area of hierarchical test generation, testability analysis and test translation transparency definition. Section 3 introduces a novel transparency definition through the concept of *transparency properties*, based on which a hierarchical testability analysis methodology for bottleneck identification is devised in section 4. Experimental data validating the results is provided in section 5.

2. PREVIOUS WORK

Several testability analysis methodologies, hierarchical test generation approaches and distinct transparency definitions have been proposed, in an effort to address large designs. A representative set of these approaches is discussed in this section.

The concepts of *I-Path* and *T-path* for capturing module transparency, in terms of identity and transformation functions respectively, was introduced in [1]. These *one-to-one* and *onto* functions are limited to equal bitwidth signal entities and only the *I-Path* was pursued for hierarchical test generation. The path notion was extended in [5] through arbitrary *onto*

0-7803-5682-9/99/$10.00©1999 IEEE.

functions *(S-paths)* for providing stimuli to a module and arbitrary *one-to-one* functions *(F-paths)* for propagating fault responses. A hierarchical test generation scheme and a test result propagation methodology based on *ambiguity sets* were introduced in [11,12]. Complexity issues limit the applicability of these methods to small combinational datapath circuits. A methodology for extracting test knowledge for hierarchical designs in terms of *modes* was proposed in [16]. Modes are combined over test justification and propagation paths [17], in order to translate test. Highly translatable local test is generated in [15] by incorporating global design knowledge in the test generation process in terms of *constraints*. High-level *architectural information* is used in [8] for similar purposes. Constraint extraction complexity limits these approaches, necessitating an efficient transparency definition. Finally, various general approaches for addressing behavioral and RTL testability analysis have been suggested in [3, 4, 7].

3. TRANSPARENCY PROPERTIES

In this section we introduce a mechanism for capturing test translation related behavior of modules in a design. This behavior is utilized for examining the reachability of the module under test and identifying the test translation bottlenecks. Since exhaustive examination of the complete functional space is impossible due to complexity issues, we rely on capturing only test translation related behavior for each module and utilizing only this behavior for test justification and propagation. Although this scheme sacrifices some of the behavioral space, it provides the ability to reason on the translation capabilities of the design in an automated fashion.

The proposed scheme is based on arithmetic properties of modules and targets mainly data path modules, while the control logic is handled through FSM analysis. The basic concept underlying the property-based methodology is the utilization of arithmetic properties that can provide a simple transparency mechanism over modules. The algebraic scheme composed by the arithmetic properties, allows bulk mode translation of test requirements.

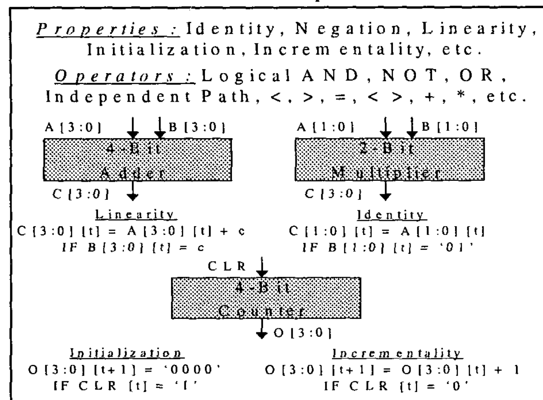


Fig. (2): Transparency Properties of RTL Modules

Arithmetic properties do not capture the complete functional space of the modules but only the behavior that is most appropriate for test translation. Using this behavior, testability requirements at the inputs of a module are translated to equivalent requirements at the outputs of the module and vice versa. The new requirements are possibly related through *operators* and are based on the satisfiability of a number of *conditions*, as demonstrated in the examples of Fig. (2).

Similarly, condition compliance examination is based on the same property combination scheme. The types of properties that our scheme considers include identity, linearity, negation, initialization and incrementality. The operators used for combining these properties are both in the arithmetic domain (e.g. +, *, <, =, >, etc.) and in the structural path domain (e.g. logical AND, NOT, OR, etc.). The arithmetic property scheme is capable of handling both combinational and sequential modules since timing is captured as part of the properties, as demonstrated on the 4-bit counter.

