Rule Checker User Guide

ZamiaCad

Table of distribution

CNES	Name and function	Contact
	Gabriel LIABEUF Project Leader	gabriel.liabeuf@cnes.fr
	Florent MANNI Technical Responsible	florent.manni@cnes.fr
ALTRAN	**Name and function**	**Contact**
	Christophe NIESNER Consultant	christophe.niesner@altran.com
	Arnaud Daniel Project Leader	arnaud.daniel@altran.com +33 6 29 68 27 52

Revision History

Date	Version	Description	Author
24 July 2015	1.0	Document Creation	Arnaud DANIEL Clément LAMBERT
03 august 2015	1.1	Update with version of Java	Arnaud DANIEL Clément LAMBERT
03 september 2015	1.2	Update with rule checker GUI and LOG ROOT definition	Arnaud DANIEL Christophe NIESNER
21 september 2015	1.3	Addition of sections “Launch Tools” and “Launch Rules	Arnaud DANIEL

		Verification" Rewording of Configure the rule checker" content to clarify rc_* files usage. Use of complete path in hypertext links. Addition of Sources/Libraries sections in Import project.	Christophe NIESNER
04 december 2015	2.1	Update with WP-CLK and WP-RST.	Arnaud DANIEL Christophe NIESNER
11 march 2016	V3.0	Addition of features covering WP-PROC and WP-CDC	Arnaud DANIEL Christophe NIESNER
11 April 2016	V3.1	Addition of Eclipse plug in	Antoine BAILLY
01 june 2016	V3.2	Final Release	Arnaud DANIEL

Table of content

A. Scope

1. Identification

This document aims to give information about how to use Rule Checher add-on of ZamiaCad tool.

B. Mentioned Document

1. Application documents

Index	Title	Reference	Date
[A1]	DESIGN AND VHDL HANDBOOK FOR VLSI DEVELOPMENT STANDARD Edition	handbook_STD.xml	4 March 2015
[A2]	DESIGN AND VHDL HANDBOOK FOR VLSI DEVELOPMENT CNES Edition	handbook_CNE.xml	4 March 2015
[A3]	Plasma project archive	plasma.zip archive to be downloaded from http://zamiacad.sourceforge.net/web/?q=tutorial	November 2010

2. Reference documents

Index	Title	Reference	Date
[R1]	ZamiaCAD tutorial	zamiaCAD_0.10_tutorial.pdf document to be downloaded from http://zamiacad.sourceforge.net/web/?q=tutorial	December 2010
[R2]	ZamiaCad Rule Checker Features	zamiacad_rule_checker_features_decription_v3.2.pdf	June 2016
[R3]	Handbook UserGuide	CNES_TOOL_VHDL_UserGuide_v1.1.pptx	12 March 2015

C. Project objectives

ZamiaCad is a software tool used to help VHDL language users.

Rule Checker add-on has been developed in order to improve the way VHDL code is written and to reduce the time spent while performing code review.

Priority is given to code review.

Most VHDL editors are often not free and just provide syntactic highlighting.

The purpose of ZamiaCad Rule Checker is to cover these limitations and to go further by providing a tool that can be helpful for several VHDL user profiles: project managers, quality supervisors, reviewers, peers, designers.

ZamiaCad Rule Checker is licensed under the GNU Global Public License v3. The GNU General Public License is a free, copyleft license for software and other kinds of works.[1]

[1] See http://opensource.org/licenses/gpl-3.0.html

D. Preamble

ZamiaCAD tutorial is available at http://zamiacad.sourceforge.net/web/?q=tutorial. As an example, Zamia uses a public domain MIPS CPU design from OpenCores.org called **plasma**.

Plasma project is used as reference design in this guide. It can be downloaded from the zamiaCAD website at http://zamiacad.sourceforge.net/web/?q=tutorial.

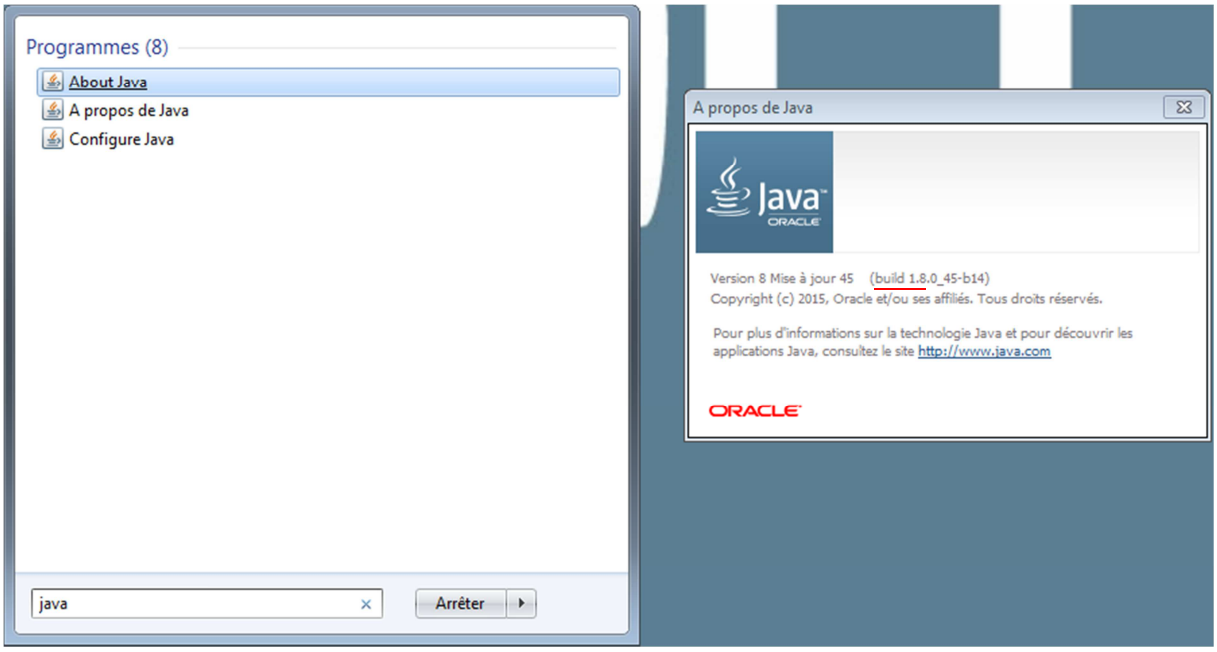
The CNE and STD handbook rulesets are also needed. Those rulesets can be downloaded from https://github.com/VHDLTool/VHDLHandbook/releases. Once the archive is unzipped, rulesets are located in directory \src\RuleSets\.

