CERBERUS: A Hierarchical DFT Rule Checker

Reinhard Knopf  Erwin Trischler

Siemens AG
Department E-STE 44
P.O. Box 700079
D-8000 Munich 70, West Germany

Abstract
The hierarchical design for testability (DFT) rule checker CERBERUS has been developed to handle hierarchical circuits supporting a variety of scan structures with different types of scanable storage devices. CERBERUS utilizes a general approach to testability rule checking and is part of a Siemens' CAE system to integrate design and test.

1 Introduction
Recent advances in VLSI technology made it possible to design very large application specific integrated circuits (ASIC's). The size of these ranges from several thousands to nearly a hundred thousands of gates. VLSI circuits promise lower cost per gate and better product performance. However, an effective product strategy requires the developer to provide more functions with higher performance at a shorter design schedule, while holding development costs at a constant level. That means, VLSI and especially ASIC's require increased productivity.

Because there is only limited accessability to internal nodes of a circuit, testing of ASIC's becomes much more difficult. The economic impact of testing operations, however, is substantial so testability is now receiving considerable attention. Consequently, DFT has become widely accepted by the electronics community.

The design techniques associated with DFT refer to the common aim of enhancing the controllability, observability and partitionability of digital circuits. They are divided into two classes [WiPa83]:
- Ad hoc techniques intend to enhance the testability of a completed functional design
- Structured methods result in a general logic structure achieved during the design phase.

Whereas ad hoc techniques are often in use with circuits of several thousands of gates, structured DFT approaches appear to be imperative for larger circuits (>10**4 gates) [Derv88], [RiHo88].

Time to market pressures require DFT techniques which can be fully and easily automated. Scan design fulfills that important requirement.

All DFT techniques, especially scan design techniques, require that certain design rules are obeyed. To verify whether a certain circuit is designed according to a desired DFT technique, a tool called DFT rule checker or DFT audit is to be a part of a computer aided engineering (CAE) environment.

Various DFT rule checker approaches and implementations are known [God077], [Hors83], [Bhav83], [Agra84], [Son85] and others. Godoy et al. [God077] presented the first approach to verify the LSSD [EiWi77] rules based on a technique similar to logic simulation. Horstmann [Hors83] presented an concept to verify the same rules using an AI-based approach. A general approach has been presented by Bhavsar [Bhav83] which is suited to cover various DFT techniques as well as to handle hierarchical structures.

Our present work is based on Bhavsar's approach, the underlying ideas have been refined and extended to a concept for hierarchical DFT rule checking considering the different nature of DFT rules.

The utilization of the concepts has resulted in a tool for DFT rule checking, called CERBERUS.

2 A hierarchical DFT rule check approach
To incorporate DFT into a VLSI circuit, design constraints specified as design rules for testability have to be obeyed. The DFT rule check should be invoked in early design stages when the logic functions are realized bottom up beginning with available basic functional components, e.g. standard cells.

Nevertheless the investigation of a circuit in a hierarchical manner is a need by shortening the time spent in the design-check-redesign cycle. The time-consuming flattening of a hierarchical design description is avoided as well as the hierarchical processing can be done in an efficient way by analyzing a circuit bottom-up preserving the results of lower levels and referencing them at higher levels.
2.1 DFT rules

DFT rules have been developed to cover different aspects of testing. They are forming three categories:

- Rules to support the ad hoc approach to testability, e.g. a limited sequential depth of a circuit.
- Rules related to timing and delay problems. An example is to avoid reconvergence of a clock line what is supposed to be hazardous.
- Rules to ensure the correct design of some structured approach to testability, e.g. the control and observation of a scan element via a legal scan path in a scan design circuit.

In a CAE environment only certain types of logic structures can be handled, thus a specific set of DFT rules reflects which structures are supported by the system. Whereas structured approaches to testability are well supported by elaborated rules in a general view of their constraints, the ad hoc methods require a very specific support by adequate design rules.

Thus, pre-defined rules are preferred to ensure the structured testability approaches and consequently the automatic test pattern generation (ATG). The ad hoc methods are best assisted by user-defined rules.

2.2 The circuit model

The network description used in the approach is characterized as a levelized directed graph representation as shown in figure 1.

Within one level a module is seen as a representative of a certain type embedded into a topological structure (external view).

Non-primitive modules are decomposable, their type information consists of the internal topological structure that will be found one level further down the graph representation of figure 1 (internal view).

Primitive modules are not decomposable, a DFT related behavioral model is used as the type information.

The dynamic aspects of using the hierarchical graph representation become obvious by traversing the circuits' nodes.

One module at the top is reserved to carry the type information of the circuit under consideration.

The traversal starts at the sources of signal flow, i.e. the primary inputs. The modules concerned by the traversal are instances of a type, the associated information is used by some needs to propagate versus the next module. If this information is not available, the module may be traversed one level below using its type referred topological structure.

While traversing a module internally, relevant information is extracted recursively and stored as type information which is applied for all further instances of the same type.

2.3 The "Design for Test Calculus"

The "Design for Test Calculus" (DfTC) as presented in [Bhav83] performs DFT rule checking by a one-pass algorithm. It employs a mechanism of transferring DFT relevant properties through the network nodes, called signal set transfer. It examines those properties stored in the signal sets at the inputs and outputs of certain nodes to establish compliance or violation to a rule, i.e. structural information is used to perform DFT rule checking.

The essential requirements for hierarchical DFT rule checking are also given by the approach. The results of examining a (sub-)network at an arbitrary level are stored in DFT descriptor sets (dds). They provide the information needed in a higher level of description.

A detailed application model using the DfTC is described in [Bhav83].

2.4 A general concept for DFT models

The previously described techniques are considered as a basic approach to testability rules checking, a general concept for DFT models is described below.

Properties are associated with the circuit inputs and outputs as well as the terminals of modules and the modules themselves. They are used to respectively distinguish between path and module types.

Clock paths, scan paths, paths for asynchronous control and test access paths are examples of one kind, combinational modules, modules providing either asynchronous
or synchronous control of internal states and scan devices are examples of the other kind.

Transfer rules control the transmission of path properties while traversing the modules. The path properties after transmission are determined by the properties of a module and its terminals together with the actual path properties.

A specific DFT rule has to be transformed into one or more checking conditions. A checking condition is related to a class of modules and evaluates the path properties and the information at the modules’ terminals to check for compliance or violation to a DFT rule.

Transfer rules and checking conditions together specify the behavior of a rule checking algorithm.

For hierarchical processing, information referring to the internal structure/behavior is associated with the terminals of a module. This information is generated by a previous analysis of the modules’ structure. It is a projection of internal path properties, relationships between paths, and similar relations. In each case the information is sufficient for the transfer rules and checking conditions applied at the module.

A specific application model for DFT rule checking is constructed by defining all attributes and the provision of a set of transfer rules and checking conditions. Hierarchical processing is supported by generating descriptions suited to the transfer rules and checking conditions for each type of module.

Using these general concepts, a wide range of structures and appropriate DFT rules can be handled.

2.5 An example for DFT rule checking
Rule: If the output of a latch is connected to the input of another latch through combinational logic, then the latches may not depend on the same clock primary input.

This rule must be transformed to checking conditions in terms of the rule checking algorithm. For the hierarchical relationship of the latches in the example circuit, the checking condition reads like:

\[
\text{if } \text{class}(M2) = \text{"MN"} \\
\text{then for all } Ci(M2) \\
\text{for all } Dj(M2) \in \text{dds}(Ci) \\
\text{if } \text{ss}(Ci) \cap \text{ss}(Dj) \cap \text{cis} \neq \emptyset \\
\text{then violated}
\]

where ‘MN’ indicates a memory node, Ci and Dj respectively refer to clock and non-clock inputs and cis is the set of all declared clock primary inputs of the network under consideration. The clock dependency operator \( \cap \) is used to get the clock signal(-s) which have previously clocked the signals of the signal set (ss).

3 The DFT rule checker CERBERUS
3.1 The dataflow
An overview of the CERBERUS dataflow is shown in figure 3.

![Diagram of CERBERUS dataflow](image)

Fig. 3 The DFT rule checking system CERBERUS
The techniques described in this paper are implemented in the Siemens' hierarchical DFT rule checker CERBERUS, which is available as part of the SITEST 300 CHIP CAE system to support design and test of large ASIC's using libraries of different foundries.

At present, CERBERUS utilizes the approach to handle various serial scan structures by applying a structural network analysis.

The involved rules are predefined for best support of scan structures and to guarantee further processing by the ATG-system SOCRATES [Schu87].

CERBERUS consists of the following components:

- An extractor is used to provide a levelized directed graph representation from a circuit netlist. Attributes associated with network nodes are preserved.
- The preprocessor uses efficient graph algorithms [Aho74], [Even79], [GaJo79] for cycle recognition, to provide acyclic graphs and for topological sort of circuit nodes.
- The rule checker nucleus performs the DFT verification process hierarchically.
- The protocol generator is used to output the rule checker results in a user-friendly representation.

If a circuit is recognized suitable to ATG, additional components are used for ATG preparation:

- A scan-device-modelling component replaces all scan element types by combinational equivalents. Scan elements are considered in their generic sense [Gutf83].
- The circuit flattener provides a flattened combinational circuit description and processes hierarchical test access information for use by ATG.

3.2 A brief System Overview

The Siemens' CAE system SITEST 300 Chip presently consists of a hierarchical graphic schematic entry system, the multi-level logic and concurrent fault simulator SMILE [Gon84], the hierarchical DFT rule checker CERBERUS, the automatic test pattern generation system SOCRATES [Schu87] and other tools. A unified library which consists of logic gates (AND, OR, EXOR etc.) and complex primitives (adder, multiplexer, decoder, register, counter etc.) supports all test-related tools (e.g. SOCRATES and CERBERUS).

One of the major features of this CAE system is the effective use of hierarchy not only by the schematic entry and simulation, but also for DFT rules checking to speed up the whole DFT verification process significantly.

3.3 Experimental results

The DFT rule checker CERBERUS has been used to check ASIC's with different complexity for compliance with a set of 11 DFT rules suited to the verification of scan design circuits.

The circuits of table 1 are characterized by the number of scan elements, the number of gates and the number of modules of the hierarchical circuit model as processed by CERBERUS.

![Table1 CERBERUS: Experimental results](https://example.com/table1.png)

<table>
<thead>
<tr>
<th>circuit</th>
<th># scan elements</th>
<th># gates</th>
<th># modules</th>
<th>CPU-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>1857</td>
<td>25.000</td>
<td>2322</td>
<td>2' 03&quot;</td>
</tr>
<tr>
<td>#2</td>
<td>1615</td>
<td>31.000</td>
<td>6964</td>
<td>7' 35&quot;</td>
</tr>
<tr>
<td>#3</td>
<td>1434</td>
<td>43.000</td>
<td>4327</td>
<td>5' 37&quot;</td>
</tr>
</tbody>
</table>

The measured times include the whole process as shown in figure 3.

4 Conclusions and Future Work

The approach for DFT rule checking is based on a general classification of design rules and utilizes methods for hierarchical DFT verification.

CERBERUS exploits the benefits of hierarchical analysis to handle VLSI sized circuits fast and efficiently.

The tool provides test access information allowing the use of the automatic test pattern generator SOCRATES and to support the conversion of the generated tests into a general interchange tester format.

Further enhancements will apply to the board level and necessary interface structures at the IC level, the boundary scan [MaBe87] is an outstanding example. The approach implies, that the same principles could be used.

The application of only structurally described rules has drawbacks with respect to the topological restrictions. To follow the aim of a general DFT approach, the logical DFT verification methods will be subject of future investigations.

References


