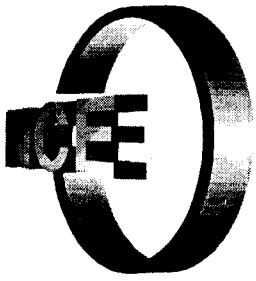

Relatório Técnico

Núcleo de Computação Eletrônica

A Tool for Test Automation with Support for Remote Tests

Magalhães, G. R.
Oliveira, C. E. T.
Anido, M. L.

NCE - 19/99

Universidade Federal do Rio de Janeiro

A Tool for Test Automation with Support for Remote Tests

Magalhães, G. R., Oliveira, C.E.T. and Anido, M.L
Núcleo de Computação Eletrônica - Universidade Federal do Rio de Janeiro - Brazil
e-mail: mlois@nce.ufrj.br

Abstract

This paper presents an interactive MS [illegible] we provide an object-oriented tool for Boundary Scan and LSSD test automation with support for remote tests, including interfaces to circuit description, chip interconnection, test vector analysis and test vector generation. The system is a reasonably complete CAE test system that includes features like remote testing (using client-server technology), project management, computer-aided learning support, menu-based command entry, user-defined configuration and a comprehensive set of commands. Thus, by using this system, a test engineer or a circuit designer are able to specify and verify tests for printed circuit boards or for integrated circuits, either locally or remotely.

1. Introduction

One of the major problems faced by electronic board designers, manufacturing engineers and field engineers is to have automatic, cheap and [illegible] IEEE 1149.1 standard and other [illegible] required circuits using scan design, like the LSSD testing methodology. As novel contributions

ithout a *design for testability* philosophy mployed at design time, the designer has to build external hardware for each different chip to be sted and drive its I/O pins using many different mbinations of the inputs to have a small degree certainty about circuit operation. However, the ajor problems are : how can it be guaranteed at all possible faults (in the fault model) can be tected ? How to find out where the problem is? nfortunately, the problem becomes much worse nen the number of pins increases. Among the sign for testability (DFT) methodologies mployed by IC designers two of the most ficient are *scan design for external test* and *ilt-in self test* (BIST).

esting complete boards is also a complex oblem because boards are composed of several terconnected chips using printed circuit metal cks which can either be short circuited or ened. The IEEE 1149.1 standard [3,4] specifies e methodology to test the entire board (or bmodule) using the boundary scan ethodology, that is, by controlling the board rough its input and output pins.

ne IEEE 1149.1 standard describes the ethodology to implement boundary scan. While propriated to test PCBs and ICs by mmanding and observing I/O pins, this test ethodology is not appropriate to test the inner odules of integrated circuits because it essuposes the use of edge triggered flip flops

The increasing complexity of printed circuit boards and of integrated circuits has demanded sophisticated CAE test tools, of which some have been developed recently[1]. However, to our knowledge, none of these tools support the IEEE 1149.1 boundary scan standard and at the same time the LSSD testing methodology, nor support remote test. In a previous paper [02] we presented and discussed the feasibility of developing a CAE framework to simultaneously support boundary scan and LSSD testing, concentrating the discussion on the hardware and low-level software which was necessary to support such system. Differently, this paper concentrates on the characteristics of the CAE framework and other high-level features of the system.

(which consume much more chip area than normal latches) and usually employ a single clock line. Although the standard allows using the INTEST instruction to perform boundary scan test on chips, no instruction is defined which realizes an access to an internal scan path of an integrated circuit. For scan testable circuits we support an 1149.1 public instruction termed SCANTEST, such as described in [05]. It is worth noting that using scan design with boundary scan to test complex ICs can lead to very long test sequences, but this combination can be useful for testing small and medium sized ASICs. On the other hand, the use of self-test techniques in conjunction with the RUNBIST[3] instruction to test complex VLSI chips on printed circuit boards(e.g. for maintenance tests) is recommended.

The standard methodology employed to test the inner part of integrated circuits is Level Sensitive Scan Design (LSSD)[6,7,8] which requires, among other things, level sensitive flip flops (latches) and nonoverlapping clocking schemes.

