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
11 september 2015	1.2	Version number aligned on delivered Rules Checker version. No modification done inside document.	Arnaud DANIEL
21 september 2015	1.3	Document renaming in Features Description Addition of a glossary to explain log parameters Move of SRS_REQ_STD_01800 rule from Tools section to Rules Selector section Addition of “Detailed Description” and “Limitations” for each tool and rule.	Arnaud DANIEL

04 december 2015	2.1	Update with WP-CLK and WP-RST.	Arnaud DANIEL Christophe NIESNER
11 march 2016	V3.1	Addition of features covering WP-PROC and WP-CDC	Arnaud DANIEL Christophe NIESNER
01 june 216	V3.2	Final Review	Arnaud DANIEL

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_v3.2		01 june 2016

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 by default in directories:

- “$PROJECT_ROOT\rule_checker\reporting\rule\Algo\rc_report_rule_RuleID_RuleName\” for rules algo
- “$PROJECT_ROOT\rule_checker\reporting\rule\Help\rc_report_rule_RuleID_RuleName\” for rules help
- “$PROJECT_ROOT\rule_checker\reporting\tool\rc_report_tool_RuleID_RuleName\” for tools.

Reports are named rc_report_tool_RuleID_RuleName.xml for tools selector execution and rc_report_rule_RuleID_RuleNam.xml for rules (algo and help) selector execution.

E. Detailed description of features

1. Tools Selector

a. Clocks Identification (SRS_REQ_FEAT_FN15)

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

Detailed Description: Clocks are detected according to the definition given by the IEEE standard. This covers the following use cases:

The criteria for detecting clock are:

- Clock signal detection is done inside “process”
- The following expressions shall represent a rising edge clock
  - RISING_EDGE(*clk_signal_name*)
  - *clk_signal_name* = ‘1’ and *clk_signal_name*’EVENT
  - *clk_signal_name*’EVENT and *clk_signal_name* = ‘1’
  - *clk_signal_name* = ‘1’ and not *clk_signal_name*’STABLE
  - not *clk_signal_name*’STABLE and *clk_signal_name* = ‘1’
- The following expressions shall represent a falling edge clock

- FALLING_EDGE(*clk_signal_name*)
- *clk_signal_name* = '0' and *clk_signal_name*'EVENT
- *clk_signal_name*'EVENT and *clk_signal_name* = '0'
- *clk_signal_name* = '0' and not *clk_signal_name*'STABLE
- not *clk_signal_name*'STABLE and *clk_signal_name* = '0'

The VHDL TOOL shall be able to detect different clocks per process.

Limitations: The VHDL TOOL is only capable to detect clocks inside process. At this stage of progress, it is not possible to detect sources (*eg* PLL) and sinks (*eg* gated clock) of those clocks.

Furthermore, the VHDL TOOL does not detect clocks when they are combined with additional combinatory.

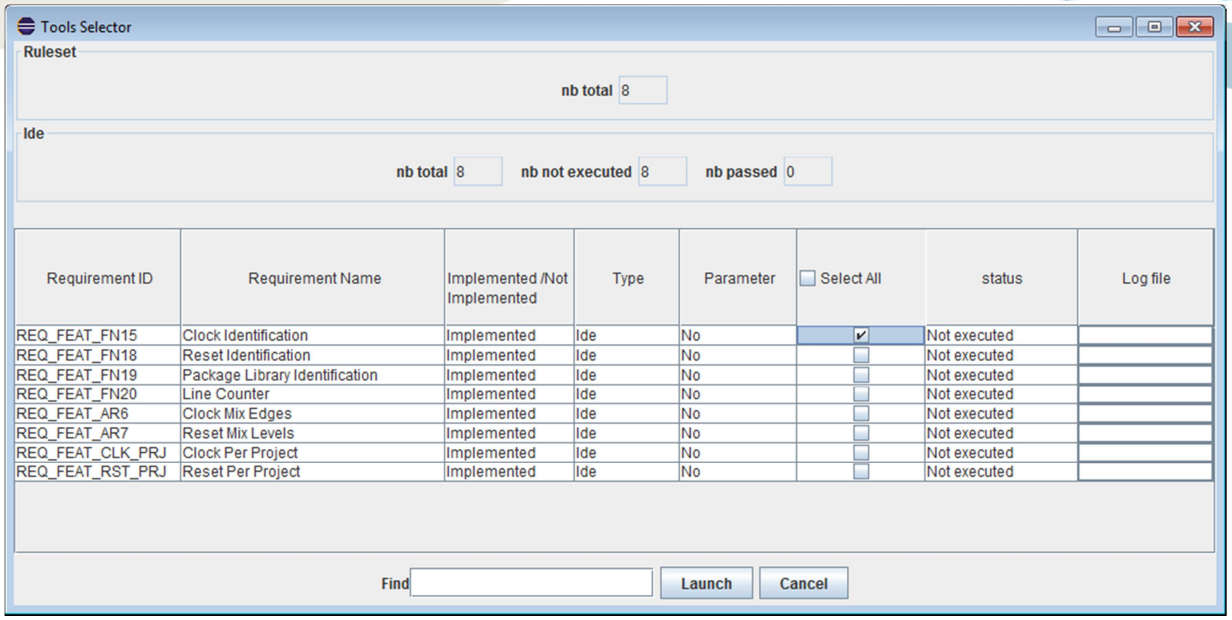
In this example the clock signal `clk_A` isn't detected.

```
P_TEST2 : process(clk_A, A_reset, B_reset)
begin
      if A_reset='0' then
         clk_selA_int <= '0';

      elsif falling_edge(clk_A) and B_reset='0' then
        clk_selA_int <= '0';
      end if;
end process;
```

The clocks are attached to process not registers at this stage of progress.

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_Clock Identification.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>
  <version>V1.3 date 21/09/2015</version>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN15</description>
  <creationDate>Tue Sep 22 13:45:55 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>
          <processIsSynchronous>yes</processIsSynchronous>
          <clockSignal>
            <clockSignalName>CLK</clockSignalName>
            <clockSignalLoc>59</clockSignalLoc>
          </clockSignal>
        </process>
      </architecture>
    </entity>
  </file>
</REQ_FEAT_FN15>
```

In this other example, no clock signal is detected in file “\alu.vhd” entity “ALU” architecture “LOGIC” process “ALU_PROC”, we can see “no” to “processIsSynchronous”.

```
  <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>
          <processIsSynchronous>no</processIsSynchronous>
        </process>
      </architecture>
    </entity>
  </file>
</REQ_FEAT_FN15>
```

b. Reset Identification (SRS_REQ_FEAT_FN18)

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

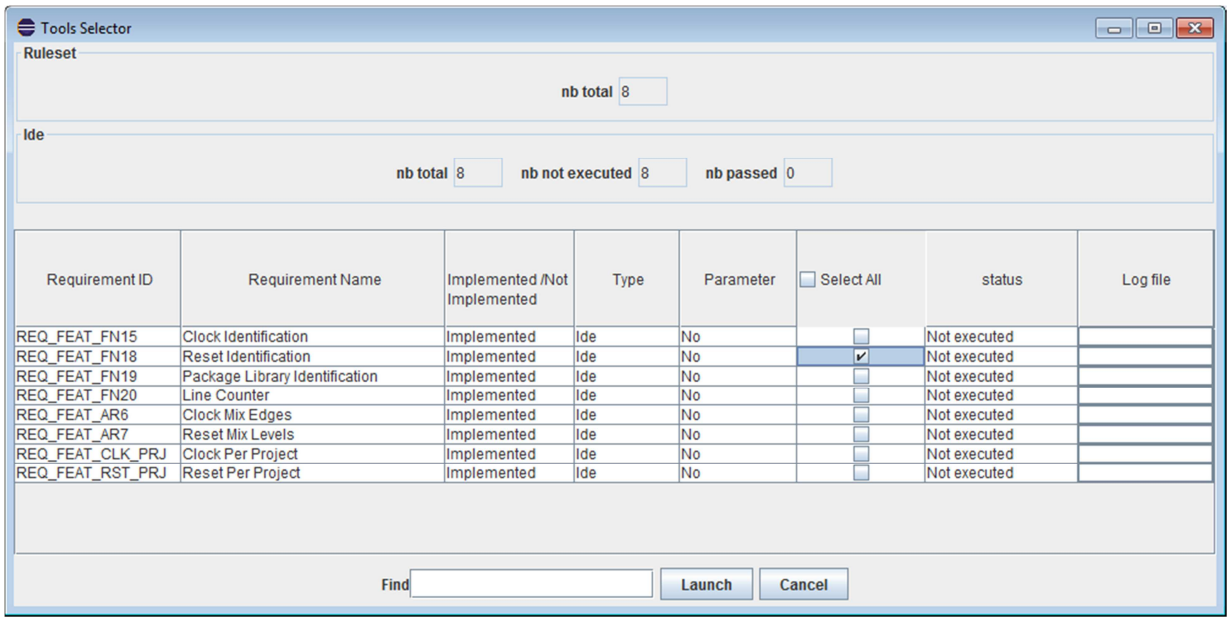
Detailed Description: Asynchronous reset detection is done according to the following sequence:

- Detect process of type synchronous. Such a process involves a clock signal.
- Identify asynchronous reset by searching any signal declared in condition statement out of the "if…end if" statement involving the clock signal.

Limitations: The resets are attached to process not registers at this stage of progress.

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.



Requirement ID	Requirement Name	Implemented /Not Implemented	Type	Parameter	Select All	status	Log file
REQ_FEAT_FN15	Clock Identification	Implemented	Ide	No		Not executed	
REQ_FEAT_FN18	Reset Identification	Implemented	Ide	No	✔	Not executed	
REQ_FEAT_FN19	Package Library Identification	Implemented	Ide	No		Not executed	
REQ_FEAT_FN20	Line Counter	Implemented	Ide	No		Not executed	
REQ_FEAT_AR6	Clock Mix Edges	Implemented	Ide	No		Not executed	
REQ_FEAT_AR7	Reset Mix Levels	Implemented	Ide	No		Not executed	
REQ_FEAT_CLK_PRJ	Clock Per Project	Implemented	Ide	No		Not executed	
REQ_FEAT_RST_PRJ	Reset Per Project	Implemented	Ide	No		Not executed	

We can see the result of this detection in rc_report_tool_REQ_FEAT_FN18_Reset Identification.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 version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN18>
  <author>rule checker</author>
  <version>V1.3 date 21/09/2015</version>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN18</description>
  <creationDate>Tue Sep 22 13:55:27 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>
          <processIsSynchronous>yes</processIsSynchronous>
          <clockSignal>
            <clockSignalName>CLK</clockSignalName>
            <clockSignalLoc>59</clockSignalLoc>
            <clockSignalHasAsynchronousReset>yes</clockSignalHasAsynchronousReset>
            <resetSignal>
              <resetSignalName>PAUSE_IN</resetSignalName>
              <resetSignalLoc>53</resetSignalLoc>
            </resetSignal>
            <resetSignal>
              <resetSignalName>RESET_IN</resetSignalName>
              <resetSignalLoc>57</resetSignalLoc>
            </resetSignal>
          </clockSignal>
        </process>
      </architecture>
    </entity>
  </file>
```

In this other example, no reset signal is detected in file \mlite2sram.vhd” entity “MLITE2SRAM” architecture “LOGIC” process “SET_STATE” associate to clock signal “CLK”, we can see “no” to “clockSignalHasAsynchronousReset”.

```
  <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>
          <processIsSynchronous>yes</processIsSynchronous>
          <clockSignal>
            <clockSignalName>CLK</clockSignalName>
            <clockSignalLoc>48</clockSignalLoc>
            <clockSignalHasAsynchronousReset>no</clockSignalHasAsynchronousReset>
          </clockSignal>
        </process>
        <process>
          <processName>WORK</processName>
          <processLoc>53</processLoc>
          <processIsSynchronous>no</processIsSynchronous>
        </process>
      </architecture>
    </entity>
  </file>
</REQ_FEAT_FN18>
```

c. Package/Library Identification (SRS_REQ_FEAT_FN19)

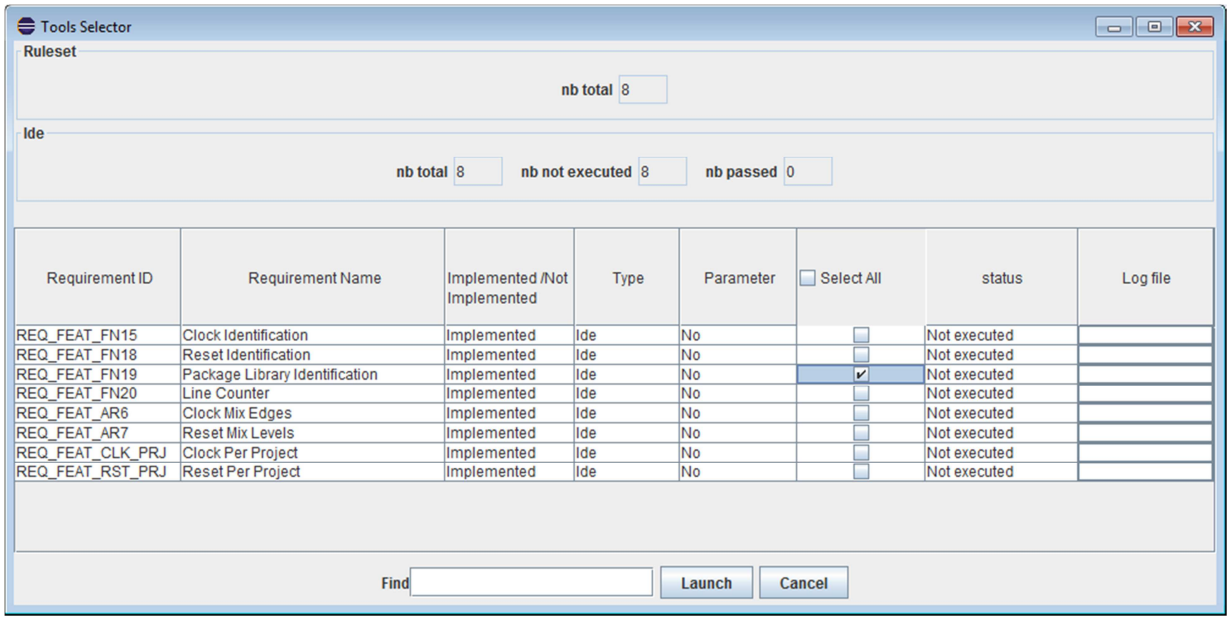
Rationale: The objective is to make an exhaustive list of declared libraries so that to identify which libraries are standard ones (IEEE) and which are custom ones developed especially for the project.

Detailed Description: The VHDL TOOL shall report the libraries declared in all VHDL files of the VLSI project.

Limitations: None.

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_Package Library Identification.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
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN19>
  <author>rule checker</author>
  <version>V1.3 date 21/09/2015</version>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN19</description>
  <creationDate>Tue Sep 22 13:58:49 CEST 2015</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <nbLine>68</nbLine>
    <library>
      <libraryName>IEEE.STD_LOGIC_1164.ALL</libraryName>
      <libraryLoc>13</libraryLoc>
    </library>
    <library>
      <libraryName>WORK.MLITE_PACK.ALL</libraryName>
      <libraryLoc>14</libraryLoc>
    </library>
  </file>
</REQ_FEAT_FN19>
```

d. Line Counter (SRS_REQ_FEAT_FN20)

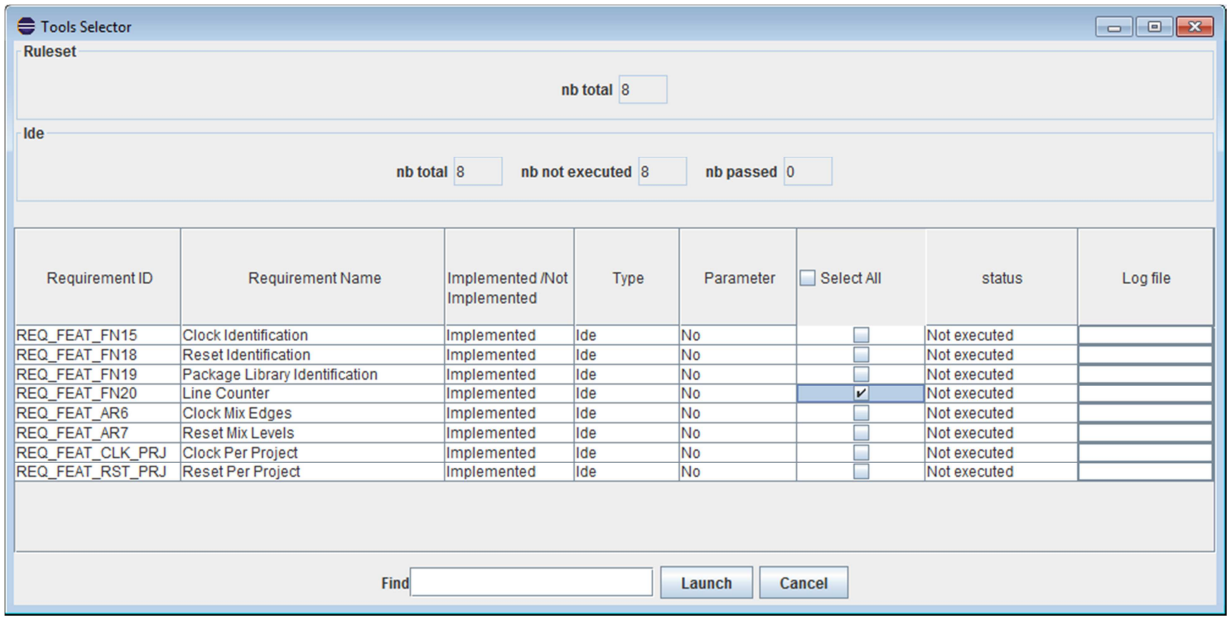
Rationale: The objective is to have information about code complexity by providing the number of lines per VHDL files.

Detailed Description: The VHDL TOOL shall report the number of lines for all VHDL files of the VLSI project. The line number reports the number of lines in the VHDL file until the End Of File (EOF) special character is met.

Limitations: None.

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_line Counter.xml file. In this example, the file "\pc_next.vhd" has 68 lines.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN20>
  <author>rule checker</author>
  <version>V1.3 date 21/09/2015</version>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN20</description>
  <creationDate>Tue Sep 22 14:00:00 CEST 2015</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <nbLine>68</nbLine>
  </file>
  <file>
    <fileName>\mem_ctrl.vhd</fileName>
    <nbLine>243</nbLine>
  </file>
</REQ_FEAT_FN20>
```

e. Clock domain change (SRS_REQ_FEAT_FN22)