The present user guide relies on following software versions:

- For standalone version of ZamiaCad Rule Checker :
  - Windows 7 64bits
  - java 1.8.0_45
  - ZamiaCad 0.11.4
  - Rule Checker 3.2
- For deployable version of ZamiaCad Rule Checker :
  - Linux Debian 64bits
  - java 1.8.0_60
  - ZamiaCad 0.11.4
  - Rule Checker 3.2

E. Installation

Make sure you have java 1.8 installed on your computer (https://www.java.com/en/)



1. Case 1: Standalone Version

The Standalone Version of ZamiaCad Rule Checker is contained within archive zamia_standalone_V3.2.zip.

This archive contains:

- A portable version of Eclipse, ZamiaCAD plug-in and Rule Checker add-on
- A document called ZamiaCad_Rule_Checker_Features_Description_3.2.pdf that describes features of rule checker add-on
- A document called ZamiaCad_Rule_Checker_User_Guide_3.2.pdf that describes how to use Rule Checker.

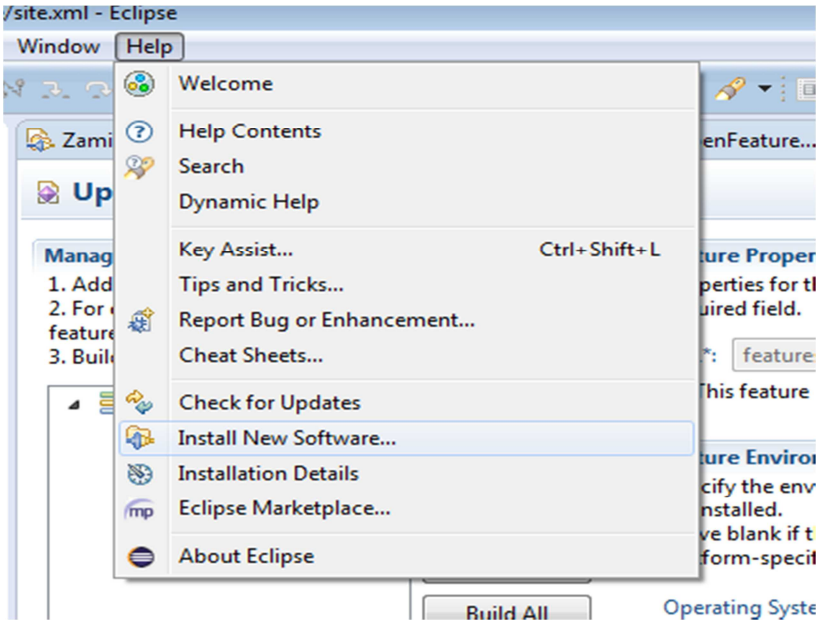
Use a file archiver to extract RuleChecker_3.2.zip.

Start ZamiaCad Rule Checker by executing zamia.exe. A pop-up will ask for your workspace folder. Every files / projects will be saved in this folder.

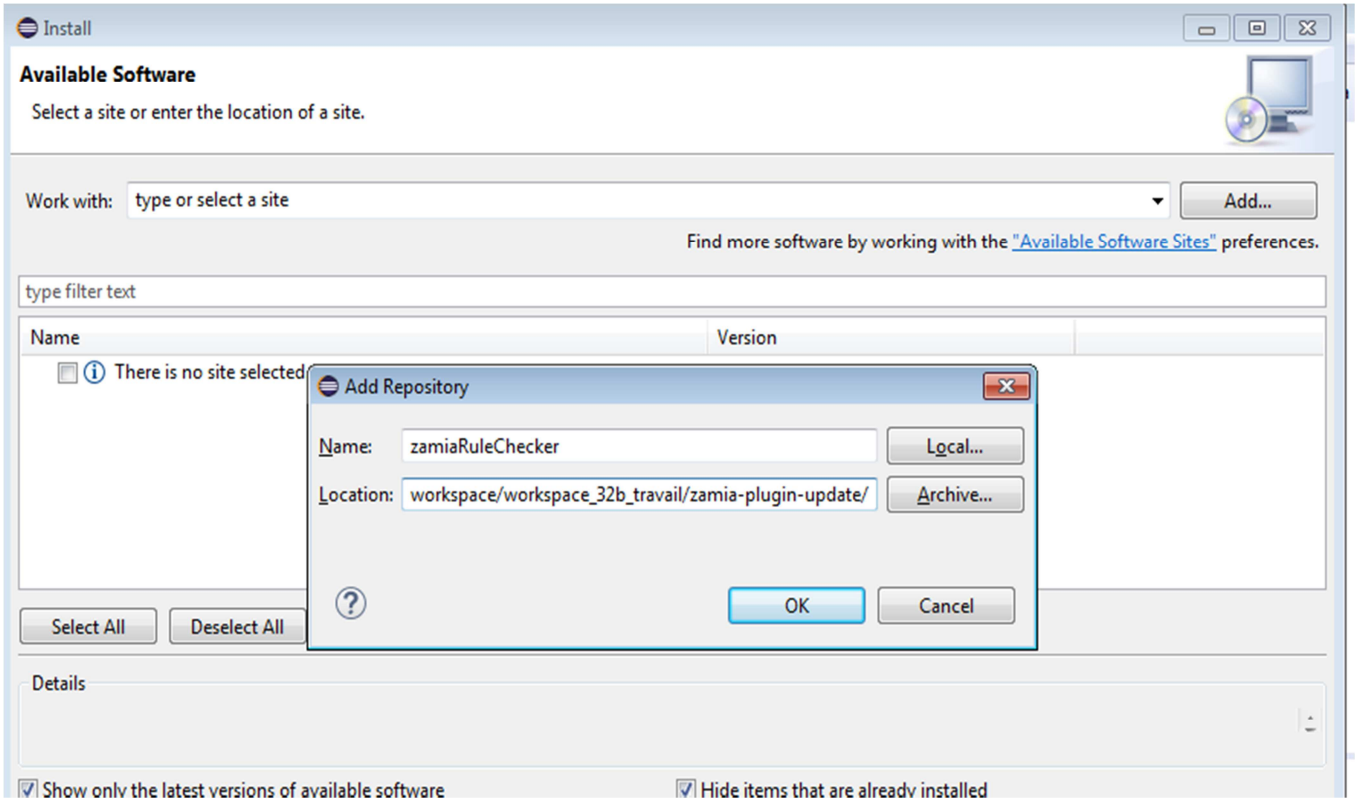
2. Case 2: Deployable Plugin

The Deployable Plugin is an archive called zamia-plugin-update_V3.2.zip.

