TRANSPARENT: A System for RTL Testability Analysis, DFT Guidance and Hierarchical Test Generation*

Yiorgos Makris, Jamison Collins, Alex Orailoğlu
Reliable Systems Synthesis Lab - CSE Department
University of California, San Diego
La Jolla, CA 92093

Praveen Vishakantaiah
Intel Corporation
Hillsboro, OR 97124

Abstract

We discuss a methodology for analyzing the testability of large hierarchical RTL designs, based upon the existence of module reachability paths, suitable for automatically deriving globally applicable test from locally generated vectors. Such reachability paths utilize module transparency behavior, as captured by the introduced *channel transparency* definition. Lack of transparency and unreachable module I/Os pinpoint testability bottlenecks apt for efficient DFT modifications. Application of this methodology on example designs results in significant fault coverage improvement and test generation speedup, as compared to complete design gate-level ATPG.

Introduction

Continuous improvements in silicon manufacturing technology 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, either top/down or bottom/up, for accommodating current and future design and test needs. The hierarchical test generation concept, depicted in Fig. 1, has been long ago proposed as a promising and viable alternative to the slow and complex global design test generation process (1-5). 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.

Although such approaches are independent of the local test generation mechanism, the actual test vectors and the targeted fault models, their success has been heavily dependent on the efficacy of the test translation process. Test translation approaches that reason exhaustively on the functional space of the upstream vector justification and downstream response propagation logic, such as in (1, 4, 6, 7), are doomed by complexity, despite being complete. DFT modifications can alleviate this problem by providing alternative reachability paths, however they are expensive and need to be incorporated judiciously.

Test transparency related behavior has been alternatively employed for the surrounding modules, while translating local into global test (3, 8-10). Although such approaches make test translation a fast and trivial process when transparency exists, partial transparency or lack of transparency, even for a single module, proves catastrophic for the test translation process. In order to preserve high fault coverage, an efficient mechanism for defining modular transparency and guiding DFT modifications to address the lack of such transparency is necessitated.

In the following sections, we introduce *TRANSPARENT*, a system for RTL testability analysis, DFT guidance and hierarchical test generation, based on the concept of *transparency channels*. We first provide an overview of the system, followed by a detailed discussion of its constituent parts. We present the channel transparency definition and its application on module reachability analysis and we discuss testability bottleneck identification and minimization, along with its potential for guiding DFT modifications. We then derive a hierarchical test generation scheme, wherein reachability paths are utilized for translating local module vectors into complete design test. Finally, we demonstrate the proposed methodology on example hierarchical designs and we report our results in comparison to complete circuit ATPG.

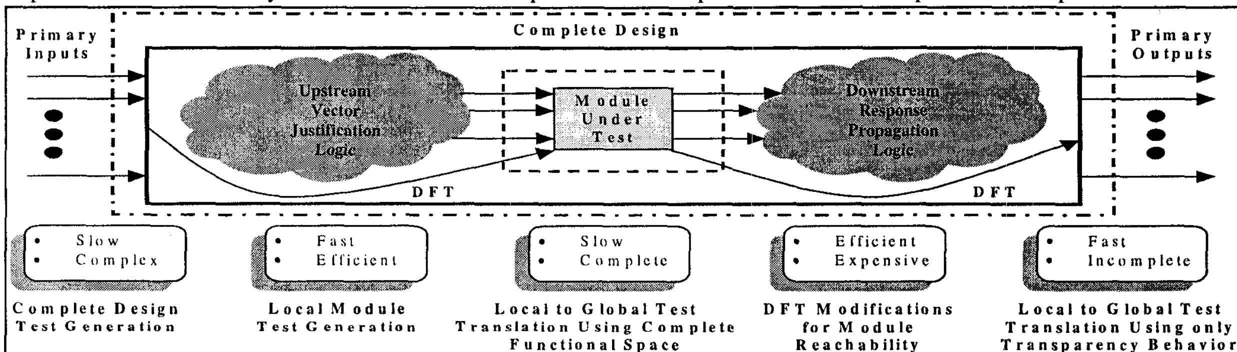

FIGURE 1: HIERARCHICAL TEST GENERATION, TEST TRANSLATION AND DFT APPROACHES – BENEFITS & DRAWBACKS

* This work is supported in part through a research grant from Intel Corporation under contract CSE 0129-58678A.

