



# Sigasi Training

## Setup

---

Requirements:

- Sigasi Studio 4.2
- Valid Sigasi Studio XPRT licenses for all participants
- Training VHDL sources:
  - The sources can be cloned from [https://github.com/sigasi/sigasi\\_training.git](https://github.com/sigasi/sigasi_training.git)
  - The source repository contains folders with VHDL code for each of the following parts, named `part1` through `part4`.
  - The VHDL source files contain `TODO` comments, which explain the tasks of the exercises. This document is an extra tool to guide you through the exercises. It also helps you to keep track of your progress.

## Troubleshooting

---

- Where are my views? → **Window > Perspective > Reset perspective**
- Unexpected error markers? → **Project > Clean...**
- Error markers are not updated? → Make sure **Automatic build** is enabled
- More on <https://insights.sigasi.com/manual/trouble.html>



# Part 1: HDL files

---

The goal of the first part is to get comfortable with the Sigasi Studio HDL editor. By the end you will be able to *easily detect and fix HDL syntax issues, customize editor preferences and know the basic keyboard shortcuts.*

## Setup

- Open the Sigasi application
- In the status bar the status of the Sigasi license is shown
  - If you have a commercial license it should indicate **Sigasi Studio Creator**, **Sigasi Studio XL** or **Sigasi Studio XPRT**
- Import project `Part1`
  1. **File > Import...**
  2. **General > Existing project into Workspace, Next**
  3. **Browse to** `Part1`
  4. **Finish**
- Close unrelated projects (Right Click on the project in Project Explorer and select **Close Unrelated Projects**)

## Edit

- Open file `part1_1_edit.vhd` in the *Project Explorer* view
- Make the editor *full screen* (Double click on the tab)
- Double click again to exit *full screen* mode
- Format the VHDL code (**Ctrl+Shift+F**, *Make sure that no text is selected*)
- Undo (**Ctrl+Z**) your previous change to correct a mistake
- Redo (**Ctrl+Shift+Z**) to restore a change that was undone
- Fix all VHDL syntax issues, use the QuickFix where possible (When in doubt, **Hover** over the error Marker)
- Delete a line with **Ctrl+D**
- Comment a line with **Ctrl+/\*\***
- Move a line up and down with **Alt+Up** and **Alt+Down**
- Jump to the next error using **Ctrl+.**
- Autocomplete (**Ctrl+Space**)
  - Signal Name
  - Constant declaration
  - `if` statement in a `process` (Use **tab** to jump to the next field, use **enter** to exit the *template*)

*mode)*

- `if generate` in an `architecture`
- Block select (**Shift+Alt+A**, or button in toolbar)
  - Delete a Block
  - Add content on multiple lines
- Enable *Show whitespace* (¶-button) and type some *tabs* and *spaces* to see the difference
- Find & Replace
  - Experiment with **Ctrl+F**, e.g. find all `TODO` 's
  - Use **Ctrl+K** to find the next occurrence of the current selection
  - Use **Ctrl+J** (and start typing) for an inline, incremental search. *Look at the status bar to see what you have already typed.*
- Use **Quick Access** (top-right text field or **Ctrl+3**) to convert an identifier to UpperCase (Select the identifier, type `upper` in the Quick Access Field in the Toolbar and confirm with **Enter**)
- Mess up the indentation of a part of the source code, select it and **format only the selection**
- Save all changes to your file (**Ctrl+S**)

## Navigate

- Open file `part1_2_navigate.vhd`
- Go to Declaration can be done in various ways (Hover, Right-click Menu, **F3**)
- Move back to last location (**Alt+Left**)
- Move forward to your original location (**Alt+Right**)
- Go to Line (**Ctrl+L**)
- Navigate to errors with the gutter (Click red rectangles in scroll bar)
- Navigate to next problem marker (**Ctrl+.**, *Ctrl-key and dot-key*)
- Navigate with the *Outline view* (**Ctrl+O**)

## Customize settings

Preferences can be set via **Window > Preferences**.

- Open file `part1_3_custom.vhd`
- Switch to VHDL 2008 (**Preferences > Sigasi > VHDL**) and note that the syntax error get resolved
- After switching VHDL versions, the *Common Libraries* need to be reset (Use the *Quick Fix* offered on the first line of the file)
- Tabs or spaces ( Use the search box in the preference dialog and type `tab` ) Use *Show whitespace* to check.
- Uppercase keywords (**Preferences > Sigasi > VHDL > Formatting**) (Use *format* to check)
- Add a keyboard shortcut for *Show whitespace*.

- Reset the keywords formatting to your preference (uppercase, lowercase or ignore)

## Optional Extra tasks

- Open file `part1_4_extra.vhd`
- Try *structured select* using **Shift+Alt+cursor** <https://insights.sigasi.com/manual/editor.html#structured-selection>
- *Stuttering* (in the editor, double tap the `.`, `;` or `,` key)
- Drag and drop a file from your file explorer in the Editor part of Sigasi Studio
- Create a new file (**File > New > VHDL File** or right click on project or folder)
- In **Outline** and **Project Explorer**: Figure out what *Link with editor* button ( $\Leftrightarrow$ ) does
- Add a keyboard shortcut preference for *show whitespace* (Search for the **Keys** preference page)
- Use the eclipse file search (**Search > Search...**) to do a textual search/replace in you entire project or workspace.



## Part 2: VHDL projects

---

The goal of Part 2 is to understand what a project is, why you want projects and how to set one up. You will learn how to add files to your projects and make sure the VHDL library is set up correctly.

As an extra, you will learn how to efficiently work with Finite State Machines.

### Setup

- Import project `Part2`
  1. **File > Import...**
  2. **Existing project into Workspace, Next**
  3. **Browse to** `Part2`
  4. **Finish**
- Close unrelated projects (Right Click on the project in Project Explorer and select **Close Unrelated Projects**)

### Navigation

- Open `navigation.vhd` with **Open Resource (Ctrl+Shift+R)** or via the Project Explorer
  - Project Explorer: Enable the *Link with editor* button ( $\leftrightarrow$ )
  - Hover over `MAX_COUNT`
  - Open declaration of `std_logic` and other declarations
  - Navigate back (**Alt+Left**)
- Open `components.vhd`
  - Navigate from the component instantiation to the matching entity: press and hold the **Ctrl** key and hover over the component to see a menu with hyperlinks. Click the link to open it.
  - Find References of port `rst`
  - Open two editors side by side (Drag editor tab)
- Navigate through the project with the following Views: (If they are not visible, open them with **Quick Access** or **Window > Show View**)
  - Libraries View
  - Problems View
  - Task View
  - Dependencies View

- Move the views around by dragging them to another location.
- Close views by clicking the X button
- Reset all views by right-clicking on the perspective icon (the small Sigasi icon in the top-right corner) & choose **Reset**

## Edit

- Open file `edit.vhd`
- Use Quickfix to fix the invalid sensitivity list
- Remove trailing whitespace (Use **Quick Access** to find and run this action)

## Generating components and instantiations

- Open file `instantiations.vhd`
- Generate a component for entity `dut`
- Generate an entity instantiation
- Generate the missing signals using the quick-fixes
- Generate a component instantiation
- Autocomplete an instantiation for the component
- Generate a port map and generate a generic map separately
- Use the quick-fix to add the missing port associations

## Libraries

- Open `libraries.vhd`
- Set folder `my_lib` to library `my_lib` (Project explorer: **Set Library**)

## State machines

- Open file `fsm.vhd`
- Declare enum type `state_type : ( init , running , ready )`
- Declare variable `state` of `state_type`
- Autocomplete `case statement`
- Add some transitions
- Add case choice `when idle` and use quickfix to add enum literal to enum type (QuickFix)
- Remove case choice `when idle` and use the quickfix on `case state` to add it again.
- **Ctrl+Hover** on a state assignment and select **Open Matching when Clause**.

## Rename

- Open file `rename.vhd`



- Manual rename: Change (edit) `port_1` to `port_1a` and manually update the matching instantiations
- Rename (**Refactor > Rename element** or **Shift+Alt+R**) port `port_a` to `port_aa`
- Save the file (**Ctrl+S**)
- Compare file with *Local history* (Right-click on file, **Compare With > Local History...**) and inspect the changes you made

## Import (non-Sigasi) project + setup libraries

- First: Check that you are in VHDL 2008 mode
- **File > Import...**
- **Sigasi > Import a VHDL project, Next >**
- **Browse...** to `vunit_example`, **Finish**
- Set the project to library `uart_lib` (right click, **Set Library**)  
Check the library annotations in the project explorer
- Add library `vunit_lib`:
  - Drag and drop the `vunit_lib` folder from your File System Explorer (e.g.: Windows Explorer, Mac Finder, Nautilus) into the `Common Libraries` folder (in the Project explorer). Choose the option to *link to files and folders*.
  - Set the library to `vunit_lib`
- Set the library for the test files ( `uart/src/test` ). Read the source code for the library name.
- One more library is not correctly set. Try to fix this. (Tip: the sources are already in the project)

## Optional Extra Tasks

- Select `edit.vhd` and `navigate.vhd` and compare them via the right click menu.
- Configure and enable an External Compiler (See `extra_compiler.vhd` )
- Set the top level and start the Simulator (See `extra_compiler.vhd` )
- Export a CSV file with all files in your project (**File > Export... > Sigasi > CSV file**)



## Part 3: Linting and other features

---

In Part 3 you will learn about VHDL linting (code checks) and how you can get to zero warnings.

We will also explore other features that are not specific to VHDL.

### Linting

- Import Project `part3`
- Close unrelated projects (Right Click project in Project Explorer)
- Fix all **Errors** (Track your progress in the Problems View)
- Fix all `--TODO: fix` **Warnings** (Track your progress in the Task View). We will get to the other warnings later.
- Change the severity of the **Check for position associations in instantiations** linting via the Preferences (**Sigasi > VHDL > Errors/Warnings**). Set to severity to **Error**. Note that this warning in the code is now an error.

### Custom problems view

- Open the **Problems View**
- Click on the triangle on the top-right of the view
- Select **New Problems View**
- Give it the name `Problems of this file`
- Move the view next to the original **Problems view**
- Click the triangle in your new **Problems view**
- Choose **Filters...**
- Untick **Show all items**
- Select only the Configuration **Errors/Warnings on Selection**
- Limit the scope to **On selected elements only**
- The new problems view will only show problems in the currently selected file.

### Waive specific warnings

- Open `waive.vhd`
- Suppress the *Null Range* warning by adding `-- @suppress`
- The warning disappears shortly after typing the `@suppress` comment
- Note that the warning marker is removed from the **Problems View** once you save the file
- Explicitly specify which warning you suppress by adding the warning message prefix  
`-- @suppress "Null range"`
- Note that the warning marker re-appears when you type another prefix or make a typo
- Now also explain why you suppress the warning by adding an explanation

```
-- @suppress "Null range" Ok here, see http://...
```

## VHDL version

- Set the workspace VHDL version to '93 via the Preferences (**Sigasi > VHDL**)
- Open `version.vhd`
- Review the errors
- Change the VHDL version of the project
  - **Right-Click** on the project and select **Properties**
  - Change the version to 2008
- Review the errors in `version.vhd`
- Force a clean-build (**Project > Clean...**)
- The project still uses the 93 libraries, replace them with the 2008 libraries by selecting **Right-Click project > Set Library > Reset Library Mapping**
- All errors should be fixed

## Other features

- Project Explorer
  - Show hidden files (Click ▾, **Filters and Customization**, disable the `.* resources` )
  - Delete a file from your project. Next, restore this file from local history: In the Project Explorer, right click the project **Part3** and select **Restore from Local History...**
- Bookmarks:
  - Right click a line number and select **Add Bookmark**
  - Open the Bookmarks view (**Window > Show View > Other... > General > Bookmarks**)
- Multiple screen support
  1. Open a new Window (**Window > New Window**)
  2. Drag the new windows to a different screen
- Customize Autocomplete templates: Add a custom header to the `entity/architecture pair` template:
  1. **Window > Preferences > Sigasi > VHDL > Templates**
  2. Select **entity/architecture pair**
  3. Click **Edit...**
  4. Add a custom header (e.g. `-- Confidential, Copyright 2016` )
  5. Confirm with **OK**
  6. Create a new VHDL file and select your customized template (**File > New > VHDL file**, choose a name, **Next** and choose the `entity/architecture pair` template)

## Optional Extra tasks

- Customize your perspective:
  1. **Window > Perspective > Customize Perspective...**
  2. Enable the **Print** button in **File**
- Configure the **Naming Convention** settings to check for *Uppercase* constants ( `[A-Z_]+` )
- Open and explore the Properties Page (Right click file, **Properties**)
- Multiple Search Views
  1. Run a first search (e.g. Find Reference on `std_logic` )
  2. Click the **Pin the Search View** button
  3. Start a new search. This search will open in a new *Search View*
- Configure project specific settings <https://insights.sigasi.com/manual/linting.html#project-specific-linting-settings>
- Experiment with customized Templates
  - Use different contexts: DesignFile, ConcurrentStatement, AnyWhere
  - Explore the pattern syntax by looking at the existing templates.
- Create a **Problems View** that only shows problems from the **External Compiler** in the selected Project



## Part 4: Hierarchy, Graphics and documentation


---

In Part 4 you will learn how to use the Hierarchy View. You will also learn how to use and export Graphics and Documentation with Sigasi Studio.

### Hierarchy View

- Import project `part4` (and close unrelated projects)
- Open `testbench.vhd`
- Right click the `STR` architecture and select **Set as Top Level**
- Explore the resulting tree in the Hierarchy View:
  - Inspect the value of generics (Notice how the value in `dut_instance` differs from its default value)
  - Change the value of a generic and note how the Hierarchy View refreshes when you save the file.
  - Add a syntax error in a file and note that Sigasi Studio is able to recover and still show valid content in the Hierarchy View.
  - Disable the **Toggle Hierarchy Auto Refresh** button, and note Sigasi Studio no longer auto-updates the hierarchy when edit your files. (This is useful for big hierarchies)

### Block Diagram View


- Open `testbench.vhd`
- Right click the VHDL editor and select **Show in > Block Diagram**
- Double click ports or connections to navigate to the corresponding VHDL code. (This also works the other way around by clicking **Show in > Block Diagram** on ports in the VHDL code)
- Make some changes in the VHDL code and note how the Block Diagram updates when you save your file.
- Select `dut_instance` in the diagram, right click and select **Open Entity Declaration** to navigate to the corresponding entity, `dut`.
- Click the save icon , and export the diagram to a PNG or SVG file.

### State Machine View

- Click **Window > Show View > State Machines** to open the State Machine view
- Open `dut.vhd`
- Double click on a few nodes and transitions in to navigate to the corresponding VHDL code.
- Change the VHDL code of the state machine and notice how the view updates (See the TODO tag)
- Add a comment to the `when idle: state := preparing;` transition. For example:

```
-- start preparing
```

Notice how the transition labels update
- Toggle the (Aa)-button to show/hide the transition labels.

- Click the save icon  to export the diagram to file.

## Graphics Configuration

- Open `testbench.vhd`
- In the Block Diagram View, press the **Group all wires between blocks** button. This button creates a new Graphics Configuration file that groups all wires between blocks, except for clock and reset lines.
- You can change the name of the new bus and control the wires that are grouped by it.
- In the new blockdiagram file, define a grouped block by adding `def block group tbcontrol (identifiers)`. Note that you can use the autocomplete to avoid having to type the identifiers. Create a group *tbcontrol* that contains the instances of the *clock\_generator* and the *stimgen*.
- Give a color to the *dut* instance by adding `block dut_instance { color COLOR}`.
- Also give the *stimgen* or *clock\_generator* blocks a color. Note that these need to be accessed differently because of the group they are in. Remember the auto-complete for help.
- Hide the details in *tbcontrol* by collapsing this group. Use `block tbcontrol { collapse}`.
- Navigation: note that you can also navigate from the identifiers in the blockdiagram file to the HDL code.
- Edit the HDL code, e.g. by modifying the label of an instantiation, and note that this will cause errors in the blockdiagram file.
- Documentation and more examples for Graphics Configuration can be found on <https://insights.sigasi.com/manual/graphics.html>

## Documentation

- Import project `Sigasi_doc`
- Open the Documentation View
- Open file `testbench.vhd`
- Edit comments and note how the documentation is updated:
  - Document a port (by adding a comment)
  - Document a generic
  - Document an architecture
  - Document an entity
- Experiment with Markdown
  - Paragraphs and line breaks
  - *emphasis* and **strong**
  - lists and tables
  - external links and email addresses
- Export PDF
  - Explore the generated files in the *sigasi-doc* folder
  - Inspect the DocBook file



## Net search

- In `stimgen.vhd` (Project `part4`)
  - Select a the port `stim_data`
  - **Right-Click > Find Net**
  - Review the result in the **Net Search View**
  - Perform a net search on `clk`
- In `testbench.vhd`
  - Find the instantiation for `stimgen`
  - Perform a net search on `stim_data`

## Optional Extra tasks

- State Machine View: Add a second state machine in `dut.vhd` and note how it appears in the State Machine View.
- Graphics Configuration: use a regex to group some of the signals in the blockdiagram file instead of listing all wire names.
- Add a Graphics Configuration file for the state machine in `dut.vhd`.
- Finish the documentation of the Project to get a clean and complete PDF.