www.testonica.com

Trainer 1149.1 – Boundary-Scan Teaching Tool

Technical Documentation

Revision: 258

Last modified: 24 November 2009

CONTENTS

LIST OF ABBREVIATIONS

AGM	Alternative Graph Model
BS	Boundary-Scan
BSTT	Boundary-Scan Teaching Tool
BSDL	Boundary Scan Description Language
HTML	Hyper Text Markup Language
GIF	Graphics Interchange Format
GUI	Graphical User Interface
IEEE	The Institute of Electrical and Electronics Engineers
JAR	Java Archive
JDK	Java Development Kit
JPEG	Joint Photographic Experts Group
PCB	Printed Circuit Board
PDF	Portable Data Format
PNG	Portable Network Graphics Format
RTF	Rich Text Format
SSBDD	Structurally Synthesized Binary Decision Diagram
TAP	Test Access Port
TCK	Test Clock
TDI	Test Data Input
TDO	Test Data Output
TDR	Test Data Register
TIFF	Tag Image File Format
UI	User Interface
XML	Extensible Markup Language

1. INTRODUCTION

1.1. About Boundary-Scan Teaching Tool

The developed software is illustrating main principles of boundary scan-based PCB testing and design-for-testability. The main idea of this software is to provide an open training environment where the user can create or import own examples/projects or even experiment with industrial integrated circuits by loading Boundary Scan descriptions (BSDL files), and saving own board configurations using simple netlist format. Besides that, the system is supplied with a built-in collection of chips and boards. In future, it will be possible to extend the system to interact with real hardware and run a new set of real exercises and diagnostic experiments.

Another important feature of the software is its capability to *visualize* most of the aspects of Boundary Scan-based testing. The software visualizes the board under test and BS infrastructure inside each BS-enabled integrated circuit: instruction register, ID register, boundary register, bypass register, etc. The special visualization panel is also illustrating the data flow trough the scan chains and board-level interconnects. Additional interactive panel is used to illustrate the work of TAP controller. Using the same panel it would be possible to control the operation of BS structures on the board.

To sum up, the software gives the user an opportunity to better understand the ideas of IEEE 1149.1 standard by performing interactive experiments with virtual boards and integrated circuits. The application have the following main features implemented:

- Illustration of work of BS registers;
- Simulation of TAP Controller operation;
- Injection and further diagnosis of interconnect faults (shorts, opens, etc.);
- Design or modification of BS structures inside the target chip using the BSDL;
- Design or modification of boards containing several BS-enabled chips;
- Design/description of the target board using several chips.

In the following we assume that the reader is familiar with the PCB test conceptions and the related IEEE 1149.1 standard.

2. PLATFORM

2.1. Common statements

The software is implemented in a form of Java Application, so Java Runtime Environment (JRE) version 5.0 or later is required for operation.

The application works as stand-alone tool, but needs an internet connection for checking for an updated version.

The application has Windows and Unix/Linux distributions.

2.2. Distribution

The software is provided to end user in two forms: zip archive (multiplatform) and executable installer (Windows only). Both packages have an executable JAR file inside as well as documentation and several examples. During extraction or installation procedure user would be free to select any target directory for the application.

Windows installer has options for internal-JRE or system-JRE installation types. The first option will install JRE directly to the installation folder. This guaranties that software will run properly on a correct JRE version, but required additional hard drive space. Second option will configure software to use JRE pre-installed on the system.

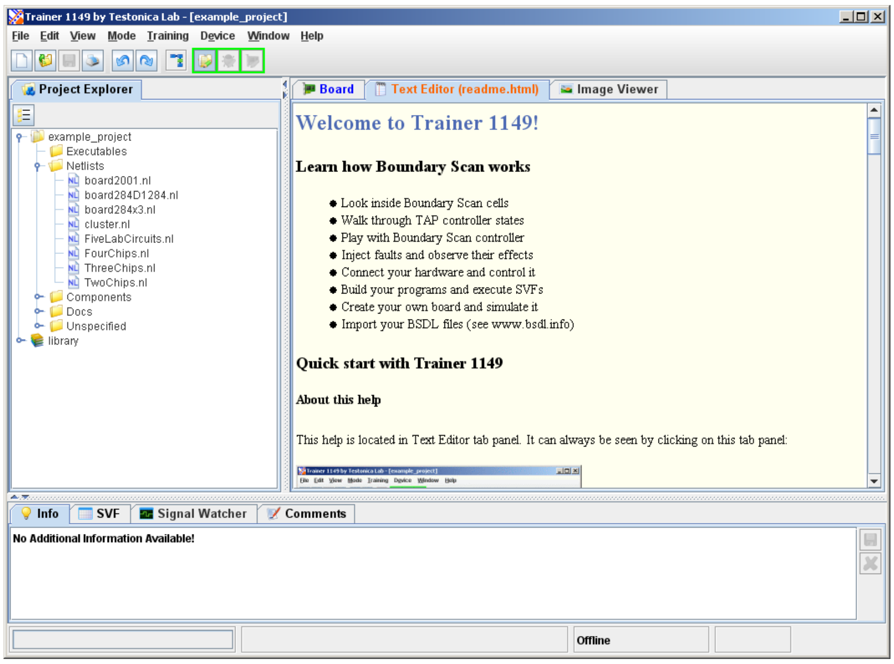
2.3. Update Mechanism

The application has an integrated internet-based update system. During every start-up procedure the it checks whether the current application version number is the latest one or not. If a newer version exists, then user is informed about update released with a special message. This message contains a recommendation to download an updated version as well as direct internet link to new installation file.

3. Graphical User Interface

3.1. *Main Window*

The main window is separated by splitters into three parts: left panel (control), central panel (main information) and bottom panel (specific information). Each panel has a one or several tab panels inside that can be activated by user during workflow. Each tab panel contains one editor or viewer. Number and existence of these panels depends on the current *working mode* (see Section 5). The example of main window is shown in Figure 1.



Welcome to Trainer 1149!

Learn how Boundary Scan works

- Look inside Boundary Scan cells
- Walk through TAP controller states
- Play with Boundary Scan controller
- Inject faults and observe their effects
- Connect your hardware and control it
- Build your programs and execute SVFs
- Create your own board and simulate it
- Import your BSDL files (see www.bsdl.info)

Quick start with Trainer 1149

About this help

This help is located in Text Editor tab panel. It can always be seen by clicking on this tab panel:

No Additional Information Available!

Figure 1

3.2. *Main Menu*

The main menu of BSTT s structured according to menu construction principles of modern applications. The most popular functions of menu (such as opening files and projects, creating new files, copying/pasting to/from clipboard, etc) are also duplicated in the toolbar (Section 3.3).

3.2.1. *File* menu

File menu contains all functions supported by the project management system. Below, each of the menu items is described in detail:

New	Contains sub-menu with a list of formats, that can be designed inside the system. The first of them is *Project*, that provides a dialog window with main parameters of a new project: name and location. For *location* parameter there are two options: default location (from configuration) and user-specific. A list of formats supported by the system comes further: *BSDL, Netlist, SVF, Logic model (SSBDD).* Also there are two additional items: *Folder* (for creating folders) and *Plain File* (for creating plain-text files). All these items provide a common dialog window with a single parameter: name of a new file.
Open	Provides a standard dialog window with possibility to choose file or project to open.
Close	Close opened file in a current viewer or editor
Close Project	Closes currently opened project.
Save	Stores modifications of currently active file. If file was newly created using the application and was not stored to file system yet, then *Save* is considered as *Save as*.
Save As	Provides a standard dialog window with possibility to choose location and name for storing currently active file.
Import	Provides a standard dialog window with possibility to choose file for importing it into the currently opened project.
Export	Provides a standard dialog window with possibility to choose location and name for exporting selected file from the currently opened project.
Print	Provides a standard dialog window for configuration of printer and paper settings. If user confirms printing using *OK* button of this dialog window, the application prints formatted data from active editor.
(Recently opened projects)	Variable list of recently opened projects. Each item of this list opens the corresponding project. This functionality is implemented for the ability to rapidly open frequently-used projects.
Exit	Finishes the work of the application and closes it. All active information as well as internal configuration files will be stored automatically during this procedure.

