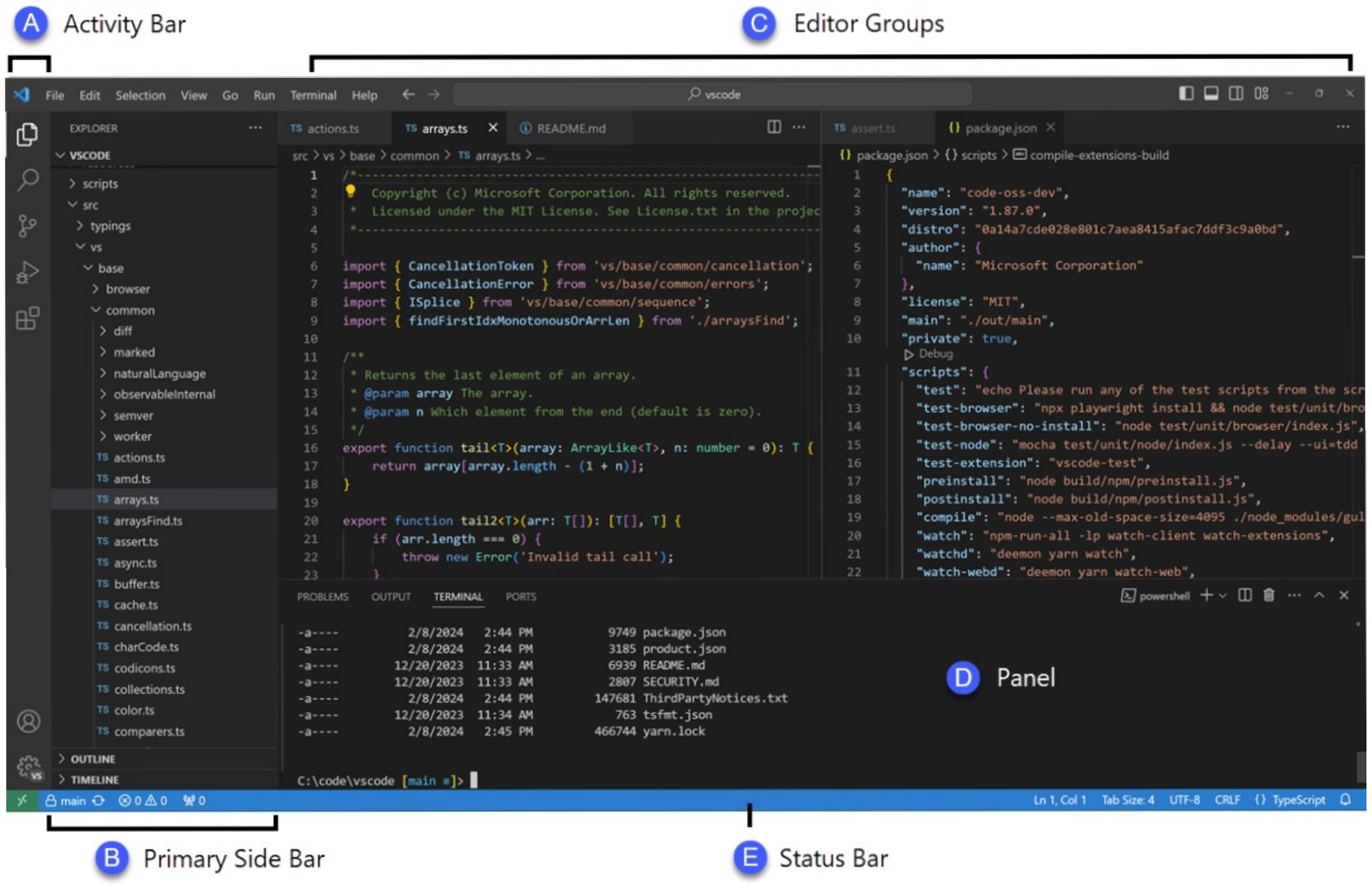
DVT IDE for VS Code

My First Verilog/SystemVerilog Project

VS Code Layout



```
/*---------------------------------------------------------------
 *  Copyright (c) Microsoft Corporation. All rights reserved.
 *  Licensed under the MIT License. See License.txt in the projec
 *---------------------------------------------------------------

import { CancellationToken } from 'vs/base/common/cancellation';
import { CancellationError } from 'vs/base/common/errors';
import { ISplice } from 'vs/base/common/sequence';
import { findFirstIdxMonotonousOrArrLen } from './arraysFind';

/**
 * Returns the last element of an array.
 * @param array The array.
 * @param n Which element from the end (default is zero).
 */
export function tail<T>(array: ArrayLike<T>, n: number = 0): T {
    return array[array.length - (1 + n)];
}

export function tail2<T>(arr: T[]): [T[], T] {
    if (arr.length === 0) {
        throw new Error('Invalid tail call');
    }
```

```
{
  "name": "code-oss-dev",
  "version": "1.87.0",
  "distro": "0a14a7cde028e801c7aea8415afac7ddf3c9a0bd",
  "author": {
    "name": "Microsoft Corporation"
  },
  "license": "MIT",
