

DIGI-KEY ELECTRONICS & IMAGINATION TECHNOLOGIES
ACADEMIC COMPONENT
REFERENCE GUIDE

RISC-V

By: Richard Sikora



DIGI-KEY: YOUR PARTNER FOR ACADEMIC SOLUTIONS

Your **FREE** Tools for Success!



EEWIKI

The eewiki is an environment for learning about various topics provided by Digi-Key's very own Application Engineers.



REFERENCE DESIGNS

Search and compare from manufacturer provided reference designs through Digi-Key's Reference Design Library.



EDA AND DESIGN TOOLS

Digi-Key's unparalleled design tool solutions are ready to help bring your designs to fruition quicker and easier than ever before.



CONVERSION CALCULATORS

Digi-Key's conversion calculators include tools for converting temperature, pressure, capacitance, resistance codes, and more.



SCHEME-IT®

Scheme-it is an online schematic and diagramming tool that allows anyone to design and share electronic circuit diagrams.



BOM MANAGER

With the Digi-Key BOM Manager, you can import your Bill of Materials, name and save your BOMs, and share your designs.



VIDEO AND ARTICLE LIBRARY

Learn from Digi-Key's library of tutorial videos, how-to demos, in-depth design articles and how-to articles.



AND MUCH MORE!

The vast engineering resources paired with the extensive inventory makes Digi-Key a great tool for your success!

Educational Kits

As the technological world continues to explode at an unprecedented rate, individuals looking to take the first step towards learning about this fast-growing field have a tough task ahead of them. Digi-Key Electronics has made this move a little easier with educational kits and projects.

Learn more about the newest technologies from companies at the forefront of technological innovation with help from Digi-Key!



Online Conversion Calculators

A one-stop resource for many electronics industry calculations!

TEMPERATURE CONVERSION	VOLUME CONVERSION	BATTERY LIFE CALCULATOR
PRESSURE CONVERSION	RESISTOR COLOR CODE CALCULATOR	LENGTH CONVERSION
DBM TO WATTS CALCULATOR	CAPACITANCE CONVERSION	TIME CONSTANT CALCULATOR
BTU/JOULES CONVERSION	WEIGHT CONVERSION	LED SERIES RESISTOR CALCULATOR
DECIMAL FRACTION CONVERSION	OHM'S LAW CALCULATOR	FIND THEM ONLINE AT: DIGIKEY.COM/CALCULATORS

No One Knows Boards Like Digi-Key: DIGIKEY.COM/BOARDS

Over 10,000 Different Boards and Modules in Stock from These Suppliers and More!



sparkfun
ELECTRONICS



Raspberry Pi
AUTHORISED DISTRIBUTOR



adafruit



ARDUINO



NORDIC
SEMICONDUCTOR



PIMORONI

VISIT DIGI-KEY'S ACADEMIC OFFERING AT
DIGIKEY.COM/EDU



DIGI-KEY AND IMAGINATION TECHNOLOGIES GUIDE TO RISC-V

Table of Contents

1. Introduction.....	3
2. RISC-V Background.....	3
3. RISC-V Basics	4
4. Hands On: Using the Seeed Technologies Maix BiT Board	5
5. Hands On: Using SiFive SoC on SparkFun RED-V Boards.....	13
6. Hands On: Implementing a Soft Core Using the Digilent Nexys A7	18
7. Soft Core Software Installation	21
8. Building the SweRVolf Core and Downloading it to a FPGA.....	27
9. Product Development using RISC-V	30
10. More Information on RISC-V	32
11. Conclusion	33
12. Acknowledgements.....	33
13. Author Profile	33
14. References.....	33

1 Introduction

RISC-V is a relatively new computer technology that is being actively promoted as a competitor to [ARM](#).

This guide has been written as a brief introduction for Digi-Key customers and students who would like to gain knowledge of RISC-V software and hardware.

At the RISC-V Global Forum on September 3, 2020, Imagination Technologies announced the “RVfpga: Complete Course in Understanding Computer Architecture” global teaching project for RISC-V as part of its university programme to be launched in November 2020. Some of the material from this course <https://university.imgtec.com> is presented here.

This document assumes the reader has little or no prior knowledge of RISC-V, which was exactly the position the author was in at the start of the project. It is targeted at university students who may be studying computer architecture and also electronics/computer engineers wishing to expand on their knowledge, either in the classroom or at home. Some basic knowledge of Linux is required.

1.1 What is RISC-V?

RISC stands for “**R**educed **I**nstruction **S**et **C**omputer”. Here the **V** stands for the Roman number 5. Hence RISC-V is the 5th Generation of a family of computer cores. It is pronounced “Risk Five”.

The RISC-V logo is a registered trademark of RISC-V International.



Figure 1: RISC-V Logo. Image Source: www.RISC-V.org

Detailed information such as specifications is available from www.RISC-V.org

1.2 Hands On

Rather than being purely theoretical, this guide provides hands-on experience of RISC-V using three different boards available from Digi-Key. All prices stated were correct at the time of writing but may be subject to later change, as is the software:

1. **Easy, student-friendly price and fun.** [Seeed Technologies Co Ltd](#) Maix BiT board gives a dual-core RISC-V processor, camera, LCD screen and image recognition software for just \$25. It is programmed using MicroPython. (Link to Maix BiT section of this guide).
2. **Mid-range System on a Chip (Soc).** RISC-V SOC programming in C or assembly language on two [SparkFun](#) boards costing \$30 and \$36. (Link to SparkFun section of this guide).
3. **Serious, intellectually demanding but expensive** using the Western Digital SweRV soft core in a [Xilinx](#) Field Programmable Gate Array (FPGA) used on the [Digilent](#) Nexys A7 board costing \$265. However, it can all be run just on a software simulator instead. (Link to Soft Core section of this guide).

The good news is that all the software used is free.

2 RISC-V Background

2.1 History

RISC-V was created by David Patterson, Krste Asanovic, Andrew Waterman and Yunsup Lee at the University of California, Berkeley in 2010. At that time there was Linux open source software, but there was no equivalent open source hardware. To facilitate their research, they created their own instruction set architecture (ISA). They built their first RISC-V chip in 2011 and launched their first commercial product in 2014.

There have been several other RISC implementations over the years including [Microchip](#) PIC, ARM, [Atmel](#) AVR, MIPS, SuperH and SPARC. The original RISC-I implementation was back in 1981.

2.2 Open Architecture and Open Source

In order to use an ARM® core in an integrated circuit such as a processor or application specific integrated circuit (ASIC), a license must first be obtained. This can be in millions of \$ for the latest cores and takes time. To a large multinational such as [NXP](#) or [Freescale](#), then the cost is a relatively small percentage of the \$100 million total development cost. However, when the new integrated circuit is complete, royalties have to be paid to ARM for every one sold.

RISC-V is different. There is no license fee and no royalties. This means a RISC-V implementation can be started at any time and there are no ongoing payments; good news to small companies or emerging markets such as India and China. The RISC-V architecture is fixed, which ensures backwards compatibility of future products.

Additionally, open source means that the design can be modified. Special instructions can be created to improve performance or make life difficult for hackers.

The [RISC-V development software](#), in particular PlatformIO, is free. Licensed software such as the ARM MDK using the [Keil](#) compiler can be an expensive overhead and the license always seems to run out the day before a critical deadline.

RISC-V cores are usually implemented in Linux rather than Windows, which makes both the cores and the software open source.

2.3 Key Players

Andes Technology is a Taiwanese semiconductor manufacturer. One of its products is the [N22](#) minimal RISC-V core running at 700 MHz, which takes only 0.013 mm² of silicon. It is designed for wearables and Internet of Things (IoT) applications.

Imagination Technologies is an intellectual property (IP) company that specializes in Graphic Processing Units (GPUs) used in mobile phones, tablets, computers and vehicle vision. Its best known product is the [PowerVR](#) and the 10th generation GPU “the A series” uses an embedded RISC-V controller.

PlatformIO provides free software tools for RISC-V development.

SiFive was founded by three of the original RISC-V creators Krste Asanović, Yunsup Lee and Andrew Waterman. It provides RISC-V cores, Systems on a Chip (SoCs), IPs and development boards.

Western Digital (makers of SanDisk products) is committed to using RISC-V in future products and has produced its own [family of certified RISC-V cores](#) known as SweRV.

3 RISC-V Basics

Although stated that RISC-V is a computer core, more correctly, RISC-V is an ISA. It is up to the users to produce their own implementation.

The RISC-V specifications are provided by [RISC-V.org](#) in two documents: the Unprivileged ISA Specification (for basic operations) and the Privileged ISA Specification (for use with operating systems).

Level	Encoding	Name	Abbreviation
0	00	User/ Application	U
1	01	Supervisor	S
2	10	Reserved	
3	11	Machine	M

Table 1: RISC-V Privilege Levels. Image Source: RISC-V.org

For most applications, the User/Application level is used, including the Zephyr operating system.

3.1 Core Naming Conventions

A RISC-V implementation can be made in 32bits, 64bits or 128bits, with 16 or 32 registers and uses a laid-down naming convention.

Name	Functionality
RV32I	Integer instruction set. 32bits and 32 registers
RV32E	Integer instruction set for embedded devices. 32 bits and 16 registers
RV64I	Integer instruction set. 64bits and 32 registers
RV128I	Integer instruction set. 128bits and 32 registers

Table 2: RISC-V Base ISAs. Image Source: RISC-V.org

Additionally, there is choice of which instructions are implemented. The most common MAFD are also referred to collectively as G (general).

Letter	Functionality
G	M Integer multiplication and division
	A Atomic Instructions
	F Single precision floating point
	D Double precision floating point
	Q Quad precision
	L Decimal floating point
	C Compressed instructions (16 bit instructions)
	B Bit manipulation
	J Dynamically translated languages
	T Transactional memory
	P Packed SIMD instructions
	V Vector operations
	N User level interrupts
	H Hypervisor

Table 3: RISC-V Standard ISA Extensions. Image Source: RISC-V.org

So, for a small SoC which used minimal resources, RV32EMAB might be specified. This means *Integer Instruction set for embedded devices, 32 bits and 16 registers, Integer multiplication and division, Atomic instructions* (to avoid contention with interrupts), no floating-point maths, but with *Bit manipulation*.

On the other hand, if a 64-bit RISC-V implementation is required, including all four general instructions MAFD, plus *Bit Manipulation* and *User Level Interrupts*, it would be referred to as RV64GBN

3.2 Available Cores for Download

A large number of RISC-V cores have been developed, mostly by universities around the world. The cores have been written in a number of Hardware Description Languages (HDLs) such as VHDL, Verilog, System Verilog and the less well-known Chisel. For C programmers, the most easy to use is Verilog as it has a C-like syntax. System Verilog is a superset and is also used for validation. Note that Verilog is a hardware description language, not a programming language.

For a list of available of RISC-V cores, please see <https://github.com/riscvarchive/riscv-cores-list> and type “riscv cores list” in the Search box.

Figure 2 shows the timeline and performance of some RISC-V cores.

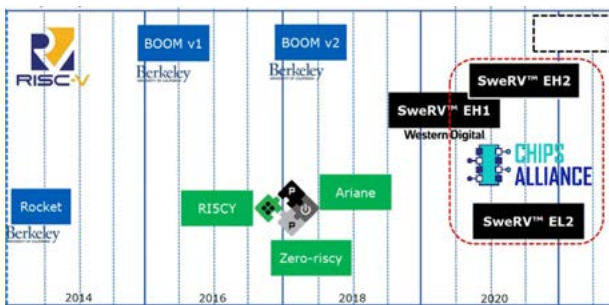


Figure 2: Boom, Rocket, Riscy and SweRV Cores. Image Source: Western Digital

Boom is the Berkeley Out-of-Order RISC-V Processor written in Chisel.

The Rocket core from lowRISC is an RV64G variant with a 5-stage pipeline.

RISCY is a simple core developed by ETH Zurich and the Università di Bologna.

SweRV is a family of 3 RISC-V processors from Western Digital for industrial applications.

Additionally, Freedom is a core used by SiFive.

3.3 Performance Against ARM and Intel

Figure 3 provided by Western Digital shows the relative performance of ARM, Intel and three RISC-V cores – Boom, Rocket and SweRV. The CoreMark is a benchmark devised by the Embedded Microprocessor Benchmark Consortium (EEMBC) for the performance of a microprocessor when carrying out a specified range of common used operations e.g. sort algorithm, cyclic redundancy check (CRC), state machine etc.

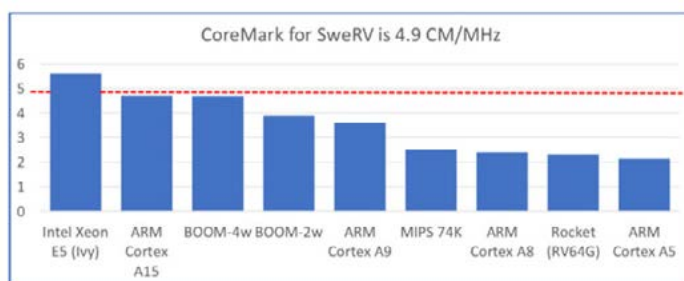


Figure 3: Western Digital CoreMarks. Image Source: Western Digital

It shows that in terms of performance, the various RISC-V cores are comparable with ARM Cortex A5, ARM Cortex A8, ARM Cortex A9 and ARM Cortex A15 in terms of performance. The Intel Xeon has higher performance.

4 Hands On: Using the Seeed Technologies Maix BiT Board

Of the four hands-on boards, this has the lowest cost and is the simplest to use.

4.1 Maix BiT Board Description

The board is designed for Internet of Things (IoT) and Artificial Intelligence (AI), such as image recognition. It offers a very high performance processor at a low price.

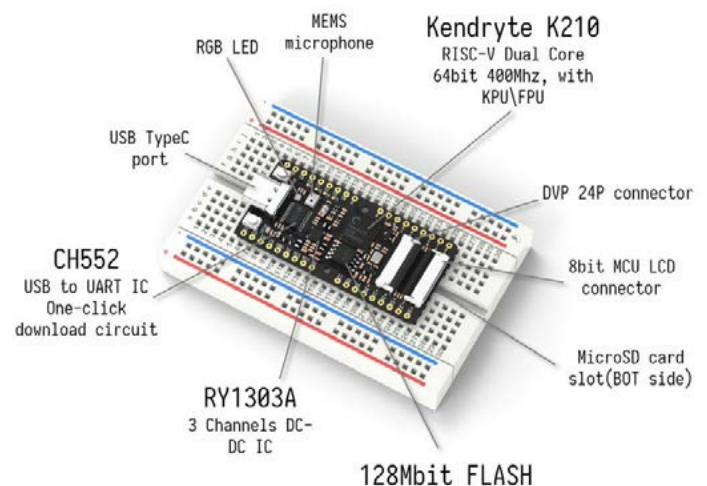


Figure 4: Maix BiT Details. Image Source: Seeed Technologies Co Ltd

4.2 Basic Maix BiT Board

The basic Maix BiT board can be used, which is:

Seeed Technology Co. Ltd Sipeed Maix [BiT RISC-V AI+IoT](#), Digi-Key 1597-1714-ND \$14.65

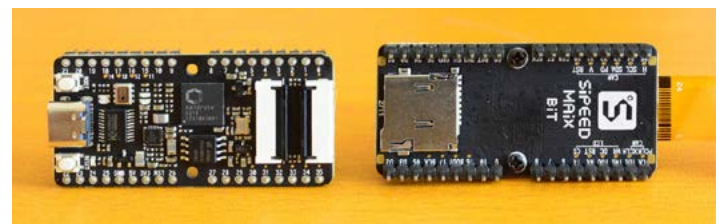


Figure 5: Seeed Technologies Maix BiT Board

4.3 Maix BiT Board with Camera and LCD

However, for the hands-on session, the following kit with a camera and LCD screen is recommended:

Seeed Technology K210 [Sipeed Maix BiT Kit RISC-V AI+IoT](#) Digi-Key 1597-1713-ND \$24.21

Note: the Maix BiT board is not supplied with a USB-A to USB-C cable, so one also needs to be purchased:

Seeed Technology [USB-C cable](#) Digi-Key 1597-106990248-ND \$2.42

4.3.1 Maix BiT Components Supplied

The box contains a Maix BiT board, a camera and a LCD display. Also supplied are a screwdriver and 3 screws + nylon spacers, although only 2 screws and spacers are actually needed.



Figure 6: MaixPy BiT Box and Board with Camera, LCD and Components.

4.3.2 Assembling the MaixPy BiT Board

Assembly time: About 5 minutes. The components are small and therefore a little fiddly.

Additional tool required: pliers to hold the nylon spacer while tightening the screws.

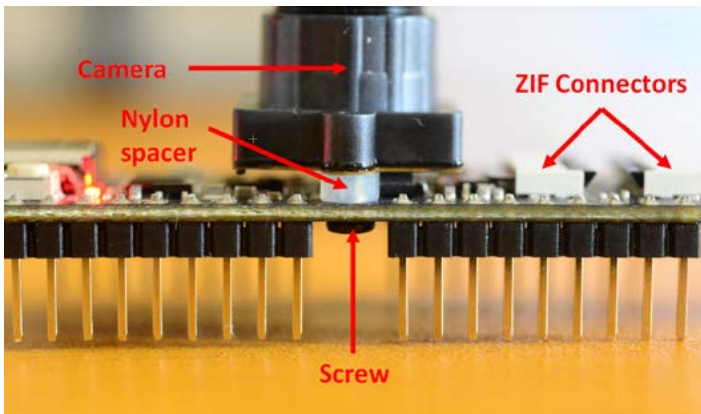


Figure 7: Camera Mounting

4.3.3 Maix BiT Assembly Instructions

1. Fit the two screws and nylon spacers.
2. With the black locking bars upwards, plug the camera cable into the white ZIF connector near the middle of the board.
3. Push the black locking bar downwards to lock the camera cable in place.
4. It is necessary bend the camera cable to align the camera over the screws.
5. Tighten the two screws with the screwdriver to hold the camera in place.

