Installing and Using GHDL

Richard Susta, Dept. of Control Eng. CTU-FEE in Prague

Version 1.1 from Sept 30, 2024

Document home page: https://dcenet.fel.cvut.cz/edu/fpga/install_en.aspx

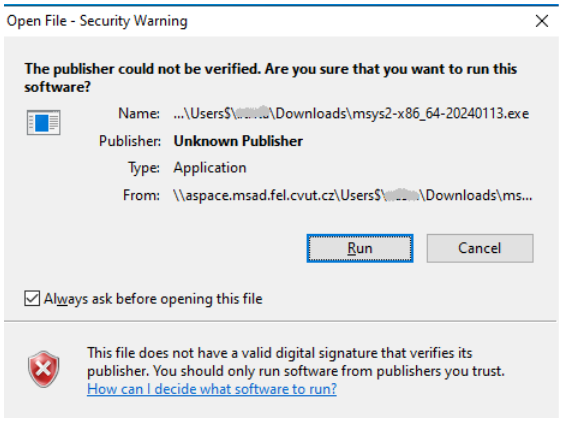
Content

About GHDL

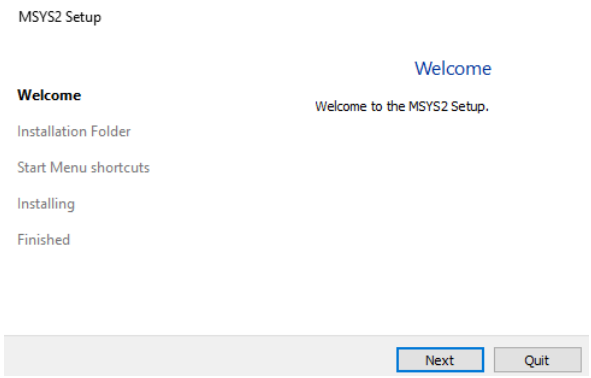
- GHDL compiles the VHDL source code into a Windows exe file. According to our tests, the whole translation-simulation cycle is as fast as in the professional simulators. The latter, however, requires us to know its development environment (quite different) as well as the creation of the project. It specifies the bugs in VHDL more precisely, but one must know how to use it.
- GHDL quickly runs even from the command terminal of Visual Studio Code.
- The entire installation of GHDL with GTKWave requires approximately 2.7 GB, slightly less than ModelSim or Questa, and was created for Linux. It does not use the Windows registry or its system folders and can be cleanly uninstalled.
- GHDL fully supports the 1987, 1993, and 2002 versions of the IEEE 1076 VHDL standards and the main features of its 2008 revision.
- GHDL runs on Linux, Windows, and Apple OS X. It is an open-source project with active contributors, see https://github.com/ghdl/. You can freely download a binary distribution for your OS or try to compile GHDL on your machine.
- Linux installation of GHDL is straightforward but more complicated in Windows because it requires adding MINGW, Minimalist GNU for Windows.
- **Beware**! Native Windows versions also exist, but they are all obsolete. The authors no longer generate such distributions.
- The GHDL project now supports only **MSYS2**, **M**inimal **Sys**tem, i.e., MinGW component (**Min**imalist **G**NU for **W**indows) as the established Software Distribution and Building Platforms for Windows.
- **We must start by installing MSYS2.**

Installing MSYS2

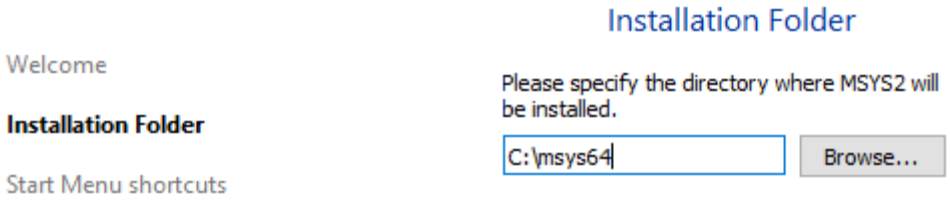
Go to the official project's WEB site and download the actual installer. All images from the document are screenshots of msys2-x86_64-20240113.exe version from Jan 13,2024. Run the installer and confirm the security warning:



Click [Next] on the introductory dialog.

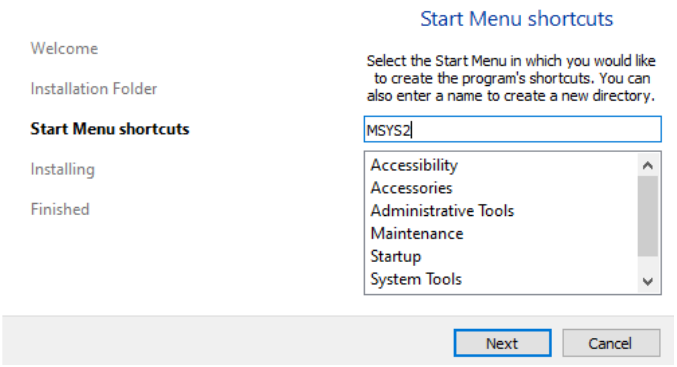


Enter your desired installation Folder. It must have a **short ASCII-only path** to an SSD or HD with **NTFS volume**. The default is **C:\msys64**

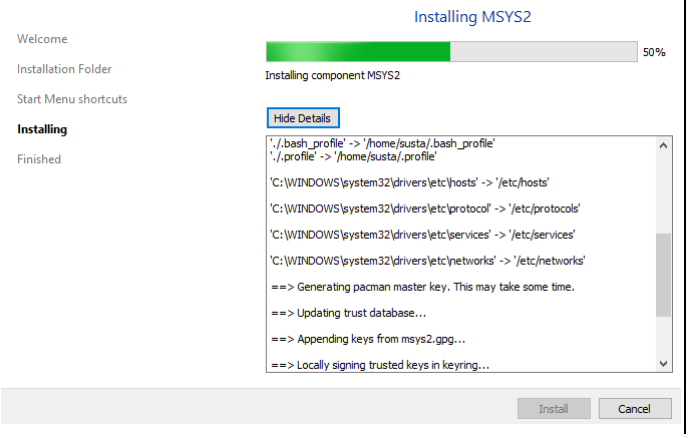


Its path cannot contain accents, diacritic marks, spaces, symlinks pointing to another file or directory, and associated subst to virtual drives. The network drives, and FAT volumes are also not supported.