To install this plugin in an existing eclipse, you must follow the procedure shown below:

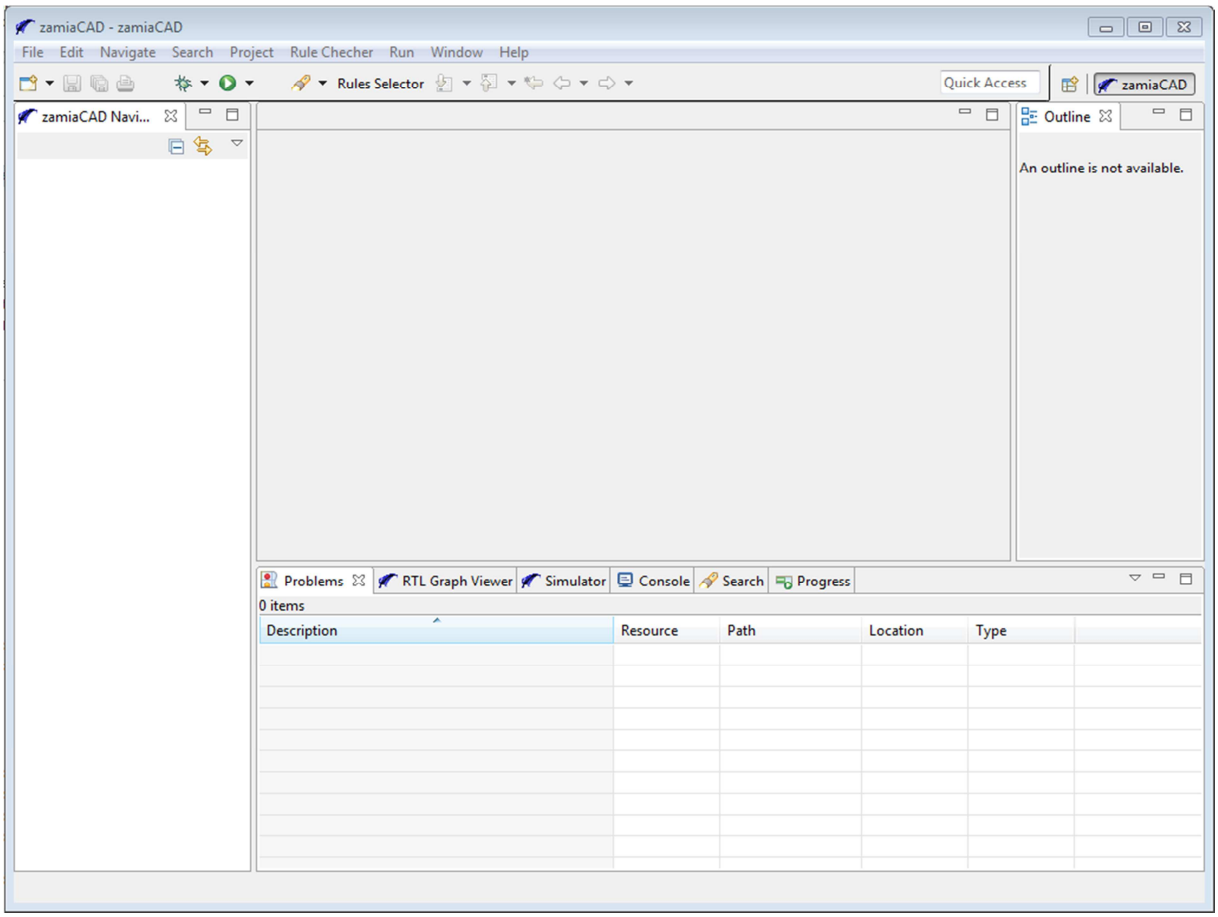


Use the path of the project « zamia-plugin-update » as in the following screenshot:



F. First Execution

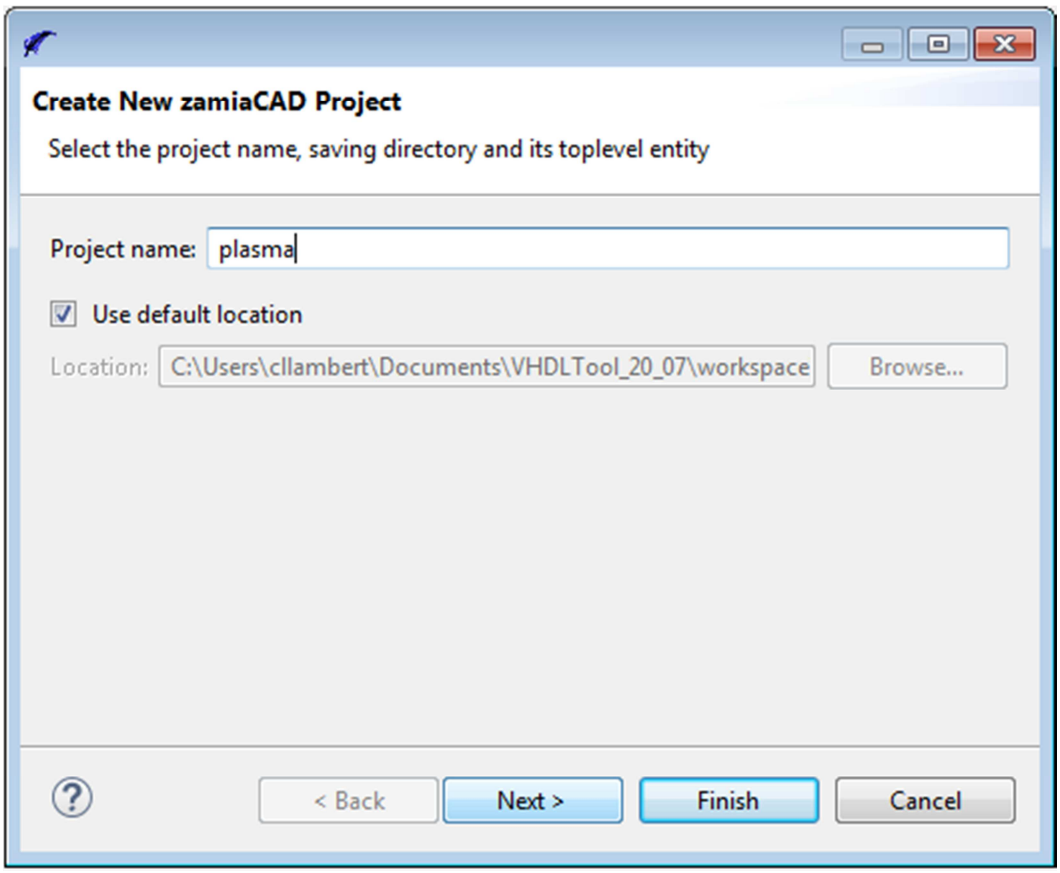
The software is ready to use and should look like this:



G. Create a VLSI project

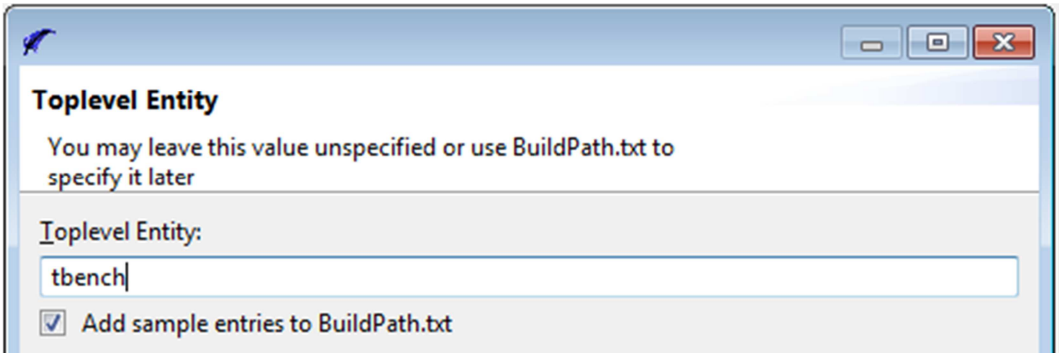
Create a new VLSI project: File -> New -> ZamiaCAD project

Enter a project name (eg: plasma). Go next.

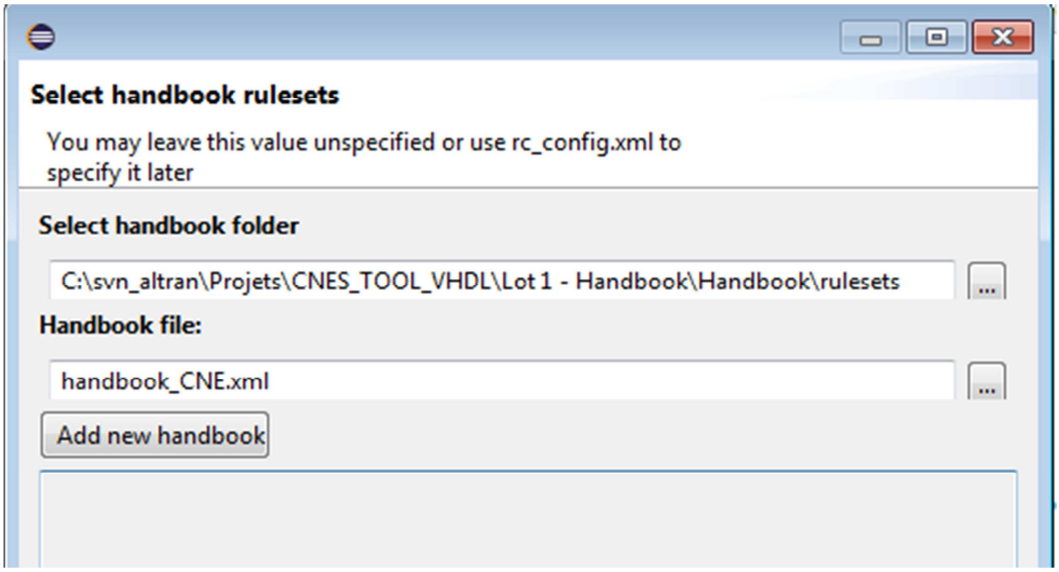


You can choose the top level of your fpga (eg: tbench). You may leave it empty and add it later. Go next.

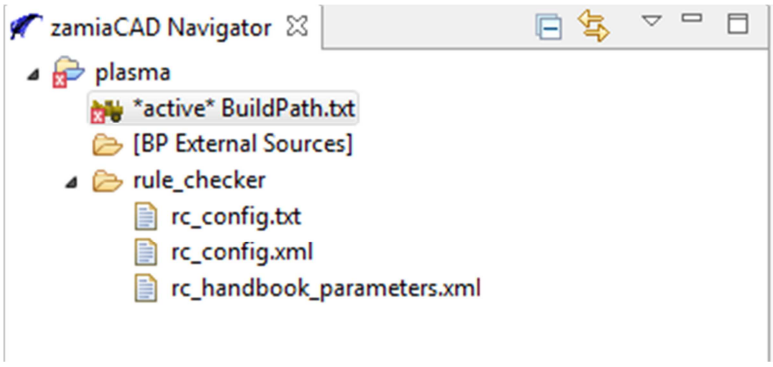
Please note that “toplevel” is a **keyword** used by ZamiaCad and so it can’t be used as an entity or architecture name.



You can choose any handbook ruleset in the tool (eg: handbook_CNE.xml). If you have more than one handbook ruleset, press “add new handbook” and choose a new handbook ruleset. You may leave it empty and add it later. Handbook rulesets have XML extension. Go next.



A full project build will be requested. You can decline it since we don’t have source files in our project. If you build the project, a folder [BP External Sources] is created and BuildPath.txt will report an error.