6. Insert the LCD cable into the ZIF connector near the edge of the board.
7. Push the black locking bar downwards to lock the LCD in place.

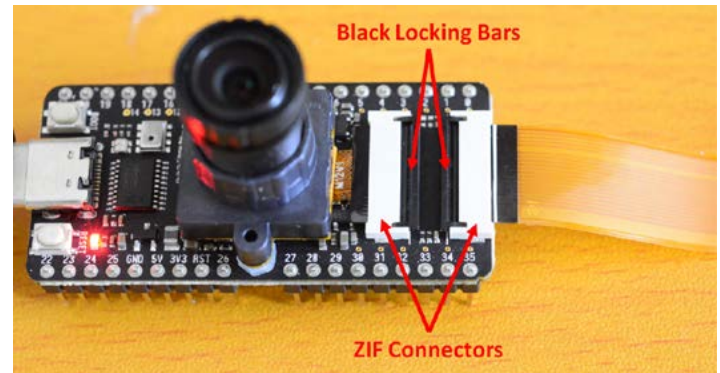


Figure 8: Camera and LCD Fitted in ZIF sockets

4.4 Maix BiT Board Software

The board comes pre-programmed with a MicroPython interpreter called MaixPy. Programs can be written using either the MaixPy IDE software or simply by typing commands into a serial terminal.

Software is available for download both for Windows and Linux, although Linux has some extra steps. (Link to install Linux part of this guide).

4.5 Installing MaixPy IDE on Windows

This is an Integrated Development Environment (IDE) used to write code and run the Maix BiT board.

Installation time: About 10 minutes.

The MaixPy IDE software is a little difficult to find as what looks like the location is blank.

Go to: <https://dl.sipeed.com/MAIX/MaixPy/ide/v0.2.5>

4.5.1 Select MaixPY IDE for Windows

Click on maixpy-ide-windows-0.2.5.exe. The file size is 85.5MB.

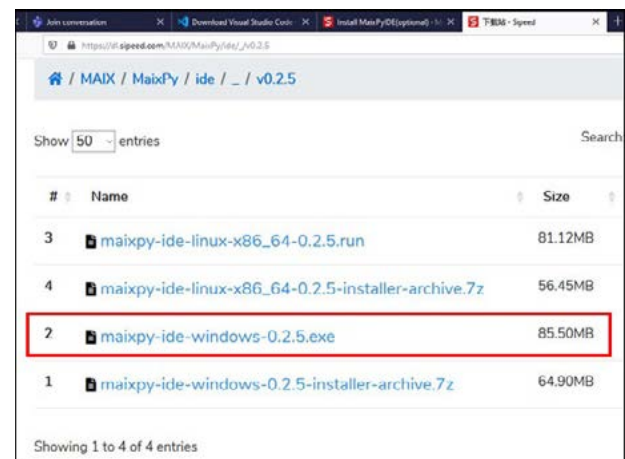


Figure 9: Click on MaixPy IDE for Windows

4.5.2 Download MaixPy IDE Windows

Click on *Save File* then double click on `maixpy-ide-windows-0.2.5.exe` to install.

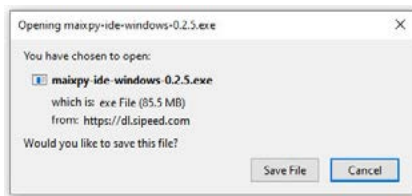


Figure 10: Download maixpy-ide-windows

The program is installed to: `C:\Program Files (x86)\MaixPyIDE\bin\maixpyide.exe`

4.5.3 Shortcut to MaixPy Windows

By default, MaixPy does not put a shortcut icon on the Desktop. To create a shortcut to MaixPy, right click on `maixpyide.exe` and select *Create shortcut*, which can be saved on the Desktop.

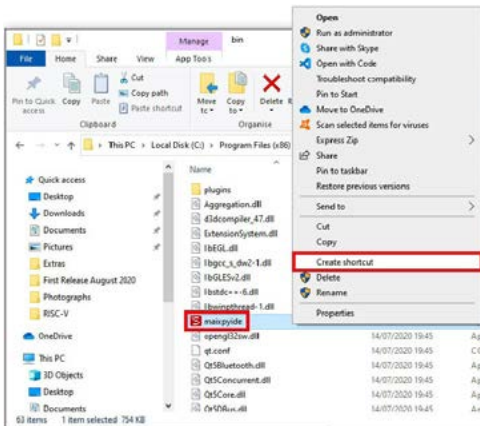


Figure 11: Shortcut to MaixPy IDE

([Link to Running a Program in MaixPy IDE](#))

4.6 Installing MaixPy IDE using Linux

Installation time: About 10 minutes. The downloaded file size is 287 MB.

There is also a command line version available which takes 40 minutes to download and install.

The MaixPy IDE software is a little difficult to find, as what looks like the location is blank.

Go to <https://dl.sipeed.com/MAIX/MaixPy/ide/v0.2.5>

This goes to the MaixPy Documentation Page which contains useful information.

4.6.1 Select File

Click on: `maixpy-ide-linux-x86_64.0.2.5.run`

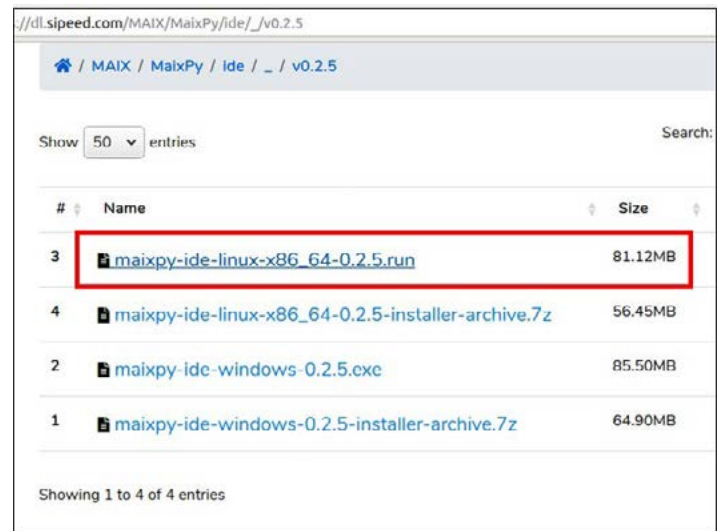


Figure 12: Select Linux Run File

4.6.2 Save Linux Install File

Click on *OK* to save the file.

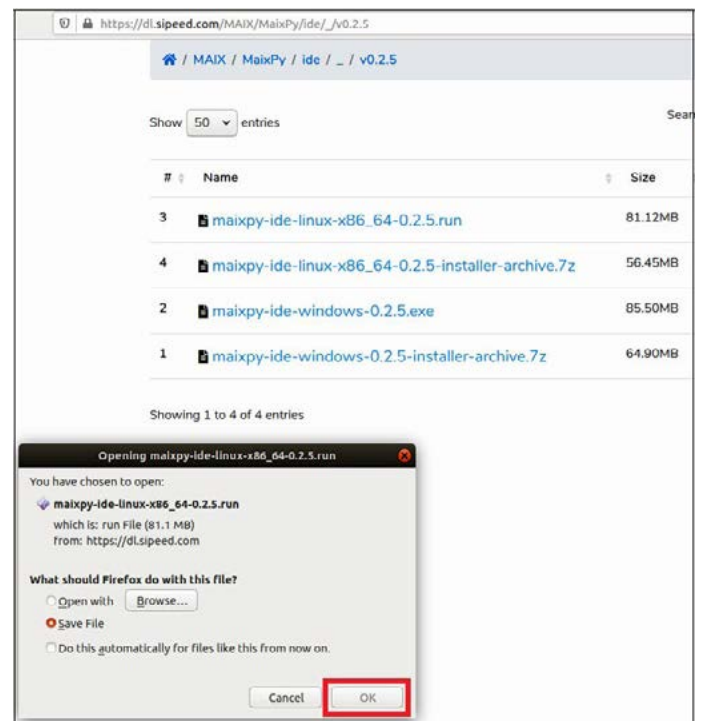


Figure 13: Save Linux Install File

4.6.3 USB Port Access

By default the USB port is not generally available, which prevents a MaixPy program from running.

Open an *Ubuntu Terminal*  and type:


```
sudo adduser myname dialout
```

where myname is substituted by the computer user name, the same as in /home/myname/

Restart the computer so the change has effect.

If this step is not carried out, a prompt will appear later.

4.6.4 Running the Downloaded file in Linux

Open an *Ubuntu Terminal* .

The instructions supplied for MaixPy were written for version 0.2.2 rather than version 0.2.5, so are out of date.

```
chmod +x maixpy-ide-linux-x86_64-0.2.2.run
./maixpy-ide-linux-x86_64-0.2.2.run
```

Figure 14: MaixPy Instructions

Instead type in the later file name:

```
chmod +x maixpy-ide-linux-x86_64-0.2.5.run
./maixpy-ide-linux-x86_64-0.2.5.run
```

The *MaixPy IDE Setup Wizard* should appear.


4.6.5 MaixPy IDE Setup Wizard



Figure 15: MaixPy Setup Wizard

Click on *Next*➤, then follow the steps in the Wizard using the defaults. When complete, the opening screen should appear.

4.6.6 MaixPy Opening Screen

If the opening screen does not show the image from the camera (Frame Buffer) and the Red Green Blue (RGB) signals, click on the  icon on the right hand side of the screen.

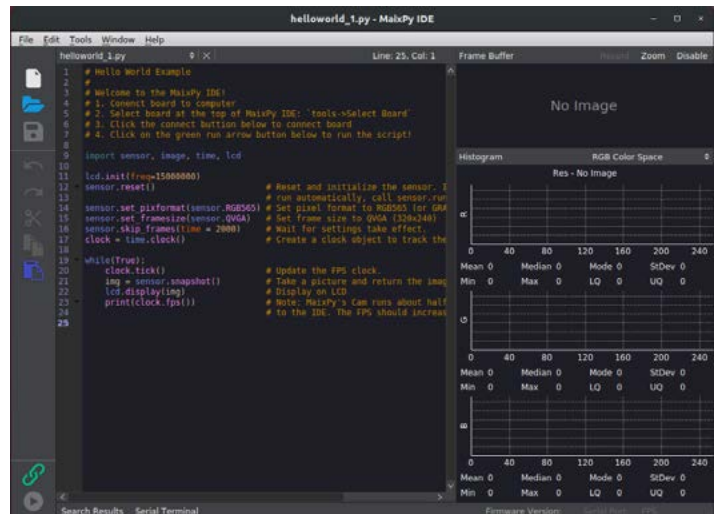


Figure 16: MaixPy Linux IDE Screen

4.7 Running a Program in MaixPy IDE

The procedure for Linux and Windows is similar.

4.7.1 Plugging in the Board

Plug the USB cable into the computer and the welcome message will be displayed on the LCD.

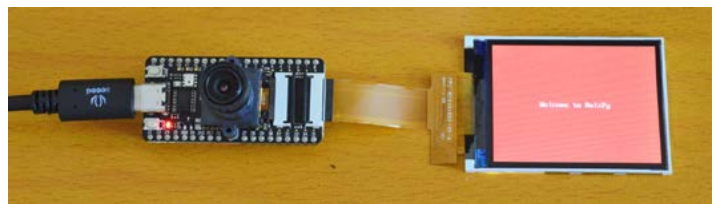


Figure 17: Maix BIT camera and LCD Operating

4.7.2 Starting MaixPy

If MaixPy IDE is not already running, go to the Ubuntu search tool and type in MaixPy. Click on the *MaixPy IDE* icon.



Figure 18: MaixPy IDE Linux Icon

4.7.3 Selecting the Program

The *helloworld_1.py* program is loaded by default at first power up.

4.7.4 Selecting the Board

On the toolbar select *Tools, Select Board* then *Sipeed Maix Bit (with Mic)*.

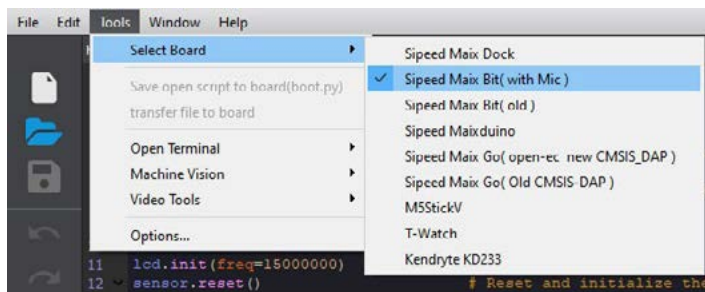



Figure 19: MaixPy IDE Selecting the Board

4.7.5 Selecting the Serial Port

Click on the green icon  on the bottom left of the screen to show the Connect – MaixPy IDE menu. There will be more than one serial port to choose from.

When the correct serial port has been selected, it connects in seconds and the green icon on the bottom left turns red.

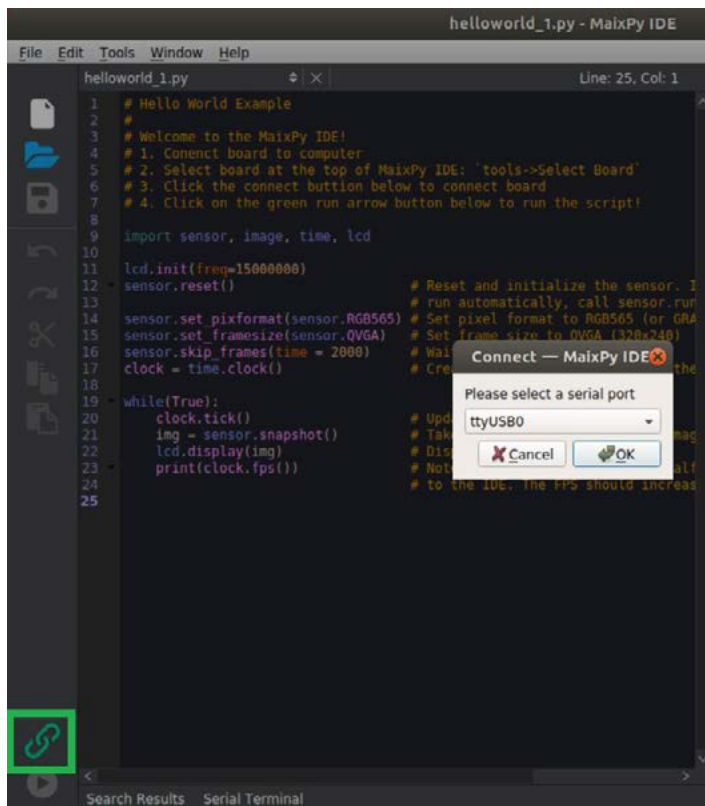


Figure 20: MaixPy Serial Connection

4.7.6 Linux Pop up the First Time Through

If the Linux USB ports have not been made available to MaixPy IDE, a reminder similar to this will pop up, with the actual computer username rather than 'richard'. Follow the instructions.

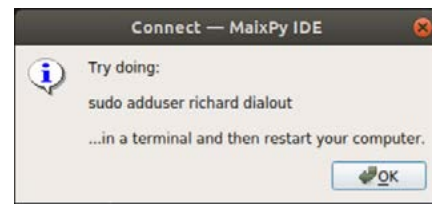



Figure 21: Dialout Group

4.7.7 Running the Program

Click on the green arrow  at the bottom left of the screen to run the program.

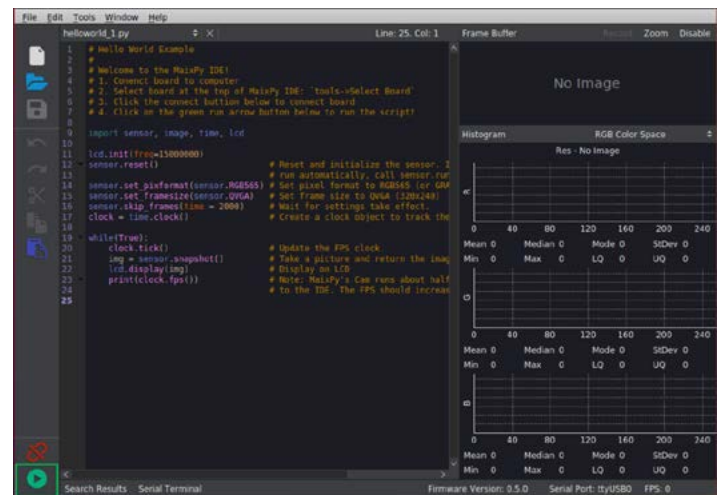


Figure 22: Program Set Up Ready to Run

4.7.8 Hello World Program Running

The helloworld_1.py program is now running. The picture from the camera is displayed on the computer, as well as the RGB spectrum. The picture is also shown on the LCD screen.