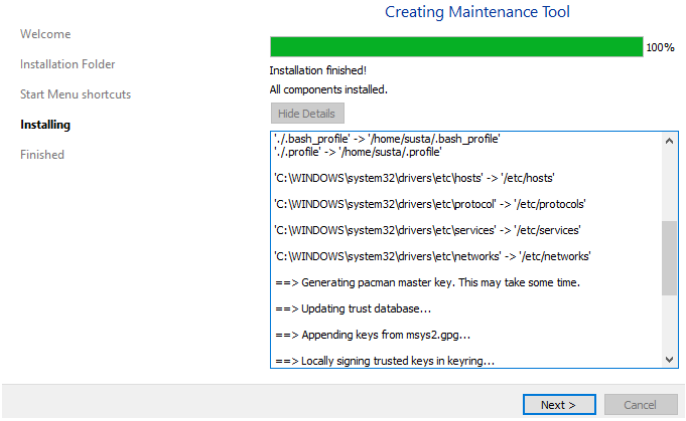
Choose a menu shortcut on the following screen and run the installation by [Next].



The installation will report all its actions



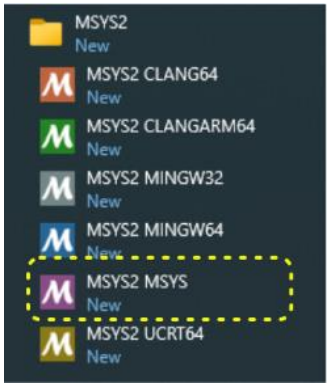
And finally, it finishes.



If the UCRT64 terminal window appears, you can close it. We do not need it.

Installing GHDL

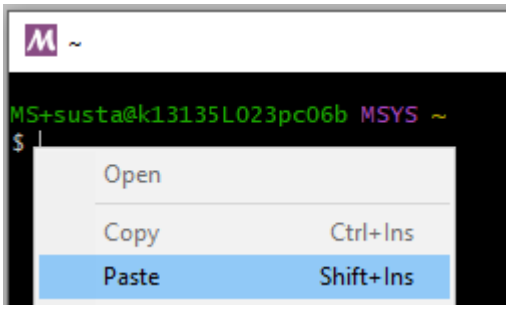
With MSYS2, we can install GHDL. Open a **MSYS2 MSYS** console from the Windows *Start* menu.

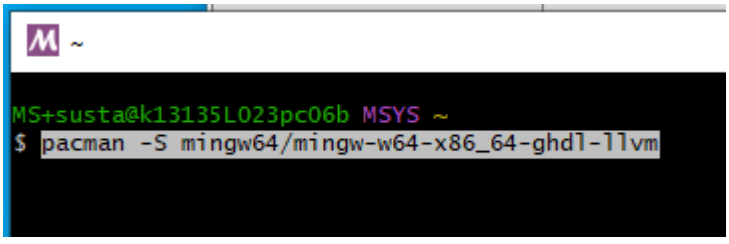


To install the 64-bit version of GHDL, we use the following command :

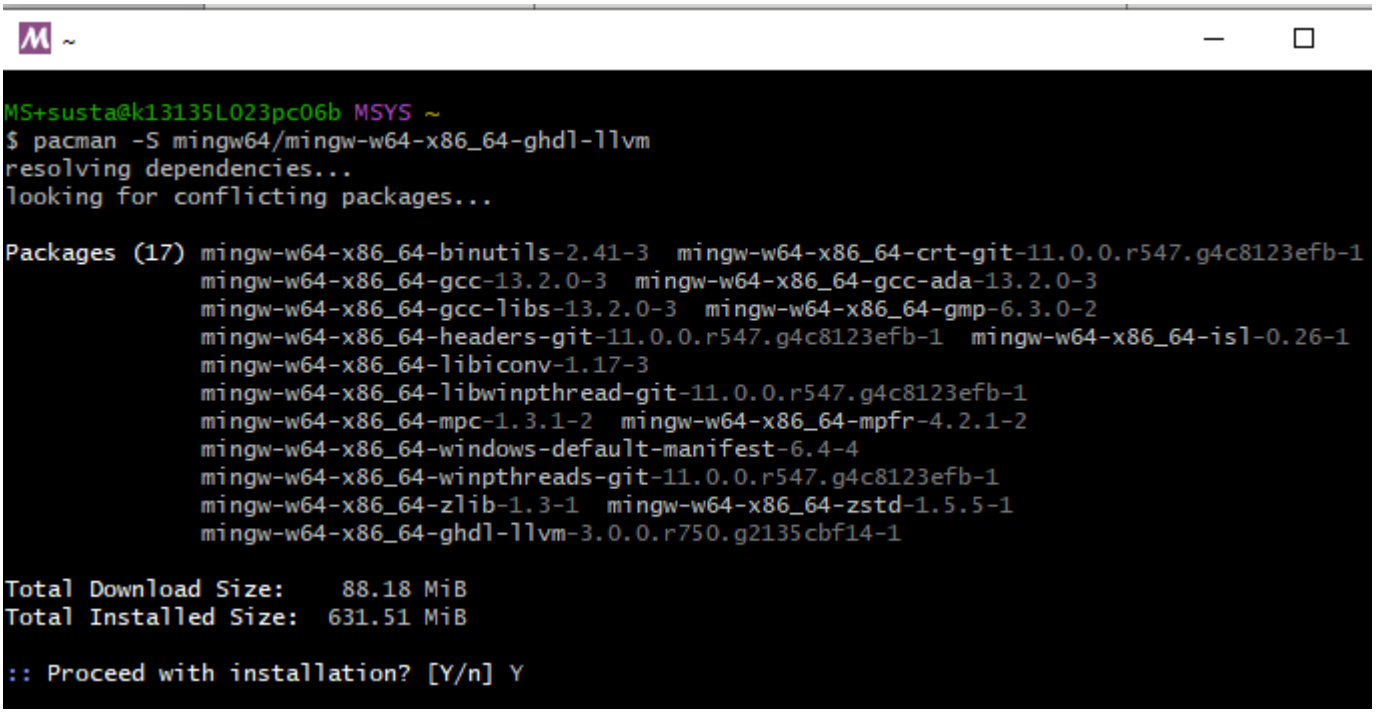
```
pacman -S mingw64/mingw-w64-x86_64-ghdl-llvm
```