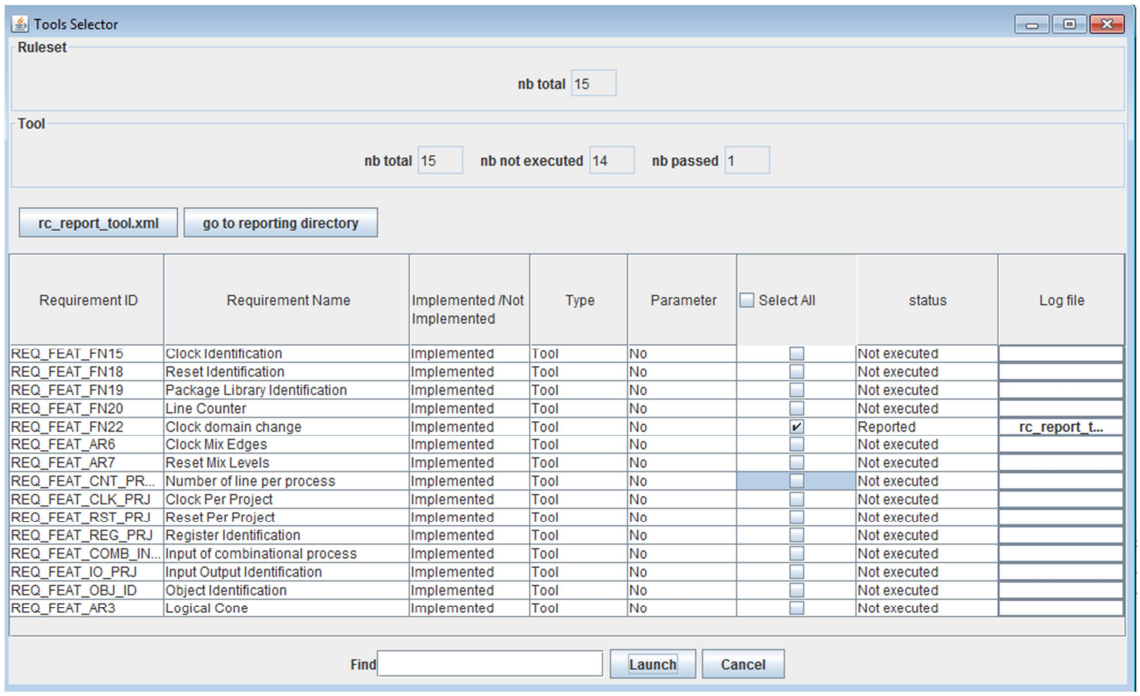
Rationale: The objective is to return registers that are generated by one clock source, and are consumed with another clock source.

Detailed Description: with the help of the "SRS_REQ_FEAT_REG_PRJ" function, a report is built containing all registers of VLSI project. Rule Checker checks the clock domain where a register is generated, and the clock domain where the register is consumed. A violation report is built when a clock domain change is detected.

Limitations: none at this level.

Expected Result:

When we select the feature Clock Domain change, as shown in this figure, the rule checker tool says the clock used when a register is generated and consumed.



A report is built for clock domain change. We can see the result in rc_report_tool_REQ_FEAT_FN22_Clock_domain_changer.xml file. In this example, the file "\pc_next.vhd" has "clockSource0" for its register "PC_REG".

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_FN22>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_FN22</description>
  <creationDate>Fri Apr 15 16:22:19 CEST 2016</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <fileNbLine>68</fileNbLine>
    <entity>
      <entityName>PC_NEXT</entityName>
      <entityLoc>16</entityLoc>
      <architecture>
        <architectureName>LOGIC</architectureName>
        <architectureLoc>28</architectureLoc>
        <process>
          <processName>PC_NEXT</processName>
          <processLoc>34</processLoc>
          <processNbLine>31</processNbLine>
          <processIsSynchronous>yes</processIsSynchronous>
          <register>
            <registerName>PC_REG</registerName>
            <registerClockSourceTag>clockSource0</registerClockSourceTag>
            <registerLoc>60</registerLoc>
            <violationType/>
          </register>
          <register>
            <registerName>PC_REG</registerName>
            <registerClockSourceTag>clockSource0</registerClockSourceTag>
            <registerLoc>58</registerLoc>
            <violationType/>
          </register>
        </process>
      </architecture>
    </entity>
  </file>
  <file>
```

f. Logical Cone (SRS_REQ_FEAT_AR3)

Rationale: The objective is to return logical cone ascending and descending for a selected object in the VLSI project.

Detailed Description: with the help of the "SRS_REQ_FEAT_OBJ_ID" function, a report is built containing all object of VLSI project (register, combinational input, I/O component or top of FPGA). With the "rc_handbook_parameters.xml" the user choose an object and the number of stages.

When the user executes the "SRS_REQ_FEAT_AR3" function, 2 reports are built one descending and one ascending.

In the descending report, the logical cone function searches the different sources of the selected object.

To find a signal source of the selected object, we consider all signals at the right of the operator "<=", for example: in this assignment signal_A <= signal_B and signal_C; signal_B and signal_C are signal sources of the selected signal_A. For this research the user can see some stop condition:

- Max stage : in case when max stage is reached
- IO PAD: in case of Input of top FPGA
- IO : in case of Input/output of a black box
- State machine : in case of perpetual loopback condition
- Constant : in case of a signal assignment to a constant

In the ascending report, the logical cone function searches the different sinks of the selected object.

To find a signal sink of the selected object,

- We consider all signals at the left of the operator "<=", for example: in this assignment signal_A is signal sink of the selected signal_B.

```
Signal_A <= signal_B and signal_C;
```

- We consider the signals appearing in the condition of an if: in this example the signal “clk_selA_int” is signal sink of signals “A_reset” and “clk_A”.

```
P_TEST1 : process(clk_A, clk_B, A_reset, B_reset)
begin
   if A_reset='0' then
         clk_selA_int <= '0';

   elsif falling_edge(clk_A) then
        clk_selA_int <= '0';
   end if;
end process;
```

For this research the user can see some stop condition:

- Max stage : in case max stage is reached
- IO PAD: in case of output of top FPGA
- IO : in case of Input/output of a black box
- State machine : in case of perpetual loopback condition
- Constant : in case of a signal assignment to a constant

In this example, we can see the max stage number (10) and the selected object tag (“REG_10”)

rc_config.xml | rc_handbook_parameters.xml

```
133
134     <hb:Rule UID="REQ_FEAT_AR3">
135         <hb:RuleUID>REQ_FEAT_AR3</hb:RuleUID>
136         <hb:RuleGEN>REQ_FEAT_AR3</hb:RuleGEN>
137         <hb:RuleParameter>
138             <name>nbStage</name>
139             <type>Integer</type>
140             <value>10</value>
141         </hb:RuleParameter>
142         <hb:RuleParameter>
143             <name>signalTag</name>
144             <type>String</type>
145             <value>REG_10</value>
146         </hb:RuleParameter>
147     </hb:Rule>
148
149 </handbook_parameters>
```

Limitations: The user can choose the max stage number, some signals may have many sources or sinks, and treatment of logical cone can be very long, it is recommended to gradually increase the stage number to prevent the tool turns long without giving control to the user.

g. Clock Mix Edge (SRS_REQ_FEAT_AR6)

Rationale: The objective is to check that there is only one edge detection mechanism used for each clock source.

Detailed Description: Search Clock VHDL elements in VLSI project with "Clock identification" function, and verify that clock are always used with the same edge (falling edge or rising edge).

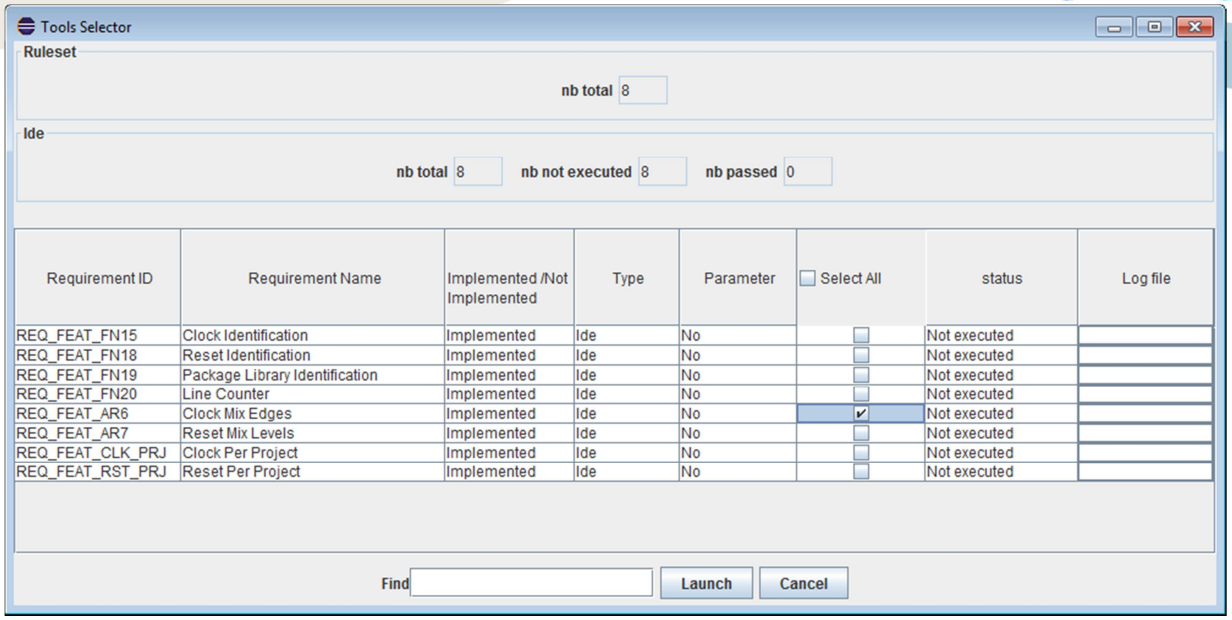
3 reports are created:

- The most detailed report contains all clocks per entity per process. Each clock is connected to a specific clock source as obtained in SRS_REQ_FEAT_CLK_PRJ. Then, for each clock used in process, the edge is indicated, so it permits to identify on which process a particular clock source is used either on rising or falling edge.
- The second report says for each entity of the VLSI project and for each clock source if it is used on rising edge, falling edge or both edges. It permits to identify in which entity a clock is used on both edges.
- The third report says for every clock source in a VLSI project if it is used exclusively on rising edge, falling edge or both edges.

Limitations: none at this level.

Expected Result:

When we select the feature Clock Mix Edge, as shown in this figure, the rule checker tool says when a clock is used on both edges in all vhdl files.


Tools Selector
Ruleset
nb total 8
Ide
nb total 8
nb not executed 8
nb passed 0
Requirement ID
Requirement Name
Implemented /Not Implemented
Type
Parameter
Select All
status
Log file
REQ_FEAT_FN15 Clock Identification Implemented Ide No Not executed
REQ_FEAT_FN18 Reset Identification Implemented Ide No Not executed
REQ_FEAT_FN19 Package Library Identification Implemented Ide No Not executed
REQ_FEAT_FN20 Line Counter Implemented Ide No Not executed
REQ_FEAT_AR6 Clock Mix Edges Implemented Ide No Not executed
REQ_FEAT_AR7 Reset Mix Levels Implemented Ide No Not executed
REQ_FEAT_CLK_PRJ Clock Per Project Implemented Ide No Not executed
REQ_FEAT_RST_PRJ Reset Per Project Implemented Ide No Not executed
Find
Launch
Cancel

h. Reset mix level (SRS_REQ_FEAT_AR7)

Rationale: The objective is to check that reset signal uses two distinct activation level.

Detailed Description: Search Reset VHDL elements in VLSI project with "Reset identification" function, and verify that the reset signal is used with two distinct activation levels.

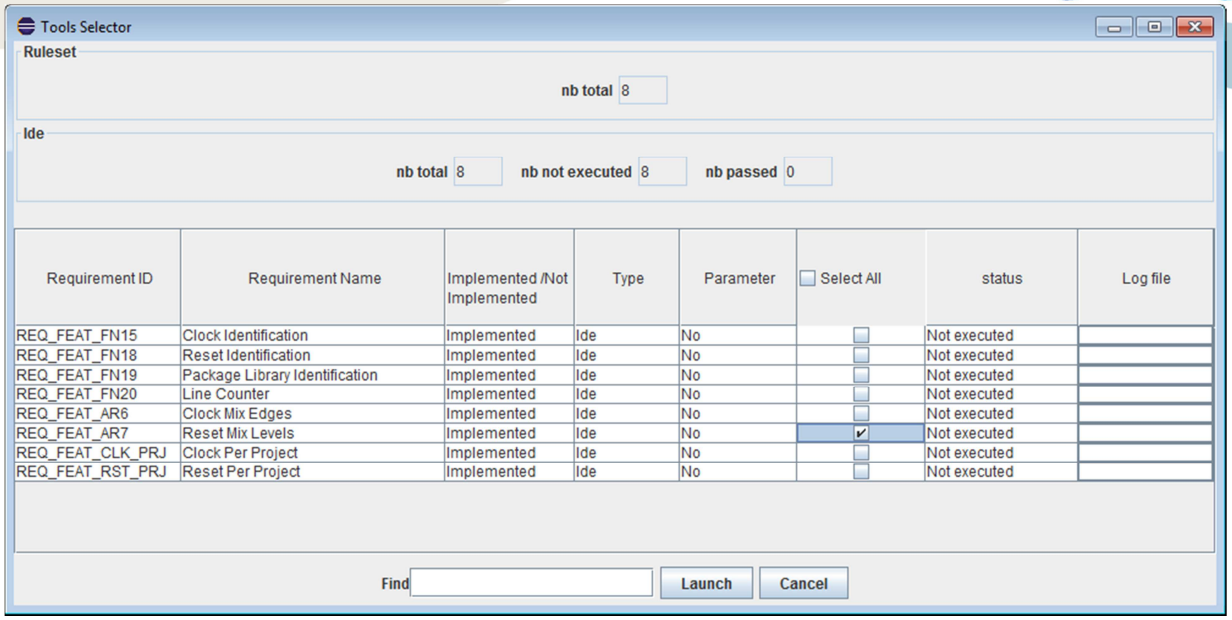
3 reports are created:

- The most detailed report contains all reset per entity per process. Each reset is connected to a specific reset source as obtained in SRS_REQ_FEAT_RST_PRJ. Then, for each reset used in process, the level is indicated, so it permits to identify on which process a particular reset source is used either on high or low level.
- The second report says for each entity of the VLSI project and for each reset source if it is used on high level, low level or both levels. It permits to identify in which entity a reset is used on both levels.
- The third report says for every reset source in a VLSI project if it is used exclusively on high level, low level or both levels.

Limitations: none at this level.

Expected Result:

When we select the feature Clock Mix Edge, as shown in this figure, the rule checker tool says when a clock is used on both edges in all vhdl files.


Tools Selector
Ruleset
nb total 8
Ide
nb total 8
nb not executed 8
nb passed 0
Requirement ID
Requirement Name
Implemented /Not Implemented
Type
Parameter
Select All
status
Log file
REQ_FEAT_FN15 Clock Identification Implemented Ide No Not executed
REQ_FEAT_FN18 Reset Identification Implemented Ide No Not executed
REQ_FEAT_FN19 Package Library Identification Implemented Ide No Not executed
REQ_FEAT_FN20 Line Counter Implemented Ide No Not executed
REQ_FEAT_AR6 Clock Mix Edges Implemented Ide No Not executed
REQ_FEAT_AR7 Reset Mix Levels Implemented Ide No Not executed
REQ_FEAT_CLK_PRJ Clock Per Project Implemented Ide No Not executed
REQ_FEAT_RST_PRJ Reset Per Project Implemented Ide No Not executed
Find
Launch
Cancel

i. Clock per project (SRS_REQ_FEAT_CLK_PRJ)

Rationale: The objective is to list all clocks sources in the project.

Detailed Description: For each clock raised by the clock identification tool (see SRS_REQ_FEAT_FN15), find the clock source. 3 types of clock source can be retrieved:

- Input port : clock source comes from an input port of top of VLSI project

```
entity fpgaicu is

	port(
		Clk_20MHZ	: in  Std_Logic;	-- input port
		CLK_1HZOBC	: in Std_Logic;		-- input port
                                                                    …
```

- Instance Output : clock source comes from an output of a black box (eg PLL, clock buffer, crypted file)

