2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

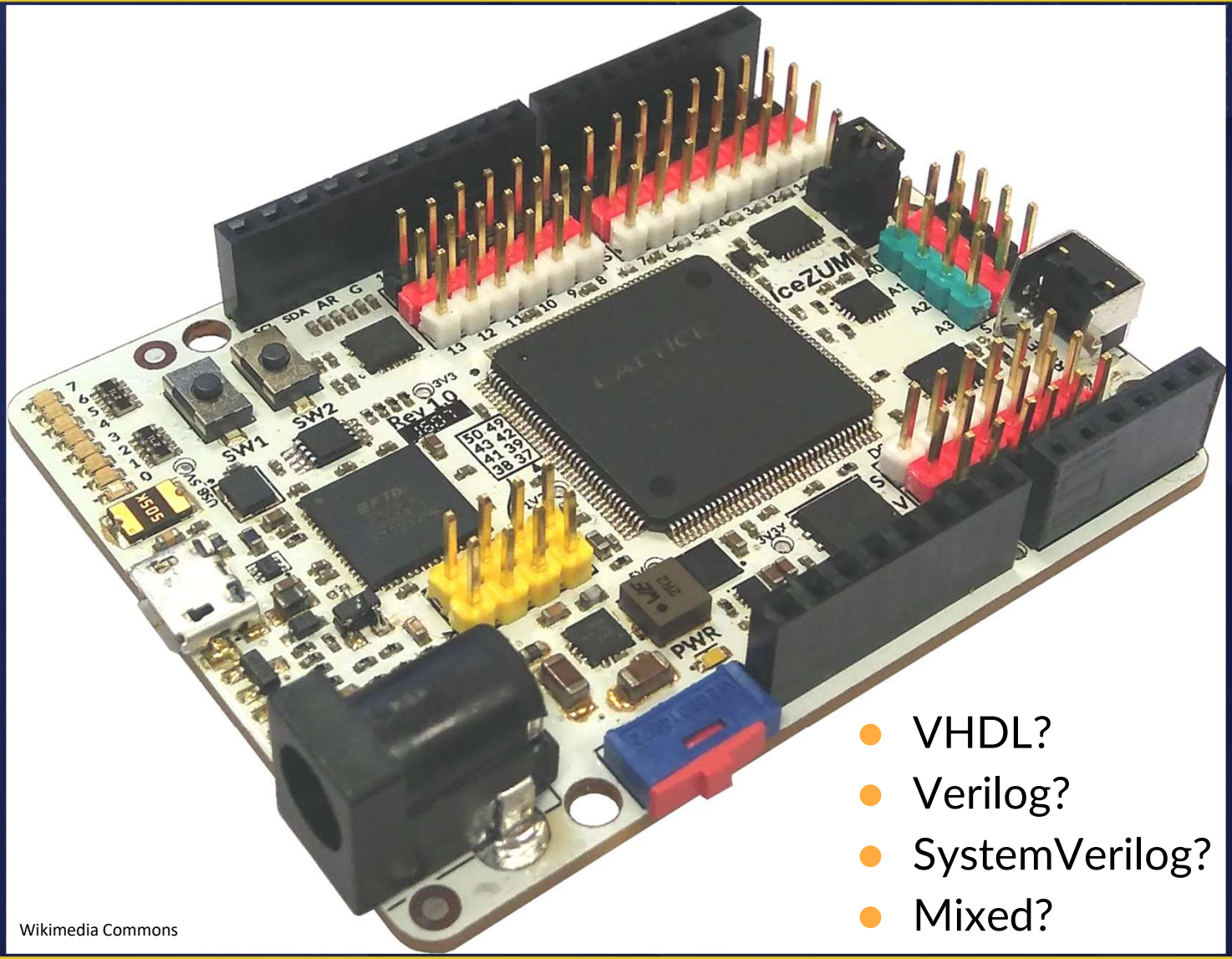
Boost your productivity in FPGA & ASIC design and verification

Bart Brosens

Application Engineering Lead

Sigasi;

accellera SYSTEMS INITIATIVE



Wikimedia Commons

- VHDL?
- Verilog?
- SystemVerilog?
- Mixed?

accellera SYSTEMS INITIATIVE

Focus

More productive

Focus on user

read
write
modify

- **SystemVerilog**
- **Verilog**
- **VHDL**
- **Mixed**

Most EDA tools

Focus on chip

simulation
synthesis
p&r
...

Focus

More productive

Focus on user

read
write
modify

- **SystemVerilog**
- **Verilog**
- **VHDL**
- **Mixed**

Most EDA tools

Focus on chip

simulation
synthesis
p&r
...

“Upfront Verification” : you design, we verify, let’s keep moving.

Topics

- Upfront Verification
- Project Exploration
- VS Code extension
- CI/CD

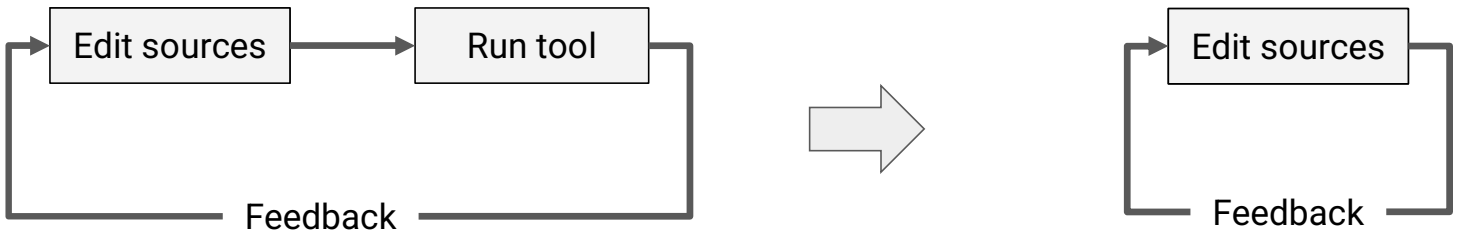
Topics

- Upfront Verification
  - Direct feedback
  - Language aware code checks
  - Intelligent content assist
- Project Exploration
- VS Code extension
- CI/CD

Upfront Verification

Direct feedback

- User-focused - relevant
- No need to switch context



Upfront Verification

Language aware code checks

- Code validation: Checking whether code is syntactically correct
- Code linting: Checking for problems in syntactically correct code
  - programming errors
  - bugs
  - stylistic errors (coding and naming style, indentation, header comments…)
  - suspicious or problematic constructs

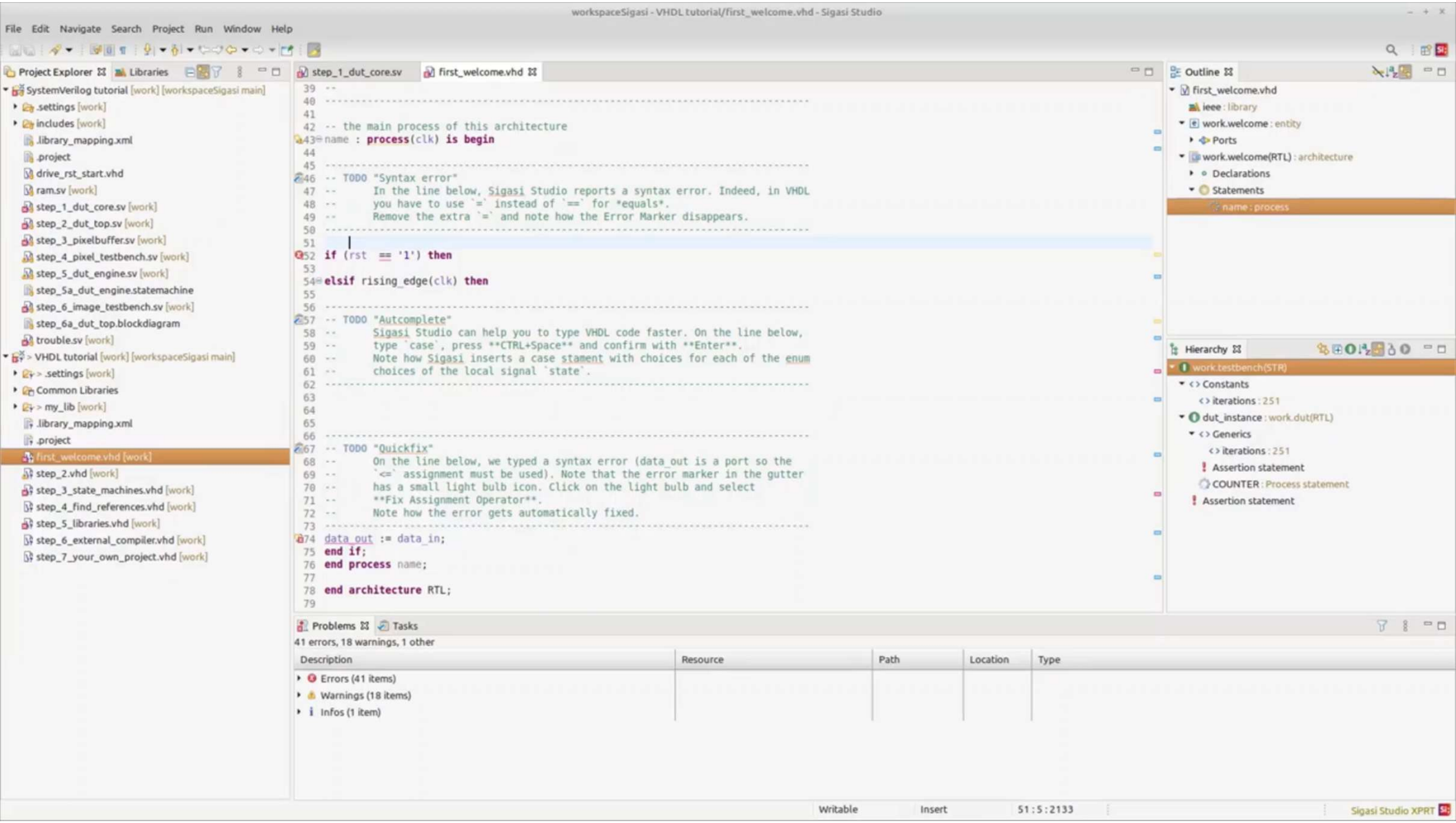
Upfront Verification

Intelligent content assist

- Quick Fixes to resolve common language issues
- Help with a consistent formatting of your code
- Use templates for common declarations and statements
- Use the content of the project to suggest appropriate identifiers as autocompletion candidates

Upfront Verification

Demo


workspaceSigasi - VHDL tutorial/first_welcome.vhd - Sigasi Studio
File Edit Navigate Search Project Run Window Help
Project Explorer
Libraries
SystemVerilog tutorial [work] [workspaceSigasi main]
.settings [work]
includes [work]
.library_mapping.xml
.project
drive_rst_start.vhd
ram.sv [work]
step_1_dut_core.sv [work]
step_2_dut_top.sv [work]
step_3_pixelbuffer.sv [work]
step_4_pixel_testbench.sv [work]
step_5_dut_engine.sv [work]
step_5a_dut_engine.statemachine
step_6_image_testbench.sv [work]
step_6a_dut_top.blockdiagram
trouble.sv [work]
VHDL tutorial [work] [workspaceSigasi main]
.settings [work]
Common Libraries
my_lib [work]
.library_mapping.xml
.project
first_welcome.vhd [work]
step_2.vhd [work]
step_3_state_machines.vhd [work]
step_4_find_references.vhd [work]
step_5_libraries.vhd [work]
step_6_external_compiler.vhd [work]
step_7_your_own_project.vhd [work]
step_1_dut_core.sv
first_welcome.vhd
-- the main process of this architecture
name : process(clk) is begin
-- TODO "Syntax error"
-- In the line below, Sigasi Studio reports a syntax error. Indeed, in VHDL
-- you have to use `=` instead of `==` for *equals*.
-- Remove the extra `=` and note how the Error Marker disappears.
if (rst == '1') then
elsif rising_edge(clk) then
-- TODO "Autcomplete"
-- Sigasi Studio can help you to type VHDL code faster. On the line below,
-- type `case`, press **CTRL+Space** and confirm with **Enter**.
-- Note how Sigasi inserts a case statement with choices for each of the enum
-- choices of the local signal `state`.
-- TODO "Quickfix"
-- On the line below, we typed a syntax error (data_out is a port so the
-- `<=` assignment must be used). Note that the error marker in the gutter
-- has a small light bulb icon. Click on the light bulb and select
-- **Fix Assignment Operator**.
-- Note how the error gets automatically fixed.
data_out := data_in;
end if;
end process name;
end architecture RTL;
Outline
first_welcome.vhd
ieee : library
work.welcome : entity
Ports
work.welcome(RTL) : architecture
Declarations
Statements
name : process
Hierarchy
work.testbench(STR)
Constants
iterations : 251
dut_instance : work.dut(RTL)
Generics
iterations : 251
Assertion statement
COUNTER : Process statement
Assertion statement
Problems
Tasks
41 errors, 18 warnings, 1 other
Description
Resource
Path
Location
Type
Errors (41 items)
Warnings (18 items)
Infos (1 item)
Writable
Insert
51 : 5 : 2133
Sigasi Studio XPRT

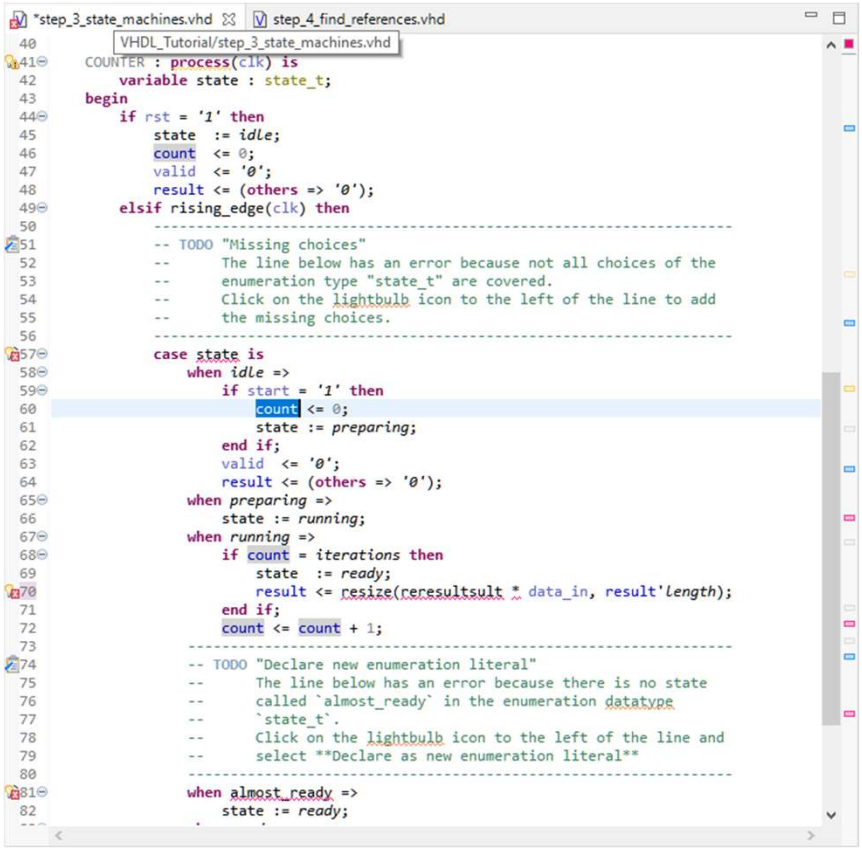
Topics

- Upfront Verification
- Project Exploration
  - HDL objects -> hyperlinks
  - Hovers show information
  - Multiple views for more insight
- VS Code extension
- CI/CD

Project Exploration

HDL objects -> hyperlinks

- Select Identifier then
  - Right-Click “Open Declaration”
  - F3
- Hyperlinks
  - Hold Ctrl and Click on Identifier

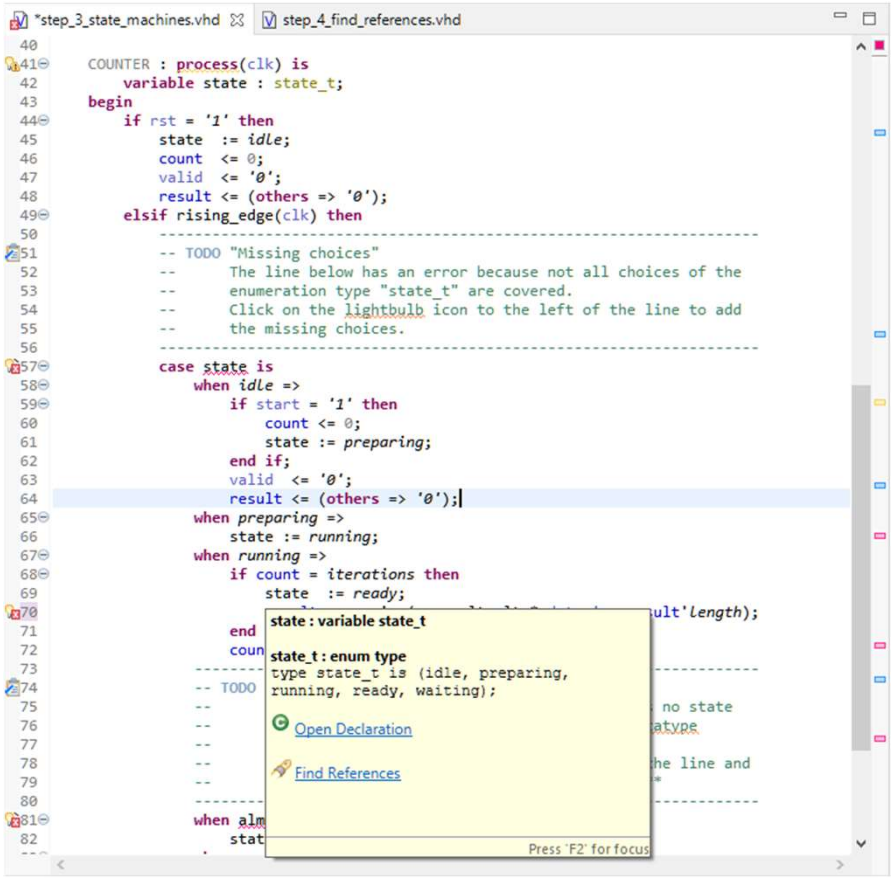


Project Exploration

Hovers show information

A tooltip shows information about a data object. No need to navigate away from the code.

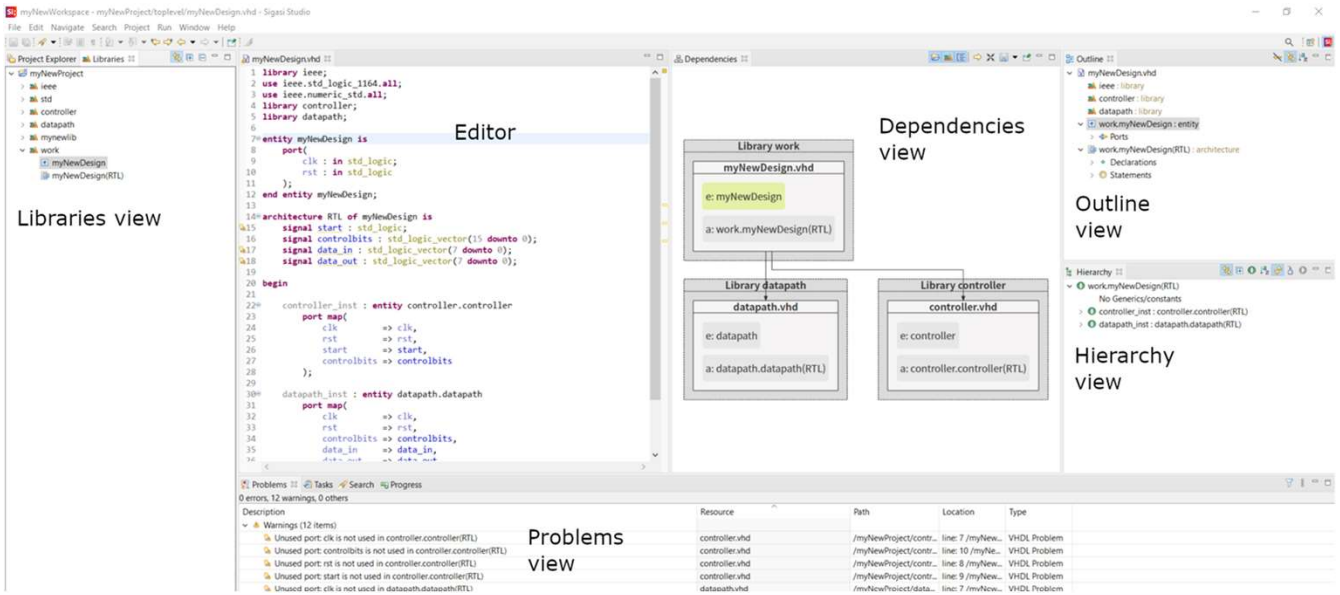
- Object definition
- Object type
- Useful hyperlinks



Project Exploration - views

Multiple views for more insight

- Views highlight particular *aspects* of the project
- Some examples in the next slides



Project Exploration - views

Views help navigation

- View ⇒ editor

  Double-click in view ⇒ editor opens corresponding code

- Editor ⇒ view

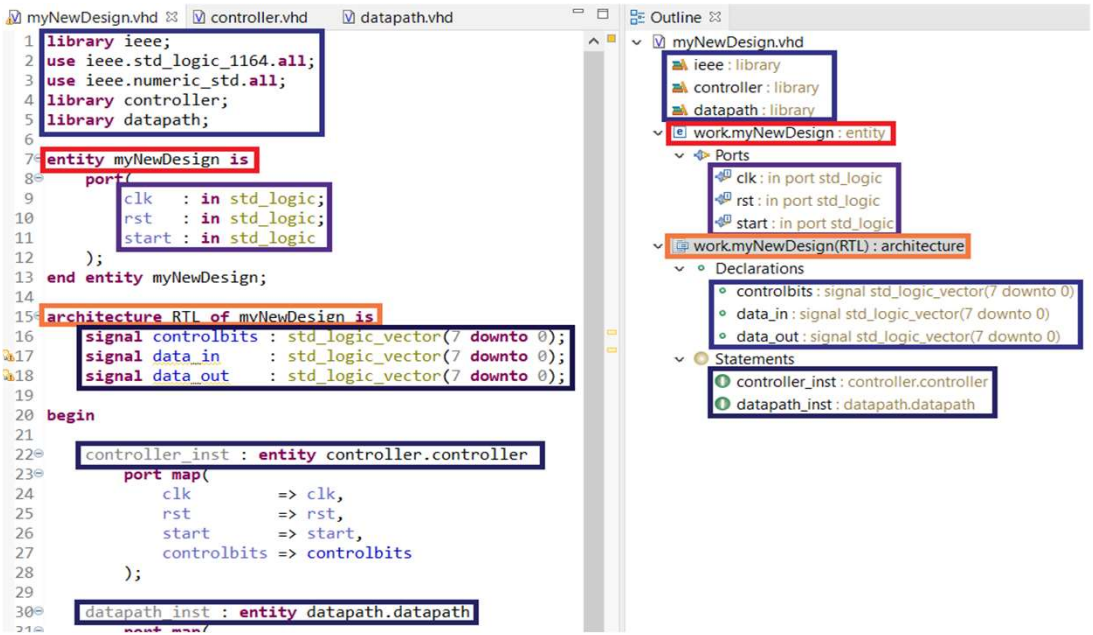
  Enable or disable link between view and editor

  If enabled: view highlights element being edited

Project Exploration - views

Outline View (single file)

- Libraries
- Entities
- Modules
- Architectures
- Ports & signals
- Assignments
- Instantiations
- Processes



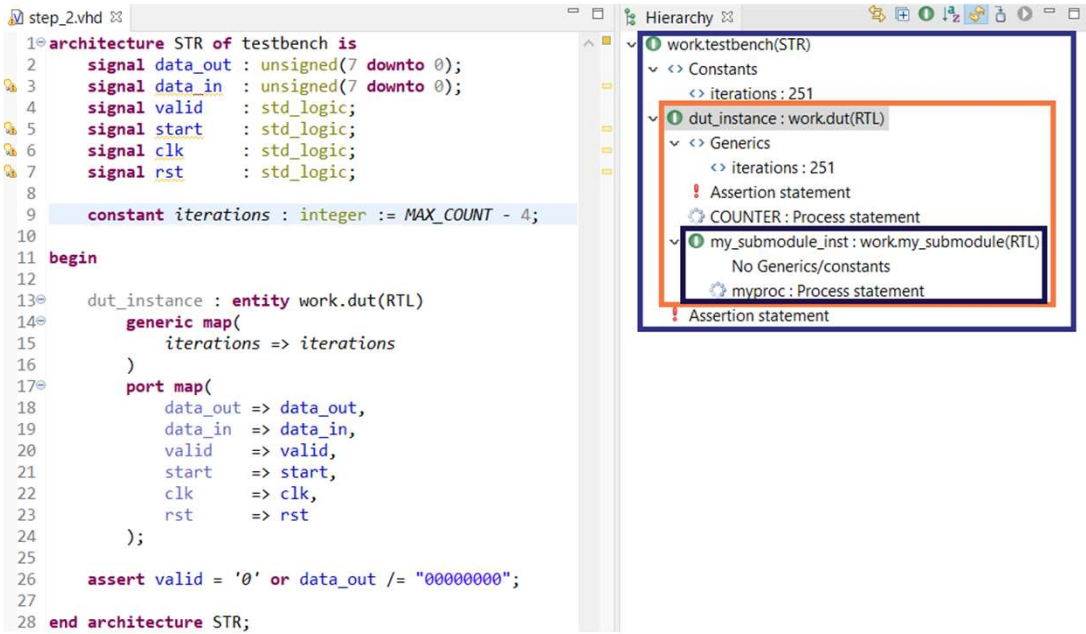
Project Exploration - views

Hierarchy View (entire design)

- Similar to outline view
- Spans multiple files

Navigation:

- Hierarchy => editor
- Editor => hierarchy

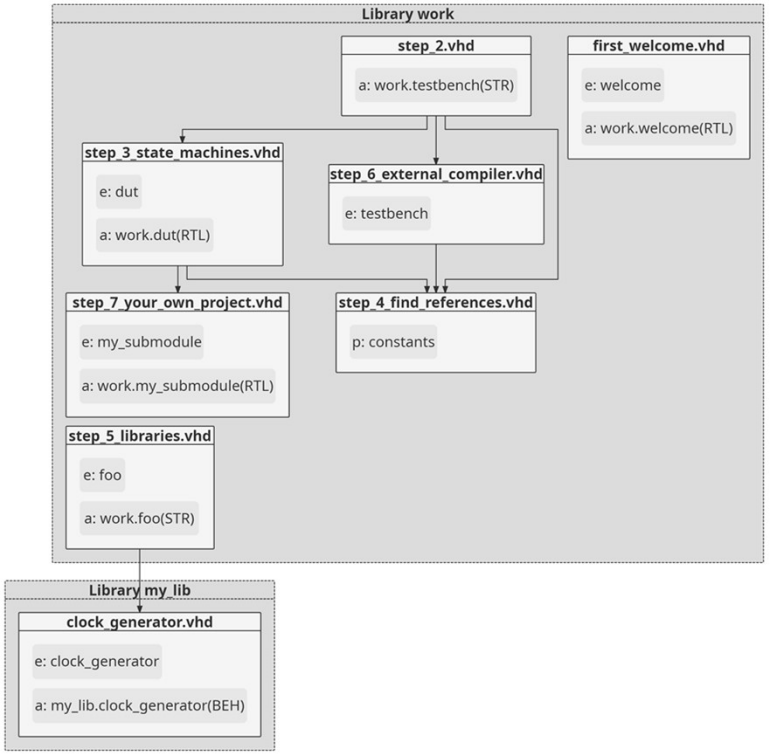


Project Exploration - views

Dependencies between files in a project

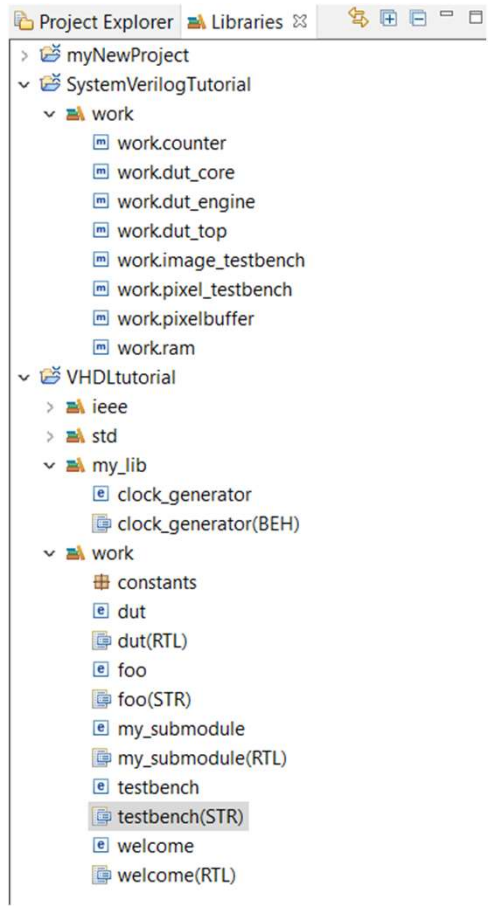
Configurable scope & details

Navigate from the dependencies view to the editor



Project Exploration - views

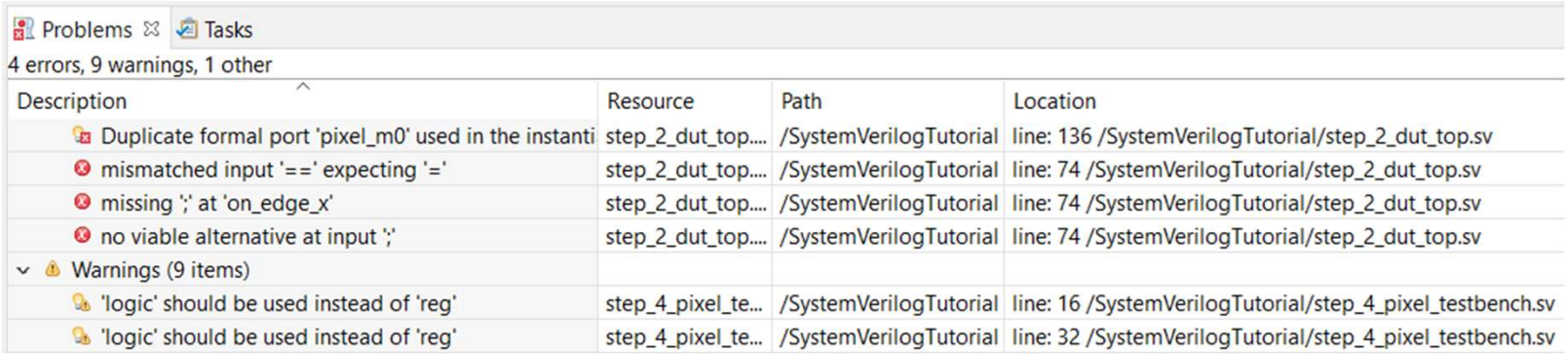
- Overview of all libraries in a project
- List of design units in libraries
- Navigate between libraries view and editor



Project Exploration - views

List of errors and warnings in open projects in the workspace

- Navigate to editor
- Apply Quick Fix



Project Exploration - views

List of tasks in open projects in the workspace

- Configurable tags: TODO, FIXME, ...
- Navigate to editor

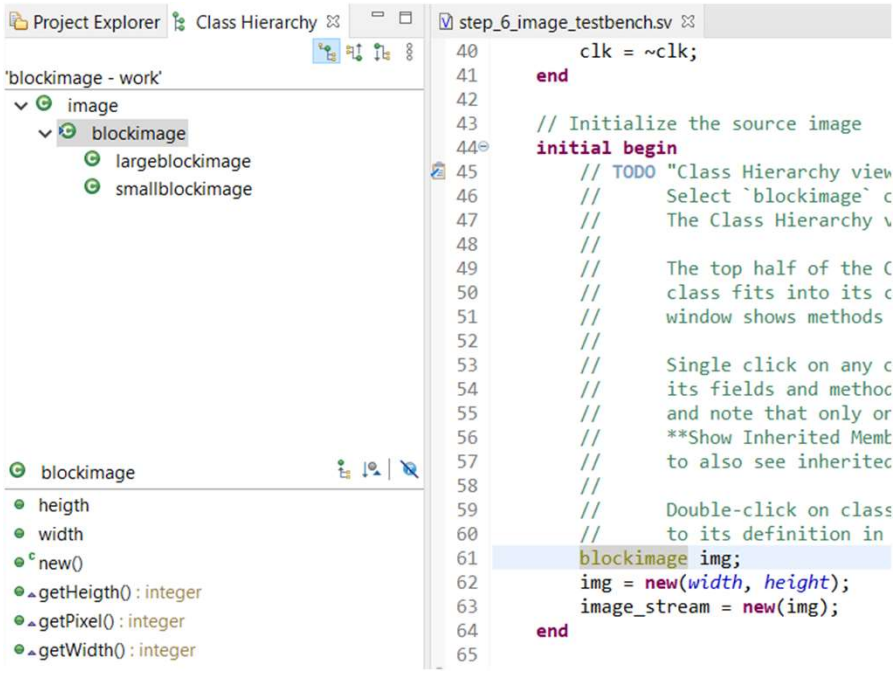
Project Exploration - views

SystemVerilog class hierarchy

Overview of

- (derived) classes
- Members

Navigate to editor

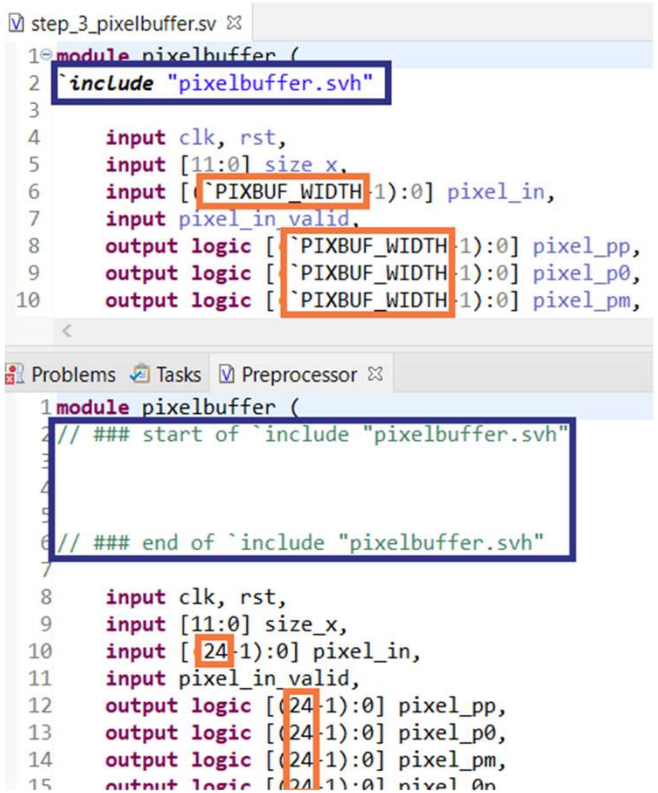


Project Exploration - views

Preprocessor View shows file in the editor, after preprocessing

- Include files are included
- Preprocessor macros expanded

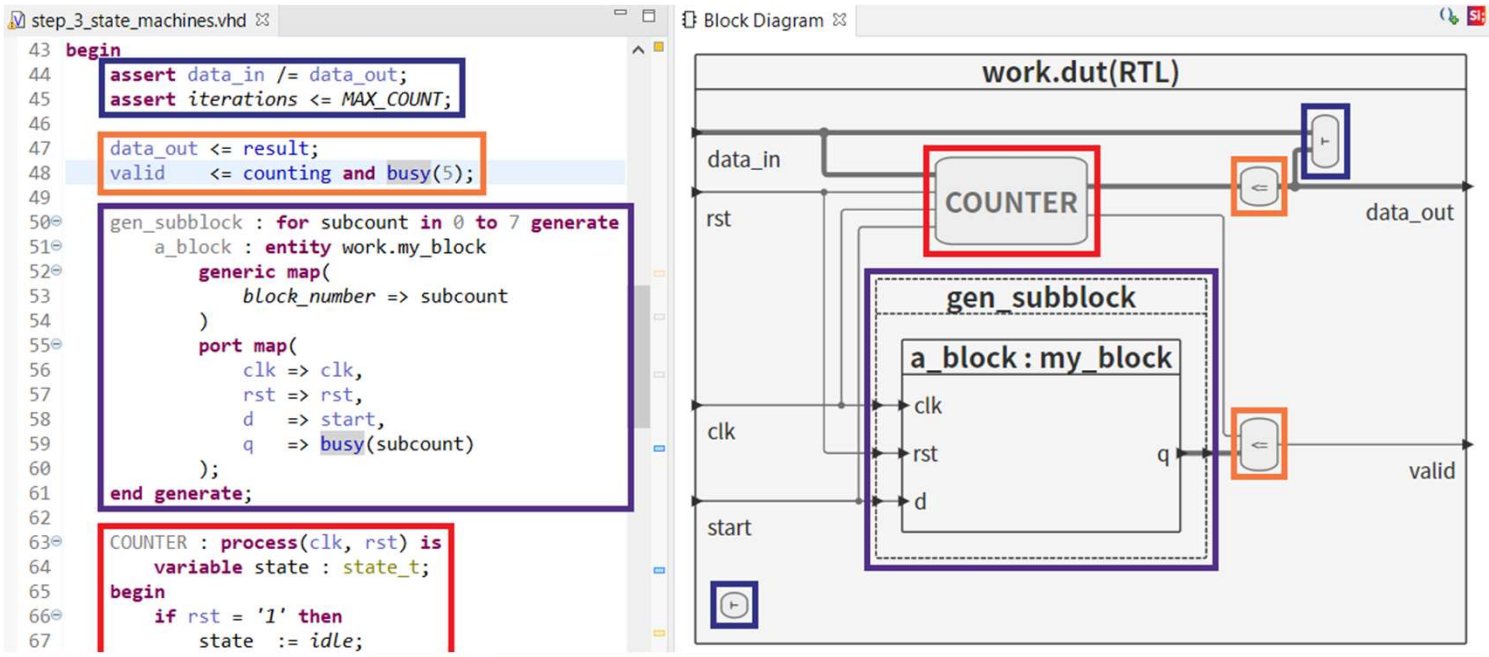
Focus of editor and preprocessor view are linked



Project Exploration - views

Block diagram

- Navigation
- Documentation



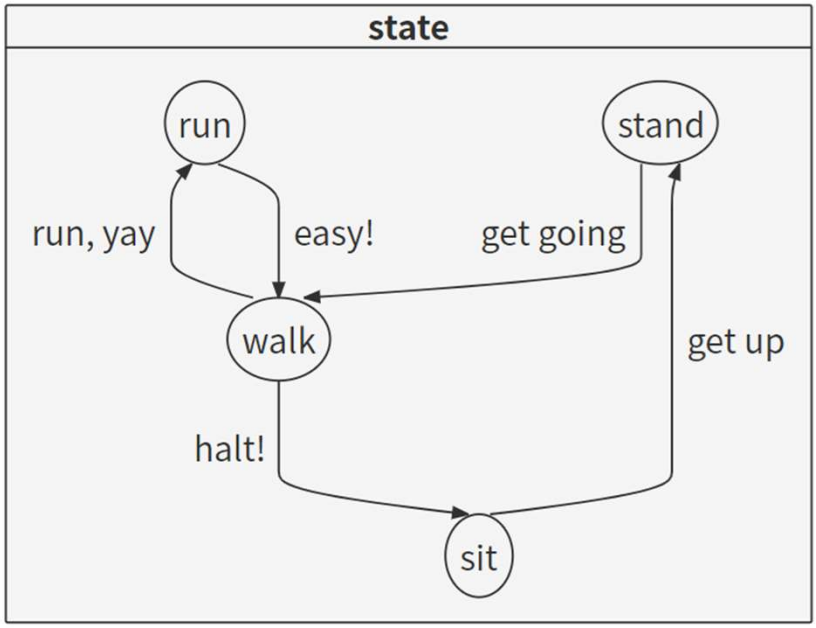
Project Exploration - views

State machine

- Navigation
- Documentation

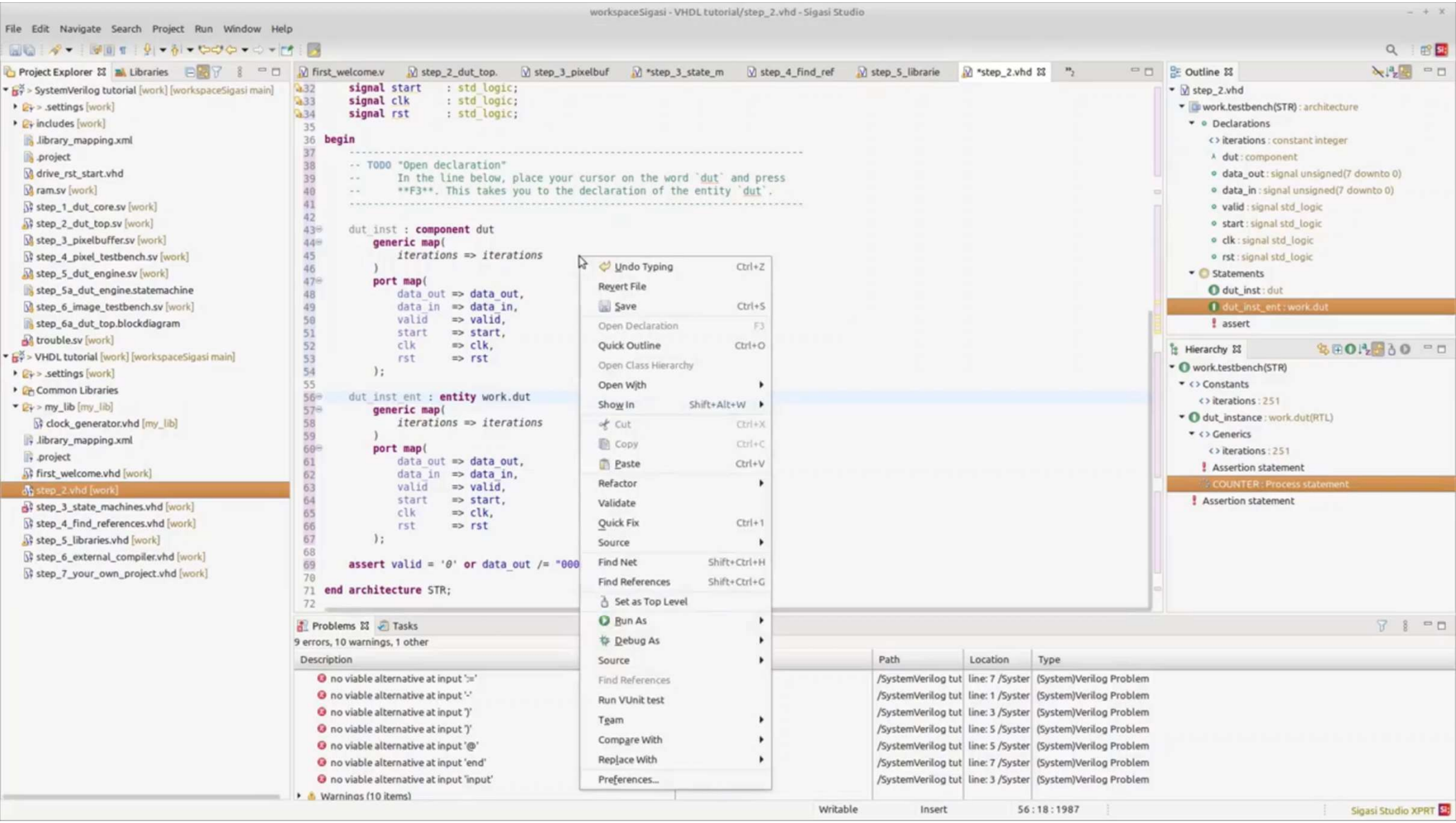
```
module the_fsm(
    input clk,
    input rst,
    input energetic
);

    typedef enum {sit, stand, walk, run} t_state;
    t_state state;
    always @(posedge clk) begin
        if (rst == 1)
            state = sit;
        else
            case (state)
                sit : state = stand;   // get up
                stand : state = walk;  // get going
                walk:
                if (energetic == 1)
                    state = run;       // run, yay
                else
                    state = sit;       // halt!
                run:
                if (energetic == 0)
                    state = walk;      // easy!
                default : state = sit;
            endcase
    end
endmodule
```



Project Exploration

Demo


workspaceSigasi - VHDL tutorial/step_2.vhd - Sigasi Studio
File Edit Navigate Search Project Run Window Help
Project Explorer
Libraries
SystemVerilog tutorial [work] [workspaceSigasi main]
.settings [work]
includes [work]
.library_mapping.xml
.project
drive_rst_start.vhd
ram.sv [work]
step_1_dut_core.sv [work]
step_2_dut_top.sv [work]
step_3_pixelbuffer.sv [work]
step_4_pixel_testbench.sv [work]
step_5_dut_engine.sv [work]
step_5a_dut_engine.statemachine
step_6_image_testbench.sv [work]
step_6a_dut_top.blockdiagram
trouble.sv [work]
VHDL tutorial [work] [workspaceSigasi main]
.settings [work]
Common Libraries
my_lib [my_lib]
clock_generator.vhd [my_lib]
.library_mapping.xml
.project
first_welcome.vhd [work]
step_2.vhd [work]
step_3_state_machines.vhd [work]
step_4_find_references.vhd [work]
step_5_libraries.vhd [work]
step_6_external_compiler.vhd [work]
step_7_your_own_project.vhd [work]
first_welcome.v
step_2_dut_top.
step_3_pixelbuf
*step_3_state_m
step_4_find_ref
step_5_librarie
*step_2.vhd
signal start : std_logic;
signal clk : std_logic;
signal rst : std_logic;
begin
-- TODO "Open declaration"
-- In the line below, place your cursor on the word `dut` and press
-- **F3**. This takes you to the declaration of the entity `dut`.
dut_inst : component dut
generic map(
iterations => iterations
)
port map(
data_out => data_out,
data_in => data_in,
valid => valid,
start => start,
clk => clk,
rst => rst
);
dut_inst_ent : entity work.dut
generic map(
iterations => iterations
)
port map(
data_out => data_out,
data_in => data_in,
valid => valid,
start => start,
clk => clk,
rst => rst
);
assert valid = '0' or data_out /= "000
end architecture STR;
Undo Typing Ctrl+Z
Revert File
Save Ctrl+S
Open Declaration F3
Quick Outline Ctrl+O
Open Class Hierarchy
Open With
Show In Shift+Alt+W
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Refactor
Validate
Quick Fix Ctrl+1
Source
Find Net Shift+Ctrl+H
Find References Shift+Ctrl+G
Set as Top Level
Run As
Debug As
Source
Find References
Run VUnit test
Team
Compare With
Replace With
Preferences...
Outline
step_2.vhd
work.testbench(STR) : architecture
Declarations
iterations : constant integer
dut : component
data_out : signal unsigned(7 downto 0)
data_in : signal unsigned(7 downto 0)
valid : signal std_logic
start : signal std_logic
clk : signal std_logic
rst : signal std_logic
Statements
dut_inst : dut
dut_inst_ent : work.dut
assert
Hierarchy
work.testbench(STR)
Constants
iterations : 251
dut_instance : work.dut(RTL)
Generics
iterations : 251
Assertion statement
COUNTER : Process statement
Assertion statement
Problems
Tasks
9 errors, 10 warnings, 1 other
Description
Path
Location
Type
no viable alternative at input ':='
no viable alternative at input '-'
no viable alternative at input ')'
no viable alternative at input ')'
no viable alternative at input '@'
no viable alternative at input 'end'
no viable alternative at input 'input'
/SystemVerilog tut line: 7 /Syster (System)Verilog Problem
/SystemVerilog tut line: 1 /Syster (System)Verilog Problem
/SystemVerilog tut line: 3 /Syster (System)Verilog Problem
/SystemVerilog tut line: 5 /Syster (System)Verilog Problem
/SystemVerilog tut line: 5 /Syster (System)Verilog Problem
/SystemVerilog tut line: 7 /Syster (System)Verilog Problem
/SystemVerilog tut line: 3 /Syster (System)Verilog Problem
Warnings (10 items)
Writable
Insert
56 : 18 : 1987
Sigasi Studio XPRT

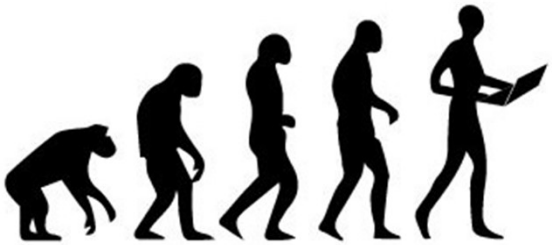
Topics

- Upfront Verification
- Project Exploration
- VS Code extension
  - Learning from software world
  - Language Server Protocol (LSP)
  - Available on marketplace
- CI/CD

VS Code extension

Learning from Software IDEs:

- The Software industry is much larger by number of engineers than the Hardware industry
- Wider market drives larger tool investments
- Adopting Proven Software Practices to Hardware needs is a common trend (i.e., version control, linting, coverage, unit testing)
- Software IDEs are currently a decade ahead
- Catching IDE trends from Software is a natural evolution path for the Hardware domain

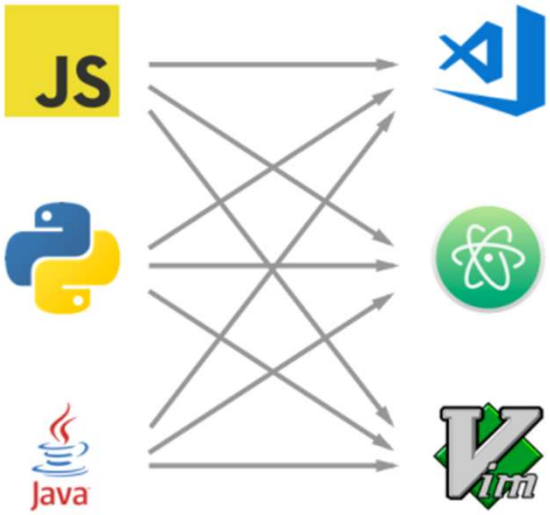


www.broughtonsoftware.com

Language Server Protocol (LSP)

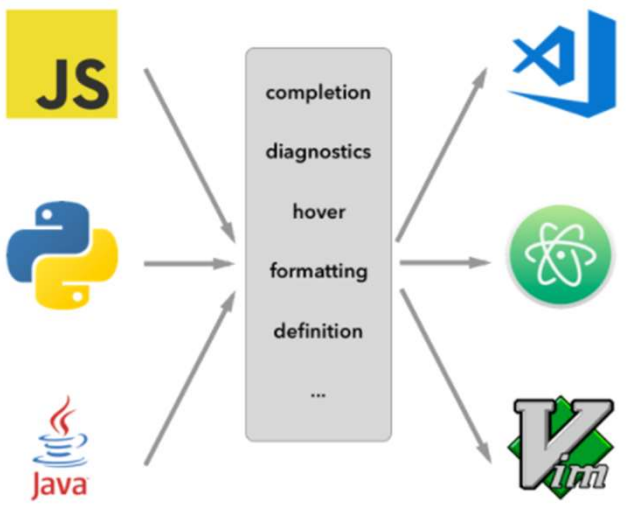
Pre-LSP: Every Language x Every Editor

N x M



LSP: Editors and Language Servers Decouple

N + M

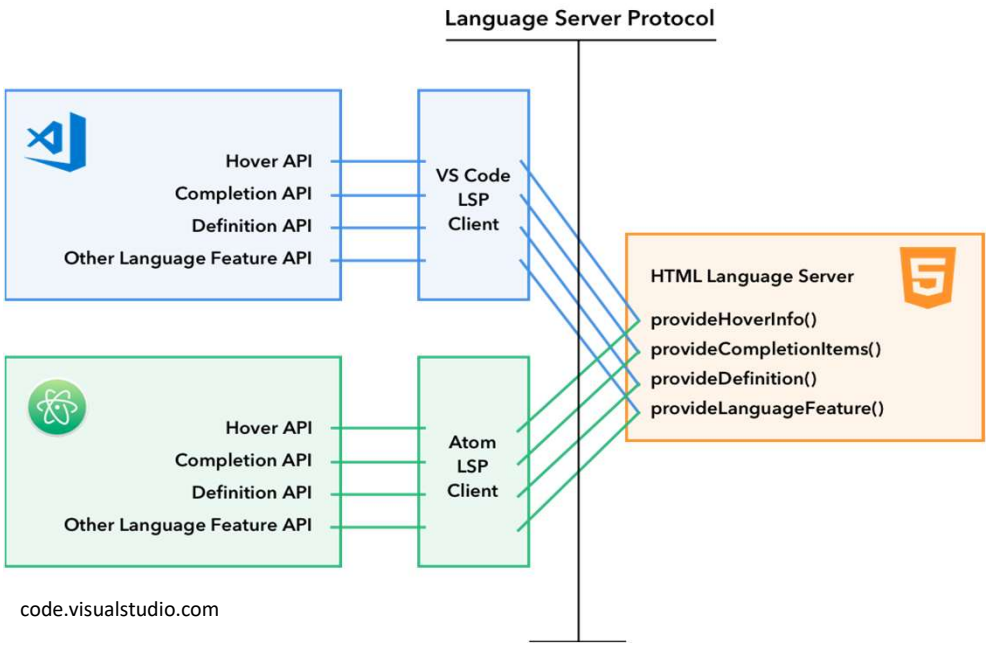


Microsoft

Suggested by Microsoft, adopted by many vendors

images: medium.datadriveninvestor.com

LSP: a Unified IDE-to-Server Protocol



LSP Highlights:

- LSP is a high-level message protocol defining typical operations/requests about documents, positions, ranges
- **IDE-agnostic** & **Language-agnostic**
- Covers IDE operations: hover, auto-complete, jump to definition, error checking, formatting, refactoring, folding, …
- Language servers provide rich editing **capabilities** (not necessarily all of them)
- At low-level based on JSON-RPC communication, standardized message exchange structures and sequences

https://langserver.org/

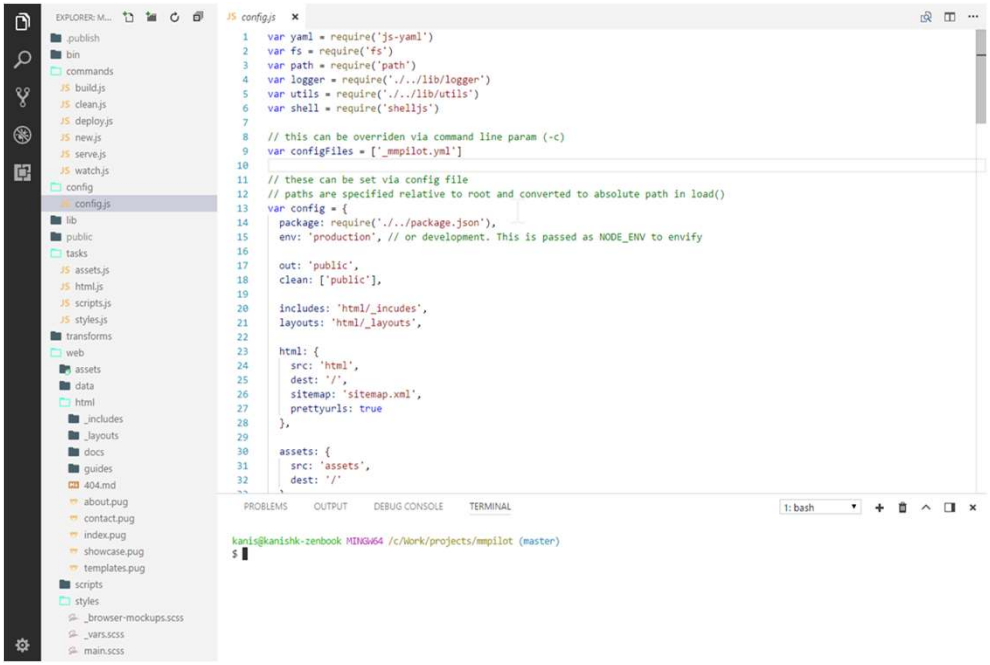
VisualStudio Code: LSP-based tool example

VisualStudio Code

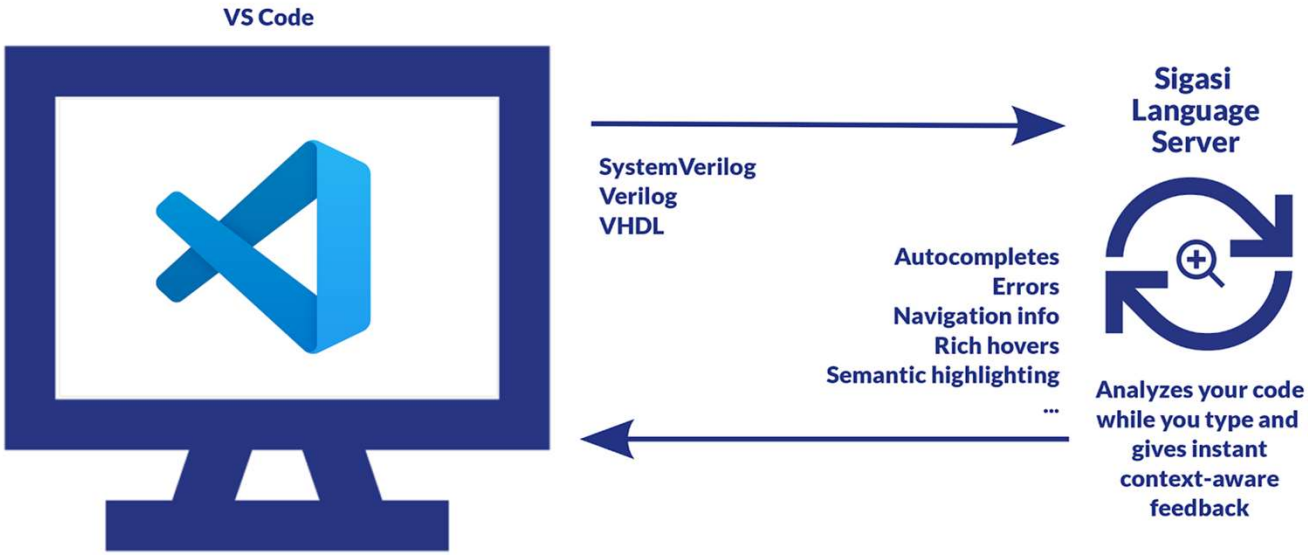
- Cross-platform browser-like app
- Free & Open Source
- Highly customizable with extra plugins
- Supports numerous languages (via LSP)
- Most editing capabilities of desktop IDE, but lightweight in nature
- Targets small projects / components, most popular for web-frontend parts

Other references

- Eclipse Theia
- Atom
- IntelliJ
- Sublime
- Vim + LSP
- Emacs + LSP

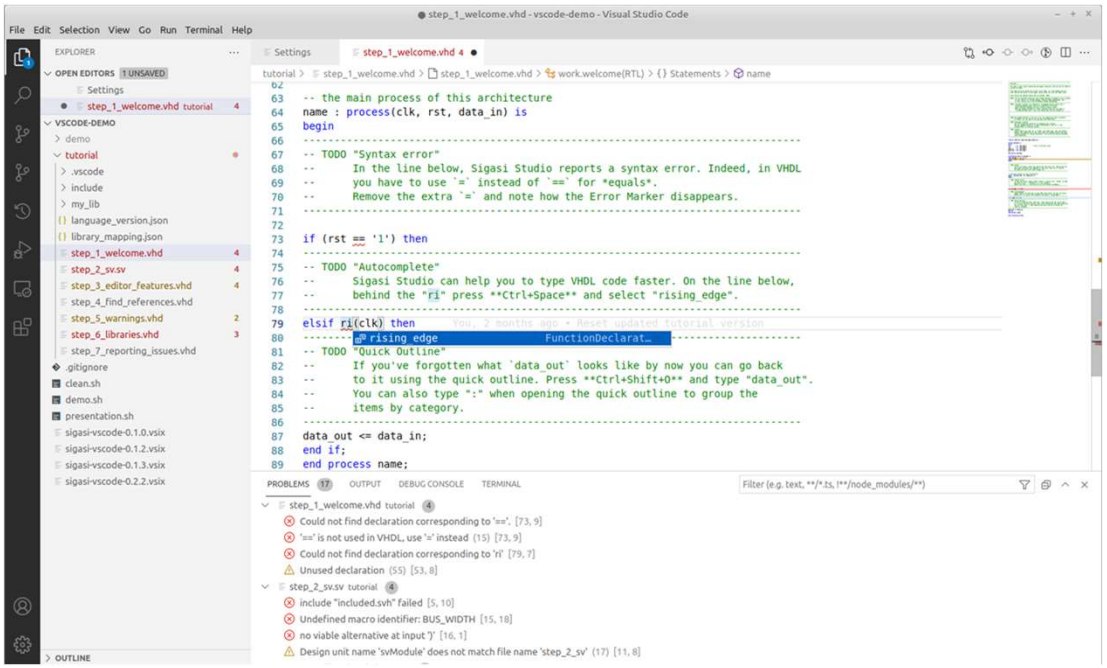


Sigasi Language Server



Sigasi LSP: VS Code extension

www.sigasi.com/vscode/



VS Code extension

Demo

Extension: Sigasi - tutorial - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

TUTORIAL

- .vscode
- include
- my_lib
- language_version.json
- library_mapping.json
- step_1_welcome.vhd 4
- step_2_sv.sv 4
- step_3_editor_features.vhd 4
- step_4_find_references.vhd
- step_5_warnings.vhd 2
- step_6_libraries.vhd 3
- step_7_reporting_issues.vhd

OUTLINE

TIMELINE

Extension: Sigasi | Settings | step_1_welcome.vhd 4

Sigasi v0.3.1

Sigasi | 372 | ★★★★★(1)

Enables Sigasi support for VHDL and (System)Verilog in VS Code

Disable Uninstall

This extension is enabled globally.

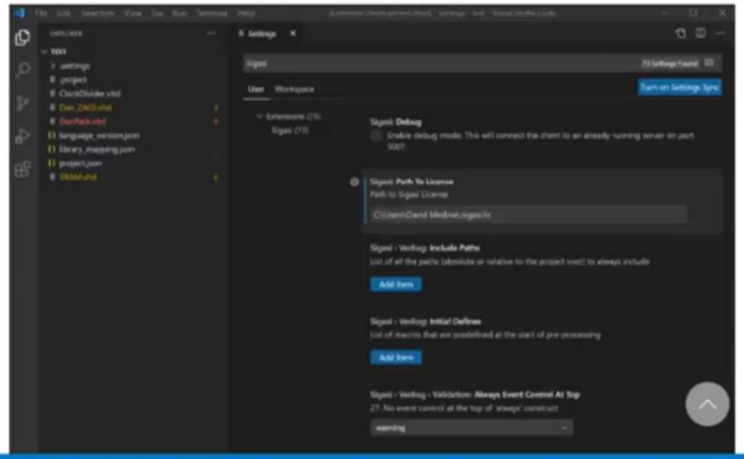
Details Feature Contributions Runtime Status

Sigasi extension for Visual Studio Code

An extension developed by Sigasi with rich support for VHDL and (System)Verilog, including features such as code navigation, project management, linting, code formatting, tooltips, refactoring and much more!

Getting started

- **Step 1.** Set up your Sigasi license:
  1. Open the command palette (**Ctrl+Shift+P**)
  2. Type `Settings`, select *"Preferences: Open Settings (UI)"* and confirm with **enter**
  3. Search for `Sigasi` and navigate to `Path To License`.
  4. Enter the path to your Sigasi license. This can either be a node locked license (`<path to license file>`) or a floating license (`<port>@<server>`).



Categories

Programming Languages | Linters | Formatters | Snippets

Resources

Marketplace
License

More Info

Released on	7/6/2021, 10:23:23
Last updated	7/23/2021, 14:43:35
Identifier	sigasi.sigasi-vscode

master ⊗8 ⚠9

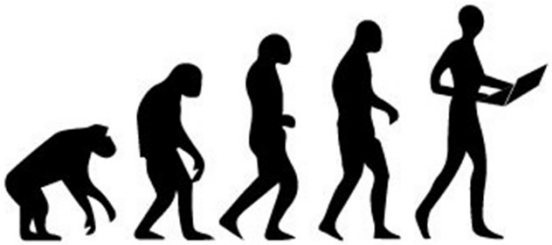
Topics

- Upfront Verification
- Project Exploration
- VS Code extension
- CI/CD
  - Same rules as in Sigasi Studio (Eclipse and VS Code)
  - Align your team
  - Gatekeeper for the CI/CD flow

CI/CD

Learning from Software ~~IDEs~~ methodologies:

- The Software industry is much larger by number of engineers than the Hardware industry
- Wider market drives larger tool investments
- Adopting Proven Software Practices to Hardware needs is a common trend (i.e., version control, linting, coverage, unit testing)
- Software ~~IDEs~~ methodologies are currently a decade ahead
- Catching ~~IDE~~ trends from Software is a natural evolution path for the Hardware domain



www.broughtonsoftware.com

CI/CD

Demo


~/veresta/sigasi-veresta
File Edit View Search Terminal Help
~/veresta/sigasi-veresta
13:34:23

Try it yourself

Explore on your own code what you've seen.

Get your free trial of Sigasi Studio XPRT on:

sigasi.com/try

pxhere.com

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Questions?

You can contact me on

bart.brosens@sigasi.com or support@sigasi.com

accellera SYSTEMS INITIATIVE