Copy it into MSYS2 MSYS command console:

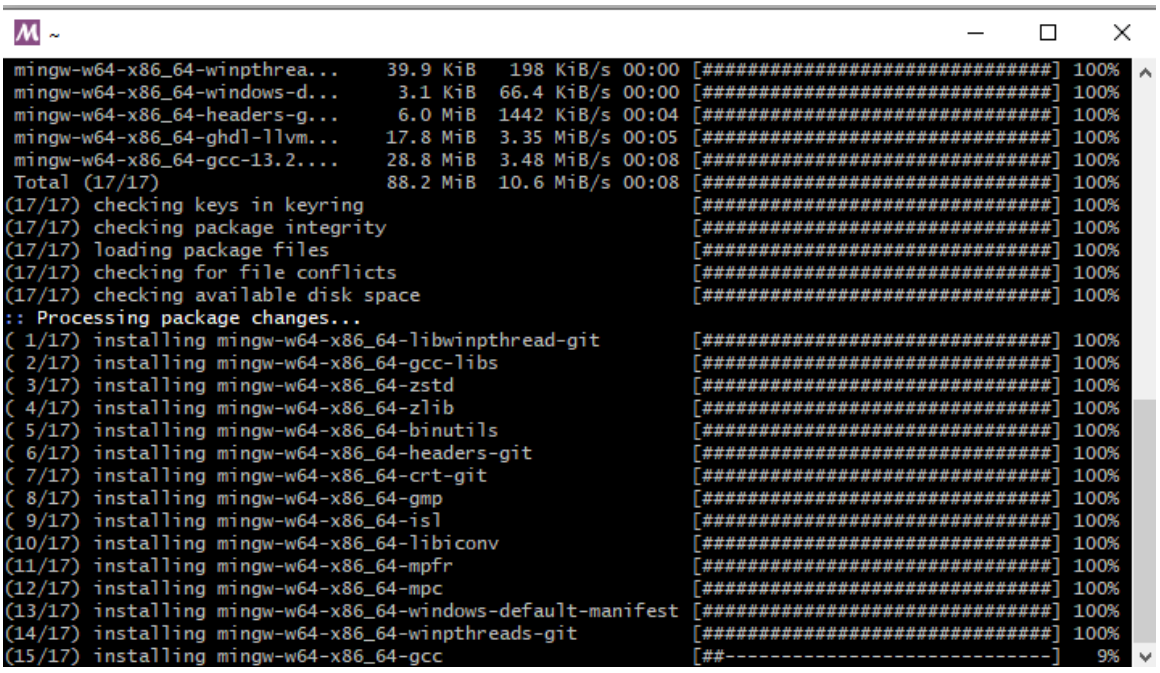




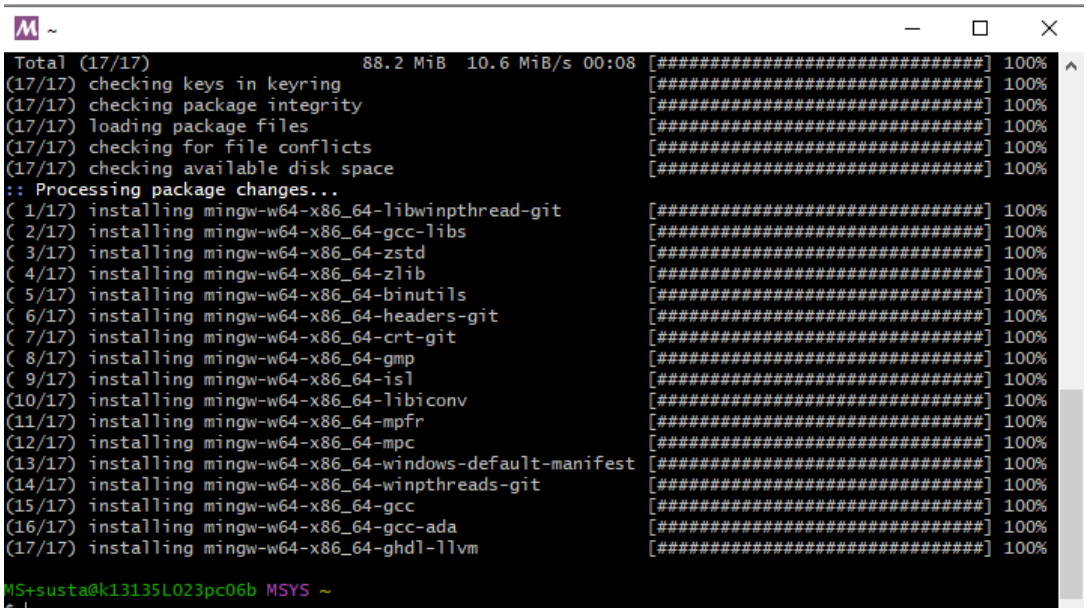
The Pacman (Packet Manager) finds necessary changes. Confirm them by entering Y.



