"Property-based RTL Test Justification and Propagation Analysis"*

Yiorgos Makris and Alex Orailoglu
(makris@cs.ucsd.edu) (alex@cs.ucsd.edu)
Reliable Systems Synthesis Lab
CSE Department MS-0114
University of California San Diego
9500 Gilman Drive, La Jolla CA-92093
Phone: (619) 534-0914 Fax: (619) 534-7029

Extended Abstract:

As the size and complexity of modern digital circuits increases, tools and methodologies for designing and testing them face extremely challenging barriers. Consequently, a number of approaches have been employed in order to alleviate the burdens associated with such ambitious tasks. Among them, functional abstraction from the transistor and gate level to the register transfer and behavioral level and modular decomposition of the complete design into its constituent blocks have provided an edge. The ultimate goal of these methodologies is to address larger designs by operating at a coarser granularity and to improve the tools' sophistication by applying a "divide & conquer" type of approach. In the test domain, the insufficiency of even the state-of-the-art test generation tools to handle large and complex designs has necessitated the utilization of the above methodologies. Functional abstraction has allowed test generation tools to exploit behavioral descriptions of the design in order to generate high-level test, resulting in encouraging but nonetheless, still inadequate results. Modular decomposition, on the other hand, has enabled highly accurate tests to be derived at the structural level for design modules. However, the justification and propagation through the rest of the design of locally derived tests, has imposed a serious obstacle to combining them into a complete and applicable scheme. Attempts to benefit from the advantages of both approaches require employing modular decomposition for the test generation process and functional abstraction to a higher description level for test justification and propagation.

The above observations emphasize the urge for efficient mechanisms that will support the combination of functional abstraction and modular decomposition in order to assure high testability of the complex, modern digital circuit designs. Various researchers have identified the need and have contributed along some of the above directions. Murray and Hayes [MuHa'91] have proposed a test result propagation scheme through modules based on *ambiguity sets*. Although their scheme is well founded from a theoretical algebraic perspective, it results in extreme complexity and cannot tolerate large datapath designs. Vishakantaiah *et al.* [ViAb'92] have proposed a test knowledge extraction methodology for hierarchical designs, wherein behavioral capabilities of the modules are extracted in terms of *modes* and can be further combined in order to provide test justification and propagation paths. The complexity of extracting the modes and globally realizing them has been the limiting factor of this approach. Recently, Tupuri and Abraham [TuAb'97] have proposed a functional test generation method for embedded modules by incorporating the test justification and propagation capabilities of the surrounding logic into the test generation process in the form of *constraints*. However, these constraints have been manually generated.

Our work is focused on deriving an RTL analysis methodology that will assess the ability of a modularly decomposed design to justify and propagate tests and responses, in order to facilitate block level, structural test generation. What distinguishes this work from the aforementioned approaches is the automated extraction and combination of a sufficient set of behavioral properties through an algebraic scheme. Thus, it provides a mechanism for reasoning on justification and propagation capabilities of the design modules within reasonable complexity boundaries. In line with modular decomposition and functional abstraction, we assume a basic underlying test framework as depicted in figure (1). The test methodology targets one block at a time, while the surrounding blocks are grouped into upstream test justification logic and downstream test response propagation logic.

* This work is supported partially through a research grant from Intel Corporation.