```
\$1I218\ : CLKINT port map(A => CLK_B, Y => \$1N96\);
```

- Signal assignment : clock source comes from signal assignment : direct assignment, assignment with combinational logic, assignment in process

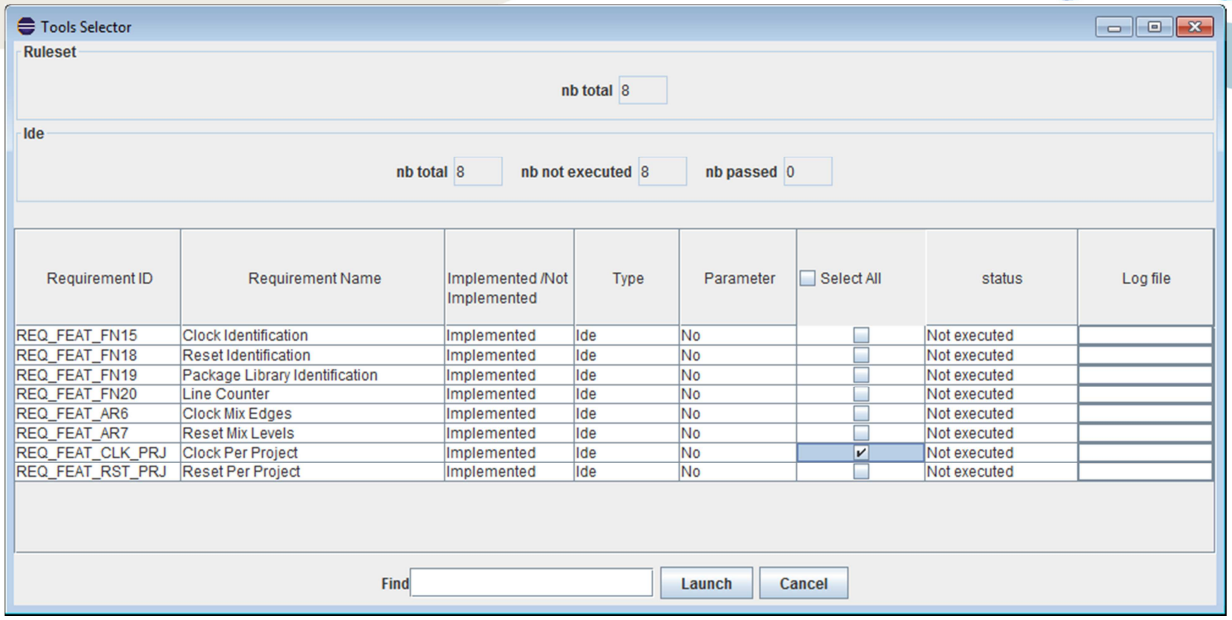
```
clk1 <= clk;  -- direct assignment

clk1 <= clk and not B; -- assignment with combinational logic

process(clk, rst)
begin
	if rst ='0' then
	 clk_div <= '0'; -- assignment in process
	elsif rising_edge(clk) then
	 clk_div <= NOT clk_div;
	end if;
end process;
```

Each clock source is tagged with the entity name, architecture name, the filename, line number. This creates a list of clock sources.

Limitations: none

Expected Result:

When we select the feature Clock per project, as shown in this figure, the rule checker tool lists all the clock source of a project.


Tools Selector
Ruleset
nb total 8
Ide
nb total 8
nb not executed 8
nb passed 0
Requirement ID
Requirement Name
Implemented /Not Implemented
Type
Parameter
Select All
status
Log file
REQ_FEAT_FN15 Clock Identification Implemented Ide No Not executed
REQ_FEAT_FN18 Reset Identification Implemented Ide No Not executed
REQ_FEAT_FN19 Package Library Identification Implemented Ide No Not executed
REQ_FEAT_FN20 Line Counter Implemented Ide No Not executed
REQ_FEAT_AR6 Clock Mix Edges Implemented Ide No Not executed
REQ_FEAT_AR7 Reset Mix Levels Implemented Ide No Not executed
REQ_FEAT_CLK_PRJ Clock Per Project Implemented Ide No Not executed
REQ_FEAT_RST_PRJ Reset Per Project Implemented Ide No Not executed
Find
Launch
Cancel

j. Reset per project (SRS_REQ_FEAT_RST_PRJ)

Rationale: The objective is to list all reset sources in the project.

Detailed Description: For each reset issue to the reset identification, find the reset source.

3 types of reset source can be retrieved:

- Input port : reset source comes from an input port of top of VLSI project

```
entity fpgaicu is

	port(
		Reset_n : in  Std_Logic;	-- input port
		…
```

- Instance Output : reset source comes from an output of a black box (eg CLKINT)

```
\$1I218\ : CLKINT port map(A => RESET_B, Y => \$1N96\);
```

- Signal assignment : reset source comes from a signal assignment : direct assignment, assignment with combinational logic, assignment in process

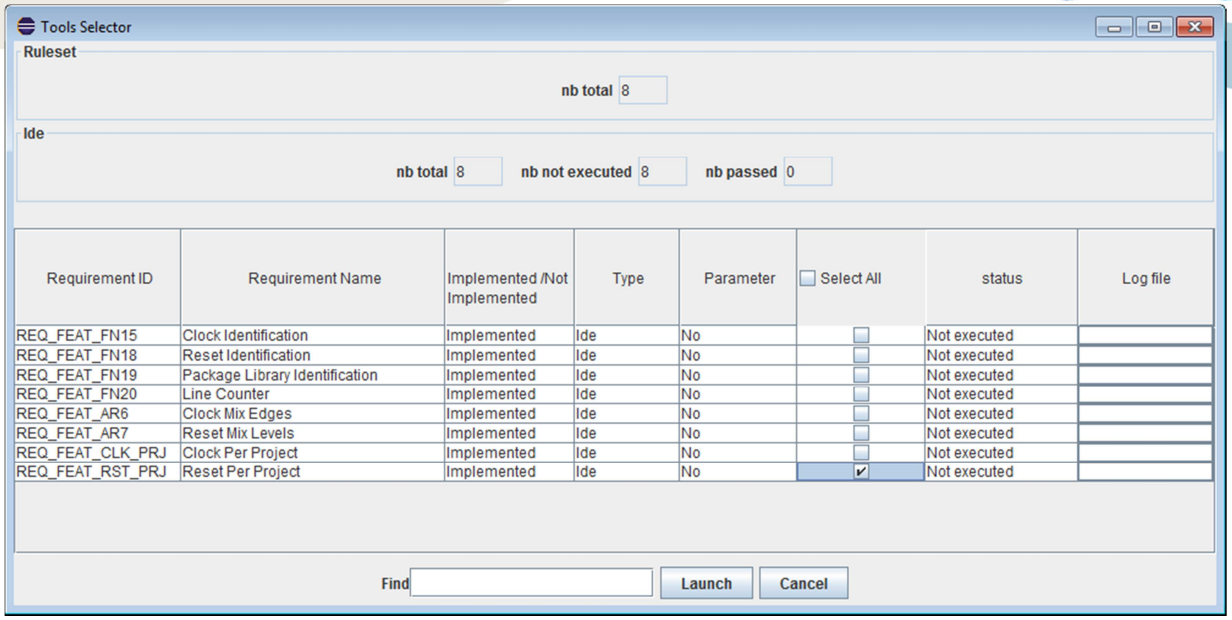
```
reset1 <= reset;  -- direct assignment

reset1 <= reset_A and not B; -- assignment with combinational logic

process(clk, rst)
begin
	if rst ='0' then
	 rst_resync <= '0'; -- assignment in process
	elsif rising_edge(clk) then
	 rst_resync <= '1';
	end if;
end process;
```

Each reset source is tagged with the entity name, architecture name, the filename, line number. This creates a list of reset sources.

Limitations: none at this level.

Expected Result:

When we select the feature Reset per project, as shown in this figure, the rule checker tool lists all the reset source of a project.


Tools Selector
Ruleset
nb total 8
Ide
nb total 8
nb not executed 8
nb passed 0
Requirement ID
Requirement Name
Implemented /Not Implemented
Type
Parameter
Select All
status
Log file
REQ_FEAT_FN15 Clock Identification Implemented Ide No Not executed
REQ_FEAT_FN18 Reset Identification Implemented Ide No Not executed
REQ_FEAT_FN19 Package Library Identification Implemented Ide No Not executed
REQ_FEAT_FN20 Line Counter Implemented Ide No Not executed
REQ_FEAT_AR6 Clock Mix Edges Implemented Ide No Not executed
REQ_FEAT_AR7 Reset Mix Levels Implemented Ide No Not executed
REQ_FEAT_CLK_PRJ Clock Per Project Implemented Ide No Not executed
REQ_FEAT_RST_PRJ Reset Per Project Implemented Ide No Not executed
Find
Launch
Cancel

k. Registers identification (SRS_REQ_FEAT_REG_PRJ)

Rationale: The objective aims to estimate the resource of the VLSI project in terms of registers, prior to synthesis and also to identify all registers so that it can be possible to check their correct initialization and the potential clock domain change between each register.

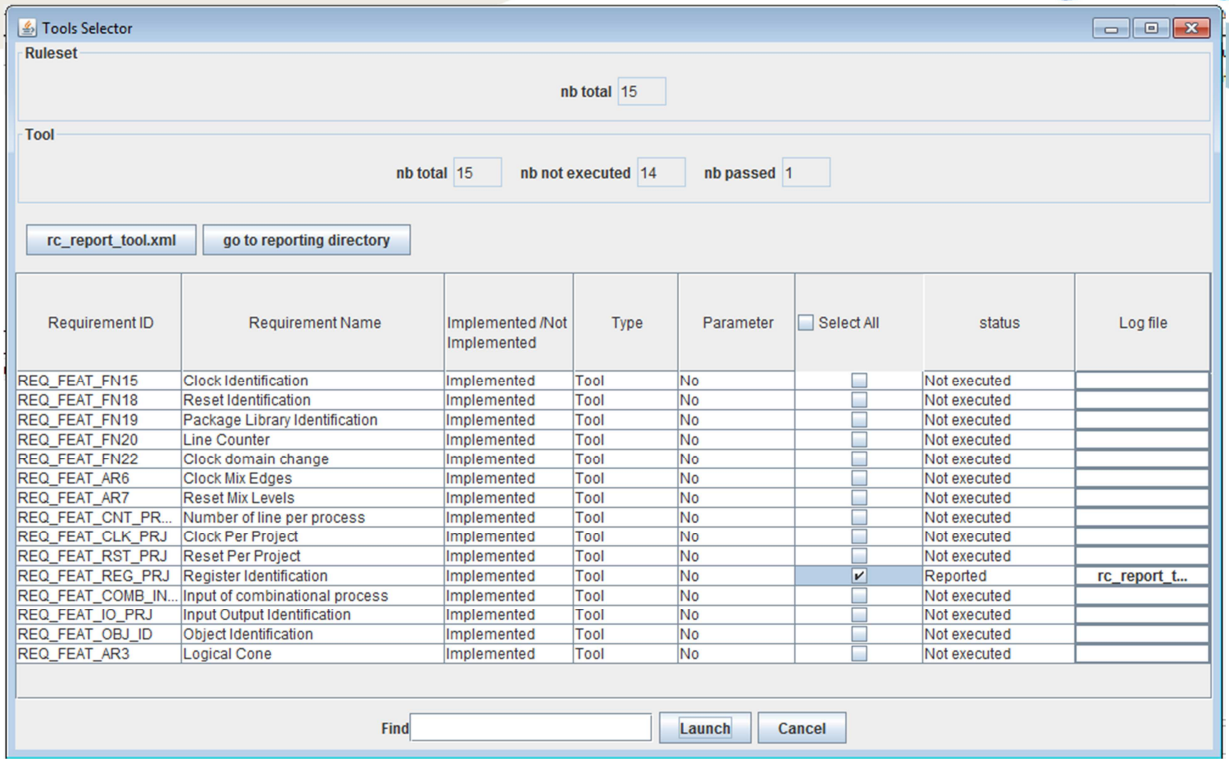
- Detailed Description: With the help on the clock identification tool (see SRS_REQ_FEAT_FN15), all the synchronous process are listed.
- All signals listed after the clock use (rising_edge(clk), falling_edge(clk), etc.) until next `end if;` statement are seen as registers.
- There are different types of registers:
    - Discrete : Std_logic
    - Vector : Std_logic_vector, Array
    - Vector Part : individual use of bit in vector signal
    - Record : Aggregation of heterogenous elements – custom definition
    - Enumeration : List of states – custom definition
- Each register is attached to a synchronous process and thus a clock signal.

The list of parameters reported for each register is given in the synthesis section.

Limitations: at this level none, see parent for any limitation

Expected Result:

When we select the feature Registers Identification, as shown in this figure, the rule checker tool lists all registers of a project.


Tools Selector
Ruleset
nb total 15
Tool
nb total 15
nb not executed 14
nb passed 1
rc_report_tool.xml
go to reporting directory
Requirement ID
Requirement Name
Implemented /Not Implemented
Type
Parameter
Select All
status
Log file
REQ_FEAT_FN15 Clock Identification Implemented Tool No Not executed
REQ_FEAT_FN18 Reset Identification Implemented Tool No Not executed
REQ_FEAT_FN19 Package Library Identification Implemented Tool No Not executed
REQ_FEAT_FN20 Line Counter Implemented Tool No Not executed
REQ_FEAT_FN22 Clock domain change Implemented Tool No Not executed
REQ_FEAT_AR6 Clock Mix Edges Implemented Tool No Not executed
REQ_FEAT_AR7 Reset Mix Levels Implemented Tool No Not executed
REQ_FEAT_CNT_PR... Number of line per process Implemented Tool No Not executed
REQ_FEAT_CLK_PRJ Clock Per Project Implemented Tool No Not executed
REQ_FEAT_RST_PRJ Reset Per Project Implemented Tool No Not executed
REQ_FEAT_REG_PRJ Register Identification Implemented Tool No Reported rc_report_t...
REQ_FEAT_COMB_IN... Input of combinational process Implemented Tool No Not executed
REQ_FEAT_IO_PRJ Input Output Identification Implemented Tool No Not executed
REQ_FEAT_OBJ_ID Object Identification Implemented Tool No Not executed
REQ_FEAT_AR3 Logical Cone Implemented Tool No Not executed
Find
Launch
Cancel

I. Input of combinational process (SRS_REQ_FEAT_COMB_INPUT)

Rationale: The objective aims to list all inputs of combinational processes in order to check with another requirement whether the sensitivity list is complete.

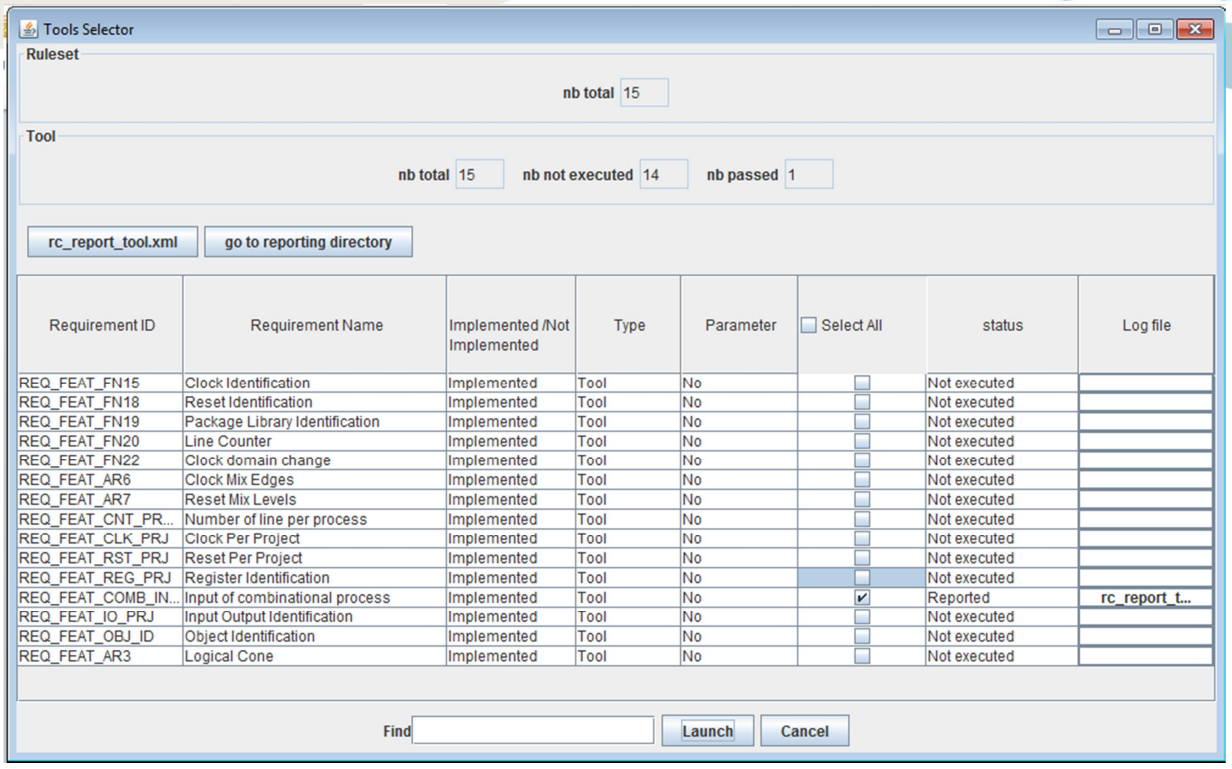
- Detailed Description: With the help on the clock identification tool (see SRS_REQ_FEAT_FN15), all the combinational processes are identified.
- Within the body of the process, all signals used at the right side of "<=" or ":=" statements are considered as inputs.
- Similarly, all signals used in the conditional expression (if, elsif, etc) or case are also considered as inputs.
- There are different types of inputs:
    - Discrete : Std_logic
    - Vector : Std_logic_vector, Array
    - Vector Part : individual use of bit in vector signal
    - Record : Aggregation of heterogenous elements – custom definition
    - Enumeration : List of states – custom definition

The list of parameters reported for each input is given in the synthesis section.

Limitations: at this level none, see parent for any limitation.

Expected Result:

When we select the feature Input of combinational process, as shown in this figure, the rule checker tool lists all the inputs of combinational processes.


Tools Selector
Ruleset
nb total 15
Tool
nb total 15
nb not executed 14
nb passed 1
rc_report_tool.xml
go to reporting directory
Requirement ID
Requirement Name
Implemented /Not Implemented
Type
Parameter
Select All
status
Log file
REQ_FEAT_FN15 Clock Identification Implemented Tool No Not executed
REQ_FEAT_FN18 Reset Identification Implemented Tool No Not executed
REQ_FEAT_FN19 Package Library Identification Implemented Tool No Not executed
REQ_FEAT_FN20 Line Counter Implemented Tool No Not executed
REQ_FEAT_FN22 Clock domain change Implemented Tool No Not executed
REQ_FEAT_AR6 Clock Mix Edges Implemented Tool No Not executed
REQ_FEAT_AR7 Reset Mix Levels Implemented Tool No Not executed
REQ_FEAT_CNT_PR... Number of line per process Implemented Tool No Not executed
REQ_FEAT_CLK_PRJ Clock Per Project Implemented Tool No Not executed
REQ_FEAT_RST_PRJ Reset Per Project Implemented Tool No Not executed
REQ_FEAT_REG_PRJ Register Identification Implemented Tool No Not executed
REQ_FEAT_COMB_IN... Input of combinational process Implemented Tool No Reported rc_report_t...
REQ_FEAT_IO_PRJ Input Output Identification Implemented Tool No Not executed
REQ_FEAT_OBJ_ID Object Identification Implemented Tool No Not executed
REQ_FEAT_AR3 Logical Cone Implemented Tool No Not executed
Find
Launch
Cancel

m. Input output identification (SRS_REQ_FEAT_IO_PRJ)

Rationale: The objective is to list all Input/output in the project.

Detailed Description: For each component in VLSI project, list all Input/output.

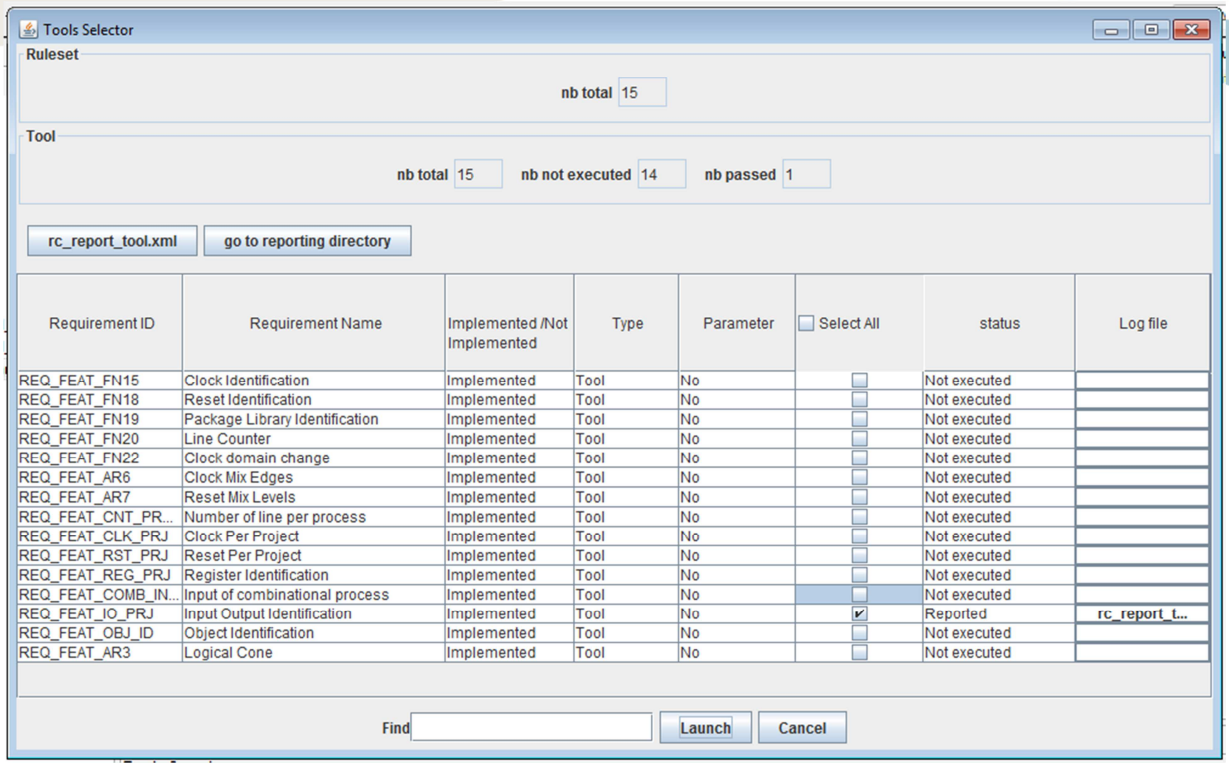
In this example, the function list the signals appearing in the “port” element: “Clk_20MHZ”, “RESET_N”, … as input/output object.