The installation will again report all actions as general Linux program behavior.



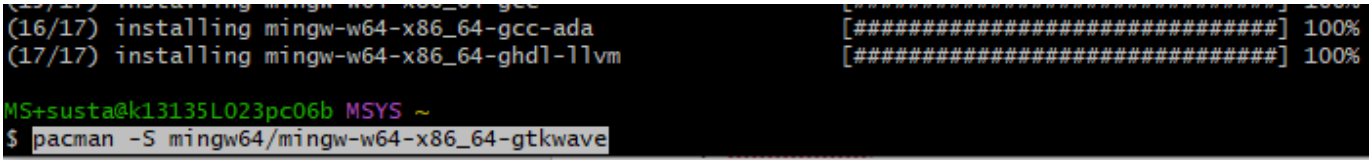
Finally, it reports done.



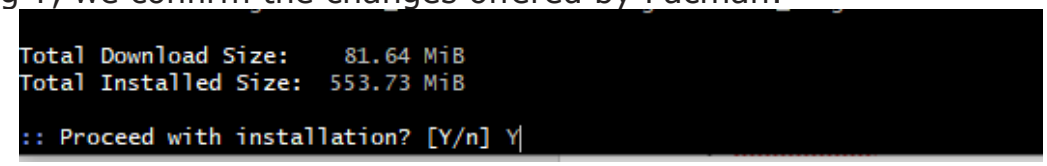
Installing GtkWave Utility

We also need GTKWave to see the signals generated by the GHDL simulation. Copy the following line to the MSYS2 window and start the command:

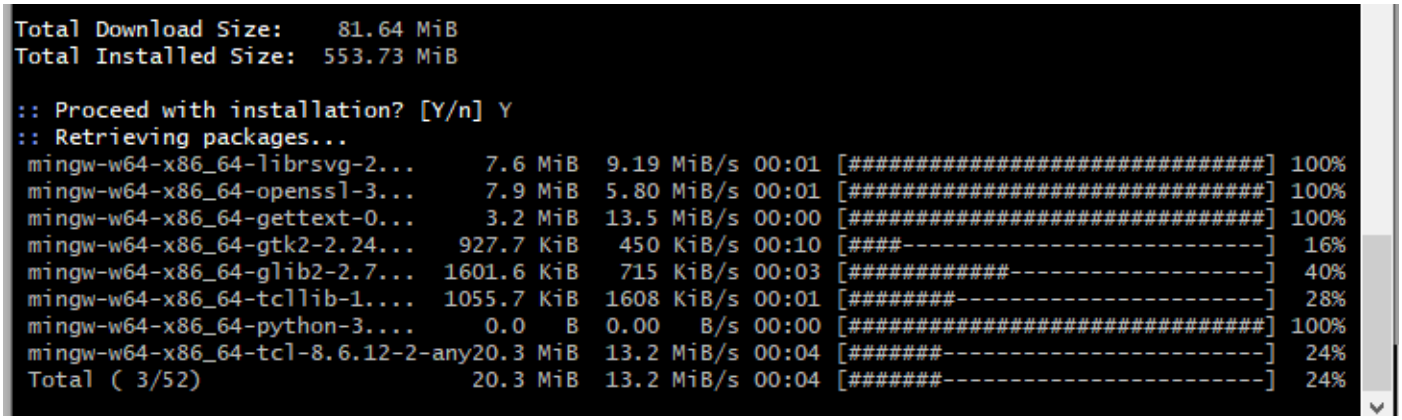
```
pacman -S mingw64/mingw-w64-x86_64-gtkwave
```



By entering Y, we confirm the changes offered by Pacman:



GtkWave installation also reports all its actions:

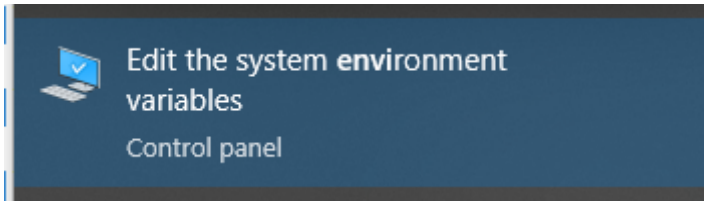


Finally, it terminates. We can close the MSYS command line window.

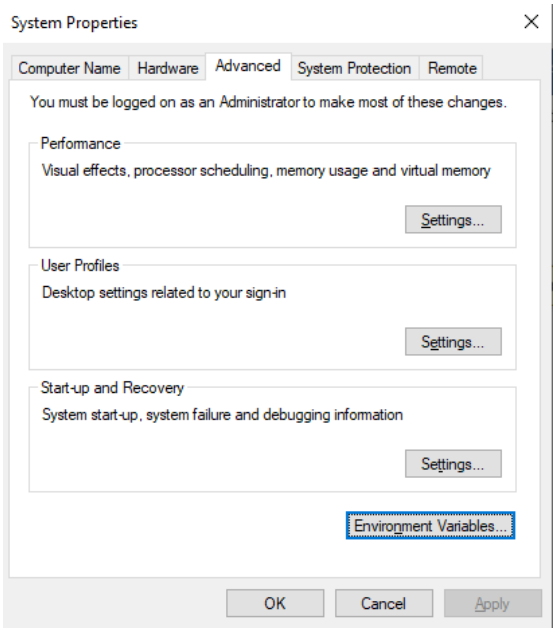
Environment Setup

We add the location of GHDL to **the end of PATH**, which is more robust. In batch command shell files, we can temporarily move it to the front of PATH at any time.

