



FP6-IST-507219

PROSYD

Property-Based System Design

Instrument: Specific Targeted Research Project

Thematic Priority: Information Society Technologies

Property-based analysis of simulation traces

(Deliverable 3.2/14)

Due date of deliverable: 30.06.2006

Actual Delivery date: 28.06.2006

Start date of project: 01.01.2004

Duration: 3 years

Organisation name of lead contractor for this deliverable: IBM

Revision: 1.0

Project co-funded by the European Commission within the Sixth Framework Programme (2000-2006)		
Dissemination Level		
PU	Public	<input checked="" type="checkbox"/>
PP	Restricted to other programme participants (including the Commission Services)	<input type="checkbox"/>
RE	Restricted to a group specified by the consortium (including the Commission Services)	<input type="checkbox"/>
CO	Confidential, only for members of the consortium (including the Commission Services)	<input type="checkbox"/>

Notices

For information, contact pidan1@il.ibm.com.

This document is intended to fulfil the obligations of the PROSYD project concerning deliverable 3.2/14, described in contract number 507219.

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

© Copyright PROSYD 2004-2006. All rights reserved.

Table of Revisions

Version	Date	Description and Reason	By	Affected Sections
0.1	01.05.2006	Document initial structure	Dmitry Pidan	new
0.2	21.05.2006	User Guide draft	Michael Shamis	Chapter 4
0.3	28.05.2006	First draft of the document	Dmitry Pidan	all
0.4	01.06.2006	Revised draft	Dmitry Pidan	all
0.5	21.06.2006	Revised after manager's review	Dmitry Pidan, Michael Shamis	all
1.0	28.6.06	Final version	Cindy Eisner	Version number

Authors

Dmitry Pidan

Michael Shamis

Executive Summary

This document provides a description of a trace analyzer tool for the post-simulation PSL specification-based analysis of the simulation traces. The materials in this document cover the basic description of the trace analysis process and provide a simple user's guide for the trace analysis tool.

Purpose

The purpose of this document is to provide a description of the IBM trace analysis tool.

Intended Audience

This document is intended for anyone who uses PSL for the trace analysis process.

Background

Trace analysis, sometimes referred to as "offline checking", is a part of the dynamic verification process. Contrary to simulation with monitors ("online checking") trace analysis is a post-simulation process that does not affect simulation time. The process works on the simulation output ("trace"), and verifies it against the provided specification

The trace analysis process has some advantages and disadvantages when compared to simulation with monitors. The greatest advantage of it is that it does not slow down the simulation time, which is a major issue for some projects. On the other hand, the trace analysis process very often requires a great deal of the storage space to save the potentially huge simulation trace. Given the advantages and disadvantages, the verification engineer should decide, which verification method is more suitable for this work – simulation with monitors, trace analysis, or both.

Contents

Table of Revisions	iii
Authors.....	iii
Executive Summary	iii
Purpose.....	iii
Intended Audience	iii
Background.....	iii
List of Figures	vi
List of Tables	vi
Glossary	vii
1 Introduction.....	1
2 Technical approach.....	2
Trace Analysis in Dynamic Verification	2
Verifying the Trace.....	3
Trace Reader	3
Model Builder	3
Simulation Engine.....	5
3 User manual	6
Main Window Overview	6
Opening Files.....	7
Opening the Trace File	7
Opening the PSL File.....	7
Setting the verification clock	8
Viewing and Selecting PSL Assertions	8
Verifying the trace	8
Running the Verification	8
Viewing the Verification Results.....	9
Manipulating the trace view pane.....	9
Adding and Removing Signals to/from the Trace View	9
Moving Signals on the Trace View	9
Adding a Visual Separator.....	10
Adding a Bus Sub-Range to the Trace View.....	10
Flipping the Bus in the Trace View	10
Combining Signals.....	10
Change the Base of Buses.....	11
Change Colours of Signals	11
Selecting All Signals in the Trace View	11
Zooming the Trace View	11
Navigating in the Trace View	12
4 Tool Properties.....	13
Feature List	13
Work Summary.....	14
Starting Point	14
Work Performed	14

Supported PSL Subset	14
5 References	15

List of Figures

Figure 1: Dynamic verification flow with trace analysis	2
Figure 2: Flow of a trace analysis	4
Figure 3 : Panes of trace analyzer main window.....	6
Figure 4: Dialog for the composition of simple Boolean expressions	12

List of Tables

Table 1:Property "assert always {a;b;c} =>{d}" simulation running	5
Table 2: Feature list.....	13

Glossary

Assertion:

A property that is expected to hold for a specific design.