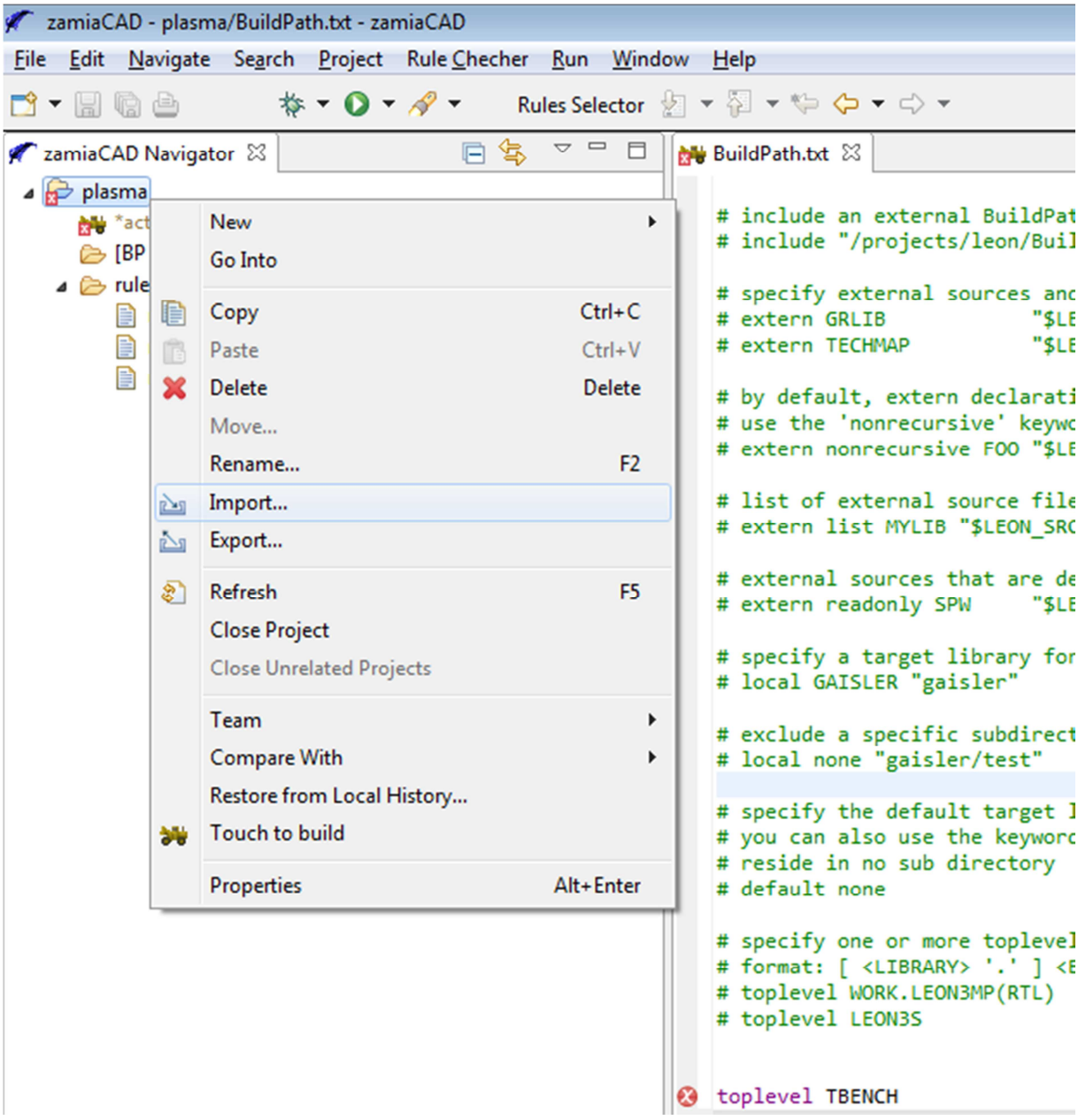


H. Import project

1. Sources

Now, we will import the VHDL sources into ZamiaCAD:

Select Import... from the plasma project's context menu (right-click on the plasma folder in the ZamiaCAD Navigator view).



Select General->Archive File, click Next > and open the downloaded archive file by using the Browse... button and click Finish....

Accept to do the full build.

2. Libraries

a. How to import library from a file imported in the project?

- Import libraries into project by means of eclipse (through import -> file system, etc.)
- Add in the BuildPath.txt: `local LIB_NAME "file"`

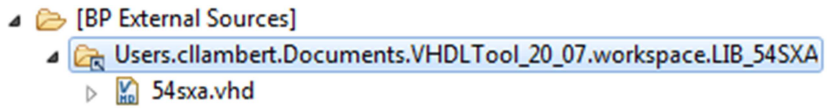
```
58 # specify a target library for sources inside your workspace
59 # local GAISLER "gaisler"
60 local A54SXA "54sxa.vhd"
```

b. How to import a library from external files to the project?

- Add in the BuildPath.txt: `extern readonly LIB_NAME "folder"`

```
55 # specify a target library for sources inside your workspace
56 # local GAISLER "gaisler"
57 extern readonly A54SXA "C:\Users\cllambert\Documents\VHDLTool_20_07\workspace\LIB_54SXA"
```

- where LIB_54SXA is a folder that contains external files. After project build, following view should be observed:



c. How to change default library?

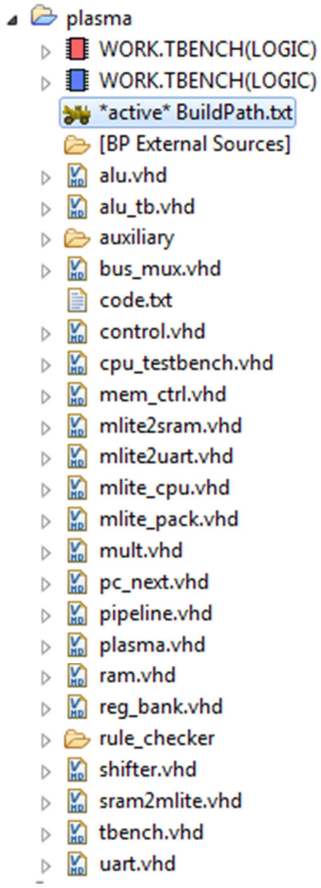
- By default, all compiled files are grouped in WORK library. To modify the default library, the following lines should be specified in BuildPath.txt.

```
69 # specify the default target library for local sources
70 # you can also use the keyword "NONE" here to exclude sources that
71 # reside in no sub directory
72 # default none
73
74 default "SOLODPUL"
75
76 # specify one or more toplevel(s) (each toplevel will be elaborated automatically)
77 # format: [ <LIBRARY> '.' ] <ENTITY> [ '(' <ARCHITECTURE> ')' ]
78 # toplevel WORK.LEON3MP(RTL)
79 # toplevel LEON3S
80
81 toplevel SOLODPUL.DPULOGIC
```

- In that case, all files belonging to the project defined by top DPULOGIC will be compiled within library SOLODPUL.
- Note that brackets are mandatory when defining the default library name.