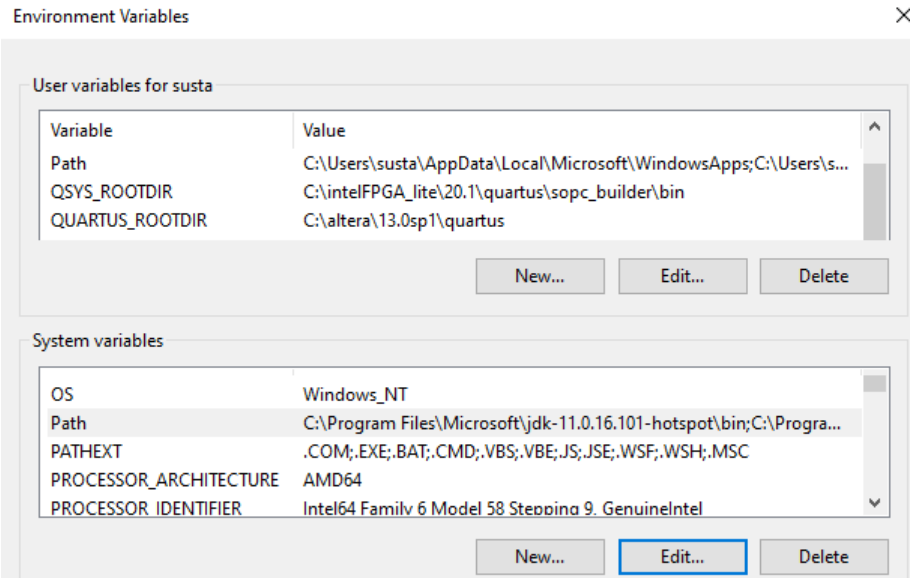
We press the Windows keyboard Key and start writing the word **environment**. After its first three letters, Windows should offer the choice:



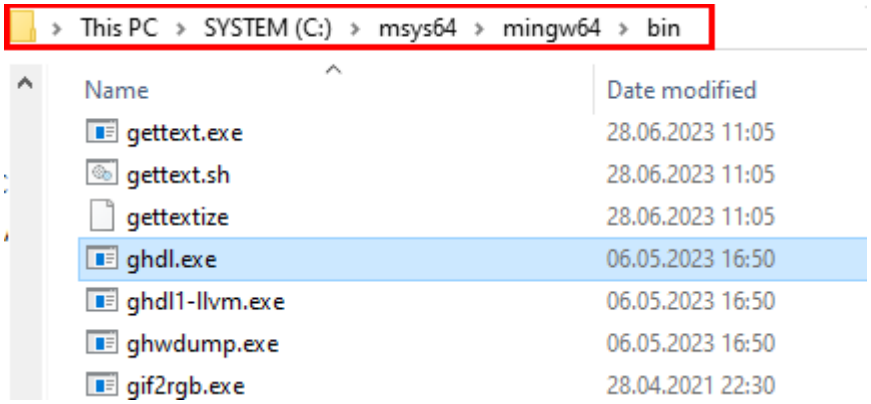
We select it and choose [Environment Variables…] in the System Properties window.



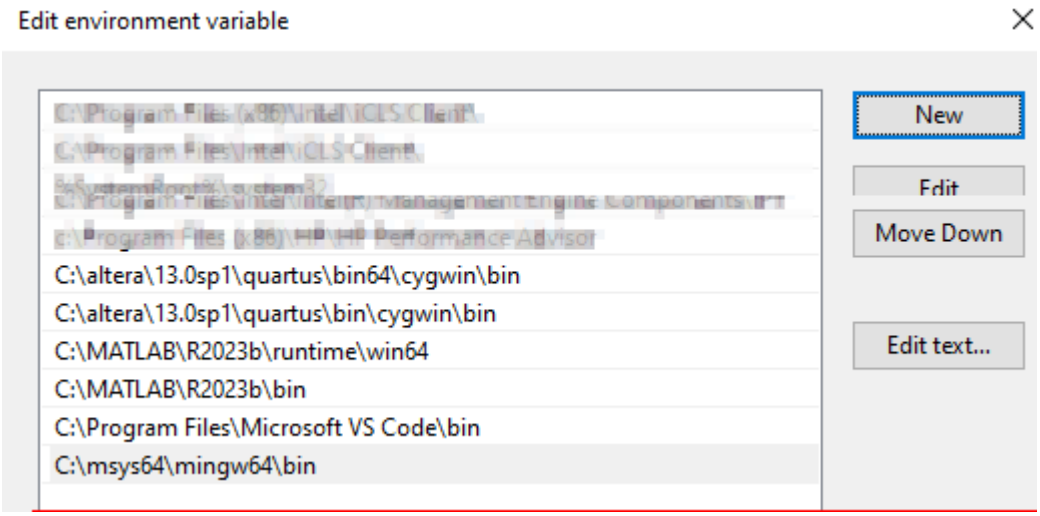
Then, we edit the Path in the System variables.



