DFT GUIDANCE THROUGH RTL TEST JUSTIFICATION AND PROPAGATION ANALYSIS*

Yiorgos Makris and Alex Orailoğlu

Reliable Systems Synthesis Lab
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093

Abstract

We introduce a formal mechanism for capturing test justification and propagation related behavior of blocks. Based on the identified test translation behavior, an RTL testability analysis methodology for hierarchical designs is derived. An algorithm for pinpointing the local-to-global test translation controllability and observability bottlenecks is presented. The analysis results are validated through an ATPG-based experimental flow and the applicability of the scheme for addressing test challenges in large designs by guiding DFT decisions is discussed.

1. Introduction

Silicon manufacturing technology improvements have facilitated an explosion in the size and complexity of modern designs. Consequently, extreme challenges are imposed on tools and methodologies employed in the design and test of complex, digital circuits. Addressing these challenges efficiently becomes increasingly difficult. Test currently ranks among the most expensive and threatening aspects in a circuit design cycle, revealing the imperative need for test-related innovative solutions.

As a result, several techniques and approaches have been developed, both for enhancing the testability of a design through DFT modifications [2, 4, 12, 15] and for improving the test generation and application process [3, 5, 13]. Each approach has a relative cost and efficiency factor associated with it and applies on certain description levels and design types. The non-trivial task of deciding the precise test framework for each design is left to the test engineer. In order to select judiciously among this wide variety of choices, *a priori* testability information of the design is required [3, 7, 14].

Testability analysis is the means of acquiring test knowledge for a design but its scope has been limited so far by a number of challenges arising in modern designs. The substantial size of modern circuits rules out approaches that handle the complete circuit as a monolithic entity. State of the art practices for large designs employ both modular decomposition and functional abstraction for test problem alleviation. Gate-level ATPG is applied at the boundary of each design block for efficient local test generation, while higher description levels are utilized for translation of the local into global test. Consequently, a viable analysis methodology needs to consider the modular nature of the designs and the associated test approaches in order to provide meaningful data.

Furthermore, the severe time-to-market requirements of a typical design impose constraints as to when testability enhancements can be introduced. The complexity associated with test modification and validation of a gate-level description prohibits reasoning at this level. The high level information available at behavioral RTL descriptions addresses both size problems and complexity constraints, making RTL the latest point where efficient testability modifications can be introduced. Consequently, any analysis guiding test-related design changes has to be performed at a higher level than the gate-level description.

Large datapaths, intricate control and complex sequential logic are common features of modern designs, imposing additional burdens on testability analysis. The functional space of such sizable designs cannot be exhaustively examined using traditional behavior capturing mechanisms such as FSMs and BDDs. Any attempt to reason upon the complete functional space on a value-by-value manner is unlikely to succeed. As a result, a judicious way of pruning to a restricted set of test related behavior is necessitated. Temporal reasoning and handling complex control logic remain essential parts of such test related behavior.

Section 2 motivates the proposed approach, presenting a test framework for large, modularly designed

*This work is supported in part through a research grant from the University of California Micro Program and Intel Corporation.

0-7803-5092-8/98 $10.00 © 1998 IEEE

circuits targeted by our analysis methodology. Previous work in the area is discussed and the associated challenges are pinpointed. These challenges are addressed in section 3, through a behavioral formalism that captures test-related behavior of blocks in a design. The formalism is presented, followed by a number of examples on small combinational and sequential blocks and a discussion of the potential of the scheme. Section 4 describes an RTL testability analysis methodology that exploits the test-related behavior through an algebraic scheme and identifies the test justification and response propagation bottlenecks. The analysis algorithm is provided and demonstrated on an example circuit. Section 5 presents an experimental validation flow, based on the use of public domain tools and provides data obtained from benchmark circuits. Section 6 concludes with a discussion of the applicability of our analysis results for enhancing testability of a design through either DFT modifications or test synthesis and motivates further research in the area.

Our research aims at providing an early testability assessment tool for large, modularly designed circuits, capable of reasoning in a uniform fashion on combinational or sequential, data or control path. The goal of our analysis is the identification of the testability bottlenecks associated with the test justification and propagation processes at the RTL. The structural nature of the bottlenecks identified through our analysis provides concise data for guiding design testability enhancements in an informed manner.

2. Research Motivation

The described research stems from the need to address the challenges associated with testing large, modularly designed circuits. We describe a framework that is used to facilitate an effective test methodology for such circuits and that is consequently often employed in real designs. We refer to previous research efforts invested in the area and discuss the challenges arising from this methodology.

2.1 Test Framework

The approach proposed in this paper targets large designs that are decomposable into modules. During test application, each block is tested individually while the remaining blocks are grouped into upstream test justification logic and downstream response propagation logic. As a result of feedback loops, the block under test, the justification logic and the propagation logic may overlap. Test is initiated at the primary inputs and terminated at the primary outputs as depicted in figure (1).