2. The IEEE 1149.1 Standard

The IEEE 1149.1 standard[3,4] provides a capability for board-level connectivity tests without the need to propagate test vectors through the core of an IC and backdrive the outputs of an IC. This is achieved by adding test circuitry and test pins to each IC, which enable all input and output pins to be connected together in a scan chain. The scan chain is a shift register with a parallel load facility which can be used to control and read the signal states on all IC pins. In addition, the IEEE 1149.1 methodology allows the tester to interrogate an IC buried in the middle of the PCB, to run diagnostic checks, to identify the IC, or to sample the signal states of its pins during normal operation. Not all of these features are available on all IEEE 1149.1 compliant ICs. Some features are mandatory and some are optional as is illustrated in the table below.

Table 1 - Some IEEE 1149.1 Features and Instructions

Features	*Instruction*	*Status*
Test Inter-IC connect. on PCB	EXTEST	Mandatory
Bypass the test logic of an IC	BYPASS	Mandatory
Sample signal states on pins	SAMPLE	Mandatory
Test de functionality of an IC	INTEST	Optional
Activate self-test capabilities	RUBINST	Optional
LSSD test of an IC (**)	SCANTEST	Expansion

(**) Special Instruction for LSSD methodology such as described in [05]

3. Overview of the Test System

To generate test vectors, the user can either develop his own test vectors by using some sort of tool such as an editor, or use a Boundary-Scan Test Pattern Generator (BTPG) tool. There are also other well-known proprietary languages like STL[9] or HILO[10] for specifying test vectors. By using a BTPG tool, test patterns can be automatically generated for all Boundary-Scan testable interconnections (nets) of a PCB. Design data is imported in the form of files in BSDL or EBST format[01], and netlist information in EDIF V. 2.0 format or BNET format [01]. Tools to generate test vectors for memories can also be used. This is illustrated by figure 1.

Test vectors can then be used in conjuction with the TEDTEST software and a Boundary-Scan controller to provide a complete Boundary-Scan test system. The sophistication of the Boundary-Scan controller hardware determines the speed and power of the system. At the moment, only a very simple hardware controller is employed, because it is a development system and not a test production system.

During the test preparation phase, Test Data Files (TDF) are generated. These are submitted to the hardware under test via the Boundary-Scan Controller, which sends and receives vectors. The third phase is the test result analysis phase, that is, vectors received and stored in a test result file are then analyzed. Several types of analyses can be performed on the test result file, such as

Boundary-Scan diagnostics, truth table reporting and statistical process control. The Boundary-Scan diagnostics software package is a specialized tool and it is not under development yet. By using such tool it will be possible to provide the location of the fault in the user's terms for netlist, IC namen and pin number. For the time being, only truth table reporting is available.

