Setting up Your Environment for RTL Development and Verification

Introduction

This document describes the Environment Setup for RTL Development and Verification using CocoTB. There are separate instructions for both Windows/Linux and Mac OS systems. If you are just getting into RTL development and have some experience using Linux or system commands this guide is a perfect jumping off point to start writing Verilog code and testing it yourself!

Remember! You do not have to do everything at once. It may help you keep track of things, but feel free to take a break at any point. Some of these commands will result in installations that can take quite a while so there is no need to watch each one all the way through.

Requirements:

- A computer with a Linux terminal (Window Powershell, Mac Terminal, etc.)
- Internet connection
- This assumes that the user is using a bash profile for their terminal.

Steps for Windows/Linux

Step 1: Prepare Your Materials

1. Open up your Linux terminal (Windows Powershell, etc.)
2. Congratulations! Your materials are now prepared

Step 2: Install Prebuilt RISC-V Toolchains

1. Make a directory to store the RISC-V GCC toolchain in
    - type **sudo mkdir /opt/riscv**
      ** Note: Sometimes sudo is necessary and sometimes sudo is not: this just depends on your permissions. If a command is not working without sudo, consider trying it with sudo, and vice versa!
2. Enter the directory you just created

    - type **cd /opt/riscv**
3. Copy file from server to local
    - type **wget https://github.com/stnolting/riscv-gcc-prebuilt/releases/download/rv32i-4.0.0/riscv32-unknown-elf.gcc-12.1.0.tar.gz**
      ** Note: this might take a while, but have patience!!
4. Unzip the downloaded folder
    - type **sudo tar -xzf riscv32-unknown-elf.gcc-12.1.0.tar.gz -C /opt/riscv/**
      ** Note: you may get a pop up here asking to allow changes to your computer. Don't worry, this is all safe stuff :)
5. Setup the link
    - If you are on a Linux system, type **export PATH=$PATH:/opt/riscv/bin**
    - If you are on PowerShell, type **$env:PATH += ";C:\opt\riscv\bin"**
6. Store the link to configure the file
    - use the text editor of your choice to open ~/.profile
        - ex: type **sudo vim ~/.profile**
    - Add "export PATH=$PATH:/opt/riscv/bin" to the end of file, and store
7. Verify that the installation was successful
    - type **riscv32-unknown-elf-gcc -v**
        - if you get something other than "command not found", you should be good!

Step 3: Install all Necessary Tools

1. Make sure your system's package information is up to date
    - type **sudo apt update**
2. Install a ton of tools
    - type **sudo apt install -y autoconf automake autotools-dev curl libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex texinfo gperf libtool patchutils bc zlib1g-dev libexpat-dev python3**

Step 4: Install iVerilog

1. Perform the installation
    - type **sudo apt install iverilog**
2. Verify the installation was successful
    - type **iverilog -v**
    - if you get back the version, that means it is installed!

Step 5: Install CocoTB Environment

1. If you are working on linux, you need to install the Windows Subsystem for Linux (WSL)
    - type **wsl --install**
2. Open the wsl terminal

    - this is as simple as typing **wsl** or opening your WSL application if you downloaded one
3. Install python, python-venv, iverilog, and gtkwave in the wsl environment
    - type **sudo apt update && sudo apt upgrade -y**
    - type **sudo apt install python3 python3-pip -y**
    - type **sudo apt install python3-venv -y**
    - type **sudo apt install iverilog -y**
    - type **sudo apt install gtkwave -y**
4. Create a python virtual environment, feel free to get creative with the naming
    - type **python3 -m venv YOUR_VENV_NAME**
5. Activate the virtual environment
    - type **source myenv/bin/activate**
6. Install cocoTB
    - type **pip install cocotb**
7. Modify ./.bashrc to mount your working directory and activate your virtual environment
    - go to the home directory
    - type **nano ./.bashrc**
    - Add the following commands to the end of the file that has just opened
        - “cd /mnt/YOUR_WORKING_DIRECTORY”
        - “source ~/YOUR_VENV_NAME/bin/activate”
    - Type **ctrl+X** to exit and save the file

Steps for Mac

Step 1: Prepare You Materials

- Open your Linux terminal (Terminal, etc.)