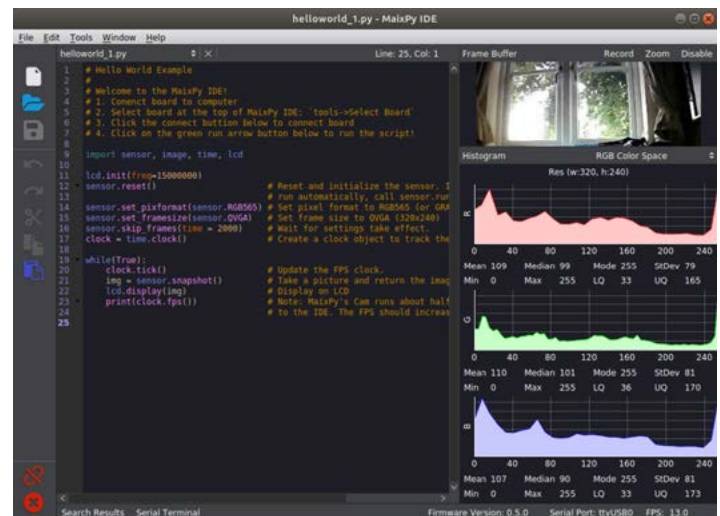


Figure 23: Running the Hello World Application in Linux

To stop the program, click on the Stop  icon.

4.8 Further MaixPy Projects

There are further projects available, but these need to be downloaded. In MaixPy IDE, click on *File, More examples, Examples on github repo*, which goes to Github. This applies to both Linux and Windows versions.

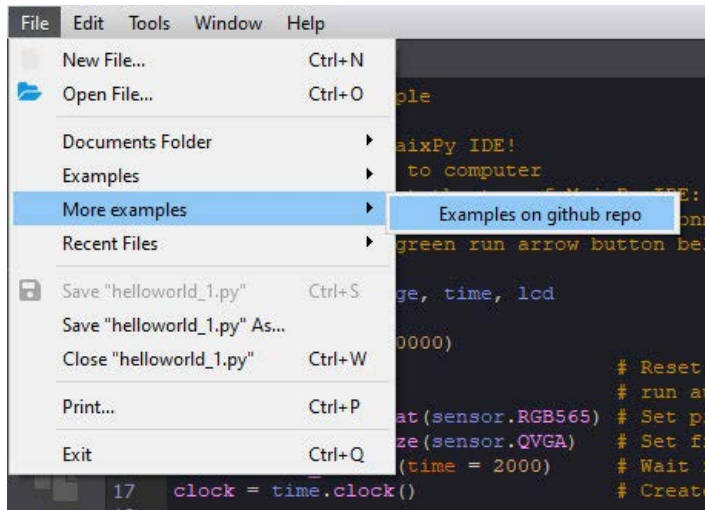


Figure 24: More MaixPy Examples

4.8.1 Download MaixPy Examples

Download `MaixPy_scripts-master.zip`, save it, and then unzip the files.

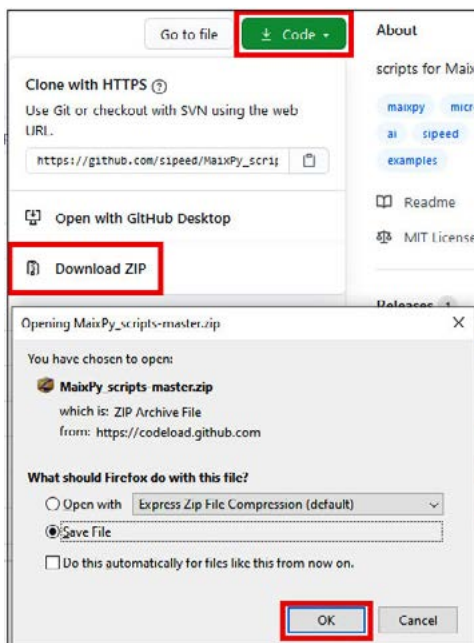


Figure 25: MaixPy Examples

4.9 Face Recognition Project

Time required: about 15 minutes. This involves downloading and installing two more programs.

The further examples for MaixPy provide a face recognition program `demo_find_face.py` in the `machine_vision` directory. Load this into MaixPy IDE.

4.9.1 Additional File Required

This project requires an additional model called `face_model_at_0x300000.kfpkg` to be loaded onto the MaixPy BiT board before running. To do this, the `kflash_gui` tool is needed.

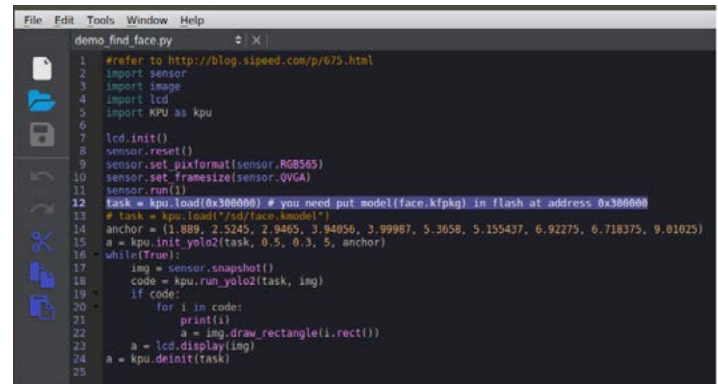


Figure 26: Face Recognition Program Additional File

Note: on the earlier versions of `kflash_gui`, the default download location in was `0x00000`. This meant it was possible to download `face_model_at_0x300000.kfpkg` to address `0x00000`, rather than the correct address `0x300000`. This overwrote the MicroPython interpreter and the board did not run at all. It was then necessary to reprogram the board.

4.10 Updating the Maix BiT Flash

Time required: About 10 minutes.

The `kflash_gui` tool runs on both Windows and Linux. Flashing the board with the latest software is quick and easy to do.

4.10.1 Download kflash_gui

The `kflash_gui` can be downloaded from:

https://github.com/sipeed/kflash_gui/releases.

Here `kflash_gui_v1.5.3_windows.7z` was used. Later versions caused problems with the Avast Virus checker.

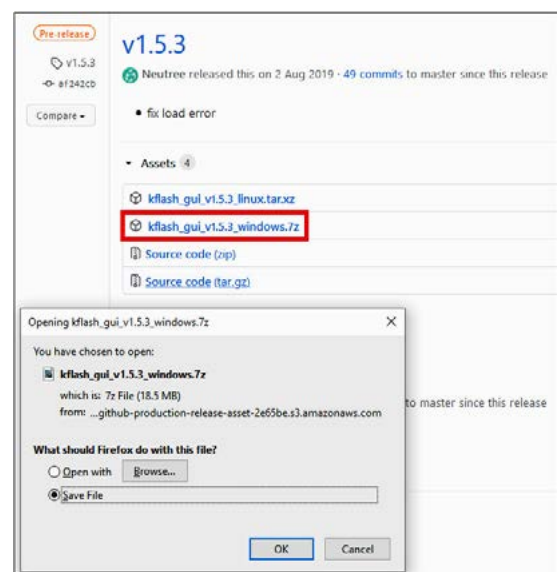



Figure 27: Download of kflash GUI

Extract the .7z file. Should the computer not support .7z files, there are free tools online such as Express Zip available from:

<https://www.nchsoftware.com/software/utilities.html>.

To run the program, go to the extracted file and double-click on the  kflash_gui icon.

Note: the kflash_gui installation does not put a shortcut on the Desktop. Should one be required, this will need to be done manually.

4.10.2 Download the Face Model

Download `face_model_at_0x300000.kfpkg` from <https://github.com/sipeed/MaixPy/releases> and save the file.

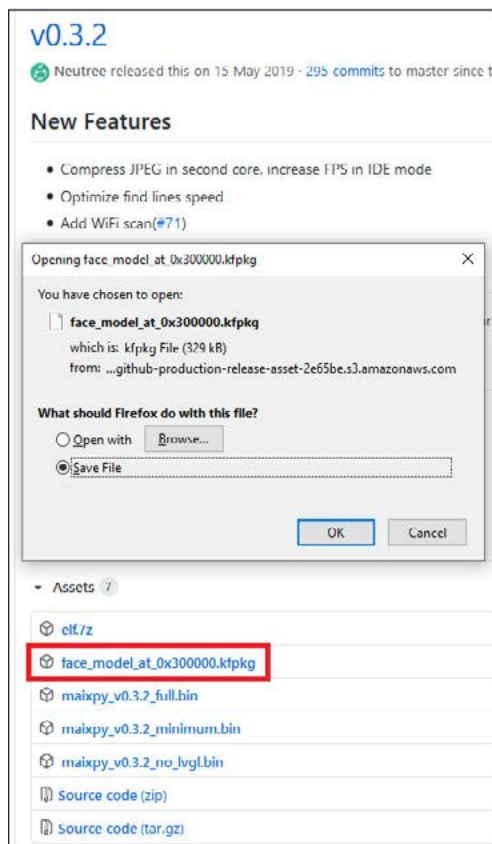



Figure 28: Download the Face Model

4.10.3 Programming the Face Model using kflash GUI

Click on the  kflash_gui icon. Click on Open File and select `face_model_at_0x300000.kfpkg`. Select the Speed Maix Bit (with Mic) board, serial Port and then click on Download.

The download takes about 20 seconds.

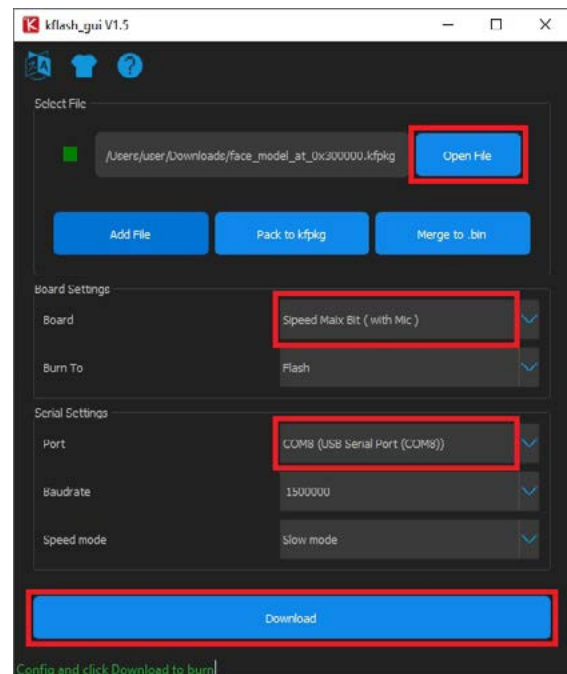


Figure 29: kflash GUI

4.11 Running the Face Recognition Program

Return to MaixPy IDE and run `demo_find_face.py` from the `machine_vision` directory.

When a face is recognized, a rectangle is drawn on the LCD and on the computer screen.

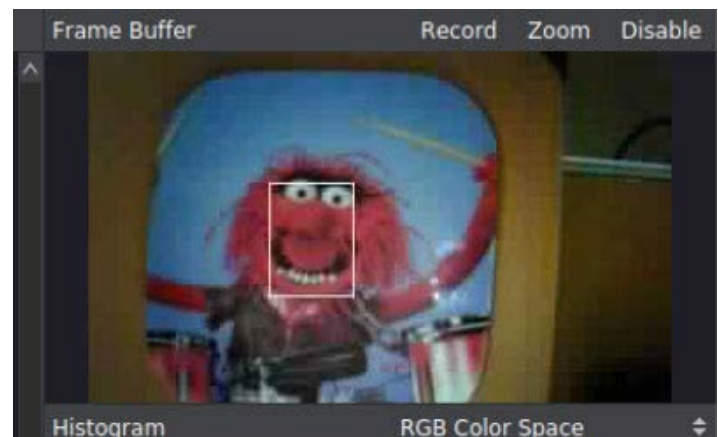


Figure 30: Face Recognition Example

4.12 Reflashing the Maix Bit Board Main Software

The kflash_gui is also used to reprogram the MicroPython interpreter on the Maix Bit board. There are frequent firmware updates, but not all are compatible with MaixPy IDE.

4.12.1 Download Latest Software

Go to: <https://dl.sipeed.com/MAIX/MaixPy/release/master>. This contains a list of versions of MaixPy. Click on the latest version.

4.12.2 Selecting Latest MaixPy

Click on the file that ends with `_minimum_with_ide_support.bin` and save it.

It is important that the file has IDE support.

4.12.3 Program the Maix BiT board

Open `kflash_gui`. Use Open File and select the file downloaded in Section 4.12.2.

Select the *Board*, serial *Port* then click on *Download*.

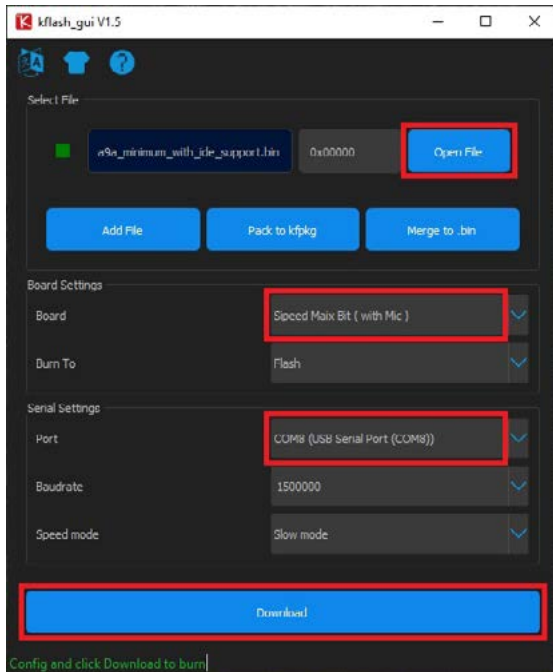


Figure 31: Reprogramming the Maix BiT flash

4.13 Running MaixPy on a Serial Terminal

Time required: about 5 minutes.

Besides using MaixPy IDE, programs can also be run on a serial terminal. This is quick and easy to do and is useful for checking the configuration of Maix BiT boards and the software version. It can also be used as a tool for learning MicroPython.

4.13.1 Setting up a Serial Terminal

If not already installed, download and install any free serial terminal program e.g. Putty or TeraTerm.

Set up a serial port speed of 115200 Baud. The Port COM8 may be different on another computer.

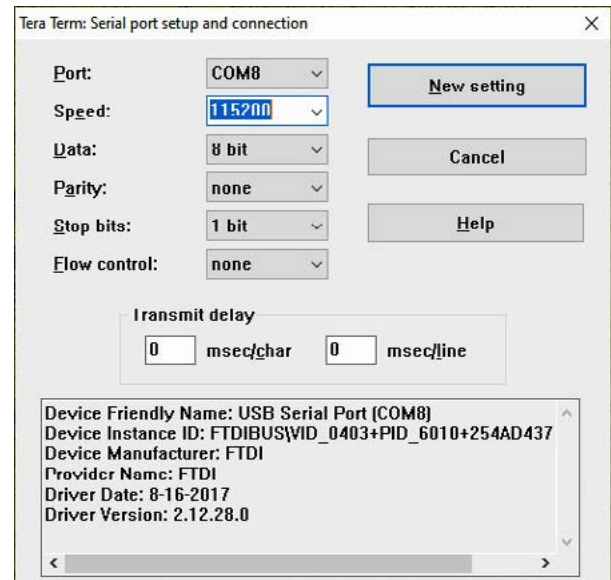


Figure 32: Serial Terminal Configuration

4.13.2 Press and Release the Reset Button

Press and release the Reset Button on the MaixPy BiT board.

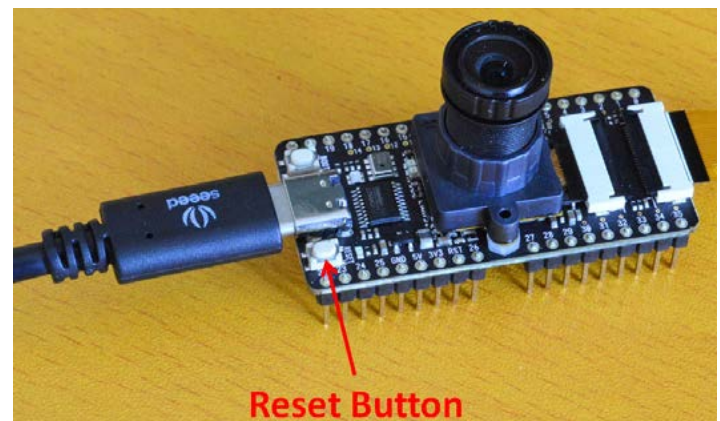


Figure 33: Maix BiT Board Reset Button

4.13.3 Running MaixPy on a Serial Terminal

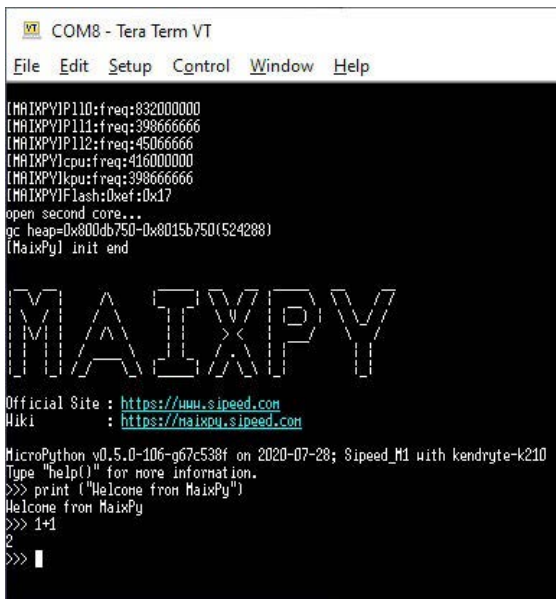


Figure 34: MaixPy on a Serial Terminal

Type `print ("Welcome from MaixPy")` the response is "Welcome from MaixPy".