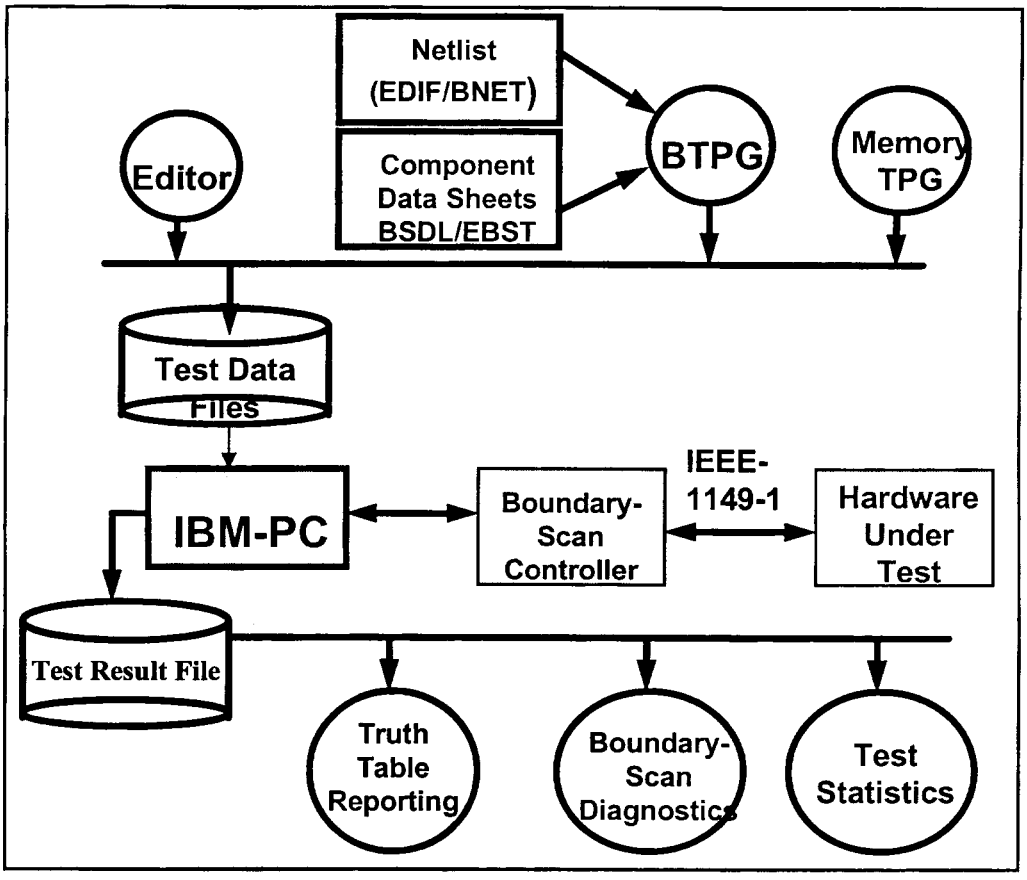

Figure 1 - Overview of the test system

4 -TEDTEST System Interface Characteristics

The TEDTEST System has been developed using Borland's Delphi Language. Delphi is a visual programming language for Windows and has many support facilities for the construction of interfaces. Aiming at compatibility with the JTAG system [01] and also to validate the software, the same hardware controller (PM2705) was used. However, there is no restriction about the hardware controller. Other controllers using the parallel port of IBM-PC like computers can be employed because all the low level procedures that deal with the hardware were written. In fact, this was mandatory for two reasons: firstly the low level procedures are not provided with the JTAG system and, secondly, it was necessary to understand the low level procedures to be able to develop proper procedures for the Scan Path approach.

To exemplify system operation, we present a sequence of tests of a demo board. This process encompasses several steps, from project creation to infrastructure tests. Interconnection tests were not included yet because they are under development.

Figure 2 illutrates a typical operation window. As it can be seen, several menu options have alternatives in the form of quick buttons, which accelerates the operation of experienced users. Moreover, other options, such as background color, can be customized by the user.

Naturally, the first step to be performed when beginning a new project is to create it in the system, which is accomplished by the usual *Project/New* menu option, or by clicking the corresponding speed button (which usually has hints). The project will be created in the user indicated directory. The printed circuit board or the integrated circuit will be accompanied by a file containing a circuit netlist and its board datasheet, or the files used in the two-level tests, which are for infrastructure and interconnection.

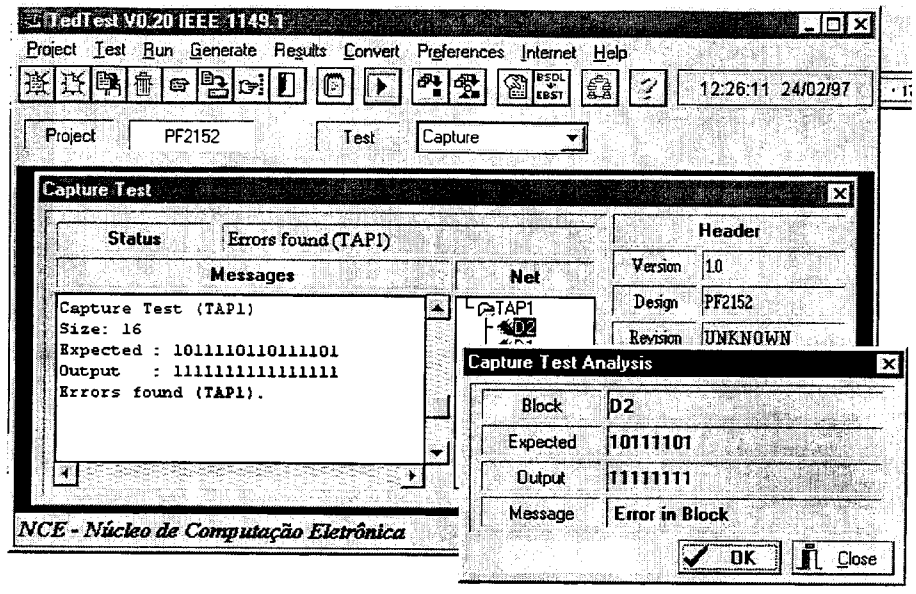

Figure 2 - A Capture Test Analysis Screen

After the user has selected this option, it will suffice specify the drive/directory of the source and destination file. Once doing this, all the files will be copied to their respective project directories. An alternative to this is to edit these files, which can be done by using a Text Editor.

