Eclipse/VDT – User's Manual

Contents

Links To Main External Components

Sun JDK 1.5.0

http://java.sun.com/j2se/1.5.0/download.jsp

Eclipse

http://eclipse.org

Eclipse 3.2M4, Linux x86, GTK look and feel:

http://download.eclipse.org/eclipse/downloads/drops/S-3.2M4-200512151506/download.php?dropFile=eclipse-SDK-3.2M4-linux-gtk.tar.gz

Eclipse 3.2M4, Linux x86, Motif look and feel:

http://download.eclipse.org/eclipse/downloads/drops/S-3.2M4-200512151506/download.php?dropFile=eclipse-SDK-3.2M4-linux-motif.tar.gz

Eclipse 3.2M4, Win32:

http://download.eclipse.org/eclipse/downloads/drops/S-3.2M4-200512151506/download.php?dropFile=eclipse-SDK-3.2M4-win32.zip

Plug-in Verilog Editor

http://veditor.sourceforge.net/
http://sourceforge.net/project/showfiles.php?group_id=103963&package_id=111746&release_id=410197

CVer

http://www.pragmatic-c.com/gpl-cver/downloads/gplcver-2.11a.src.tar.bz2
http://www.pragmatic-c.com/gpl-cver/downloads/gplcver-2.11a.linux.bin.tar.bz2

VDT Installation and Launch Order

Convention: In this document the Eclipse/VDT plug-in is called VDT, for shortness.

1) Download and install the Sun JDK 1.5.0_x package.

Attention: you need to install JDK, not JRE. The JDK version must be not earlier than 1.5.0. VDT cannot work with JDK 1.4.x и 1.3.x.

In OS Linux, the system `PATH` variable must be set to include the `bin` directory installed by JDK 1.5.0. For example, if the Sun JDK 1.5.0_06 package had been installed into `/usr/java/jdk1.5.0_06` directory, the command will look like:

```
export PATH=/usr/java/jdk1.5.0_06/bin:$PATH
```

2) Download the Eclipse 3.2M4 package.

Attention: VDT is being developed and tested with this version 3.2M4 of Eclipse. Other versions may work not with Eclipse/VDT, or work incorrectly.

Unpack the VDT archive somewhere on the local disk. In Linux, for example, it may be the `/home/user` directory:

```
cd /home/user
tar xvzf ~/downloads/eclipse-SDK-3.2M4-linux-gtk.tar.gz
```

or in Win32 – the `D:\user` directory:

```
cd D:\user\
unzip D:\downloads\eclipse-SDK-3.2M4-win32.zip
```

The new `eclipse` subdirectory will appear.

3) Take the VDT package and unpack it into the Eclipse installation directory. For example, in Linux with Eclipse installed in `/home/user/eclipse` it will be:

```
cd /home/user/eclipse
unzip ~/downloads/com.elphel.vdt_1.1.0.zip
```

or, for Win32 with Eclipse in `D:\user\eclipse` :

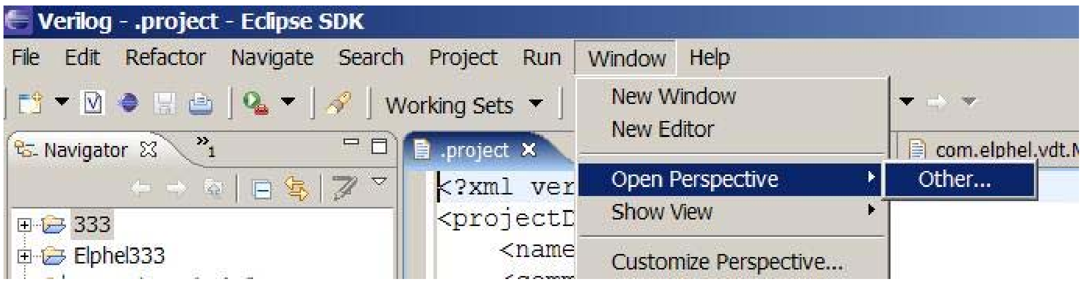
```
cd D:\user\eclipse
unzip D:\downloads\com.elphel.vdt_1.1.0.zip
```

4) Run the Eclipse (the executable file is `eclipse` or `eclipse.exe` in the root of the installation directory).

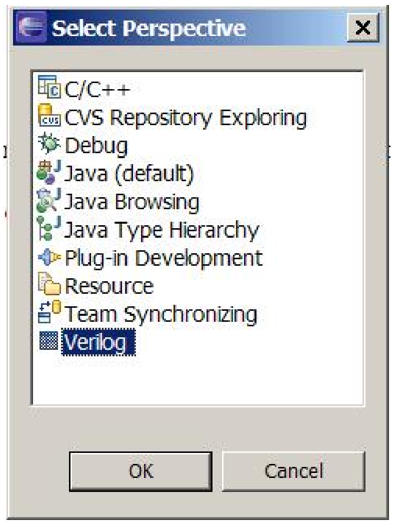
The Verilog Perspective

All components provided for Eclipse by the VDT plugin to do Verilog programs development are combined into the perspective called Verilog. A ***perspective*** in Eclipse is a set of visible windows, control facilities and their arrangement within the Eclipse screen at the moment. The perspective allows to select and arrange for usage only those components from the overall set of components provided by various plugins, that are needed for solving the current task.

On start, Eclipse opens the last used perspective. To open the Verilog perspective, choose **Window** → **Open Perspective** → **Other** in the main menu:



Then choose **Verilog** in the pop-up list of perspectives:



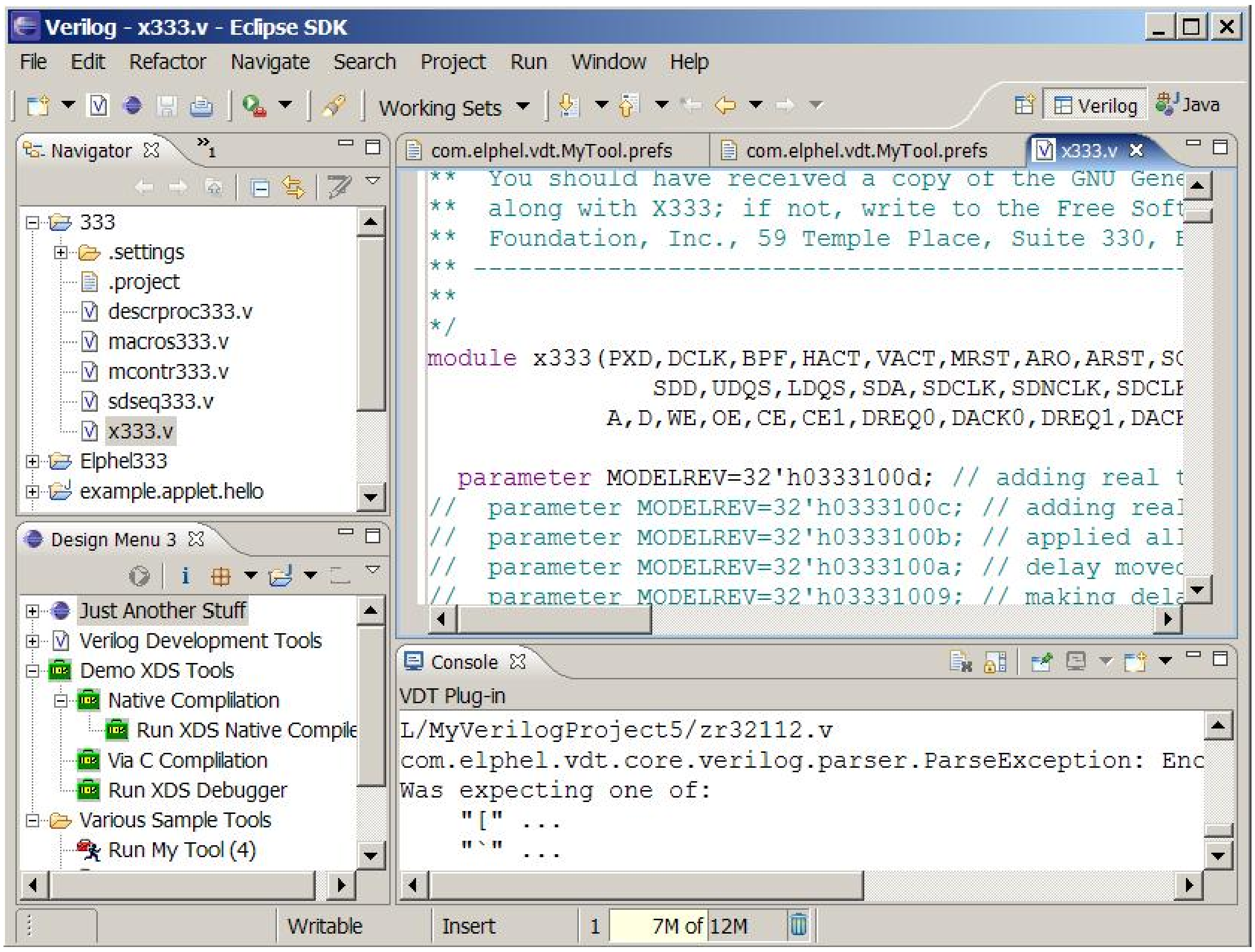
The Verilog perspective consists of four windows called ***views***: Navigator, Verilog Navigator, Design Menu, Console, and also the biggest window containing text editors windows.

The **Navigator** view serves for navigation through project files and directories and for performing operations with them. The elements of the view are organized in a tree-like structure. The root nodes are projects; they may contain any number of files and directories. Files and directories may be added, deleted, moved or renamed in this window – right-click an element and choose the proper action in the pop-up menu.

The **Verilog Navigator** view displays a VDT project as a list of Verilog modules, rather than source files. This window is useful, for example, when a single Verilog module must be chosen to pass it to a utility. By default, the Verilog Navigator window is hidden behind the Navigator window; to switch to it, click the tab with its name or the "**>>**" icon, if the window is not wide enough to show all tabs.

The **Design Menu** view displays a hierarchical menu of ***tools*** (which stand for calls of external utilities) and is used to configure and launch them. The title of the Design Menu window depends on actual project.

The **Console** view accepts standard output streams of the called external utilities. A special console titled **VDT Plug-in** accepts messages of the VDT plugin itself.



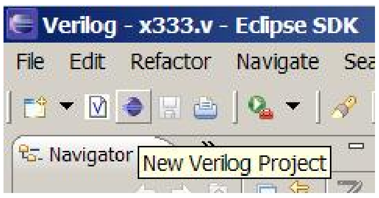
```
**  You should have received a copy of the GNU Gene
**  along with X333; if not, write to the Free Soft
**  Foundation, Inc., 59 Temple Place, Suite 330, E
** ---------------------------------------------
**
*/
module x333(PXD,DCLK,BPF,HACT,VACT,MRST,ARO,ARST,SC
                SDD,UDQS,LDQS,SDA,SDCLK,SDNCLK,SDCLK
            A,D,WE,OE,CE,CE1,DREQ0,DACK0,DREQ1,DACK

  parameter MODELREV=32'h0333100d; // adding real t
//  parameter MODELREV=32'h0333100c; // adding real
//  parameter MODELREV=32'h0333100b; // applied all
//  parameter MODELREV=32'h0333100a; // delay moved
//  parameter MODELREV=32'h03331009; // making dela
```

```
L/MyVerilogProject5/zr32112.v
com.elphel.vdt.core.verilog.parser.ParseException: Enc
Was expecting one of:
    "[" ...
    "`" ...
```

The central area of the Eclipse window is the texts view window, where one or more files may be opened to edit. Syntax of texts may be hilighted according to their file types. E.g., files with extension `.v` are recognized as Verilog programs and hilighted according to Verilog syntax (if the Verilog editor plugin is installed).

Besides these four windows, the following buttons are added to the tools panel ot the Verilog perspective:

New Verilog Project;
New Verilog Module.


Verilog - x333.v - Eclipse SDK
File Edit Refactor Navigate Sea
Navigator
New Verilog Project

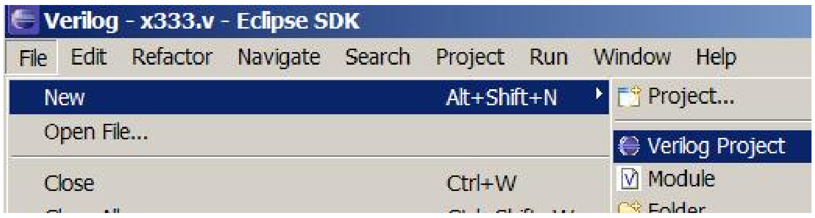
Project Management

All capacities of the VDT plug-in provided for external utilities setup and launch are available only when working with VDT projects.

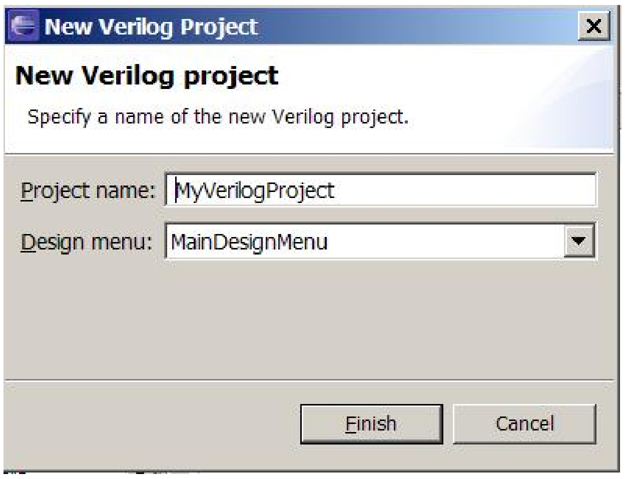
Creating a New Project

A new VDT project may be created with New Verilog Project wizard.