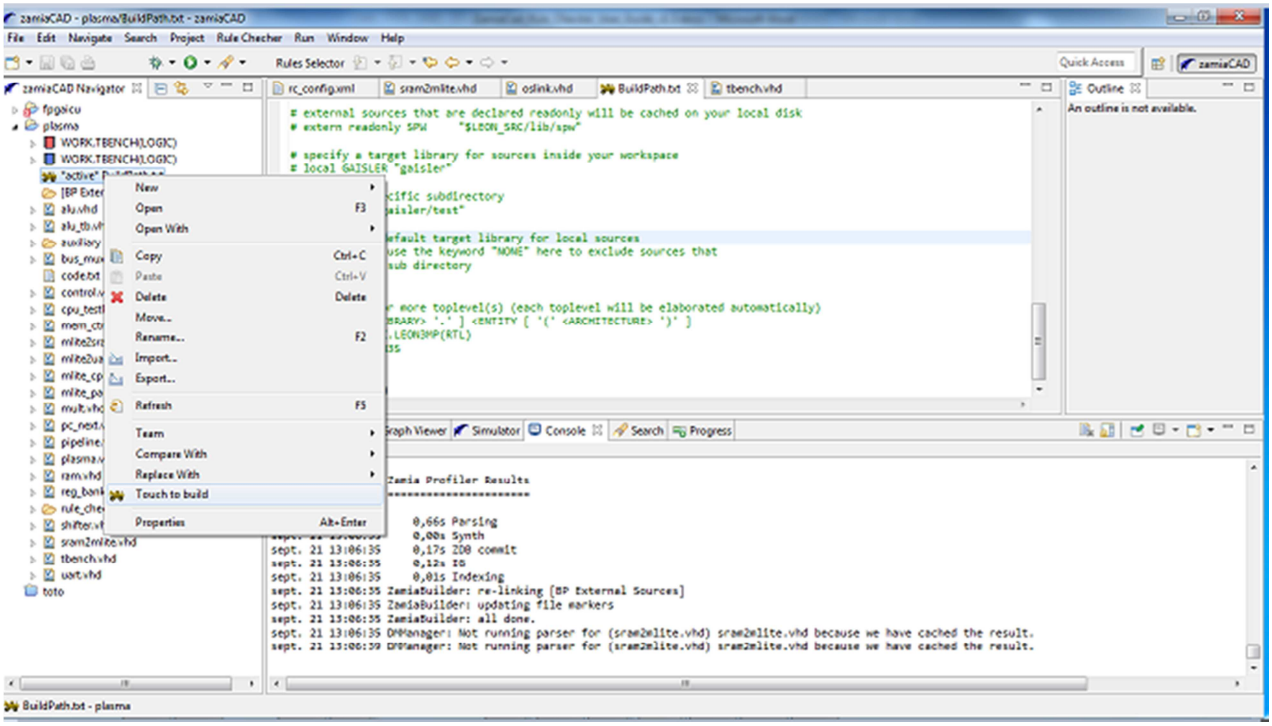
I. Build a project

In order to build a project, you need to know the top entity of your fpga (here: tbench). If you haven't filled the toplevel entity while creating the project, open BuildPath.txt file that is placed at the following location:

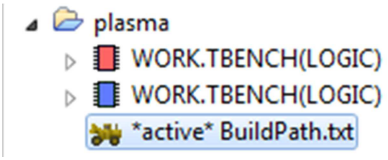


Then, write at the end of the file: toplevel TBENCH. Even if tbench is declared in lower case in tbench.vhd, it is mandatory to write TBENCH in uppercase in BuildPath.txt otherwise the top entity can't be found.

If you are not asked to build your project or have declined it, you can build it later. Right-click on the plasma folder in the ZamiaCAD Navigator view and Touch to build on BuildPath.txt.



Two nodes (blue and red) should appear as a result of the build.



J. Configure the Rule Checker

The Rule Checker uses several configuration files:

- ZamiaCAD's BuildPath.txt,
- rc_config.txt,
- rc_config.xml,
- rc_handbook_parameters.xml.

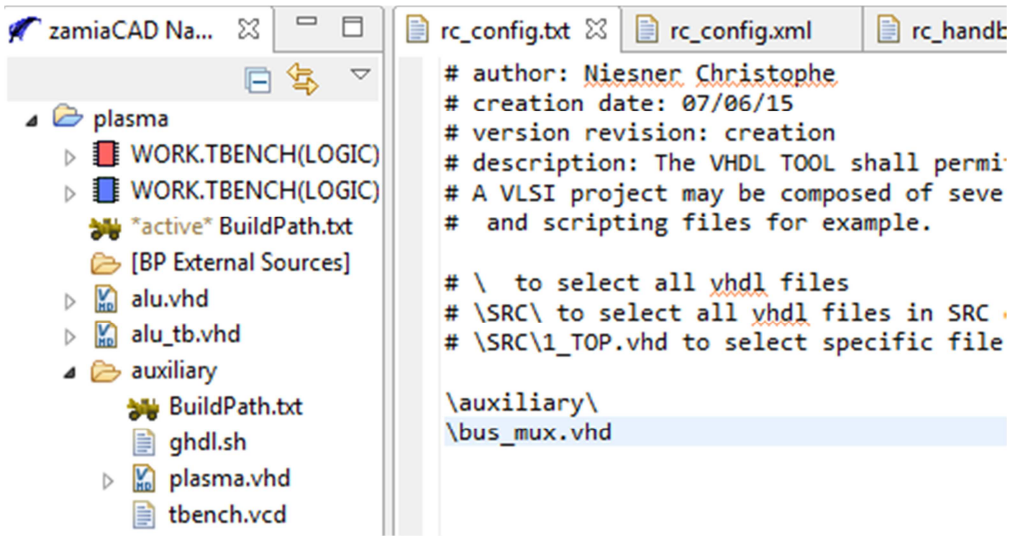
Each rc_* file must be located in rule_checker folder, BuildPath.txt must be at the root of the project.

1. rc_config.txt

While using tools of the software (eg: Line counter REQ_FEAT_FN20), you can choose the files that are parsed by editing the file **rc_config.txt**. This file is auto-generated with **** as default. This way, every files from the top of the project will be parsed.

The file rc_config.txt can be edited as below for example:

- Select a specific folder with his name: \folder_name\
- Select a specific file with his name: \file_name.vhd or \folder_name\file_name.vhd



Only plasma.vhd and bus_mux.vhd are selected

- Save rc_config.txt and only selected folders and files will be included during next rules checking

2. rc_config.xml

The **rc_config.xml** file allows at configuring the paths where handbook rulesets are located, where log reports will be stored.

Default rc_config.xml should look like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<vhdlTool>
<author>Niesner Christophe</author>
<creation_date>07/06/15</creation_date>
<description>The user shall configure Rule Checker with handbook files location, verifiers parameters, file locat
<version_revision>creation</version_revision>


    <root_directory>
        <alias>HANDBOOK_ROOT</alias>
        <path>C:\svn_altran\Projets\CNES_TOOL_VHDL\Lot 1 - Handbook\Handbook\rulesets</path>
    </root_directory>



    <handBook>
        <handBook_fileName>$HANDBOOK_ROOT\handbook_CNE.xml</handBook_fileName>
        <handBook_fileName>$HANDBOOK_ROOT\handbook_STD.xml</handBook_fileName>
    </handBook>


    <verifiers_parameters>$ROOT\rule_checker\rc_handbook_parameters.xml</verifiers_parameters>

    <log>
        <tool_directory>\rule_checker\reporting\tool\</tool_directory>
        <tool_fileName>rc_report_tool.xml</tool_fileName>
        <rule_directory>\rule_checker\reporting\rule\</rule_directory>
        <rule_fileName>rc_report_rule.xml</rule_fileName>
    </log>
</vhdlTool>
```

The user can modify location and rebuild the project to apply changes.

a. Root Directory

You can for example select a log folder. Log folder will have every reports generated by the Rule Checker. Its location can be modified by changing rc_config.xml as follows:

```
<root_directory>
        <alias>LOG_ROOT</alias>
        <path>C:\RC_Reports\plasma</path>
</root_directory>

<verifiers_parameters>
        $LOG_ROOT\rule_checker\rc_handbook_parameters.xml
</verifiers_parameters>