  "main": "./out/main",
  "private": true,
  ▷ Debug
  "scripts": {
    "test": "echo Please run any of the test scripts from the scr
    "test-browser": "npx playwright install && node test/unit/bro
    "test-browser-no-install": "node test/unit/browser/index.js",
    "test-node": "mocha test/unit/node/index.js --delay --ui=tdd
    "test-extension": "vscode-test",
    "preinstall": "node build/npm/preinstall.js",
    "postinstall": "node build/npm/postinstall.js",
    "compile": "node --max-old-space-size=4095 ./node_modules/gul
    "watch": "npm-run-all -lp watch-client watch-extensions",
    "watchd": "deemon yarn watch",
    "watch-webd": "deemon yarn watch-web",
```

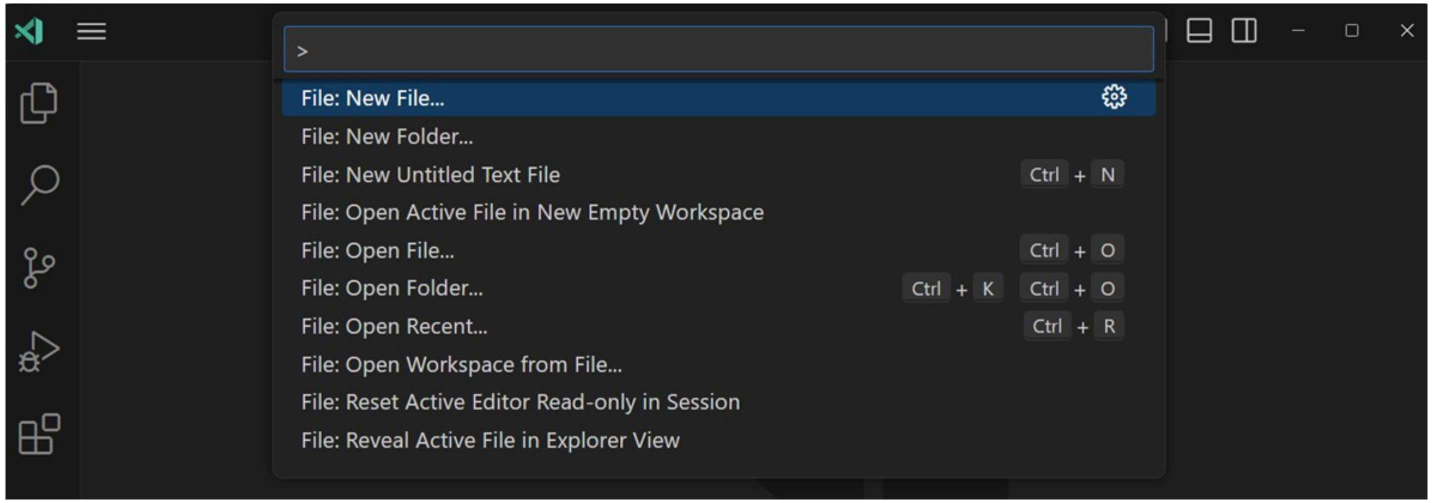
```
-a----          2/8/2024   2:44 PM           9749 package.json
-a----          2/8/2024   2:44 PM           3185 product.json
-a----        12/20/2023  11:33 AM           6939 README.md
-a----        12/20/2023  11:33 AM           2807 SECURITY.md
-a----          2/8/2024   2:44 PM         147681 ThirdPartyNotices.txt
-a----        12/20/2023  11:34 AM            763 tsfmt.json
-a----          2/8/2024   2:45 PM         466744 yarn.lock

C:\code\vscode [main ≡]>
```

