Install and use GHDL and GtkWave

Table of Contents

GHDL

GHDL is an open-source simulator for the VHDL hardware language. GHDL is not an interpreter; it allows you to analyze and elaborate sources to generate machine code from your design. Native program execution is the only way for high speed simulation.

Features as given on the official GHDL Github page.

- GHDL fully supports the 1987, 1993, 2002 versions of the IEEE 1076 VHDL standard, and the latest 2008 revision
- Partial support of PSL.
- By using a code generator (LLVM, GCC or, x86_64/i386 only, a built-in one), it is much faster than any interpreted simulator. It can handle very large designs, such as leon3/grlib.
- GHDL runs on GNU/Linux, Windows and macOS, both on `x86` and on `x86_64`.
- GHDL can write waveforms to a GHW, VCD or FST file. Combined with a GUI-based waveform viewer and a good text editor, GHDL is a very powerful tool for writing, testing and simulating code.
- Supported third party projects: VUnit, OSVVM, cocotb (through the VPI interface), ...

GtkWave

GTKWave is a fully featured GTK+ based waveform viewer for Unix, Win32, and Mac OSX which reads LXT, LXT2, VZT, FST, and GHW files as well as standard Verilog VCD/EVCD files and allows their viewing.

Get the necessary programs and tools

This is a guide to install both tools on Linux-Mint_20.4 or Ubuntu_20.04 and possibly coming releases of Linux-Mint and Ubuntu. Download GHDL and GtkWave and download tools and programs necessary to install and run both tools.

GHDL:

- This guide describes the GHDL - LLVM installation.
- Download the GHDL source files from its GitHub location.
  Click the [code] button and download the ZIP file.
  This file is called **ghdl-master.zip** and should be saved in *~/Downloads/Ghdl*.
- Uncompress the zip file and let it store itself in the directory set in the zip file.
- Other tools necessary to install GHDL-LLVM are:
  - GNAT - ADA compiler: The latest community version at the time of writing/rewriting was/is **gnat-2020-20200429-x86_64-linux-bin**. Get it here.
  - LLVM - Low Level Virtual Machine.

GtkWave

- Download GtkWave from here(1) or from here(2)
  At time of writing/rewriting this article the latest version Linux Ubuntu it was "**gtkwave-3.3.105.tar.gz**".
- (1) Just click the big [Download] button and the latest version for the used operating system is selected and downloaded.
- (2) In the first line of text "*... You can grab version 3.3.105 here. Documentation in pdf format can be found here.*" click the blue `here` button to download the latest .tar.gz file and the latest version of the GtkWave documentation.
- Save the downloaded file under */Downloads/Ghdl* as you did for other files related to GHDL.
- Uncompress the .tar.gz file and let it occupy it's own directory under */Downloads/Ghdl*

Install the tools

Install GNAT

- Assumed is that the earlier downloaded files are stored in */Downloads/Ghdl*
- Make the downloaded file executable:

  ```
  cd \Downloads\Ghdl
  chmod 766 gnat-2020-20200429-x86_64-linux-bin
  ```

  Run the file and install the ADA compiler under */opt/Gnat/*
  Be aware that installing software in /opt requires one to be sudo or root.
  `sudo ./gnat-2020-20200429-x86_64-linux-bin`

- Add following line to your *.bashrc* or better *.bash_aliases* file.

  ```
  # GNAT GPL (ADA compiler) for GHDL
  export PATH=$PATH:/opt/Gnat/<year>/bin
  ```

- The tool is installed and usable in command line mode. It is possible to install a GNAT studio.
  To do this, go to were the GNAT tool is installed, in this case */opt/Gnat/2020* and run the *doinstall* script.

```
cd /opt/Gnat/2020
sudo ./doinstall
```

- Follow the instructions of the installer.
- GNAT Studio will automatically add itself to the PATH.
  Note: that GNAT Studio path will be add at the end of the PATH, meaning that it will find first any other GNAT installations that you have in your PATH.
  Hint: Remove the automatically add entry for GNAT-studio from the path and add it manually to the path, but do it at the same place where the other GNAT options have been add to the *.bashrc* or better *.bash_aliases* file.