HDL (Hardware Description Language)

One of several specialized high-level languages used by semiconductor designers to describe the features and functionality of chips and systems prior to handoff to the IC layout process. HDL descriptions are used in both the design implementation and verification flows. Currently, the two standard HDLs in use worldwide are Verilog HDL and VHDL. Several proprietary HDLs also exist, mainly for describing logic that is targeted for vendor-specific programmable logic devices.

Simulation monitor

An HDL code unit that runs with the design in the simulation, checks the design under test for the user-defined properties, and reports an error for any property that is violated. A simulation monitor is sometimes referred to as "checker".

Trace

A record of the design signals behaviour during the running of the design. In general, the trace is an output of the simulation or formal verification tools.

Value Change Dump (VCD)

A commonly used format for storing a trace data. This is part of Verilog IEEE standard [4].

Vunit

A PSL construct that can contain a number of assertions and additional modelling code.

1 Introduction

Trace analysis is one of the commonly used methods in dynamic verification. In this method, a trace is verified against the specification to see whether the specification holds on this trace. When the trace is produced by the simulation, a trace analysis can be viewed as an alternative for the simulation with monitors, while the checking of the design behaviour during the simulation is replaced by the post-simulation analysis of this design behaviour on the simulation output – a trace. Trace analysis is sometimes referred to as "offline checking", while simulation with monitors is sometimes called "online checking". The conceptual difference between the two approaches is clear. While simulation with monitors checks the design behaviour "online" during the simulation process itself, the trace analysis does the same work "offline" after the simulation.

There are several advantages and disadvantages of trace analysis versus simulation with monitors. One of the greatest advantages of it is that it does not affect the simulation time. In many verification projects, the performance of the simulation is a very important, if not critical, issue, while adding monitors to the simulation process natively slows it down. On the other hand, trace analysis, which is a process that runs independently of the simulation, can run concurrently to it and thus be more efficient in terms of time spent on the verification than simulation with monitors. However, the trace analysis process requires sufficient storage space to save the whole trace of the simulation; the size of the trace can sometimes be huge.

Trace analysis can also be used for debugging purposes. Consider the situation where there is known that the design under verification contains a bug but, but the exact problem remains still to be determined. A set of checks can be provided for the purpose of determining exactly where the incorrect behaviour occurs. Since the entire simulation is an expensive task in terms of time and resources, trace analysis can be used in order to verify this set of checks and find the bug.

This document presents a tool for trace analysis based on PSL [5] specifications. PSL is an assertion language, accepted by Accellera and, more recently, by the IEEE as a standard. The PSL language allows the user to specify properties that describe design behaviours and/or checks to be performed in a mathematically precise and compact way. PSL is a language for assertion-based verification – a verification concept in which assertions are used to verify design behaviour.

The remainder of this document is organized as follows: Chapter 2 presents a high level description of the trace analysis process and its implementation. Chapter 3 presents a short user guide for the tool and Chapter 4 describes the main features of the tool and summarizes the work done.

2 Technical approach

This chapter describes the technical aspects of the trace analysis tool including how the tool evaluates a trace using the PSL assertions [5]. The tool is built on top of the IBM FoCs – tool for generating simulation monitors [2], [3] and uses an algorithmic framework reported in [1].

Trace Analysis in Dynamic Verification

Dynamic verification using trace analysis is a post-simulation process, as opposed to dynamic verification done with simulation monitors [1]. It does not have any online connection to the simulation itself and does not affect the simulation in any way. The interface between the simulation process and trace analysis process is a trace in some format (VCD [4] for the trace analysis tool discussed in this document).

Figure 1 illustrates the flow of the dynamic verification process with the trace analysis. As can be observed from the Figure, PSL properties do not participate in the simulation process; they only enter the flow afterwards.