The user can select the type of test philosophy to be used, that is, Boundary Scan or Scan Path, by choosing the proper option in the main menu. The default option is Boundary Scan and for each project included in the system, three sub-directories are created, that are: BS, SP and DS (datasheets). All configuration options are saved in the TEDTEST.INI file, located in the Windows sub-directory.

Once the project has been created, every time that the user wants to load it in main memory, it should choose the the *Project/Option* of the main menu, or choose a Speed Button for this task. The system also offers the possibility of copying all the files related to a project to a new project. This is highly desirable in the case of similar projects, or in the case of identical projects but with different interconnection tests, in which the tests would not need to be totally rewritten.

To accomplish this task it suffices to select the option *Project/Copy* of the main menu and in the dialog box select the project sources among the existing ones and the name of the destination project. The destination directories will be created according to the directories customized by the user. If the user wants to remove a specific project from the system, it suffices to select the option Project/Delete from the main menu. This operation deletes all the data related to a specific project and its sub-directories. These operations illustrate one important feature of the framework to support projects.

Figure 3 illustrates a *Capture Test* which is used to identify any problem in the TDI-TDO link and in the TMS and TCK signals. To perform it, the user needs the General Data File (BTSL-GEN) of the project under analysis. If the user only has the datasheet files (EBST), that usually come with the chips that support Boundary Scan, and the corresponding netlist, it will be necessary to create the General Data File, by using the procedures that generate infrastructure and generate interconnection. Once the BTSL-GEN file has been created, it suffices to select the proper test via a ComboBox, termed *Test* or via *Menu/Test* and select the option *Run/Test* of the main menu. Figure 2 also illustrates this operation.

In the left window the user can see the messages, while the result file is interpreted. The system builds the board structure or system to be tested and organizes it in the ListBox *Net*. By clicking twice over the name of a certain IC, the system will present both the read value and the expected value.

For the same circuit tested above, figure 4 illustrates an *Identification* test. This test is used to identify the IC in the PCB, to determine if it is the one that slhould have been mounted in the PCB and if it was

properly mounted. This test always has to be accomplished after the *Capture* test.

The third and last *Infrastructure* test is the TRST (reset) test. It is used to check if the TRST pin, that is optional in the IEEE 1149.1 protocol, is working properly and it is performed similarly to the previous tests. The only difference is in the selection of the proper test, either via a ComboBox test or via the *Test* option in the main menu.