3.2.2. *Edit* menu

Edit menu contains clipboard operations as well as some common editing functions.

Undo	Editor-specific undo option. Cancels results of last modification.
Redo	Editor-specific redo option. Cancels results of last *undo* command.

Cut	Places selected information from active editor to the system clipboard with subsequent removing of copied part.
Copy	Provides a copying option from active editor to the system clipboard.
Paste	Inserts information from the system clipboard to active editor.
Delete	Provides an editor-specific removing option from active editor.
Select All	Selects all the information in currently active editor.

3.2.3. *View* menu

View menu contains a list of possible representations of board supported by Board Editor. See Section 4 for details.

System Overview	Default representation of virtual board. Visualized components are: board components, signal nets, board inputs and outputs.
TAP Chain	TAP chain oriented representation. Visualized components are BS-supported components only.

3.2.4. *Mode* menu

Mode menu contains a list of available working modes See Section 5 for details.

Project	If selected, Project Working Mode become active. See Section 5.1.
Debug	If selected, Debug Working Mode become active. See Section 5.2.
Board Edit	If selected, Board Edit Working Mode become active. See Section 5.3.

3.2.5. *Training* menu

Inject Fault	Provides a dialog window with parameters for injecting a “virtual” fault into current board. See section 7.5 for details.
Check Fault	Provides a dialog window with all fault variations supported by the system. User is prompt to choose one of them. After choice confirmation by pressing *OK* button, the application checks if the selected fault coincides with injected one and shows a dialog window with corresponded message: is user right or not. In a case, user was right, the application visually demonstrates faulty area and finishes the diagnostic process. Otherwise user is able to continue with the diagnosis.
Remove injected faults	Finish diagnostic process without giving the right answer.

3.2.6. *Device* menu

Device menu has options for connecting and disconnecting BS-enabled boards with LPT connector (see Section 8 for details).

Connect Hardware	Provides a dialog window with number of parameters of hardware connector.
Disconnect	Provides a disconnecting from device procedure.

3.2.7. *Window* menu

Outlook	Two-state menu item. Checking or un-checking this item makes a Outlook Window visible or invisible correspondingly. See section 7.1 for details.
Component Details	Two-state menu item. Checking or un-checking this item makes a Component Details Window visible or invisible correspondingly. See section 7.2 for details.
TAP State Diagram	Two-state menu item. Checking or un-checking this item makes a TAP State Diagram Window visible or invisible correspondingly. See section 7.3 for details.
Settings	Provides a dialog window for configuring the application. See section 9.1 for details.

3.2.8. *Help* menu

Welcome	Provides a Welcome dialog screen. See Section 7.4 for details.
About	Provides a dialog window with common information about product version, authors and copyright.

3.3. Main Toolbar

The toolbar of BSTT contains buttons representing most frequently used functions. The first six items of the toolbar are: *New*, *Open*, *Save*, *Print*, *Undo* and *Redo* (see 3.2.2). Each of these items has completely the same functionality as corresponded menu item.

Seventh toolbar menu item calls TAP State Diagram Window (see 3.2.7).

The toolbar also contains three working mode select buttons that allow to switch between *working modes* (see Section 3.2.4). Pressing one of these items makes corresponded *working mode* active.

3.4. Status Bar

Status bar of the application contains four elements.

The first one is a common used progress bar. It illustrates different kind of internal application processes.

The second one is a common informational label, that can be accessible by any editor or any process. It is used to display information about current event or currently selected element, e.g. "*Project is open*" or "*Simulation is in progress*".

The third one is a label that shows is some hardware currently connected, e.g. "*Offline*" means no any device connected.

The fourth one is a label that shows if a "virtual fault" currently injected into board. simulates fault (the label can have one of two values: "*Fault injected*" or just empty).

4. Board Viewer

Board Viewer is an interactive graphical representation of the board schematic.

Board Viewer supports visualization of three types of elements: BS chips (Figure 2, left), logic clusters (Figure 2, right) and signal nets (wires).

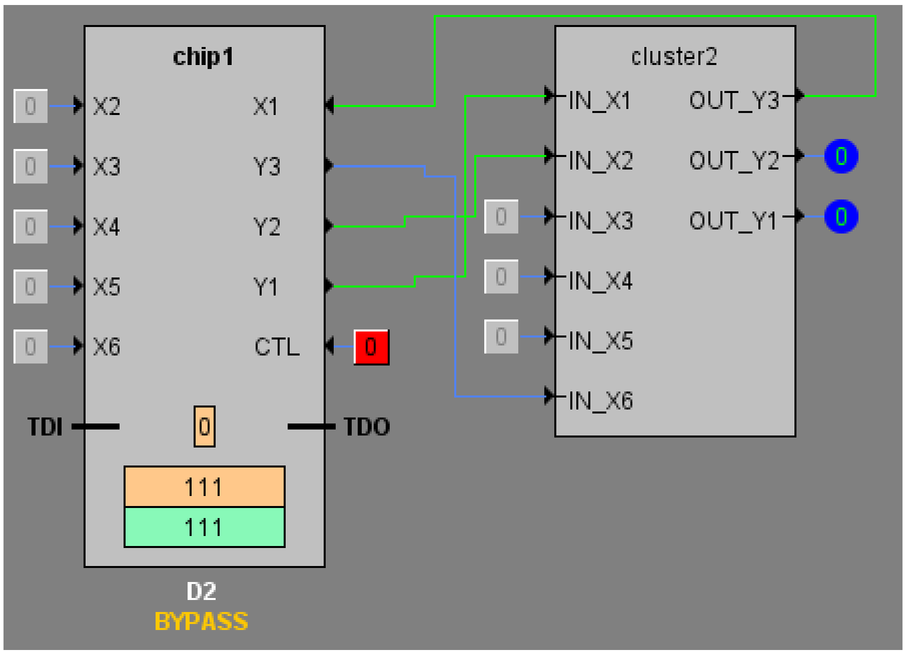


Figure 2

4.1. Components

Board Viewer supports two representations of the loaded board: *System Overview* (presented in Figure 2) and *TAP Chain* (Figure 3). In *System Overview* all supported components are shown in the graphical panel as well as all connections between these components. Some of supported components like BS chips have also parts of internal structure that are illustrated. Details of *System Overview* outlook will be described later.

The *TAP Chain* representation is limited to represent BS scan chain elements only. The goal of this outlook is to show schematically a chain of BS components connected from TDI to TDO. Representation of BS-supported components is also very schematic: just a small gray rectangle with no internal structures neither interconnects (except TDI-TDO line).

BS chips are shown in the *System Overview* as grey rectangles. According to the BS standard all the chips on the board should be connected into a scan chain via TDI and TDO pins.