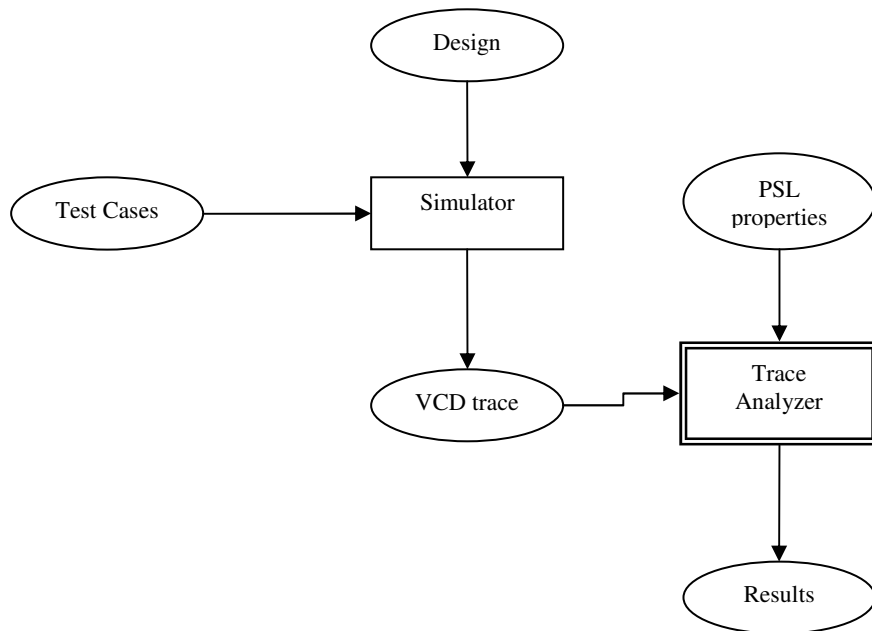


Figure 1: Dynamic verification flow with trace analysis

VCD trace is written by the simulation. As far as VCD is a commonly used standard [4], most of the industrial simulators support generation of the output traces in this format. The trace analyzer reads the trace on a cycle-by-cycle basis and verifies that the trace satisfies the specification provided as the PSL properties. In some ways, the trace analysis process can be compared to simulation with PSL-originated simulation monitors, where the design and test cases are replaced by a trace.

Verifying the Trace

PSL properties are converted into an executable model [1], to verify them on the trace. This executable model performs an evaluation of the PSL properties in the current cycle. Given the values of the signals that may affect property evaluation, the executable model checks whether the property fails based on the internal state of the property evaluation. The trace analyzer reads the trace cycle by cycle and runs the executable model of the PSL properties on every cycle. The output of one executable model iteration is a new state of the properties evaluation and/or failure of the properties.

Figure 2 depicts the flow of the trace analysis process.

Trace Reader

The trace reader module reads the VCD trace and converts it to an internal representation. This internal representation is used by the trace viewer to represent a trace and by a model runner that extracts information about the change of the values and cycle information. The cycle information is based on the clock signal characteristics provided by the user.

Model Builder

The model builder module reads the PSL properties and builds an executable model that evaluates the source PSL properties at a given point in time. An executable model is an internal representation of the properties as a set of statements that evaluates that the property holds at the specific point in time. At any point in time, the model keeps its internal state, in order to use it the next time an evaluation is performed. The resulting executable model performs the following operations:

- Update – updates the new values of the signals that affect the properties.
- Reset – sets the internal state of the model to the initial value.
- Evaluate – performs a single time-point evaluation of the PSL properties based on the current signal values, updates the internal state and reports failures if necessary.

The process of model building is similar to building the simulation monitor [3] with one exception: the stage of the HDL code writing is omitted; instead, an interface for running the model is built.