0-7803-5443-5/99/$10.00©1999 IEEE

IEEE 1999 CUSTOM INTEGRATED CIRCUITS CONFERENCE

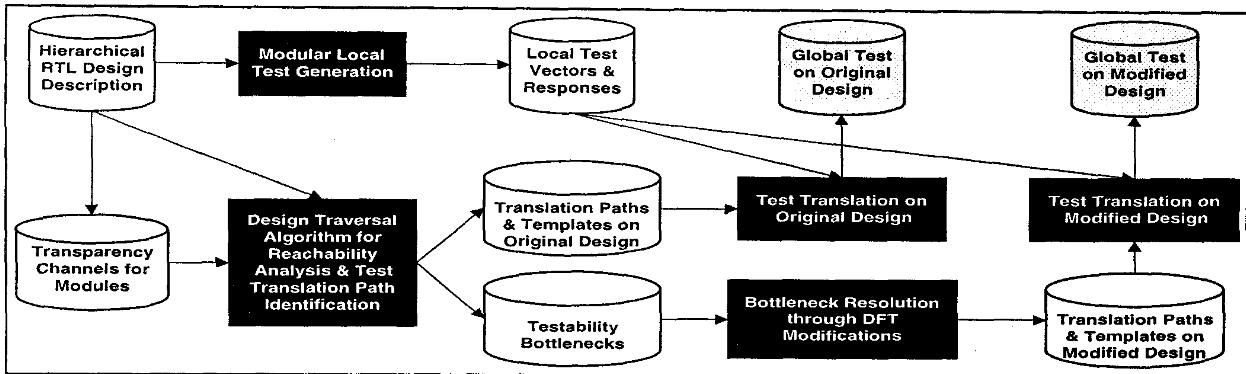

FIGURE 2: TRANSPARENT – SYSTEM OVERVIEW

System Overview

The system *TRANSPARENT*, described herein, comprises a methodology for analyzing the testability of RTL hierarchical design, reporting testability bottlenecks for guiding efficient DFT modifications and generating test in a hierarchical fashion. It provides early in the design cycle a hierarchical testability assessment, based on modular transparency, without exhaustively reasoning on the complete design functional space, in order to avoid complexity issues. The analysis is symbolic and therefore independent of the actual test vectors, the test generation mechanism and the underlying fault model. Furthermore, it handles variable bitwidth and sub-word signal entities and addresses both data and control path modules, combinational and sequential. The proposed system is significantly faster than full circuit, gate-level ATPG, while providing very high fault coverage through few, yet effective DFT modifications.

An overview of the *TRANSPARENT* system is depicted in Fig. 2. Starting with the hierarchical RTL design description, local test vectors and responses are generated for each module. Subsequently, a design traversal algorithm examines the reachability of each module in the design and identifies test translation paths, based on the notion of *transparency channels* that is introduced in the following section. The traversal algorithm provides the reachability paths and test translation templates on the original design, based on which the local test is translated into global design test. It further reports the identified testability bottlenecks in the design and a set of alternative reachability paths and test translation templates, valid on a modified design wherein the bottlenecks are resolved. These templates are used for obtaining global test from local vectors on the modified for testability design. The constituent parts of the system are further discussed in the following sections.

Testability Analysis

In this section we describe the testability analysis phase of the *TRANSPARENT* system. We define the *transparency channel* notion and we provide the design traversal algorithm that examines reachability paths for each module and reports test translation templates and potential testability bottlenecks.

A. *Transparency Channels*

The transparency channel definition, introduced in this section, alleviates the complexity of examining the complete functional space of each module during test translation. It constitutes, thus, a search space pruning mechanism that facilitates an efficient trade-off between completeness and complexity of the test translation process. The underlying theme for distinguishing test translation related behavior is the requirement for bulk mode, instead of case-by-case, test justification and propagation, while utilizing only existing module functionality. In this sense, channels provide a *pessimistic* view of a module's functional space.