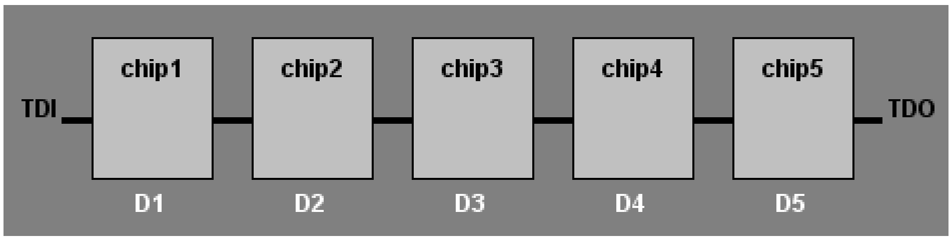


Figure 3

The white area inside the chip represents its core logic, which is not illustrated in detail. The wrapper around the core logic is the Boundary Scan Register. Each scan cell of this register consists of two flip-flops. One is used for capturing the state of the control point (marked by yellow color). Another one (marked by blue) is needed for keeping the control value of the control point during the test mode. The control values and captured responses are shifted in and out in series via TDI and TDO pins.

When cells of BS register are shown inside the chips it is also possible to click on them. This will make a *Component Details* window (Section 7.2) visible. The *Component Details* window contain an image of possible logic-level schematic of selected cell. Ten BS types are defined by the standard are supported: BC_1, BC_2, BC_3, BC_4, BC_5, BC_6, BC_7, BC_8, BC_9 and BC_10.

The register at the very bottom of the chip is the Instruction Register. It shows BS instruction (in binary format) currently selected for the chip.

The one-bit register just above the Instruction Register is the Bypass Register. It is activated by BYPASS instruction and usually used for faster test data shifting.

The Instruction Register and the Bypass Register are mandatory TDRs.

Although BSDL description can also contain some optional design-specific TDRs, they will be not shown inside the chip in the Board Viewer. The only optional registers that will be illustrated are the Device ID Register and Device User Code Register.

Logic clusters are represented as grey rectangles with the name of cluster and signal names written on them (see Fig.3).

The following graphical elements represented by Board Viewer have a tooltip message: BS chips have a "*<name> (<type>)*" message string, e.g. "*chip1 (SN74BCT8244A)*"; BS registers inside chips have "*<type>*" message string, e.g. "*Bypass register*"; logic representation inside BS chips has "*<logic file name>*" message string, e.g. *"SN74BCT8240A.agm*"; clusters and connections have just a "*name*" message string.

The following graphical elements also have a context menu. BS chips have three items in a context menu: *Open <related BSDL file> as Text*, *Open <related AGM file> as Text* and *Open in Graphic Viewer*. The first one opens content of chip related BSDL file in a text editor. In the same way, the second option opens chip related AGM (SSBDD) model of in a text editor. The third option opens BSDL file of corresponded BS chip in the Board Viewer (separately from board). Clusters have only one item in context menu: *Open as Text*. It opens AGM (SSBDD) model of corresponded cluster in a text editor.

4.2. Signals and Wires

Board input pins are represented as clickable buttons. Ordinary data input pins are grey, while the control pins are red. By clicking the pins user can drive a desired input line to either logic 0 or 1. Depending on the BSDL description for a particular chip the value 0 or 1 at a control input pin drives some of the outputs to the high impedance state.

Board output pins are represented as indicators (colored circle with a value), showing the current state of the output line.

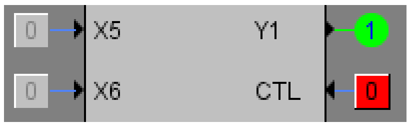


Figure 4

There are four possible states for any signal line in the design. Logic 1 and logic 0 are indicated by green and blue colors correspondingly. The wires driven into the high impedance state are white. In last case, the output indicator will hold the value "Z". If a wire is red or the indicator shows "X", then the logic value of the corresponding signal is unknown. Highlighting signals by different colors is intended for making it easier to follow the simulation. Any changes in the state of a board can be quickly noticed visually.

The wire connecting all the TDIs and TDOs of all the chips together is displayed by bold black line.

4.3. Additional Features

Board Viewer also supports a scaling feature. For this reason, the toolbar of Board Viewer has a special control that allows to enter any percentage zoom value (positive number in a range 1 to 1600) or select predefined value from the drop-down list ("*10%*", "*25%*", "*50%*", "*75%*", "*100%*", "*150%*", "*200%*", "*Fit Width*", "*Fit Height*", "*Fit Page*"). The *Fit Width* option has the same effect as the maximal zoom value, when the left border and the right border of the represented board are still visible on the screen. *Fit Height* is equal to the maximal zoom value, when the top and bottom borders of the represented board are still visible. *Fit Page* is equal to the maximal zoom value, when the represented board is completely visible.

The toolbar of Board Viewer has additional toggle button with highlighting function. When the button is pressed the click on any connection makes it be highlighted with yellow color. Second click un-highlights the connection and restores it normal color. Holding the *Shift* button makes it possible to highlight several connections at once. To un-highlight group of connections user need to click to any other element of board.

5. WORKING MODES

5.1. Project Working Mode

Project working mode of the application provides user with fully-functional project management system. The key GUI element of this system is the *Project Explorer*.

5.1.1. Project Explorer

Project Explorer can be empty or showing contents of one opened project.

Project Explorer can show *library project* in addition to opened project. The elements of opened project and library project are represented by different icons. By altering configuration settings of the application (Section 9.1) user can change the location of the library project or turn off this feature completely.

Project Explorer tree can represent project contents in a two ways: flat layout and hierarchical layout. The first one is a mirror representation of the project file tree in the file system. All folder structures are also present in the tree in this mode. Hierarchical layout contains a built-in filtering system, that groups project files according to their types. Supported groups are the following: *Netlists* (netlist files), *Components* (BSDL and SSBDD files) and *Docs* (text files, MSWord, RTF and PDF documents). Files with unrecognized types are grouped into *Unspecified* group. User can switch layout mode using toggle button in the toolbar:

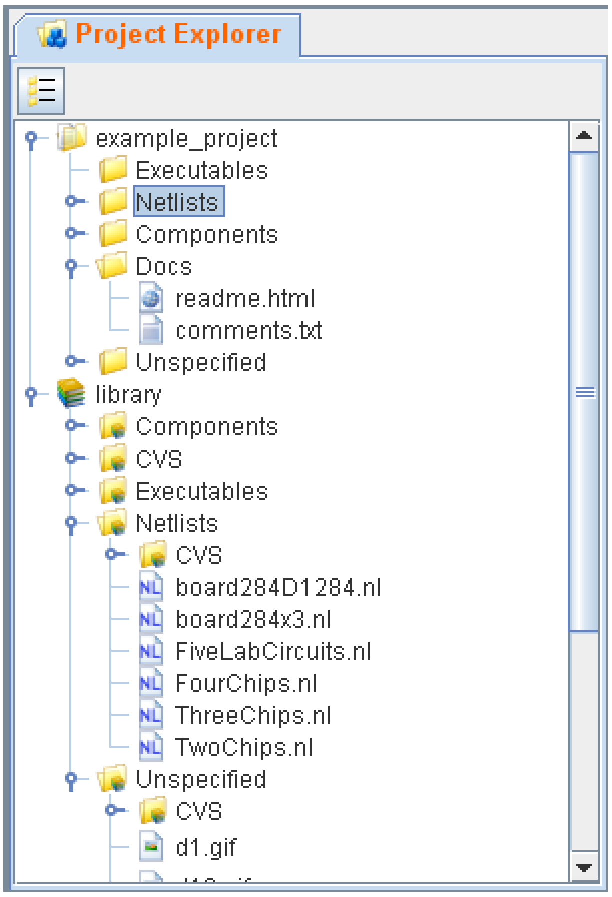


Figure 5

Project Explorer also supports drag-and-drop technique. Files can be moved (or copied, if Ctrl button is held) from one folder to another or from the library project to an opened one.