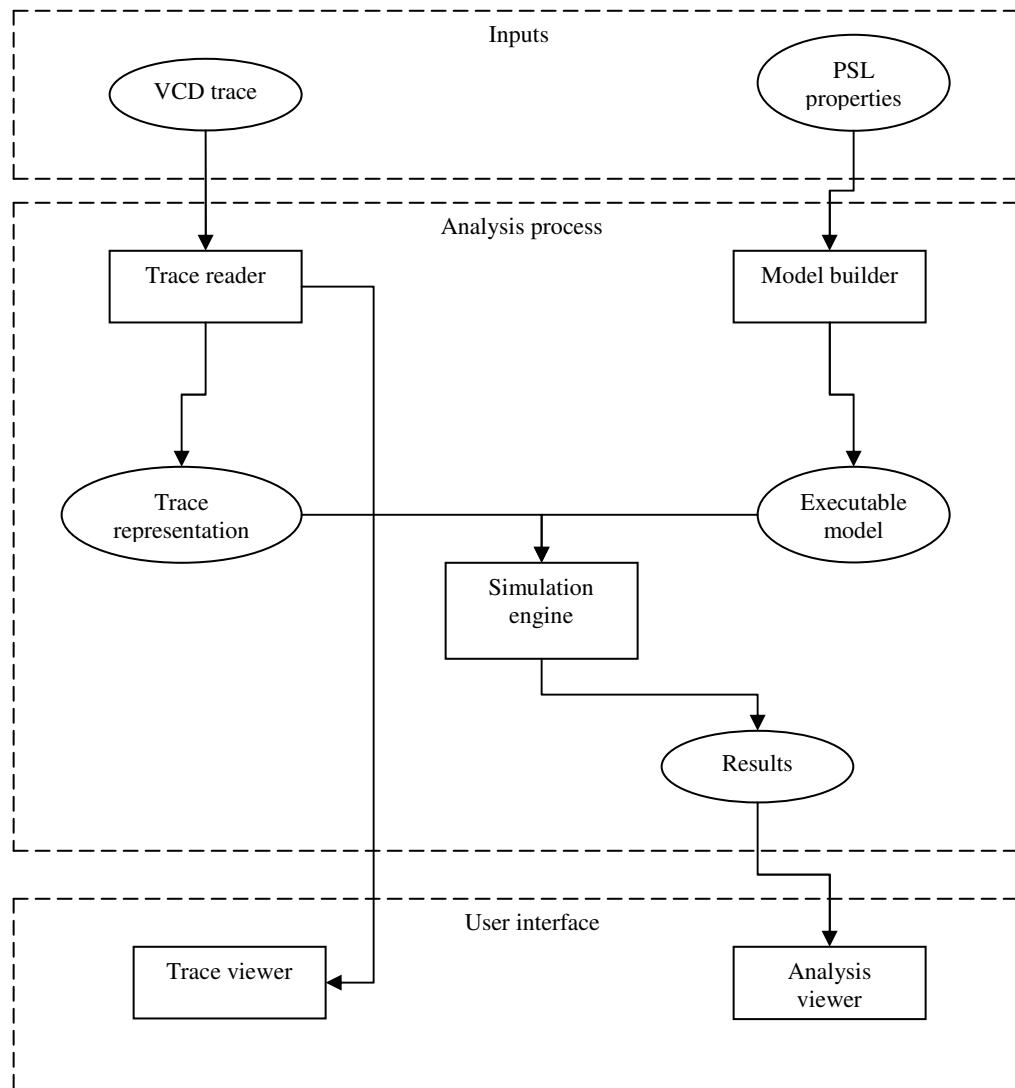


Figure 2: Flow of a trace analysis

The following example presents the executable model that evaluates "assert always { a; b[*]; c } |=> { d }" PSL property, written as pseudo-code:

```

Update(a, b, c, d)
{
  _a = a;  _b = b;  _c = c;  _d = d;
}
Reset()
{
  es = "11000"  // evaluation state
}
Evaluate()
{
  tmp_es = "00000"
  if es[0] and 1 then
    tmp_es[0] = 1;  tmp_es[1] = 1;
  if es[1] and _a then
    tmp_es[2] = 1;  tmp_es[4] = 1;
  if es[2] and _b then
    tmp_es[2] = 1;  tmp_es[3] = 1;
  if es[3] and _c then
    tmp_es[4] = 1;
  
```

```

    if es[4] and !_d then
        report failure
    es = tmp_es;
}

```

Simulation Engine

The simulation engine module is an engine of the trace analyzer. It receives the trace as an internal representation from one side and receives an executable model of PSL properties from another side. The module reads the information from the trace internal representation one cycle at a time and runs the executable model for every cycle.