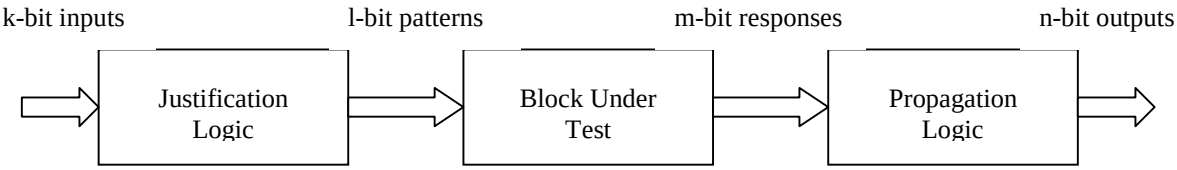


Figure (1): *Basic Underlying Test Methodology*

The naïve way of exhaustively examining the functionality of the upstream justification logic through the k-bit inputs, in order to find all possible l-bit patterns proves unrealistic for large designs, let alone sequential logic, wherein not only patterns but also sequences of patterns need to be realized. The same observation holds for the downstream propagation logic, strengthened by the constraint that any test response needs to be propagated in a distinguishable manner in order to avoid aliasing. Although test vectors and responses are not available in advance, we need to compromise with a methodology that will trade-off between the complexity and the completeness of the results of the analysis in a highly realistic manner. Towards this end, we have derived a property-based algebraic scheme for analyzing the justification and propagation capabilities of the upstream and downstream logic, respectively.

The elements of the algebraic scheme are briefly described in figure (2) accompanied by representative examples that pinpoint the type of analysis performed. We assume an RTL model of the design, wherein we have associated a number of properties from the property set of the algebra with each module of the RTL library. Based on the algebraic *rules*, we apply an algorithmic sequence that tries to satisfy the analysis *objective* by propagating *properties* combined through the algebra's *operators*. During the algorithmic traversal a number of conditions are imposed, both on control logic and on datapath logic. Our current scheme justifies manually compliance of these conditions vis-a-vis each other. Initially, we are targeting datapath circuits, attempting to identify main streams through which values can be propagated.

The properties utilized represent primitive behavioral structures that preserve the capability of bitwise reasoning. This simplification is inevitable in order to derive a tolerable scheme, possibly sacrificing some of the inherent functional capabilities of the design. During analysis, unrealizable values and sequences are identified in terms of signals that cannot be justified or propagated and the root cause is pinpointed based on the existence or non-existence of certain properties. Most of the straightforward limitations of such a scheme, such as feedback loops, sequential dependencies and data vs. control distinctions, are being addressed in the algorithm through either conventions or heuristics. Timing is being taken care of as part of the algebra as well.

General Algebraic Scheme: **Rule = Objective** $\xleftarrow{\text{(Property)}}\rightarrow$ **Objective {Operator Objective}***

Properties Examples:

IDENTITY: OUT [a: b] [t_1] = IN [c: d] [t_2] IF Conditions On
(IN [0: c-1] [t_3], IN [d+1: N] [t_3], IN[c: d] [t_4]), where t_0=test initiation time,
$M \geq b \geq a \geq 0$, $N \geq d \geq c \geq 0$, $t_1 \geq t_2 \geq t0$, b-a=d-c, t_3, t_4 ε [t_0, t_1], $t_4 \neq t_2$

DISPLACEMENT, REPLACEMENT, NEGATION, INCREMENTALITY, INITIALIZATION, THREADING, CONTROLLABILITY, OBSERVABILITY, etc.

Operators Examples: **AND, OR, IND** (Independent of) etc.

Objectives Examples: **JUSTIFY ALL, PROPAGATE ALL MUTEX** etc.

Rules Examples: **JUSTIFY ALL (in)** $\xleftarrow{\text{(Identity)}}\rightarrow$ **JUSTIFY ALL (out)**

JUSTIFY ALL (in) $\xleftarrow{\text{(Displacement with Replacement)}}\rightarrow$ **JUSTIFY ALL (out#1) IND JUSTIFY ALL (out#2)**

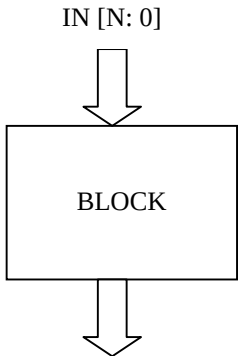
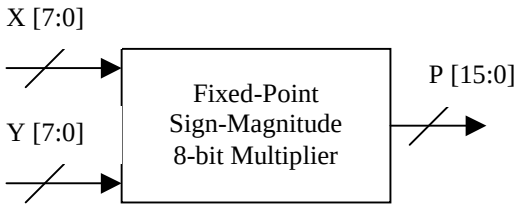