Channels attempt to capture bulk mode, test translation related behavior of modules, in terms of bijection functions between input and output *signal entities*. Channels are instantiated upon the compliance of a number of *conditions* that require a specific *potential* on signal entities. The required potential may be either controllability or observability of the signal entity to a set comprising all possible values, a known constant value, an unknown but constant value, mutually exclusive values, or same values on the bits of the signal entity. Channels incorporate time considerations and may be defined either between an input signal entity and an output signal entity or between two output signal entities, in order to account for state-dependent, sequential logic behavior. Signal entities may be defined either on the full word bitwidth or on sub-word bitwidths. In order to be able to support and combine variable bitwidth channels on a search path, the notions of *well* and *drain* are employed. Wells are either primary inputs or internal modules with controllability potential on signal entities. Similarly, drains are either primary outputs, or internal modules with observability potential on signal entities. A succinct definition of the transparency channels is given in Fig. 3, along with a few examples of simple modules and associated channels.

In summary, channels capture test translation behavior of data and control path, combinational and sequential modules, variable bitwidths and sub-word signal entities. An extensive description of the channels is given in (10).

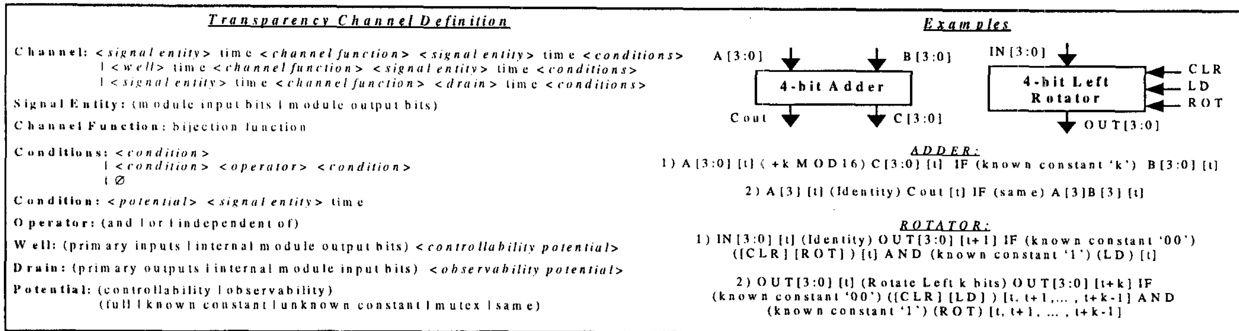

FIGURE 3: TRANSPARENCY CHANNEL DEFINITION AND EXAMPLES

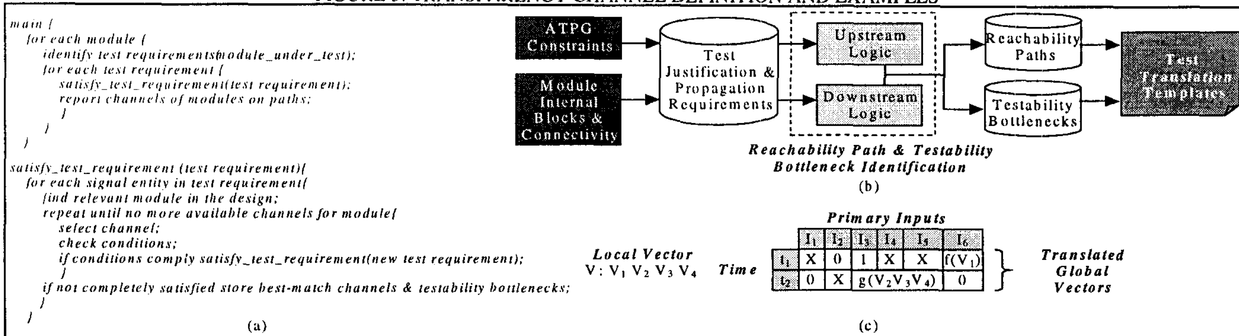

FIGURE 4: DESIGN TRAVERSAL FOR REACHABILITY PATH, TEST TRANSLATION TEMPLATE AND BOTTLENECK IDENTIFICATION

B. *Design Traversal Algorithm*

Starting at the boundaries of the module under test, the recursive design traversal algorithm described in Fig. 4a attempts to identify the channels required from the upstream and downstream modules for test translation. For each module under test, the local test generation constraints and the internal module connectivity are examined and a set of test justification and propagation requirements is defined for the module. Test requirements are expressed as *potential* that needs to be justified or propagated through transparency channels, to the input or the output signal entities of the module under test respectively. The algorithm traverses the design, backtracking as necessary, in order to satisfy the requirements using channel behavior. While traversing an