STA: A System Testability Assistant*

Walid Maroufi, Meryem Marzouki†
LIP6 Laboratory
Couloir 55-65, 4 Place Jussieu
75252 Paris Cedex France
Tel. (+33)1 44 27 53 98 - Fax. (+33)1 44 27 62 86
E-mail : Maroufi@asim.lip6.fr, Meryem.Marzouki@lip6.fr

Abstract

This paper presents a Computer Aided Testability tool named STA (System Testability Assistant), aiming at proposing a testability strategy for system as early as possible in the design process.
Given a system description, STA first compiles it in order to create system, boards, and circuits objects, and then proceeds bottom-up, from the circuit to the system, in order evaluate the actual testability of the whole system, recommend a testability strategy and testability features inclusion in some modules, and provides support for including these testability features in the system by calling external generators or synthesis for testability tools.

1 Introduction

As HW/SW systems become more complex, their testability become a severe problem. Consequently Synthesis for Testability (SFT) techniques should provide an efficient answer to system testability requirements and so permit an efficient and flexible further maintenance. Currently, The aim is to introduce testability features as early as possible in the design and synthesis processes of system devices, but the question *which features for each element?* still need an answer.

This paper presents a methodology based on a Computer Aided Testability tool named STA (System Testability Assistant for modeling, evaluation and support for inclusion of testability features in the whole HW/SW system. The specification of the system is done according to functional and non-functional requirements. Specifically concerning testability requirements, once the HW/SW system has been specified, and once the HW modules have been synthesized, it is essential to evaluate testability according to test quality and test cost constraints. This tool helps the user viewing the global testability for his system, and recommends testability features to add whenever needed to some modules.

Section 2 presents the adopted approach to model and specify systems to be processed by STA and the tool which drives it. STA processing methodology details are provided in section 3, section 4 describes STA interaction with its environement, and section 5 gives some conclusions.

2 STA: A computer Aided Testing Tool

2.1 System Modeling and Analysis

The adopted approach to system modeling is based on object-oriented models, different classes being defined, representing hierarchical view of the whole product: the *system* class, the *PCB/MCM* class, and the *circuit* class. Inside each class, object are defined, with description attributes and their associated operating methods. These objects can be testable according to different testability methodologies or non testable. Among the methods associated to each object, we can find testability evaluation and testability features adding tasks, while attributes are mainly testability features or requirements of the element (system, PCB, MCM or circuit).

Let us give the example of the circuit class where each circuit of the system is instantiated as an object. It is first graded into three main sub-classes according to the circuit nature: *Digital Circuits Class*, *Analog Circuits Class*, *Mixed Circuits Class*. The Digital Circuits Class is then graded into four other subsub-classes where two of them are representing the level of the circuit description: *Behavioral Circuits Class* and *RTL Circuits Class*, the third one instantiates existing circuits: *Existing Circuits Class* (this is used when off-the-shelf circuits are used to compose the system) and the last one is a special class which instantiates software modules of HW/SW systems which will be run on standard processors. Going further down in *Digital Circuit Class* we find two subsubsub-classes: the *Testable Circuit Class* instantiating circuits where test functionalities are already available and the optional*PrefTest circuit Class* instantiating circuits for which the user gives suggestions in terms of testability. The modeling approach is detailed in [1].

The specification approach is driven by an analyzer (STA ANALYZER : already developed) [1] which is a tool compiling a text describing a system and written in a specific grammar in order to create objects according to our OO Model properties. The description contains the system structure and the suggestions of the user in terms of testability preferences.

*This work has been partly supported by MEDEA-SMT-A-403 project.(System Level Methods and Tools)
†Corresponding Author

2.2 STA Principles

The methodology is first based on an *interactivity* principle. Since, user contribution is very important assuming that he is the main driver of tool processing. The second principle is *modularity*: to make decision on the global testability of a system the user must be able to reach several tools allowing him synthesizing device descriptions at different levels, generating specific testability features, evaluating overhead and so on. The third principle is *evaluation*: STA methodology considers a system testability evaluation model taking into account test quality and test cost constraints. In this way, STA tool tries to fit in with these three principles. Figure 1 shows how STA embeds these three principles, in order to interact with other testability and synthesis for testability tools.