<log>
        <tool_directory>$LOG_ROOT\reporting\tool\</tool_directory>
        <tool_fileName>rc_report_tool.xml</tool_fileName>
        <rule_directory>$LOG_ROOT\reporting\rule\</rule_directory>
        <rule_fileName>rc_report_rule.xml</rule_fileName>
</log>
```

<root_directory> enables to create an alias. You can create as many as aliases as needed by using each time keyword <root_directory>.

In the example, $LOG_ROOT can then be used instead of the full path C:\RC_Reports\plasma.

b. Log reports

Two kinds of reports are made:

- Tool reports
- Rule reports (based on handbook files).

Both are XML and can be opened with an editor (ZamiaCAD, Notepad++, Excel …)

If no Log directory is defined through <root_directory>, default logs will be in \...\workspace\project\rule_checker\reporting\.

Tool reports will be created at <tool_directory> location. <tool_fileName> will define the prefix to be added at the log filename. The other part of the log filename will contain the tool name that has been executed.

Example

Tool “Line Counter (REQ_FEAT_FN20)” is called

Rc_config.xml has `<tool_fileName>rc_report_tool.xml</tool_fileName>`

Then the report file is named rc_report_tool_REQ_FEAT_FN20_Line counter.xml

The principles are identical for <rule_directory> and <rule_fileName>.

In this manner, you can have different location/prefix for tools and rules reports.

Please refer to ZamiaCad_Rule_Checker_Features_Description document [R2] for further information about Tools.

c. Import Rules / Handbook files

Rule Checker is based on XML handbook rulesets. We need to load handbook rulesets into the software. Handbook rulesets can be imported while creating a project or by modifying rc_config.xml.

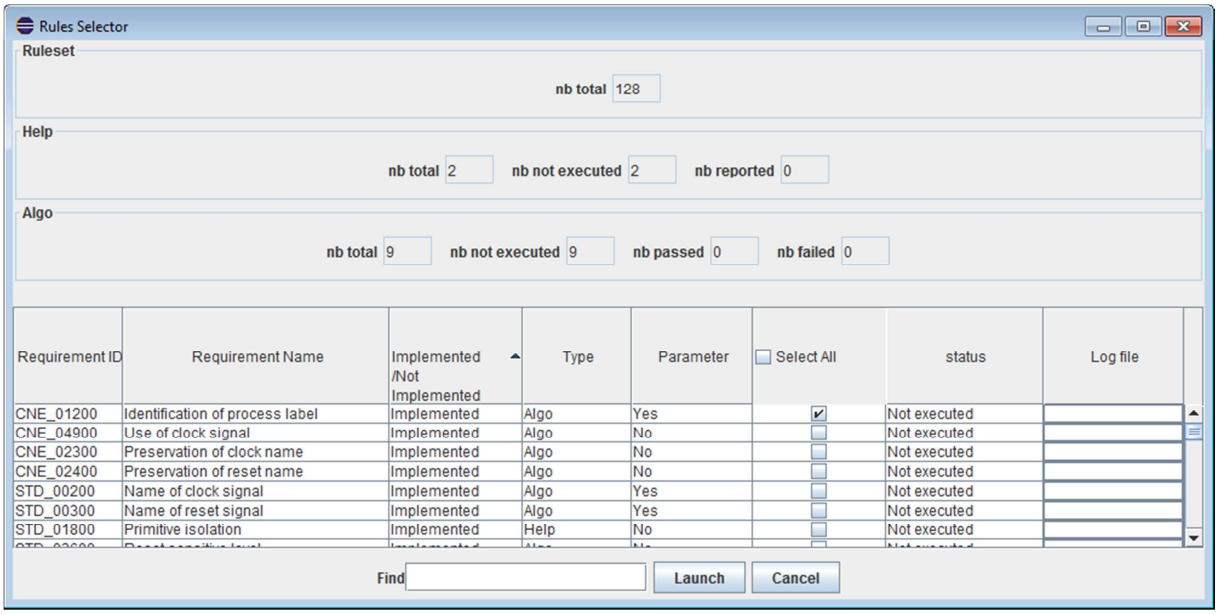
Default rc_config.xml has following content:

```
<!--
<handBook>
        <handBook_fileName>C:\Projects\Handbook\handbook_CNE.xml</handBook_fileName>
</handBook>
-->
```

First, remove lines <!- -and - - >. This will activate <handbook> tag and thus import XML files. Don't forget to modify the path if needed.

```
<handBook>
	<handBook_fileName>C:\Projects\Handbook\handbook_CNE.xml</handBook_fileName>
</handBook>
```

Handbooks are now available, and handbook_CNE.xml should be imported. Click on Rule Selector and see if rules are detected in the pop-up.



You can import several handbooks at once. Just add <handbook_fileName> between <handBook> tags with the path of another handbook (e.g.: `<handBook_fileName>C:\Handbook\handbook_STD.xml</handBook_fileName>`. STD rules are now imported)

Alias created in Root Directory can also be used:

```
<root_directory>
	<alias>HANDBOOK_ROOT</alias>
	<path>C:\dev_svn\FPGA\Projects\CNES_TOOL_VHDL\LOT 1 - Handbook\rulesets\</path>
</root_directory>

<handBook>
	<handBook_fileName>$HANDBOOK_ROOT\handbook_CNE.xml</handBook_fileName>
	<handBook_fileName>C:\Handbook\handbook_STD.xml</handBook_fileName>
</handBook>
```

As you can see, $HANDBOOK_ROOT replaces `C:\dev_svn\FPGA\Projects\CNES_TOOL_VHDL\Lot 1 - Handbook\Handbook\rulesets\`

3. rc_verifiers_parameters.xml

Handbook rules may rely on existing generic rules. A generic rule is composed of an algorithm and some generic parameters. The generic rule and parameters values are configured through the verifiers parameters file (*eg* rc_handbook_parameters.xml). In case that some parameters are missing in the parameters file, an error is reported to user and rule checking cannot be launched

Following figure shows content of “rc_handbook_parameters.xml” configuration file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<handbook_parameters xmlns:hb="HANDBOOK">
    <author>Niesner Christophe</author>
    <creation_date>07/06/15</creation_date>
    <description>Handbook rules may rely on existing generic rules. A
        generic rule is composed of an algorithm and some generic parameters.
        The generic rule and parameters values are configured through the
        verifiers parameters file. In case that some parameters are missing in
        the parameters file, then they are replaced with default values that
        are hard coded.</description>
    <version_revision>creation</version_revision>


    <hb:Rule UID="CNE_01200">
        <hb:RuleUID>CNE_01200</hb:RuleUID>
        <hb:RuleGEN>GEN_01200</hb:RuleGEN>
        <hb:RuleParameter>
            <name>position</name>
            <type>PositionE</type>
            <value>prefix</value>
        </hb:RuleParameter>
        <hb:RuleParameter>
            <name>partName</name>
            <type>String</type>
            <value>p_</value>
        </hb:RuleParameter>
    </hb:Rule>
    <hb:Rule UID="CNE_02300">
        <hb:RuleUID>CNE_02300</hb:RuleUID>
        <hb:RuleGEN>GEN_02300</hb:RuleGEN>
    </hb:Rule>
```

In this example, the rule CNE_02300 uses the generic rule GEN_02300 without parameter. The other rule “CNE_01200” is based on generic rule GEN_01200 with two parameters “position” and “process_label” specified. The type of the position parameter is “PositionE”, this type is an enumeration with values: “prefix”, “suffix” and “contain”.

