



Smash HDL bugs on the fly!

# What added value we can bring to you

---

We help leading-edge teams write bug-free VHDL and Verilog code.

By automating deep code checks on commits and pull requests.

# Why you should use our approach and tooling

---

Write bug-free, secure and highly maintainable code

Comply with standards

Reduce hardware development costs

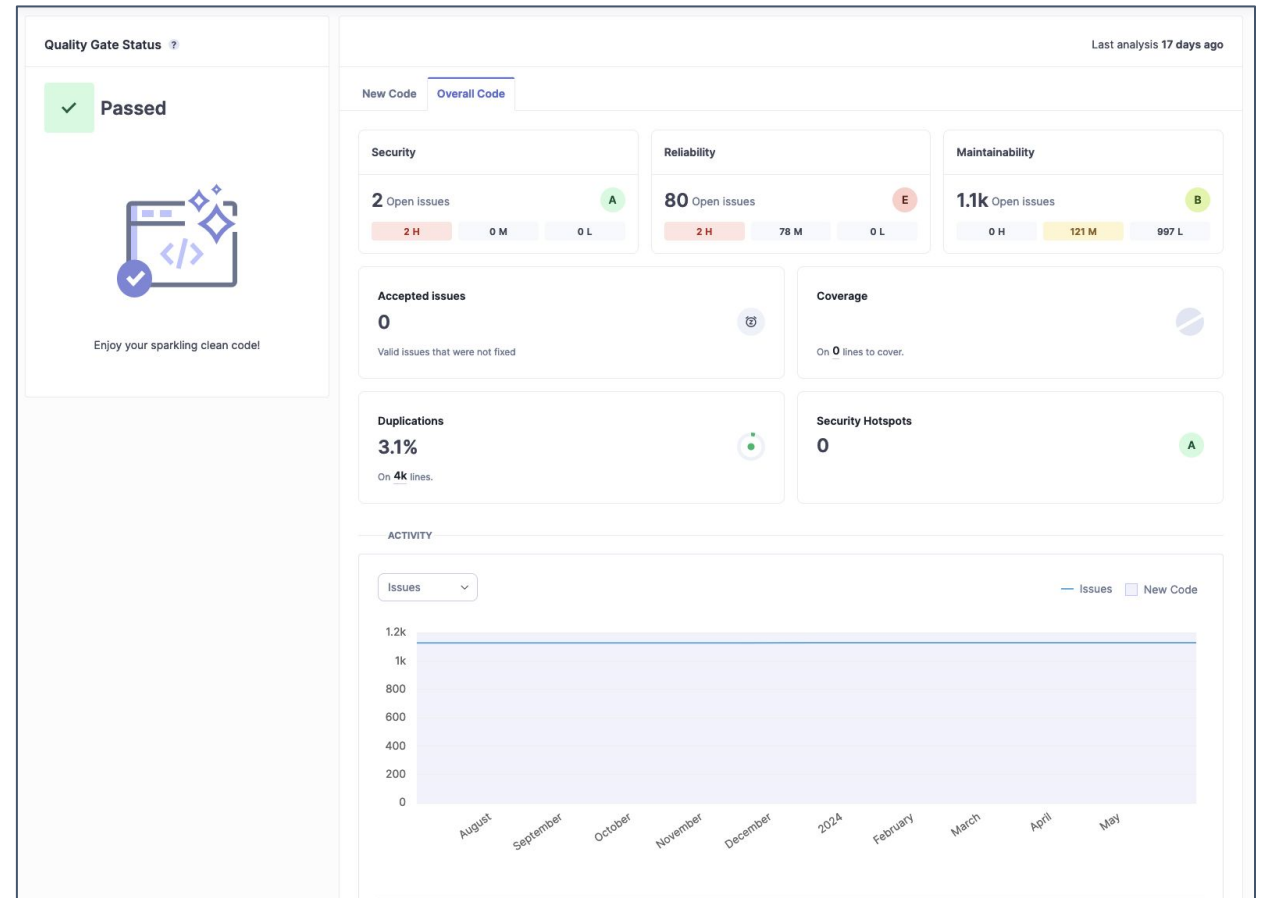
Improve productivity and reduce time to market