The introduced transparency properties constitute a powerful mechanism for capturing test translation related behavior of modules in a hierarchical design, facilitating an efficient hierarchical testability analysis scheme that reveals the testability bottlenecks of the design, as discussed in the following section.

4. HIERARCHICAL ANALYSIS

The central ideal of the analysis methodology is the examination of the satisfiability of test justification and propagation *objectives* defined at the boundaries of each module. An objective is defined as the justification of values to the inputs of a module or the propagation of responses from the outputs of a module using transparency properties of surrounding modules. Objective examination for a particular module is a two-step process. First, the objectives have to be identified at the boundary of the module under test. Subsequently, their satisfiability needs to be analyzed, in order to pinpoint potential test translation bottlenecks.

i) Objective Identification: Since the actual test vectors and responses are not known at the time of analysis, the most naïve way is to try to justify every possible value to the inputs of the module under test and propagate every possible value from the outputs of the module under test. A more informed approach is taken herein to ease this overkill. We examine the module under test and identify the cones of logic that each input drives and the cones of logic that each output is driven by. This input/output mapping is an indication of the decomposability of the module, through which the dependent sets of inputs and outputs can be extracted and used for defining more realistic objectives in a stream-wise manner. Another factor determining the time aspect of the objectives is the sequential depth of the module. This idea is depicted in Fig. (3), identifying the justification and propagation objectives on a 4-bit register. The sequential depth of the register is 1, therefore we need 2 back-to-back vectors to test it.

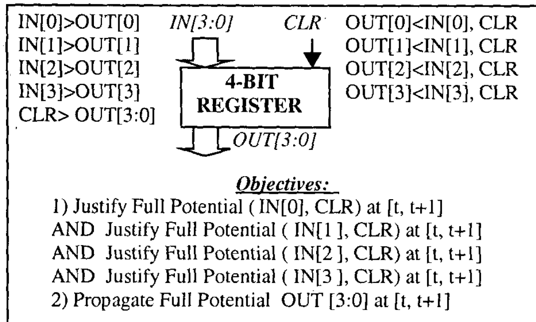
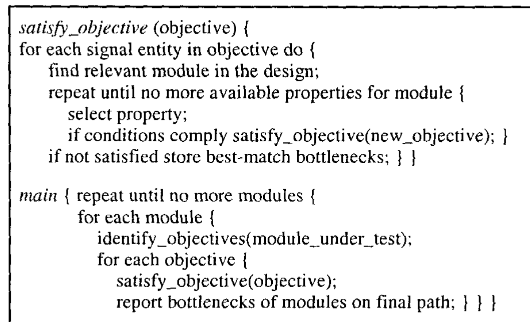


Fig. (3): Objective Identification Example



```
satisfy_objective (objective) {
for each signal entity in objective do {
    find relevant module in the design;
    repeat until no more available properties for module {
        select property;
        if conditions comply satisfy_objective(new_objective); }
    if not satisfied store best-match bottlenecks; } }

main { repeat until no more modules {
        for each module {
            identify_objectives(module_under_test);
            for each objective {
                satisfy_objective(objective);
                report bottlenecks of modules on final path; } } }
```

Fig. (4): Objective Satisfiability Algorithm

ii) Objective Satisfiability: The algebraic scheme is based on objective transformation through the notion of *properties*. Starting from the signal entities on which the objective is defined, we utilize the global circuit connectivity information to identify the module that the signal entity is coming from or going to. Then, we select a *property* for this module that will transform the *objective* into a number of *objectives* at the other boundary of the module, related through *operators* and a number of *conditions*. The compliance of the conditions is subsequently examined and if there is no conflict the procedure is repeated for the new objectives. Otherwise, a new property is selected. At the end, objectives are either satisfied through controllability/observability points such as primary inputs/outputs or they are not satisfiable in which case we report the corresponding signal entities as bottlenecks. This procedure is further described in the algorithm of Fig. (4), details of which are given in [9].

Selecting judiciously among the alternative properties that satisfy an objective over a module has a significant impact on the amount of backtracking performed by the search algorithm. A number of *decision factors* are examined in order to speed things up. Factors are extracted from the circuit connectivity model and include the number of signal entities, conditions and clock cycles of alternative properties and the potential formation of reconvergence or loops.