```
library IEEE;
use IEEE.Std_Logic_1164.all;


entity RAZWDG is

   port(

            -- Horloge et reset

            Clk_20MHZ          : in  Std_Logic;
            RESET_N            : in  Std_Logic;

            CS_WDG             : in  Std_Logic
            Data_in            : in  Std_Logic;
            RAZ_WDG            : out  Std_Logic
            );
```

Limitations: none.

Expected Result:

When we select the feature Input/Output idnetification, as shown in this figure, the rule checker tool lists all the Input/Output of modules with VLSI project.


Tools Selector
Ruleset
nb total 15
Tool
nb total 15
nb not executed 14
nb passed 1
rc_report_tool.xml
go to reporting directory
Requirement ID
Requirement Name
Implemented /Not Implemented
Type
Parameter
Select All
status
Log file
REQ_FEAT_FN15 Clock Identification Implemented Tool No Not executed
REQ_FEAT_FN18 Reset Identification Implemented Tool No Not executed
REQ_FEAT_FN19 Package Library Identification Implemented Tool No Not executed
REQ_FEAT_FN20 Line Counter Implemented Tool No Not executed
REQ_FEAT_FN22 Clock domain change Implemented Tool No Not executed
REQ_FEAT_AR6 Clock Mix Edges Implemented Tool No Not executed
REQ_FEAT_AR7 Reset Mix Levels Implemented Tool No Not executed
REQ_FEAT_CNT_PR... Number of line per process Implemented Tool No Not executed
REQ_FEAT_CLK_PRJ Clock Per Project Implemented Tool No Not executed
REQ_FEAT_RST_PRJ Reset Per Project Implemented Tool No Not executed
REQ_FEAT_REG_PRJ Register Identification Implemented Tool No Not executed
REQ_FEAT_COMB_IN... Input of combinational process Implemented Tool No Not executed
REQ_FEAT_IO_PRJ Input Output Identification Implemented Tool No Reported rc_report_t...
REQ_FEAT_OBJ_ID Object Identification Implemented Tool No Not executed
REQ_FEAT_AR3 Logical Cone Implemented Tool No Not executed
Find
Launch
Cancel

n. Object identification (SRS_REQ_FEAT_OBJ_ID)

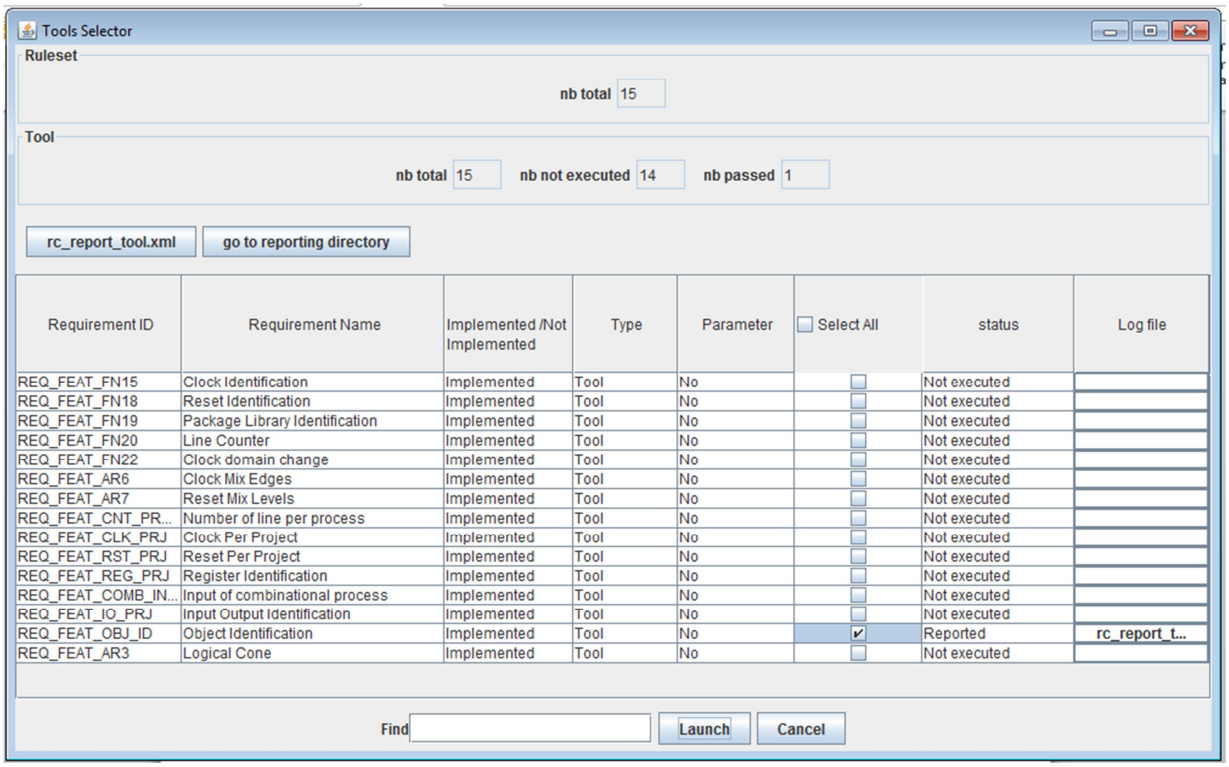
Rationale: The objective is to list all objects in the project.

Detailed Description: For each component in VLSI project, list all Input/output, for each synchronous process list all register and for each asynchronous process list all combinational input. For more details, see the specific function.

Limitations: at this level none, see parent for any limitation.

Expected Result:

When we select the feature Object Identification, as shown in this figure, the rule checker tool lists all the objects of a project (I/O, registers, input of combinatory).



Requirement ID	Requirement Name	Implemented /Not Implemented	Type	Parameter	Select All	status	Log file
REQ_FEAT_FN15	Clock Identification	Implemented	Tool	No		Not executed	
REQ_FEAT_FN18	Reset Identification	Implemented	Tool	No		Not executed	
REQ_FEAT_FN19	Package Library Identification	Implemented	Tool	No		Not executed	
REQ_FEAT_FN20	Line Counter	Implemented	Tool	No		Not executed	
REQ_FEAT_FN22	Clock domain change	Implemented	Tool	No		Not executed	
REQ_FEAT_AR6	Clock Mix Edges	Implemented	Tool	No		Not executed	
REQ_FEAT_AR7	Reset Mix Levels	Implemented	Tool	No		Not executed	
REQ_FEAT_CNT_PR...	Number of line per process	Implemented	Tool	No		Not executed	
REQ_FEAT_CLK_PRJ	Clock Per Project	Implemented	Tool	No		Not executed	
REQ_FEAT_RST_PRJ	Reset Per Project	Implemented	Tool	No		Not executed	
REQ_FEAT_REG_PRJ	Register Identification	Implemented	Tool	No		Not executed	
REQ_FEAT_COMB_IN...	Input of combinational process	Implemented	Tool	No		Not executed	
REQ_FEAT_IO_PRJ	Input Output Identification	Implemented	Tool	No		Not executed	
REQ_FEAT_OBJ_ID	Object Identification	Implemented	Tool	No	✔	Reported	rc_report_t...
REQ_FEAT_AR3	Logical Cone	Implemented	Tool	No		Not executed	

o. Number of Line per process (SRS_REQ_FEAT_CNT)

Rationale: The objective is to count the number of lines per process so as to identify which process is complex.

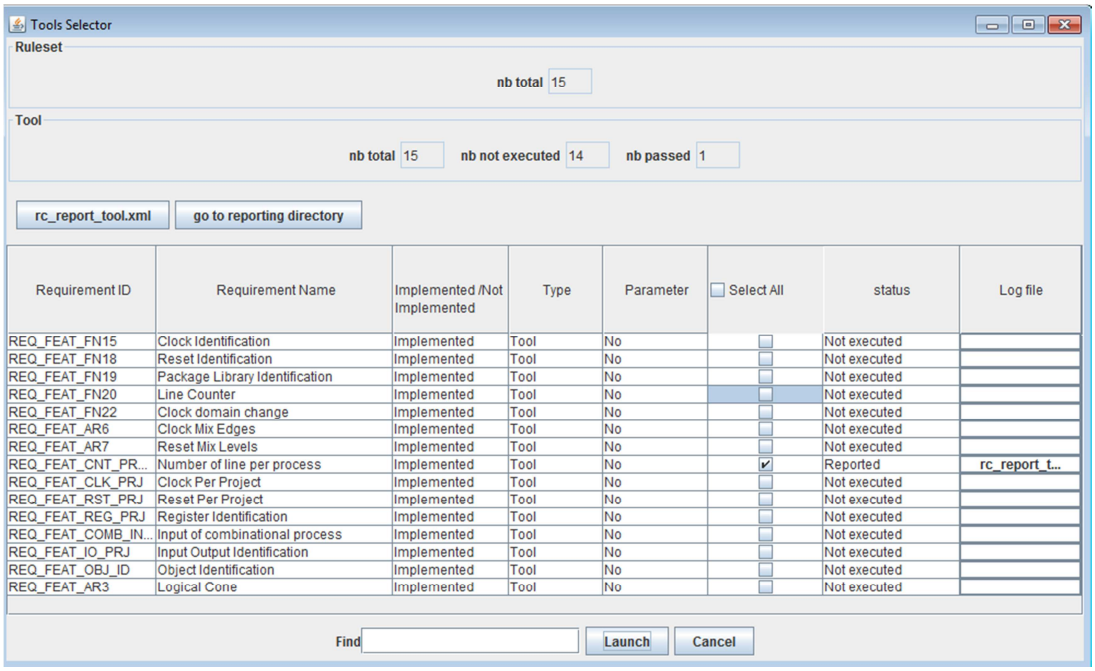
Detailed Description:

- With the help of AST of ZamiaCad, all the processes are identified
- For each one, the number of line is counted between the declaration of the process and the last statement in the process.
- A report lists for each process listed in the VLSI the number of lines
- The list of parameters reported for each process is given in the synthesis section

Limitations: The measured line number is not strictly the line number of the process. The figure is a bit lower because the number of lines is given between process declaration and latest statement.

Expected Result:

When we select the feature Number of Line per process, as shown in this figure, the rule checker tool lists all the process and give the number of line for each one.



We can see the result in rc_report_rule_REQ_FEAT_CNT_PROC_Number of line per process.xml file. In this example, the file “pc_next.vhd”, contains a process “PC_NEXT” of 31 lines

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<REQ_FEAT_CNT_PROC>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_FEAT_CNT_PROC</description>
  <creationDate>Fri Apr 15 16:17:32 CEST 2016</creationDate>
  <file>
    <fileName>\pc_next.vhd</fileName>
    <fileNbLine>68</fileNbLine>
    <entity>
      <entityName>PC_NEXT</entityName>
      <entityLoc>16</entityLoc>
      <architecture>
        <architectureName>LOGIC</architectureName>
        <architectureLoc>28</architectureLoc>
        <process>
          <processName>PC_NEXT</processName>
          <processLoc>34</processLoc>
          <processNbLine>31</processNbLine>
        </process>
      </architecture>
    </entity>
  </file>
```

2. Rules Selector

a. Help

(1) Primitive Isolation (SRS_REQ_STD_01800)

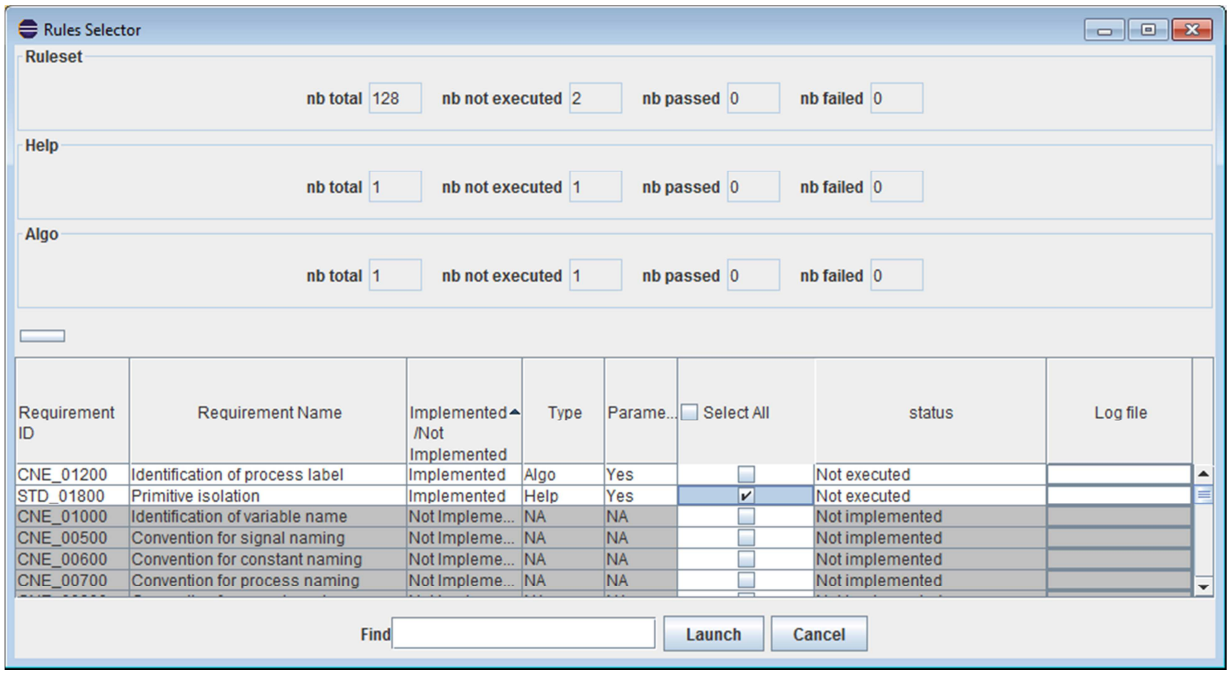
Rationale: The objective is to help reviewers to identify which VHDL files declare libraries other than IEEE (eg ALTERAMF, AXCELERATOR) and verify if concerned files are correctly isolated from the rest of the project.