Cut down technical debt

Attract and keep top-level designers

# Linty for everyone

- Manager and Team Leader
  - Keep your developers happy
  - Rework less, innovate more
  - Minimize risks, maximize reputation
  - Gated quality
  - Configure rules
  - Set requirements
  - Define quality gate
  - Monitor Technical Debt



# Linty for everyone

- Architect and Senior Designer
  - Protection of the main branch
  - CI/CD pipeline under GitHub, GitLab
  - MR/PR decoration
  - Peer reviews assistant
  - Safety-critical standards evidences (DO254, IEC61508,...)
  - Code coverage "on new code"

The screenshot shows a code editor with a coverage report on the left and C code on the right. The coverage report includes a table with the following data:

Category	Value
Security Review	>
Coverage	66.7%
Overview	
New Code	
Coverage	66.7%
Lines to Cover	3
Uncovered Lines	1
Line Coverage	66.7%

The code on the right is C code with annotations for coverage. The code is as follows:

```
152 mult_function := mult_signed_divide;
153 david... when "011011" => --DIVU s->lo=r[rs]/r[rt]; s->hi=r[rs]*r[rt]; New Code
154 mult_function := mult_divide;
155 when "100000" => --ADD r[rd]=r[rs]+r[rt];
156 c_source := c_from_alu;
157 alu_function := alu_add;
158 david... when "100001" => --ADDU r[rd]=r[rs]+r[rt];
159 c_source := c_from_alu;
160 alu_function := alu_add;
161 when "100010" => --SUB r[rd]=r[rs]-r[rt];
162 c_source := c_from_alu;
163 alu_function := alu_subtract;
164 when "100011" => --SUBU r[rd]=r[rs]-r[rt];
165 c_source := c_from_alu;
166 alu_function := alu_subtract;
167 when "100100" => --AND r[rd]=r[rs]&r[rt];
168 c_source := c_from_alu;
169 alu_function := alu_and;
170 when "100101" => --OR r[rd]=r[rs]|r[rt];
171 c_source := c_from_alu;
172 alu_function := alu_or;
173 when "100110" => --XOR r[rd]=r[rs]^r[rt];
174 c_source := c_from_alu;
175 alu_function := alu_xor;
176 when "100111" => --NOR r[rd]=~(r[rs]|r[rt]);
```

The screenshot shows a GitHub pull request interface. At the top, there is a comment from 'sonarqube-linty-cloud' bot, dated Sep 20, 2023, stating 'Quality Gate failed'. Below this, there are links for 'Failed conditions', '3 New Bugs (required ≤ 0)', and 'See analysis details on SonarQube'. There is also a link to 'Catch issues before they fail your Quality Gate with our IDE extension' and a 'SonarLint' logo.

Below the comment, there is a message from 'racodond' dated Sep 25, 2023, stating 'deleted a comment from sonarqube-linty-cloud bot on Sep 25, 2023'.

At the bottom, there is a section for 'Add more commits by pushing to the feature/Lets\_dream branch on Linty-Services/bugfinder-sample'.