Simulation algorithm:

```

Reset;
Go to the first cycle;
While not at the end of the trace do
    Update (using values from the current cycle);
    Evaluate;
    If failure (result of evaluate) report;
    Go to the next cycle;
End do

```

Table 1 demonstrates the behavior of the simulation engine run on the executable model that evaluates the PSL property "assert always { a; b[*]; c } |=> { d }":

a	b	c	d	action	_a	_b	_c	_d	Internal state
0	0	0	0	Reset					11000
				Update	0	0	0	0	
				Evaluate					
1				Update	1				
				Evaluate					11100
0	1			Update	0	1			
				Evaluate					11010
	0	1		Update		0	1		
				Evaluate					11001
				Update					
				Evaluate					11001 Failure

Table 1:Property "assert always {a;b;c}|=>{d}" simulation running

3 User manual

This chapter presents a short user's guide for the trace analysis tool discussed in this document.

Main Window Overview

The main window of the tool consists of four panes:

- **Vunits list pane** – located on the left of the main window. List all the vunit names in the PSL properties file.
- **Trace view pane** – the largest pane, located in the middle of the screen. Displays the signals loaded from the trace and their values during different trace cycles. You can easily configure the signals, zoom level and cycles range.
- **Log pane** – located at the bottom of the screen. Displays the run progress, parse errors and any other messages generated by the tool at runtime.
- **Assertions text pane** – located above the log pane. Displays the text of the selected assertions in the PSL file.

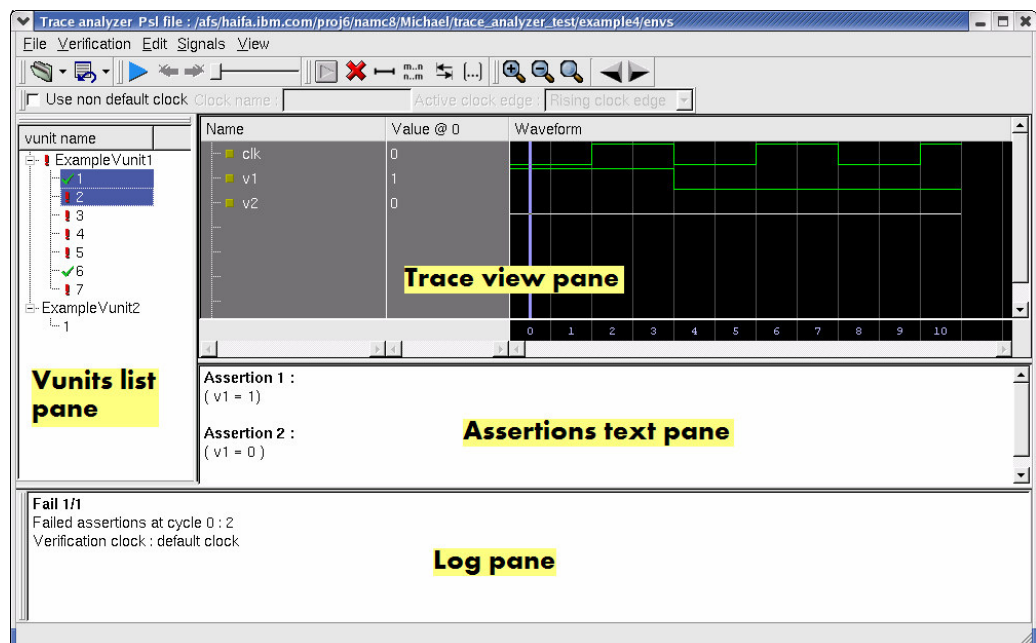


Figure 3 : Panes of trace analyzer main window

Opening Files


Opening the Trace File

Before you run the verification, you should open the trace on which the assertions will be checked.

To open the trace file using the menu:

1. Select **File->Open->Open Trace File**.
2. In the dialog, choose the trace file on which to check your assertions.
3. Click **Open**.