In addition Project Explorer supports a context menu.

5.1.2. Project Explorer: context menu

Context menu of Project Explorer differs according to currently selected element. In general, there are four types of context menu items: project-specific, folder-specific, file-specific and common items.

The only project-specific context menu item is *Export Project as Zip Archive*. It creates a zip archive that includes contents of currently opened project and then provides a standard dialog window with possibility to choose location and name for the packed file.

The folder-specific context menu items are *Expand* and *Collapse*. *Expand* is available when folder is collapsed and makes contents of this folder be visible. *Collapse* is available when folder is already expanded and hides the contents of the folder. Of course, the same result can be achieved just by clicking the corresponded folder by mouse.

The file-specific context menu items are *Open* and *Open in*. The first one opens file in specific editor according to its type. If file has unknown type it will be opened in text editor by default. The second option is available for files that are supported by several editors. For example, BSDL file can be opened in Board Editor (as a board with one chip) as well as in text editor. The same also make sense for netlist files.

The common items of context menu are listed below:

- *New* – identical to *File->New* menu item.
- *Cut* – identical to *Edit->Cut* menu item, copies selected file with possibility to paste it somewhere else (after pasting source file will be removed).
- *Copy* – identical to *Edit->Copy* menu item. Copies selected file with possibility to paste it somewhere else.
- *Paste* – identical to *Edit->Paste* menu item. Pastes previously copied file into selected folder. In case if file was selected instead of folder, then copied file will be pasted into parent folder of selected file.
- *Delete* – identical to *Edit->Delete* menu item. Removes selected file or folder. If folder is not empty the folder contents will be also removed.
- *Rename* – renames selected file or folder using dialog window.
- *Import* – identical to *File->Import* menu item.
- *Export* – identical to *File->Export* menu item.
- *Refresh* – reloads contents of current project to project tree.
- *Properties* – shows a dialog window with properties of file or folder. The list of properties includes path of file, file size, last modification date, is file read-only or hidden. In case if file format is supported by the application the properties window can also contain format-specific information.

All file operations performed in Project Explorer have immediate effect on the file system.

5.1.3. Text Editor

Text Editor is a special panel for viewing and editing text. Its supports only minimal number of functions: copying, cutting, pasting and removing parts of text and showing the exact caret position (column and row). Any project file can be opened in Text Editor.

5.1.4. Image Viewer

Image Viewer is a special panel dedicated for image viewing. It supports GIF, PNG, Bitmap, JPEG and TIFF image formats.

5.2. Debug Mode

The purpose of *Debug* mode is testing and diagnostic of faults. It combines different possibilities of test pattern insertion, tests applying and results analysis.

There are two ways how the test data can be created:

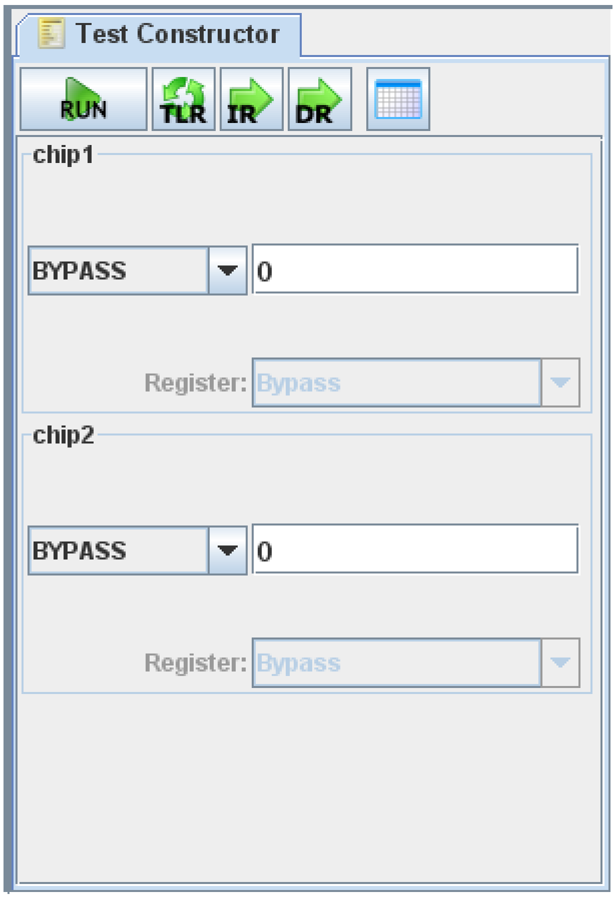
1. Using *Test Constructor* view
2. Directly in the *Board* view.

Both possibilities are synchronized between each other. Therefore, the values of test data vector in *Test Constructor* View and *Board Editor* always match each other. Also there are controlling buttons which start instruction or data shifting processes. They are located in the toolbar of *Test Constructor* view (that can be also considered as a control panel of *Debug* mode).

5.2.1. Test Constructor

Test Constructor panel is shown in Figure 6. It contains lists of available BS instructions for each chip. These instructions are defined in the BSDL description. On the right side from each list a special input fields are located. These fields are used for entering test vectors. The vector inside of this field (it is a part of whole test pattern) will be applied for specific chip. Only 0 or 1 are allowed as a valid input and the length of input data should be equal to the length of currently selected register in a chip. Above these input fields there are list of names of signals connected to BS cells (in case of boundary register) or bit ordering numbers (in case of any other register).

Test Constructor view has five buttons in the toolbar.



chip1
BYPASS 0
Register: Bypass
chip2
BYPASS 0
Register: Bypass

Figure 6

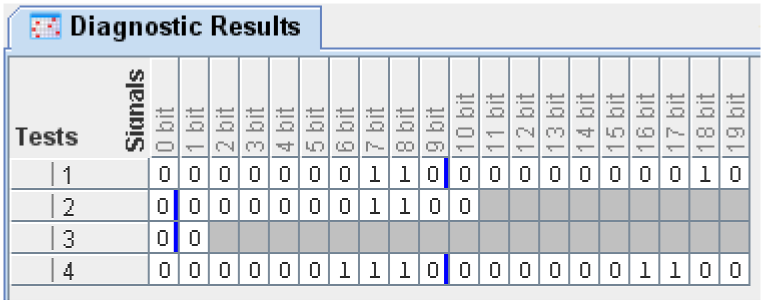
The first one is *Run* button, that equals to sequential pressing *Test-Logic-Reset*, *Scan-IR* and *Scan-DR* buttons. The second one is *Test-Logic-Reset* button, that places TAP controller into the “RESET” state. The third option is *Scan-IR* button, that loads selected instructions to corresponded chips. The fourth option is *Scan-DR* button, that shifts-in input data to corresponded BS register. The last button switches on/off record tracking for SVF editor (Section 5.2.4). Record tracking means that user actions, e.g. scanning data values through IR or DR registers will be automatically stored as SVF test program. This program can be run or exported later on.

5.2.2. Board Viewer: Debug mode

Functionality of Board Viewer is different in the *Debug* mode comparing to *Project* mode. It becomes more oriented to test data insertion. Directly on the board there are input areas inside BS cells, instruction registers and bypass registers. These areas show the value, that will be shifted into this cell during next test data loading. User can modify this value by mouse clicking. The name of BS instruction, which will be loaded into each chip, is shown inside the input area of instruction register of correspondent chip. To modify instruction user chooses another one from the list, that appears with a click on the input area of instruction register. In the same way clicking in the input area of bypass register inverts the value that will be loaded into this register during serial shifting in case if “bypass” instruction is selected.