Install LLVM

- Browse to this page: https://apt.llvm.org/
- On the page go to the Ubuntu section and copy the two text lines for the latest stable released version and for the latest version of Ubuntu *Focal (20.04)*
  - This are the lines:

```
deb http://apt.llvm.org/focal/ llvm-toolchain-focal main
deb-src http://apt.llvm.org/focal/ llvm-toolchain-focal main
```

  Adding above two lines to the repositories of your system will search for what's called *(old stable branch)*, during time of writing this was version 9 of LLVM. If you want to install from repositories a newer version, like the *(stable branch)*, or version 10 during time of writing, the you need to add the version number to the repository lines. Like this:

```
deb http://apt.llvm.org/focal/ llvm-toolchain-focal-10 main
deb-src http://apt.llvm.org/focal/ llvm-toolchain-focal-10 main
```

- Press the windows key and start typing **Software & Updates**. As soon as some letters are typed, icons will appear. Click the [Software & Update] icon and select the [Other Software] tab. Click [Add] and enter the first “deb …” line in the new popup. Click [Add Source] and provide the sudo(root) password.
  Add the second ”deb …“ line the same way.
- Hit [Close] and allow a new scan of the repositories.
- Install the stable version of all key packages
  - Lower on the page with repository archive information find how to install all key packages or just read on here.
  - Open a terminal (Right click the desktop and select [Open Terminal]).
  - Type or better copy, one by one, following lines in the terminal:
    - `wget -O - https://apt.llvm.org/llvm-snapshot.gpg.key|sudo apt-key add -`

      provide the sudo password to install the archive signature.

Remark:
From here on the LLVM tools will be installed.
I've set the repository lines for the *(old stable release)* in my system. Reason release shift trough and now I'm sure I always get an update installed that is just one behind the newest stable released version.

To install the tools that go together with the added repository lines, mention the correct version in the install commands. In this case it's version 9.

- Install LLVM

  `apt-get install libllvm-9-ocaml-dev libllvm9 llvm-9 llvm-9-dev llvm-9-doc llvm-9-examples llvm-9-runtime`

- Install Clang & co

  `apt-get install clang-9 clang-tools-9 clang-9-doc libclang-common-9-dev libclang-9-dev libclang1-9 clang-format-9 python-clang-9`

- Libfuzzer

  `apt-get install libfuzzer-9-dev`

- lldb

  `apt-get install lldb-9`

- lld (linker)

  `apt-get install lld-9`

- libc++

  `apt-get install libc++-9-dev libc++abi-9-dev`

- OpenMP

  `apt-get install libomp-9-dev`

- To make use of this latest version of LLVM tools some settings need to be done. In the terminal continue typing or copying and executing following lines one after the other.

```
sudo update-alternatives --install /usr/bin/clang clang /usr/bin/clang-9
100
sudo update-alternatives --install /usr/bin/clang++ clang++
/usr/bin/clang++-9 100
sudo update-alternatives --install /usr/bin/clang-apply-replacements
clang-apply-replacements /usr/bin/clang-apply-replacements-9 100
sudo update-alternatives --install /usr/bin/clang-check clang-check
/usr/bin/clang-check-9 100
sudo update-alternatives --install /usr/bin/clang-query clang-query
/usr/bin/clang-query-9 100
sudo update-alternatives --install /usr/bin/clang-tidy clang-tidy
/usr/bin/clang-tidy-9 100
sudo update-alternatives --install /usr/bin/scan-build scan-build
/usr/bin/scan-build-9 100
sudo update-alternatives --install /usr/bin/scan-view scan-view
/usr/bin/scan-view-9 100
```

The above lines set a permanent link for the system to use the newly installed LLVM tools instead of possible pre-installed or earlier installed versions.

Install GHDL

In the assumption that GNU-ADA and LLVM are installed and the downloaded *ghdl-master.zip* file is uncompressed in */Downloads/Ghdl/ghdl-master* , do following to install GHDL.

- Open a terminal window and change directory (cd) to the */opt* folder.
  Create here a new directory called /Ghdl `sudo mkdir Ghdl`.
- Change the owner and group of the directory;
  `sudo chown <user> Ghdl`
  `sudo chgrp <user> Ghdl`
  = your user name.
  This is done to make later modifications and additions easier.
- Change directory to the folder used by the uncompressed GHDL files, `cd /home/Downloads/Ghdl`.
- Run the terminal command:

```
./configure --with-llvm-config=/usr/lib/llvm-9/bin/llvm-config --
prefix=/opt/Ghdl
```

- When the previous run is done, type `make` and let run again.
- Run `make install` when previous run has finished.
- The tool should now be installed in */opt/Ghdl*, showing three folders (bin, include, lib).
- Add the path */opt/Ghdl/bin* to the systems path by editing the *.bashrc* or *.bash_aliases* file.

```
# GHDL simulator
export GHDL_ROOT="/opt/Ghdl"
export PATH=$PATH:$GHDL_ROOT/bin
```

- Change directory to */opt/Ghdl/bin* and create a symbolic link for ghdl.

```
sudo ln -s  /opt/Ghdl/bin/ghdl /usr/bin/ghdl
```

Install GHDL vendor libraries

The GHDL simulator tool is installed on the system but only usable for generic VHDL simulations. The guess is that GHDL has been installed for simulations using a specific FPGA vendor. GHDL does not contain contains libraries for FPGA or other device vendors but contains scripts allowing to easily compile and install vendor supplied libraries. There are ready made scripts available for Altera/Intel, Lattice, Xilinx, OSVVM and UVVM. These scripts can be found in */opt/Ghdl/lib/ghdl/vendors*.