We found out the location of our ghdl.exe. If we did not change the defaults, it is:



We append it to **the end** of the Path:

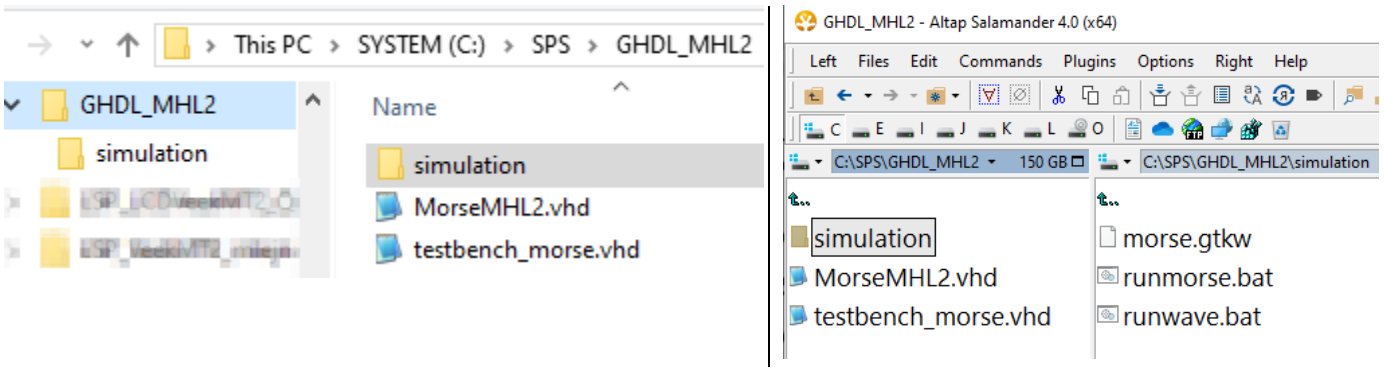


Using GHDL

We also have Quartus installed, which uses MinGW but a different version than GHDL, which is nothing new in the installation hell of the computer world where it is not wise to mix versions. Because of this, we have added mingw64 to the end of PATH!

For the same reason, we cannot simply run *ghdl.exe* stand-alone or utilize a makefile, but we must embed it into the Windows shell script, in which we can temporarily move the path to GHDL utilities to the top.

Suppose we placed all necessary files into **C:\SPS\GHDL_MHL2**



Morse beacon needs only *MorseMHL2.vhd* and *testbench_morse.vhd* for GHDL. The testbench_morse was explained in the lectures.

In the simulation subfolder, we have 2 batch files. The first is **runmorse.bat,** which contains the following Windows shell commands:

```
@ECHO OFF
rem We specify the following sets as temporary
SETLOCAL
rem Testbench filename is without extension
set TBNAME=testbench_morse
rem Files used by the testbench have extensions and relative paths
set FILES=../MorseMHL2.vhd
rem Simulation running time, us=microsecond
set SIMTIME=1us
rem Our name of simulation temporary results
set SIMNAME=MySim

rem Temporary moving mingw64 to top of PATH
set PATH=C:\msys64\mingw64\bin\;%PATH%
rem Compile for VHDL-2008
set GHDL_FLAGS=-fsynopsys --std=08
rem We use %SIMNAME%.vcd as the flag of succesful compilation
if exist %SIMNAME%.vcd del %SIMNAME%.vcd
rem Analyse -a create object files, -e create exe, -r run exe
@ECHO ON

ghdl.exe -a %GHDL_FLAGS% %FILES% ../%TBNAME%.vhd
@IF ERRORLEVEL 1 GOTO BAT-END
ghdl.exe -e %GHDL_FLAGS% %TBNAME%
@IF ERRORLEVEL 1 GOTO BAT-END
ghdl.exe -r %GHDL_FLAGS% %TBNAME% --vcd=%SIMNAME%.vcd --stop-time=%SIMTIME%
:BAT-END
```

Its commands:

`ECHO OFF` – do not print lines, `ON` printing all lines that do not begin by @

`SETLOCAL` - it starts the localization of environment variables that will be valid until the end of the batch file or a matching ENDLOCAL command is reached.

`rem` - the next text is comment till the end of the line.

`set TBNAME` - the entity of testbench, just name, no extension or path

`set FILES=` - file(s) referenced in testbench separated by spaces

`set SIMTIME` - running time, here, 1 microsecond

`set SIMNAME..` – any free name for temporary files

`set PATH` - the temporary move mingw64 path to the top.

`set GHDL_FLAGS` - switching on VHDL 2008 support.

`ghdl.exe -a`

It analyzes VHDL files in the upper directory. **Notice their compilation order**!
The FILES must precede *testbench file*.

`IF ERRORLEVEL 1 GOTO BAT-END` - go to end if the previous command exits on an error.

`ghdl.exe -e` creates the simulation in *testbench_morse.exe* file.

`ghdl.exe -r` runs the executable and creates *morse.vcd*.

Note: If we want to simulate another VHDL file, change set commands.

Running runmorse.bat from Visual Studio Code terminal:

```
C:\SPS\GHDL_MHL2\simulation> .\runmorse.bat
C:\SPS\GHDL_MHL2\simulation>ghdl.exe -a -fsynopsys --std=08 ../MorseMHL2.vhd ../testbench_morse.vhd
C:\SPS\GHDL_MHL2\simulation>ghdl.exe -e -fsynopsys --std=08 testbench_morse
C:\SPS\GHDL_MHL2\simulation>ghdl.exe -r -fsynopsys --std=08 testbench_morse --vcd=MySim.vcd --wave=MySim.ghw --stop-time=1us
.\testbench_morse.exe:info: simulation stopped by --stop-time @1us
PS C:\SPS\GHDL_MHL2\simulation>
```

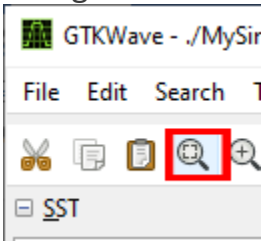
After successful compilation, we run runwave.bat

```
C:\SPS\GHDL_MHL2\simulation> .\runwave.bat
```