5.2.3. Diagnostic Results

During the shift-in of test data, the current state of Data Registers is shifted out. The obtained bit sequence (vector) will be represented in the Diagnostic Results view (Figure 7). Diagnostic Results panel has a table-based structure, where one row represents one test vector. After applying a test input, the actual and expected output vectors are automatically compared bit by bit and not matched bits are highlighted with a red color.



Diagnostic Results

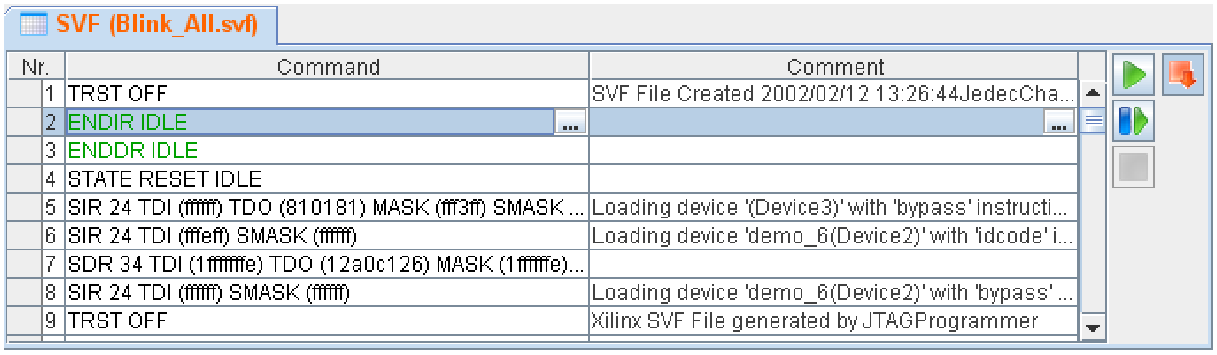
Tests	Signals	0 bit	1 bit	2 bit	3 bit	4 bit	5 bit	6 bit	7 bit	8 bit	9 bit	10 bit	11 bit	12 bit	13 bit	14 bit	15 bit	16 bit	17 bit	18 bit	19 bit
1		0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0
2		0	0	0	0	0	0	0	1	1	0	0									
3		0	0																		
4		0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	0	0

Figure 7

Diagnostic Results view has two functional buttons: *Clear* and *Save*. The first one just removes all rows from the table. The *Save* button provides a standard dialog window with possibility to choose location and name for a new file to which diagnostic results will be stored.

5.2.4. SVF Editor

SVF editor is a panel oriented to the work with SVF files. It has a table structure, where one row represents one SVF command (Figure 8). By double clicking to table cell user can modify parameters of the correspondent SVF command. After click a dialog window with command specific parameters will appear. Parameters are chosen according to SVF specification. Saving of modified is also available using *File->Save* menu item. SVF editor have four two toolbar buttons: run and step-by-step run. First two buttons represent two types of running SVF program: complete and step-by-step correspondently. First one is sending sequentially all commands to the connected device or virtual board or both and the represents responded output results. Using step-by-step running method it is also possible to observe state of the board opened in the Board Viewer after every next command has been performed.


SVF (Blink_All.svf)

Nr.	Command	Comment
1	TRST OFF	SVF File Created 2002/02/12 13:26:44JedecCha...
2	ENDIR IDLE	
3	ENDDR IDLE	
4	STATE RESET IDLE	
5	SIR 24 TDI (ffffff) TDO (810181) MASK (fff3ff) SMASK ...	Loading device '(Device3)' with 'bypass' instructi...
6	SIR 24 TDI (fffeff) SMASK (ffffff)	Loading device 'demo_6(Device2)' with 'idcode' i...
7	SDR 34 TDI (1fffffffe) TDO (12a0c126) MASK (1fffffffe)...	
8	SIR 24 TDI (ffffff) SMASK (ffffff)	Loading device 'demo_6(Device2)' with 'bypass' ...
9	TRST OFF	Xilinx SVF File generated by JTAGProgrammer

Figure 8

A third button on the right side of toolbar is *Stop on Errors* option. If it is selected, running of SVF will be stopped if error appear, otherwise program will continue and error messages will be listed after finish. The last button is *Cancel Execution.* It will immediately stop running program.

5.3. Board Edit Mode

Board Editor Mode allows for any board design to be created using arbitrary BS chips or existing design to be modified. There are two possibilities to work with board design: text-based and interactive. Both possibilities are synchronized with each other. Therefore, board netlist description in text-based editor always reflects changes made in interactive editor.

5.3.1. Board Edit: text

Text version of Board Editor is just a panel which allows entering, editing and viewing text contents of netlist files. It has a support of minimal number of functions only. They are copying, cutting, pasting and removing parts of text and showing exact caret position in a form: column and row. Also there is a refresh button, which overloads currently opened board with changes made in netlist during editing.

5.3.2. Board Edit: interactive Board Viewer

Functionality of Board Viewer is different in the Board Edit Mode comparing to Project Mode. It becomes more oriented to board editing.

There are four additional buttons in the Board Viewer toolbar in case of Board Editing Mode: *Add Component*, *Add Net*, *Remove Net* and *Refresh*.

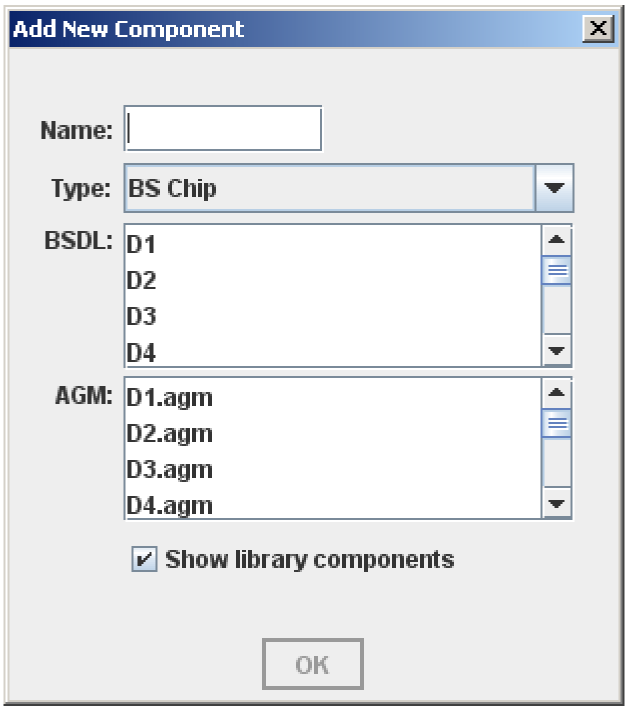
Add Component button provides a dialog window (conceptual UI is illustrated in Figure 9) with the following parameters: name, type (one from listed) and few type-specific settings (e.g. BSDL file for BS chip or SSBDD file for cluster). This dialog has an name controlling feature, that disallows (disables OK button) to use name already existing in the netlist or name deprecated by netlist syntax. List of BSDL files that can be used for describing BS chip consists of BSDL files located in the current project as well as BSDL files located in the library. If user has chosen a file located in the library, then it will be automatically imported to the current project. There is a possibility to turn off the feature of displaying library components in the list using special checkbox.