1) To create a new VDT project, press the **New Verilog Project** button on the tools panel or choose **File** → **New** → **Verilog Project** in the main menu:


Verilog - x333.v - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Project...
Verilog Project
Module
Folder

2) In the **New Verilog Project** dialog, enter a unique project name in the **Project Name** field and choose a proper menu for this project in the **Design Menu** list (that is, a menu that contains necessary tools for this project). The name of the chosen menu will be then shown in the Design Menu window title.


New Verilog Project

New Verilog project

Specify a name of the new Verilog project.

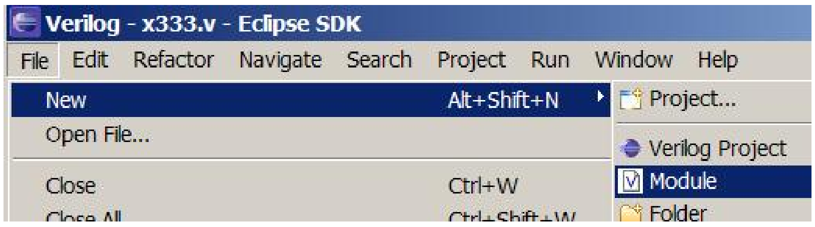
Project name: MyVerilogProject

Design menu: MainDesignMenu

Creating a New Verilog Module

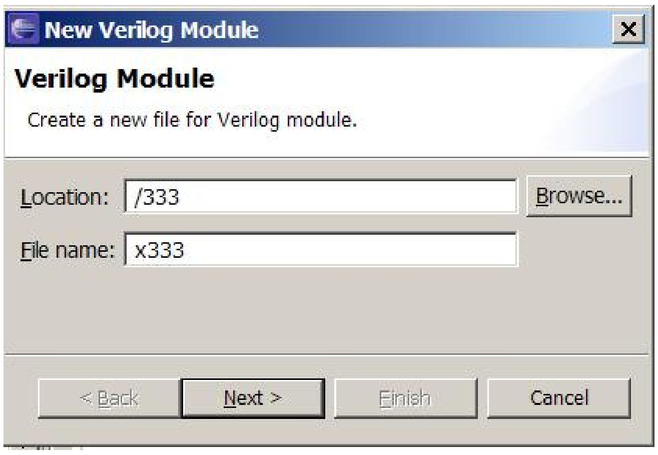
A new Verilog module may be created with New Verilog Module wizard.

1) Press the **New Verilog Module** button on the tools panel or choose **File** → **New** → **Module** in the main menu.



2) In the **New Verilog Module** dialog, field **Location**, type in a path to the project directory where, if necessary, the Verilog file will be created. By default, this field is filled with the current project name selected in the Navigator window.

3) In the **File name** field, type in a name of the file in which the new module will be put. (extension ".v" may be omitted).

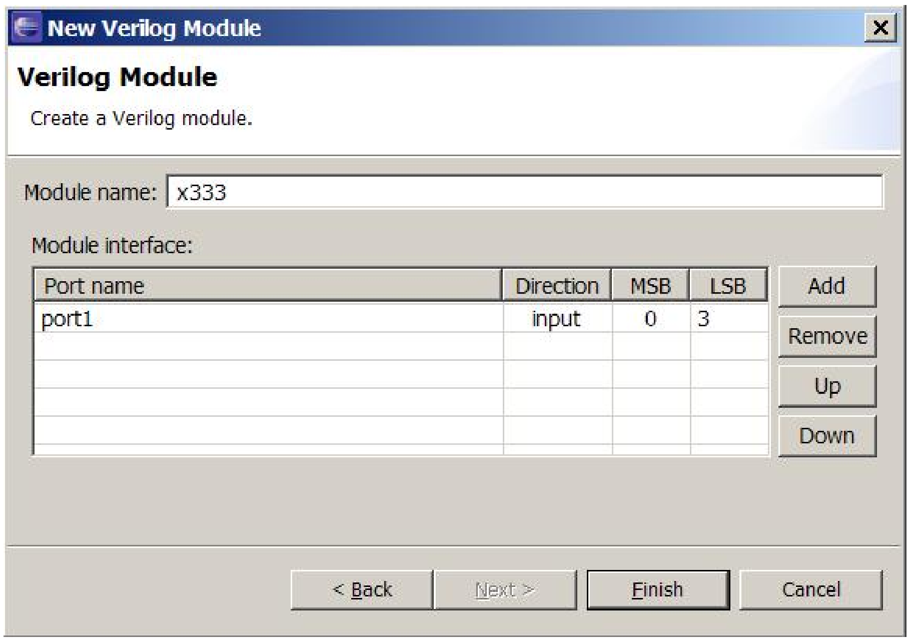


Location: /333

File name: x333

4) Press **Next** to proceed to the next page.

5) On the next page, enter a module name and specify its interface: number, names and properties of its interface ports. To add, remove or change the order of interface ports, use buttons **Add**, **Remove**, **Up**, **Down** at the left of the list of ports.



Module name: x333

Module interface:

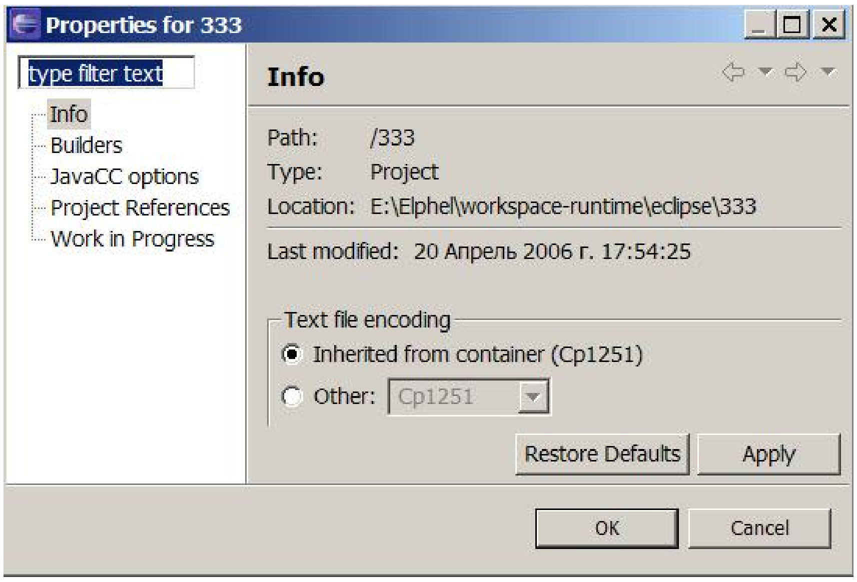
Port name	Direction	MSB	LSB
port1	input	0	3

6) When the interface specification is over, press the **Finish** button. In the chosen project directory a new file with the specified name will be created, and a module template with the specified interface will be inserted in it.