Enter `1+1` and the response is 2.

5 Hands On: Using SiFive SoC on SparkFun RED-V Boards

This is the mid-range option with two boards designed to be programmed in C using the FreedomStudio IDE.

5.1 SparkFun Video

There is an [excellent short video by Shawn Hymel](#) on the Digi-Key website showing how to use these boards. Before purchasing a board, installing the software, or later while waiting for the software to download, it is well worth watching.



Figure 35: Getting Started with RED-V. Source: Digi-Key website.

The link to the video is also on the Digi-Key website on the Red Board and RED-V Thing Plus pages.

5.2 Hardware

Two boards are available from SparkFun Electronics based on the SiFive Freedom Everywhere RISC-V SoC. Both boards are compatible with the SiFive1 Rev B boards and software:

The small SparkFun [RED-V SIFIVE RISC-V THING PLUS](#) - 1568-DEV-15799-ND \$29.95

The larger SparkFun Electronics FE310 [Red Board](#) - 1568-DEV-15594-ND \$39.95

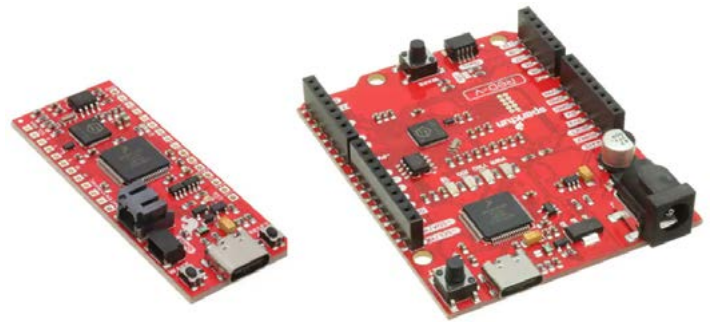


Figure 36: SparkFun RED-V Thing Plus (left) and Red Board (right). Image Source: SparkFun Electronics

Note: Neither the Thing Plus nor the Redboard are supplied with a USB-C to USB-A cable, so one needs to be purchased e.g.:

Seeed Technology USB-C cable Digi-Key 1597-106990248-ND \$2.42

Software is available from SiFive for both Windows and Linux. Unfortunately, the author could not get the Linux version to run, so only the Windows version is considered here.

5.3 SiFive Freedom Studio Software Installation Windows

The installation takes about 20 minutes and the downloaded file size is 1.3GB.

Go to www.SiFive.com/software and scroll down the page to the Freedom Studio section.

Click on **Windows** then save the .zip file.

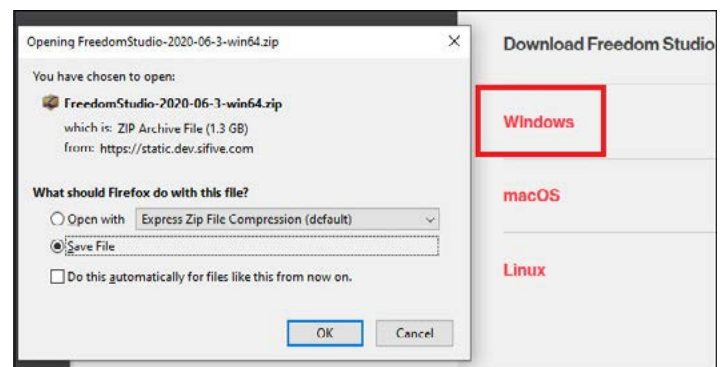


Figure 37: SiFive Freedom Studio Windows

5.3.1 Freedom Studio File Location

Extract the files. Navigate to the install directory e.g.:

`C:\Users\user\FreedomStudio-2020-06-3-win64\`

Double click on  FreedomStudio to run it.

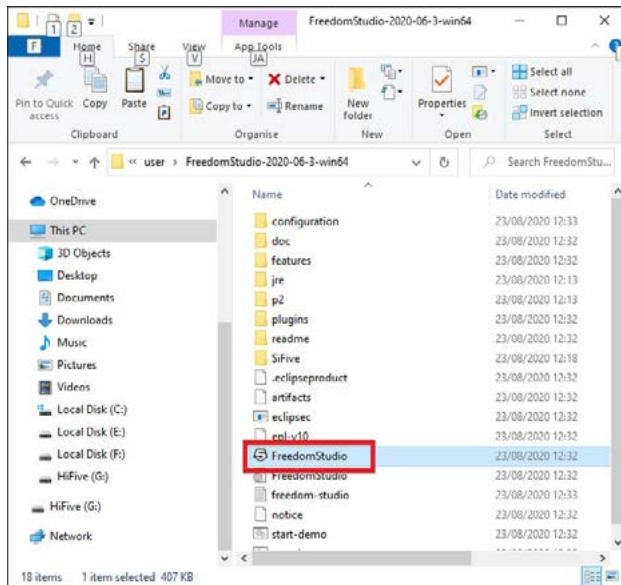


Figure 38: FreedomStudio File Location

5.3.2 Freedom Studio Startup Screen

The Freedom Studio Icon should appear. A shortcut can be added to the Desktop if required.



Figure 39: Freedom Studio Startup Screen

5.4 Creating a New Software Project

If not already running, start Freedom Studio.

5.4.1 Plug in the Board

It is recommended to plug in the board before creating a program because it automatically sets up the debug configuration. Use the USB-C to USB-A cable. Either SparkFun board can be used.

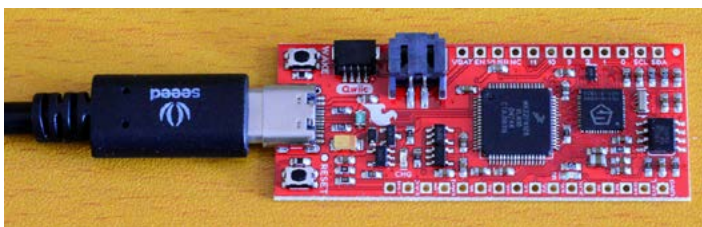


Figure 40: Thing Plus Board connected with USB-C cable

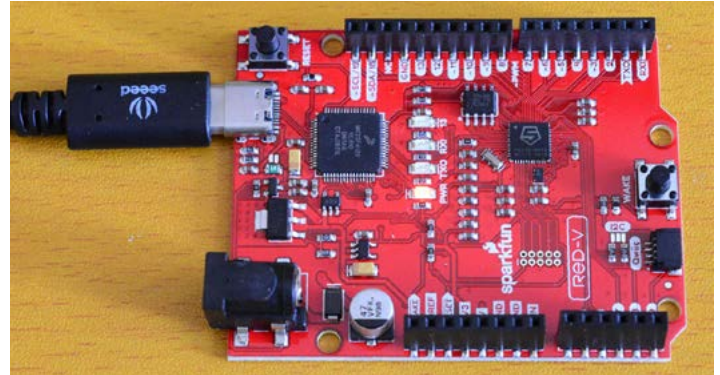


Figure 41: Red Board connected with USB-C cable

5.4.2 Create a new Freedom E SDK Software Project

Go to the *SiFive Tools* tab at the top of the screen and click on *Create a new Freedom E SDK Software Project*.

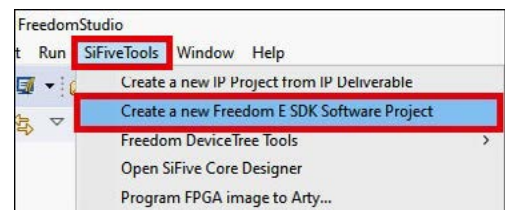


Figure 42: Create a new Freedom E SDK Software Project

5.4.3 Choice of Board

Several boards are supported. From the pull-down menu select *siFive-hifive1-revb*. This is compatible with both the Red Board and the Thing Plus Board.

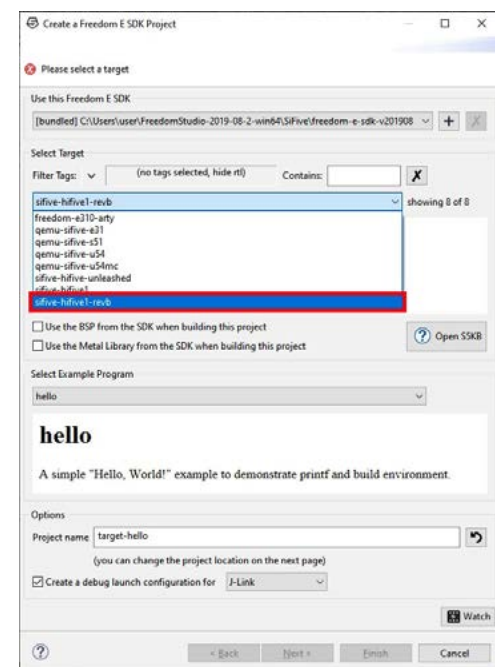


Figure 43: SiFive Board Selection

5.4.4 Choice of Example Program

From the *Select Example Program* pull down menu, a range of example projects are available.

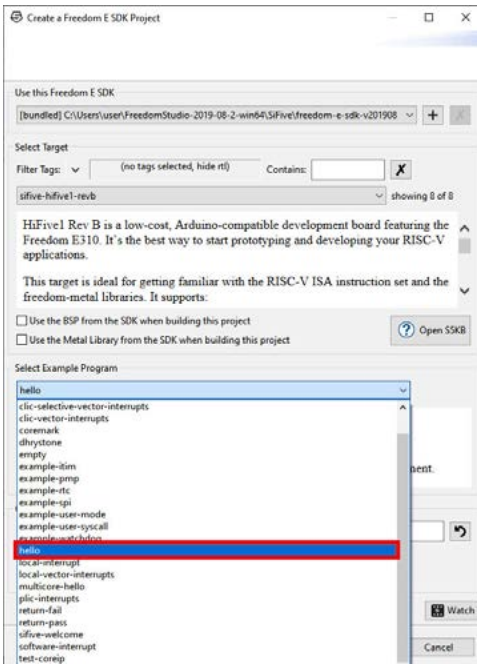


Figure 44: SIFive Sample Projects

These example projects make an excellent starting point for other projects. For now stay with *hello*.

5.4.5 Debug Launch Configuration

At the bottom of the window, the *Create a debug launch configuration for J-Link* has automatically been set up. This is because the board is plugged in. Click on *Finish*. The project will automatically build.

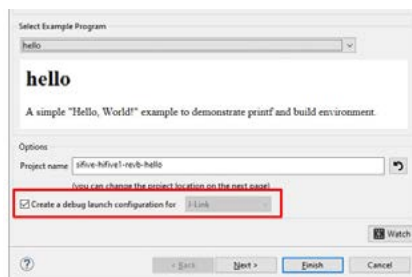


Figure 45: Select Example Program and Debug Launch Configuration

5.4.6 Check the .elf File

The Edit Configuration window will pop up automatically. Check that the Executable and Linkable File *hello.elf* has been created. This is needed to program the board.

It is possible to go straight to debug by clicking on *Debug*, but for now click on *Close* as there are a few more things to set up.

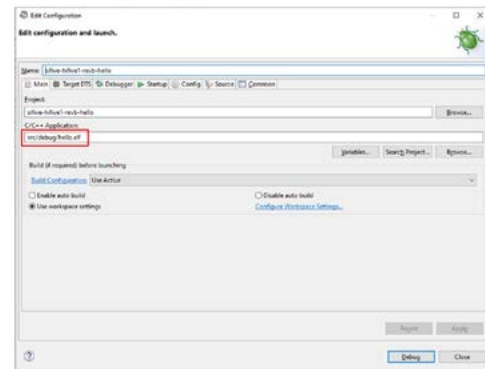


Figure 46: Edit Configuration Menu

5.4.7 Console and Source Code View

The *Console* window gives information about the build – which compiler was used, the code size and the build time.

The first build takes a while as it is compiling all the files in the project. Later builds only recompile the files that have actually changed, so are much quicker.

The *hello.c* window gives the C code.

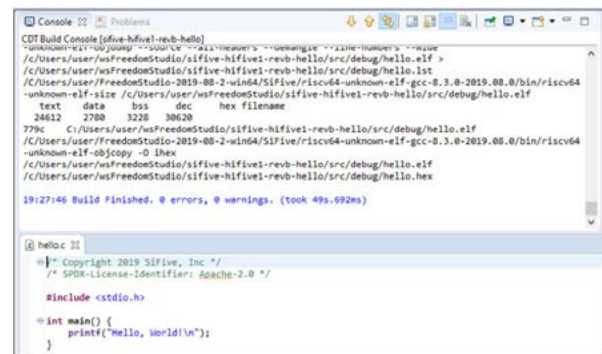



Figure 47: Console and Source Window

5.4.8 Modifying the hello.c Source Code


If the *hello.c* program were debugged as it is, not a great deal would be seen because all it does is terminate. Therefore, modify *hello.c* as shown in Figure 48 to print *Hello, World!* ten times.



Figure 48: Modified hello.c File

The program can be built by clicking on the *Build* icon  on the toolbar to check the syntax, but when Debug is run, this is done automatically.

5.4.9 Debugging the Program

To start debugging the program, click on the Debug icon  on the toolbar or press the keyboard F11 key or Run from the toolbar then *Debug*.

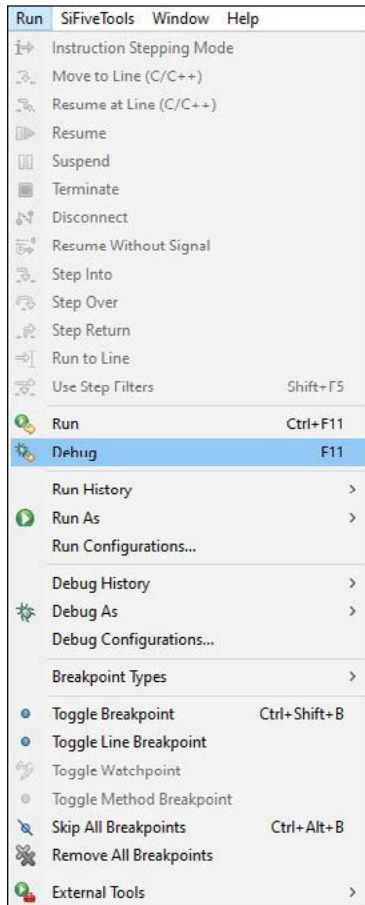


Figure 49: Start Debugging



5.4.10 Debugger Stops at main()

The debugger will stop at the beginning of `main()`.



Figure 50: Debugger stops at main()

5.4.11 Setting up a Serial Terminal for printf

Click on the *Terminal*  tab then click on *Open a Terminal*  icon to the right of the screen.

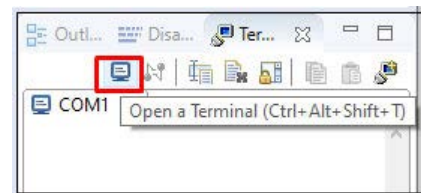


Figure 51: Open a Terminal Icon

5.4.12 Launch Terminal

Setup a Serial Terminal at 115200 Baud. There is more than one *Serial port*, so several attempts may be necessary.

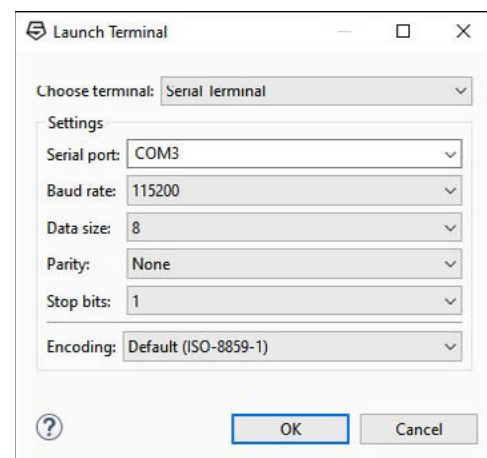



Figure 52: Serial Terminal Setup

5.4.13 Debug Output

To step through the code, click on the Step Over  icon on the toolbar or press the F6 function key on the keyboard several times so that Hello, World! appears ten times.

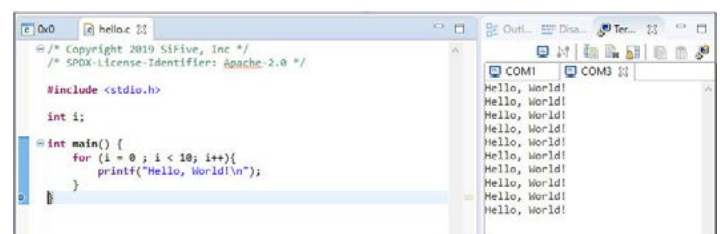


Figure 53: Debug Output on COM3

The fact that the `printf` messages are displayed in a window in the same program is very useful for debugging and code coverage. It is much easier than having to connect to a separate terminal.

5.4.14 Disassembly Window

Figure 54 shows a slightly modified version of the `hello.c` program for comparison purposes with ARM®.

To view the assembly language instructions that make up the C code, click on the Disassembly tab .