It contains the following commands. Note: SIMNAM must agree with runmorse

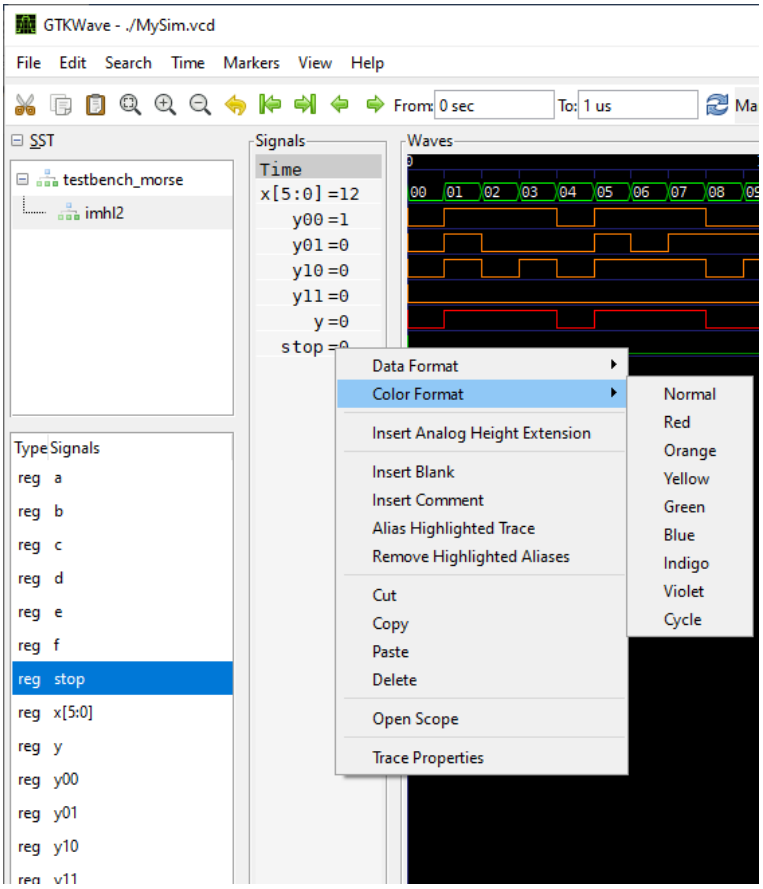
```
@ECHO OFF
rem We specify the following sets as temporary
SETLOCAL
set SIMNAME=MySim
rem Temporary moving mingw64 to top
set PATH=C:\msys64\mingw64\bin\;%PATH%
IF exist ./%SIMNAME%.vcd (
   rem Run gtkwave and continue batch commands
   start gtkwave ./%SIMNAME%.vcd
 ) ELSE (
echo There's nothing to view. First, the compilation must exit successfully.
)
```

The shell command "`start`" opens a separate Command Prompt window, in which it runs `gtkwave.exe`. The *runwave.bat* shell continues and is closed.

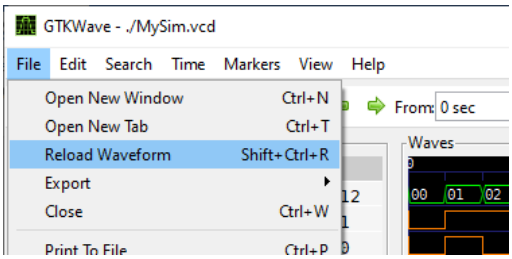


In GtkWave, we first press Zoom Fit to see the simulated time interval and adjust the zoom with the mouse wheel.

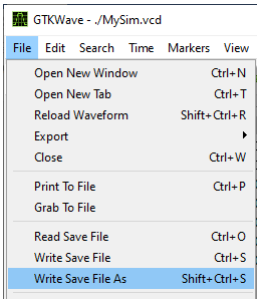
By double-clicking on the signal list, we first add desired waves, and by right-clicking on them, we can change their colors or listed data formats.

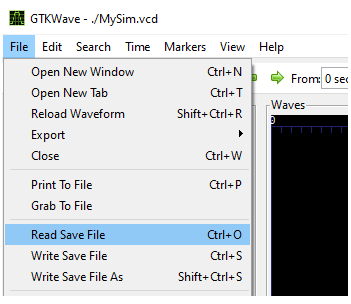


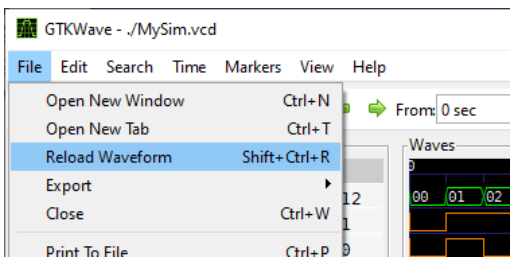
The gtkwave program loads **.vcd** (.vcd=Value Change Dump) result of ghdl.exe.
If we create a new *.vcd by subsequent runmorse.bat, we only reload *.vcd from GTKWave: **File->Reload Waveform**



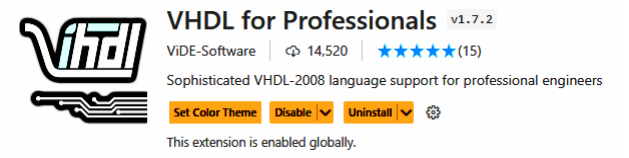
We can reload next time if we store the simulation to keep signals and their settings. The command "Read Save File" loads the saved signals and their waves, and then we reload them to refresh data. The colors and formats remain.