Importing New Modules Into A Project

To import existing external modules into a project, do the following:

1) Find the project's location on a disk (in the Navigator window, right-click on the project name, choose **Properties** → **Info** in the context menu and see the value in the **Location** field):



Properties for 333

Info

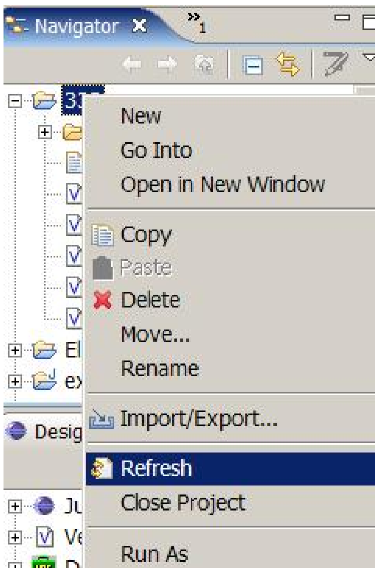
Path: /333
Type: Project
Location: E:\Elphel\workspace-runtime\eclipse\333

Last modified: 20 Апрель 2006 г. 17:54:25

Text file encoding
(●) Inherited from container (Cp1251)
( ) Other: Cp1251

2) Copy external files and directories into the project directory.

3) In the Navigator dialog, right-click on the project name and choose **Refresh** in the context menu of the project:

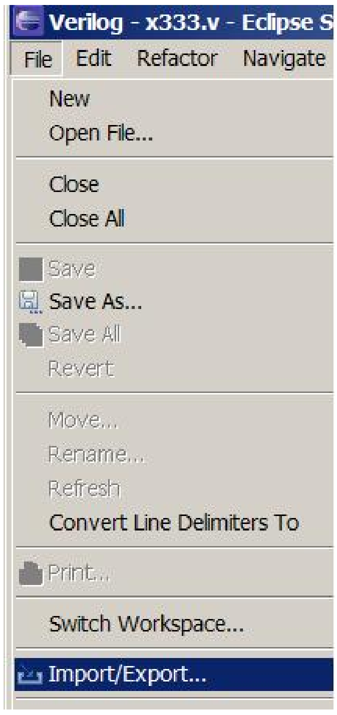


Exporting VDT Projects

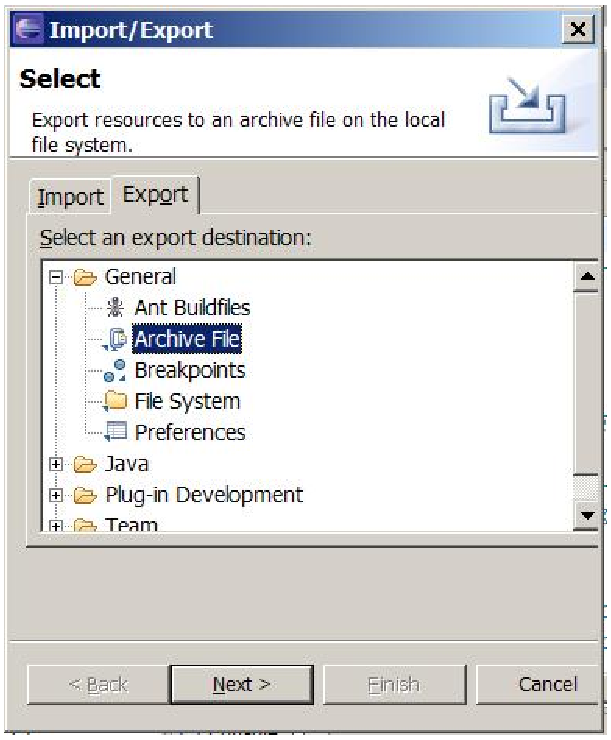
Export of VDT projects may be needed, for example, to port them to another computer.

A VDT project may be exported with Eclipse Export Wizard:

1) Choose **File** → **Import/Export** in the main menu:



2) In the pop-up dialog, go to the **Export** tab, choose **General** → **Archive File** in the menu and press **Next**:


Import/Export

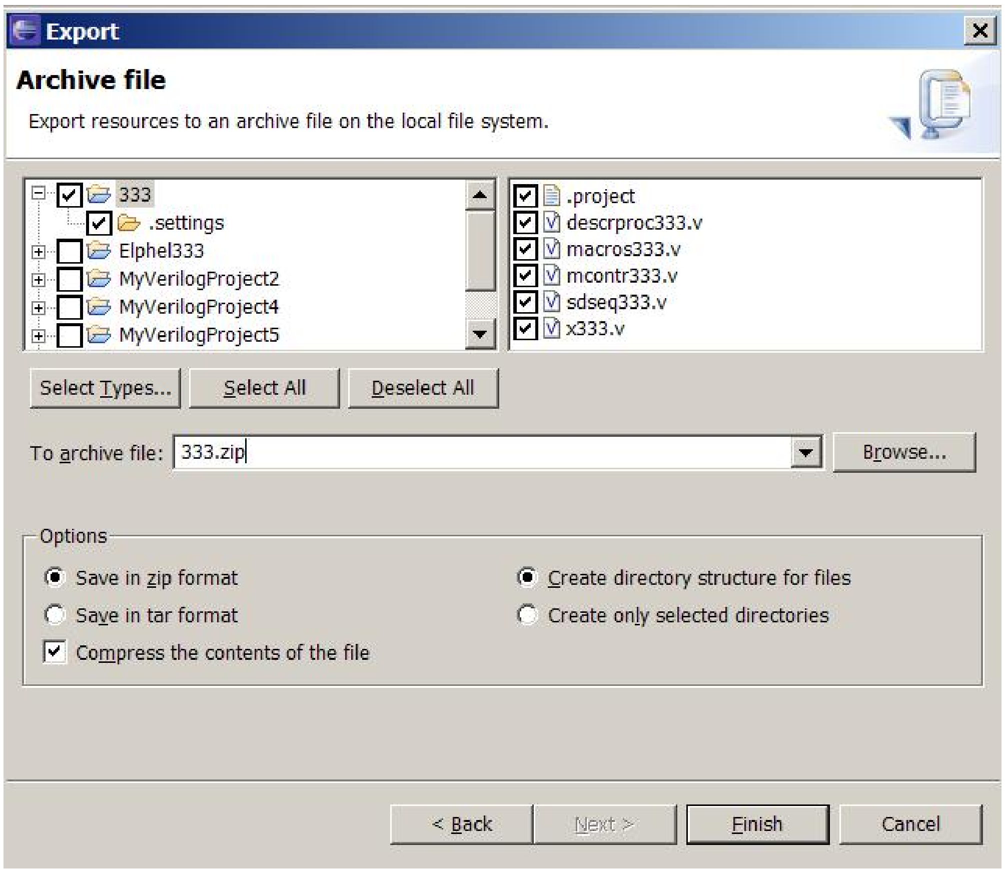
Select

Export resources to an archive file on the local file system.