Or

1. Click the  icon in the toolbar.
2. Select **Open Trace File** in the drop down menu.
3. In the dialog, choose the trace file on which to check your assertions.
4. Click **Open**.

If the format of the trace file is illegal you are notified of errors in the **Log pane**.

To reopen the trace file:

1. Click the  icon.
2. Select **Reopen Trace File** in the drop down window.

Note: reopening the verification file invalidates all verification results.


Opening the PSL File

Before you run the verification, you should open the PSL file that contains the assertions to be checked.

To open the PSL file using the menu:

1. Select **File->Open->Open PSL File**.
2. In the dialog, choose the PSL file.
3. Click **Open**.

Or

1. Click the  icon in the toolbar.
2. Select **Open PSL File** in the drop down menu.
3. In the dialog, choose the PSL file.
4. Click **Open**.

If there are any parse errors in the PSL file, the errors are displayed in the **Log pane**.

To reopen the PSL file:

1. Click the  icon.
2. Select **Reopen PSL File** in the drop down window.

Setting the verification clock

To run verification on your trace you must specify the clock to use for the verification. The tool provides two options for clocks:

- Every cycle in the trace generates a clock event. The PSL assertions are evaluated at every single clock cycle.
- The clock is explicitly specified for the tool. The PSL assertions are evaluated only on cycles with the rising/falling or both edges of the specified clock.

The clock settings are set in the **Clock toolbar**:



To evaluate PSL assertions at every cycle:

1. Unselect the **Use non default clock** checkbox in the toolbar

To check assertions with the explicitly specified clock:

1. Select the **Use non default clock** checkbox in the toolbar.
2. Enter the name of the clock in the **Clock name** field in the toolbar.
3. Select **Rising clock edge** to evaluate PSL assertions at the rising edge of clock, **Falling clock edge** to evaluate PSL assertions at the falling edge of clock or **Both clock edges** to evaluate PSL assertions on both rising and falling edges of the clock.

Viewing and Selecting PSL Assertions

After your PSL file successfully opens you can see the list of vunit names in the **Vunits list pane** on the left the window.

To select vunit:

1. Click the vunit name.

You can see all the assertions of the selected vunit in the **Assertions text pane**.

To select more than one assertion in vunit:

1. Click the “+” on the left of the vunit name.
2. Hold the CTRL button and click the desired assertion numbers.

You can see all the selected assertions in the **Assertions text pane**.

Verifying the trace

Running the Verification

After opening the PSL and trace files, you can check whether your PSL assertions hold on the opened trace.



To run verification on the trace:

1. Select the name of the vunit to check. (Selecting some of the assertions in a vunit is the same as selecting the whole vunit for verification purposes.)
2. Select **Verification -> Run Verification**.



Or

2. Click the  icon in the toolbar.

Viewing the Verification Results

Verification result icon appears on the left hand side near the name of the verified vunit. You see the  icon if all the assertions in the vunit passed or  icon if any of the assertions in the vunit failed.

To navigate through the fail cycles of assertions:

1. Select the assertions to see the fail cycles. (Selecting the whole vunit is the same as selecting all of the assertions in it.)
2. Click the  icon to go to the next fail cycle of one of the selected assertions, or the  icon to go to the previous fail cycle of one of the selected assertions.

Manipulating the trace view pane


Adding and Removing Signals to/from the Trace View

When you open your trace file, by default none of the signals in the trace are shown in the trace view. To see the signals you must add them to the trace view.

To add signals to the trace view:

1. Select **Signals->Add Signals**.

Or

1. Click the  icon in the toolbar.
2. In the dialog, select the signals to add to the trace view.
3. Click **"Add"**.

To delete signals from the trace view:

1. Select the signals to remove.
2. Press the **"Delete"** key on your keyboard.

Or

2. Select **Signals->Remove Signals**.

Or

2. Click the  icon in the toolbar.

Moving Signals on the Trace View

You can choose the location of the signals in the trace view pane.

To move signals in the trace view:

1. Click the desired signal and hold down the (left) mouse button.
2. Move the mouse to the new signal location.
3. Release the mouse button.