Xilinx

Lets use the script to compile and install libraries for Xilinx components using the Vivado tools.

Remark: Be sure to have the Xilinx Vivado tools installed on your system! If not, install these tools first.

- As first we need to tell, in a script (sh file) where the Xilinx tools are installed and were the vendor library sources are stored and were we ant the compiled libraries to be written.
  To do that go to the `/opt/Ghdl/lib/ghdl/vendors` directory.
- Open with a text editor the `config.sh` file and modify following lines:

    - In the file section *declare -A InstallationDirectories* modify the line `InstallationDirectories[XilinxVivado]=""` by adding the main install path of the Xilinx Vivado tools. On my system the line looks like: `InstallationDirectories[XilinxVivado]="/opt/Xilinx/Vivado/2020.1"`
    - In the file section *declare -A DestinationDirectories* modify the line `DestinationDirectories[XilinxVivado]="xilinx-vivado"` to point to the directory were the compiled libraries must fit. In my case the line looks as: `DestinationDirectories[XilinxVivado]="/opt/Ghdl/lib/ghdl/Xilinx93"`
    - As last modification, in the file section *declare -A SourceDirectories* modify the line `SourceDirectories[XilinxVivado]="data/vhdl/src"`. In fact this line doesn't need any modification because the Xilinx VHDL source libraries are stored at Xilinx tool install in the given directory.
- If you did not start a terminal for previous actions, launch a terminal window and now and chnage directory to: `cd /opt/Ghdl/lib/ghdl/vendors`.
- Run in the terminal the `compile-xilinx-vivado.sh` script with the required options.
  in this case the libraries need to be compiled for VHDL-93, and all Xilinx libraries must be generated.

  The command must be run as: `` `./compile-xilinx-vivado.sh --vhdl93 --all ``
- When other libraries must be compiled for Xilinx or for other vendors or VHDL library suppliers consult the *readme.md* file in the vendors directory.

Remark: The above described process fails when the target directory already exist!
In other words: when *DestinationDirectories[XilinxVivado]* gets a path to a already existing directory (with or without content) the library compile process mentions at the end of it's run *"Compiling Xilinx Vivado libraries [FAILED]"*

UVVM

A second example of VHDL simulation library is the Universal VHDL Verification Methodology (UVVM) library that can be downloaded here . UVVM Utility Library is tool and library independent, but it must be compiled with VHDL 2008.

- Download the library by hitting the [code] button and downloading the *UVVM-master.zip* file.
- Copy the *UVVM-master.zip* into the /opt/Ghdl/lib/ghdl/vendors directory.
- Uncompress the file and let it store everything in the directory available in the zip file (UVVM-master).
- Go to the */opt/Ghdl/lib/ghdl/vendors directory and open with a text editor the *config.sh* file and change following:
    - in the file section *declare -A InstallationDirectories* modify the line `InstallationDirectories[UVVM]=""` by adding the main install path of the Xilinx Vivado tools. On my system the line looks like: `InstallationDirectories[UVVM]="/opt/Ghdl/bin/ghdl/UVVM-master"` This is the directory of the unzipped UVVM-master.zip file from GitHub.
    - In the file section *declare -A DestinationDirectories* modify the line `DestinationDirectories[UVVM]="xilinx-vivado"` to point to the directory were the compiled libraries must fit. In my case the line looks as: `DestinationDirectories[UVVM]="/opt/Ghdl/lib/ghdl/UVVM"`
    - As last modification, in the file section *declare -A SourceDirectories* modify the line `SourceDirectories[XilinxVivado]="."`. In fact this line doesn't need any

modification because the Xilinx VHDL source libraries are stored at Xilinx tool install in the given directory.

- If you did not start a terminal for previous actions, launch a terminal window and now and chnage directory to: `cd /opt/Ghdl/lib/ghdl/vendors`.
- Run in the terminal the `compile-xilinx-vivado.sh` script with the required options.
  in this case the libraries need to be compiled for VHDL-93, and all Xilinx libraries must be generated.

  The command must be run as: `` `./compile-xilinx-uvvm.sh --all ``
- Find, when the command finishes successful under the UVVM directory a set of directories containing simulatable code.