We define further the relevant test framework for the proposed analysis by outlining the following assumptions:

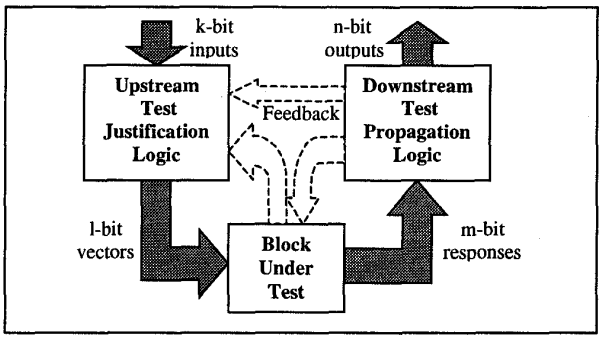


Figure (1): *Underlying Test Framework.*

i) Gate-level test generation is performed for each block; local vectors, expected block pinout responses and associated fault coverages are thus obtained. The local tests are subsequently translated to global test at the primary inputs and outputs. The actual local test generation and translation processes do not affect our analysis methodology. It is assumed that a partitioning into local blocks is informed by the necessity to generate local vectors of high and appropriate fault coverage. As such, the proposed methodology is primarily concerned with preserving the high local coverages by ensuring propagation of the local vectors to primary pinouts.

ii) RTL is the latest point in the design cycle where modifications for testability enhancement are typically allowed. It follows that RTL is the latest level at which our analysis methodology can be performed, if it is to facilitate testability modifications. The analysis methodology aims at enhancing the results of the test translation process. Figure (2) provides a temporal view of the test framework. The analysis objective is to preserve intact after the translation process the levels of local fault coverage (LFC) within the global fault coverage (GFC). The lack of precise knowledge of test vectors during testability analysis adds an additional layer of complexity and challenge to the task, while extending its applicability to additional and highly meaningful test contexts.

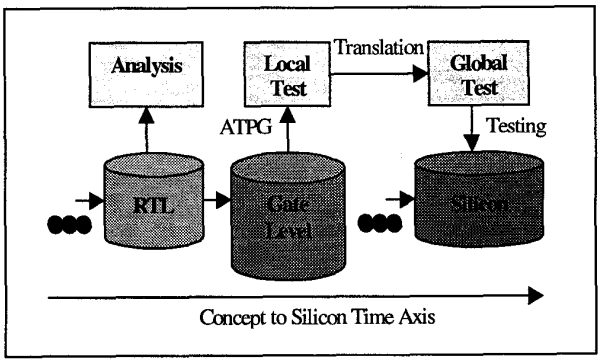


Figure (2): *Temporal View of Analysis Goal.*

2.2 Related Work

Several researchers have identified the benefit of combining functional abstraction and modular decomposition, in order to assure high testability of large designs. Murray and Hayes [8] propose a test result propagation scheme through modules, based on *ambiguity sets*. Vishakantaiah *et al.* [14] present a test knowledge extraction methodology for hierarchical designs, wherein behavioral capabilities of the modules are extracted in terms of *modes*. Ghosh *et al.* [5] describe a similar DFT and test generation technique for core-based systems. Recently, Tupuri and Abraham [13] introduce 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*. Also, Chen *et al.* [2], Corno *et al.* [3] and Lee and Patel [7] suggest various approaches for addressing RTL testability analysis.

2.3 Challenges

Various challenges are associated with identifying test translation bottlenecks. We distinguish two major analysis tasks and discuss the challenges associated with each.

i) Justification and propagation of exact test vectors for translation bottleneck identification is highly computationally expensive since the complexity of numerous individual test translation tasks in such large scale designs would be overwhelming. Furthermore, any changes in the actual test set would invalidate the analysis results. Identification of a set of sufficient symbolic test justification and propagation requirements for each module, coupled with an analysis of the design as to its capability to satisfy these requirements, possibly offers a solution. An identification of independent cones of logic, as discussed in section 4, is suggested as a heuristic approach in this direction.

ii) Further complications are associated with the local to global test translation, as demonstrated in figure (3), using a small circuit, originally presented in [14].

- Exhaustive examination of the complete functional space of the modules in the design can be very expensive. Large datapaths (e.g. 16-bit adder), sequential logic (e.g. 2^{12} states of the 12-bit counter), and inter-module complex behavior (e.g. feedback from OUT to the adder) prohibit exhaustive functional reasoning during the translation. Therefore, a subset of the complete functional space that captures the most useful test translation behavioral features needs to be identified. In the example of figure (3), utilization of the LD capability of the counter for value justification to the rest of the circuit eliminates the need to reason upon the complex counter FSM. Section 3 describes a mechanism for capturing test translation behavior.

- Identification of the appropriate domain for examining the test translation behavior of a design is a critical decision. The output of the adder in figure (3) can be examined either in the value domain through algebraic reasoning or in the signal domain through boolean logic on signal entities. The value domain allows exploitation of arithmetic properties while the signal domain supports complex control logic and variable bit-widths. Our scheme examines both attributes, while preserving a structural reasoning mechanism that resembles the nature of most DFT techniques.

- Modular traversal does not necessitate the use of the complete word width. For example, while propagating the counter's test responses over the adder in figure 3, we only need to reason on the sub-word that comprises the lowest 12 bits; the most significant 4 bits do not add material information to test translation. An efficient scheme requires handling of dynamic changes in the signal bit-width.

- Such dynamic variance impedes the *a priori* extraction of module transparency behavior. The connectivity model of the complete design and the test requirements need to be taken into account. For example, the fact that one of the inputs of the adder splits into a 4-bit and a 12-bit signal hints at the necessity of extracting *on-the-fly* transparency behavior for these signal bit-widths.

- Sequential logic, reconvergent paths and feedback loops further complicate test translation. In figure (3), sequences of two vectors are necessary in order to test the register, creating a cycle, since the register is in its own justification path. Moreover, the feedback line from the adder's OVF to the counter's CLR signal, along with the FSM behavior of the counter, compose a complicated sequential logic that is difficult to capture. Handling sequential logic and cyclic behavior is an inseparable part of a viable test translation analysis scheme.