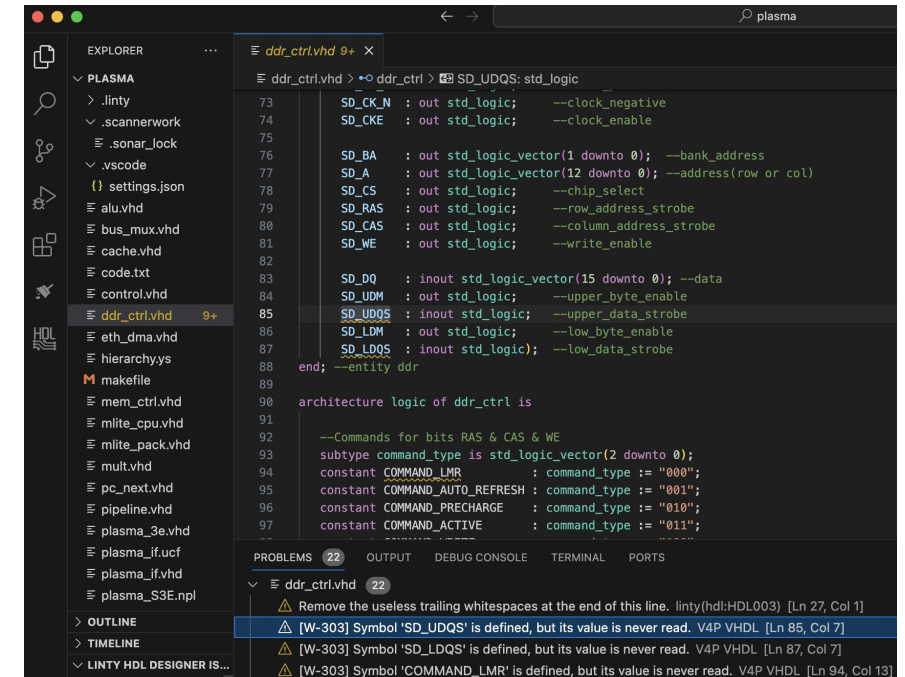
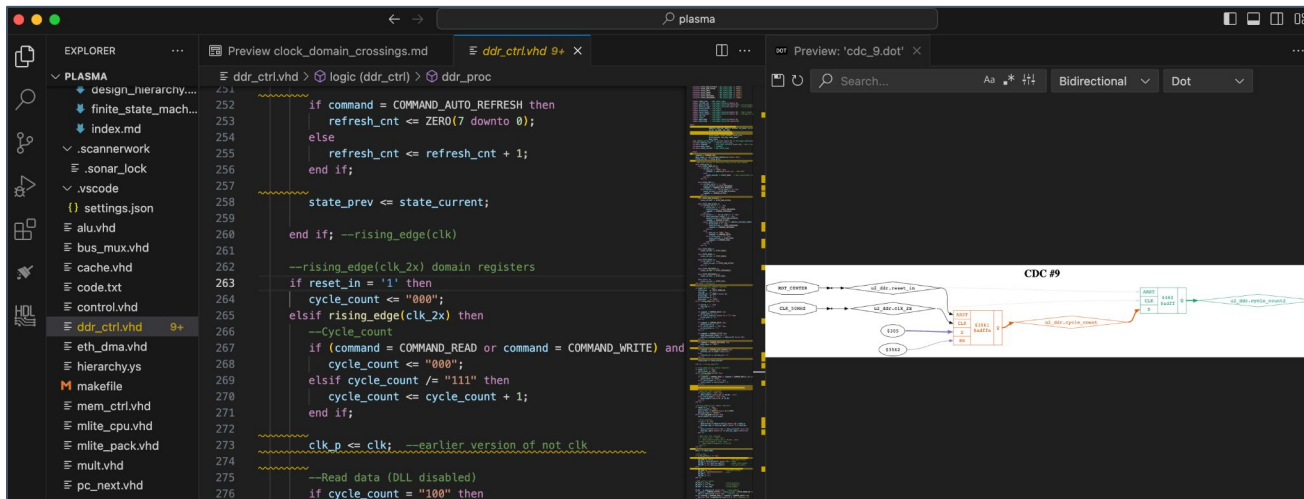
The main content of the pull request shows a green checkmark and the text 'Approval not required'. Below this, there is a section for '1 pending reviewer' and a red 'X' icon with the text 'All checks have failed' and '3 failing checks'. The failing checks are:

- Linty / Linty (pull\_request) Failing after 1m (Required)
- Linty / Linty (push) Failing after 1m (Required)
- SonarQube Code Analysis Failing after 23s — Quality Gate failed (Details)

At the bottom, there is a section for 'Required statuses must pass before merging' and a button for 'Merge pull request'.