Detailed Description: The VHDL TOOL shall report the libraries other than IEEE declared in all VHDL files of the VLSI project.

Limitations: None.

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_rule_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>
  <version>V1.3 date 21/09/2015</version>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule REQ_STD_01800</description>
  <creationDate>Tue Sep 22 14:01:05 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>
```

(2) Reset assertion and deassertion (SRS_REQ_STD_03700)

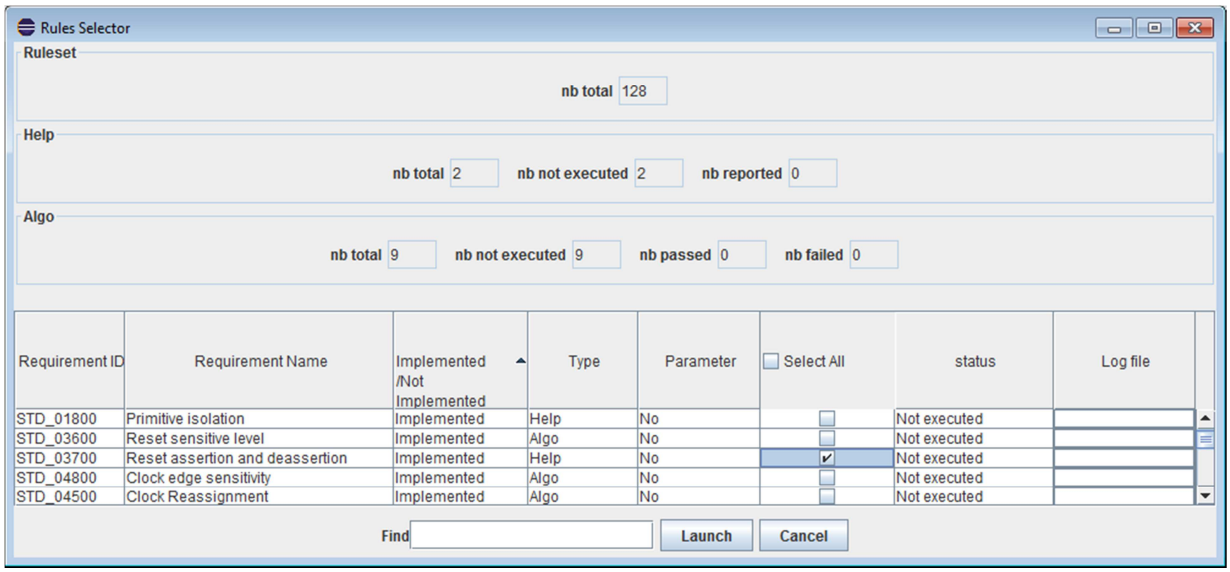
Rationale: The objective is to return reset sources so that user can check if reset sources are correctly generated.

Detailed Description: with the help of the "SRS_REQ_FEAT_RST_PRJ" function, a report is built containing all reset sources of VLSI project. 3 possible cases may be identified: input port, instance output, assignment.

Limitations: At this level, this is a rule help so it is user responsibility to judge according to reset source report and actual design whether the reset are correctly generated.

Expected Result:

When we select the feature Reset assertion and deassertion, as shown in this figure, the rule checker tool lists all reset sources of VLSI project



We can see the result of this detection in rc_report_rule_REQ_STD_03700_Reset assertion and deassertion.xml file. In this example, the reset source "Y" is an input port in entity "QCLKDRIVER".

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_03700>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>report for rule STD_03700</description>
  <ruleName>Reset Assertion and Deassertion</ruleName>
  <creationDate>Fri Nov 27 16:21:28 CET 2015</creationDate>
  <resetSource>
    <fileName>FPGA_FM_Datapackage\02_Code\DPULogic\Definitions\ModuleMap_pkg.vhd</fileName>
    <entityName>QCLKDRIVER</entityName>
    <architectureName>DPULOGIC_STR</architectureName>
    <resetSourceName>Y</resetSourceName>
    <resetSourceType>Input Port</resetSourceType>
    <resetSourceLoc>444</resetSourceLoc>
  </resetSource>
```

(1) Identification of Process Label (SRS_REQ_CNES_01200)

Rationale: The objective is to check that all processes are correctly labeled.

Detailed Description: The requirement SRS_REQ_CNES_01200 is based on a generic requirement (GEN_01200).

This generic requirement is responsible of, with the help of the native AST from ZamiaCad, listing all processes within VLSI project, and identifying the label name if there is any.

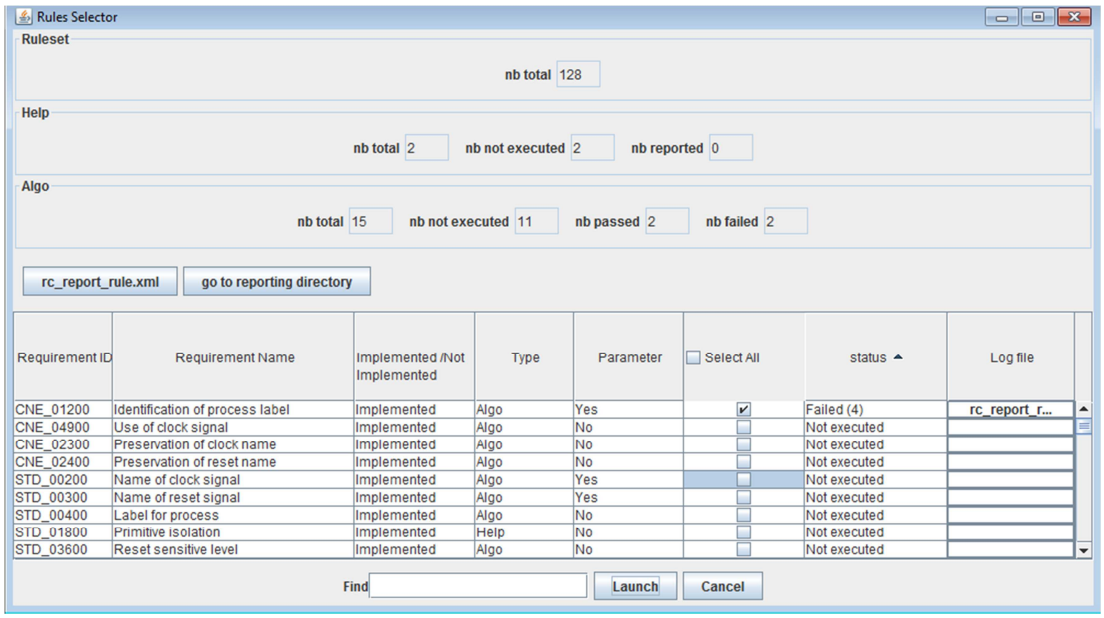
The label name is then compared with the generic parameter defined in the rc_handbook_parameters XML file.

For SRS_REQ_CNES_01200, the generic parameter is a prefix that consists of “P_”.

Limitations: none at this level.

Expected Result:

When we select the feature Identification of Process Label, as shown in this figure, the rule checker tool lists all process which have not the prefix “P_”.



If there is no label present or if the label doesn't match generic parameter, then a violation is reported indicating the type of error: MissingLabel or BadLabelName.

The parameters listed in the violation report are described in the synthesis section.We can see the result in rc_report_rule_CNE_01200_Identification_of_process_label.xml file. In this example, the process "PC_NEXT" hasn't got the prefix "P_".

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<CNE_01200>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule CNE_01200</description>
  <ruleName>Identification of process label</ruleName>
  <creationDate>Fri Apr 15 16:32:11 CEST 2016</creationDate>
  <process>
    <violationType>invalid process label</violationType>
    <processName>PC_NEXT</processName>
    <processLoc>
      <fileName>\pc_next.vhd</fileName>
      <entityName>PC_NEXT</entityName>
      <architectureName>LOGIC</architectureName>
      <processLoc>34</processLoc>
    </processLoc>
  </process>
```

(2) Preservation of clock name (SRS_REQ_CNE_02300)

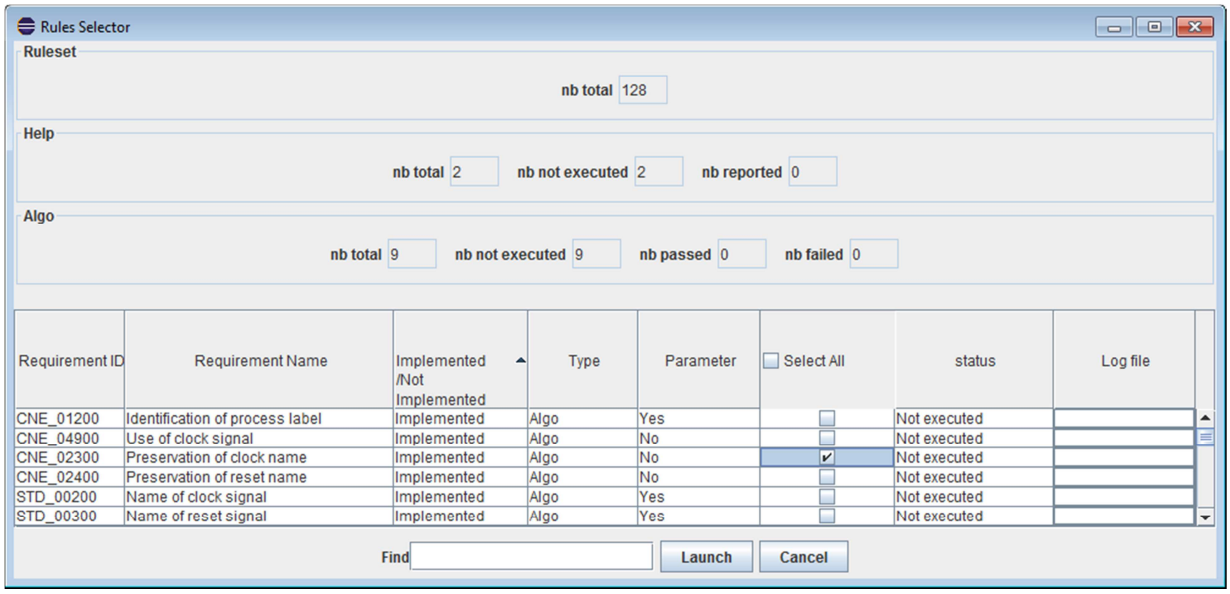
Rationale: The objective is to check that clock does not change name when it is connected to components.

Detailed Description: Search Clock VHDL elements in VLSI project with "Clock per project" function, and verify that the signal name does not change when entering components at lower level.

Limitations: none at this level.

Expected Result:

When we select the feature Preservation of clock name, as shown in this figure, the rule checker tool list clock signal are used inside combinatory function.



When a violation is detected, a report shall be established with following information:

- Entity name
- Architecture name
- File name
- Clock name before component
- Clock name after component

- Line number of name change
- Component instance name

We can see the result of this detection in rc_report_rule_REQ_CNE_02300_Preservation of clock name.xml file. In this example, the clock signal “WR” change name for “EEP_WRITE” when the signal go in the component “EEPD” in file “AddressDecoder.vhd”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<CNE_02300>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule CNE_02300</description>
  <ruleName>Preservation of Clock Name</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <clockSignalName>
    <violationType>noPreservationName</violationType>
    <entityName>ADDRESSDECODER</entityName>
    <architectureName>ADRESSDECODER_STR</architectureName>
    <fileName>FPGA_FM_Datapackage\02_Code\DPULogic\AddressDecoder\AddressDecoder.vhd</fileName>
    <clockSignalNameBefore>EEP_WRITE</clockSignalNameBefore>
    <clockSignalNameAfter>WR</clockSignalNameAfter>
    <componentName>EEPD</componentName>
    <mapLoc>94</mapLoc>
  </clockSignalName>
```

(3) Preservation of reset name (SRS_REQ_CNE_02400)

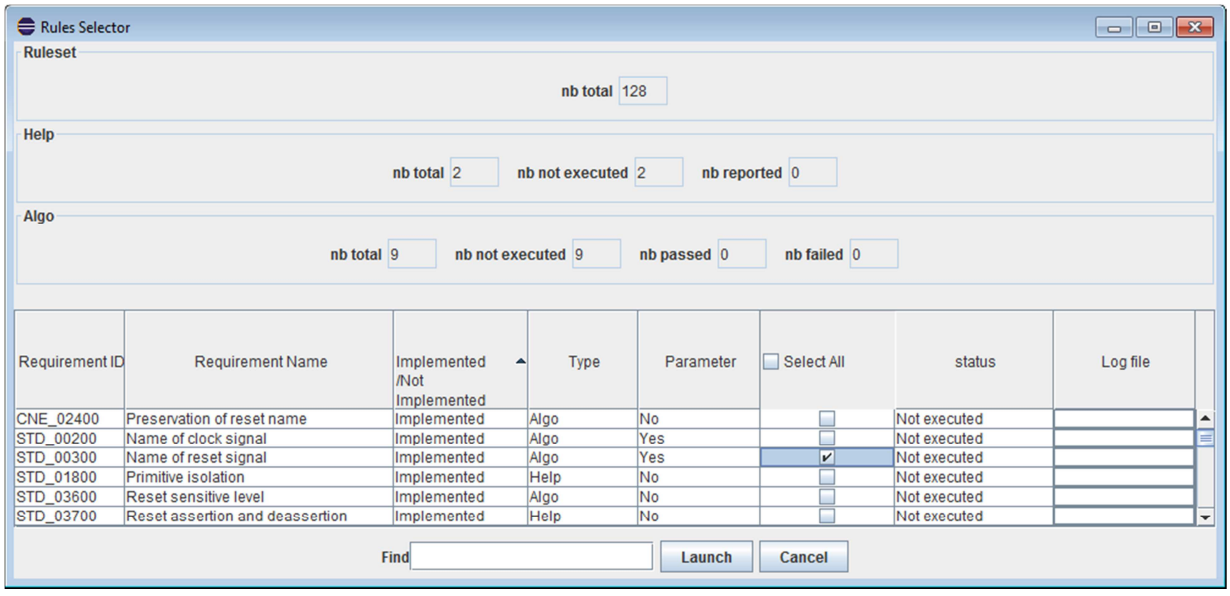
Rationale: The objective is to check that reset does not change name when it is connected to components.

Detailed Description: Search Reset VHDL elements in VLSI project with "Reset identification" function, and verify that the signal name does not change when entering components at lower level.

Limitations: none at this level.

Expected Result:

When we select the feature Preservation of reset name, as shown in this figure, the rule checker tool list clock signal are used inside combinatory function.