Add Net is a toggle button, that selects a connection creating mode. If it is selected user is able to create nets by clicking on two or more pins. The existing net can also be extended by clicking on this net and additional pins. When user hovers mouse pointer over some pin it became highlighted.

Remove Net is a toggle button, that controls a connection removing mode. If it is selected user is able to remove whole net by clicking on it or remove one standalone pin from net by clicking on this pin.

Add Net and *Remove Net* buttons are interconnected with each other in a way that selecting one deselects another.

Refresh button updates visual part after there were made some changes in textual part.


Add New Component

Name:

Type: BS Chip

BSDL: D1 D2 D3 D4

AGM: D1.agm D2.agm D3.agm D4.agm

☑ Show library components

Figure 9

When Board Viewer is in the Board Editing Mode it became possible to change the order of the BS chips in a scan chain. This can be performed by clicking on the graphic representation of the chip and dragging it to another location. During dragging a black separator line appears. It shows the exact place where currently dragged chip will be inserted (if user drops it).

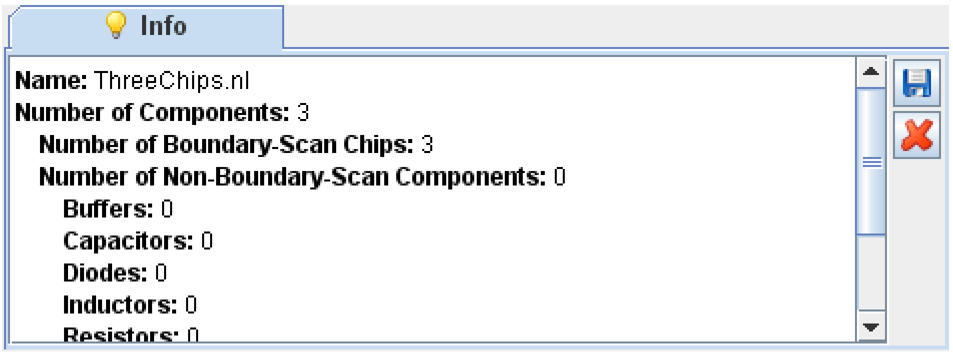
All changes made by Board Viewer are instantly reflected in the textual version of editing netlist.

5.4. Additional Panels

The panels described below are independent on currently selected working mode.

5.4.1. Informational panel

Informational panel contains a text area inside and is aimed for displaying various useful information about currently selected items (e.g. commands/registers for BSDLs, number of I/O and internal signals for SSBDD, etc). It supports informational data in HTML format. Informational panel has two functional buttons: *Clear* and *Save*. The first one removes all text from the informational panel. The *Save* button provides standard dialog window with possibility to choose location and name for a new file to store informational data in HTML format.

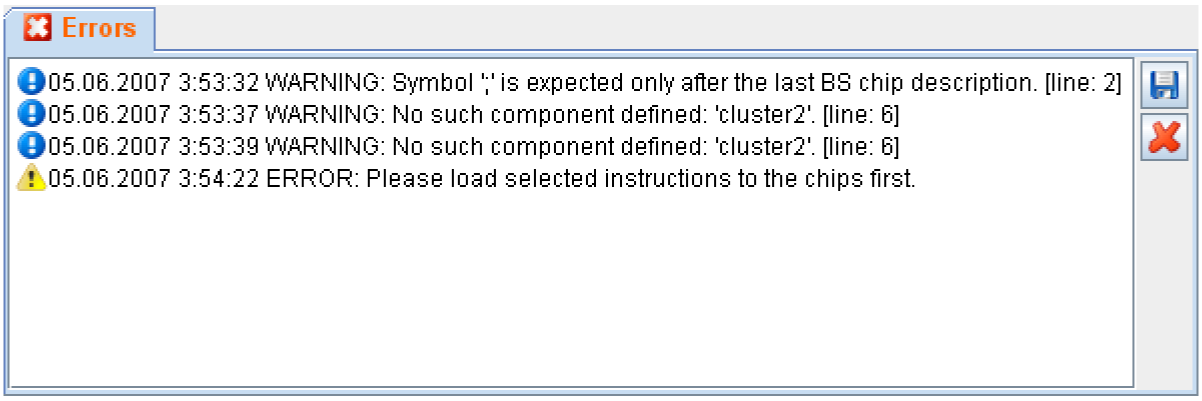


Name: ThreeChips.nl
Number of Components: 3
Number of Boundary-Scan Chips: 3
Number of Non-Boundary-Scan Components: 0
Buffers: 0
Capacitors: 0
Diodes: 0
Inductors: 0
Resistors: 0

Figure 10

5.4.2. Errors panel

Errors panel contains a text area inside and represents contents of application logger. Any error, warning or informational message coming to system logger is immediately shown as one errors panel line. Message types are distinguished by a color: red for error, yellow for warning and blue for information. Errors panel has two functional buttons: *Clear* and *Save*. The first one just removes all messages from the log panel. The *Save* button provides standard dialog window with possibility to choose location and name for a new file to save log messages in a plain text format.



```
05.06.2007 3:53:32 WARNING: Symbol ';' is expected only after the last BS chip description. [line: 2]
05.06.2007 3:53:37 WARNING: No such component defined: 'cluster2'. [line: 6]
05.06.2007 3:53:39 WARNING: No such component defined: 'cluster2'. [line: 6]
05.06.2007 3:54:22 ERROR: Please load selected instructions to the chips first.
```

5.4.3. Comments panel

Comments panel contains a text area inside and gives user an opportunity to enter some comment to opened project. It can be any plain text message and will be automatically stored as a text file inside project. If user opens project with already existing comment message it will be shown in Comments panel, where it can editor or cleared.

6. Formats

6.1. SVF

SVF parser is implemented according to SVF specification^1.

6.2. BSDL

BSDL parser is implemented according to BSDL specification^2.

6.3. Netlist Format

Netlist format parser has support of a number of components and multi-pin nets.

6.3.1. Netlist format syntax

Netlist description consist of several blocks. Each block begins with one of the following statements: *#BS*, *#NON-BS, #CLUSTERS* and *#NETS*. Block ends with starting of another block or in case if end of file occurs. Each of these blocks can be specified in a description only once. *#NETS* must always be the last block in a file.

#BS block is a description of BS-enabled chips presented in board. The statements of this block have the following format:

<name of chip>:*<name of BSDL description of chip>*[(*<name of SSBDD file of chip>*)]

Name of SSBDD file can be omitted if there are no need in simulation of internal chip logic. The description of one chip can be followed by the description of next chip. In that case a special sign ">" (which symbolically illustrates chips ordering relation in the scan chain) should be used. To finish description of chips a special sign ";" is used as a terminator. A number of chips in a scan chain is not limited.

#NON-BS block is a description of all non-BS components excepting logic clusters. Currently the following elements are supported: buffer, capacitor, diode, inductor, resistor, switch and transistor. The format of non-BS component description block is the following:

<name of component>:*<type of component>*;

Where type is one of the following: *RESISTOR*, *TRANSISTOR*, *CAPACITOR*, *INDUCTOR*, *BUFFER*, *SWITCH.*

A semicolon is used as a separator between descriptions of elements. A number of components described in *#NON-BS* block is not limited.

#CLUSTERS block is a description of logic clusters presented in the netlist in the following format:

<name of cluster>:*<name of SSBDD description of cluster>*;

Descriptions of logic clusters are separated from each other by semicolon. A number of components described in *#CLUSTERS* block is not limited.