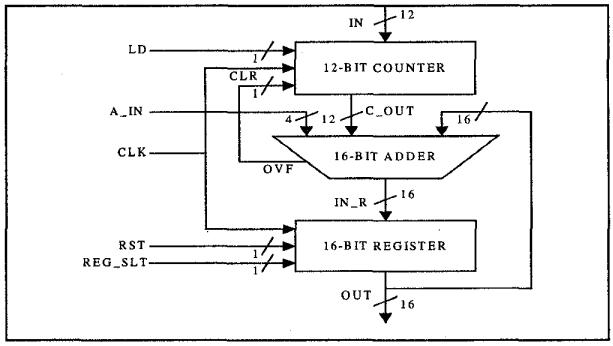


Figure (3): *Example Demonstrating Challenges.*

3. Behavioral Formalism

A behavioral formalism assisting testability analysis is presented and demonstrated on simple example circuits and its capabilities discussed. The introduced formalism captures a restricted set, comprised only of test-related behavior of blocks. The transparency aspects of the blocks are captured in a structural manner *on-the-fly*, through the notion of *channels*. Our formalism handles variable width signals and feedback loops in a stream-wise fashion. It additionally possesses the ability to reason in the temporal domain. The formalism facilitates consequently an efficient scheme for test analysis that handles uniformly combinational or sequential, data or control path circuits.

The constituent components of the proposed behavioral formalism are defined below:

Channel: A *channel* is defined to be a *mapping* between an input *signal entity* and an output *signal entity* or between a *well* and an output *signal entity* or between an input *signal entity* and a *drain*, based on the compliance of a set of zero or more *conditions*.

Mapping: A *mapping* is defined as a one-to-one and onto function from the set of possible values of an input *signal entity* or a *well*, to the set of possible values of an output *signal entity* or a *drain*.

Signal Entity: A *signal entity* represents a bundle of one or more signals at a certain point in time. Both the signals and the time point can be defined either statically or dynamically. Dynamic definition is a collective way of capturing many static definitions, enabling increased flexibility and generality. As an example, a static definition of a signal entity might be *"IN[2] at [t]"* while a dynamic definition would look like *"ONE OF IN[3], IN[2], IN[1] at any t where* $t>t_0$*"*. The dynamic definition can be a choice from either a list of alternatives or from a closed type set definition.

Well: A *well* is the equivalent of a controllability point and captures the ability of a module to generate vectors on signal entities. It also captures the controllability of the primary inputs. Each well has associated with it a *potential* on signal entities.

Drain: A *drain* is the equivalent of an observability point and captures the ability of a module to evaluate vectors on signal entities. It also captures the observability of the primary outputs. Each drain has associated with it a *potential* on signal entities.

Potential: The *potential* of a well or a drain captures the type of vectors that can be generated or evaluated on a signal entity. For example, a full potential well of width k can generate the complete set of all possible 2^k vectors and a full potential drain of width k can evaluate likewise the complete set. The potential of a well or a drain is not defined in the value domain but rather in the signal domain. This implies that a well cannot generate a set of specific values unless a structural property defining the elements of the set in the signal domain (e.g. properties such as *mutual exclusion*, *same*, *inverse*) exists. These sets are captured in a stream-wise manner, avoiding the complete functional space complexity problem.

Conditions: *Conditions* are defined on one or more input signal entities and are combined through *operators*. In order for a channel to be activated, the conditions associated with it need to be satisfied.

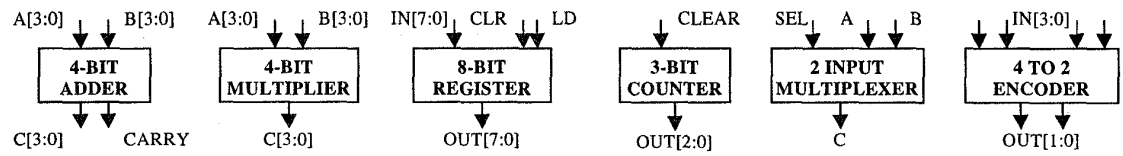
Operators: *Operators* between signal entities are employed to create the above conditions. These operators can be either logical (e.g. *AND*, *OR*), stream-related (e.g. *INDEPENDENT OF*), or arithmetic (e.g. $=, \neq, \leq \geq$). The only requirement is that they have to be expressed in a stream-wise fashion. As discussed before, such restrictions to stream reasoning reduce drastically the sizable complexity of the translation process, thus underscoring the inability to perform a value-based analysis directly. Instead, a highly generic, stream-wise reasoning imitates implicitly such analysis.

Time-Extended Channels: A *time-extended channel* is a sequence of channels instantiated in consecutive clock cycles, with respect to the input or output signal entity.

A number of channel examples for basic RTL blocks are shown in figure (4), displaying how the behavioral formalism captures block traversal capabilities. These capabilities are utilized by the algorithm described in section 4 for test justification and propagation in terms of *objectives*. The primary strengths and weaknesses of the behavioral formalism are outlined as follows:

- The introduced behavioral formalism constitutes an abstraction scheme that decouples the actual DFT modification and the traversal capability extraction process from testability analysis, providing a clean interface mechanism between them. Consequently, each of these tasks can be individually addressed.