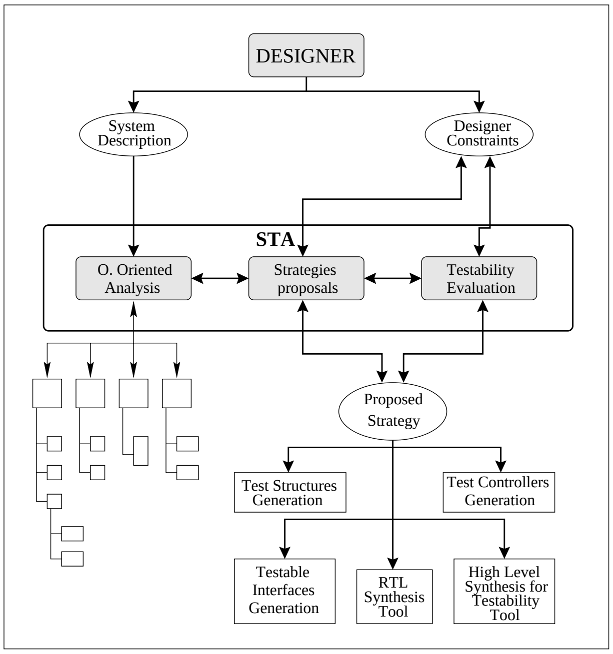

Figure 1: STA functional view

3 STA Methodology

The main program is represented by the scheme of figure 2. STA is a bottom/up process beginning with circuit testability to reach the overall system one. It is based on a global trade-off between system specifications, user suggestions and testability constraints. Before any processing, the system description is compiled by the STA ANALYZER in order to create all the objects of the STA Object Oriented Model (see section 2).
During STA processing, the objects can migrate from one class to another, according to the transformations which occur on them. For example a circuit X migrates from the general class *Circuit Class* to the specific one *Circuit Test Class* whenever test functionalities have been added to its architecture.

3.1 STA Circuit Processing

This block is in charge of circuit test adding features, it encompasses the following tasks:

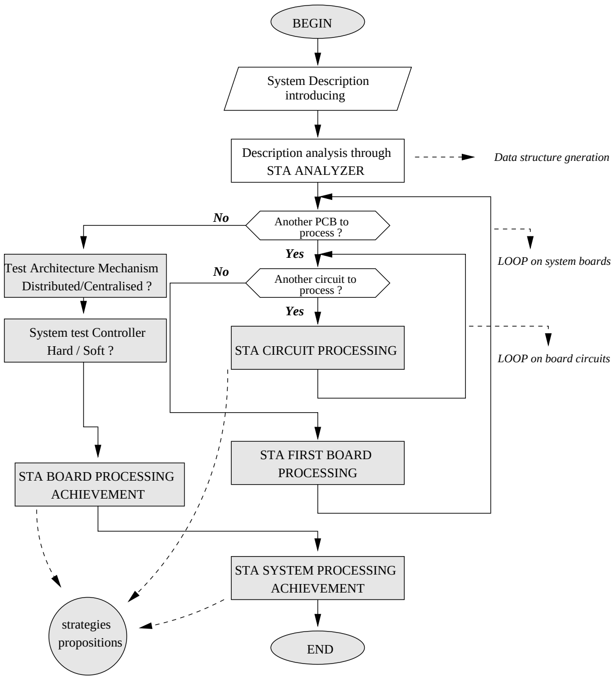

Figure 2: STA Methodology

- First of all, it investigates circuit architecture and evaluates its testability when initial test functionalities exist.
- Then it proposes a test strategy for the circuit, taking into account the initial circuit specifications, user suggestions and conventional test techniques properties. All this will be interactively driven by the user. Synthesis (behavioral, RTL or logic) tools can be called during this step to synthesize the architecture, in order to get more information about the circuit description and so refine the study of test inclusion features.
- Once a proposed test strategy has been accepted, test structure generators are called to introduce test features in the architecture.

It is important to note that during processing, STA will only guide user decisions since it is neither a synthesis tool nor a test structure generator (figure 3). During "Circuit Processing" task, each circuit is represented by a state vector (figure 4). All transformations done in the circuit description are reported on this vector. The bit labeled T (Treatment) informs about the final state of the circuit: T='X' means that the circuit description was unchanged, T='0' means that changements occur on the circuit but no test feature was yet added, T='1' means that the circuit description was modified and test features were added.