#NETS block is a description of connections between components described in *#BS*, *#NON-BS* and *#CLUSTERS* blocks. *#NETS* block must be the last one in the netlist description. The format of net description is the following:

<name of net>:*<name of component1>*.*<name of pin1>*-*<name of component2>*.*<name of pin2>*[-*<name of component3>*.*<name of pin3>*][…];

Each endpoint in the net description is separated by using minus sign '-'. A semicolon is used to separate descriptions of nets. A number of nets described in *#NET* block is not limited.

There are a certain naming conventions for components and nets. Name should be a text string, which contains numbers and letters only (but name cannot start with a number). Note, that the format keywords (such as *#BS*, *#CLUSTERS, #INDUCTOR* and others) can also be used as names.

Netlist format is case insensitive.

Netlist format can contain comments. The part of line after an exclamation symbol '!' or a pair of slashes '//' is considered as a comment and should be ignored by netlist parser.

6.3.2. Netlist format example

The text below is an example of netlist description of board with 2 BS-enabled chips and one buffer:

```
#BS
chip2:sn74bct8244a(sn74bct8244a.agm)>
chip1:sn74bct8244a(sn74bct8244a.agm);
#NON-BS
buf1:Buffer;
#NETS
net0:chip1.y1(1)-chip2.a2(3);
net1:chip1.y1(2)-chip2.a2(2);
net2:chip1.y1(3)-chip2.a2(1);
net3:chip1.y1(4)-chip2.a1(4);
net4:chip1.y2(1)-chip2.a1(3);
net5:chip1.y2(2)-chip2.a1(2);
net6:chip1.y2(3)-buf1.pin1;
net7:chip1.y2(4)-buf1.pin2;
```

7. Additional Tools

7.1. Outlook Window

Outlook Window (Figure 11) is a window with board graphic representation identical to Board Viewer. Goal is to make navigation through complex board easier. Outlook window has four functional elements: "+" button, "-" button, slider and board graphic representation itself. "+" button increase zoom value of Board Viewer to one step. "-" button decrease zoom value of Board Viewer to one step. These steps are: 10%, 25%, 50%, 75%, 100%, 150%, 200%. Slider value is also strongly connected with zoom value of Board Viewer. Possible values of slider are from 10% to 200% with step 1%. Graphic board representation located in a central part of Outlook Window has a frame inside.

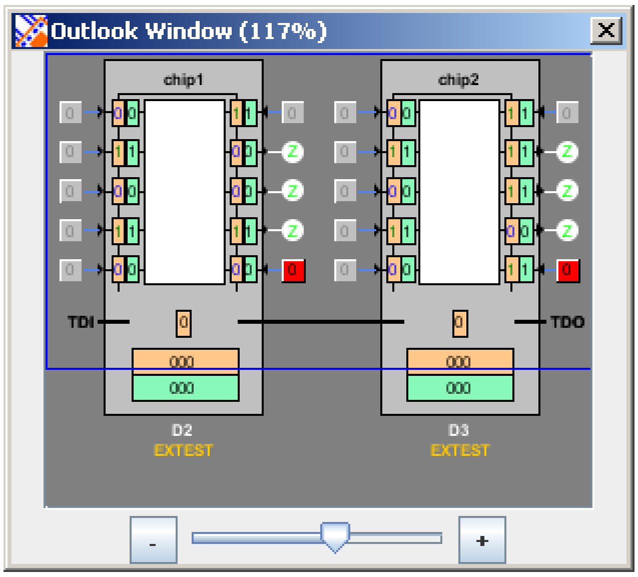


Figure 11

This frame represents a currently visible in Board Viewer part of board. User is able to move this frame making visible another board parts. Any action made in Outlook Window including pressing buttons, changing slider value and moving visible part frame has immediate effect in Board Viewer.

7.2. Component Details Window

Component Details Window is a simple window with possibility to view image files. It can be called from any editor according to editor specific needs.

7.3. Interactive TAP Diagram Window

Interactive TAP Diagram is a graphical representation of TAP controller (that is implemented as a finite state machine). There are three buttons representing three TAP controlling signals: TMS (toggle), TDI (toggle) and TCK. Pressing of TCK button generates a raising edge of clock and changes the state of TAP controller according to TMS button signal (1 if toggled, otherwise 0). Releasing of TCK button generates falling edge of clock and produces corresponded action for current TAP state (e.g. data shifting in SHIFT-DR or SHIFT-IR states). The value of TDI button (1 if toggled, otherwise 0) will be shifted in during shifting operations.

Another possibility to control state changes of TAP diagram is clicking directly on the graphical representation of states. If mouse pointer is over some state the path to this state from the current one becomes highlighted. Clicking some state will force state machine to enter this state. During this procedure all intermediate states passed by TAP controller will be sequentially highlighted.

7.4. Welcome Screen

7.5. Fault Injection Mechanism

7.5.1. Overview

BSTT also provides a fault injection mechanism. The supported fault models are stuck-at fault model, wired-and, wired-or and dominant fault models.

The type of fault to be injected is selected using *Inject Fault* dialog (see section 7.5.2). It is also possible to insert a random fault. Also the faulty net (or two nets for wired/dominant fault models) can be selected from the provided list or randomly.

After the fault is inserted, all signals in Board Editor are losing value-specific color (become black). By applying the sequence of test user should try to find the faulty signal and identify the type of fault. After the fault is identified, the diagnosis can be checked using *Training -> Check Fault* menu item. To remove the injected fault from the board *Training ->*

Remove Injected Faults menu item should be used.

If the fault is injected into the board the corresponded mark is shown in the status bar.

7.5.2. Inject Fault

Inject Fault Dialog is shown in Figure 12. User is able to insert a completely random fault or choose the type of fault from the list.

Stuck-at-1 and *Stuck-at-0* fault types allow user to choose one net to inject the fault. *Wired-And*, *Wired-Or* and *Dominant* allow user to select two shorted nets.

After inserting the fault by using *Inject Fault* button, board in Board Editor begins to simulate it. User is able to investigate the behavior of faulty board using Debug Mode (see section 5.2).

