Rule Checker Features

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
03 august 2015	1.0	Document creation	Arnaud DANIEL Christophe NIESNER
03 september 2015	1.1	update with latest rules checker GUI	Arnaud DANIEL Christophe NIESNER

Table of contents

Contenu

A. Scope

1. Identification

This document aims to describe the features of the Rule Checker add-on of ZamiaCad tool.

B. Mentioned documents

1. Reference documents

Index	Title	Reference	Date
[R1]	ZamiaCad_Rule_Checker_User_Guide_v1.2		03 september 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

In the following sections, we consider the project “plasma” with default configuration for “XML Files Selection” as described in User Guide [R1].

All report files are created in directory “$PROJECT_ROOT\rule_checker\reporting\” They are named rc_report_tool_RuleID.xml for tools execution and rc_report_rule_RuleID.xml for rules selector execution.

E. Detailed description of features

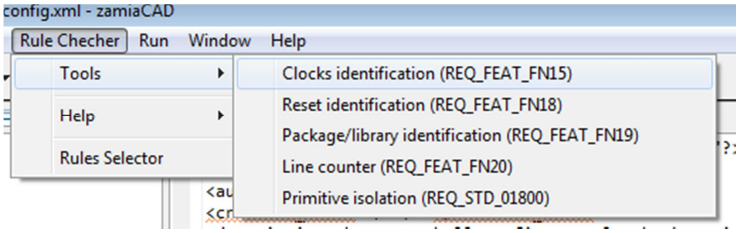
1. Tools

a. Clocks Identification (SRS_REQ_FEAT_FN15)

Rationale: The objective is to identify all clock signals in a VLSI project.

Expected Result:

When we select the feature Clocks Identification, as shown in this figure, the rule checker tool detects the clock signal in corresponding VLSI project as described.



We can see the result of this detection in rc_report_tool_REQ_FEAT_FN15.xml file. In this example, the clock signal “CLK” is detected in file \pc_next.vhd” entity “PC_NEXT” architecture “LOGIC” process “PC_NEXT”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN15>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN15</description>
  <creationDate>Thu Jul 23 11:25:32 CEST 2015</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <nbLine>68</nbLine>
    <entity>
      <entityName>PC_NEXT</entityName>
      <entityLoc>16</entityLoc>
      <architecture>
        <architectureName>LOGIC</architectureName>
        <architectureLoc>28</architectureLoc>
        <process>
          <processName>PC_NEXT</processName>
          <processLoc>34</processLoc>
          <clockSignal>
            <clockSignalName>CLK</clockSignalName>
            <clockSignalLoc>59</clockSignalLoc>
          </clockSignal>
        </process>
      </architecture>
    </entity>
  </file>
```

In this other example, no clock signal are detected in file "\alu.vhd" entity "ALU" architecture "LOGIC" process "ALU_PROC", we can see "NA" to "clockSignalName".

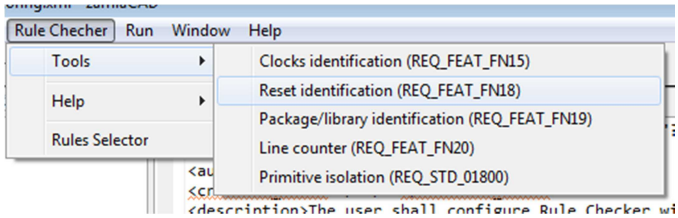
```
  <file>
    <fileName>\alu.vhd</fileName>
    <nbLine>71</nbLine>
    <entity>
      <entityName>ALU</entityName>
      <entityLoc>16</entityLoc>
      <architecture>
        <architectureName>LOGIC</architectureName>
        <architectureLoc>23</architectureLoc>
        <process>
          <processName>ALU_PROC</processName>
          <processLoc>31</processLoc>
          <clockSignal>
            <clockSignalName>NA</clockSignalName>
          </clockSignal>
        </process>
      </architecture>
    </entity>
  </file>
```

b. Reset Identification (SRS_REQ_FEAT_FN18)

Rationale: The objective is to identify all asynchronous reset signals in a VLSI project.

Expected Result:

When we select the feature Reset Identification, as shown in this figure, the rule checker tool detects the reset signals in corresponding VLSI project.



We can see the result of this detection in rc_report_tool_REQ_FEAT_FN18.xml file. In this example, the reset signals “PAUSE_IN” and “RESET_IN” are detected in file “\pc_next.vhd” entity “PC_NEXT” architecture “LOGIC” process “PC_NEXT” associate to clock signal “CLK”.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN18>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN18</description>
  <creationDate>Thu Jul 23 11:30:01 CEST 2015</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <nbLine>68</nbLine>
    <entity>
      <entityName>PC_NEXT</entityName>
      <entityLoc>16</entityLoc>
      <architecture>
        <architectureName>LOGIC</architectureName>
        <architectureLoc>28</architectureLoc>
        <process>
          <processName>PC_NEXT</processName>
          <processLoc>34</processLoc>
          <clockSignal>
            <clockSignalName>CLK</clockSignalName>
            <clockSignalLoc>59</clockSignalLoc>
            <resetSignal>
              <resetSignalName>PAUSE_IN</resetSignalName>
              <resetSignalLoc>53</resetSignalLoc>
            </resetSignal>
            <resetSignal>
              <resetSignalName>RESET_IN</resetSignalName>
              <resetSignalLoc>57</resetSignalLoc>
            </resetSignal>
          </clockSignal>
        </process>
```

In this other example, no reset signal are detected in file \mlite2sram.vhd" entity "MLITE2SRAM" architecture "LOGIC" process "SET_STATE" associate to clock signal "CLK", we can see "NA" to "resetSignalName".