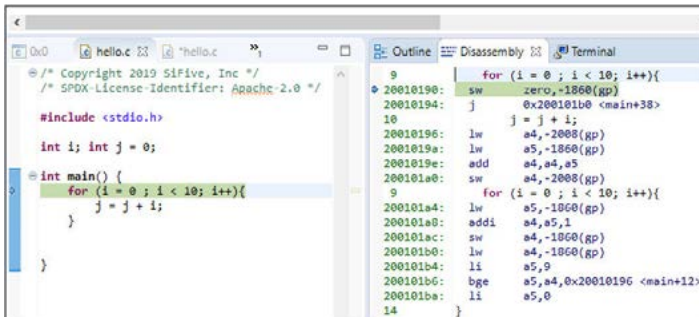


Figure 54: Disassembly Window

5.4.15 Comparison with ARM Cortex M0

Figure 55 shows the equivalent ARM® Cortex® M0+ disassembly code.

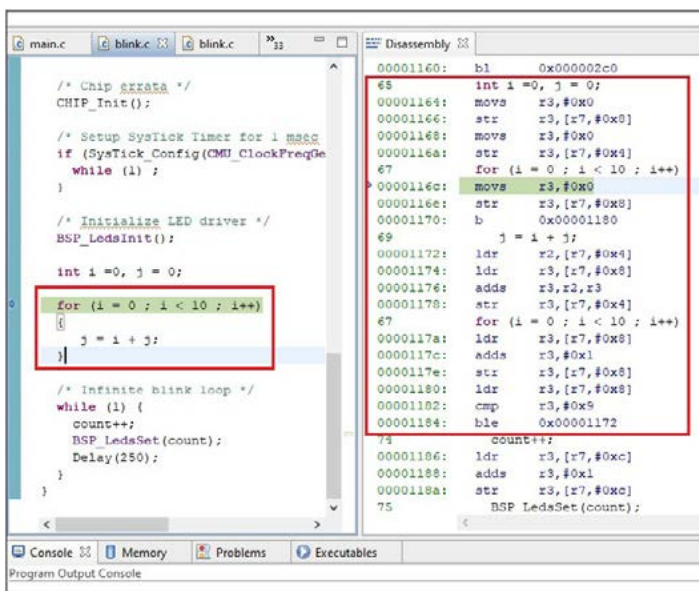


Figure 55: Comparison with ARM Cortex M0

It shows that in this case RISC-V requires fewer instructions than ARM.

5.4.16 Stopping the Program

To stop the program and to run the program again from the beginning, click on the **Terminate** icon either on the **Toolbar** or in the **Console** window.

To start debugging the program again, click on the **Debug** icon on the toolbar or press the keyboard F11 key.

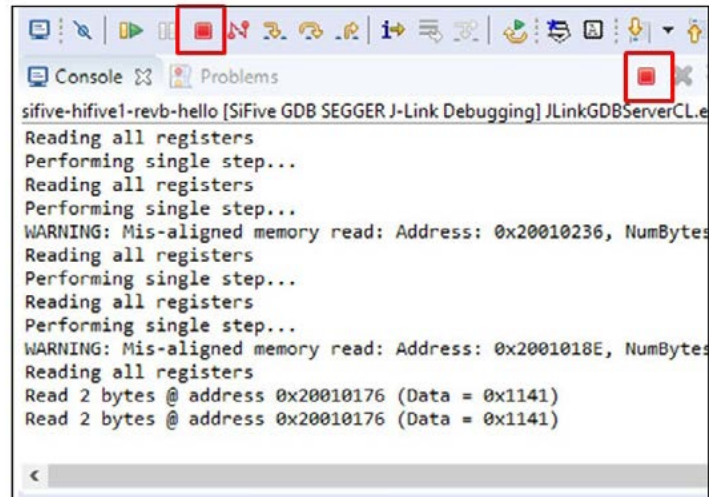


Figure 56: Terminate Program

5.4.17 Closing and Reopening an Existing Project

There is a minor annoyance when an existing Freedom Studio project is closed and opened again. Close Freedom Studio then open it again. Click on the project then click on the **Debug** icon. The program does not debug and an error message is displayed:

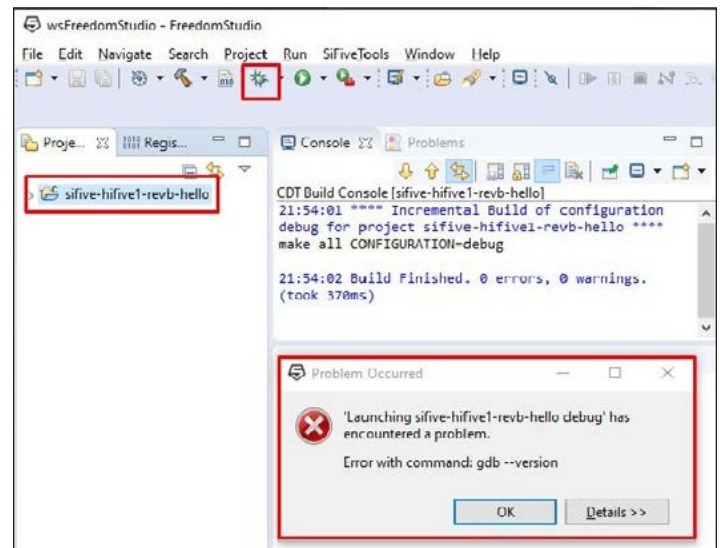


Figure 57: Problem when reopening a Project

5.4.18 Select Debug Configurations

From the **Toolbar** at the top of the screen, select **Run** then **Debug Configurations**.

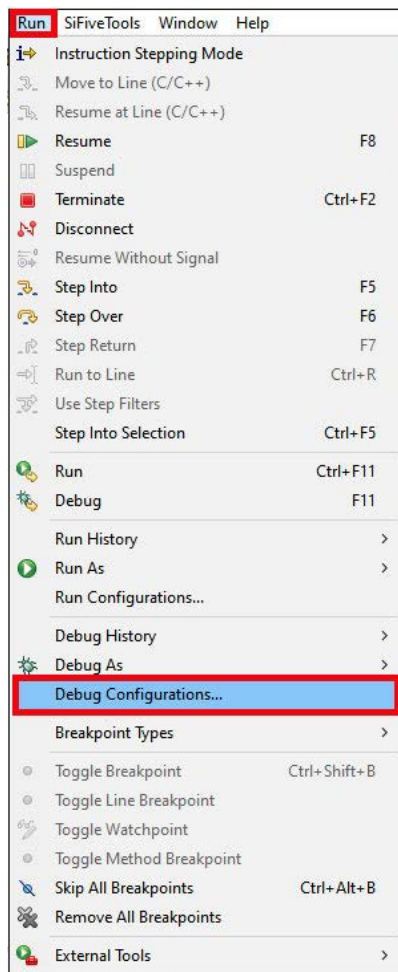


Figure 58: Select Debug Configurations

5.4.19 Create, Manage and Run Configurations

Expand the *SiFive GDB SEGGER J-Link Debugging* tab and click on the project to be debugged, here `sifive-hifive-1-revb-hello`. Click on *Debug*. It should now be possible to debug the program.

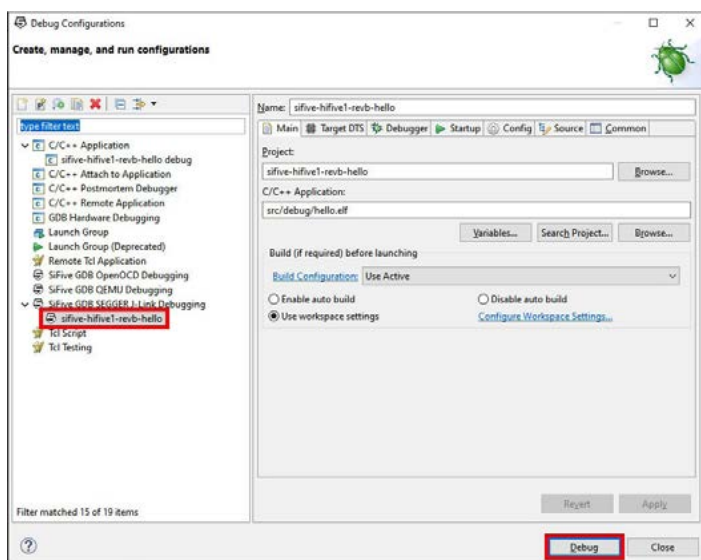


Figure 59: Select on project to debug

6 Hands On: Implementing a Soft Core Using the Digilent Nexys A7

This is the most technically demanding part of the guide and involves embedding a RISC-V core into a FPGA. It is ideal for the detailed study of computer architecture, but may be too complex for some readers.

6.1 Support from the Imagination Technologies University Programme



Implementing a soft core on a FPGA is a major undertaking. Fortunately, the Imagination Technologies University Programme is releasing worldwide teaching materials to implement a RISC-V core on a Xilinx Artix-7 FPGA, known as the "RVfpga: The Complete Course in Understanding Computer Architecture". This consists of a full package of 20 laboratories, documentation, examples and exercises with solutions. All the necessary software can be downloaded for free. This saves the user a great deal of work and headaches. The version of the material presented here represents 10% of the complete course.

To gain access to the RVfpga materials, it is first necessary to register at:

<https://university.imgtec.com/forums/?wpforo=signup>

The registration form is unfortunately a little on the long side and is designed for universities. However, the RVfpga materials are available to people outside universities, so where a university related field is asked for, the equivalent from a commercial or other organisation can be entered. The time required to register is under 10 minutes as not all the fields have to be filled in. Spaces are not allowed in phone numbers.

When registered, log in and request a download of the materials from the Teaching Resources page:

<https://university.imgtec.com/teaching-download/>

The download material is equivalent to three semesters of undergraduate study at university and covers 20 laboratories, all of which is free. There are also forums to discuss issues and answer questions.

The Masters-level "Creating a SoC" course follows in Q1 2021.

6.2 Soft Core Option

Providing that a project is not too cost sensitive, having a soft core in an FPGA gives the designer much more freedom than using a standard processor. There are a range of off-the-shelf peripherals available to download from www.opencores.org. Algorithms such as Fast Fourier Transforms (FFTs), digital filters and complex encryption algorithms can be implemented in hardware, not software, giving higher processing speed. PCB layout is much easier as peripherals can be routed to virtually any pin.

6.3 Board Required to Run Programs

For software simulation and software debugging only, the board is not required.

The board used is:

[Diligent Nexys A7 Nexys A7 ECE FPGA Trainer Board](#) Digi-Key 1286-1081-ND \$265.00

Its predecessor, the now obsolete Nexys4 DDR board can also be used, as it is very similar.

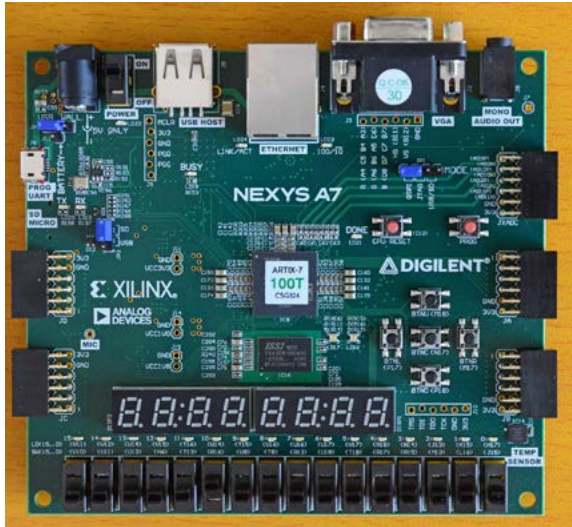


Figure 60: Digilent Nexys A7 Board.

6.4 Software Required

The following programs are required:

Program	Purpose
Xilinx Vivado 2019.2	Builds the Verilog .v and System Verilog .sv files that make up the RISC-V core. Also downloads the generated .bit file to the hardware
Visual Studio Code (VSCode)	IDE for C and assembly language software development
PlatformIO	An add-on to VSCode that provides support for RISC-V processors
Open OCD	Open source On Chip Debugger
RISC-V Toolchain	Compiler and other files required to build RISC-V projects
Verilator	Hardware simulator and processor for Verilog .v and System Verilog .sv files
Whisper	Instruction Set Simulator (ISS) from Western Digital. Allows code to be run and debugged on a computer without hardware
Digilent Board Files	Specific configuration files for the Digilent Nexys A7 Board
GTKWave	Waveform viewer for viewing low-level system timings
RVfpga	Software cores, code examples and Verilog / System Verilog projects
RVfpga Labs	20 Laboratories

Table 4: Programs Required

At present, all the programs run on Linux only, although Windows and Mac support is in development. The full download and installation time is in the region of 4 hours.

6.5 Running a RISC-V Implementation

There are two tasks required in order to run code on a RISC-V in a Xilinx FPGA:

1. Configure the FPGA hardware using a .bit file.
2. Compile the code and download the .elf file to the FPGA.

6.5.1 Relationship between FPGA Hardware and Software Components

Figure 61 shows how Xilinx Vivado is used to process the Verilog .v and System Verilog .sv files and convert them to a .bit file, which can be programmed into the FPGA.

C or assembly code is compiled using PlatformIO and the .elf file is downloaded to the FPGA where it can be run and debugged.

Communication with the Digilent Nexys A7 board is via a single shared USB port. It is not possible to communicate with the board using Vivado and Visual Studio Code at the same time.

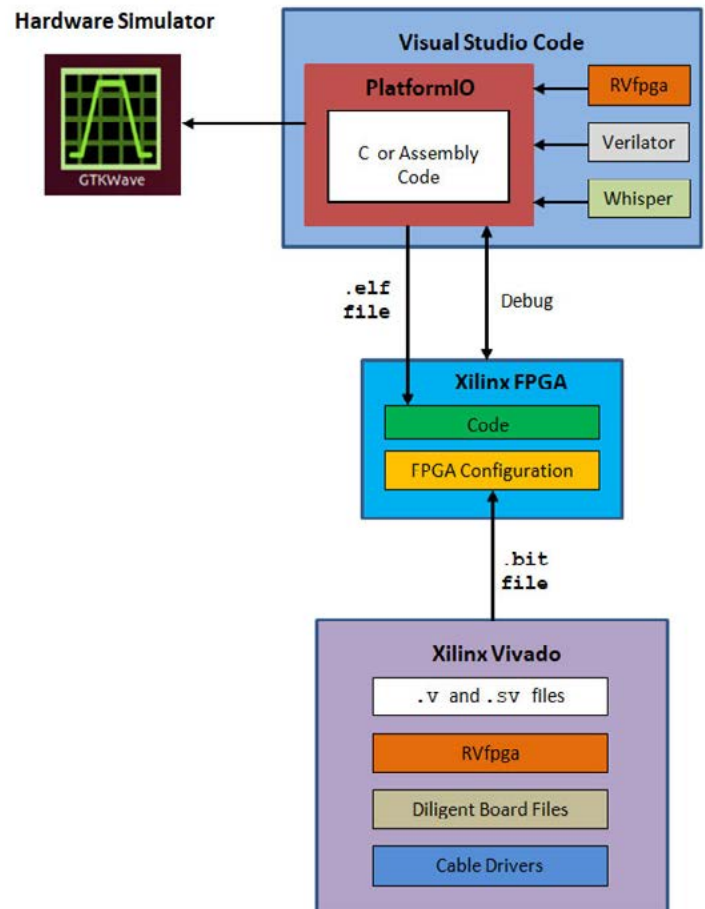


Figure 61: Visual Studio, FPGA and Xilinx Vivado Relationship

6.6 Western Digital SweRV Cores

Although there are a wide range of RISC-V cores available, a large number of them are part of academic research rather than being fully

certified. It is not suggested that these cores do not work properly, but for industrial products, the RISC-V core would have to be fully validated and certified to avoid the potential high costs of product liability.

Western Digital (Revenue \$16.57 billion in 2019) has developed a family of three certified SweRV cores for use in commercial products known as EH1, EH2 and EL2.

6.6.1 List of SweRV Cores

Core Name	Pipeline Stages	Threads	Size in mm @TSMC	CoreMarks/MHz
EH1	9	Single	0.110@28nm	4.9
EH2	9	Dual	0.067@16nm	6.3
EL2	4	Single	0.023@16nm	3.6

Table 5: Western Digital SweRV Cores. Image Source: Western Digital

Here the sizes refer to figures obtained from the Taiwan Semiconductor Manufacturing Company (TSMC).

All three cores are RV32IMC, which means Integer Instruction Set, 32 bits and 32 registers, Integer multiplication and division and Compressed instructions (16 bits).

6.6.2 Difference between SweRV Cores

The EH1 and EH2 cores both have full functionality in terms of pipeline, EH1 with a single thread and EH2 with dual threads. The EH2 has higher specification terms of 6.3 CoreMarks/MHz, but has a more complicated architecture.

Where size and cost are important, the EL2 core is a good choice, although there is a drop in performance down to 3.6 CoreMarks/MHz. It uses a simplified pipeline with 4 stages rather than 9.

The EH1 core has already been implemented and tested on the Digilent Nexys A7 board, and therefore has been chosen for the hands-on part of this guide. There are plans for the future to add the higher performance EH2 core and lower resources EL2 core.

6.6.3 SweRV EH1 Core in Detail

The block diagram of the SweRV EH1 core is shown in Figure 62:

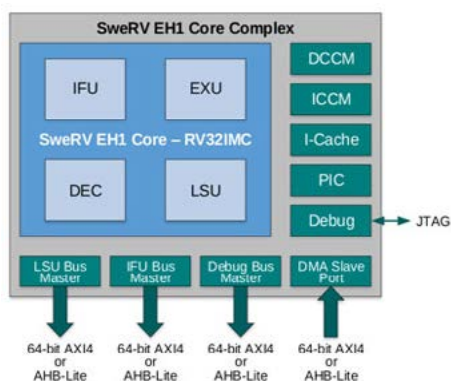


Figure 62: SweRV EH1 Core Complex. Image Source: Western Digital

The core is divided into 4 Units for instruction fetch (IFU), execute (EXU), decode (DEC) and load/store (LSU).