Figure (2): *Algebraic Scheme for Test Justification and Propagation Analysis*



BEHAVIORAL DESCRIPTION:

P [14:0] [t+8] = X [6:0] [t] ***** Y [6:0] [t]
&
P [15] [t+8] = X [7] [t] **XOR** Y [7] [t]

Figure (3): *Behavioral Description of an 8-bit Fixed-Point Sign-Magnitude Binary Multiplier*

We have applied our algebraic scheme on a sequential circuit in order to evaluate its potential. The circuit employed is a fixed-point, sign-magnitude binary 8-bit multiplier described in [Hayes'88]. The behavioral description of the multiplier that is given in figure (3) implies that the only values that we can obtain through the multiplier are all 16-bit numbers that are a product of any two 8-bit numbers. Since we assume fractional magnitudes, the last bit is 0. The set of such resultant numbers is far smaller than the 2^{16} possible outputs. Our analysis has been performed on the RTL model that is provided in [Hayes'88] and has shown that all possible values are realizable at the output, considering the multiplier as the upstream logic. Furthermore, it has shown that when the multiplier is used as downstream justification logic, the most significant bit of the multiplicand (Y [7]) cannot be easily propagated using the algebraic properties employed due to a dependency on datapath values. Similar results on the justification and propagation capabilities of a number of small circuits have been obtained by applying our algebraic analysis scheme.

The knowledge generated by the proposed analysis scheme can be utilized for a number of different tasks. First of all, it can support techniques like [TuAb'97] by automatically providing the constraints needed to increase the efficiency of the test generation process. Furthermore, it can pinpoint the potential testability bottlenecks as far as test justification and propagation is concerned, thus providing a foundation on which a smart DFT decision mechanism can be developed. Since the analysis is based on RTL descriptions of the design datapath, synthesis of the control logic can also be supported in order to provide efficient false paths utilized for test purposes. Also, false datapaths can by utilized during test, a capability that as indicated above cannot be provided through behavioral analysis. Finally, an RTL justification and propagation analysis scheme can be used for tuning the test methodology by exploring the effect of test mode related constraints, such as the number of controllable or observable pins and the number of clock cycles required by the various test patterns.

Having verified the applicability of our scheme on a number of small examples, we are currently implementing a prototype that will be able to automatically apply the RTL testability analysis on large hierarchical designs in order to estimate the test justification and propagation capabilities of each block of the design. Further research work is planned in the directions of automating the control logic constraint justification and the property extraction and associating the results with expected test requirements of the block under test. Furthermore, we are planning to investigate how to judiciously guide decision-taking processes like DFT insertion and high level synthesis in a manner similar to [OrHa'97], through our testability analysis scheme.

References:

[Hayes'88] John P. Hayes, "Computer Architecture and Organization", McGraw-Hill International Editions, 1988.

[MuHa'91] Brian T. Murray and John P. Hayes, "Test Propagation Through Modules and Circuits", Proceedings of International Test Conference 1991, pp. 748-757.

[OrHa'97] Alex Orailoglu and Ian Harris, "Microarchitectural Synthesis for Rapid BIST Testing", IEEE Transactions on CAD of Integrated Circuits and Systems, June 1997, vol.16, no.6, pp. 573-586.

[TuAb'97] Raghuram S. Tupuri and Jacob A. Abraham, "A Novel Test Generation Method for Processors using Commercial ATPG", Proceedings of International Test Conference 1997, pp. 743-752.

[ViAb'92] Praveen Vishakantaiah, Jacob Abraham and Magdy Abadir, "Automatic Test Knowledge Extraction From VHDL (ATKET)", Proceedings of 29th ACM/IEEE Design Automation Conference 1992, pp. 273-278.