5. EXPERIMENTAL VALIDATION

The proposed methodology for test justification and propagation analysis identifies the potential controllability and observability bottlenecks in the RTL design. This section describes the experimental validation framework employed for examining the analysis accuracy.

An overview of the validation flow is provided in Fig. (5). In compliance with the test framework of Fig. (1), our experimental setup utilizes HITEC [13], a gate-level ATPG tool and is based on fault coverage comparison acquired from the fault simulator PROOFS [14]. Starting with an RTL description of the circuit, the described analysis methodology is applied and a list of controllability and observability bottlenecks is acquired. Subsequently, synthesis provides a gate-level model on which our ATPG experiments are performed.

First, the ATPG tool is applied on each design module and local tests are generated. Each local test is translated to the boundaries of the complete circuit and global test is obtained and fault simulated, providing the global fault coverage GFC. The same experiment is then performed on an enhanced version of the design, wherein all the identified bottlenecks are considered controllability and observability points respectively, providing the modified GFC fault coverage.

The above experimental setup was applied on three benchmark designs, and the results are summarized in Table (1). The first design (MTC100) is a 3-module circuit first introduced in [16], with interesting feedback loop behavior, resulting in major slow-down for ATPG tools. The second design is an 8-bit binary sign-magnitude shift-&-add multiplier (MUL) described in [6], on which analysis is performed both with and without the controller. The third circuit is a pipelined multiplier accumulator (MAC) for complex numbers, described in [2], consisting of 23 modules, including arithmetic blocks such as multipliers and adders, along with registers and simple control logic.

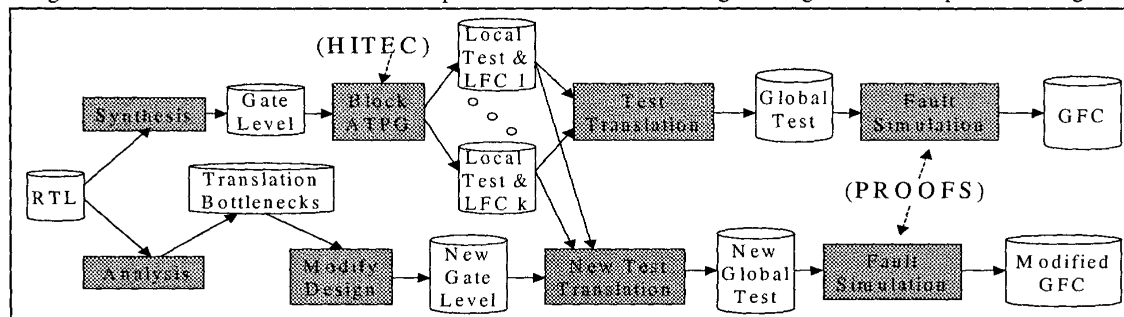


Fig. (5): Experimental Validation Flow

Benchmark Circuit	Σ (LFC)	GFC	Modified GFC
MTC100	96.68 %	89.70 %	94.62 %
MUL (w. control)	97.45 %	90.57 %	95.76 %
MUL (w/o control)	94.52 %	62.67 %	88.56 %
MAC	96.44 %	67.24 %	90.15 %

Table (1): Experimental Results

The analysis methodology indicates that the MTC100 and the MUL without the controller are highly transparent and that no major bottlenecks exist in the design. Consequently, we expect translation to be highly successful. Summing up the total covered faults and dividing by the total number of faults, we obtain 96.68% and 97.45% fault coverage, respectively. After the local test is translated to test applicable at the global design boundary, coverage drops slightly to 89.70% and 90.57% due to the small number of bottlenecks. A new translation after bottleneck resolution through test point insertion increases the coverage to 94.62% and 95.76%, respectively.

When the controller of the MUL is also considered, the corresponding coverage loss due to test translation bottlenecks is considerably larger, resulting in a drop from 94.52% to 62.67%. Resolving the many bottlenecks, revealed by testability analysis in this case, goes a long way towards eliminating the translation problems, as can be seen by the drastically improved coverage of 88.56%. Similarly, in the case of the MAC, the property-based analysis identifies numerous justification and propagation bottlenecks in the design. The coverage drops from 96.44% achieved by the local tests to 67.24% achieved by the translated test on the original circuit. After the bottlenecks are resolved, the coverage of the new translated test increases to 90.15%, validating the accuracy of the methodology in identifying test translation bottlenecks.