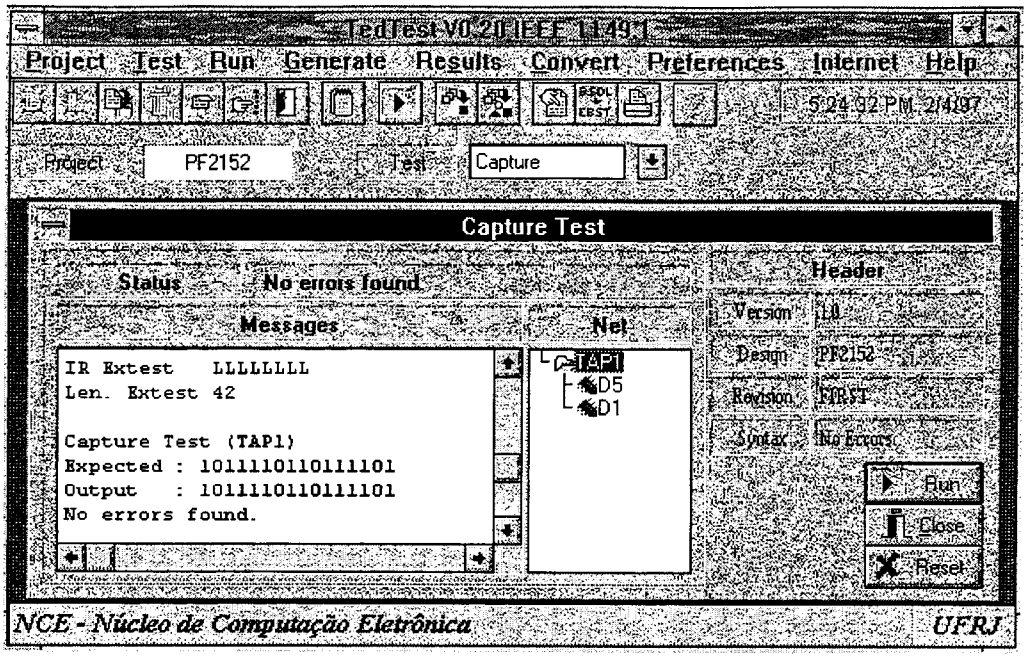

Figure 3 - Capture Test Screen

The option shown below in figure 3 is used to configurate the sub-directory where will be stored the files associated to Boundary Scan tests, Scan Path tests and the Datasheets for a certain system.

The TEDTEST system incorporates some characteristics of a broad concept termed *framework*. Besides the project management operations already described, another system characteristic is that the user can configurate several system options, which will be saved and retrieved when the system is restarted. Other characteristics such as the ability to specify the path to useful tools, such as a text editor, and the screen background color or the parallel port to be used are also available.

5- TEDTEST Structure

A. Structuring the System

The system is totally object-oriented and by so being it presents all the advantages of using this programming paradigm, such as inheritance, polimorfism, encapsulation, etc. It is also event-driven, once it was designed to run in the Windows environment, which is based on this methodology.

Basically, each system Form was associated to an object that represents it. For example, the Form that represents the *Capture* test incorporates methods that are directly related with the test and also includes methods and variables related with the Form itself.

In the case of a set of non-visual methods, the alternative was the creation of specific classes that execute the related functions. For example, a class termed *PointCharUtil* that has a set of methods that deal with character pointers. Every object that needs this class has an instance of *PointCharUtil* created.

The software interface between the TEDTEST system and the hardware controller that generates IEEE 1149.1 compatible signals, installed on the parallel port, was best defined as a Dynamic-Link-Library (DLL). The rationale behind this choice was allowing possible future changes on the interface, such as working with other types of controllers. By so doing, future changes on the interface would not affect the TEDTEST system. The details of this DLL are described in the next section.

B. Interface Between the TEDTEST System and the Hardware Controller

As stated above, the software interface between the TEDTEST system and the hardware controller attached to the parallel port is performed by a DLL, termed Scan Function Library (SFL). This DLL is specific for a certain type of controller. However, it can be easily customized, allowing the TEDTEST system recognize other controllers, sufficing to modify the source code of the SFL functions.

This DLL can also be used by programmers wishing to create their own graphics interface in systems targeted to test PCBs or integrated circuits, sufficing including them in their applications. Internally, the Scan Function Library implements the state machine specified by the IEEE 1149.1 standard, which is shown below.

