System Testability Evaluation with STA

Walid Maroufi, Meryem Marzouki*
LIP6 Laboratory, Couloir 55-65, 4 Place Jussieu
75252 Paris Cedex France Tel. (+33)1 44 27 74 74 - Fax. (+33)1 44 27 72 80
E-mail : Walid.Maroufi@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 systems 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 to 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.

It should be noted that STA deals with the testability of system equipment made up of different elements. It doesn't address the testability of Hardware/Software systems described at the behavioral level (such approaches can be found in [1] [2]). 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 testability evaluation through STA, an example of a board testability evaluation is presented, and section 5 gives some conclusions.

*Corresponding Author

2 STA: A computer Aided Testing Tool

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).

The specification approach is driven by an analyzer (STA ANALYZER [3]) 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.

The methodology is first based on the *interactivity* principle, and 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 use several tools allowing him synthesizing device descriptions at different levels, generating specific testability features, evaluating overhead, etc. The third principle is *evaluation*: STA methodology considers a system testability evaluation model taking into account test quality and test cost constraints. The fourth principle is *test reuse*: testability features are generically used (e.g. test controllers). Figure 1 shows how STA tool embeds these four principles, in interaction with other testability and synthesis for testability tools.

3 STA Methodology

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). Then STA main processing steps are:

- **STA Circuit Processing:** This module is in charge of adding test features to the circuits. At this step STA only guides user decisions since it is neither a synthesis tool nor a test structure generator.
- **STA Board First Processing:** This module makes a first study of the board testability, considering all the ICs

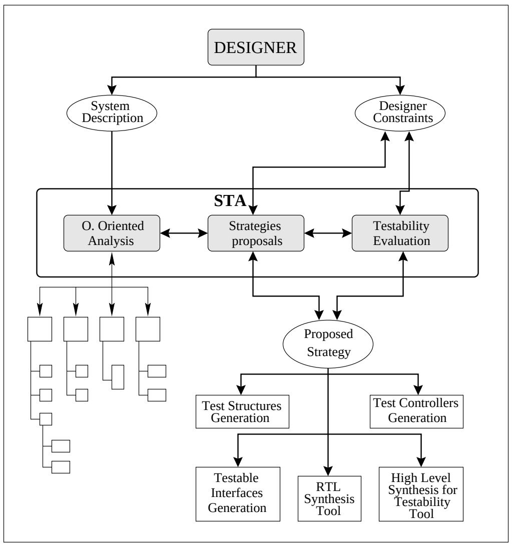

Figure 1: STA functional view

populating it, in order to prepare the system test architecture mechanism choice. It is also in charge of (re-)evaluating the board complexity.

- **System Test Architecture Mechanism Decision Making:** This task is in charge of making decision on the system test architecture mechanism which can be distributed, centralized or mixed(distributed/centralized). A trade-off between test speed, cost of hardware controller design and fabrication, and system fault tolerance must be reached, taking into account user suggestions or requirements and PCBs complexity. Each parameter is weighted by the user according to its importance, and the decision-making process is based on a fuzzy-logic method. 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 [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 or Software one. Test speed, cost, fault tolerance requirements and number of PCBs in the system are the main parameters which influence the choice.
- **STA Board Processing Achievement:** At this step, circuits have been treated, a test architecture mechanism has been chosen and the System Test Controller nature is defined. Therefore, it is possible to achieve the test strategy study of each board. This encompasses two main tasks: the board controllers generation and the chains construction. Chain construction task consists in connecting BS circuits to each other through TDI and TDO pins. If the IC is an MCM, then if it contains an internal BS controller, it will be directly connected to system controller. Bord test controllers are of two types:
  - 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 which is programmable (already developed [5]) and the SBSC which is specific for a given BS chain (we are developing a BSC generator which, given a BS chain, generates its specific BSC).
  - The BTC (Board Test Controller) which controls testable circuits without BS features (BIST or SCAN).