6. CONCLUSIONS

Size and complexity considerations impede test generation tools that attempt to address modern designs as monolithic entities, revealing the imperative need for hierarchical approaches. In order to constitute viable alternatives, such hierarchical approaches require that locally generated vectors can be translated into global design meaningful test. We present a testability analysis methodology that identifies the bottlenecks of the test translation process in a prompt and efficient manner, without exhaustive reasoning on the complete design functionality, in order to avoid complexity pitfalls. The introduced concept of *transparency properties* for capturing test translation related behavior of modules facilitates a traversal algorithm that searches for test justification and propagation reachability paths for every module in the hierarchical design and reports the test translation bottlenecks. The accuracy of the identified bottlenecks is validated through the described experimental setup, making the proposed analysis scheme a useful resource for assisting hierarchical test generation and DFT related decisions.

REFERENCES

[1] M. S. Abadir, M. A. Breuer, "A Knowledge-Based System for Designing Testable VLSI Chips", *IEEE Design and Test of Computers*, vol. 2, no. 4, pp. 56-68, 1985.

[2] P. Ashenden, *The Designer's Guide to VHDL*, Morgan-Kaufmann Publishers Inc., 1996.

[3] C-H. Chen, T. Karnik, D.G. Saab, "Structural and Behavioral Synthesis for Testability Techniques", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 13, No. 6, pp. 777-785, 1994.

[4] F. Corno, P. Prinetto, M. Sonza Reorda, "Testability Analysis and ATPG on Behavioral RT-Level VHDL", Proceedings of the International Test Conference, 1997, pp. 753-759.

[5] S. Freeman, "Test Generation for Data-Path Logic: The F-Path Method", *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 421-427, 1988.

[6] J.P. Hayes, *Computer Architecture and Organization*, McGraw-Hill, 3rd Edition, 1998.

[7] J. Lee, J. Patel, "Testability Analysis Based on Structural and Behavioral Information", Proceedings of the 11th IEEE VLSI Test Symposium, 1993, pp. 139-145.

[8] J. Lee, J. H. Patel, "Hierarchical Test Generation under Architectural Level Functional Constraints", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1144-1151, 1997.

[9] Y. Makris, A. Orailoğlu, "RTL Test Justification and Propagation Analysis for Modular Designs", *Journal of Electronic Testing: Theory & Applications*, Kluwer Academic Publishers, vol. 13, no. 2, pp. 105-120, 1998.

[10] Y. Makris, A. Orailoğlu, "DFT Guidance Through RTL Test Justification and Propagation Analysis", Proceedings of the International Test Conference, pp. 668-677, 1998.

[11] B. T. Murray, J. P. Hayes, "Hierarchical Test Generation Using Precomputed Tests for Modules", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 6, pp. 594-603, 1990.

[12] B. T. Murray, J. P. Hayes, "Test Propagation through Modules and Circuits", Proceedings of the International Test Conference, pp. 748-757, 1991.

[13] 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.

[14] T. Niermann, W. T. Cheng, J. H. Patel, "PROOFS: A Fast, Memory Efficient Sequential Circuit Fault Simulator", Proceedings of the 27th ACM/IEEE Design Automation Conference, pp. 535-540, 1990.

[15] R. S. Tupuri, J. A. Abraham, "A Novel Test Generation Method for Processors using Commercial ATPG", Proceedings of the International Test Conference, pp. 743-752, 1997.

[16] P. Vishakantaiah, J. A. Abraham and M. S. Abadir, "Automatic Test Knowledge Extraction From VHDL (ATKET)", Proceedings of the 29th ACM/IEEE Design Automation Conference, pp. 273-278, 1992.

[17] P. Vishakantaiah, J. A. Abraham, D. G. Saab, "CHEETA: Composition of Hierarchical Sequential Tests using ATKET", Proceedings of the International Test Conference, pp. 606-615, 1993.