# Linty for everyone

- Designers
  - Discover issues from the moment you write code
  - +220 HDL checks in real-time
  - CDC, RDC, FSM graphs
  - Navigation / Auto-completion
  - Code colourisation



# Linty ROI from our customers

---

- ROI less than 3 months
- 3 - 5 hours per designer per week saved
- Improved designer experience
- Maintained stable designs by reducing technical debt
- HDL code quality standardization across the organization
- Reinforceable coding best practices



Smash HDL bugs on the fly!

## CONTACT US

Sales - Stephen Rimbault  
CTO - David Racodon  
CEO - Vincent Louis

[stephen.rimbault@lnty-services.com](mailto:stephen.rimbault@lnty-services.com)  
[david.racodon@lnty-services.com](mailto:david.racodon@lnty-services.com)  
[vincent.louis@lnty-services.com](mailto:vincent.louis@lnty-services.com)



# Useful links

---

Website:	<a href="https://lnty-services.com">https://lnty-services.com</a>
Demo platform:	<a href="https://demo.lnty-services.com">https://demo.lnty-services.com</a>
VHDL checks:	<a href="https://vhdl.lnty-services.com">https://vhdl.lnty-services.com</a> <a href="https://hdl.lnty-services.com">https://hdl.lnty-services.com</a>
Verilog/SystemVerilog checks:	<a href="https://verilog.lnty-services.com">https://verilog.lnty-services.com</a> <a href="https://hdl.lnty-services.com">https://hdl.lnty-services.com</a>
Coverage of DO254 and CNES standards:	<a href="https://standards.lnty-services.com">https://standards.lnty-services.com</a>
Documentation/Installation:	<a href="https://doc.lnty-services.com">https://doc.lnty-services.com</a>
Merge Request decoration on GitLab:	<a href="https://gitlab-mr.lnty-services.com">https://gitlab-mr.lnty-services.com</a>
Pull Request decoration on GitHub:	<a href="https://github-pr.lnty-services.com">https://github-pr.lnty-services.com</a>
Linty HDL Designer for VS Code:	<a href="https://hdl-designer.lnty-services.com">https://hdl-designer.lnty-services.com</a>

# Linty features

---

- Easy integration within your continuous integration process (GitHub, GitLab, Bitbucket, etc.)
- Wide range of rules: Conventions, complexity, CDC, FSM and many mores
- Clean as you code with Linty HDL Designer: Powerful VS Code extension
- Cloud platform or on-premise installation
- Automated qualification kit (DO-254, DO-330, IEC 61508, EN 50129, ISO 26262, etc.)
- Focus on new/modified code
- Based on SonarQube, leading platform for continuous code quality control
- Also analyze your C, C++, Python (and many more) code on the same platform

# Linty Services

**2017**

- 1st Lexer/Parser VHDL prototype – SonarQube Plugin
- Promoted by French Defense Innovation Agency



**2020**

- New post synthesis rules
- Linty Safety-Critical (qualification kit)  
DO254, IEC 61508, ISO 26262, EN50129, ECSS



**2018**

- 1st Customers
- CNES : VHDL new rules implementation



**2022**

- CNES strategic partnership
- Verilog/SystemVerilog plugin (+100 rules)
- Enhanced rules based on Yosys technology



# Linty Safety-Critical

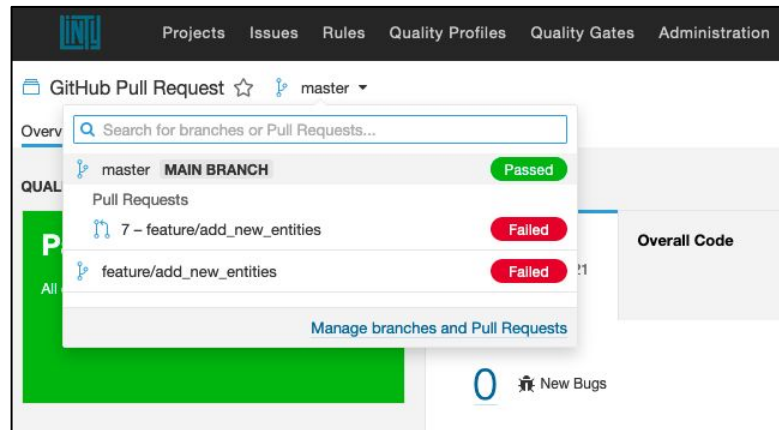
---

Airborne systems:	DO254, DO178C, DO330
Space:	ECSS
Automotive:	ISO26262
Electronic:	IEC 61508
Defense:	Mil-STD-882-E, DGA Technical Note
Naval:	SCAT 12805
Medical:	IEC 62304
Nuclear:	IEC 61513
Railway:	EN5012x

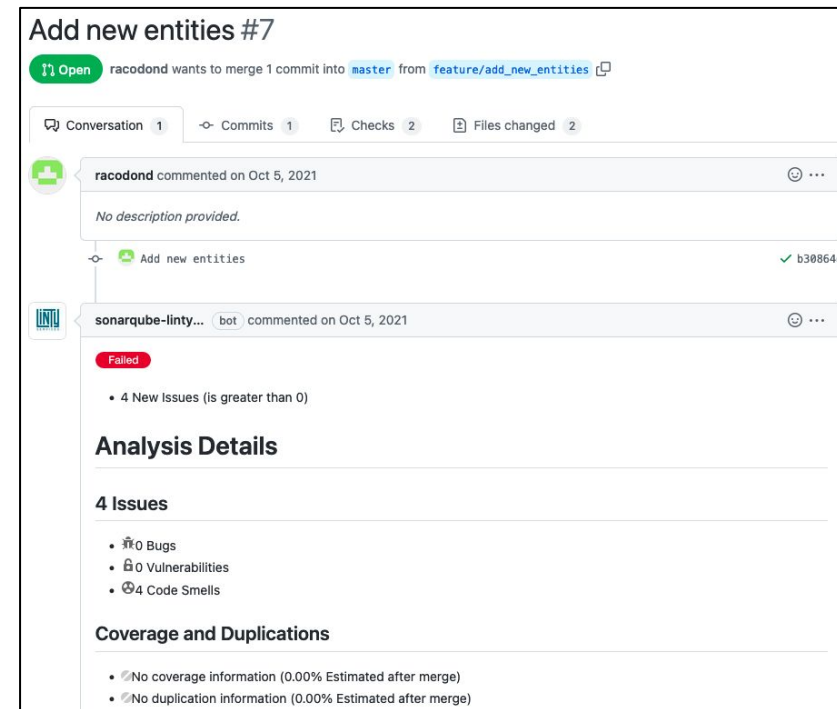
Hardware design audit experience : > 20 years, several hundreds reviews, world wide

# Linty DevOps - Continuous Integration

- Multi-branch analysis - Pull request decoration - GitLab - Bitbucket- GitHub - Jenkins
  - You would get the most added value from Linty while it is fully integrated in your DevOps / Continuous Integration process



Multi-branch analysis



Pull request decoration

# Synthetic dashboard

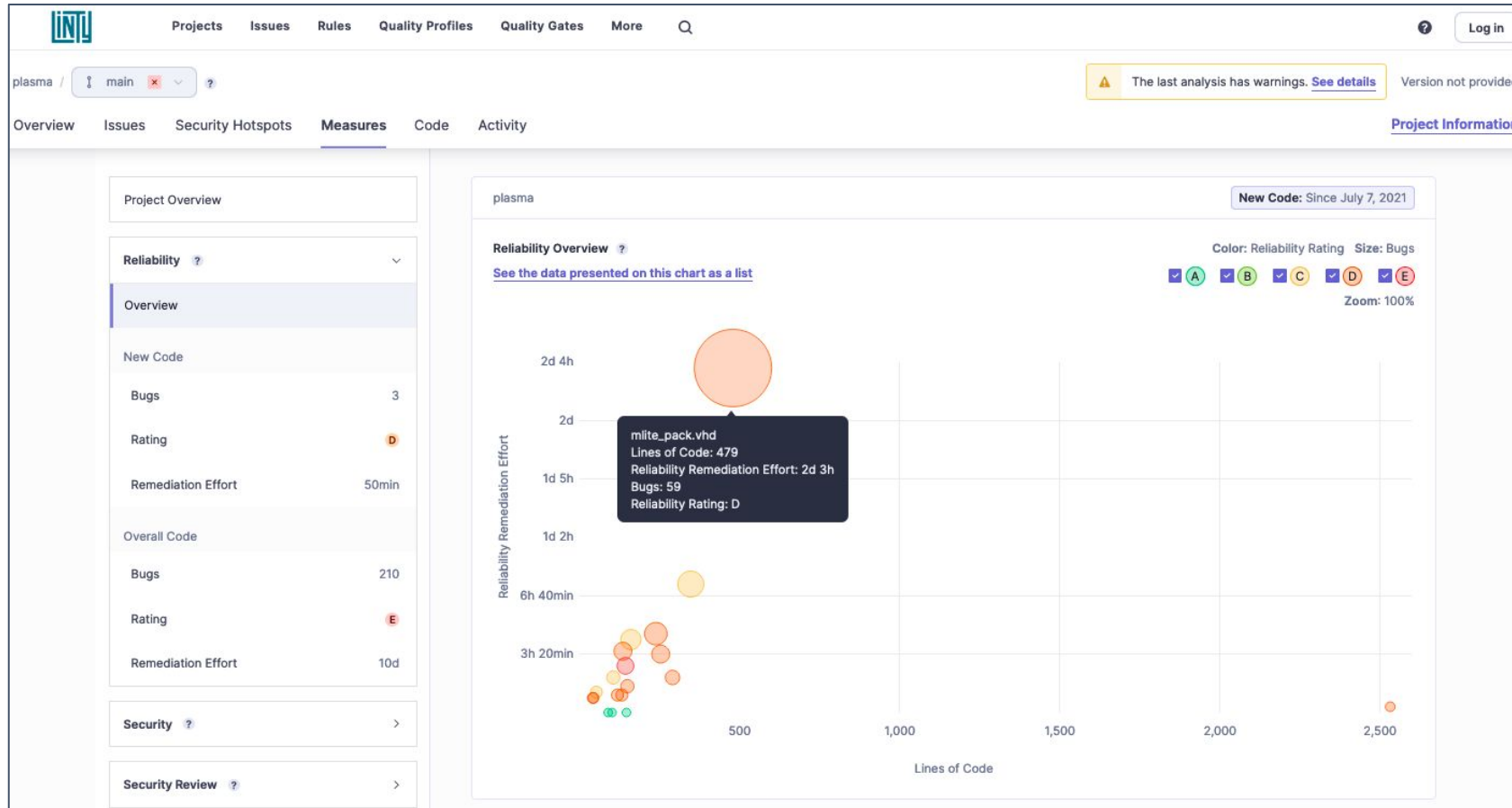
The screenshot displays the Lintly Synthetic dashboard for a project named 'GitLab BugFinder Sample'. The dashboard is divided into two main sections: 'Quality Gate Status' and 'Measures'.

**Quality Gate Status:** The Quality Gate is **Failed** due to 1 failed condition(s). The condition is: '3 New Bugs is greater than 0'. A tip suggests: 'Fix issues before they fail your Quality Gate with SonarLint in your IDE. Power up with connected model!'.

**Measures:**

- Reliability:** 3 New Bugs (Grade D)
- Security:** 0 New Vulnerabilities (Grade A)
- Security Review:** 0 New Security Hotspots (Grade A)
- Maintainability:** 2 New Code Smells (Grade A)
- Coverage:** Coverage on 0 New Lines to cover (Estimated after merge)
- Duplications:** 0.0% Duplications on 8 New Lines (Estimated after merge)

# Synthetic measures



# Code coverage on "New code"

The screenshot displays the Lintu IDE interface for a project named 'ModelSim Coverage'. The 'Measures' tab is active, showing a sidebar with various metrics. The 'Coverage' measure is expanded, showing a table with the following data:

Metric	Value
Coverage	66.7%
Lines to Cover	3
Uncovered Lines	1
Line Coverage	66.7%
Conditions to Cover	0

The main editor area shows a code snippet with line numbers 146 to 188. A vertical red dashed line indicates the current position in the code. A blue highlight is present on line 153, which is labeled 'New Code' in the right margin. The code snippet includes various conditional operations like `when "011001" => --MULT s->lo=r[rs]*r[rt]; s->hi=0;`.



# Code duplication

The screenshot displays the Lintu IDE interface. At the top, there is a navigation bar with 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'More'. A search bar and a 'Log in' button are also present. Below the navigation bar, the current project is identified as 'plasma' with a 'main' branch selected. A notification banner indicates 'The last analysis has warnings. See details' and 'Version not provided'. The main interface is divided into a left sidebar and a central code editor.

The left sidebar contains a 'Measures' section with the following metrics:

- Reliability ?
- Security ?
- Security Review ?
- Maintainability ?
- Coverage
- Duplications (expanded):
  - Overview
  - New Code
  - Duplicated Lines: 0
  - Duplicated Blocks: 0
  - Overall Code
  - Density: 10.2%

The central code editor shows a Verilog file with the following content:

```
1
2
3 -- TITLE: Memory Controller
4 -- AUTHOR: Steve Rhoads (rhoadss@yahoo.com)
5 -- DATE CREATED: 1/31/01
6 -- FILENAME: mem_ctrl.vhd
7 -- PROJECT: Plasma CPU core
8 -- COPYRIGHT: Software placed into the public domain by the author.
9 -- Software 'as is' without warranty. Author liable for nothing.
10 -- DESCRIPTION:
11 -- Memory controller for the Plasma CPU.
12 -- Supports Big or Little Endian mode.
13
14 library ieee;
15 use ieee.std_logic_1164.all;
16 use work.mlite_pack.all;
17
18 entity mem_ctrl is
19     port (clk          : in std_logic;
20          reset_in     : in std_logic;
21          pause_in      : in std_logic;
22
23          mem_source    : in mem_source_type;
24          data_write    : in std_logic_vector(31 downto 0);
25          data_read     : out std_logic_vector(31 downto 0);
26          pause_out     : out std_logic;
27
28          address_next  : out std_logic_vector(31 downto 2);
29          byte_we_next  : out std_logic_vector(3 downto 0);
30
31          address       : out std_logic_vector(31 downto 2);
32          byte_we       : out std_logic_vector(3 downto 0);
33          data_w        : out std_logic_vector(31 downto 0);
34          data_r        : in std_logic_vector(31 downto 0);
35
36 end; --entity mem_ctrl
37
38 architecture logic of mem_ctrl is
39     --"00" = big_endian; "11" = little_endian
40
41
```

A tooltip is visible over the code, indicating a duplication: 'Duplicated By /mlite\_pack.vhd Lines: 219 - 238'. The code editor also shows a 'Duplicated Lines (%) 10.2%' header at the top of the code block.

# Linty Rules (several hundreds)

- Post-synthesis rules
- Clock Domain Crossing (CDC)
- Finite State Machine (FSM)
- Cyber threats

## Coverage of Standards by Linty Rules

Standard	Number of Rules	Number of Automatable Rules	Overall Coverage of Automatable Rules	Full Coverage	High Coverage	Low Coverage	No Coverage
<b>CNES</b>	73	67	78%	47	5	3	12
<b>CNES CUSTOM</b>	55	43	80%	30	5	1	7
<b>DO-254</b>	49	46	74%	30	2	5	9