  Chains construction task consists in connecting BS circuits to each other through TDI and TDO pins. For MCMs, if the MCM does not contain any internal controller, a local BS chain is constructed and connected to the board controller. In the other case, 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 flattened the MCM. 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.
- **STA System Processing Achievement:** The main task at this stage of STA process is to make decision on the system test controller type, when it was decided that it will be a hardware element. Two global strategies can be proposed:
  - System test can be made by the MTM (Module Test and Maintenance) bus protocol specified in the IEEE1149.5 standard [6]. A programmable MTM bus controller is now under developpement. It is based on Master/Slave communication protocol and will be able to perform its 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 [7]. 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). An architecture for a TMM module compatible with the BSC interface is under study.

  According to the previous processing results, and taking into account user suggestions and requirements, 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. More details on the overall methodology can be found in [8].

4 Evaluation

This process can occur at different stages of the STA global process. In three cases it is processed automaticaly without user confirmation. The first time, it occurs after the STA analyzer step in order to evaluate the initial testability of the system boards (or MCMs) before any transformation. Then it is processed for each board (or MCM) after treating its circuits and this in order to re-evaluate its testability when new test features are added to the circuits. Finally, an automatic evaluation step is processed at the end of the STA process to re-evaluate all the boards (or MCM) testability and this after board test controllers (BSCs or BTCs) incorporations. Apart from these three automatic evaluation steps, the user can, at any time, ask for an evaluation process.

In [9] a methodology of testability analysis of digital boards is proposed and is based on a qualitative approach. In our case, different STA processes necessitate a quantitative evaluation of testability. So we propose a process based on controlability and observability(and so testability) calculation at the board level in order to detect hard to test circuits interconnections nodes, classify these nodes through their testability values and so classify circuits through their inputs/outputs testability average. This classification will be useful to guide circuit testability features introduction. In fact, by introducing Boundary Scan features within critical circuits we can enhance the whole board testability. Similar quantitative approaches like [10] are available, but they only consider the circuit level. It is important to notice that

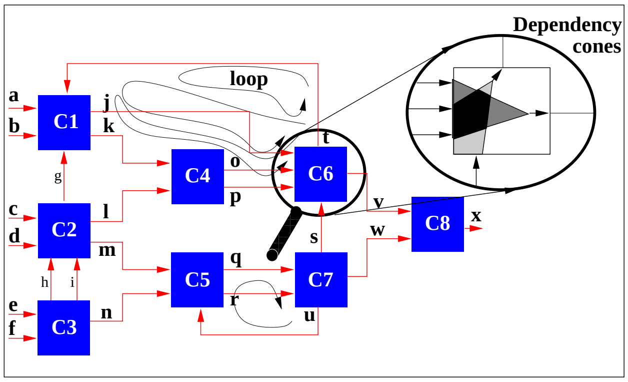

Figure 2: Board Testability Evaluation

we consider controlability and observability propagation through circuits as available information, since STA has been designed for systems. The same assumption is made for dependency cones, which represent the relation between circuits inputs and outputs (see figure 2).
The approach is first based on a loop detection process taking into account the board netlist and the dependency cones information. Then these loops are studied to decide where they can be opened in order to propagate controlability, then observability values, through the board nodes. An efficient algorithm recalculates nodes testability taking into account loops effect. At the present time all the evaluation algorithms (loops detection, loops virtual opening, controlability and observability propagations, etc...) are completely automated.