Step 2: Install Command Line Tools

1. Install Apple’s command line utilities
    - type **xcode-select --install**
2. Next we need “homebrew”, which is a third party command line tool
    - type **/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"**
      ** Note: This may take a while so feel free to brew yourself a cup of coffee or tea while you wait!
3. To set up *homebrew*, open the “.bashrc” with a text editor (vim, nano, etc.) This file can be accessed within your home directory
    a. type or copy the following lines into your “.bashrc” file
      ** Note: You can do this using your text editor of choice (Nano, Vim, etc.)
      # homebrew
      export PATH="/opt/homebrew/bin:${PATH}"
      eval "$(/opt/homebrew/bin/brew shellenv)"

b. Now you should close and reopen your terminal which will allow you to use *brew* commands

Step 3: Install Prebuilt RISC-V Toolchains

1. After restarting your terminal do the following:
    a. type **brew tap riscv-software-src/riscv**
    b. type **brew install riscv-tools**
2. You can check if the installation finished with the following:
    a. type **brew test riscv-tools**
        - This should bring up a line that says “Testing riscv-software-src/riscv/riscv-tools” if things are working correctly

Step 4: Install All Necessary Tools

1. Use *brew* to run the following command
    a. type **brew install cmake ninja wget git**
2. While in your home directory enter the following commands in order:
    a. **sudo mkdir -p /opt/bison/bison-3.7.91**
    b. **wget http://alpha.gnu.org/gnu/bison/bison-3.7.91.tar.xz**
    c. **tar -xvf bison-3.7.91.tar.xz**
    d. **cd bison-3.7.91**
    e. **./configure --prefix=/opt/bison/bison-3.7.91 CFLAGS="-isysroot $(xcrun -show-sdk-path)" CXXFLAGS="-isysroot $(xcrun -show-sdk-path)"**
    f. **make**
    g. **sudo make install**
3. Type or copy the following lines into your “.bashrc” file

    # bison
    export CFLAGS="-I/opt/bison/bison-3.7.91/share/include ${CFLAGS}"
    export CXXFLAGS="-I/opt/bison/bison-3.7.91/share/include ${CXXFLAGS}"
    export LDFLAGS="-L/opt/bison/bison-3.7.91/lib ${LDFLAGS}"
    export PATH="/opt/bison/bison-3.7.91/bin:${PATH}"
4. Now go back to you home directory and enter the following commands in order:
    a. **sudo mkdir -p /opt/flex/flex-2.6.4**
    b. **wget https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz**
    c. **tar -xvf flex-2.6.4.tar.gz**
    d. **cd flex-2.6.4**
    e. **./configure --prefix=/opt/flex/flex-2.6.4 CFLAGS="-isysroot $(xcrun -show-sdk-path)" CXXFLAGS="-isysroot $(xcrun -show-sdk-path)"**
    f. **make**
    g. **sudo make install**
5. Type or copy the following lines into your “.bashrc” file

    # flex

export CFLAGS="-I/opt/flex/flex-2.6.4/include ${CFLAGS}"
export CXXFLAGS="-I/opt/flex/flex-2.6.4/include ${CXXFLAGS}"
export LDFLAGS="-L/opt/flex/flex-2.6.4/lib ${LDFLAGS}"
export PATH="/opt/flex/flex-2.6.4/bin:${PATH}"

6. Add the following line to you ".bashrc" file as well
   # systemc
   export SYSTEMC_HOME="[PATH_TO_SYSTEMC]"
   - PATH_TO_SYSTEMC should be whatever location you decide to install the SystemC packages within
7. Now restart your terminal or use the following command to ensure the changes have taken effect
   - **source ~/.zshrc**
8. Now in your home directory and enter the following commands in order:
   a. **git clone https://github.com/accellera-official/systemc $SYSTEMC_HOME**
   b. **cd $SYSTEMC_HOME**
   c. **git checkout 2.3.3**
   d. **mkdir build**
   e. **cd build**
   f. **cmake -GNinja -DENABLE_PTHREADS=ON -DENABLE_PHASE_CALLBACKS_TRACING=OFF -DCMAKE_CXX_STANDARD=17 ..**
   g. **ninja**
   h. **ninja install**

- You're almost done! Just a few more commands and you'll be ready to go.

Step 5: Install iVerilog

1. type **brew install icarus-verilog**
2. Verify the installation was successful
   - type **iverilog**
   - You should see something that says "no source files"
3. If you do not see this add the following to your ".bashrc" file within the homebrew section
   export PATH="/opt/homebrew/Cellar/icarus-verilog/12.0/bin/iverilog"
   - Here this says "12.0" however the version number may be different for you

Step 6: Install CocoTB Environment

1. type **brew install --HEAD randomplum/gtkwave/gtkwave**

2. type **pip3 install cocotb**

Finish

Congratulations on finishing this guide! Now you should be ready to start coding up Verilog modules and testing them on your own computer. If you have any issues, try reinstalling packages that might be missing, or making sure that they are a part of your *$PATH*. If you run an installation command a second time, it will generally tell you whether or not you already have it installed at which point it is a good time to check your ".bashrc" file to ensure that the package is within the path.