6.6.4 SweRV EH1 Individual Components

Abbreviation	Description
IFU	Instruction Fetch Unit
EXU	Execution Unit
DEC	Decode Unit
LSU	Load Store Unit
DCCM	Closely-Coupled Data Memory
ICCM	Closely-Coupled Instruction Memory
I-Cache	Instruction Cache
PIC	Programmable Interrupt Controller
AXI	Advanced eXtensible Interface
AHB	Advanced High Performance Bus

Table 6: SweRV EH1 Components

AXI4 and AHB-Lite are two industry standard interconnection buses. JTAG is used for code download and debugging.

6.7 SweRVolf Core

By itself, the SweRV EH1 is just a basic computer core. It needs additional components to make it of practical use. Olof Kindgren has written a set of wrappers for SweRV known as SweRVolf. The RVfpga authors have extended SweRVolf with some useful additional peripherals and have reorganized the HDL sources.

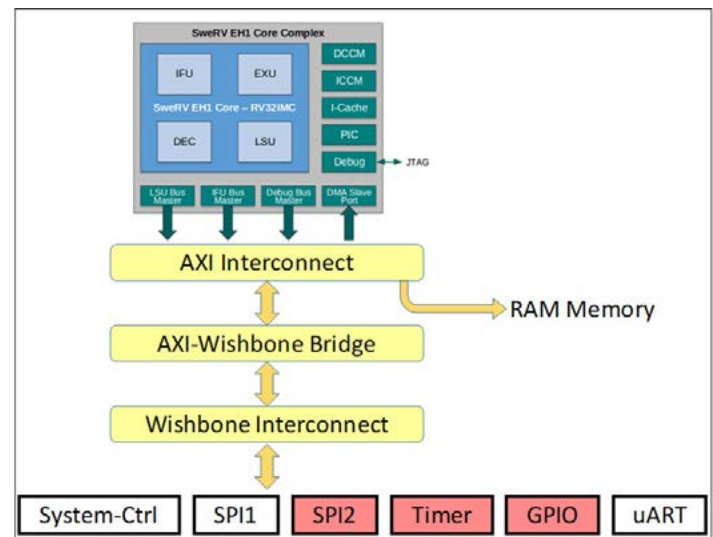


Figure 63: SweRVolf Core. Image Source: Imagination Technologies UP

6.7.1 Extended SweRVolf Components

The extended SweRVolf implementation is broken down into the following components:

- Wishbone Interconnect: an OpenCores interconnect, simpler than AXI.
- axi2wb: An AXI to Wishbone bus converter.
- GPIO: General Purpose Input Output from OpenCores with 64 input/output ports.

- Timer: from Opencores
- SPI: an open-source Serial Peripheral Interface controller (obtained from https://opencores.org/projects/simple_spi).
- UART: an open-source UART controller (obtained from <https://opencores.org/projects/uart16550>).
- Boot ROM: a Boot ROM contains a first-stage bootloader. After system reset, SweRV will start fetching the initial instructions from this area.
- RAM: the SweRV Core does not include a memory controller, but it reserves the first 128MB of its memory map and provides access to an AXI bus, so that the user can include RAM memory.
- SPI Flash: an SPI Flash memory (or Quad SPI) can also be included using the SPI controller described in the previous section.

6.7.2 SweRVolf Memory Map

All the components are memory-mapped as follows:

Core Map	Memory Address Range
Boot ROM	0x80000000 - 0x80000FFF
System Controller	0x80001000 - 0x8000103F
SPI1	0x80001040 - 0x8000107F
SPI2	0x80001100 - 0x8000113F
Timer	0x80001200 - 0x8000123F
GPIO	0x80001400 - 0x8000143F
UART	0x80002000 - 0x800020FF

Table 7: SweRVolf Memory Map

6.8 SweRVolf Verilog and System Verilog Files

The hierarchy of files for Figure 64 is shown in Verilog .v and System Verilog .sv format. These are all provided by the RVfpga package.

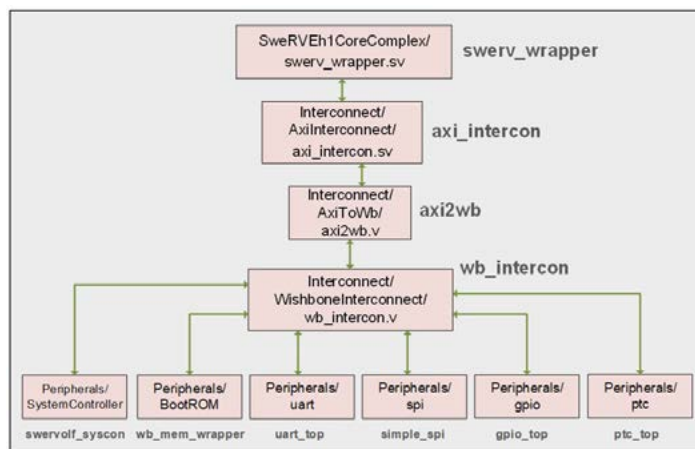


Figure 64: SweRVolf File Structure. Image Source: Imagination Technologies UP

6.9 SweRVolf Hierarchy for Nexys A7

All the components for the SweRVolf project are provided in the RVfpga package and can be downloaded.

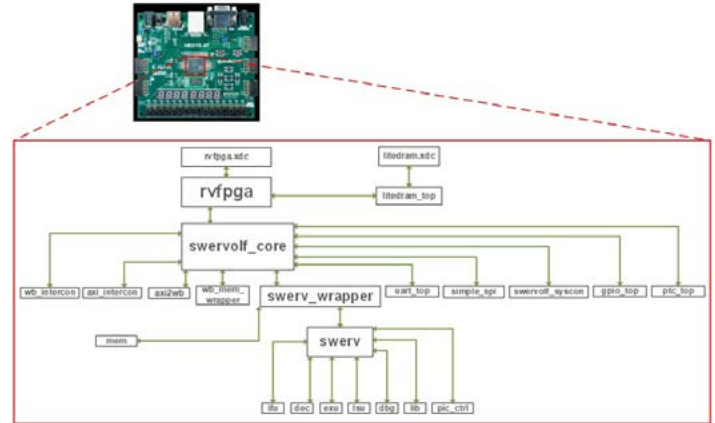


Figure 65: SweRVolf Project Hierarchy inside the Nexys A7. Image Source: Imagination Technologies UP

6.10 Enhancing the SweRVolf Implementation

The original SweRVolf implementation can be extended by adding additional peripherals.

6.10.1 Peripherals Available from Opencores

SweRVolf is an open specification and all the source code is available for download from Github. This means it can be modified and new peripherals can be added. A whole range of projects are available from www.opencores.org/projects.

6.10.2 Peripherals Added in Laboratories

In the laboratories, the following peripherals have been added for the Nexys A7 board:

- 5 pushbuttons through new GPIO
- Tri-color LEDs using PWM
- Temperature sensor using I2C
- 7-Segment display driver
- SPI Accelerometer driver

7 Soft Core Software Installation

7.1 Downloading RVfpga

Before installing the software for soft core development, the RVfpga package needs to be downloaded as it contains many of the files required. See Section 6.1 for details.

Download all the files. A good location for the RVfpga package is in the home directory e.g. `/home/myusername/RVfpga`, where "myusername" is the actual computer name. When an example states:

```
[RVfpgapath]/RVfpga/
```

Then all that needs to be entered is:

```
~/RVfpga/
```

Make sure that all the files have read/write permissions.

The examples that follow use the current version of RVfpga, which may change. It is recommended that the Imagination Technologies “RVfpga: Getting Started Guide” be referenced for the latest information.

7.2 About Xilinx Vivado

This is by far the most time-consuming operation as there are several programs that need to be set up in Ubuntu 18.04 Linux. Support for Windows and Mac is currently in development. The Xilinx download and installation takes around 2½ hours.

Xilinx has been chosen for several reasons:

1. Xilinx are at the top end of the FPGA market with a very large range of high specification products.
2. From the career perspective and marketable skill, Xilinx is the most widely asked for FPGA technology. A quick search of a jobs website will confirm this.
3. SweRVolf was written using Xilinx.
4. SiFive also use Xilinx for their cores.

7.3 Important Information about Xilinx Vivado Installation in Linux

Before performing the Xilinx download, if not already done, Ubuntu 18.04 needs to be setup on the computer. The Ubuntu partition needs to be at least 100 GB. **Do not use Ubuntu 20.04.**

Xilinx specify which versions of Ubuntu are compatible with their software and Ubuntu 20.04 is not currently one of them.

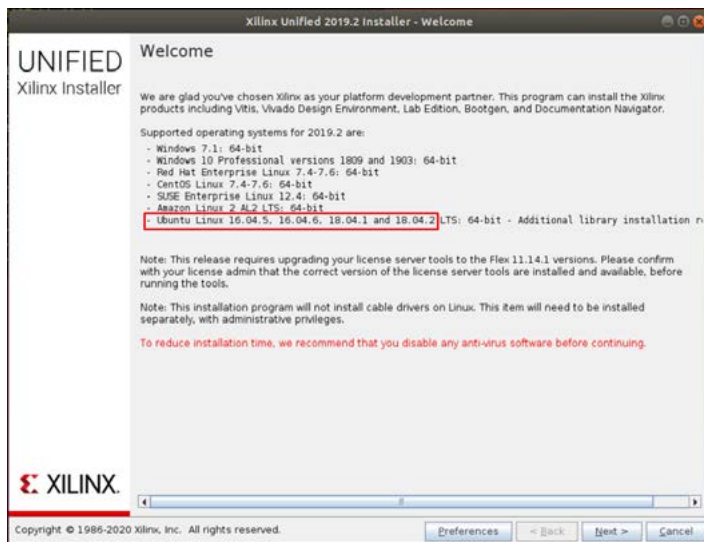


Figure 66: Xilinx Unified Installer Compatibility

Do not be like the author and download Ubuntu 20.04 only to find out 2½ hours later that it is incompatible and does not run. The process will need to be started from the beginning by deleting everything then installing the correct version Ubuntu 18.04.

7.3.1 Xilinx Vivado System Requirements

Xilinx Vivado is 16 GB in size (hence the long download time) and requires 56 GB to do the install (hence the 100 GB partition size). Should there be insufficient disk space, an error will be generated.

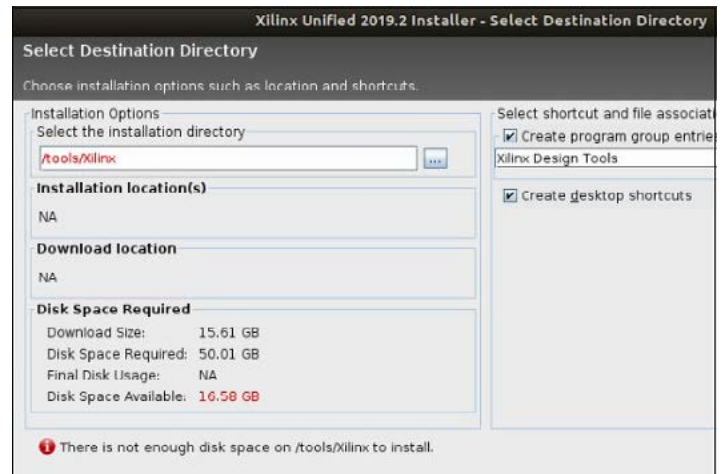


Figure 67: Xilinx Vivado System Requirements

7.4 Installing Xilinx Vivado in Linux

This program is used to modify and build the Verilog .v and System Verilog .sv files for the soft-core and download them to the Diligent Nexys A7 board.

If a Diligent Nexys A7 board is not available or just to run a computer simulation only, then Vivado can be installed later.

7.4.1 Downloading the Xilinx Software

In order to download Xilinx software, a Xilinx account will need to be created, if one does not already exist.

Go to <https://www.xilinx.com/support/download.html> and select the downloads for 2019.2

Click on Xilinx Unified Installer 2019.2: Linux Self Extracting Web Installer to download it.

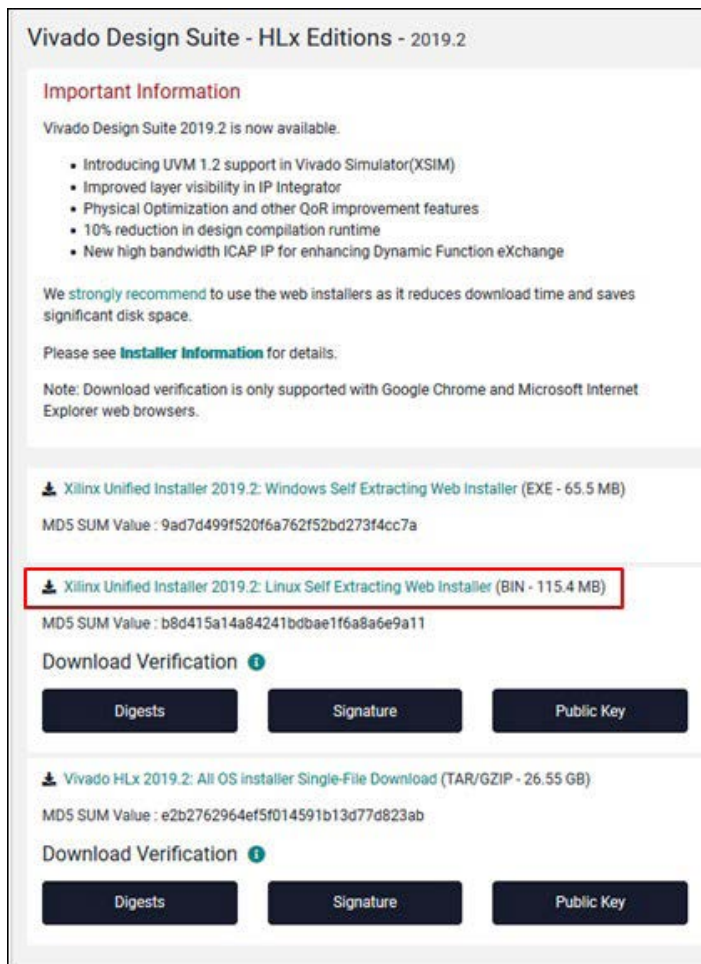


Figure 68: Xilinx Unified Installer 2019.2

7.4.2 Changing permissions of the Binary File

When the Xilinx_Unified_2019.2_1106_2127_Lin64.bin file is downloaded to the /Downloads directory, it is not executable.

Make it executable with the *Ubuntu Files* tool by right clicking on the file name, then selecting *Properties-Permissions*. Click on *Allow executing file as program*.

7.4.3 Running the Binary File

Open an *Ubuntu Terminal* and make it the root by typing the following:

```
sudo su
```

Go back to the *Ubuntu Files* tool and drag and drop the following file into the *Terminal* window:

```
Xilinx_Unified_2019.2_1106_2127_Lin64.bin
```

Press carriage return in the *Ubuntu Terminal* to run the binary file.

7.4.4 Selecting Vivado Product to Install

In the *Select Product to Install* window, select *Vivado*, not *Vitis*.

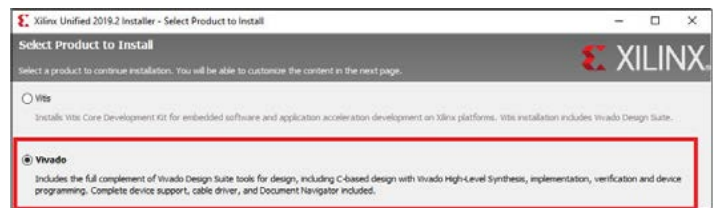


Figure 69: Selecting Vivado

7.4.5 Selecting WebPACK

In the *Select Edition to Install* window, select *Vivado HL WebPACK*, which is the free version.

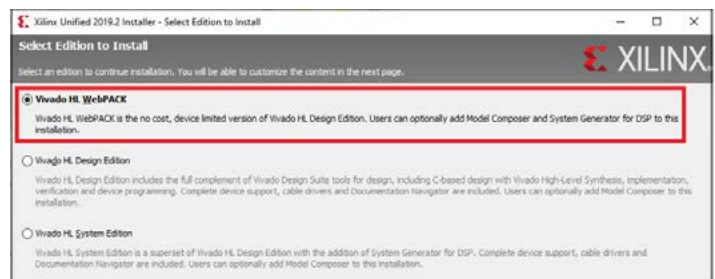


Figure 70: Select Edition to Install

Follow the remaining steps to do the installation and use the defaults.

7.4.6 Installing the Cable Drivers for the Digilent Nexys A7 Board

Open an *Ubuntu Terminal* and navigate to:

```
/tools/Xilinx/Vivado/2019.2/data/xicom/cable_drivers/lin64/install_script/install_drivers/
```

Then type:

```
sudo ./install_drivers
```

7.4.7 Installing the Board Files

Go to <https://github.com/Digilent/vivado-boards>.