- Although the formalism does not cover the complete behavioral space of the modules, it handles variable bit-widths of input and output signal entities through the well and drain notions and applies not only at the full word size but also on sub-word signal entities. The ability of the channels to express any arbitrary bijection function brings our scheme as close to the concept of transparency as possible.



Channels:

ADDER

1) ONE OF {A[3:0], B[3:0]} at [t] MAPS TO C[3:0] at [t] IF CONSTANT POTENTIAL WELL at [t] MAPS TO ONE OF {B[3:0], A[3:0]} at [t]
2) FULL POTENTIAL WELL at [t] MAPS TO CARRY at [t] IF ALL SAME (A[3], B[3]) at [t]

MULTIPLIER

1) ONE OF {A[3:0], B[3:0]} at [t] MAPS TO ANY 4 CONSECUTIVE BITS OF C[6:0] at [t] IF ALL MUTEX (ONE OF {B[3:0], A[3:0]})

REGISTER

1) IN[n] at [t_0...t_0+k] MAPS TO OUT[n] at [t_0+1...t_0+1+k] IF CONSTANT '1' POTENTIAL WELL at [t_0...t_0+k]] MAPS TO LD at [t_0...t_0+k] AND CONSTANT '0' POTENTIAL WELL at [t_0...t+k] MAPS TO CLR at [t_0...t_0+k] where n=[0...7] (Time-extended channel of depth k)

COUNTER

1) FULL POTENTIAL WELL at [t] MAPS TO OUT[3:0] at [t+1] IF FULL POTENTIAL WELL at [t] MAPS TO OUT[3:0] at [t] AND CONSTANT '0' POTENTIAL WELL MAPS TO CLEAR at [t]
2) FULL POTENTIAL WELL at [t] MAPS TO OUT[1] at [t+1,t+2] IF FULL POTENTIAL WELL at [t] MAPS TO OUT[0] at [t] AND CONSTANT POTENTIAL WELL at [t] MAPS TO OUT[2] at [t] AND CONSTANT '0' POTENTIAL WELL MAPS TO CLEAR at [t]
(Time-extended channel of depth 2)

MULTIPLEXER

1) ONE OF {A,B} at [t] MAPS TO C at [t] IF FULL POTENTIAL WELL at [t] MAPS TO SEL at [t]

ENCODER

1) FULL POTENTIAL WELL at [t] MAPS TO C at [t] IF ALL SAME (A,B) at [t]
2) FULL POTENTIAL WELL at [t] MAPS TO OUT[1:0] [t] IF ALL MUTEX (IN[3:0]) at [t]

Figure (4): *Examples of Channels in Simple RTL Blocks.*

- There is more than one way to describe a traversal through channels. Extracting all the channels in advance is not practical. A similar problem limits the approaches described in [7, 14]. In order to address this issue, we rely on an *on-the-fly* channel selection for our test justification and propagation analysis, as explained further in section 4.

- The time-extended channels provide a capability to reason on sequences of vectors and responses, thus facilitating sequential logic test analysis. This proves valuable in the cases where a clock disable signal is not provided for sequential blocks.

The introduced formalism provides a concise and efficient way to capture behavior related to test justification and propagation, in a structural manner. In conjunction with the structural nature of DFT techniques, it facilitates a realistic testability analysis methodology.

4. Analysis Methodology

The central ideal of our analysis methodology is the examination of the satisfiability of test justification and propagation *objectives* defined at the boundaries of each block.

Objective: An objective is the justification or propagation of a *potential* defined on one or more *signal entities* related through *operators*. The potential of a signal entity is the join of the potential of all the wells and drains that can map to the signal entity through one or more paths of *channels*.

Objectives are examined using an algebraic scheme that utilizes the notion of channels. The algorithm employed for objective examination for a block, as well as for optimization of the complete circuit results, is discussed below.

4.1 Algorithm

The objective examination for a particular block is a two-step process. First, the objectives have to be identified at the boundary of the block under test. Subsequently, their satisfiability needs to be analyzed, in order to pinpoint the potential bottlenecks.

i) Objective Identification: Since the actual test and responses are not known at the time of analysis, the simplest yet costly approach is an attempt at justification of the complete set of all possible values and their

responses from block to chip pinouts. In order to alleviate this cost, we examine the block 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 block, through which the dependent sets of inputs and outputs can be extracted and used for defining more realistic objectives in a stream-wise manner. The sequential depth of the block also affects the justification and propagation objectives, pinpointing their temporal requirements. Figure (5) depicts the justification and propagation objectives of a 4-bit register. The register's sequential depth of one necessitates the application of two consecutive vectors for test.

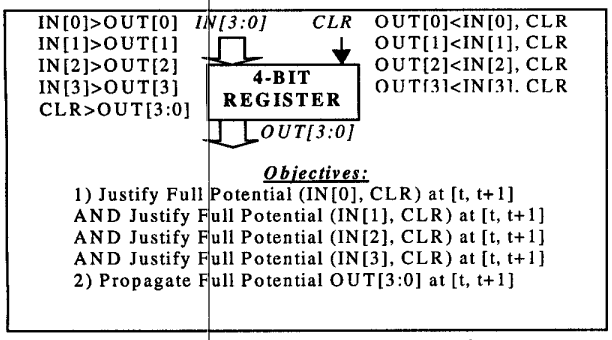


Figure (5): *Cones of Logic Identifying Objectives.*

ii) Objective Satisfiability: Our algebraic scheme is based on objective transformation through the notion of *channels*. Starting from the signal entities on which the objective is defined, we utilize the global circuit connectivity information to identify the source and sink blocks for each signal entity. We select subsequently a *channel* for these blocks that will transform the *objective* into a number of *objectives* at the other boundary of the block, related through *operators* and a number of *conditions*. Compliance of a condition implies possible channel inclusion in the test plan and in turn initiates examination of further objectives recursively. Conflicts in condition compliance necessitate retries and backtracking. At the end, objectives are either satisfied through well and drain capabilities or they prove unsatisfiable, in which case they are reported as bottlenecks.