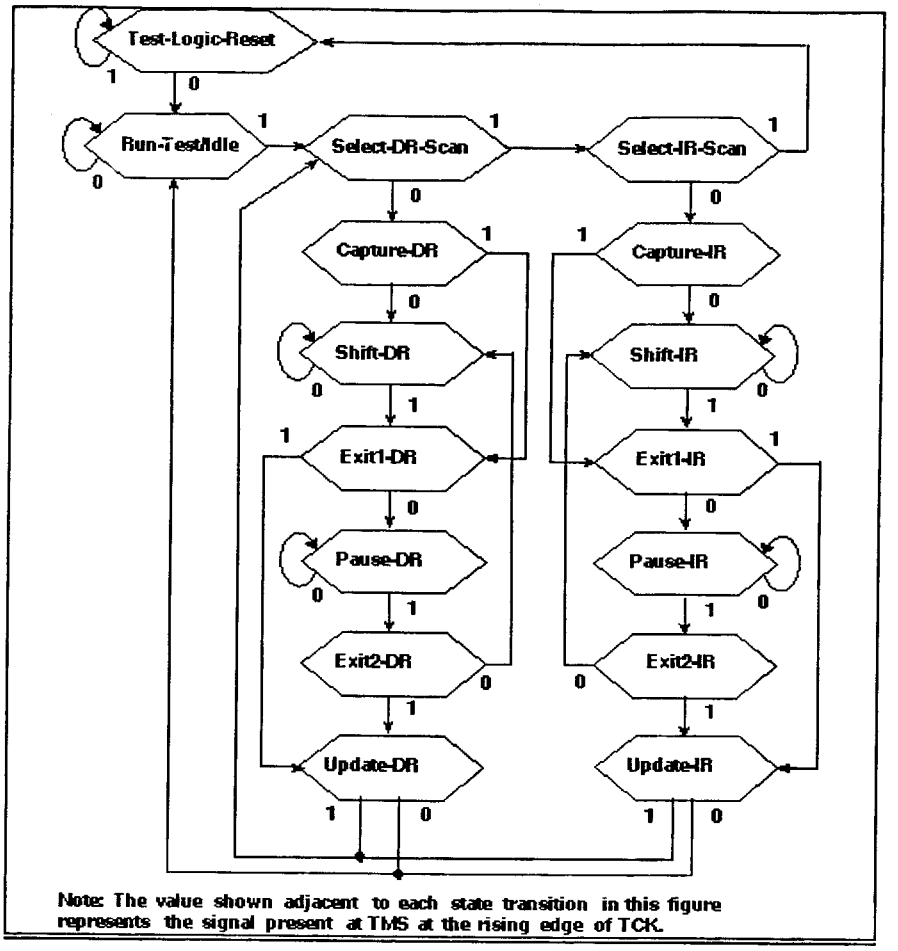

Figure 4 - State Diagram of IEEE 1149.1 Test Access Port

The SFL implements all the mandatory functions of the Boundary Scan standard. The scope and a brief explanation (in Delphi format) of some of the available functions is given below.

Procedure SetParallelPort(NPort : Ports); export;

This procedure is used to configurate the parallel port to be used by the system. The type ports has two values: LPT1 and LPT2.

procedure SetTimer(T : Word); export;

This procedure is used to configurate the time between the system generated clocks. The value is in milliseconds. In a more sophisticated

procedure TRSTReset(Tap : Word); export;

This procedure resets the specified Tap, activating the TRST signal.

procedure ScanIR(Tap : Word; var IRDataOut; IRScanLength : Word; var IRDataIn); export;

This procedure shifts **IRScanLength** bytes into the instruction registers of the TAP. Data to be shifted are located in **IRDataOut**. Simultaneously, data shifted out from the PCB are read by the system in **IRDataIn.**

procedure MultiScanIR(Tap : Word; var IRDataOut; IRScanLength : Word; IRNbrInstr : Word); export;

This procedure executes the function **ScanIR** described previously by **IRNbrInstr** times.

procedure ScanToPauseIR(Tap : Word; var IRDataOut; IRScanLength : Word; var IRDataIn); export;

This procedure puts the state machine in the **PauseIR** state, independently of the present state.

procedure ScanDR(Tap : Word; var DRDataOut; DRScanLength : Word; var DRDataIn); export;

This procedure shifts **DRScanLength** bytes into the instruction registers of the TAP. Data to be shifted are located in **DRDataOut**. Simultaneously, data shifted out from the PCB are read by the system in **DRDataIn.**

procedure ScanToPauseDR(Tap : Word; var DRDataOut; DRScanLength : Word; var DRDataIn); export;

This procedure puts the state machine in the state **PauseDR**, independently of the present state.

procedure ScanSetStopState(StableState : States); export;

This procedure defines a new stop state for all **Scan** commands (ScanDR, MultiScanDR, etc).

These procedures, and some additional ones not listed here, comprise the Scan Function Library (SFL) and are packed into a DLL. They provide the basic software interface between the TEDTEST system and the teste support hardware.

6 - Building HDL and Test Language Compilers with Yacc and Lex

A circuit can be described using different levels of detail and in different forms. The description can concentrate on the logic operations among signals and also on the interconnection of integrated circuits. Additionally, there are languages for specifying the application of tests, such as BSDL, BTSL, etc. Due to the difficulty to quickly develop several compilers to cope with the variety of languages that are necessary, the use of tools that could automate this task was mandatory. To be able to quickly build Hardware Description Language (HDL) or Test Language Compilers, it was necessary to make use of compiler writing tools such as Yacc and Lex.