When a violation is detected, a report shall be established with following information:

- Entity name
- Architecture name
- File name
- reset name before component
- reset name after component
- Line number of name change
- Component instance name

We can see the result of this detection in rc_report_rule_REQ_CNE_02400_Preservation of reset name.xml file. In this example, the reset signal “WR_EN” change name for “RD_COMMAND” when the signal go in the component “CF” in file “SISHandler.vhd”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<CNE_02400>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule CNE_02400</description>
  <ruleName>Preservation of Reset Name</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <resetSignalName>
    <violationType>noPreservationName</violationType>
    <entityName>SISHANDLER</entityName>
    <architectureName>SISHANDLER_STR</architectureName>
    <fileName>FPGA_FM_Datapackage\02_Code\DPULogic\SISHandler\SISHandler.vhd</fileName>
    <resetSignalNameBefore>RD_COMMAND</resetSignalNameBefore>
    <resetSignalNameAfter>WR_EN</resetSignalNameAfter>
    <instanceName>CF</instanceName>
    <mapLoc>91</mapLoc>
  </resetSignalName>
</CNE_02400>
```

(4) Use of clock signal (SRS_REQ_CNE_04900).

Rationale: The objective is to check that clock signals are not used inside combinatory function.

Detailed Description: Search Clock Source elements in VLSI project with "SRS_REQ_FEAT_CLK_PRJ" function. Then, launch for each clock source a search to know all uses of each clock source. Check that the search only returns:

- passages to lower levels of components,
- use in edge detection
- an output port of the FPGA,
- input port of black box (PLL for example)

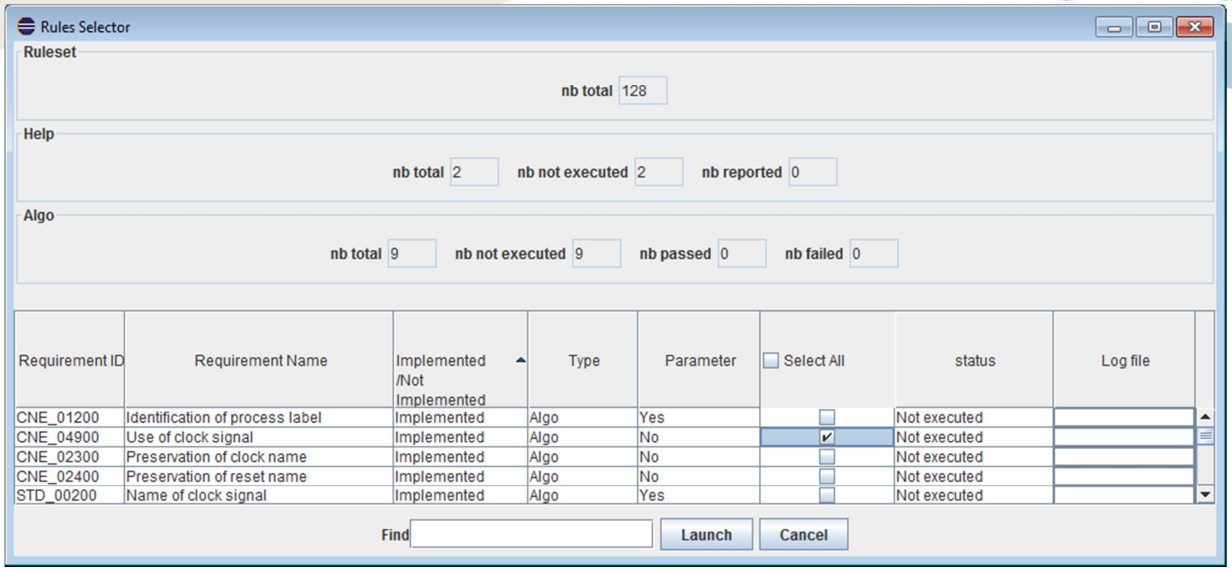
For all the other cases (logical equation, usage in the synchronous part of a process, etc.), a violation report shall be established with following information for all violated paths:

- clock source
- entity name
- architecture name
- filename
- line number

Limitations: At this level, the reason of violation is not explained. The user has to jump to the corresponding line number and analyses origin of problem.

Expected Result:

When we select the feature Use of clock signal, as shown in this figure, the rule checker tool list clock signals that are used inside combinatory function.



We can see the result of this detection in rc_report_rule_REQ_CNE_04900_Use of clock signal.xml file. In this example, the source clock "Y" is used in a logical equation in file "EEPDecoder.vhd" in line 109.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<CNE_04900>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule CNE_04900</description>
  <ruleName>Use of clock signal</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <clockSource>
    <violationType>wrong uses clock</violationType>
    <clockSourceName>Y</clockSourceName>
    <fileName>FPGA_FM_Datapackage\02_Code\DPULogic\AddressDecoder\EEPDecoder.vhd</fileName>
    <entityName>EEPDECODER</entityName>
    <architectureName>EEPDECODER_RTL</architectureName>
    <clockSignalLoc>109</clockSignalLoc>
  </clockSource>
```

(5) Name of clock signal (SRS_REQ_STD_00200)

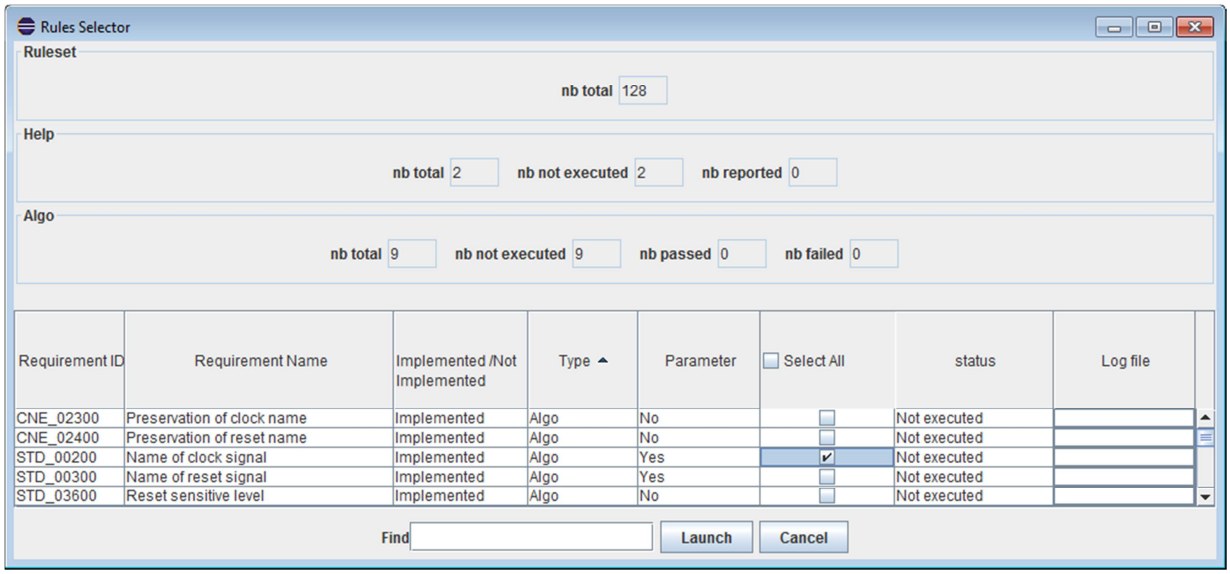
Rationale: The objective is to check that clock signal name includes “clk” or “clock” (for example).

Detailed Description: Search Clock VHDL elements in VLSI project with “Clock identification” function. Verify that the signal name contains string “Clk” or “clock".

Limitations: none at this level

Expected Result:

When we select the feature Name of clock signal, as shown in this figure, the rule checker tool lists clock signals that are badly named.



When a violation is detected, a report shall be established with following information:

- Clock name
- Entity name
- Architecture name
- File name
- Line number where clock name is badly written

We can see the result of this detection in rc_report_rule_REQ_STD_00200_Name of clock signal.xml file. In this example, the clock signal “c” do not respect the prefix “clk” or “clock” in file “ModuleMap_pkg.vhd”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_00200>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_00200</description>
  <ruleName>Name of clock signal</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <clockSignal>
    <violationType>nameInvalid</violationType>
    <clockSignalName>C</clockSignalName>
    <clockSignalLoc>
      <fileName>\FPGA_FM_Datapackage\02_Code\DPULogic\Definitions\ModuleMap_pkg.vhd</fileName>
      <entityName>CLKOUT</entityName>
      <architectureName>CLKOUT_RTL</architectureName>
      <processName>CLOCK</processName>
      <clockSignalLoc>271</clockSignalLoc>
    </clockSignalLoc>
  </clockSignal>
```

(6) Name of reset signal (SRS_REQ_STD_00300)

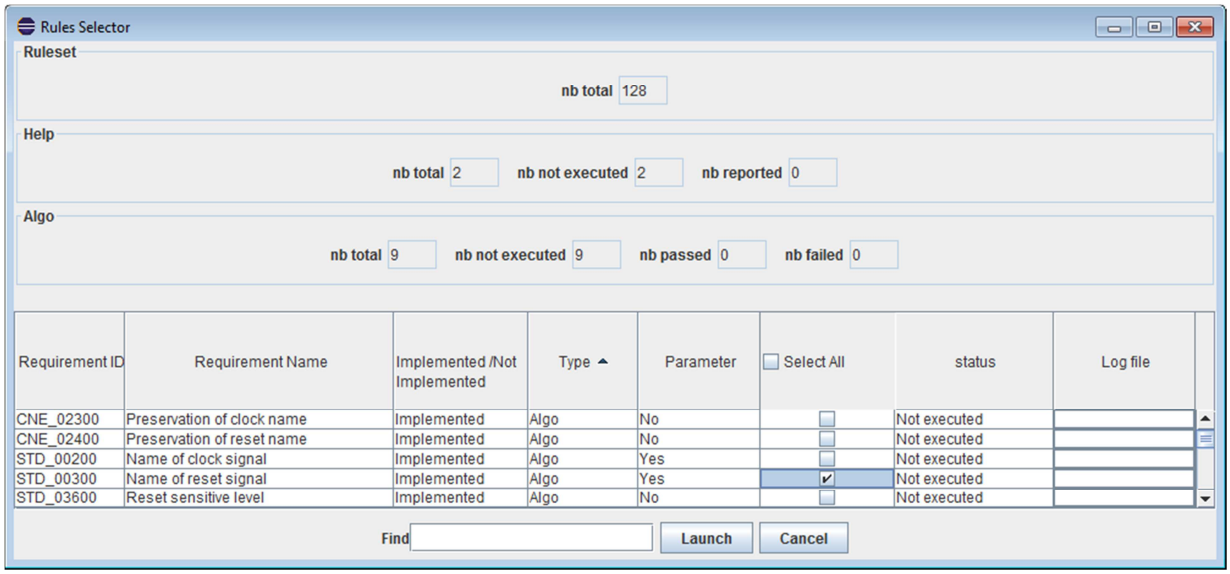
Rationale: The objective is to check that reset signal name includes “rst”, “reset” or “clr” (for example).

Detailed Description: Search Reset VHDL elements in VLSI project with “Reset identification” function. Verify that the signal name contains string “rst”, “reset” or “clr”.

Limitations: none at this level

Expected Result:

When we select the feature Name of reset signal, as shown in this figure, the rule checker tool lists reset signals that are badly named.



When a violation is detected, a report shall be established with following information:

- Reset name
- Entity name
- Architecture name
- File name
- Line number where reset name is badly written

We can see the result of this detection in rc_report_rule_REQ_STD_00300_Name of reset signal.xml file. In this example, the reset signal “DIR_CLK” do not respect the prefix “rst” or “reset” in file “DCFIFO.vhd”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_00300>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_00300</description>
  <ruleName>Name of reset signal</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <resetSignal>
    <violationType>nameInvalid</violationType>
    <resetSignalName>WR_SEQ.WR_HOLD</resetSignalName>
    <resetSignalLoc>
      <fileName>\FPGA_FM_Datapackage\02_Code\DPULogic\SISHandler\DCFIFO.vhd</fileName>
      <entityName>DCFIFO</entityName>
      <architectureName>DCFIFO_RTL</architectureName>
      <processName>SET</processName>
      <resetSignalName>DIR_CLK</resetSignalName>
      <resetSignalLoc>201</resetSignalLoc>
    </resetSignalLoc>
  </resetSignal>
```

(7) Label of process (SRS_REQ_STD_00400)

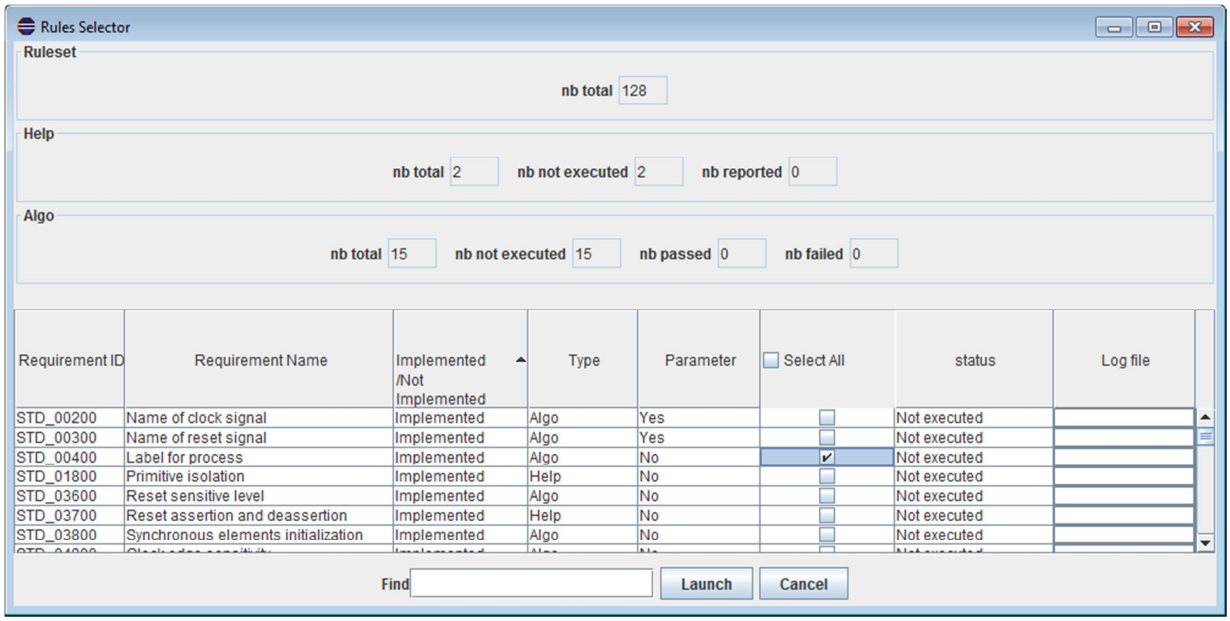
Rationale: The objective is to check that all processes are labeled within the VLSI project.

Detailed Description: Search process VHDL elements in VLSI project. Verify that the process label exists.

Limitations: none at this level

Expected Result:

When we select the feature Name of reset signal, as shown in this figure, the rule checker tool lists processes without label.



When a violation is detected, a report shall be established with following information:

- Entity name
- Architecture name
- File name
- Line number where process label is missing

We can see the result of this detection in rc_report_rule_STD_00400_Label for Process.xml file. In this example, some process labels in file "\src\AES\key_expander.vhd" are missed.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_00400>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_00400</description>
  <ruleName>Label for Process</ruleName>
  <creationDate>Mon Mar 14 11:14:13 CET 2016</creationDate>
  <process>
    <violationType>noLabelForProcess</violationType>
    <fileName>\src\AES\key_expander.vhd</fileName>
    <entityName>KEY_EXPANDER</entityName>
    <architectureName>EXPANSION</architectureName>
    <processLoc>118</processLoc>
  </process>
  <process>
    <violationType>noLabelForProcess</violationType>
    <fileName>\src\AES\key_expander.vhd</fileName>
    <entityName>KEY_EXPANDER</entityName>
    <architectureName>EXPANSION</architectureName>
    <processLoc>194</processLoc>
  </process>
```

(8) Reset sensitivity level (SRS_REQ_STD_03600)

Rationale: The objective is to check that all reset sources uses the same activation level. The detection must be done per entity and per VLSI project.

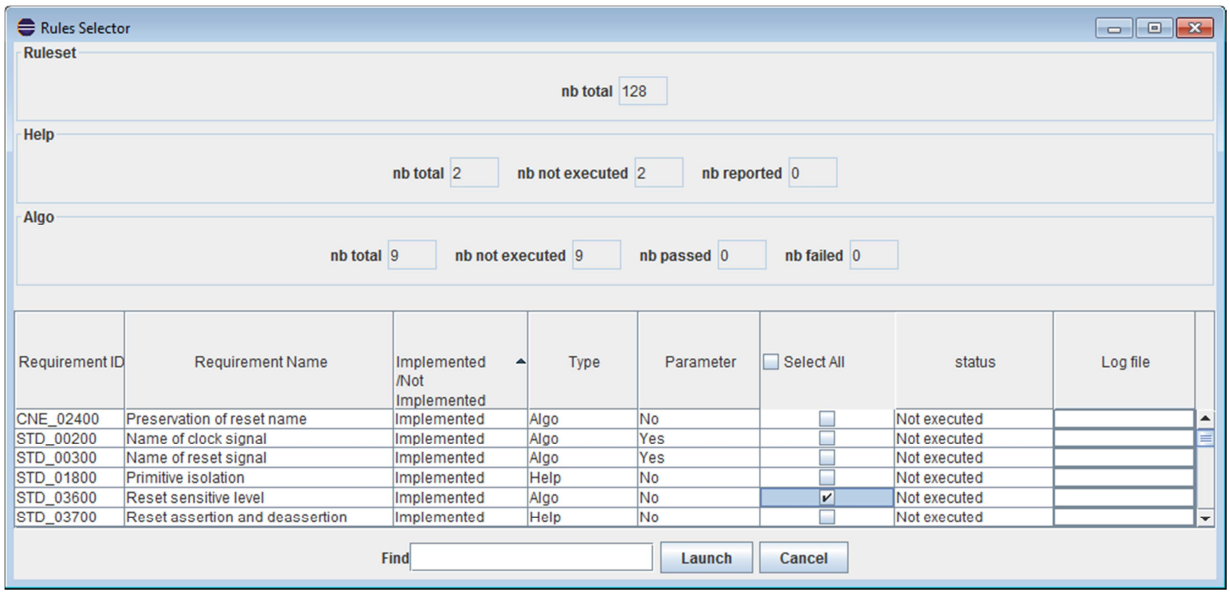
Detailed Description: Search Reset VHDL elements in VLSI project with "Reset identification" function, and verify that the reset signals are always used with the same activation level in each file. When reset sources are used on 2 different levels, a violation report is built showing the different usage of all reset sources in the file.

The same detection mechanism is done at VLSI project level. When reset sources are used on different levels between entities, then a violation report is done and all files using reset sources are reported with the corresponding level.

Limitations: none at this level

Expected Result:

When we select the feature Reset sensitivity level, as shown in this figure, the rule checker tool list all resets when there are some different sensitivity levels for reset.



We can see the result of this detection in rc_report_rule_REQ_STD_03600_Reset sensitivity level.xml file. In this example, the reset source tag "resetSourceF" is used in both level in file "DCFIFO.vhd".

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_03600>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_03600</description>
  <ruleName>Reset Sensitive Level</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <entity>
    <violationType>mixLevelPerProject</violationType>
    <fileName>\FPGA_FM_Datapackage\02_Code\DPULogic\SISHandler\DCFIFO.vhd</fileName>
    <entityName>DCFIFO</entityName>
    <resetSourceATag>resetSourceA</resetSourceATag>
    <resetSourceALevel>low</resetSourceALevel>
    <resetSourceBTag>resetSourceB</resetSourceBTag>
    <resetSourceBLevel/>
    <resetSourceCTag>resetSourceC</resetSourceCTag>
    <resetSourceCLevel/>
    <resetSourceDTag>resetSourceD</resetSourceDTag>
    <resetSourceDLevel/>
    <resetSourceETag>resetSourceE</resetSourceETag>
    <resetSourceELevel>high</resetSourceELevel>
    <resetSourceFTag>resetSourceF</resetSourceFTag>
    <resetSourceFLevel>both</resetSourceFLevel>
    <resetSourceGTag>resetSourceG</resetSourceGTag>
    <resetSourceGLevel>low</resetSourceGLevel>
    <resetSourceHTag>resetSourceH</resetSourceHTag>
    <resetSourceHLevel>low</resetSourceHLevel>
  </entity>
```