upstream/downstream module, available channels are probed as to their suitability for providing the required potential on the desired signal entity. Channels may be combined into wider channels or broken into smaller ones, provided that appropriate conditions ensure no loss of potential. Factors such as reconvergence and feedback loops are considered in order to prioritize the probing of channels and to accelerate algorithm convergence. The search is very fast since it involves only high-level primitives and does not search exhaustively the design. The algorithm ends when appropriate *wells* or *drains* satisfying the test requirements are encountered. Channels on the reachability paths to the wells and drains are reported and combined into test translation templates for each module. If the requirements cannot be completely satisfied, the best matching set of channels is reported for each module, along with a list of testability bottlenecks, as demonstrated in Fig. 4b.

DFT Modifications

The bottlenecks reported by the traversal algorithm are combined for all modules and a minimal set is obtained. Resolving these bottlenecks guarantees that reachability paths exist to each module in the design. The proposed scheme pinpoints the signals to be enhanced for testability and provides the required potential on each of them. Any type of DFT modification, such as scan or test points, can be subsequently applied. Furthermore, behavioral test synthesis may also be employed for resolving the identified bottlenecks and enhancing module reachability, as shown in (11). More details on DFT modification guidance can be found in (12).

Hierarchical Test Generation

Using the identified reachability paths the locally generated vectors are translated into global design test, either on the original or on the modified design. In order to perform this translation, the channels on the identified vector justification and response propagation paths are combined into test translation templates that capture both the translation paths and the corresponding conditions. In case vectors need to be justified consecutively and paths of appropriate time-extended channels exist, the templates are constructed accordingly. Given each vector, these templates apply the reverse effect of the channel functions on the translation path. This process provides a global vector that when justified through the path will provide the desired local vector, as shown in Fig. 4c. Using these templates, the actual local to global test translation can be rapidly performed. During fault simulation, Xs are randomly filled in order to cover collateral faults.