These two steps are described in pseudocode in the algorithm of figure (6). The primary aspects of the algorithm are highlighted in the following discussion.

Channel Selection: Selecting judiciously among the alternative channels or combination of channels that satisfy an objective over a block has a significant impact on the amount of backtracking performed. A greedy approach, *best match first*, is currently employed but only after a number of *decision factors* are examined. These factors can be either *static* or *dynamic*. Static factors are extracted from the circuit connectivity model and include the number of signal entities and conditions involved, the number of clock cycles and the possible formation of loops. Dynamic factors capture the maximum potential that has been satisfied on each signal entity and are preserved across block examination.

Condition Compliance: The conditions that the channels generate when traversing blocks and transforming objectives recursively define additional objectives. Such condition compliance is checked first since the potential required for conditions is usually simpler than the potential required on the associated justification or propagation paths. Additionally, conditions may possibly require concurrent reasoning on more than one path. Justification of the *ALL SAME (a, b)* potential on two signals *a* and *b*, for example, requires concurrent identification and examination of the wells mapping to both signals.

Bottlenecks: Wells and drains are used to satisfy justification and propagation objectives, respectively. The signal entities on which desired objectives remain unsatisfied after the final channel selection are reported as bottlenecks, along with the potential required.

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


main { repeat until no more blocks {
        select_block;
        for each block {
           identify_objectives(block_under_test);
           for each objective {
              satisfy_objective(objective);
              report bottlenecks of blocks on final path;
              }
            }
          }
        combine_bottlenecks;
}
```

Figure (6): *Objective Examination Algorithm.*

Block Ordering Heuristic: The existence of dynamic factors forces results for each block to be dependent on the sequence of block examination and channel selection. In order to address this problem efficiently and obtain a minimum set of bottlenecks, we employ a heuristic that prioritizes the blocks for justification, as well as for propagation, analysis. We assume that each signal entity in our design is an equiprobable bottleneck and we calculate the number of signal entities involved in examining each block. We then identify the block that shares the maximum number of signal entities with the remaining blocks. This assures that as many signal entities as possible have a dynamic factor associated after each block is examined, thus informing the channel selection process. The scheme is repeated until all blocks are examined.

Bottleneck Combination Heuristic: A second heuristic is employed for combining the results of the individual block analysis and minimizing the reported bottlenecks. Blocks are examined once more in the same order; this time though appropriate DFT modifications are assumed on the design. For each block examined, the resolution of the bottlenecks identified through the analysis of the remaining blocks is assumed. A new set of bottlenecks, possibly disparate from the one provided by the first heuristic, is obtained for the block. The minimal set of signal entities that covers all reported bottlenecks is selected as the final set of bottlenecks to be resolved.

These heuristics attempt to combine and minimize the reported bottlenecks, enhancing the accuracy and validity of our analysis methodology. The two heuristics have significantly improved the results and a separate evaluation of their effect on the proposed scheme is planned.

4.2 Example

Our analysis methodology is applied and demonstrated in detail on an RTL model of a pipelined multiplier accumulator shown in figure (7). The algorithm and both the behavioral and RTL VHDL code for the MAC can be found in [1]. The circuit is mainly datapath oriented and comprises combinational and sequential blocks, feedback loops and reconvergent paths and a small control-like logic for the overflow calculation.