Source: https://code.visualstudio.com/docs/getstarted/userinterface#_basic-layout

Command Palette

One of the most essential features of VS Code is the **Command Palette**, which allows you to find and access all functionalities, including keyboard shortcuts for common operations.


```
>
File: New File...
File: New Folder...
File: New Untitled Text File                          Ctrl + N
File: Open Active File in New Empty Workspace
File: Open File...                                    Ctrl + O
File: Open Folder...                       Ctrl + K   Ctrl + O
File: Open Recent...                                  Ctrl + R
File: Open Workspace from File...
File: Reset Active Editor Read-only in Session
File: Reveal Active File in Explorer View
```

Source: https://code.visualstudio.com/docs/getstarted/userinterface#_command-palette

*Use **View → Command Palette...** or the **Ctrl+Shift+P** keybinding to open the Command Palette.*

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 in the **Compiled Files** and **Compile Order** views from the DVT Activity.*

- DVT creates a **.dvt** directory within the project’s root folder, containing various DVT specific project settings.

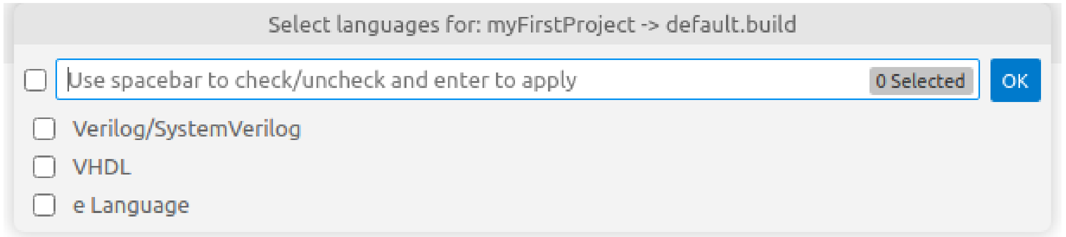
```
my_vip/
    sv/
        core/
        examples/
```

→

```
my_vip/
    .dvt
    sv/
        core/
        examples/
```

Creating a new DVT Project

- From Command Palette, invoke the *DVT: Create a Project...* command
- Specify the project location
- Specify the project nature. (This step is necessary only if the project was not already configured ⟺ **.dvt** directory doesn't exists)

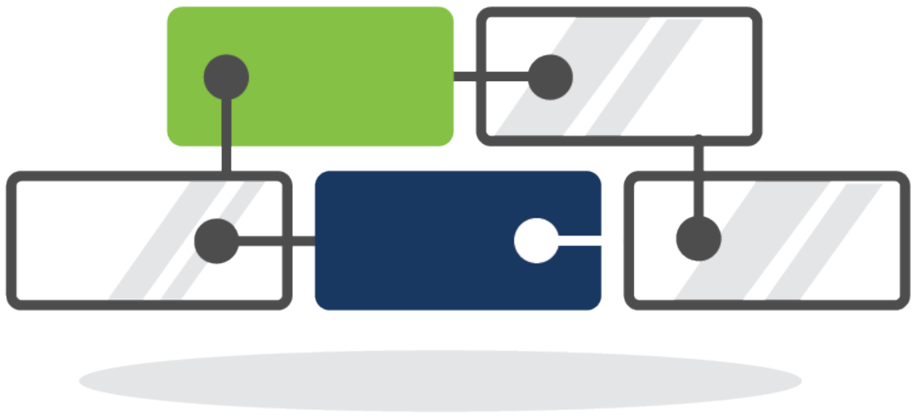

Select languages for: myFirstProject -> default.build

- [ ] Use spacebar to check/uncheck and enter to apply — 0 Selected — OK
- [ ] Verilog/SystemVerilog
- [ ] VHDL
- [ ] e Language

*Use the **DVT: Open a Predefined Project...** command to open one of the **Predefined Projects**, if you want to see how an example project is configured.*

Build Configurations [1]

- In order to provide advanced functionalities (like error signaling, hyperlinks, autocomplete, design and UVM components hierarchy, 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.