3) In the left panel of the Export window choose a project and in the right panel select files to be exported. Always add file `.project` in the list of exported files!

4) In the left panel, select the `.setting` directory of the exported project and all its contents.

5) In the **To archive file** field, input an archive name.



Export

Archive file

Export resources to an archive file on the local file system.

- [✓] 333
  - [✓] .settings
- [ ] Elphel333
- [ ] MyVerilogProject2
- [ ] MyVerilogProject4
- [ ] MyVerilogProject5

- [✓] .project
- [✓] descrproc333.v
- [✓] macros333.v
- [✓] mcontr333.v
- [✓] sdseq333.v
- [✓] x333.v

To archive file: 333.zip

Options

- [✓ Save in zip format] Save in tar format
- [✓] Compress the contents of the file
- [✓ Create directory structure for files] Create only selected directories

6) Press **Finish**.

A more detailed description of the export process may be found at the link:

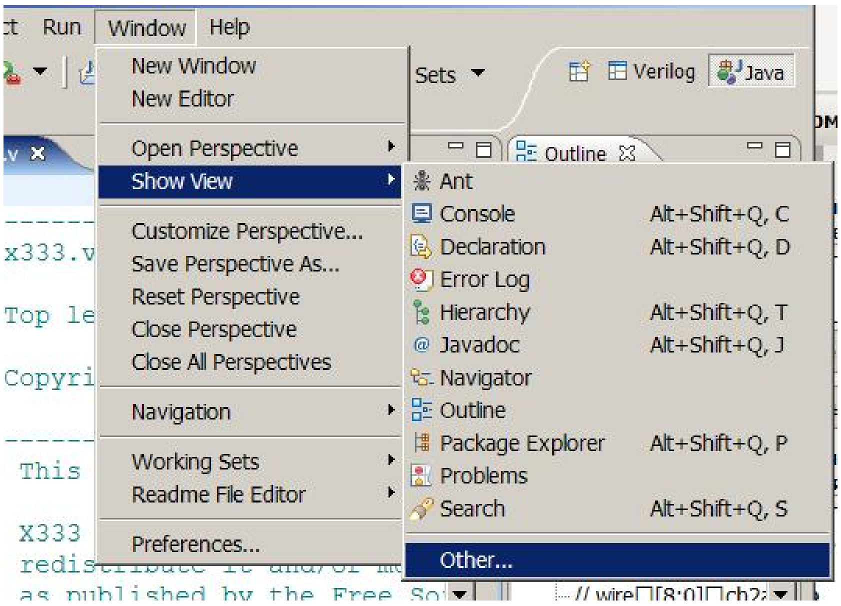
http://www.cs.laurentian.ca/badams/c1047/eclipse-tutorials/export-tutorial.html

Note: The archive created this way must be then imported not as an Archive File, but as **Existing Project into Workspace**.

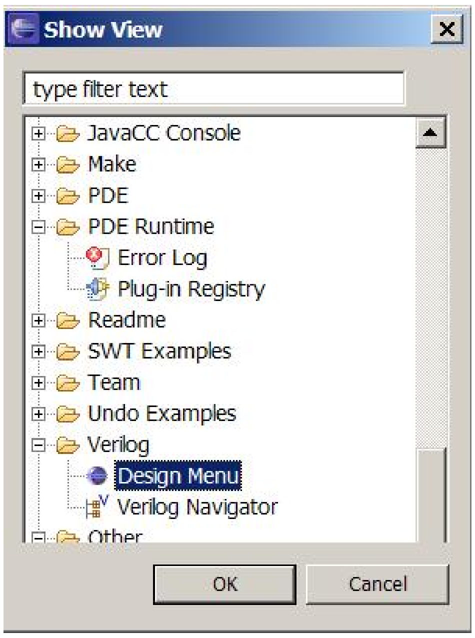
The same result may be achieved "manually": just copy a project directory. The project location may be found in the same way: in the Navigator window right-click on the project name, choose **Properties** → **Info** in the context menu and see the value in the **Location** field.

VDT Graphical Components

Basic graphical components of the VDT plug-in are windows Design Menu, Console and Verilog Navigator. By default, they all are available in the Verilog perspective, but may be opened in any other perspective, as well. To do that, choose **Window** → **Show View** → **Other** in the main menu:



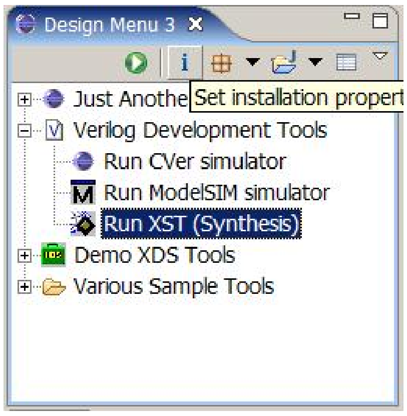
In the pop-up dialog, choose **Verilog** and the name of a window you need:



Design Menu Window

Design Menu is the main window to work with external utilities. It contains a hierarchical list of tools (utilities calls) and is purposed for their setup and launch. A list of available tools depends on a project and is chosen when a VDT project is created. For non-VDT projects the list of tools is always empty.

The Design Menu window title contains the tools menu name chosen at project creation. This title may vary from project to project.



The local tools panel of the Design Menu window contains buttons for the following actions:

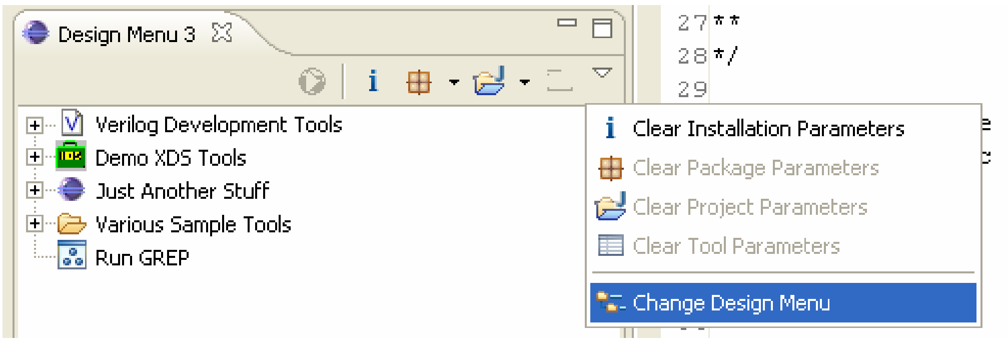
▶	Run a tool
i	Set up installation parameters
▦	Set up parameters of packages
📂	Set up parameters of projects
▤	Set up parameters of tools

Setups of packages and projects are implemented as pop-up lists containing the names of all packages or projects for which setup of parameters is possible (see "*Tools Setup*").

The contents of the tools panel is duplicated in the context menu available by right-clicking in the window.