Justification analysis for the register REG22#1 and propagation analysis for the multiplier MUL#4 are provided in detail. Starting with the objectives, we reach a conclusion for each of them, revealing the relative testability bottlenecks. The results of the block ordering algorithm for justification and propagation are also provided. Finally, the complete set of testability bottlenecks identified through the proposed methodology is reported. As discussed in section 2, the analysis is performed without the actual test stimuli and responses available; the set of reported bottlenecks may consequently not be minimal for a given set of preferred test vectors.

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 used to evaluate the ability of the algorithm to identify test bottlenecks. The comparison fundamentally relies on a similar translation of locally generated test vectors to both the original and to a version of the design modified in light of the testability analysis guidelines. By exploring the effects of the analysis guidelines, we outline the impact of the modifications on a relevant and emerging test generation approach for state-of-the-art designs.

5.1 Validation Flow

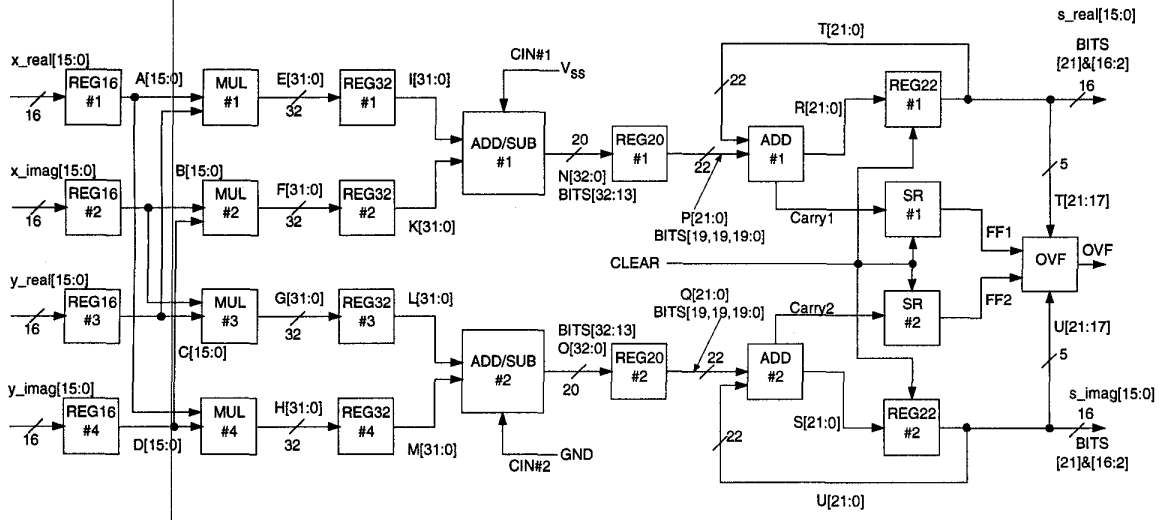
An overview of the validation flow is provided in figure (8). In compliance with the test framework for which the analysis methodology has been derived, our primary validation mechanism utilizes HITEC [9], a gate-level ATPG tool, and is based on fault coverage comparison acquired from the fault simulator PROOFS [10]. Starting with an RTL description of the circuit, the described analysis methodology is applied resulting in a list of controllability and observability bottlenecks. A gate-level model is further obtained through synthesis, on which the ATPG experiments are performed.

First, the ATPG tool is applied independently for each design block, resulting in local tests. Each local test is subsequently translated to the boundaries of the complete circuit and global test is obtained and fault simulated to provide the associated global fault coverage (GFC). The same experiment is then performed on an enhanced version of the design, wherein all the justification and propagation bottlenecks identified by the analysis are considered to be fully controllable or observable. The modified GFC, shown in the final column of table (1), is thus obtained. The underlying translation of the local to global test is manually performed.

The analysis of the results delineates among the following classes of uncovered faults:

CLASS-A: Faults that an ATPG invocation at the block level covers but nonetheless remain untested due to test translation shortcomings. CLASS-A faults force GFC to break even at best with the sum of the local fault coverages. The analysis identifies the reasons for particular CLASS-A faults; appropriate design for test modifications can then be employed to alleviate the problem. The GFC of the circuit thus modified should exceed that of the original one and should approximate closely the sum of the local fault coverages.

RTL BLOCK DIAGRAM OF PIPELINED MULTIPLIER ACCUMULATOR (MAC)



JUSTIFICATION OBJECTIVE EXAMINATION EXAMPLE: REG22#1

Objective: JUSTIFY (R[k], CLEAR), k=[0...21] at [t_0, t_0+1] ⇒ *THROUGH PRIMARY INPUT WELL CAPABILITIES AND ADD#1 CHANNEL (P[k] at [t] MAPS TO R[k] at [t] IF T[21:0]=constant)* ⇒ JUSTIFY P[k] at [t_0, t_0+1] **(justification of condition verifies its compliance through *clear='1' at[t_0-1] over feedback loop)*** ⇒ *THROUGH REG20#1 CHANNEL (N[j] at [t] MAPS TO P[t] at [t+1], j=[32...13])* ⇒ JUSTIFY N[j] at [t_0-1,t_0] ⇒ *THROUGH ADD/SUB#1 CHANNEL (I[j-1] at [t] MAPS TO N[j] at [t] IF SAME (I[j-1], K[j-1]) at [t])* ⇒ JUSTIFY SAME (I[j-1], K[j-1]) at [t_0-1, t_0] ⇒ *THROUGH REG32#1,2 CHANNELS (E[j-1] at [t] MAPS TO I[j-1] at [t+1], F[j-1] at [t] MAPS TO K[j-1] at [t+1])* ⇒ JUSTIFY SAME (E[j-1], F[j-1]) at [t_0-2, t_0-1] ⇒ *THROUGH MUL#1,2 CHANNELS (A[m] at [t] MAPS TO at E[n] at [t], m=[15...0], n=[31, 28...0], B[m] at [t] MAPS TO F[n] at [t])* ⇒ JUSTIFY SAME (A[m], B[m]) at [t_0-2, t_0-1] **(Controllability bottlenecks E[30:29], F[30:29])** ⇒ *THROUGH REG16#1,2 CHANNELS (x_real[m] at [t] MAPS TO A[m] at [t+1], x_imag[m] at [t] MAPS TO B[m] at [t+1])* ⇒ JUSTIFY SAME (x_real[m], x_imag[m]) at [t_0-3, t_0-2] ⇒ *THROUGH PRIMARY INPUT WELL CAPABILITIES* ⇒ **Satisfied**

PROPAGATION OBJECTIVE EXAMINATION EXAMPLE: MUL#4