(10) Synchronous elements initialization (SRS_REQ_STD_03800)

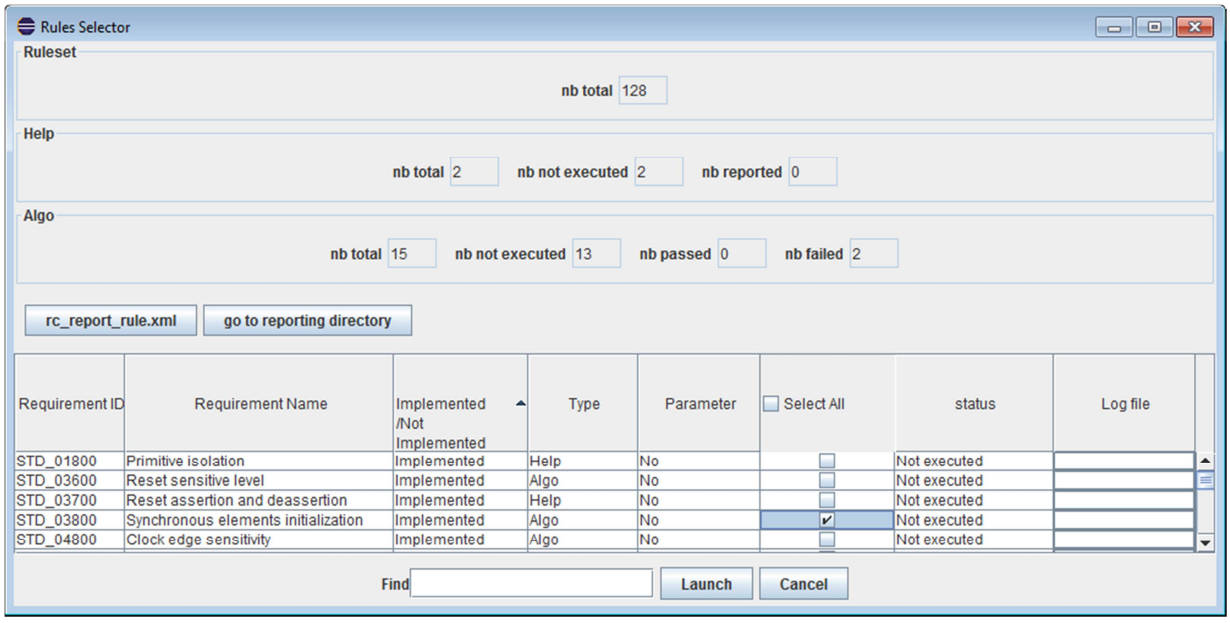
Rationale: The objective is to return uninitialized registers so that user can add the initialization for this registers.

Detailed Description: with the help of the "SRS_REQ_FEAT_REG_PRJ" function, a report is built containing all registers of VLSI project and with the help of the "SRS_REQ_FEAT_FN18" function, a report is built containing all synchronous process of VLSI project. 2 possible cases may be identified: registers are initialized but not used and registers are not initialized but used, these cases are reported for the users.

Limitations: none at this level, see parent for any limitation.

Expected Result:

When we select the feature synchronous element initialization, as shown in this figure, the rule checker tool list registers not initialized and registers not used.



We can see the result of this detection in rc_report_rule_STD_03800_Synchronous Elements Initialization.xml file. In this example, the register "ARC_MEM_WRD_D(1)" is used but not initialized in process "P_ARC_ARBITER" in file "ARC_MEMORY_CONTROLLER.vhd", and the register "S_HD_FRAME_LENGTH" is

initialized but not used in process “P_SDLS_DATA_RX” in file “TC_FRAME_RECEIVER.vhd”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_03800>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_03800</description>
  <ruleName>Synchronous Elements Initialization</ruleName>
  <creationDate>Mon Mar 14 11:59:47 CET 2016</creationDate>
  <register>
    <violationType>not initialized</violationType>
    <fileName>\src\ARC_MEMORY_CONTROLLER.vhd</fileName>
    <entityName>ARC_MEMORY_CONTROLLER</entityName>
    <architectureName>RTL</architectureName>
    <processName>P_ARC_ARBITER</processName>
    <clockSignalName>CLK</clockSignalName>
    <registerName>ARC_MEM_WRD_D(1)</registerName>
    <registerLoc>514</registerLoc>
  </register>
    <register>
    <violationType>not used</violationType>
    <fileName>\src\TC_FRAME_RECEIVER.vhd</fileName>
    <entityName>TC_FRAME_RECEIVER</entityName>
    <architectureName>RTL</architectureName>
    <processName>P_SDLS_DATA_RX</processName>
    <clockSignalName>CLK</clockSignalName>
    <registerName>S_HD_FRAME_LENGTH</registerName>
    <registerLoc>150</registerLoc>
  </register>
```

(11) Clock reassignment (SRS_REQ_STD_04500)

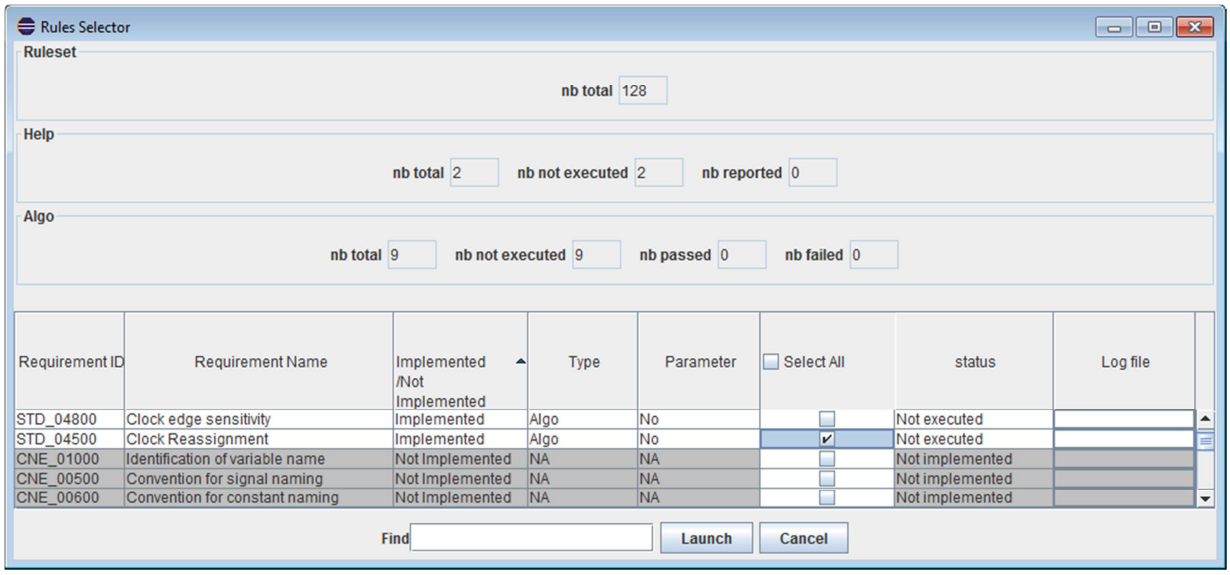
Rationale: The objective is to check that clock signal does not reassign to another signal.

Detailed Description: Search Clock Source elements in VLSI project with "SRS_REQ_FEAT_CLK_PRJ" function, and verify that clock sources are not issued of an assignment. This is symbolized with statement "<=". There should only one element on the right of "<=" symbol.

Limitations: none at this level.

Expected Result:

When we select the feature Clock reassignment, as shown in this figure, the rule checker tool lists clock signals that come from reassignment.



When a violation is detected, a report shall be generated with following information:

- Clock source name
- Entity name
- Architecture name
- File name

- Line number

We can see the result of this detection in rc_report_rule_REQ_STD_04500_clock rassignment.xml file. In this example, the clock source “Y” is assign to ‘1’in file “DCFIFO.vhd”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_04500>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_04500</description>
  <ruleName>Clock Reassignment</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <clockSource>
    <violationType>ClockReassignment</violationType>
    <fileName>FPGA_FM_Datapackage\02_Code\DPULogic\SISHandler\DCFIFO.vhd</fileName>
    <entityName>DCFIFO</entityName>
    <architectureName>DCFIFO_RTL</architectureName>
    <clockSourceName>DIR_CLK</clockSourceName>
    <clockSourceLoc>155</clockSourceLoc>
  </clockSource>
</STD_04500>
```

(12) Clock domain number in the design (SRS_REQ_STD_04600)

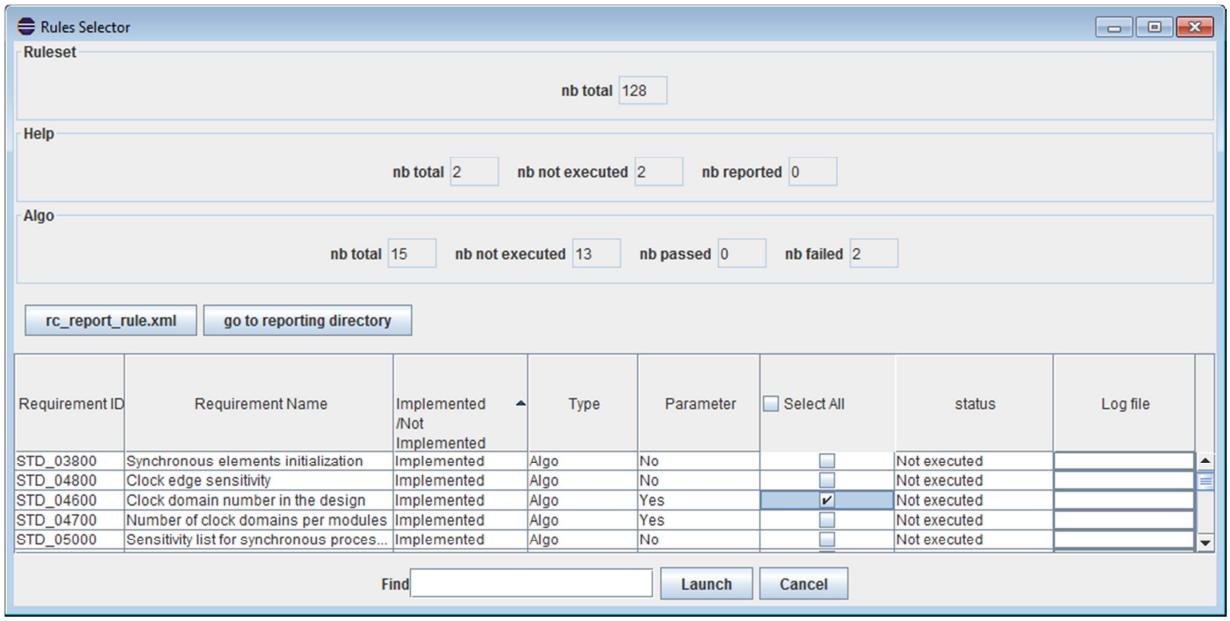
Rationale: The objective is to check that the number of clock source in VLSI project corresponds to the expectations of the user.

Detailed Description: with the help of the "SRS_REQ_FEAT_CLK_PRJ" function, a report is built containing all clock sources of VLSI project. A violation report is built showing all clock sources when this number is higher than the number of expected clock source.

Limitations: at this level none, see parent for any limitation.

Expected Result:

When we select the feature Clock domain number in the design, as shown in this figure, the rule checker tool lists all clock domains.



When a violation is detected, a report shall be generated with following information:

- Clock source name
- Entity name
- Architecture name

- File name
- Line number

We can see the result of this detection in rc_report_rule_STD_04600_Clock domain number in the design.xml file. In this example, the VLSI project use 4 clock source, we can see 2 clock sources in the beginning of the report: “clockSource0” and “clockSource1”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_04600>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_04600</description>
  <ruleName>Clock domain number in the design</ruleName>
  <creationDate>Mon Mar 14 15:07:50 CET 2016</creationDate>
  <clockSource>
    <violationType>numberClockDomaineViolation</violationType>
    <clockSourceTag>clockSource0</clockSourceTag>
    <fileName>src\IP_AUTH_TC.vhd</fileName>
    <entityName>IP_AUTH_TC</entityName>
    <architectureName>RTL</architectureName>
    <clockSourceName>CLK</clockSourceName>
    <clockSourceType>Input Port</clockSourceType>
    <clockSourceLoc>56</clockSourceLoc>
  </clockSource>
  <clockSource>
    <violationType>numberClockDomaineViolation</violationType>
    <clockSourceTag>clockSource1</clockSourceTag>
    <fileName>src\KEY_MEM_ABSTRACTION_BRIDGE.vhd</fileName>
    <entityName>KEY_MEM_ABSTRACTION_BRIDGE</entityName>
    <architectureName>RTL</architectureName>
    <clockSourceName>CLK</clockSourceName>
    <clockSourceType>Input Port</clockSourceType>
    <clockSourceLoc>44</clockSourceLoc>
  </clockSource>
```

(13) Number of clock domains per module (SRS_REQ_STD_04700)

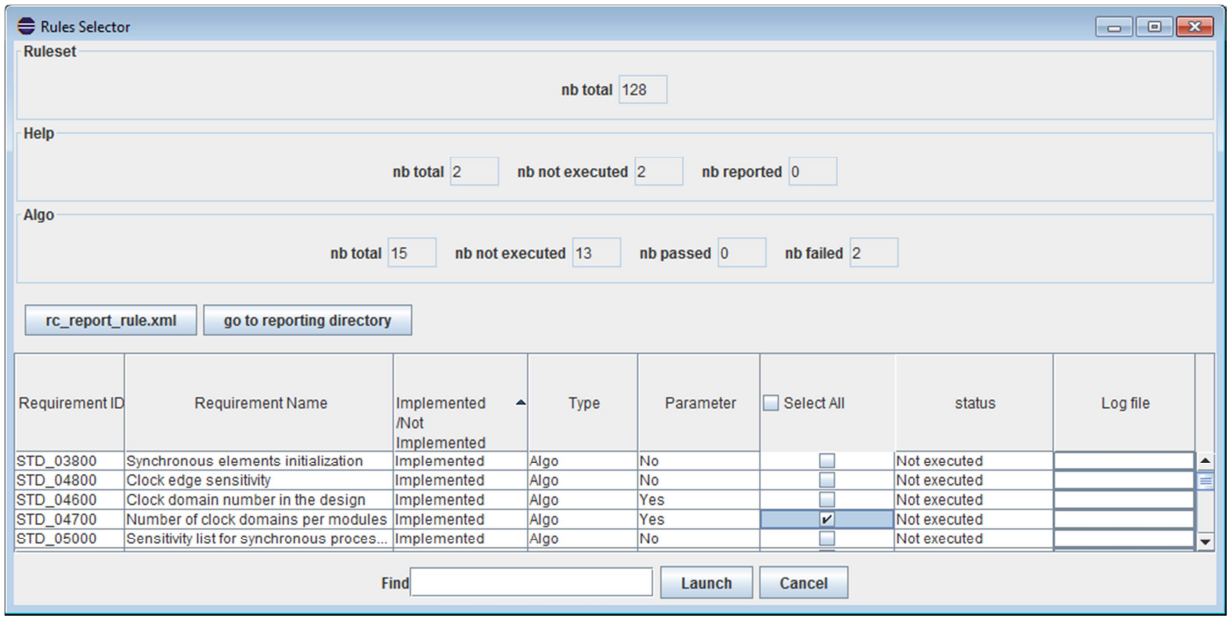
Rationale: The objective is to check that the number of clock sources in each file corresponds to the expectations of the user.

Detailed Description: with the help of the "SRS_REQ_FEAT_CLK_PRJ" function, a report is built containing all clock sources of VLSI project and with the help of the "SRS_REQ_FEAT_FN15" function, a report is built containing all clocks per files with the associated clock source. Rule Checker can build a temporary table with the number of clock source for each file, and a violation report is built showing all clock sources per file when this number is higher than the number of expected clock source.

Limitations: at this level none, see parent for any limitation.

Expected Result:

When we select the feature Number of clock domains per modules, as shown in this figure, the rule checker tool lists clock sources per file.



When a violation is detected, a report shall be generated with following information:

- Clock source name
- Entity name
- Architecture name
- File name
- Line number

We can see the result of this detection in rc_report_rule_STD_04700_Number of clock domains per modules.xml file. In this example, two clock sources “clockSourec0” and “clockSource1” are used in entity “KEY_EXPANDER” of the file “\src\AES\key_expander.vhd”.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_04700>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_04700</description>
  <ruleName>Number of clock domains per modules</ruleName>
  <creationDate>Mon Mar 14 16:05:47 CET 2016</creationDate>
  <entity>
    <violationType>numberClockDomaineViolation</violationType>
    <fileName>\src\AES\key_expander.vhd</fileName>
    <entityName>KEY_EXPANDER</entityName>
    <architectureName>EXPANSION</architectureName>
    <clockSourceTag>clockSource0</clockSourceTag>
    <clockSourceName>CLK</clockSourceName>
    <clockSourceType>Input Port</clockSourceType>
    <clockSourceLoc>56</clockSourceLoc>
  </entity> <entity>
    <violationType>numberClockDomaineViolation</violationType>
    <fileName>\src\AES\key_expander.vhd</fileName>
    <entityName>KEY_EXPANDER</entityName>
    <architectureName>EXPANSION</architectureName>
    <clockSourceTag>clockSource1</clockSourceTag>
    <clockSourceName>CLK_10</clockSourceName>
    <clockSourceType>Input Port</clockSourceType>
    <clockSourceLoc>106</clockSourceLoc>
  </entity>
  <entity>
```

(14) Clock edge sensitivity (SRS_REQ_STD_04800)

Rationale: The objective is to check that for each clock source that there is only one edge detection mechanism used. The detection must be done per entity and per VLSI project.