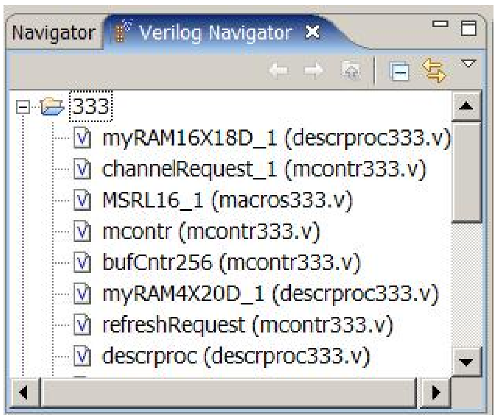
A set of tools shown in Design Menu, their hierarchical structure, call modes, parameters sets of utilities, packages and projects are all defined by the ***TSL specification*** (see *"Editing The TSL Specification Of Configurable Parameters"*).

For any project at any moment another menu may be chosen – press the white arrow on the local tools panel of the Design Menu window (the rightmost icon) and select a new menu in the pop-up dialog.

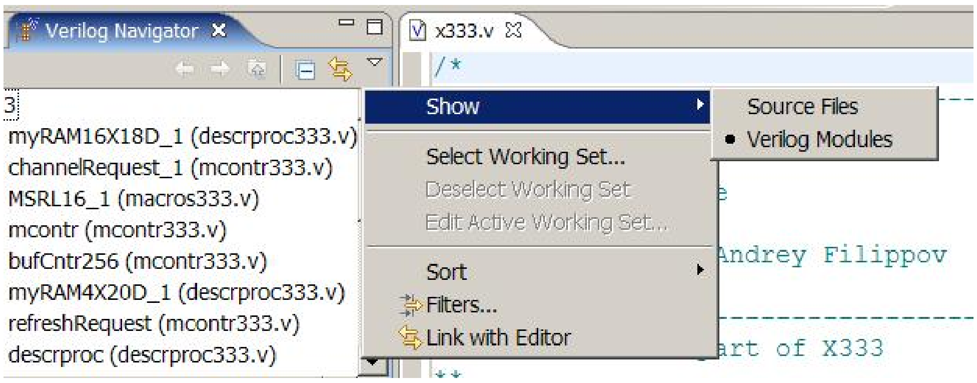


Verilog Navigator Window

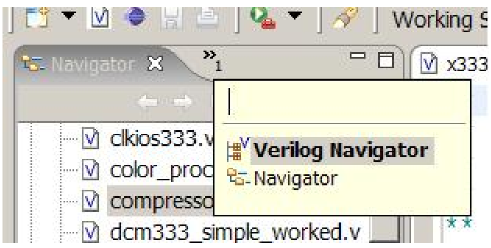
The Verilog Navigator window displays VDT projects as a list of all Verilog modules in a project rather than a list of source files. After each Verilog module name, the name of its containing source file is given in parentheses:


Navigator
Verilog Navigator
333
myRAM16X18D_1 (descrproc333.v)
channelRequest_1 (mcontr333.v)
MSRL16_1 (macros333.v)
mcontr (mcontr333.v)
bufCntr256 (mcontr333.v)
myRAM4X20D_1 (descrproc333.v)
refreshRequest (mcontr333.v)
descrproc (descrproc333.v)

Non-VDT projects are shown as a list of files and directories, exactly as they are in the Navigator window. Moreover, there is a mode of the Verilog Navigator window to show VDT projects as files and directories, too. To switch the mode, press the white arrow at the right of the local tools panel of the Verilog Navigator window and choose a mode in the **Show** item:



By default, the Verilog Navigator window is hidden behind the Navigator window. To show it, press the “**>>**” sign near the window title:



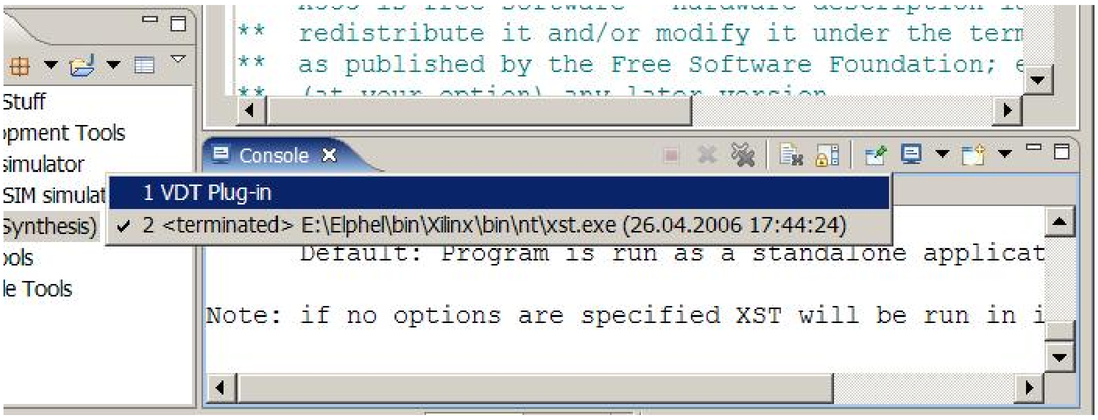
Console Window

The Console window accepts standard output streams of external utilities launched. In fact, it supports two consoles.

One console accepts output of external utility. Its contents is preserved until the next utility is run.

Another console, named **VDT Plug-in**, accepts messages from the VDT plug-in itself. This may be, for instance, error messages from Verilog files parsing, which is performed to extract the list of project modules to be shown in the Verilog Navigator window.

To switch between console views, right-click the black arrow right to the "monitor" icon in the local tools panel of the Console window and choose a console in the pop-up list:



```
**  redistribute it and/or modify it under the term
**  as published by the Free Software Foundation; e
```

```
     Default: Program is run as a standalone applicat

Note: if no options are specified XST will be run in i
```

Tools Setup

Tools are calls of utilities preset for various operation modes. Tools **settings** are assignments of values to utility parameters valus by means of which a user can control how the utility works.

Each utility parameter is described by the following aspects:

- a type and a values domain;
- input format, as represented and entered in a setup dialog;
- output format, as passed to a utility.

Sets of parameters and all their aspects are completely specified in the TSL specification, in definitions of respective TSL ***contexts***. This work, called an *installation setup*, is carried out preliminarily by an environment setup programmer who thus defines a set of available tools. An end-user can only perform *operational setup* of tools, that is, to configure their parameters to some actual values defining a utility's operation mode.