The TEDTEST system will have compilers for several test languages, such as BNET for Netlist, EBST for Datasheets, BTSL to describe the Boundary Scan infrastructure (.GEN), the test patterns (.APL) and the test results (.ERR). Up to now, only a BTSL compiler has been developed.

At the programming level, the most practical method to create the required compilers was by using the well known Yacc and Lex tools. Versions of these tools that generated Borland's Pascal code were employed. This made the portability to Delphi easier. Lex is employed to create all the required Tokens and interpretate the regular expressions of a certain language. It generates a code composed by tokens and expressions that will be used by Yacc. Yacc is used to generate the rules that govern a certain language. These rules are constructed based on the tokens and on the regular expressions defined in Lex.

7. Remote Tests Supported by a Client-Server Philosophy

The capability to support remote tests is highly desirable because it opens many new possibilities for field test engineers, remote training, etc. Additionally, client-server technology and the TCP/IP protocol have provided the necessary support to apply remote tests on boards or integrated circuits.

To support remote tests, a client and a server versions of the system were developed. Server and client machines must have the TCP/IP protocol configured properly, otherwise client and server stations will not communicate with each other.

There are some commands that can be issued remotely via menu options. One of these commands is the "Listening" command. Some other commands are: "List Clients", "Reply to all Users", "Reset TAPS", "Capure Test", etc. To perform a "Capture Test", the user has to enable the outputs for the TCP/IP protocol via an "Internet/Enable" command option. To run a remote test there is a "Run/Test"command option. Wishing to repeat the previous test, one must first click the "enable" button, which creates a TCP/IP link, and afterwards click the "Run" button.

8. Conclusions

As pointed out in this paper, some CAE tools for boundary scan testing have been developed and reported in the literature. We propose a CAE framework that supports the IEEE 1149.1 and the LSSD methodologies, using almost the same hardware platform that is needed for boundary scan.

The paper focused on the main characteristics and structure of an object-oriented CAE framework for boundary scan and LSSD test automation, including interfaces to circuit description, chip interconnection, test vector analysis and test vector generation. Additionally, the system incorporates support for remote test, using a client/server operation philosophy which will be described in future papers. The proposed test environment represents a powerful CAE test automation system, including features like project management, remote test submission, menu-based command entry, user-defined configuration of boards and a comprehensive set of manipulation commands. The system will be distributed free of charge to all universities wishing to develop plug-in tools for the system, such as ATPG tools and memory TPG tools.

9. References

[01] **The SCANTEST™ Family** - JTAG Technologies B.V. - Product Catalog.

[02] Anido, M.L, Oliveira, C.E.T. and Alves, V.C., "**An Environment to Perform Functional Tests on Boards and Integrated Circuits**", Proceedings of EUROMICRO 95 - Special Edition by Springer Verlag - to appear.

[03] IEEE Standard 1149.1-1990 "**Test access port and boundary scan architecture**", available from the IEEE Inc., 345 East 45th Street, New York, NY 1007-2394, USA, Order Number SH13144.

[04] K.Parker, "**The impact of boundary scan on board test**", IEEE Design and Test of Computers, pp. 18-31, August 1989.

[05] Haberl, O.F. and Kropf, T., "**Sef Testable Boards with Standard IEEE 1149.5 Module Test and Maintenance (MTM) Bus Interface**", IEEE Proc. of the European Test Conference, pp. 220-225, 1994.

[06] Eichelberger, E., and Williams, T., "**A Logic Design Structure for LSI Testability**," Journal of Design Automation and Fault Tolerant Computing, Vol. 2:2, , pp. 167-178, May, 1978.

[07] T.W.Williams (Ed) "**VLSI Testing**" (Advances in CAD for VLSI, Vol. 5), North Holland, 1986.

[08] H. Fujiwara, "**Logic Testing and Design for Testability**", MIT Press, 1985.

[09] **STL Language** - Cadence Design Systems, Inc.

[10] **HILO Language** - Genrad Company.

[11] K.Parker, "**Integration of CAE and ATE systems**", IEEE Computer Society, 1987.

[12] P. Bardell et al., "**Built-in test for VLSI: pseudorandom techniques**", Wiley, 1987.

[13] Borland International - "**Object Windows Library Reference Guide**".

[14] Pinson, L.J. and Wiener, R.S., "**Applications of Object-Oriented Programming**", Addison-Wesley Publ. Co., 1990.