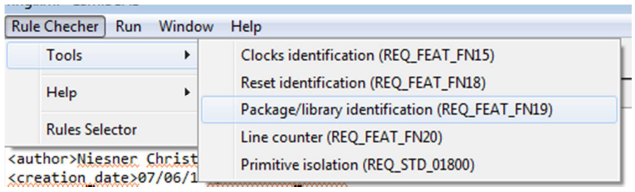
```
<file>
  <fileName>\mlite2sram.vhd</fileName>
  <nbLine>145</nbLine>
  <entity>
    <entityName>MLITE2SRAM</entityName>
    <entityLoc>10</entityLoc>
    <architecture>
      <architectureName>LOGIC</architectureName>
      <architectureLoc>32</architectureLoc>
      <process>
        <processName>SET_STATE</processName>
        <processLoc>46</processLoc>
        <clockSignal>
          <clockSignalName>CLK</clockSignalName>
          <clockSignalLoc>48</clockSignalLoc>
          <resetSignal>
            <resetSignalName>NA</resetSignalName>
          </resetSignal>
        </clockSignal>
      </process>
```

c. Package/Library Identification (SRS_REQ_FEAT_FN19)

Rationale: The objective is to make an exhaustive list of declared libraries so that to identify which libraries are standard ones (IEEE) and which are custom ones developed especially for the project.

Expected Result:

When we select the feature Package/library Identification, as shown in this figure, the rule checker tool detects the libraries used in all vhdl files.



We can see the result of this detection in rc_report_tool_REQ_FEAT_FN19.xml file. In this example, the libraries “IEEE.STD_LOGIC_1164.ALL” and “WORK.MLITE_PACK.ALL” are used in file “\pc_next.vhd”.

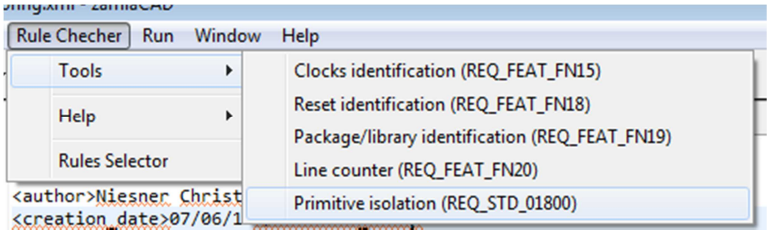
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN19>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN19</description>
  <creationDate>Thu Jul 23 11:43:45 CEST 2015</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <nbLine>68</nbLine>
    <libraryName>IEEE.STD_LOGIC_1164.ALL</libraryName>
    <libraryName>WORK.MLITE_PACK.ALL</libraryName>
    <entity>
      <entityName>PC_NEXT</entityName>
      <entityLoc>16</entityLoc>
    </entity>
  </file>
```

d. Primitive Isolation (SRS_REQ_STD_01800)

Rationale: The objective is to help reviewers to identify which VHDL files declare libraries other than IEEE (eg ALTERAMF, AXCELERATOR) and verify if concerned files are correctly isolated from the rest of the project.

Expected Result:

When we select the feature Primitive Isolation, as shown in this figure, the rule checker tool detects the vhdl files used a specific library.



We can see the result of this detection in rc_report_tool_REQ_FEAT_STD_01800.xml file. In this example, the files “mult.vhd”, “alu_tb.vhd”, … using “IEEE.STD_LOGIC_UNSIGNED.ALL” library.

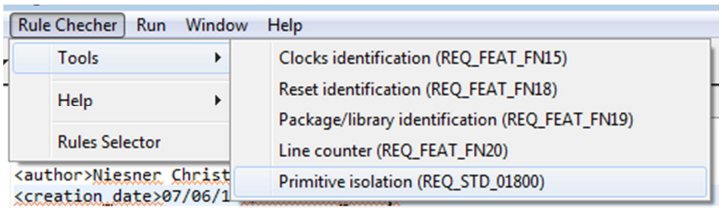
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_STD_01800>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_STD_01800</description>
  <creationDate>Thu Jul 23 11:54:43 CEST 2015</creationDate>
  <primitive>
    <libraryName>IEEE.STD_LOGIC_UNSIGNED.ALL</libraryName>
    <fileName>mult.vhd</fileName>
    <fileName>alu_tb.vhd</fileName>
    <fileName>mlite2uart.vhd</fileName>
    <fileName>ram.vhd</fileName>
    <fileName>mlite2sram.vhd</fileName>
    <fileName>mlite_cpu.vhd</fileName>
    <fileName>sram2mlite.vhd</fileName>
    <fileName>reg_bank.vhd</fileName>
    <fileName>cpu_testbench.vhd</fileName>
  </primitive>
```

e. Line Counter (SRS_REQ_FEAT_FN20)

Rationale: The objective is to have information about code complexity by providing the number of lines per VHDL files.

Expected Result:

When we select the feature line Counter, as shown in this figure, the rule checker tool count the lines in all vhdl files.



We can see the result in rc_report_tool_REQ_FEAT_FN20.xml file. In this example, the file "\pc_next.vhd" has 68 lines.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN20>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN20</description>
  <creationDate>Thu Jul 23 11:46:40 CEST 2015</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <nbLine>68</nbLine>
  </file>
  <file>
    <fileName>\mem_ctrl.vhd</fileName>
    <nbLine>243</nbLine>
  </file>
```

2. Rules Selector

This section is left empty intentionally.

F. 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