3.2 STA Board First Processing

This block makes a first study of the board testability since final study cannot occur before all boards circuits have been treated and a system testability architecture type has been chosen: see section 3.3.
Like with circuits, a board test state vector is created (figure 5). The state vector will be partially initialized during the "Board First Processing". The main tasks of this block are:

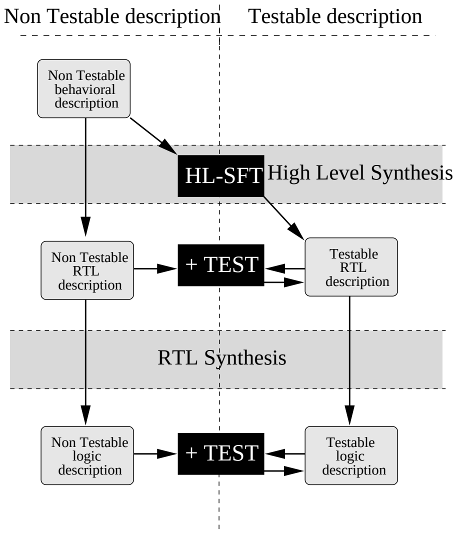

Figure 3: Circuit level testability guiding

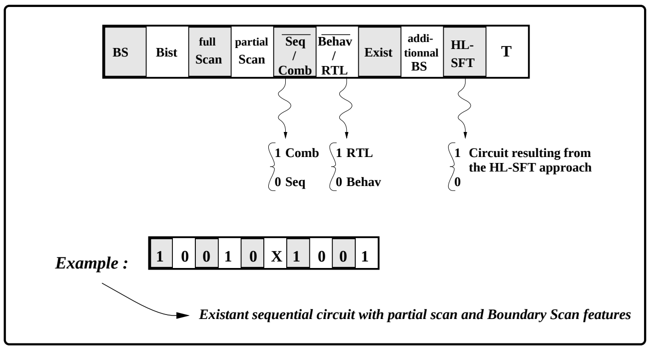

Figure 4: Circuit state vector

- Detecting Boundary Scan instructions used in the board circuits and so updating bits 5 to 9 of the state vector.
- Updating bits 12 and 13 when a board test controller (BSC or BTC, see section 3.5) exist.
- Evaluating or re-evaluating board complexity taking into account circuit modifications.
- Reviewing the position of the PCB object in the system model since circuits modifications can induce its migration from one class to another (for example, a PCB for which BS features were introduced in its circuits architectures must migrate from the class *Pcb Class* to the *Pcb BS Class*).

3.3 System Test Architecture Mechanism Decision Making

This task is in charge of making decision on the system test architecture mechanism. There are two possible mechanisms: the distributed one, where test control is made by intelligent controllers residing on boards, and the centralized one, where one global system controller assumes the task of controlling test on boards, via simple board interfaces.
A trade-off between test speed, cost of hardware controller

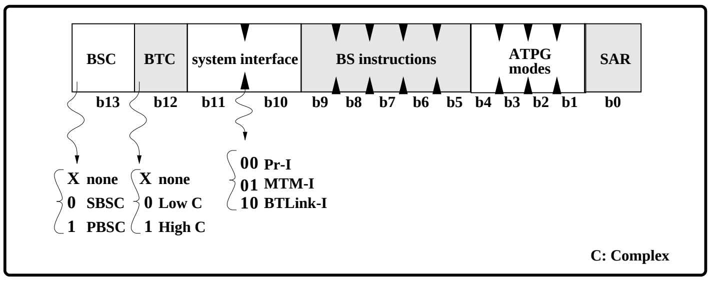

Figure 5: PCB state vector

design and fabrication, and system fault tolerance must be reached, taking into account user suggestions or requirements and PCBs complexity. It should be noted that a distributed mechanism is recommended for test speed enhancement while a centralized one is preferred when test cost represents a severe constraint during system design.
A mixed mechanism (distributed/centralized) can be proposed when the number of PCBs in the system is important and the choice of one of the two previous mechanism has become not realistic.

At the present time we have a core based on a fuzzy logic method which makes this decision. All parameters are brought back to a same scale in order to apply fuzzy logic operations on them. In this framework, fuzzy qualitative values of parameters are identified (e.g: speed is evaluated as high, low, almost high, etc.), and processed using already available tools [2] [3] [4]. On the other hand, each parameter is weighted by the user according to its importance.

3.4 System Test Controller Nature Decision Making

Once the system test architecture mechanism has been chosen, it is essential to make decision on the nature of the system test controller, which can either be a Hardware (HARD) or Software (SOFT) one.
Test speed, cost, fault tolerance requirements and number of PCBs in the system are the main parameters which influence the choice. This is explained in the table of figure 6 where, behind these parameters, the architecture mechanism is taken into account. Again, values of the parameters are expressed using fuzzy qualitative modeling, represented by the '+' and '-' signs of various levels on figure 6.

Let us comment the white column of the table which concerns a centralized test architecture with a HARD central system test controller and a large number of boards. In this case, the test speed will be low, because of the centralization of the architecture. On the other hand, the system test controller cost is very high due to its complexity, assuming that it is in charge of all the boards test control, while boards test controllers (which are mainly simple interfaces) cost is much lower. Finally, the system test fault tolerance is very bad, since when a problem occurs on the central system controller, all the system test process could fail.

		Distributed Architecture				Centralized Architecture			
		Cs HARD		Cs SOFT		Cs HARD		Cs SOFT	
		n large	n small	n large	n small	n large	n small	n large	n small
Rapidity		+ +	+ +	+	+	−	−	− −	− −
COST	Cs	+	+	+ +	+ +	− −	−	+ +	+ +
	Cb	− −	− −	− −	− −	+	+	+	+
	∑Cb + Cs	− −	−	− −	−	+	+	+	+
Fault Tolerence		+	+ +	+	+ +	− −	− −	+	+ +

n : number of PCB in th system / Cs: Cost of the system test controller / Cb: Cost of the board test controller

Figure 6: Parameters table

3.5 STA Board Processing Achievement

At this step of processing all circuits have been treated, a test architecture has been chosen and the System Test Controller nature was defined. Therefore, it is possible to achieve the test strategy study of each board.

3.5.1 Bord controllers generation

During this step of the STA process, the aim is to define and eventually generate the test controller of each board. We define two classes of test controllers:

- Boundary Scan controller (BSC) which controls the test of BS circuits according to the IEEE1149.1 Standard. We distinguish two types of BSC:
  - The PBSC (figure 7): We have already developed a Programmable BSC [5] in which characteristics of BS chain to test can be initialized. The PBSC integrates all BS instructions algorithms, ATPG/SAR modules and a memory to stock BS chain characteristics. However, it has the ability to be interfaced with HARD or SOFT (processor interface) system test controller.
  - The SBSC: Which is a Specific BSC. We are developing a BSC generator which, given a BS chain, generates its specific BSC. SBSCs are very useful when the BS chain control does not require all the PBSC functionalities.
- The BTC (Board Test Controller) which controls testable circuits without BS features (but containing BIST or Scan Path features). In case of circuits resulting from a High-Level Synthesis for Testability approach [6], the BTC will work in interaction with the On-chip Test Controllers which are automatically synthesized with the circuit architecture and which are responsible of the sequencing of different operations during circuits test [5].

Interfaces between board test controllers and the global system test controller are not yet defined at this level of processing

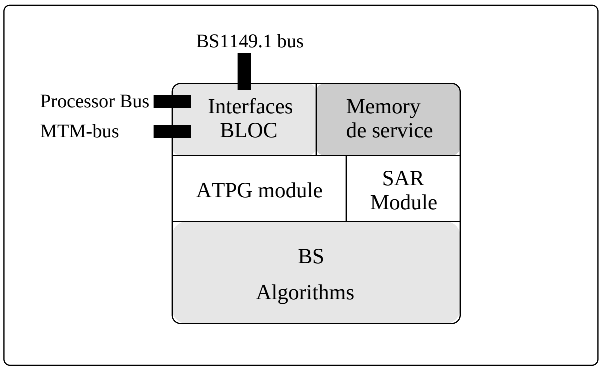

Figure 7: Synoptic view of the PBSC

since the type of the latter is not yet chosen. This will be done in next processing part. However, if system controller is SOFT these interfaces are fixed as processor ones.

3.5.2 Chains construction

One important step in this process is the Boundary Scan chain construction which consists in connecting BS circuits to each other through TDI and TDO pins. For MCMs, two different cases are possible:

- If the MCM does not contain any internal controller, a local BS chain is constructed and connected to the controller.
- If there's no controller on the MCM, we connect the local MCM BS chain to the external board BS chain. We say that we have flatten the MCM (figure 8).

In most cases we flatten MCMs when the test architecture is centralised and we try to generate MCM controllers when it is distributed. The same strategy is followed for scan chains.

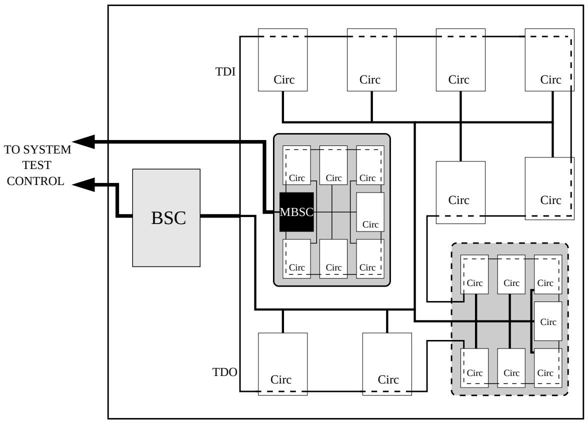

Figure 8: MCM test strategy

3.6 STA System Processing Achievement

The main task at this stage of the STA process is to make decision on the system test controller type (when it was decided

that it will be a HARD element: see section 3.4).
First of all, boards are classified as BS boards (containing BS circuits), non BS boards and heterogeneous boards [7], containing BS as well as non BS circuits. The next processing phase is called pseudo-centralization, and concerns distributed architectures where some boards are partially testable. In this case it is useful to group them and centralize their test control (figure 9). We distinguish two kinds of pseudo-centralization:

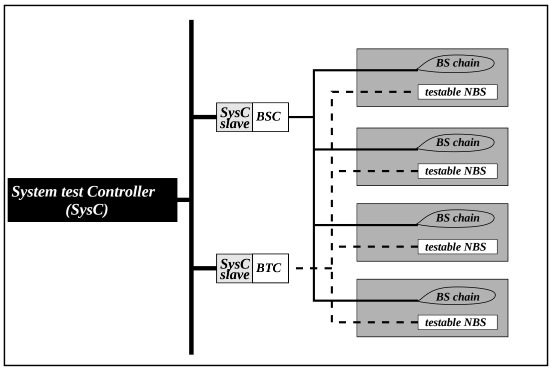

Figure 9: pseudo-centralization of a distributed architecture

the BS one and the NON BS one. In the first case, boards are partially BS and a central BSC is generated to control their test, while in the second case it is a central BTC wis in charge of controlling the test of the boards testable (NON BS) parts. Once pseudo-centralization has been completed, the system test controller definition task can start. Two global strategies are proposed (figure 10):

- System test can be made by the MTM (Module Test and Maintenance) bus protocol specified in the IEEE1149.5 standard [8]. A programmable MTM bus controller is now under developpement. It is based on Master/Slave communication protocol and will be able to perform it self test algorithms according to the standard protocol.
- When the percentage of BS PCBs in the system is very important, another alternative for system test consists in the natural extension of the basic IEEE1149.1 test at the system level. An architecture to make IEEE 1149.1 test access port and test bus protocol available on system backplanes is proposed in [9]. A Test and Maintenance Master (TMM) module controls the 1149.1 bus linked to each board via a Backplane Test Link (BT-Link) module similar to our BSC (PBSC or SBSC) (section 3.5). Now, we are thinking about an architecture for a TMM module compatible with the BSC interface.

According to the previous processing results, and taking into account user suggestions and requirements, the STA will propose a solution for the system level test control, and then the system test strategy study is achieved, and reported to the system designer.

4 STA and its environment

During the main STA processing task, the tool must be interfaced with other several tools like synthesis tools, generator of test structure, evaluation tools and so on. In this section we present some types of such tools.