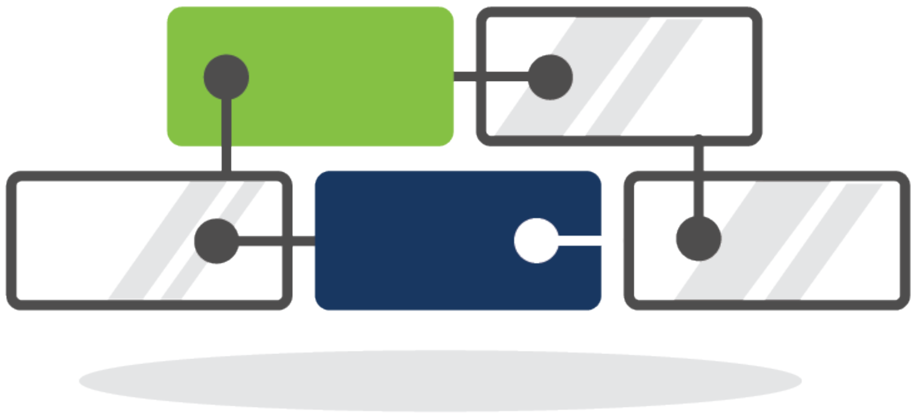


Build Configurations [2]

A quick way to set up the build configuration is to start from **a simulation log**.
Simply add **+dvt_init_from_simlog+simulation.log** to your build file.

*For a robust and scalable flow integration consider reusing the simulator arguments, for example via a dedicated **dvt_ide Makefile** target.*

FPGA Support
Create the DVT project in the same location as an existing Quartus / Xilinx ISE / Vivado project. All source files and settings defined in the Quartus / Xilinx ISE / Vivado project configuration files will be automatically recognized.



The .build File Syntax

- In a **.build file** you can specify:
  - Absolute paths or project root relative paths
  - System variables like *$var*, *${var}* or *%var%*
  - *+incdir+<path>* directives to indicate search directories for files included with *`include "filename"*
  - *+define+<DEFINE>=<replacement>* or *-define <DEFINE>=<replacement>* directives
  - *-v <file>* or *-y <dir>* to specify a Verilog source library file/directory
  - *-f <path>* or *-F <path>* to include a file containing more arguments

- For more options see: **https://eda.amiq.com/documentation/vscode/sv/toc/build-config/index.html**
- In order to reuse existing argument files that you pass to a simulator, DVT supports several compatibility modes like **vcs**, **ius**, **xcelium** or **questa**.

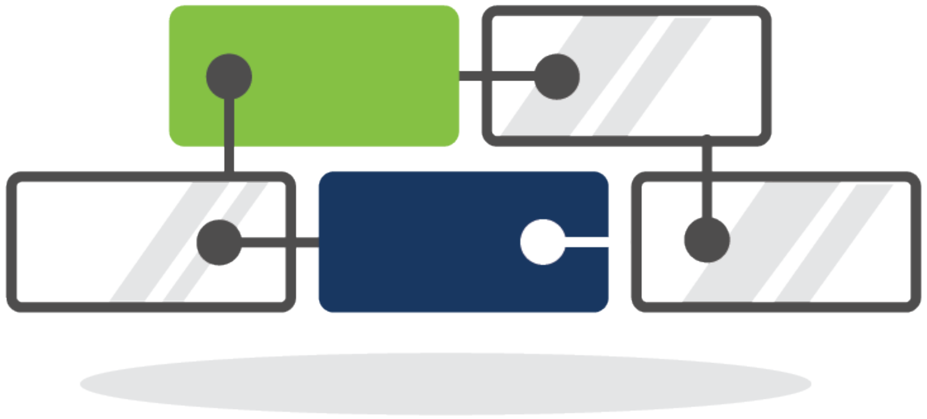
Build the Project

Building a project means compiling and indexing all the source files in order to provide hyperlinks, autocomplete, class 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 using the *DVT: Build…* command from Command Palette or by using the ⚙ button.



Check the Build

- The **Compile Order** and the **Compiled Files** views show all the compiled files

You can open the views using the following commands from the Command Palette:

- ***DVT: Focus on Compile Order View***
- ***DVT: Focus on Compiled Files View***

Both are typically located on the left side of the Editor, in the Primary Side Bar → DVT Activity.