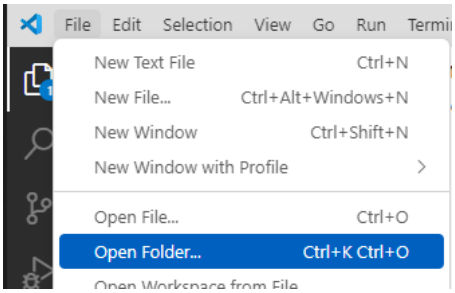




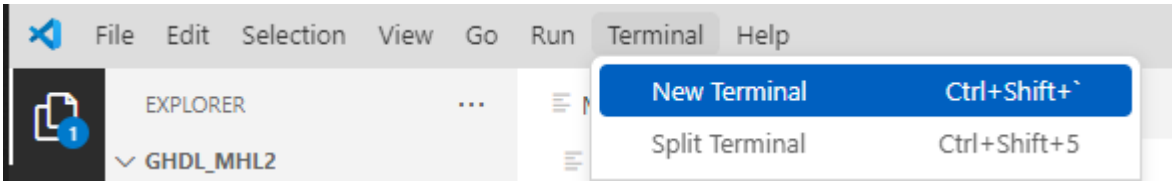
The batch scripts can be also executed from Visual Studio Code terminal, which has a lot of extensions for VHDL coding. The following pictures were created with the aid of the 2024-02-03 release of the extension:



First, from Visual Studio Code, open the folder with simulated files. In the MHL2 example case, it is: C:\SPS\GHDL_MHL2 folder.



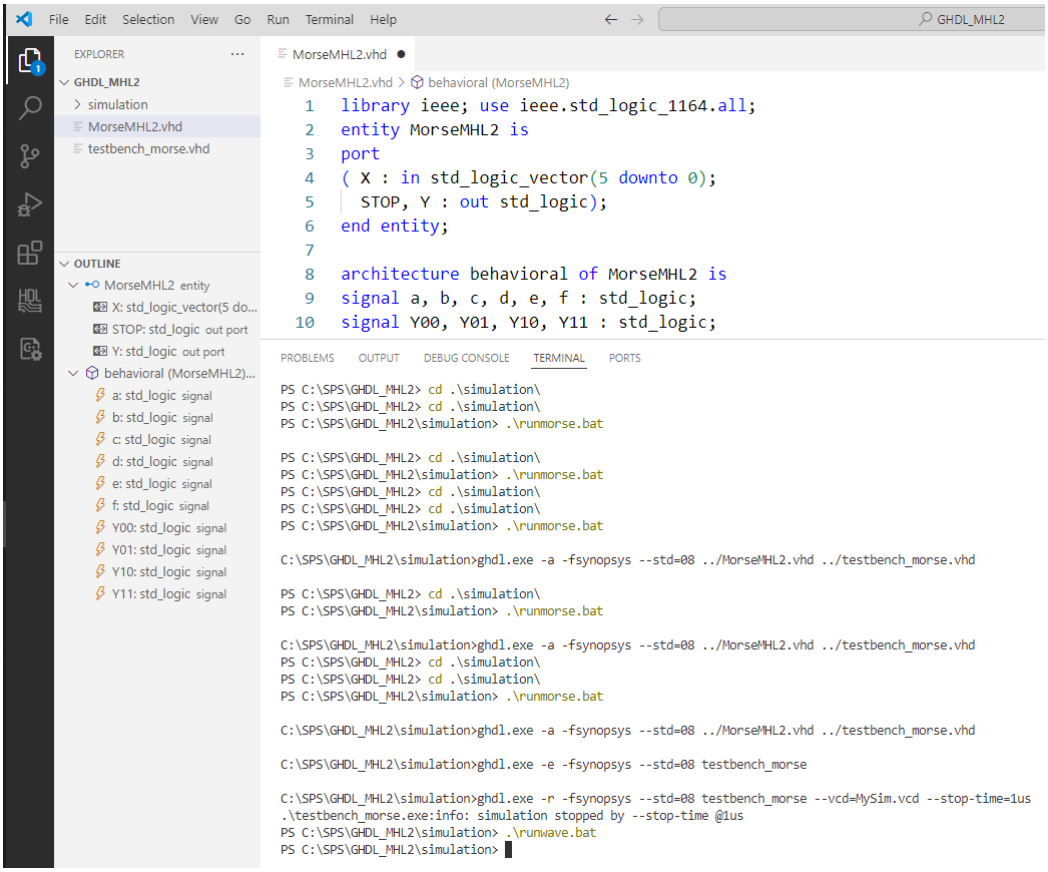
After opening the folder, create a new command line terminal:



The command window starts in the folder C:\SPS\GHDL_MHL2.

First, we must change its default path to the **simulation** subdirectory, where we intentionally placed *runmorse.bat* and *runwave.bat*. Hence, the intermediate GHDL compilation files also remain here and do not mess up the main folder.

Then, we execute the batch files:



Notes:

- The command line windows accept '**/**' or '****' characters as folder separators.
- By default, Windows PowerShell does not load files from the current location in all cases. It keeps Linux customs.
  We must add `./` or `.\` in front of all filenames in our current directory.
- **Pressing the [TAB]** autocompletes the command. If more than one option is available, we can hit [TAB] again to display the following possible choices or continue typing until only one matching choice is left.

  In the example above, we wrote down only:

cd ./	*after TAB, it was autocompleted to*	**cd .\simulation**
./r	*pressing TAB autocompleted to*	**.\runmorse.bat**
./r	*two TABs autocompleted to*	**.\runwave.bat**

Professional simulators such as ModelSim or Questa offer more possibilities, but their environments differ from programming tools, so they need effort to learn and more steps to run simulations.
GHDL simulates as fast as professional simulators and is much easier to run. Using GHDL, we can debug more advanced tasks, such as a circuit that will economically create an LCD control panel background image, which will be the following credit project in our Logic Systems and Processors course.

[The image source Corel 9 ClipArts]

~~~ The End ~~~
~~~