4.1 Controlability and Observability calculation

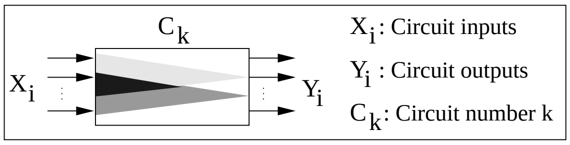

Figure 3: Dependency cones within circuits

The **controlability** of a circuit output is calculated taking into account dependency cones (DPC)(figure 3) to which it belongs. We notice that each DPC is defined with a controlability factor $DPCF$ (such as the DPC itself, its factor is considered as available information). The controlability of the output is then calculated through the formulas:

$$(a)\ C_{Y_J} = DPCF_l.C_{IN} \quad \& \quad (b)\ C_{IN} = \prod_{p \leq I \leq q} C_{X_I}$$

Where l represents the DPC to which the output belongs and X_p ... X_q are its inputs.

The **observability** of a circuit input is calculated taking into account all the DPCs where it is indicated as an input and its own observability factor OBF which is considered as an available information (since it is an internal circuit information). In [10], we can find a method used to calculate the DPCF and OBF factors within a circuit. The observability calculation formula is:

$$(c)\ O_{X_I} = OBF_{X_I}\ .\ \frac{\sum_{p \leq J \leq q} O_{Y_J}}{q-p+1}$$

Where Y_p ... Y_q represent the outputs of DPCs where X_I is indicated as an input.
Assuming that both boards inputs controlabilities and boards outputs observabilities are equal to one, these values are propagated through boards nodes using the formulas presented above. From these formulas, it seems clear that controlability propagation is done from board inputs to board outputs and that observability propagation is done conversely from board outputs to board inputs.

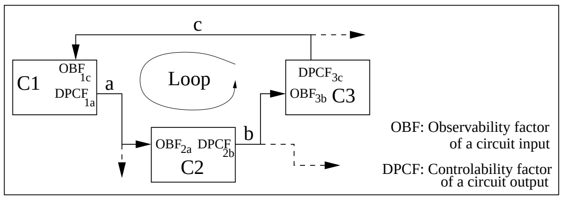

Figure 4: Loop example

In case of **loops** (figure 4), observability and controlability loop factors are calculated through the two formulas:

$$(d)\ CF_L = \prod_{I}^{J(I)} DPCF_{IJ} \quad \& \quad (e)\ OF_L = \prod_{I}^{K(I)} OBF_{IK}$$

where I represents circuits implicated in the loop L and J(I) and K(i) are respectively the output and the input of the circuit I from which the loop passes. The loop is virtually opened in order to propagate controlability and observability values (Figure 5 (a)) (resp. (b)) shows how we can do that before controlability (resp. observability) propagations. Then, the real controlability and observability values are calculated through the two formulas:

$$(f)\ C_{Y_J} = C^*_{Y_J}.CF_L \quad \& \quad (g)\ O_{X_I} = O^*_{X_I}.OF_L$$

where $C^*_{Y_J}$ and $O^*_{X_I}$ are the controlability and observability values calculated with the (a) (b) and (c) formulas.
Finally we define the **testability** of a board node which is the average of its controlability and observability values.

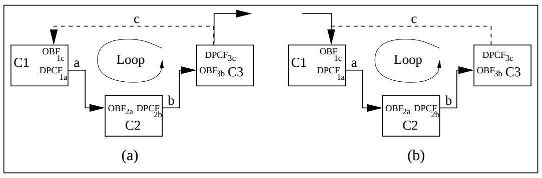

Figure 5: Loops virtual opening

4.2 Example

As an example, we have treated the board represented in the Figure 2. Since it is not a real board with real circuits, we had to impose the DPCF and OBF values of all the boards circuits inputs and outputs. We assumed that they all have the value 0.8 (Figure 6 shows the different DPC within board circuits). In the other hand, controlability of board primary inputs and observability of board primary outputs are all equal to 1 since the formers are completely controlable and the latters completely observable. Processing results are shown on the table of figure 7.