Operational settings fall into four levels:

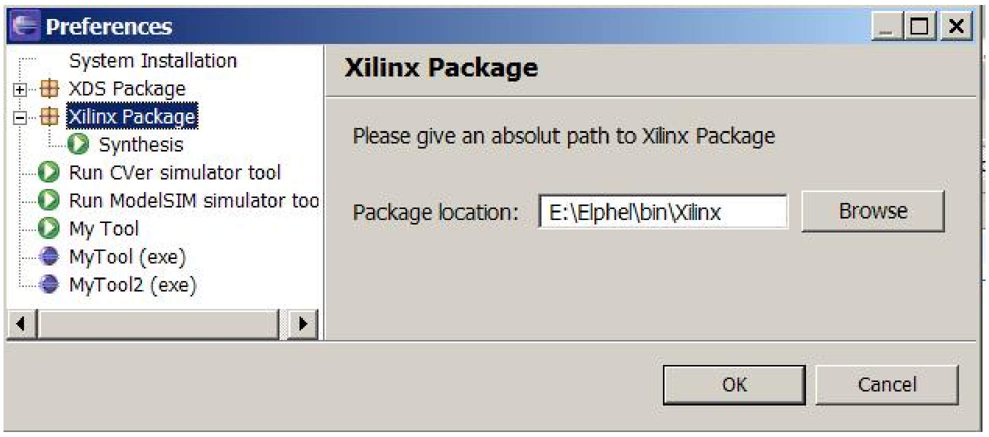
- installation settings – all that is necessary to make a utility run;
- package-level settings (correspond to a configuartion file);
- project-level settings (correspond to a project file);
- call-level settings (correspond to utility's command line or file(s)).

Installation settings include some predefined parameters not specified in the TSL specification (e.g. a path to utility's or package's directory).

Installation Settings

Installation settings define paths to packages and to executable files of utilities. Also included here are parameters defined in the TSL **installation** context. Installation settings are common for all projects; they are saved withing the plug-in's metadata and are not exported with projects.

The installation setup dialog is opened on pressing button i in the local tools panel of Design Menu. The left panel of the dialog shows a list of packages and tools, the right panel displays a list of parameters for the element selected in the first list.


Preferences

System Installation
XDS Package
Xilinx Package
Synthesis
Run CVer simulator tool
Run ModelSIM simulator too
My Tool
MyTool (exe)
MyTool2 (exe)

Xilinx Package

Please give an absolut path to Xilinx Package

Package location: E:\Elphel\bin\Xilinx

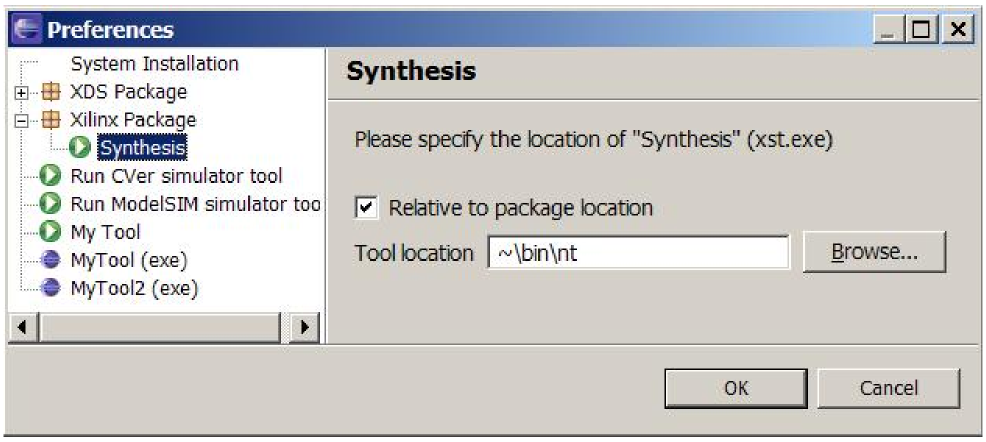
The first element in the left list always stands for parameters defined in the unique TSL **installation** context. Its label and the contents of its setup dialog are entirely user-defined, by the TSL specification (see document *"Eclipse/ExDT – Tools To Set Up Custom Interactive Development Environments"*).

Next in the list go packages, each expanding in a list of tools related to this package.

After packages go stand-alone tools not included in any package.

For a package, only one parameter is settable – a path to the package directory. It may be used to setup utilities belonging to this package which need to know its location. The entered directory must at least exist.

For a tool, a path to its executable file (utility) must be set. If the utility belongs to a package, this may be either an absolute path to an executable file, or a path relative to the utility's package location:



Synthesis

Please specify the location of "Synthesis" (xst.exe)

[✓] Relative to package location

Tool location: ~\bin\nt

When a relative path is chosen, the package path, which is a prefix of the absolute path returned by "Browse", is automatically replaced by ~.

For a utility which is not in any package, only an absolute path to an executable file may be enetered.

Package-Level Tool Settings

First of all, it should be clear here, that *package-level tool settings* are relevant to utilities in packages, but not to packages themselves! (Since packages cannot be executed as tools).

Package-level are tool settings which are typically specified in the **configuration file** of the utility (if provided), e.g.:

- properties of package installation on the given computer, which are invariant for all projects and call modes of this utility;
- settings of default values for utility parameters for the project; these values may be then redefined at project and call levels.

Each configurable parameter is relevant only for the utility for which it is defined.

In VDT, there is the only setup context for each package (defined in TSL). This means that:

- all package-level parameters of all utilities of this package are joined together;
- they all are configured together in one setup dialog;
- all utilities' configuration files containing these settings are generated on exit from this dialog.

If the design menu contains calls of utilities from several packages, each of these packages may need to be configured separately.

Package-level settings are common for all projects; they are saved in the plug-in's metadata and are not exported with projects.

To invoke a dialog to set up package parameters, press button in the local tools panel of the Design Menu window. A list of packages used in the current tools menu will pop up. The dialog looks like a set of tabs containing groups of parameters according to the TSL specification.

Project-Level Tool Settings

First of all, it is necessary to distinguish between *Eclipse project settings* and *project-level tool settings*.

Eclipse project settings include what Eclipse itself knows about a user's project: a list of project files, a design menu, a list of available utilities and packages, configuration of windows, storable values of all parameters, etc. These settings are saved in Eclipse metadata for the current project; they are not directly available neither to a user, nor to utilities.

Project-level tool settings include what a certain utility knows about the user's project. They are typically specified in a **project** file of the utility (if provided), e.g.:

- parameters settings, specific for the given project and invariant for all calls of the utility within this project;
- settings of project-specific default values for utility parameters (including redefinitions of default settings set at package level); these values may be then redefined at call level.

Each configurable project-level parameter is relevant only for the utility for which it is defined.

In VDT, there may be several *setup contexts* for a project (defined in TSL, where they are called **project templates** to emphasize absence of one-to-one correspondence with Eclipse projects). This means that:

- joining project-level parameters in a context is free: a context may contain parameters either of only one utility or of several ones, these utilities may also be either from one package or from several ones;
- each context is configured in a separate setup dialog;
- generation of each utility's project file is associated with some of these contexts and is done on exit from its respective setup dialog.

Project-level tool settings are individual for each Eclipse project; they are saved in the `.settings` subdirectory of the current project and are exported together with it.

To invoke a dialog to set project parameters, press button in the local tools panel of the Design Menu window. A list of project templates used in the current tools menu will pop up. The dialog looks like a set of tabs containing groups of parameters according to the TSL specification.