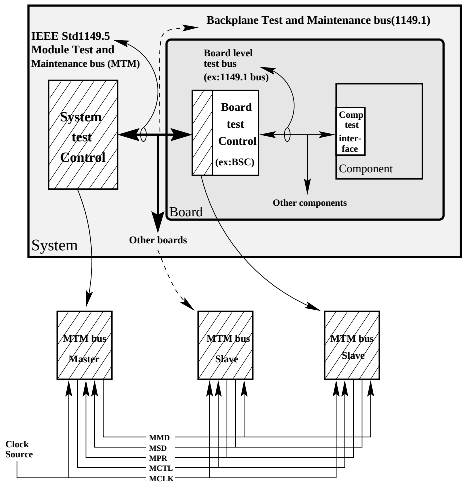

Figure 10: System Level Test Control

In some cases it is necessary to synthesize circuit descriptions to further allow a area or complexity evaluation, or simply to move the description from one level of abstraction to another. Since STA treats behavioral or RTL descriptions, it is useful to interface it with behavioral synthesis tools and RTL synthesis tools.
In addition, we have already developed a tool which introduces Boundary Scan features in RTL descriptions [10]. We are also defining a tool for High level synthesis for testability (HL-SFT) which can be viewed as the automatic generation of testable architectures [6].
Moreover, STA can also be interfaced with tools which generate testable communication interfaces since adding test to such devices must be driven by a particular approach which take into account communication protocols properties.
Furthermore, at different level of processing, evaluation tools are called for evaluating area of circuits, evaluating complexity of circuits, evaluating complexity of boards, evaluating test cost, evaluating test overhead, evaluating the global testability of a system once STA processing finished.
Finally, the user should specify the global system describing PCBs, MCMs, circuits and their organization, so that the STA analyzer can compile this desciption. He also provides the tool with information related to the system like elements complexity, elements criticity, degree of sequentiality in circuits, and so on. These details can influence the choice of the global system test strategy. Defaults values can be used by the system, however. The user can in addition specify for each system component its level of description: behavioral or RTL. In some cases the user can give the specifications of off-the-shelf circuits from a library. This allow the reuse of existing components. Finally, the user can give the system his test method preferences for the system components.

5 Conclusion

This paper has presented a methodology driven by a tool based on three principles: flexibility, modularity and evaluation. This tool called STA aims at proposing the better testability strategy for HW/SW system as soon as possible in the design process. First of all, a STA system description is compiled by the STA ANALYZER in order to create system, PCB/MCM and circuit objects composing the system data structure. Then, given conventional test techniques properties, user preferences and system data structure objects, STA evaluates existing testability features, if any, and tries to find a whole strategy for the system under process testability.

Once the proposed system testability strategy has been accepted by the system designer, new testability features can be introduced in circuits architectures and board/system test controllers are generated to control the system test. This step is achieved by calling different modules, external to the STA.

References

[1] M. Marzouki W. Maroufi. Specification ad early evaluation of system testability through object oriented modelling. In *European Test Workshop*, 1997.

[2] M. Hédi Touati, Firas Mohamed, and Meryem Marzouki. System fault diagnosis based on a fuzzy qualitative approach. In *European Design and Test Conference*. Paris, France, 1996.

[3] Firas Mohamed, M. Hédi Touati, and Meryem Marzouki. Multi-level fuzzy decision-making for dft, test and diagnosis of analog circuits. In *International Mixed Signal Testing Wprkshop*. Quebec, Canada, 1996.

[4] Firas Mohamed and Meryem Marzouki. Test and diagnosis of analog circuits: When fuzziness can lead to accuracy. *Journal of Electronic Testing: Theory and Applications (Special Issues on mixed-signal testing)*, (9):203–216, 1996.

[5] M. Marzouki, V. Castro Alves, A. Rebiero Antunes, and W. Maroufi. Global test scheduling and control in a hl-sft environment. *1st IEEE High-Level Design Validation and Test Workshop, 14-17*, 11 1997.

[6] M. Marzouki, V. Castro Alves, and A. Ribeiro Antunes. High-level synthesis for testability. In *IEEE Midwest Symposium on Circuits and Systems (special session on Testability Issues for Digital Circuits)*, pages 718–721, Rio de Janeiro (RJ) - Brazil, August 1995.

[7] M. Marzouki, M. Lubaszewski, and M. H. Touati. Unifying test and diagnosis of interconnects and logic clusters in partial boundary scan boards. In *Proc. Intl. Conf on Computer-Aided Design*, pages 654–657, November 1993.

[8] IEEE Standard Board. IEEE std 1149.5-1995,"module test and maintenance (mtm) bus protocol", 1995. [1-55937-558-2][SH94333-NYL].

[9] Dilip bhavsar. An architecture for extending the ieee standard 1149.1 test accss port to system backplanes. In *IEEE International Test Conference*, pages 768–776, 1991.

[10] Ana Luisa Lopes Antunes. Synthèse de circuits conformes au standard boundary scan. Master's thesis, Institut National Polytechnique de Grenoble, Grenoble, France, June 1995.