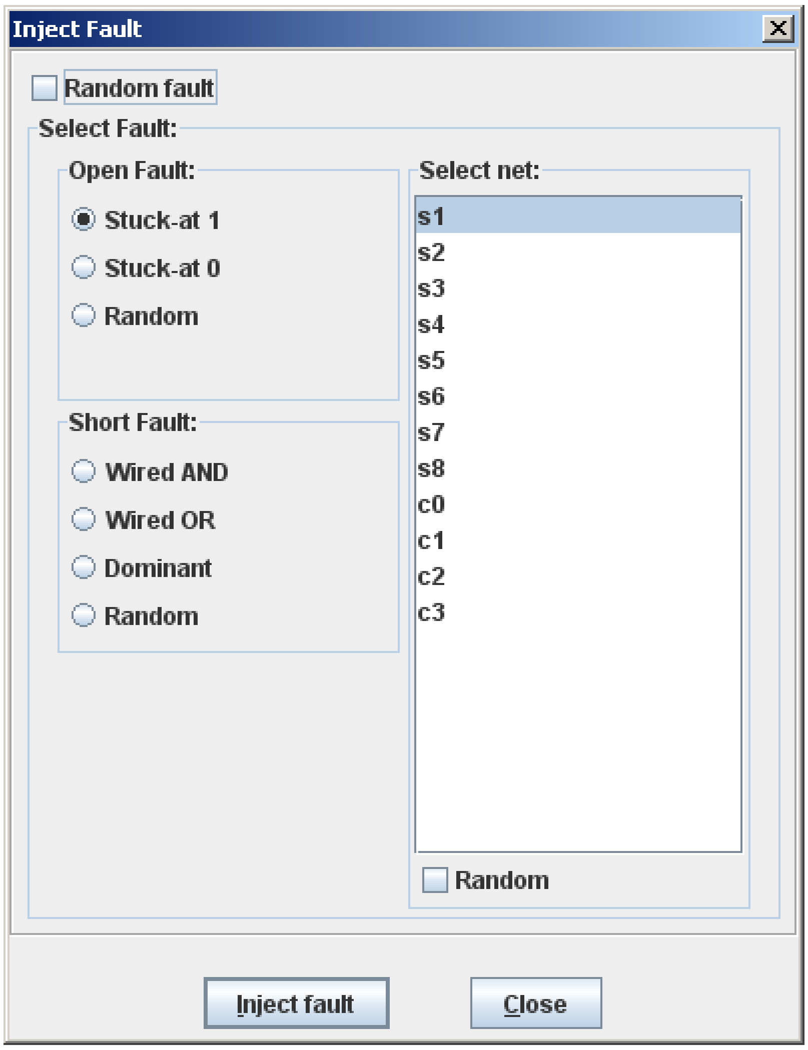
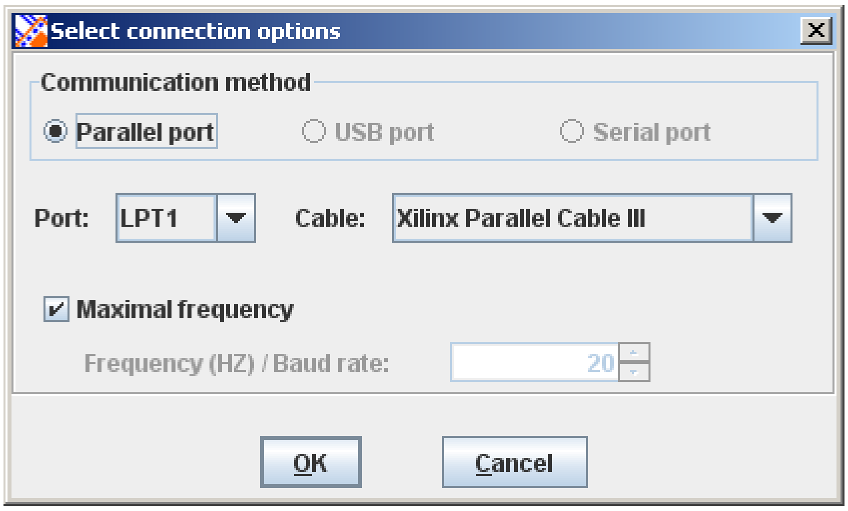

Inject Fault
Random fault
Select Fault:
Open Fault:
Stuck-at 1
Stuck-at 0
Random
Short Fault:
Wired AND
Wired OR
Dominant
Random
Select net:
s1
s2
s3
s4
s5
s6
s7
s8
c0
c1
c2
c3
Random
Inject fault
Close

Figure 12

8. HARDWARE

8.1. Connect Device window

Using *Connect Device* dialog (Figure 13) it is possible to plug in hardware device to the application. Parallel (LPT) port and Xilinx Parallel Cable III can be used for connection. User is also able to select maximal frequency that will be used for transferring data. After connection is confirmed by pressing *OK* button, the application enters into *Online* mode.



Select connection options

Communication method: [✓ Parallel port] USB port Serial port

Port: LPT1

Cable: Xilinx Parallel Cable III

[✓] Maximal frequency

Frequency (HZ) / Baud rate: 20

Figure 13

8.2. Working with Hardware

In the *Online* working mode test data will be sent to the virtual board as well as to real hardware device. This means, that values of TDI, TMS and TCK signals (that are represented by the buttons of TAP Diagram window) will be sent to the hardware device. When data is sent through SVF Editor the bit sequence responded from TDO port of connected device will analyzed and compared with expected result.

8.3. Disconnecting Hardware

User can disconnect hardware at any time by selecting *Device->Disconnect* menu item. Because of the fact that interrupting data transfer can damage hardware, application will ask user for confirmation before the disconnection. After device is disconnected the application will return back into the *Offline* mode.

9. Configuration Management

9.1. Settings Dialog Window

Settings Dialog Window is used to configure application settings. It is accessed by *Window->Setting* menu item. The only one configuration panel now is *General* panel. It contains common application settings, like working path, size of recently used files list, environment start-up settings (start with no project, start with default project or start with most recent project), disable/enabled splash-screen displaying option and disable/enabled welcome-screen displaying option

9.2. Format of Config.xml file

Configuration settings of the application are stored as XML data in */config/config.xml* file.

The root node of this XML document is **<configuration>** with one possible attribute **name**. For example:

```
<configuration name="defaultConfiguration">
```

The first child node is **<framework>** representing a common application settings. It has a number of attribute nodes: **defaultProject** (path of default project), **lastUsedPath** (last path used by application), **splash** (if true, then splash screen will be displayed during the application start-up), **startUpType** (if 1, then application will start-up with a most recent project; if 2 then application will start-up with a default project, otherwise application will start with no opened project) and **workingDir** (application working path).

The example is provided below:

```
<framework defaultProject="c:\projects\project1149\"
lastUsedPath="c:\projects\project1149\" splash="true"
startUpType="2" workingDir="c:\projects\project1149\">
```

The **<framework>** node has one child node: **<recentlyUsedFiles>**, which contains a list of recently used projects. There is a one attribute in **<recentlyUsedFiles>**, the **`size`** with a non-negative integer value of recently used files list size:

```
<recentlyUsedFiles size="3">
```

The **<recentlyUsedFiles>** node has a number of **<file>** child nodes. Each of them represents one recently used project. These nodes have two attributes: **name** (name of recently used project) and **path** (path of recently used project). For example:

```
<file name="project1149" path="C:\projects\project1149"/>
```

The next child node is **<projectExplorer>**, which contains Project Explorer specific settings. The first child node of **<projectExplorer>** is **<newProjectDefaultFolders>**, which contains list of folders that will be created with new project. The **<newProjectDefaultFolders>** node has a number of **<folder>** child nodes, each of them represents one folder. These nodes have one attribute: **name** (name of folder). For example:

```
<folder name="Components"/>
```

The next child node of **<projectExplorer>** is **<projectsLocation>** with a **path** attribute:

```
<projectsLocation path="c:\projects"/>
```

The next **<projectExplorer>** child node is **<library>** with a **path** and **enabled** attributes: **path** is a path of the library. If **enabled** attribute is true, then library is supported. Example:

```
<library enabled="true" path="c:\library"/>
```

The complete example of *config.xml* file is provided below:

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration name="defaultConfiguration">
     <framework defaultProject=""
     lastUsedPath="C:\projects\project1149" splash="false"
     startUpType="2" workingDir="C:\projects">
          <recentlyUsedFiles size="3">
               <file name="project1149"
               path="C:\projects\project1149"/>
          </recentlyUsedFiles>
     </framework>
     <projectExplorer>
          <newProjectDefaultFolders>
               <folder name="Components"/>
               <folder name="Unclassified"/>
          </newProjectDefaultFolders>
          <projectsLocation path="C:\projects"/>
          <library enabled="true" path="C:\library"/>
     </projectExplorer>
</configuration>
```

REFERENCES

[1] Serial Vector Format Specification, Revision E, 8 March 1999

[2] IEEE 1149.1b-1994, “IEEE Standard Test Access Port and Boundary-Scan Architecture”, Annex B, 1994.