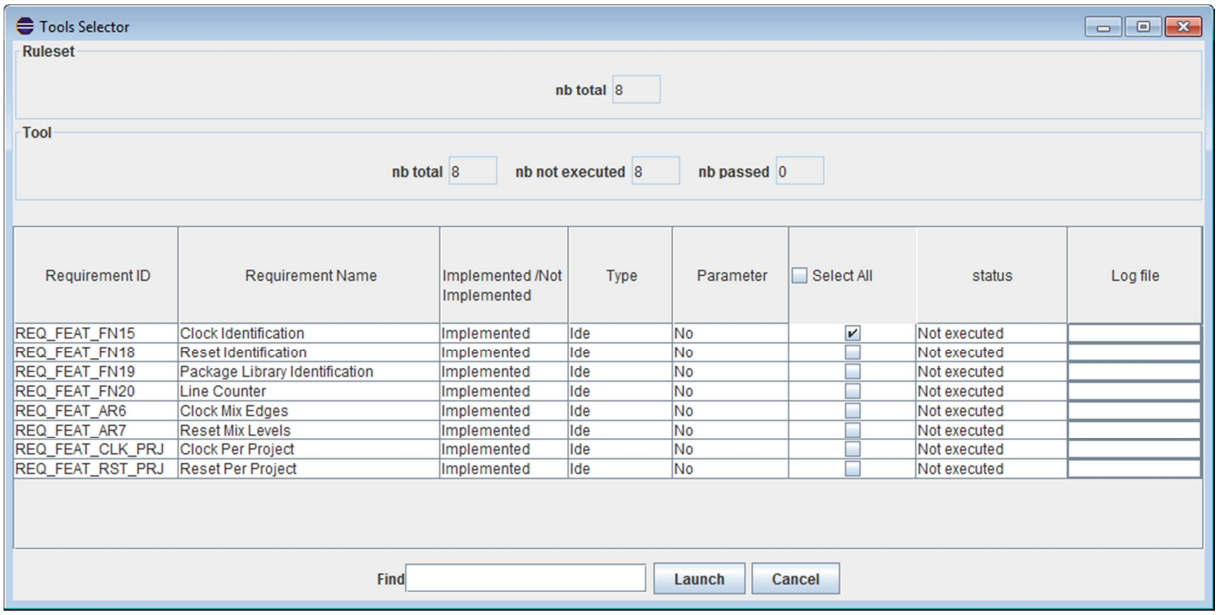
K. Create a custom handbook ruleset

Please refer to Handbook User Guide [R3].

L. Launch Tools

Tools Verification can be executed from menu `rule checker -> tools selector` on the upper bar of ZamiaCad.

A window then appears with a list of all tools supported by rule checker.



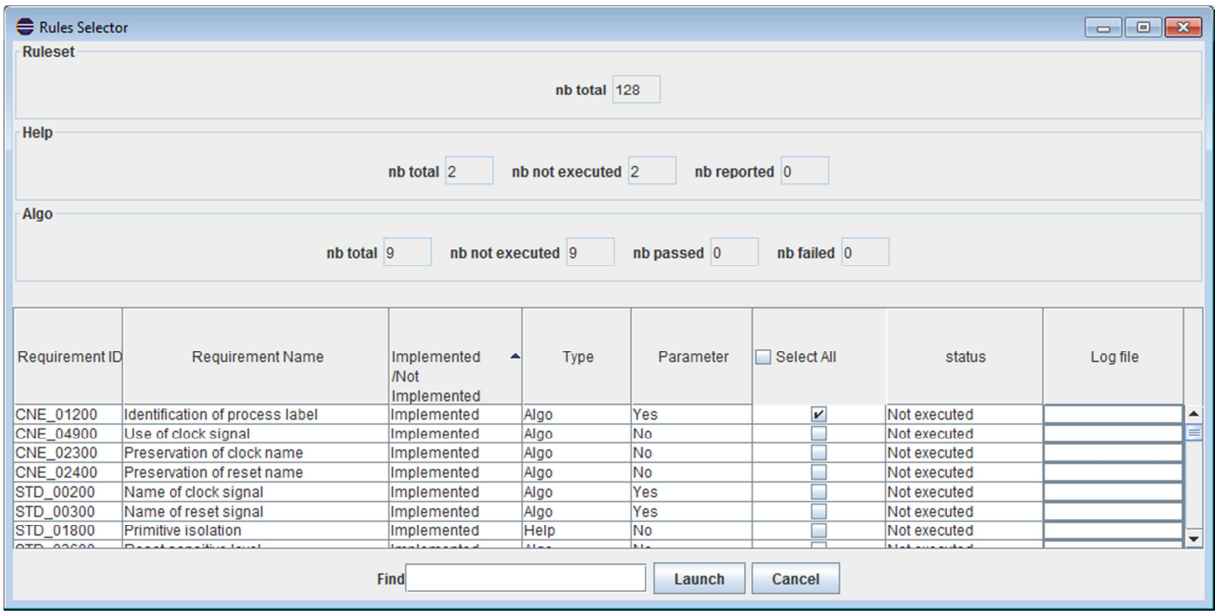
The user selects which tools he wants to check and then click on launch button.

The user can click on the report to open it.

M. Launch Rules Verification

Rules Verification can be executed from menu `rule checker -> rules selector` on the upper bar of ZamiaCad.

A window then appears with a list of all rules imported from handbook rulesets defined by rc_config.xml.



Rules are classified in 2 categories:

- help: report will be generated to help reviewers analyzing if rule is passed or not.
- algo: report will be generated and will return whether rule is passed or failed on current VLSI project.

The user selects which rules he wants to check and then click on launch button.

A status is given for all executed rules with associated reports in case of failed algo rules or reported help rules.

The user can click on the report to open it.

N. Acronyms and Abbreviations

Acronym and abbreviation	Meaning
CNES	Centre National d'Etude Spatial (French Space Agency)
FPGA	Field Programmable Gate Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration