



# DVT Eclipse IDE

## *My First VHDL Project*

# > The DVT Perspective



- Switch to the **DVT Perspective** from

*menu Window > Open Perspective > Other... > DVT*

- The DVT Perspective presents different Views (GUI components) around the Editor



## Views

*Types*

*Compile Order*

*Design Hierarchy*

*Trace Connections*

...

# > The Project Location



- You typically create a project in a folder that contains the source code files

*It is not mandatory to create a project where the source files are  
All "outside the project" sources will be presented under DVT Auto-Linked*

- DVT creates **.project** and **.dvt** in the project location folder

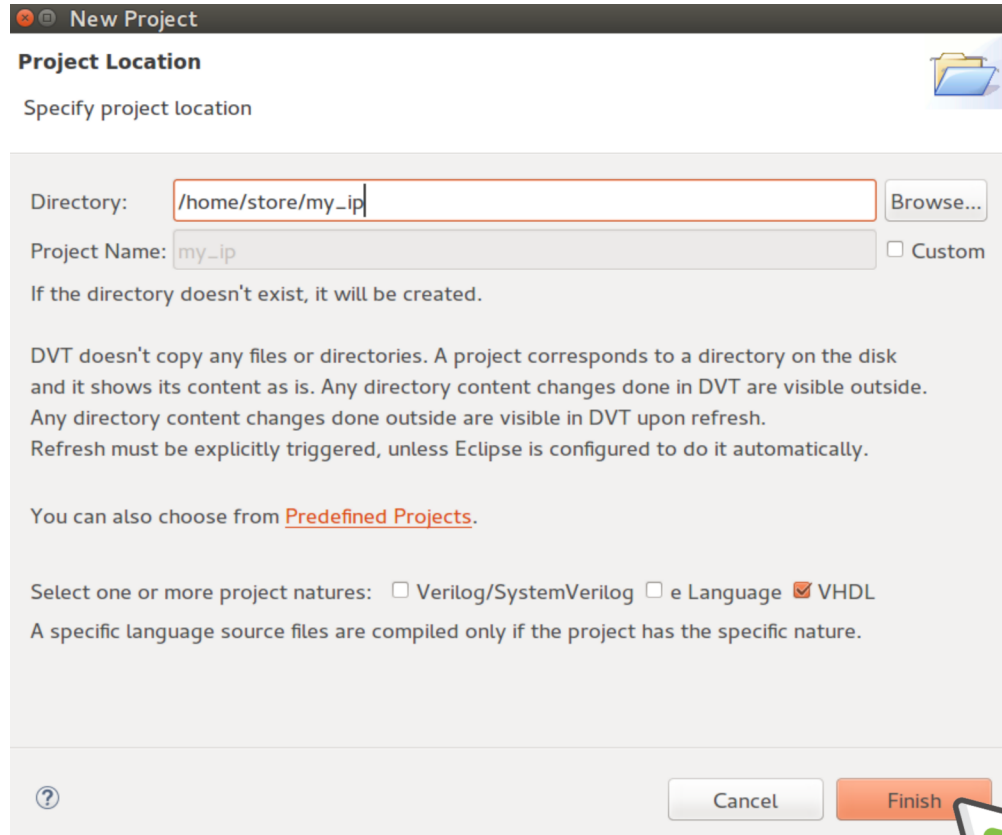
*The .project file indicates to Eclipse that a project exists in the folder  
The .dvt folder contains various DVT specific project settings*

```
my_ip/  
  vhdl/  
    core/  
    examples/
```



```
my_ip/  
  .project  
  .dvt  
  vhdl/  
    core/  
    examples/
```

# > The New Project Wizard



- Invoke the wizard

*From menu File > New > DVT Project*

- Specify the project location
- Specify the project nature. If a project was already configured, that is if .project and .dvt already exist, the wizard recognizes the existing project

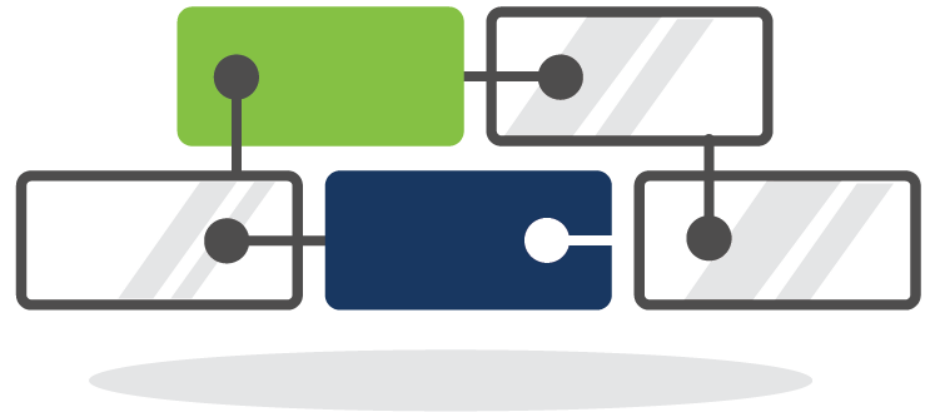
*You can open one of the **Predefined Projects**, if you want to see how an example project is configured*



# > Build Configurations



- In order to provide advanced functionality (like hyperlinks, autocomplete, design and class hierarchy, error signaling, etc.) DVT analyzes the source code files in your project. This analysis process is called **build**
- In order to build, DVT uses the compilation arguments that you specify in a build file. The default build file is **.dvt/default.build**
- By default DVT scans the project folder and automatically detects how to compile the source code files. This is specified by the **+dvt\_init\_auto** directive used by default in the build file



# > The .build File Syntax



- You can change the `.dvt/default.build`
- In a `.build` file you can specify:

*Absolute paths or project root relative paths*

*System variables like `#{var}` or `%var%`*

*-work lib1 to set a specific library for certain files -- NOTE: only one -work per +dvt\_init section is allowed*

*-f <path> or -F <path> to include a file containing more arguments*

- For more options see: [https://www.dvteclipse.com/documentation/vhdl/Build\\_Configurations.html](https://www.dvteclipse.com/documentation/vhdl/Build_Configurations.html)
- In order to reuse existing argument files that you pass to a simulator, DVT supports several compatibility modes like `vcs`, `ius` or `questa`

# > Build the Project

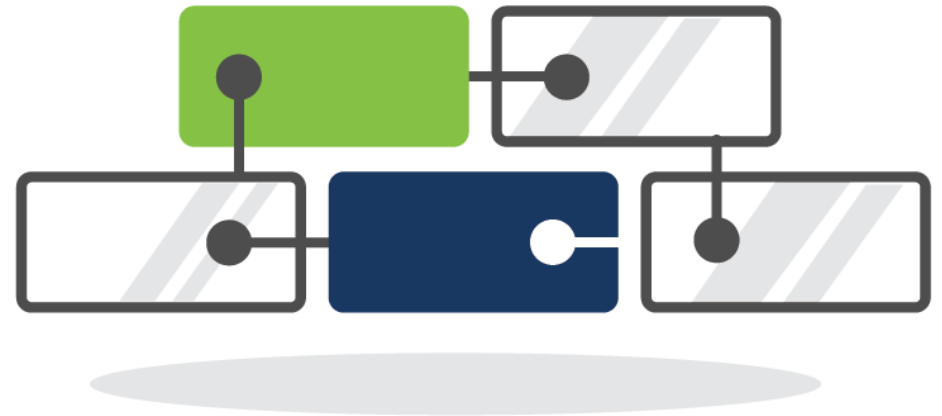
- Building a project means compiling and indexing all the source files in order to provide hyperlinks, autocomplete, design browsing ...

*Full Build or Rebuild = compile all using directives from the current build file*

*Incremental Build = compile changes. As you edit files, DVT incrementally builds the project.*



- After changing .dvt/\*.build, you have to Rebuild the project from the toolbar button



# > Check the Build



- The **Compile Order View** shows all the compiled files

*You can open the Compile Order View from menu Window > Show View > Compile Order  
It is typically located on the right side of the Editor, behind the Outline View*

- The **Problems View** shows all the errors and warnings in your project

*Build Errors: file not found ...*

*Syntax Errors: unexpected token "sgnal" instead of "signal" ...*

*Semantic Errors: duplicate entities, type not found ...*

*You can open the Problems View from menu Window > Show View > Problems*

*It is typically located below the Editor*

# > Features Overview [1]



- **Hyperlinks:** in the editor press Ctrl and hover entity names, procedure names, and in general any identifier. A hyperlink appears. Click the it to go to the definition. In addition to the hyperlink, a drop-down presents more options, for example Show Usages
- **Show Usages:** in the editor press Ctrl and hover an identifier. From the drop-down chose Show Usages to see all places were a variable, signal, procedure, entity etc. is used
- **Autocomplete:** in the editor Ctrl + Space triggers autocomplete. For example usb<Ctrl+Space here> will show all signals starting with usb
- **Quick Fixes:** in the editor on a line with errors Ctrl +1 pops-up quick fix proposals to correct typos, to declare missing variables etc.
- **Rename Refactoring:** place the cursor over an identifier and right click > Refactor > Rename or Shift+Alt+R to rename and update all usages across the entire project



# > Features Overview [2]



- **Design Hierarchy:** place the cursor over the entity name and press Shift + F4 to see the design structure
- **Diagrams:** Right click on an entity / variable to get schematics / state machine diagrams
- **Trace Connections:** Right click on a signal and Trace > Trace Driver / Load
  
- **All Entities/Packages/Types/...:** menu Window > Show View > Types
- **All TODOs/FIXMEs:** menu Window > Show View > Tasks
  
- **Code Templates:** menu Window > Show View > Code Templates
  
- **To quickly find an entity or type:** Ctrl + Shift + T
- **To quickly open a file:** Ctrl + I or Ctrl + Shift + R

# > Features Overview [3]



- **Semantic Search:** `Ctrl + H` to search for an entity, signal... both definitions and/or where it is used
- **Code Formatting:** press `Ctrl+Shift+F` to format the whole editor or selection
- **Toggle Comment:** `Ctrl + /` for current line or selection
- **Matching Begin – End:** double-click on the entity, begin, end, procedure ...
- **Column Selection:** `Shift+Ctrl+A` or from the main toolbar button
- **All Shortcuts:** press `Ctrl + Shift + L` to pop-up a list of all shortcuts



**And many more, please contact [support@amiq.com](mailto:support@amiq.com) for a demo.**

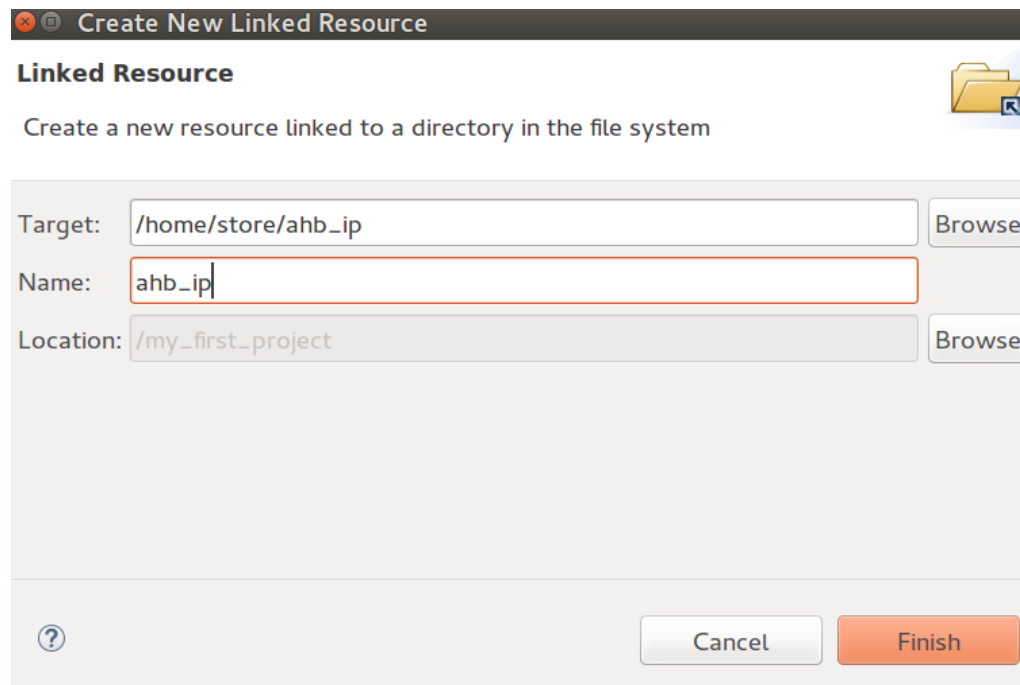
# ➤ Advanced: Linked Resources [1]



- Sometimes your source code is spread on the disk, not everything is under one “source code root” folder
- Files outside the Project Location folder (“external files”) are by default automatically brought during a full build under the **DVT Auto-Linked virtual folder**
- What you see there are the actual files, any change will be visible on the disk and the other way around
- The hierarchy under DVT Auto-Linked may be too deep. To bring the files “closer” you can:
  - Use **+dvt\_auto+link\_root+alias=/path/to/folder** directives in **default.build**
  - Use **Linked Resources**. Linked Resources are logical entities, no additional files are created on your disk. Their definition is stored in the **.project** file

# > Advanced: Linked Resources [2]

Right click on Project > **New** > **DVT Linked Resource**, specify the target location and linked resource name



**Create New Linked Resource**

**Linked Resource**

Create a new resource linked to a directory in the file system

Target: /home/store/ahb\_ip

Name: ahb\_ip

Location: /my\_first\_project

**Rebuild** the project in order to avoid duplicate files in the newly created folder and DVT Auto-Linked

## > More Information



- Demo Movies: <https://www.youtube.com/user/dvteclipse>
- Step by step basic tutorial: [https://www.dvteclipse.com/documentation/vhdl/Getting\\_Started.html](https://www.dvteclipse.com/documentation/vhdl/Getting_Started.html)
  - Please contact us for more training materials
- Features with snapshots: [https://www.dvteclipse.com/documentation/vhdl/Tips\\_and\\_Tricks.html](https://www.dvteclipse.com/documentation/vhdl/Tips_and_Tricks.html)
- User Guide: <https://www.dvteclipse.com/documentation/vhdl/index.html>
- Datasheet: [https://dvteclipse.com/datasheet/AMIQ\\_DVT\\_Datasheet.pdf](https://dvteclipse.com/datasheet/AMIQ_DVT_Datasheet.pdf)



**Mail to [support@amiq.com](mailto:support@amiq.com)**