Remark:

- Some restrictions and remarks:
  - The same issue applies here as when compiling the Xilinx libraries. Compilation fails at the end of the compilation when the destination directory already exists.
  - In the UVVM documentation, in step 2 of the "For developers who understand ....." paragraph it is mentioned that a 'compile_order.txt' file must be used. Don't know why, but I compiled the libraries without this file. In fact this file does not even exist in the vendors directory of GHDL.
  - Instead of the 'compile_order.txt' file, the things to compile or compile order is given in the *compile-uvvm.sh* script. The script by default contains a set of UVVM modules. In mean time the UVVm library has been expanded with new components. Thus what we need to do is:
    - Open the *compile-uvvm.sh* script with a text editor.
      The near top of the file shows something like this:

      ```
      uvvm_pkgs="uvvm_util
       uvvm_vvc_framework"

      uvvm_vips="
       bitvis_vip_scoreboard
       bitvis_vip_sbi
       bitvis_vip_avalon_mm
       bitvis_vip_axilite
       bitvis_vip_axistream
       bitvis_vip_gpio
       bitvis_vip_i2c
       bitvis_vip_spi
       bitvis_vip_uart
       bitvis_vip_wishbone
       bitvis_vip_clock_generator
      "
      ```

    - Open the UVVM-master directory and compare the contents with the list in the script (above text). Add the missing parts.
      - Add to uvvm_pkgs:

      ```
      xConstrRandFuncCov
      ```

- Add to uvvm_vips

```
bitvis_irqc
bitvis_vip_spec_cov
bitvis_vip_avalon_st
bitvis_vip_error_injection
bitvis_vip_gmii
bitvis_vip_rgmii
```

- DO NOT ADD:

```
bitvis_vip_hvvc_to_vvc_bridge
bitvis_vip_ethernet
```

If add the compilation will fail due to an error in the files of the *bitvis_vip_hvvc_to_vvc_bridge* model.

- Save the file and compile the library as discussed above.

GtkWave

This is a similar install procedure as for GHDL, do following:

- Open a terminal window and change directory (cd) to the */opt* folder.
  Create a new directory called */GtkWave* (*sudo mkdir GtkWave*).
- Change the owner and group of the directory
  `sudo chown <user> GtkWave`
  `sudo chgrp <user> GtkWave`

  = your user name.
  This is done to make later modifications and additions easier.
- Change directory to the folder used for the uncompressed files;
  `cd /home/Downloads/Ghdl/GtkWave<version>`.
- Run the terminal command: `./configure --prefix=/opt/GtkWave`
  - The above command should do the job when the configure script can find the Tcl/TK tools installed on the machine. The tool must be able to find following files: tclConfig.sh and tkConfig.sh. When the configure script cannot find one of the files check if tcl and tk are installed. When not install both and try again. When tcl and tk are installed run the script with following options:
  - *./configure --prefix=/opt/GtkWave --with-tcl=/opt/ActiveTcl/lib/tcl8.6 --with-tk=/opt/ActiveTcl/lib/tk8.6* (replace the path /opt/ActiveTcl/lib/tcl8.6 (tk8.) by the appropriate path on your system).
  - Another thing where the configuration script can trip over are the xz-utils (zip utilities). To avoid problems run the command:
    ./configure --prefix=/opt/GtkWave --with-tcl=/opt/ActiveTcl/lib/tcl8.6 --with-tk=/opt/ActiveTcl/lib/tk8.6 --disable-xz
- When the previous run is done, type `make` and let it run again
- Type *su* and provide the root or sudo password then type `make install` and let run again until finished.
- Check the install by running: `make installcheck`.
- The tool should now be installed in */opt/GtkWave*, showing three folders (bin, lib, share).

- Add the path */opt/GtkWave/bin* to the systems path by editing the *.bashrc* file.

```
# GtkWave viewer
export GTKWAVE_ROOT="/opt/GtkWave"
export PATH=$PATH:$GTKWAVE_ROOT/bin
```

- Change directory to */opt/GtkWave/bin* and create a symbolic link for ghdl.
    - sudo ln -s /opt/GtkWave/bin/gtkwave /usr/bin/gtkwave

The end

Both tools are installed.
Under Linux/Ubuntu opening a terminal and typing:

- ghdl
  Shows **ghdl:error: missing command, try ghdl --help**
- gtkwave
    Shows an empty black pop-up screen normally used for waveforms.

The tools are installed and functional.

Tools Documentation

GHDL

- https://ghdl.readthedocs.io/en/latest/index.html

GtkWave

- http://gtkwave.sourceforge.net/gtkwave.pdf
- VCD writer: http://pyvcd.readthedocs.io/en/latest/index.html