Adding a Visual Separator

It may be more convenient to view the signals by creating a visual separation between different groups of signals.

To add a visual separator to the trace view:

1. Select **Signals->Add Separator**.

Or

1. Click the  icon in the toolbar.


Adding a Bus Sub-Range to the Trace View

You might want to see only a part of a bus in your trace. In this case, you can select only the bus sub-range to be shown in the trace view.

To add a bus sub-range to the trace view:

1. Select the desired bus.
2. Select **Signals-> Bus Sub-Range**.

Or

2. Click  icon in the toolbar.
3. In the dialog choose the sub-range of the to be added to the trace view.
4. Click "Add".

Flipping the Bus in the Trace View

You may wish to flip the bus in the trace view (thereby invert its MSB and LSB). The tool allows you to flip the bus for viewing purposes only: flipping the bus does not affect the verification of the trace.

To flip the bus in the trace view:

1. Select the bus signal in the trace view.
2. Select **Signals-> Flip Bus**.

Or

2. Click the  icon in the toolbar.


Combining Signals

The tool provides you an easy and convenient way to group multiple signals to a single bus. Signals are combined into a bus for viewing purposes only; verification results are not affected when signals are combined for viewing.

To combine signals to a bus in the trace view:

1. Select the signals you wish to combine to a single bus.
2. Select **Signals-> Combine Signals**.

Or

2. Click the  icon in the toolbar.

Change the Base of Buses

You may wish to change the base for displaying the value of the bus in the trace view. The supported bases are binary, octal, decimal, and hexadecimal. The default base of all buses is binary.

To convert base of bus in the trace view:

1. Select the signal whose base you wish to convert.
2. Select the base under **Signals-> Convert Base** menu.

Change Colours of Signals

You can change the colours of the signals shown in the trace view.

To change the colour of the signals in the trace view:

1. Select the desired signals in the trace view.
2. Select **Signals-> Signal Line Colour**.
3. In the dialog choose the new colour for the signals.
4. Click **OK**.

To set the default color for signals:

1. Select the desired signals in the trace view.
2. Select **Signals-> Default Line Colour**.

Selecting All Signals in the Trace View

The tool allows you to easily select all signals in the trace view. You may wish to use this feature when you wish to remove all signals from trace view pane.

To select all signals in the trace view:

1. Select **Edit-> Select All**.

Zooming the Trace View

The tool allows you to perform zoom in and zoom out actions on the trace, thereby changing the scale in which the trace is shown. You can also use the **Full Zoom** which scales the trace so you can see the whole trace in the trace view without scrolling

To zoom in the trace view:

1. Select **View-> Zoom In**.

Or

1. Click the  icon in the toolbar.

To zoom out the trace view:

1. Select **View-> Zoom Out**.

Or

1. Click the  icon in the toolbar.

To view the full trace in the trace view:

1. Select **View-> Full Zoom**.

Or

1. Click the  icon in the toolbar.

Navigating in the Trace View

You can navigate in the trace view by selecting a signal and jumping to the next or previous cycle at which the signal value changes. You can also search for a cycle that satisfies a simple Boolean expression with signals from the trace view.

To go to the next cycle where the signal value changes:

1. Select **View-> Next Change**.

Or

1. Click the  icon in the toolbar.

To go to the previous cycle where the signal value changes:

1. Select **View-> Previous Change**.

Or

1. Click the  icon in the toolbar.

To search for a cycle where the Boolean expression holds:

1. Select **View->Search ...**
2. In the dialog, compose a Boolean expression.
3. Click the “**Find next**” button to search forward for the cycle where the expression holds.