Call-Level Tool Settings

Utilities are called from the tools menu. Each menu element is associated with one tool (i.e. a call of some utility) and may be set up individually.

Call-level are tool settings which are passed to a utility directly when calling it besides what is already written in configuration and project files (if they exist), e.g.:

- parameters settings which may be specified only in the command line or in a command file;
- parameters settings specific for this call (and thus characterizing the operation mode of the tool);
- call-specific redefinitions of default package-level and project-level parameters.

In VDT, there is the only *setup context* for each tool. This means that:

- all call-level parameters of this tool are kept together;
- they are configured together in one setup dialog;
- the utility's command line and command files are generated at the utility call.

Call-level tool settings are individual for each Eclipse project; they are saved in the `.settings` subdirectory of the current project and are exported together with it.

To invoke a dialog to set tool parameters, press button in the local tools panel of the Design Menu window. The dialog looks like a set of tabs containing groups of parameters according to the TSL specification.

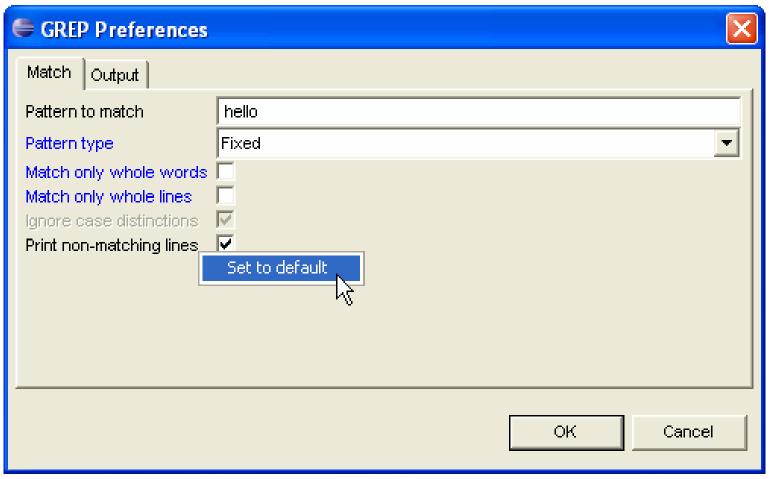
Setup Dialog Appearance

The structure of all setup dialogs for all contexts is basically the same.

Tabs in the dialog correspond to parameters groups defined in TSL (see tabs "Match" and "Output" on the sample picture below).

Each line corresponds to one parameter and contains an explanatory text (label) and an input field whose kind depends on a parameter type.

Normally, all texts are colored black. Read-only parameters cannot be modified and are colored gray (both label and value). If a parameter is in the "default" state, it's label is colored blue. When its value is modified, the label becomes black again. To reset the modified value to the default value, right-click the parameter and select "Set to default" in the pop-up context menu.



Pattern to match: hello
Pattern type: Fixed
Match only whole words ☐
Match only whole lines ☐
Ignore case distinctions ✓
Print non-matching lines ✓
Set to default

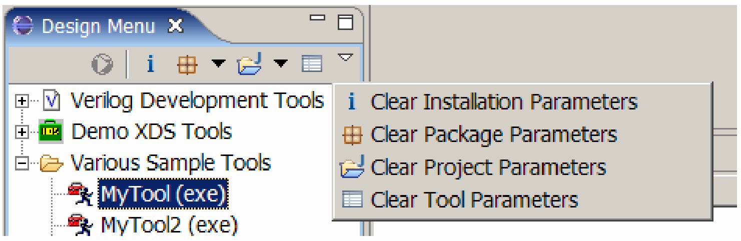
Is there any difference between a parameter's being set to the "default" state and being set to the value just equal to the default one? Yes, even two:

- any explicit value of the parameter will not change until it is explicitly modified, while the "default" value may be inherited from some parent context, and it will change implicitly as soon as it is modified at the parent;
- the "default" value sometimes may be not passed to a utility at all, and there may be a semantical difference between passing any explicit value and passing no value (this may help to set tri-state logic for flags with values "yes", "no" and "default").

Clearing Settings

Sometimes saved values of parameters need be discarded – e.g., if they become invalid after a parameter's format or a value domain have changed in the TSL specification.

To clear current settings of parameters, use a special menu that pops up on pressing the white arrow on the local tools panel in the Design Menu window (the rightmost icon):



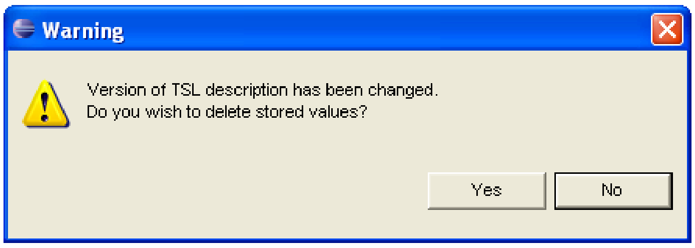
The Clear Package Parameters and Clear Project Parameters items discard saved values only for packages and projects available in the current Design Menu. The Clear Tool Parameters item discards saved values only for the selected tool.

Versions Control

It is clear that saved parameters values must be always consistent with their TSL definitions. For example, if some string parameter value is saved, then at load time the parameter must also be of the same string type (including length and all attributes).

But what if the TSL specification of the tool has changed and the parameter became of boolean type? In this case the saved value is inconsistent with the new type and an error will occur at load time. In other cases (e.g. when both old and new types are numeric) the old paramater's value may be formally correct with the new type but incorrect semantically.

In TSL, a mechanism is provided for a user to set the version of a TSL file. When any settings are saved, the file versions are also saved. When loading the settings, the saved version is compared with the actual one. In case of versions inconsistency, a message is reported with a proposal to discard all older settings to avoid possible collisions.


Version of TSL description has been changed.
Do you wish to delete stored values?

If a user declines this proposal, all saved settings are treated as correct and all potential errors are left up to him.

Launching Tools

To launch a tool:

1) In the Navigator window select a file (or in the Verilog Navigator select a module), for which a tool will be launched.

2) In the Design Menu window double-click a tool to be called. If the tool is already selected in Design Menu, it may be called by pressing the **Run** button in the local tools panel of this window.

If the executable file associated with the utility cannot be found (installation setup had not beed performed), the directory request dialog will be shown.

Editing The TSL Specification Of Configurable Parameters

The contents and appearance of menus and setup dialogs and also calls of external utilities are completely defined in the TSL specification that dynamically configures Eclipse/VDT. This TSL specification is stored in a separate directory as a set of XML files.

The Tools Specification Language (TSL) is described in a separate document.

In version 1.0.2, the TSL specification directory is located inside the Eclipse installation directory and its full path is `plugins/com.elphel.vdt_1.1.0/tools`. To modify the specification, edit its files directly by any means. In future it is supposed to make a user copy of this directory for edition, and also to develop a tool with convenient graphic interface to make additions and corrections of the TSL specification.