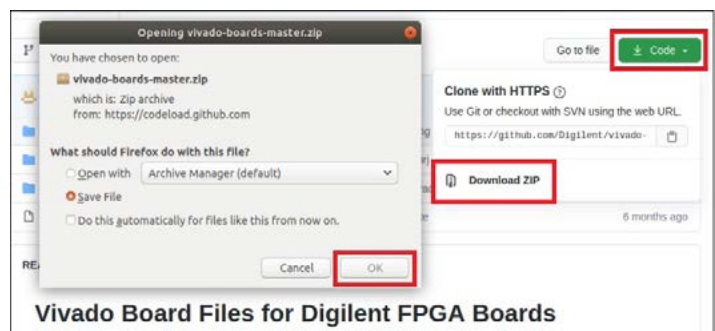


Figure 71: Download Vivado Boards Master

Download the .zip file and unzip it using the *Ubuntu Files* tool.

In an *Ubuntu Terminal* navigate to:


```
/Downloads/vivado-boards-master/new/board_files
```

The files here need to be copied to the Vivado `board_files` directory. Type:

```
sudo cp -r *
/tools/Xilinx/Vivado/2019.2/data/boards/
board_files
```

The important board file is `nexys-A7-100t`.

7.4.8 Running Vivado under Linux

Open an *Ubuntu Terminal* .

Navigate to the directory:

```
/tools/Xilinx/Vivado/2019.2 then type:
```

```
source settings64.sh
vivado &
```

7.5 Installing Visual Studio Code and PlatformIO

Time required: about 20 minutes.

Visual Studio Code (VSCode) is the IDE for software development and debugging.

PlatformIO is an add-on for VSCode that provides support for RISC-V and many other processors.

7.5.1 Download Visual Studio Code

Go to <https://code.visualstudio.com/Download>. Click on `.deb` to download the file. The file size is currently 60.5MB

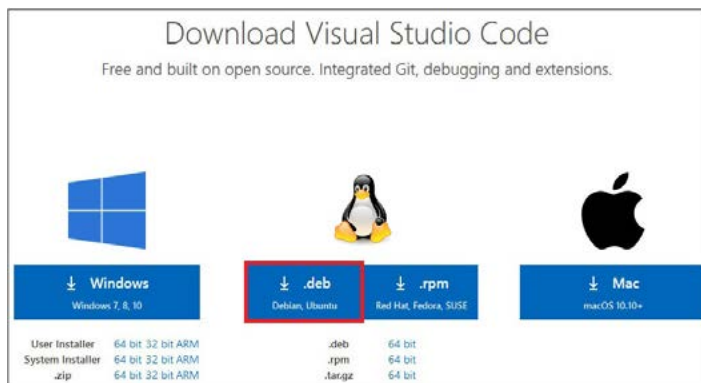


Figure 72: Download Visual Studio Code

7.5.2 Extracting Visual Studio Code

Open an *Ubuntu Terminal*  and type:


```
cd ~/Downloads
sudo dpkg -i code*.deb
```

To run VSCode type:

```
code
```

7.5.3 Installing PlatformIO

PlatformIO provides specific support for RISC-V.

In the *Ubuntu Terminal*  type the following to load some necessary utilities:

```
sudo apt install -y python3-distutils python3-venv
```

If VSCode is not already running, locate it by typing *Visual Studio Code* in the search menu then selecting it. Alternately, click on the *Visual Studio Code* icon.

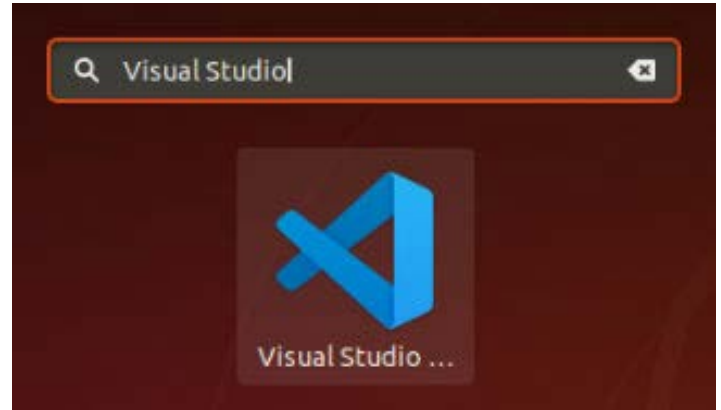



Figure 73: Visual Studio Code Icon

7.5.4 PlatformIO extension in VS Code

In VSCode, click on the Extensions icon .

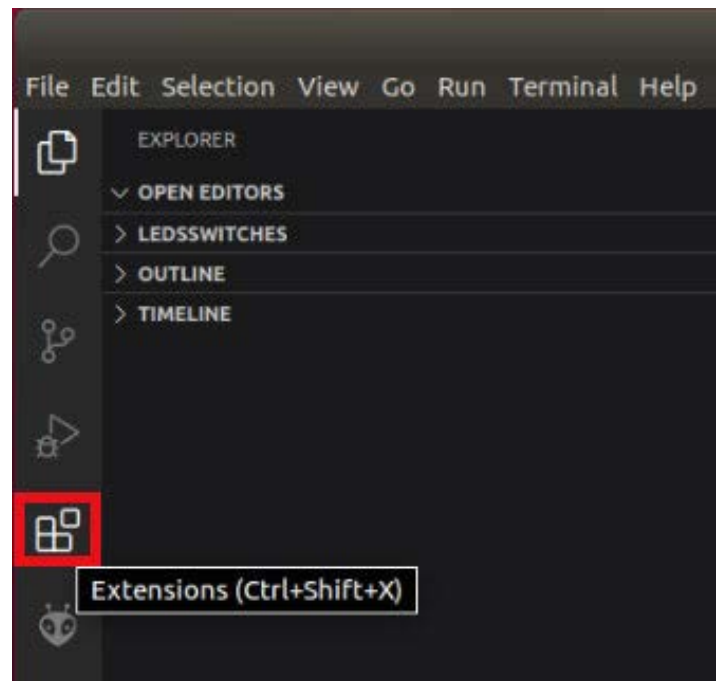


Figure 74: VSCode Extensions Bar

7.5.5 Installing PlatformIO Extension

Type `platformio` into the search box. Go to *PlatformIO IDE* then click on the *Install* button.

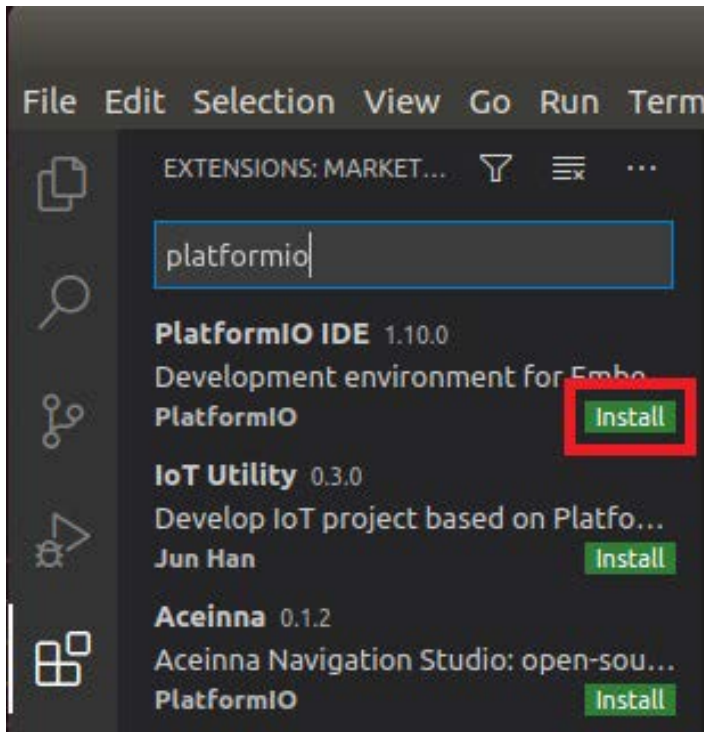


Figure 75: PlatformIO IDE Extension

7.5.6 Reloading PlatformIO

The *OUTPUT* window on the bottom gives information as to progress of the installation.

When finished, click on *Reload Now* so that PlatformIO Core will be installed in VSCode.

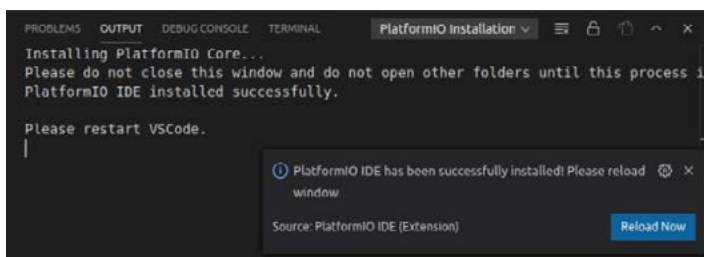



Figure 76: Reload Now after PlatformIO installs

7.5.7 Installing Chips Alliance Packages

The next step is to select the processor target. In this case it will be SweRVolf that is part of the RVfpga package that has been downloaded.

View the Quick Access menu by clicking on the Platformio  button (located on the left hand side bar) and then click on the *PlatformIO Core CLI* option.

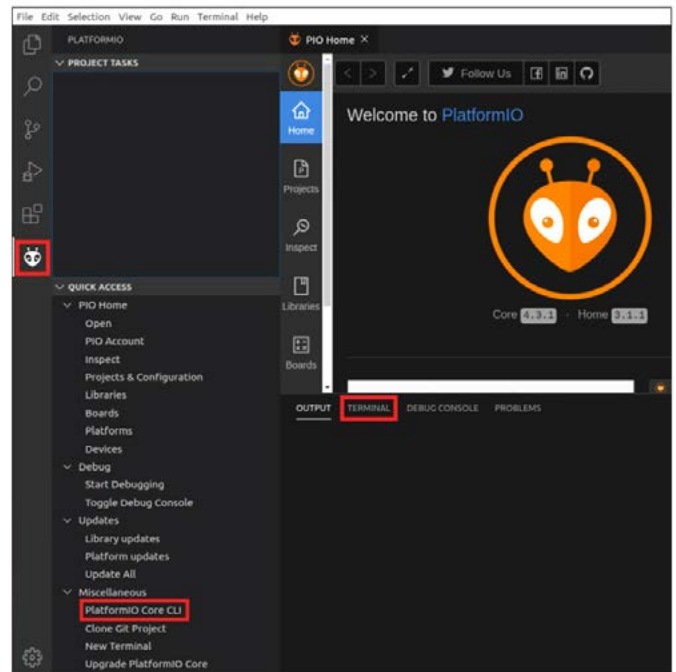


Figure 77: Selecting the PlatformIO Core Target

On the bottom window, change from the *OUTPUT* window to the *TERMINAL* window. This allows commands to be entered while in VSCode.

7.5.8 Installing Chips Alliance Platform

Install the Chipsalliance platform by running the following command on the *TERMINAL*:

```
~/platformio/penv/bin/pio platform install
[RVfpgaPath]/RVfpga/platform-chipsalliance
--with-all-packages
```

Where [RVfpgaPath] is the location where the RVfpga files were saved. For example, it might be in /home/myname/, where myname is the actual computer name.

The package includes the pre-built RISC-V toolchain, OpenOCD debugger for RISC-V, Verilator, Whisper, JavaScript and Python scripts. The RVfpga bit file and example code are also provided.

7.5.9 Optional Check that SweRVolf is Loaded

To confirm that swervolf_nexys is loaded, in the *TERMINAL* window type:

```
~/platformio/penv/bin/pio boards | more
```

then scroll down to swervolf_nexys

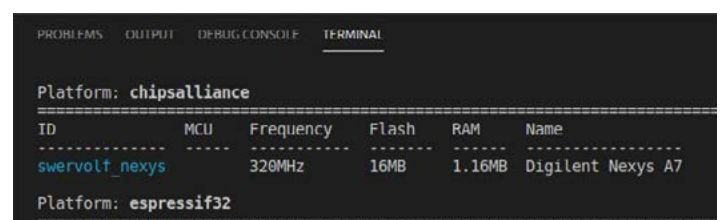



Figure 78: Check swervolf_nexys is loaded

7.5.10 Close and Reopen VSCode

So that that PlatformIO is activated, close and reopen VSCode.

Note: Normally the `.platformio` files are hidden.

To search for `.platformio` in an *Ubuntu Terminal* , type:

```
ls -a
```

This makes the hidden files visible.

Alternately, use the *Ubuntu Files* tool  to show hidden files.

7.6 Installing Verilator and GTKWave

Time required: about 20 minutes.

Verilator is a hardware simulator for Verilog and System Verilog files that is used by PlatformIO.

To install Verilator to run natively in Linux, open an *Ubuntu Terminal* .

Start by installing the necessary utilities that are not installed by default:

```
sudo apt-get install git make autoconf g++  
flex bison libfl2 libfl-dev
```

Install Verilator program

```
git clone https://git.veripool.org/git/verilator  
cd verilator  
git pull  
git checkout v4.020  
autoconf  
./configure  
make  
sudo make install
```

Finally install the waveform viewer

```
sudo apt-get install -y gtkwave
```



Figure 79: GTK Wave ICON

7.6.1 A Typical Verilator Waveform

Verilator is complicated to use as a hardware simulator and is outside of the scope of this article. Full details are given in the “RVfpga: Getting Started Guide”. However, a typical waveform of the clock and the Instruction Fetch Unit (IFU) is:

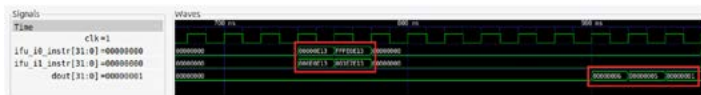


Figure 80: Typical Verilator Waveform.

7.7 Running a C Program in PlatformIO using Whisper

Western Digital has an Instruction Set Simulator (ISS) called Whisper, developed for the software verification of the SweRV RISC-V core, without any hardware. It is part of the RVfpga download.

7.7.1 Opening the Folder

Open VSCode and go into PlatformIO.

On the top menu bar of PlatformIO, click on *File* then *Open Folder*. There is no need to actually open the files.

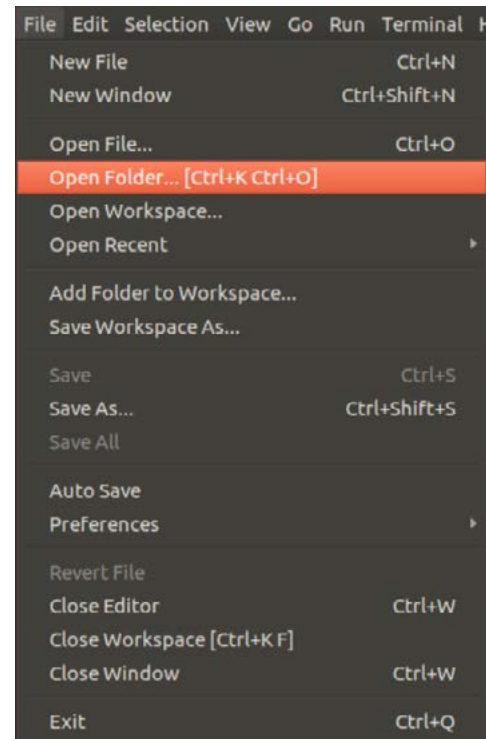


Figure 81: Opening Folder to Simulation File

7.7.2 Vector Sorting Whisper Example Code

Select the directory `VectorSorting` and click on *OK*.

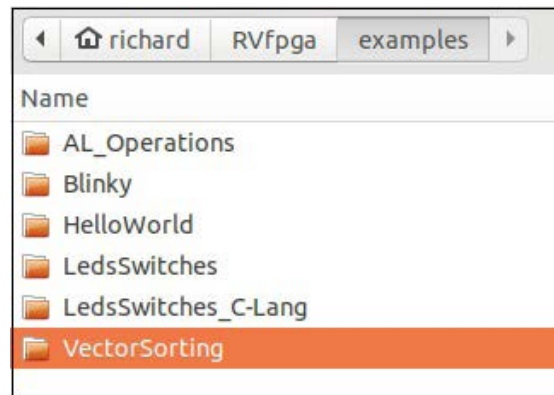


Figure 82: VectorSorting Folder

7.7.3 Modifying platformio.ini

To use Whisper, it is necessary to modify the `platformio.ini` file. Open this file.

Uncomment the line:

```
debug_tool = whisper
```

then save it.

Important: make sure that `debug_tool` lines up with the characters above and that there are no leading spaces. Figure 83 shows the error generated when an unexpected space is present.

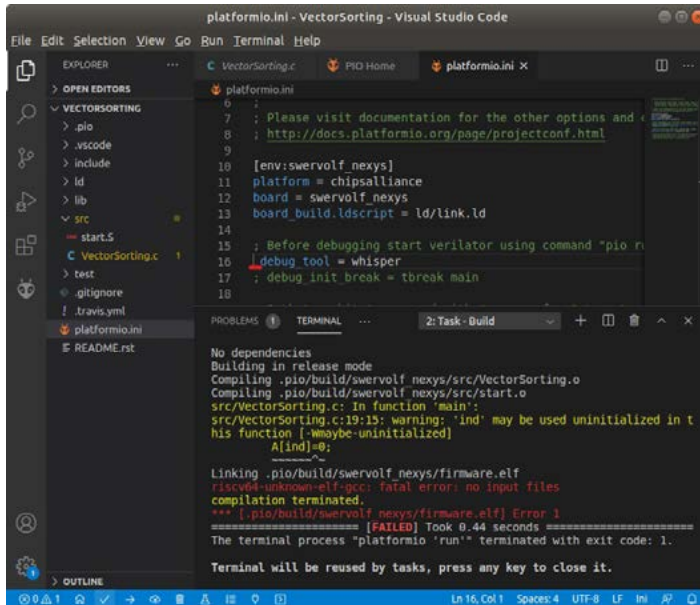


Figure 83: Incorrect platformio.ini Setup

7.7.4 Running a Simulation using Whisper

Place a breakpoint by clicking to the left of line 10.