- The **Problems View** shows all the errors and warnings in your project
- It is recommended to walk through the errors in the following order:

Build Config Errors *→ file not found, incdir not found, included file not found ...*

Syntax Errors *→ unexpected token “vitual” instead of “virtual” ...*

Semantic Errors *→ non existing port, wrong number of arguments when calling a function ...*

You can open the Problems View from menu View → Problems.

It is typically located below the Editor, in the Panels area.

Features Overview [1]

- **Hyperlinks:** in the editor, place the cursor over any class names, method names, and in general any identifier. Use *Ctrl + Click* / *Go to Definition* to go to the definition. In addition to this hyperlink, you can find more hyperlinks in the Context Menu or in the Command Palette (eg: type definition, super implementation, show instances, etc.)
- **Show Usages/Readers/Writers:** in the editor, place the cursor over an identifier, next invoke the *Find All References...* / *Show Readers* / *Show Writers* commands to see all places where a variable, signal, function, class, macro etc. is used/read/written.
- **Autocomplete:** in the editor *Ctrl + Space* / *Trigger Suggest* command triggers autocomplete. For example *driver.<Ctrl+Space here>* will show driver API.
- **Quick Fixes:** in the editor, on a line with errors, invoke the *Quick Fix...* command to correct typos, to declare missing variables etc.
- **Rename Refactoring:** place the cursor over an identifier and invoke the *Rename Symbol* command to rename and update all usages across the entire project.

Features Overview [2]

- **Type Hierarchy:** place the cursor over a class name and use *Types: Show Type Hierarchy* command see the OOP inheritance.
- **Design Hierarchy:** place the cursor over the module name and use *DVT: Show Design Hierarchy* command to see the design structure
- **Verification Hierarchy:** place the cursor over an UVM test class and use *DVT: Show Verification Hierarchy* to see the verification environment topology

- **All Classes / Modules / Interfaces / Macros /...:**
  Go to Symbol in Workspace... command / # in the Palette

- **To quickly find a class, module, macro or compiled file:** #<query> in Palette
  You can find here the list of available queries:
  https://eda.amiq.com/documentation/vscode/sv/toc/workspace-symbols/index.html

- **To quickly open a file:** *Go to File...* command / No prefix in Palette

Features Overview [3]

- **Diagrams:** use *DVT: Show Diagram…* command
  - on a module / class / variable to get schematics / UML / state machine diagrams
  - other diagrams available from dedicated contexts: UVM Components / Bitfield for UVM regs & packed data types / Wavedrom
- **Code Formatting:** use *Format Document* or *Format Selection* commands to format the whole editor or selection
- **Toggle Comment:** *Toggle Line Comment* or *Toggle Block Comment* for current line or selection
- **Matching Begin – End:** *DVT: Jump to Matching Pair* / *DVT: Select to Matching Pair* on the *begin*, *end*, *function*, *endfunction* …

- **All Shortcuts:** use *Preferences: Open Keyboard Shortcuts* to see the list of all shortcuts

And many more, please contact support@amiq.com for a demo.

More Information

- Demo Movies:
  https://eda.amiq.com/tutorials
  - *Verification features demo:* https://eda.amiq.com/tutorials/accelerating-hardware-verification-using-dvt-ide-for-visual-studio-code
  - *Design features demo:* https://eda.amiq.com/tutorials/accelerating-hardware-design-using-dvt-ide-for-visual-studio-code
  - *Getting started with DVT in VS Code:* https://eda.amiq.com/tutorials/getting-started-with-dvt-ide-for-visual-studio-code
  - *Integrating DVT with Remote-SSH:* https://eda.amiq.com/tutorials/remote-development-using-dvt-ide-for-vs-code-over-ssh

- Cheatsheet for commonly used keyboard shortcuts:
  https://eda.amiq.com/cheatsheets/DVT_IDE_for_VS_Code_Keyboard_Shortcuts_and_Commands.pdf

- Step by step basic tutorial:
  https://eda.amiq.com/getting-started/My_First_SystemVerilog_Project_with_the_DVT_for_VSCode.pdf
  Please contact us for more training materials

- Features with snapshots:
  https://eda.amiq.com/documentation/vscode-readme-changelog/latest/

- User Guide:
  https://eda.amiq.com/documentation/vscode/sv/index.html

- Datasheet:
  https://eda.amiq.com/datasheets/amiq_dvt_ide_datasheet.pdf

Mail to **support@amiq.com**