Objective: PROPAGATE H[31:0] at [t_0] ⇒ *THROUGH REG32#4 CHANNEL (H[31:0] at [t] MAPS TO M[31:0] at [t+1])* ⇒ PROPAGATE M[31:0] at [t_0+1] ⇒ *THROUGH ADD/SUB#2 CHANNEL (M[31:0] at [t] MAPS TO O[31:0] at [t] IF CONSTANT POTENTIAL L[31:0] at [t])* ⇒ JUSTIFY CONSTANT POTENTIAL L[31:0] at [t_0+1] **(justification of condition verifies its compliance)** AND PROPAGATE O[31:0] at [t_0+1] ⇒ *THROUGH REG#2 CHANNEL (O[32:13] at [t] MAPS TO Q[19:0] at [t+1])*⇒ **(Observability bottleneck on O[12:0])** PROPAGATE Q[19:0] at [t_0+2] ⇒ *THROUGH ADD#2 CHANNEL (Q[21:0] at [t] MAPS TO S[21:0] at [t] IF CONSTANT POTENTIAL U[21:0] at [t])* ⇒ JUSTIFY CONSTANT POTENTIAL U[21:0] at [t_0+2] **(justification of condition verifies its compliance)** AND PROPAGATE S[21:0] at [t_0+2] ⇒ *THROUGH REG22#2 CHANNEL (S[21:0] at [t] MAPS TO U[21:0] at [t])* ⇒ PROPAGATE U[21:0] at [t_0+3] ⇒ *THROUGH DRAIN CAPABILITIES OF PRIMARY OUTPUTS AND OVF BLOCK* ⇒ **Satisfied (Observability bottleneck on U[1:0])**

ORDERING ALGORITHM RESULTS

JUSTIFICATION	*PROPAGATION*
1) REG22#1,2	*1) REG16#1,2,3,4*
2) OVF	*2) MUL#1,2,3,4*
3) SR#1,2	*3) REG32#1,2,3,4*
4) ADD#1,2	*4) ADD/SUB#1,2*
5) REG20#1,2	*5) REG20#1,2*
6) ADD/SUB#1,2	*6) ADD#1,2*
7) REG32#1,2,3,4	*7) REG22#1,2*
8) MUL#1,2,3,4	*8) SR#1,2*
9) REG16#1,2,3,4	*9) OVF*

LIST OF BOTTLENECKS

CONTROLLABILITY

1) Full Potential on E[30:29], F[30:29],G[30:29], H[30:29], P[21:20], Q[21:20], E[11:0], F[11:0], G[11:0], H[11:0] at [t, t+1]

2) Constant '0' Potential on CIN#1 at [t]

3) Constant '1' Potential on CIN#2 at [t]

4) Full Potential on (ONE OF {E[28:27], E[13:12], E[12]E[28]}), (ONE OF {F[28:27], F[13:12], F[12]F[28]}), (ONE OF {G[28:27], G[13:12], G[12]G[28]}), (ONE OF {H[28:27], H[13:12], H[12]H[28]}) at [t, t+1]

OBSERVABILITY:

Full Potential on N[12:0], O[12:0], T[1:0], U[1:0] at [t, t+1]

Figure (7): Analysis Example on a Pipelined Multiplier Accumulator (MAC).

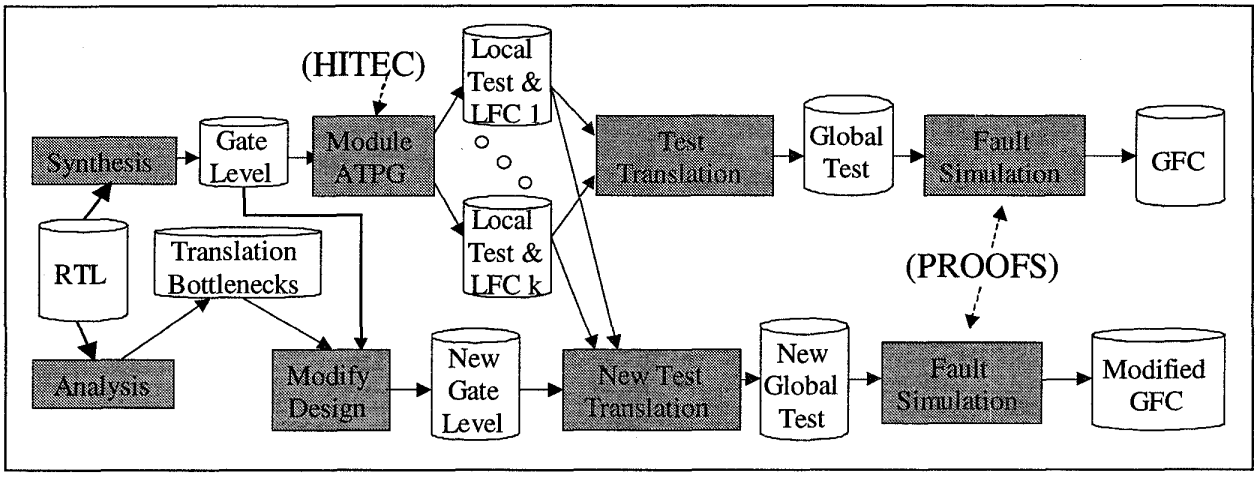


Figure (8): *Experimental Validation Flow and Results.*

CLASS-B: Faults that an ATPG invocation at the block level does *not* cover. CLASS-B faults are not targeted explicitly by the proposed analysis methodology.

5.2 Experimental Data

We apply the above experimental flow on TC100, the circuit of figure (3), a binary sign-magnitude multiplier described in [6], with and without the controller, and the pipelined multiplier accumulator (MAC) described in [1]. The analysis methodology indicates that TC100 and the binary multiplier without the controller are highly transparent and that no major test translation bottlenecks exist in the design. Consequently, we expect translation of local into global test to be highly successful. Summing up the total faults covered by the local tests and dividing by the local number of faults, we obtain 96.68% and 97.45% coverage, respectively. After the local test is translated to test applicable at chip pinouts, coverage drops slightly to 92.80% and 93.27%, due to a small number of minor bottlenecks. A subsequent translation after bottleneck resolution through test point insertion increases the coverage to 95.74% and 96.65%, indicating that our methodology identifies accurately the bottlenecks of the test translation process. When the controller of the multiplier is also considered, the corresponding loss of coverage due to the test translation bottlenecks is considerably larger, resulting in a drop from 94.52% to 68.46%. Such precipitous fault coverage drops are not surprising in this case, since the controller effects multiple modules. Resolving the bottlenecks goes a long way towards eliminating the translation problems, as can be seen by the drastically improved coverage of 90.64%.

In the case of the MAC, our analysis identifies numerous justification and propagation bottlenecks in the design. The coverage drops from 96.44%, achieved by the local tests, to 70.22%, achieved by the translated test on the original circuit. After the bottlenecks are resolved, the coverage of the new translated test increases to 91.35%, validating the accuracy of the reported bottlenecks.

The results are summarized in table (1). The faults that remain uncovered after bottleneck resolution are identified to belong to the multiplexers that were used for inserting test points. Except from these faults that do not belong to the original circuit, our analysis methodology identifies precisely the test translation bottlenecks, as indicated by the above results.

Circuit	Σ(LFC)	GFC	Modified GFC
TC100	96.68%	92.80%	95.74%
Multiplier Without Control	97.45%	93.27%	96.65%
Multiplier With Control	94.52%	68.46%	90.64%
MAC	96.44%	70.22%	91.35%

Table (1): *Experimental Data*

6. Testability Guidance