Circuit	Inputs	Outputs	DPCF
C_1	a, b, t a, b, t, g	j k	0.8
C_2	c, d, h, i	g, l, m	0.8
C_3	e, f	h, i , n	0.8
C_4	k, l	o, p	0.8
C_5	m, n, u m, n	q r	0.8
C_6	j, o, p o, p, s	t v	0.8
C_7	q, r	s, w, u	0.8
C_8	v, w	x	0.8

Figure 6: Dependency cones of the example circuits

Nodes	Controlability	Observability	Testability	Nodes classification
(a)	1	0.14	0.57	18
(b)	1 primary inputs	0.14	0.57	19
(c)	1	0.179	0.589	21
(d)	1	0.179	0.589	22
(e)	1	0.212	0.606	23
(f)	1	0.179	0.589	24
(g)	0.64	0.128	0.384	10
(h)	0.8	0.118	0.459	13
(i)	0.8	0.3	0.55	17
(j)	0.26	0.21	0.235	7
(k)	0.21	0.16	0.185	4
(l)	0.21	0.168	0.189	5
(m)	0.64	0.377	0.508	16
(n)	0.8	0.377	0.588	20
(o)	0.14	0.21	0.175	2
(p)	0.14	0.21	0.175	3
(q)	0.26	0.368	0.314	8
(r)	0.41	0.576	0.49	14
(s)	0.085	0.64	0.362	9
(t)	0.04	0.14	0.09	1
(u)	0.11	0.294	0.202	6
(v)	0.0013	0.8	0.4	11
(w)	0.085	0.8	0.442	12
(x)	0	1 primary output	0.5	15

Figure 7: Evaluation processing results

The node classification is done from the worst testable node to the best one. In this example, node (t) has the worst testability value. This can be explained by the fact that it is implicated in three different loops which damage its controlability and observability values. From this first classification we can make another one concerning circuits (by calculating for each circuit its testability which is the average of its inputs and outputs testability) and we can also define a board testability average (BTA) which is the average of all the board nodes testabilities (in this example BTA=0.406). In the circuit classification, we notice that C_4 and C_6 are the worst classified ones since they are implicated in three loops. This circuit classification is very useful, since, suppose that we decide to add Boundary Scan features to one of these two circuits, its inputs and outputs will be completely controlable and observable, this will enhance all the board nodes testabilities and so the BTA. So we don't need to add BS features to all the board circuits, we just do it for the worst classified ones.

5 Conclusion

This paper has presented a methodology driven by a tool based on four principles: flexibility, modularity, evaluation and test reuse. Based on this methodology, a System Testability Assistant proposes the best testability strategy for systems as early as possible in the design process. First of all, the 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.

An evaluation process for system boards is provided within STA. It can be processed automatically before some STA processing steps or it can be called by the user when he wants to evaluate the testability of a given board. An example of board testability evaluation is presented. It shows how can be useful to evaluate board nodes testability before or during STA processing. Since this step of evaluation permits to focus on the worst testable board nodes (and so circuits) and so guides the designer in testability features insertion where it is necessary in order to enhance the testability.

References

[1] Le Traon Y and Robach Chantal. From hardware to software testability. In *IEEE International Test Conference*, 710-719, October 1995.

[2] H. P. E. Vranken, M. P. J. Stevens, M. T. M. Segers, and J. H. M. M. Van Rhee. System-level testability of hardware/software systems. In *IEEE International Test Conference*, pages 134–142, Washington D.C. - USA, October 1994.

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

[4] 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, March 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*, 1997.

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

[7] 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.

[8] W. Maroufi M. Marzouki. STA: A system testability assistant. In *2nd IEEE European Test Workshop*. Sitges, Spain, May 1998.

[9] Ludovic Le Lan Marc Perbost and Christian Landrault. Testability analysis of digital boards and mcms. In *1st IEEE European Test Workshop*, Cagliary, Italy, 1997.

[10] Xinli Gu. *RT Level Testability Improvement by Testability Analysis and Transformations*. Department of Computer and Information Science Linkoping University, S-581 83 Linkoping, Sewden, 1996. Dissertation No, 414.