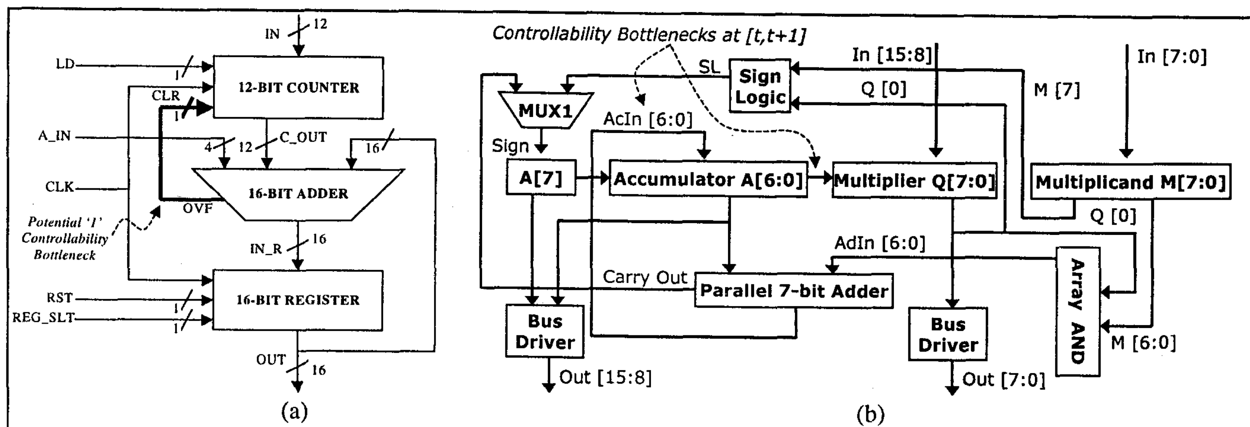

FIGURE 5: EXAMPLE CIRCUITS

Examples & Results

The *TRANSPARENT* system was applied on the two circuits depicted in Fig. 5, a simple 3-block circuit with intricate feedback loop behavior introduced in (7), and an 8-bit binary shift & add multiplier described in (13). As depicted in Fig. 5, testability analysis revealed a few bottlenecks on each design, which were subsequently resolved by using simple test multiplexers. Locally generated test vectors using HITEC (14) were translated into global test, both on the original and on the modified design and subsequently fault simulated using PROOFS (15). The results are summarized in Table 1 and compared in terms of fault coverage, test generation time and vector count to a full-circuit ATPG approach using HITEC. The ability of our system to guide DFT modifications, improve fault coverage and speed-up test generation is thus demonstrated.

Conclusions

We introduced *TRANSPARENT*, an RTL hierarchical test strategy that addresses size and complexity considerations of modern designs in a *divide & conquer* manner. A testability analysis scheme that identifies module reachability paths composed of transparency channels and reveals potential testability bottlenecks in the design was described. Consequently, a methodology for guiding DFT modifications and utilizing the obtained reachability paths for translating locally generated vectors into globally applicable test was discussed. The proposed scheme constitutes a viable alternative to full circuit gate-level ATPG and previous hierarchical test generation approaches, as revealed by the significant fault coverage improvement and test generation time reduction, obtained on example hierarchical designs.

TABLE 1
FULL CIRCUIT GATE-LEVEL ATPG VS. TRANSPARENT

	ORIGINAL		MODIFIED	
Circuit (a)	Gate-Level	Transparent	Gate-Level	Transparent
Coverage	1034/1055	1038/1055	1038/1055	1054/1055
T.G. Time	3.483 sec	0.4 sec	2.895 sec	0.3 sec
Vectors	146	178	124	157
Circuit (b)	Gate-Level	Transparent	Gate-Level	Transparent
Coverage	685/716	693/716	712/716	715/716
T.G. Time	0.25 sec	0.1 sec	0.15 sec	0.07 sec
Vectors	249	210	183	141

References

(1) 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, 1996, pp. 1144-1151.

(2) B. T. Murray, J. P. Hayes, "Hierarchical test generation using pre-computed tests for modules", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 6, 1990, pp. 594-603.

(3) I. Ghosh, A. Raghunathan, N. K. Jha, "Hierarchical test generation and design for testability of ASPPs and ASIPs", Proceedings of the 34th Design Automation Conference, 1997, pp. 534-539.

(4) R. S. Tupuri, J. A. Abraham, "A novel test generation method for processors using commercial ATPG", Proceedings of the International Test Conference, 1997, pp. 743-752.

(5) P. Vishakantaiah, J. A. Abraham, D. G. Saab, "CHEETA: composition of hierarchical sequential tests using ATKET", Proceedings of the International Test Conference, 1993, pp. 606-615.

(6) B. T. Murray, J. P. Hayes, "Test propagation through modules and circuits", Proceedings of the International Test Conference, 1991, pp. 748-757.

(7) 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, 1992, pp. 273-278.

(8) 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, 1985, pp. 56-68.

(9) S. Freeman, "Test generation for data-path logic: the F-path method", *IEEE Journal of Solid State Circuits*, vol. 23, no. 2, 1988, pp. 421-427.

(10) 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, 1998, pp. 105-120.

(11) Y. Makris, A. Orailoğlu, "Channel-based behavioral test synthesis for improved module reachability", Proceedings of the Design Automation and Test in Europe Conference, 1999 (in press).

(12) Y. Makris, A. Orailoğlu, "DFT guidance through RTL test justification and propagation analysis", Proceedings of the International Test Conference, 1998, pp. 668-677.

(13) J.P. Hayes, *Computer Architecture and Organization*, McGraw-Hill, 3rd Edition, 1998.

(14) T. Niermann, J. H. Patel, "HITEC: a test generation package for sequential circuits", Proceedings of the European Conference on Design Automation, 1992, pp. 214-218.

(15) 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, 1990, pp. 535-540.