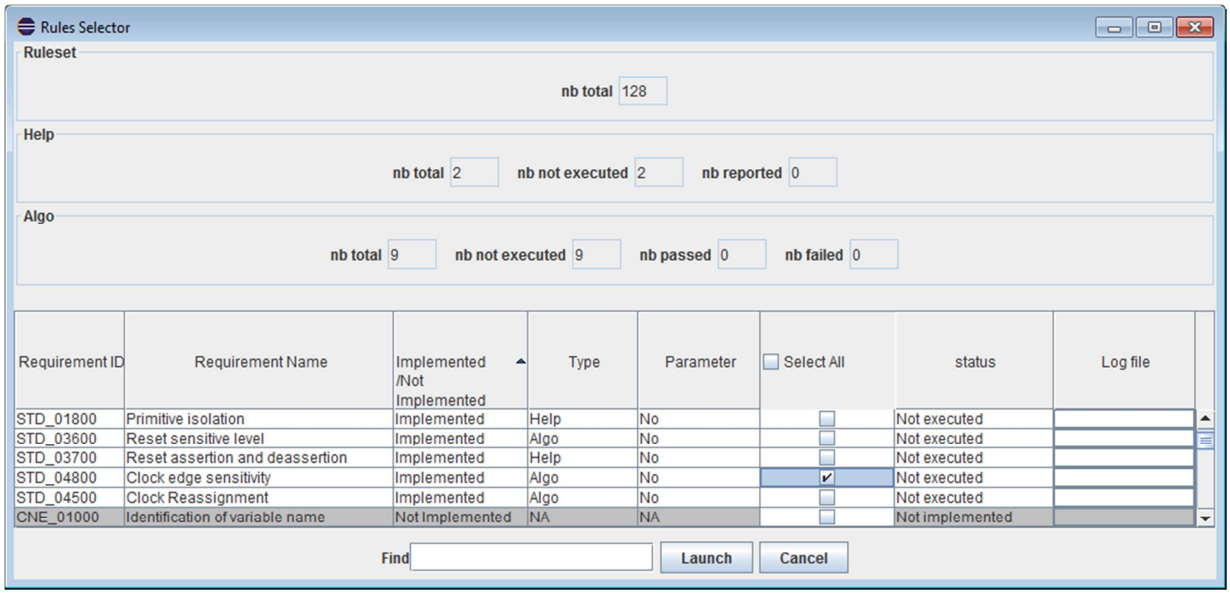
Detailed Description: Search Clock VHDL elements in VLSI project with "Clock identification" function, and verify that clock are always used with the same edge (falling edge or rising edge) in each file. When a clock source is used on 2 different edges, a violation report is built showing the different usage of the concerned clock source in the file.

The same detection mechanism is done at VLSI project level. When a clock source is used on different edges between entities, then a violation report is done and all files using the concerned clock source are reported with the corresponding edge.

Limitations: none at this level.

Expected Result:

When we select the feature Clock edge sensitivity, as shown in this figure, the rule checker tool list clock signal are used inside combinatory function.



We can see the result of this detection in rc_report_rule_REQ_STD_04800_Clock edge sensitivity level.xml file. In this example, the clock source tag “clockSourceB” is used in both edges in file “ModuleMap_pkg.vhd.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_04800>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_04800</description>
  <ruleName>Clock Edge Sensitivity</ruleName>
  <creationDate>Fri Nov 27 16:27:48 CET 2015</creationDate>
  <entity>
    <violationType>mixEdgesPerProject</violationType>
    <fileName>\FPGA_FM_Datapackage\02_Code\DPULogic\Definitions\ModuleMap_pkg.vhd</fileName>
    <entityName>CLKOUT</entityName>
    <clockSourceATag>clockSourceA</clockSourceATag>
    <clockSourceAEdge/>
    <clockSourceBTag>clockSourceB</clockSourceBTag>
    <clockSourceBEdge>both</clockSourceBEdge>
    <clockSourceCTag>clockSourceC</clockSourceCTag>
    <clockSourceCEdge/>
    <clockSourceDTag>clockSourceD</clockSourceDTag>
    <clockSourceDEdge/>
    <clockSourceETag>clockSourceE</clockSourceETag>
    <clockSourceEEdge/>
    <clockSourceFTag>clockSourceF</clockSourceFTag>
    <clockSourceFEdge/>
  </entity>
```

(15) Sensitivity list for synchronous process (SRS_REQ_STD_05000)

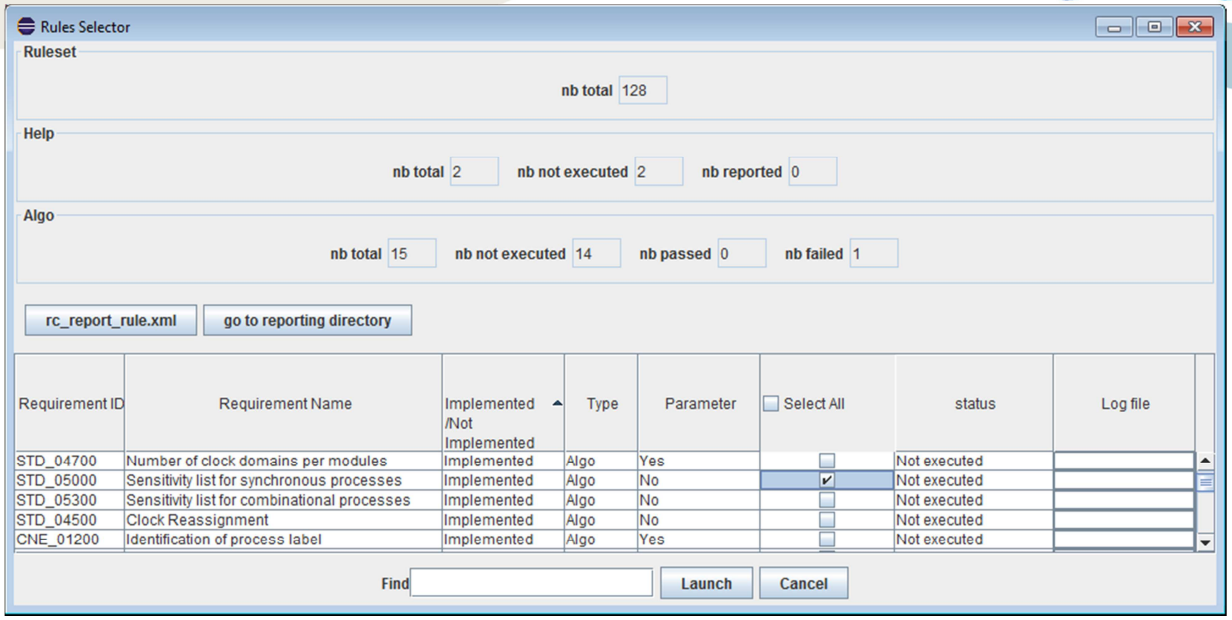
Rationale: The objective is to verify that asynchronous resets and clocks are correctly declared in the sensitivity list so there is no misbehavior while simulating the design.

- Detailed Description:
  - With the help on the clock identification tool (see SRS_REQ_FEAT_FN15) and the reset identification tool (see SRS_REQ_FEAT_FN18), the clocks and asynchronous resets for each synchronous process are identified.
  - With the help of AST from ZamiaCad, all the elements of the sensitivity list of the synchronous processes are identified.
  - Then, a comparison is done between clocks/resets list and sensitivity elements list.
  - A violation report is automatically generated when there is not a complete match between both lists.
    - When there are fewer elements in sensitivity list, it means that the missing element is used in process but not defined in sensitivity list.
    - When there are more elements in sensitivity list, it means that the defined elements in sensitivity list are not used in process.

Limitations: at this level none, see parent for any limitation.

Expected Result:

When we select the feature Sensitivity list for synchronous process, as shown in this figure, the rule checker lists violation for each synchronous process.



We can see the result of this detection in rc_report_rule_STD_05000_Sensitivity List for Synchronous Processes.xml file. In this example, in the process “COUNTER” the element “COUNT” to the sensitivity list is “not used” in this synchronous process in file “\src\AES\key_expander.vhd”.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_05000>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_05000</description>
  <ruleName>Sensitivity List for Synchronous Processes</ruleName>
  <creationDate>Tue Mar 15 09:14:33 CET 2016</creationDate>
  <sensitivity>
    <violationType>not used</violationType>
    <fileName>\src\AES\key_expander.vhd</fileName>
    <entityName>KEY_EXPANDER</entityName>
    <architectureName>EXPANSION</architectureName>
    <processName>COUNTER</processName>
    <processLoc>118</processLoc>
    <sensitivityName>COUNT</sensitivityName>
    <sensitivityLoc>118</sensitivityLoc>
  </sensitivity>
  <sensitivity>
    <violationType>not used</violationType>
    <fileName>\src\AES\key_expander.vhd</fileName>
    <entityName>KEY_EXPANDER</entityName>
    <architectureName>EXPANSION</architectureName>
    <processName>unnamed</processName>
    <processLoc>118</processLoc>
    <sensitivityName>FIRST</sensitivityName>
    <sensitivityLoc>118</sensitivityLoc>
  </sensitivity>
```

(16) Sensitivity list for combinational process (SRS_REQ_STD_05300)

Rationale: The objective is to verify that all inputs of combinational processes are correctly declared in the sensitivity list so there is no misbehavior while simulating the design.

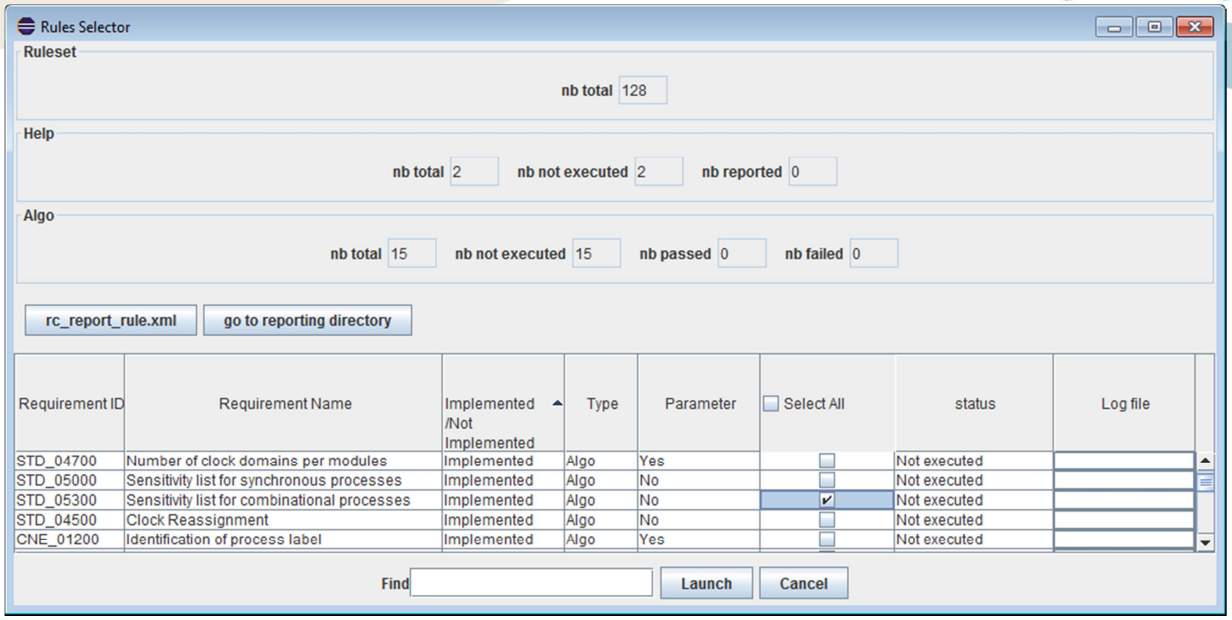
Detailed Description:

- With the help on tool SRS_REQ_FEAT_COMB_INPUT), all the inputs for all the combinational processes are identified.
- With the help of AST from ZamiaCad, all the elements of the sensitivity list of the asynchronous processes are identified.
- Then, a comparison is done between inputs list and sensitivity elements list.
- A violation report is automatically generated when there is not a complete match between both lists.
    - When there are fewer elements in sensitivity list, it means that the missing element is used in process but not defined in sensitivity list.
    - When there are more elements in sensitivity list, it means that the defined elements in sensitivity list are not used in process.

Limitations: at this level none, see parent for any limitation.

Expected Result:

When we select the feature Sensitivity list for combinational process, as shown in this figure, the rule checker tool lists violation for each combinational process.



We can see the result of this detection in rc_report_rule_STD_05300_Sensitivity list for combinational processes.xml file. In this example, the combinational input "NEW_KEY0_MEM" is defined in sensitivity list to the unnamed process at line 508 in file "aes_node.vhd".

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<STD_05300>
  <author>rule checker</author>
  <automaticGeneration>YES</automaticGeneration>
  <description>violation report for rule STD_05300</description>
  <ruleName>Sensitivity list for combinational processes</ruleName>
  <creationDate>Tue Mar 15 09:34:56 CET 2016</creationDate>
  <sensitivity>
    <violationType>not defined</violationType>
    <fileName>\src\AES\aes_node.vhd</fileName>
    <entityName>AES_NODE</entityName>
    <architectureName>MAPPING</architectureName>
    <processName>unnamed</processName>
    <processLoc>508</processLoc>
    <sensitivityName>NEW_KEY0_MEM</sensitivityName>
    <sensitivityLoc>525</sensitivityLoc>
  </sensitivity>
  <sensitivity>
    <violationType>not defined</violationType>
    <fileName>\src\AES\aes_node.vhd</fileName>
    <entityName>AES_NODE</entityName>
    <architectureName>MAPPING</architectureName>
    <processName>unnamed</processName>
    <processLoc>508</processLoc>
    <sensitivityName>NEW_KEY1_MEM</sensitivityName>
    <sensitivityLoc>526</sensitivityLoc>
  </sensitivity>
```

F. Synthesis

The following file gives a list of the tools and rules supported by the Rule Checker.

The parent references, the log report items for each tool/rule are described.

VHDL_TOOL_Tools_Rules_Synthesis_v3.2.

G.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

H.Glossary

The table below lists the definition of each item used in log reports.

Items	Description
architectureName	Name of the VHDL architecture. When no architecture is present in VLSI file, the field is left empty.
architectureLoc	Location (expressed in line number) in the VLSI file of the declaration of the architectureName.
author	Name of the author of the report or Name of the tool used for report generation.
automaticGeneration	Tells if the report has been generated automatically.
clockSourceLoc	Location (expressed in line number) in the VLSI file of the usage of the clockSourceName.
clockSourceName	Name of clock source represent the origin of the clock signal.
clockSourceTag	Tag of clock source, tag is used to avoid mismatch.
clockSignalLoc	Location (expressed in line number) in the VLSI file of the usage of the clockSignalName.
clockSignalName	Name of clock signal used In synchronous process. When process is not synchronous, the field is left empty.
clockSignalNameAfter	Name of clock signal after changed
clockSignalNameBefore	Name of clock signal before changed
creationDate	Date of report generation.
description	Name of the test done.
entityLoc	Location (expressed in line number) in the VLSI file of the declaration of the entityName.

entityName	Name of the VHDL entity. When no entity is present in VLSI file, the field is left empty.
fileName	Name of VLSI file.
hasAsynchronousReset	Tells if the synchronous process uses an asynchronous reset.
InputLoc	Location (expressed in line number) in the VLSI file of the usage of the inputName.
InputName	Name of input of combinational process used In assynchronous process. When process is synchronous, the field is left empty.
InputRange	Range of the signal called inputName (for type discrete, the range is equal to 1)
InputType	Type of the signal called inputName (discrete, vector, …)
instanceName	Name of the component when is instantiate
IOLoc	Location (expressed in line number) in the VLSI file of the usage of the ioName.
IOName	Name of input and output component's in VLSI project.
IORange	Range of the signal called ioName (for type discrete, the range is equal to 1)
IOTag	Tag of input and output component, tag is used to avoid mismatch.
IOType	Type of the signal called ioName (discrete, vector, …)
isSynchronous	Tells if the process is a synchronous process.
libraryName	Name of library declared In VLSI file.
libraryNameLoc	Location (expressed in line number) in the VLSI file of the declaration of the libraryName.
mapLoc	Location (expressed in line number) in the VLSI file of the declaration of the signal mapping.
nbLine	Number of lines of the VLSI file.
processLoc	Location (expressed in line number) in the VLSI file of the declaration of the processName.
processName	Name of the VHDL process. When no process is present in VLSI file, the field is left empty. When the process has no label, then the field returns “unnamed”.
RegisterLoc	Name of register used In synchronous process. When process is not synchronous, the field is left empty.

RegisterName	Location (expressed in line number) in the VLSI file of the usage of the registerName.
RegisterRange	Range of the signal called registerName (for type discrete, the range is equal to 1)
RegisterType	Type of the signal called registerName (discrete, vector, …)
resetSignalLoc	Location (expressed in line number) in the VLSI file of the usage of the resetSignalName.
resetSignalName	Name of asynchronous reset signal used In synchronous process. When process has no asynchronous reset, the field is left empty.
ruleName	Name of the rule
SignalTag	SignalTag is used to identify the different signal for launch logicalCone (register, inputComb, IO).
SinkLoc	Location (expressed in line number) in the VLSI file of the usage of the SinkName.
SinkName	Name of the destinataire when used the logical cone function
SinkType	Type of the signal called SinkName (process, assignment, …)
SourceLoc	Location (expressed in line number) in the VLSI file of the usage of the SourceName.
SourceName	Name of the source when used the logical cone function
SourceType	Type of the signal called SourceName (process, assignment, …)
violationType	Name of violation for rule Algo