The behavioral formalism of section 3 serves not only as a tool for carrying out the analysis methodology of section 4, but also as a formal way of expressing the results. As discussed in the overview of the MAC example, the testability bottlenecks are captured as controllability or observability enhancements necessary for signal entities, along with the potential required. Consequently, analysis results can be utilized in a number of different ways for

guiding testability enhancements in a circuit. We focus on two main directions, DFT and test synthesis.

Existing solutions in DFT, such as [5, 12, 15], add to or modify the design to provide additional test capabilities. The formal way of describing the potential problems can be extended to describe a set of available DFT modifications (e.g. SCAN, Test Multiplexers, Clock Disable) in terms of their potential, based on which, an automated process that will optimize and perform the necessary changes can be derived. Furthermore, design guidelines that provide the required potential on the problematic signal entities can be developed. Such guidelines could also support the binding phase of high-level test synthesis in a fashion similar to [11], or direct a logic synthesis tool as in [2], to provide the additional capabilities while creating the gate-level representation of the design.

7. Conclusion

Efficient test engineering decisions rely on accurate and precise testability analysis information, available as early in a design cycle as possible. Towards this direction, we have introduced an efficient formalism for capturing transparency related behavioral aspects of a block, in a structural manner. Our scheme avoids the common complexity pitfall of examining the complete behavioral space and enhances the scope of exploited transparency through the structural notion of *channels*. As a result, scalability of our techniques for large, complex circuits is expected. Additionally, an RTL testability analysis methodology for a particular test framework is derived, addressing both combinational and sequential, data and control path blocks. This methodology has been applied on a number of benchmark circuits and the accuracy of the identified bottlenecks has been validated through an ATPG-based experimental flow. The proposed analysis provides a structural view of the test justification and propagation bottlenecks of the design, facilitating a concise decision-making mechanism for testability enhancements through either test synthesis or DFT modifications. Our future research plans focus on interfacing the proposed analysis methodology to existing DFT and test synthesis approaches and on exploring new techniques, suitable to its structural nature.

References

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

[2] 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.

[3] 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.

[4] I. Ghosh, A. Raghunathan, N.K. Jha, "A design for testability technique for RTL circuits using control/data flow extraction", *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, 1996, pp. 329-336.

[5] I. Ghosh, N. Jha, S. Dey, "A Low-Overhead Design for Testability and Test Generation Technique for Core-Based Systems", *Proceedings of the International Test Conference*, 1997, pp. 50-59.

[6] J.P. Hayes, *Computer Architecture and Organization*, Mc-Graw-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] B.T. Murray, J.P. Hayes, "Test Propagation through Modules and Circuits", *Proceedings of the International Test Conference*, 1991, pp. 748-757.

[9] T. Niermann, J. Patel, "HITEC: a test generation package for sequential circuits", *Proceedings of the European Conference on Design Automation*, 1992, pp. 214-218.

[10] T. Niermann, W. T. Cheng, J. Patel, "PROOFS: a fast, memory efficient sequential circuit fault simulator", *Proceedings of the 27th ACM/IEEE Design Automation Conference*, 1990, pp. 535-540.

[11] A. Orailoğlu, I. Harris, "Microarchitectural Synthesis for Rapid BIST Testing", *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 16, no. 6, pp. 573-586, 1997.

[12] B. Pouya, N. Touba, "Modifying User-Defined Logic for Test Access to Embedded Cores", *Proceedings of the International Test Conference*, 1997, pp. 60-68.

[13] 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.

[14] 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.

[15] P. Vishakantaiah, T. Thomas, J.A. Abraham, M.S. Abadir, "AMBIANT: Automatic Generation of Behavioral Modifications for Testability", *Proceedings of the IEEE International Conference on Computer Design*, 1993, pp. 63-66.