Or

3. Click the “**Find previous**” button to search backward for the cycle where expression holds.

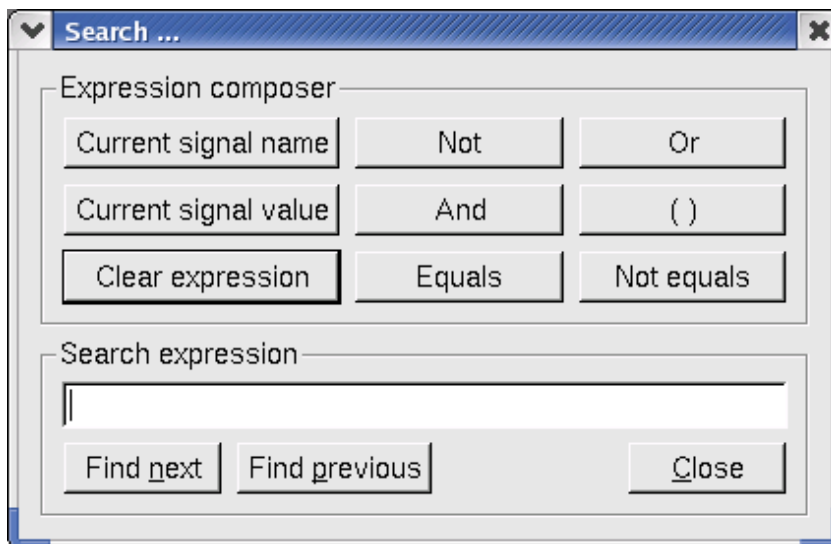


Figure 4: Dialog for the composition of simple Boolean expressions

4 Tool Properties

Feature List

	Present	Ref.
Mandatory features		
• Pointers to algorithms used	yes	Chapter 2
• List of target operating systems	yes	See below
• Explanation of coding standards	yes	See below
• Discussion of license issues	yes	See below
• User documentation, including documentation of user interface (command line switches) and imported/exported file formats	yes	See below
• Test suite	yes	See below
• Standard input language – PSL	yes	See [8]
○ Support for GDL and Verilog flavour	yes	
• Graphical User Interface	yes	Chapter 3
Desirable features		
• Restricted trace	No	
Nice-to-have features		
• Support for other flavours	No	
• Built-in editor for properties	No	

Table 2: Feature list

Target operating systems: Linux, Solaris, AIX

Coding standards: The tools are implemented in C and C++, according to the coding standards of IBM Haifa Research Laboratory.

License issues: A FlexLM license mechanism is included for the tool. The license can be purchased from IBM under a license agreement.

User documentation: The tool user guide is available in the distribution package of the tool under a license agreement. This deliverable includes relevant excerpts from the user guide.

Test suite: Information about the test suite is delivered, but not the test suite itself. The trace analysis tool simulation engine was tested on about 1000 PSL properties.

Work Summary

Starting Point

At the starting point, the executable model builder existed as part of the FoCs tool [6] and a VCD trace reader and trace viewer existed as part of the IBM RuleBase PE tool [7].

Work Performed

Simulation engine

A simulation engine module was developed.

Analysis Viewer

A GUI application for a trace analysis view and control was developed.

Tool

Interfaces between all the modules (existing and newly developed) were designed and implemented. All the modules were connected together to a standalone tool.

Supported PSL Subset

The subset of PSL supported by FoCs is described in Section 5 in [8]. Constructs listed in this section are supported, except for the constructs marked "RuleBase PE only".

5 References

1. D. Pidan, S. Keidar-Barner, D. Fisman, M. Moulin – Optimized algorithms for dynamic verification. PROSYD deliverable 3.2/5.
2. Y. Abarbanel, I. Beer, L. Gluhovsky, S. Keidar, Y. Wolfsthal. FoCs – Automatic generation of simulation checkers from formal specifications. In International Conference on Computer Aided Verification, volume 1855 of Lecture notes in Computer Science. Springer-Verlag, 2000.
3. D. Pidan, M. Shamis – Property-based automatic generation of simulation monitors for digital designs. PROSYD deliverable 3.2/11
4. The Institute of Electrical and Electronics Engineers, Inc. IEEE Standard Verilog Hardware Description Language, 2001.
5. Accellera. Accellera Property Language Reference Manual. In <http://www.eda.org/vfv/docs/PSL-v1.1.pdf>, June 2004
6. FoCs User Guide. Available as a part of the FoCs product
7. RuleBase PE User Guide. Available as a part of the RuleBase PE product
8. Porting of IBM tools to support Accelera-Standard version of PSL. PROSYD deliverable 3.3/2