Launch the debugger by clicking on and then on .

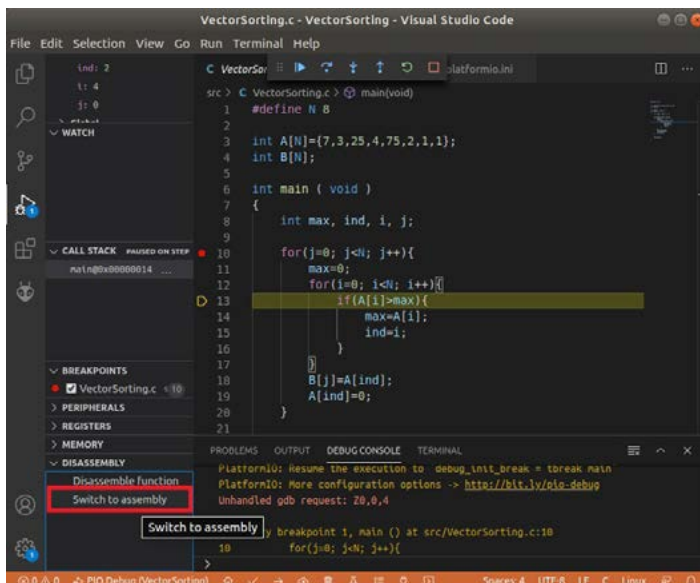


Figure 84: Showing the Program RISC-V Instructions and breakpoint at line 10

It is now possible to single step through the code using the keyboard Function Key F10.

Click on *Disassembly* on the bottom left hand corner of the screen to show the RISC-V instructions.

7.7.5 Viewing the RISC-V Assembly Language

The assembly language that has been generated by the C compiler is shown in Figure 85.

To return to the C source code, click on *Switch to code*.

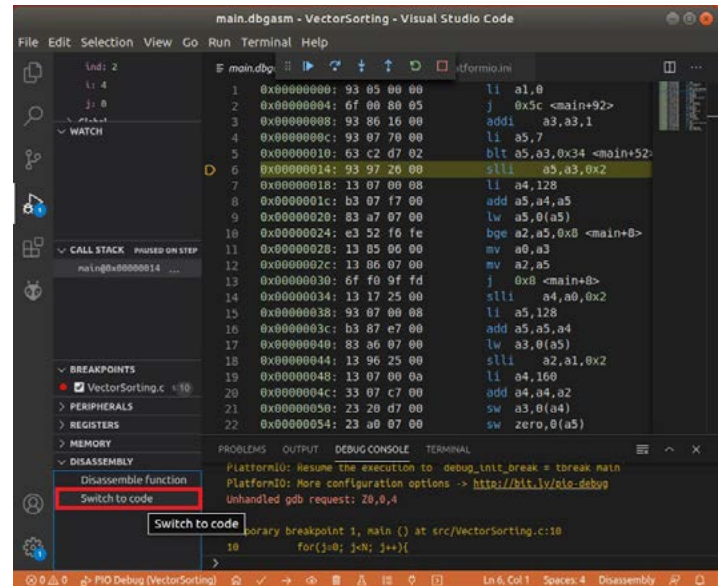


Figure 85: RISC-V Assembly Language View

For more details on debugging, see the “RVfpga: Getting Started Guide”.

8 Building the SweRVolf Core and Downloading it to a FPGA

Time Required: about 5 minutes

8.1 Building the SweRVolf Core

The good news is that this part is not needed, thanks to the RVfpga Team at University Complutense, Madrid. As part of the RVfpga package, a Xilinx Vivado project has been provided with the file `RVfpga.bit` that can be directly downloaded to the Digilent Nexys A7 hardware.

8.2 Connect the Nexys A7 Board

Plug in the Nexys4 A7 board using the USB cable supplied. Slide the Power switch to the on position as the default from the factory is off.

8.3 Downloading the Core to the Digilent Nexys A7

This requires the RVfpga package to be downloaded first as it contains the `RVfpga.bit` file. See Section 6.1.

8.3.1 Open the Hardware Manager in Vivado

Open Vivado as shown in Section 7.4.8.

Click on *Open Hardware Manager* >

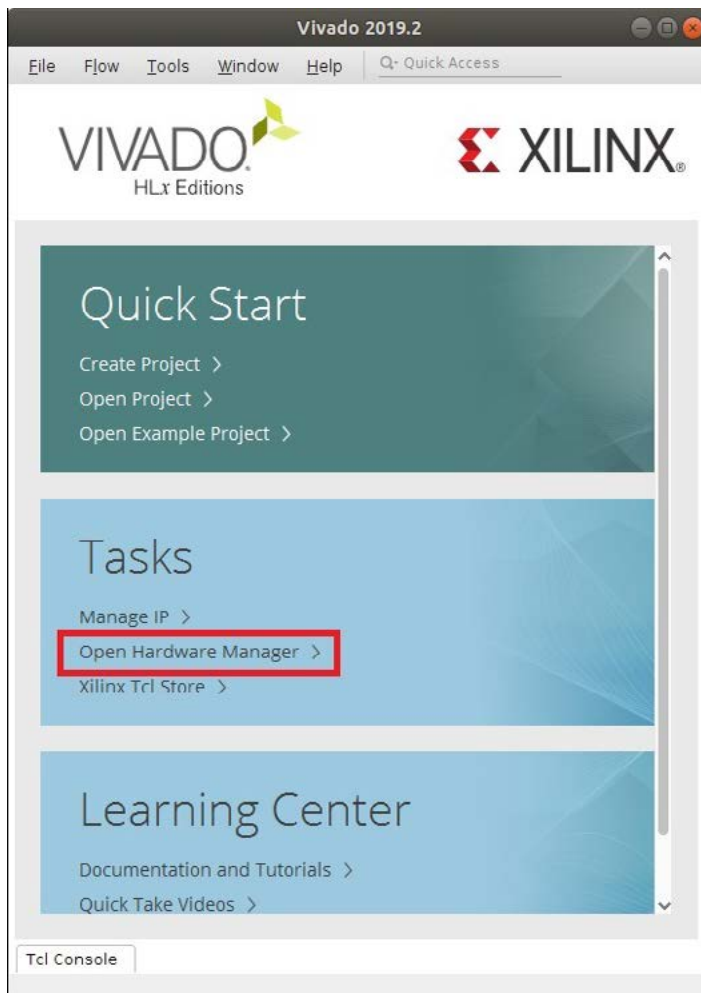


Figure 86: Open Vivado Hardware Manager

8.3.2 Open the Hardware Target

The hardware manager will automatically detect that the hardware is not open. Click on *Open Target* and then *Auto-detect*.

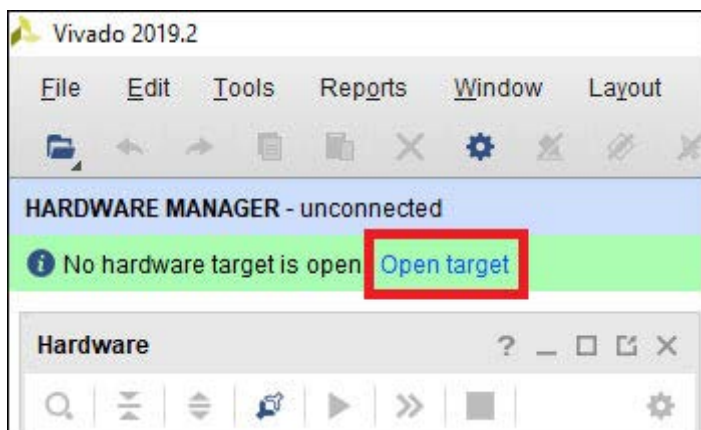


Figure 87: Open Hardware Target

8.3.3 Programming the FPGA Configuration

Click on *Program device*.

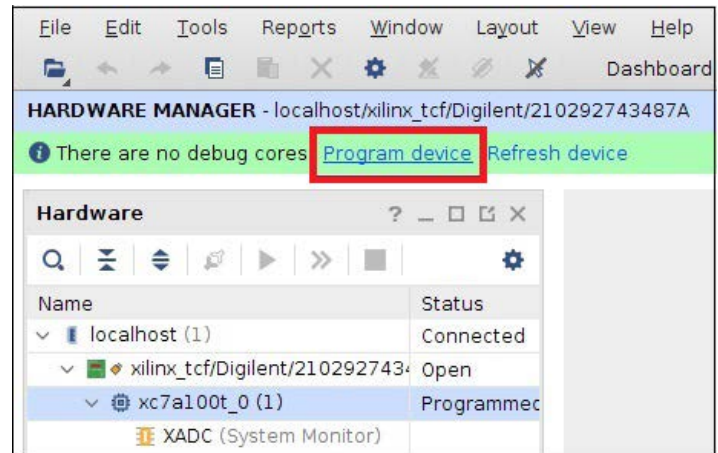


Figure 88: Program FPGA Configuration

8.3.4 Select Bitstream File

For the Bitstream file, select:

[RVfpgaPath] /RVfpga/src/RVfpga.bit

The field *Debug probes file* can be left blank. Click on *Program*. The board should program in about 10 seconds.



Figure 89: Download Bitstream File

8.3.5 Close Vivado

This is an important step. Come out of Vivado so the USB port is free, otherwise it is not possible to download and run the program using VSCode.

Click on the *Close Hardware Manager* icon  on the right hand side.

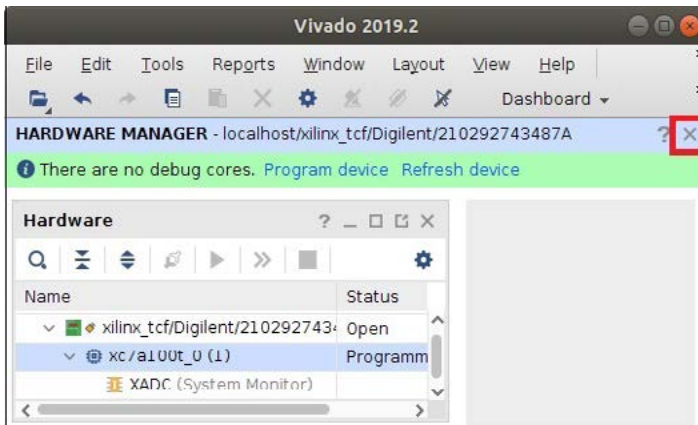


Figure 90: Exit Vivado

8.4 Running a Program on the NexysA7 Board

Time Required: about 5 minutes.

8.4.1 Opening the Program to be Run

Open VSCode by typing Visual Studio Code into the search bar and clicking on the VSCode icon.

On the top menu bar click on *File* then *Open Folder*.

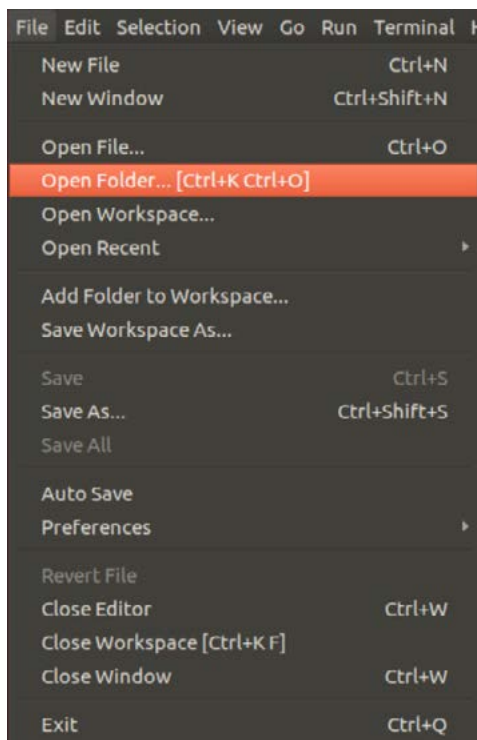


Figure 91: Open Folder

It is not necessary to open the files.

8.4.2 Select Program LedsSwitches

Navigate to `RVfpga/examples` then click on `LedsSwitches`. Other examples may be tried later.

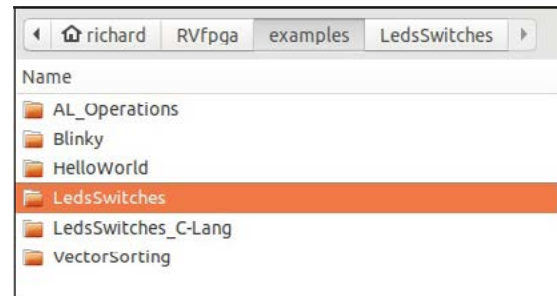


Figure 92: Leds and Switches Program

8.4.3 LEDs and Switches Program Opened

This is a minimal assembly language program which continually reads the 16 switches then outputs the value to the 16 LEDs. There is also a C code version in the RVfpga examples.

Expand `>src` then click on `LedsSwitches.S` so that the assembly code is visible.

Click on the *PlatformIO Build* icon  on the bottom left of the screen.

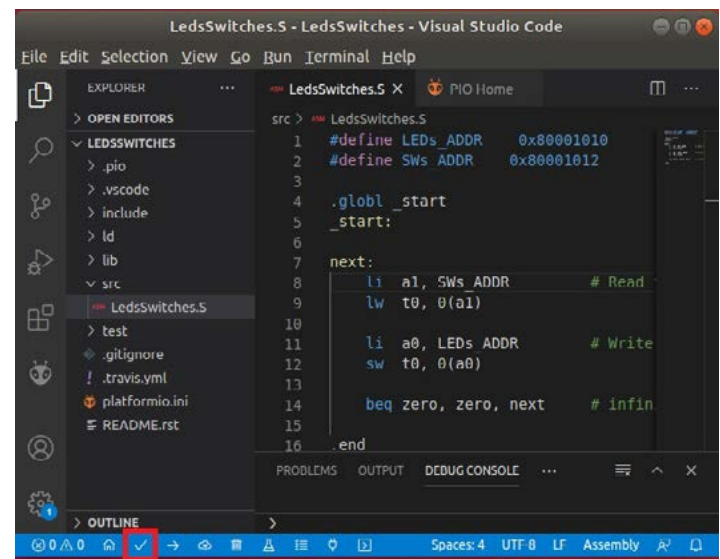


Figure 93: Build LedsSwitches Program

8.4.4 Running the LedsSwitches Program

Either from the toolbar select *Run* then *Start Debugging* or press the keyboard Function Key F5. The program will run continuously.

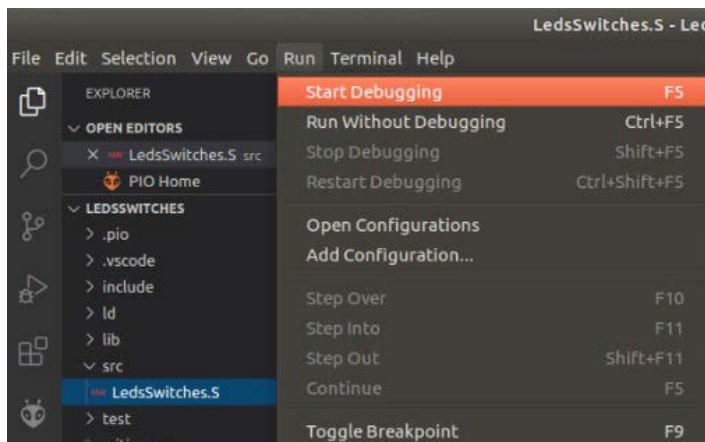


Figure 94: Start Debugging LEDs and Switches Program

8.4.5 Nexys A7 Board Running RISC-V Core - LEDs Controlled by Switches

Operate the slide switches at the bottom of the board. The green LEDs can be turned on and off individually.

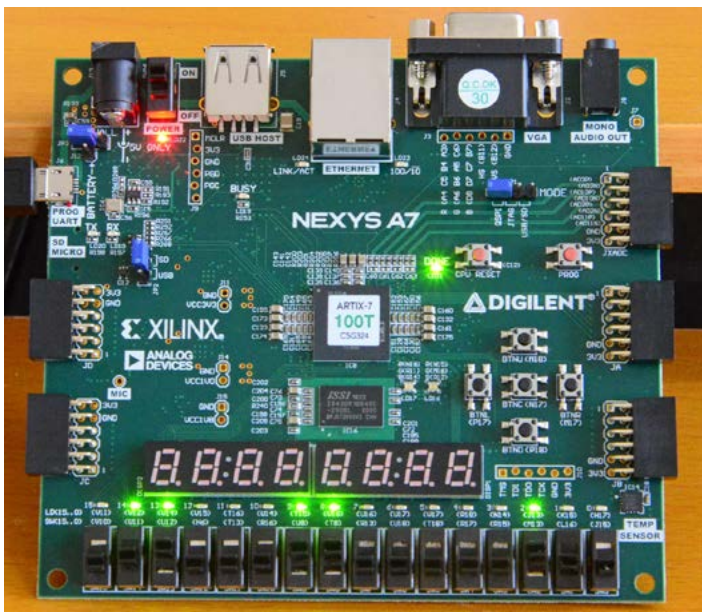


Figure 95: Nexys A7 Board Running RISC-V Core

Note: on the latest version of `RVfpga.bit`, the 7-Segment display shows 0000 0000. A simpler version is shown here for clarity.

9 Product Development using RISC-V

In this guide, two SoC RISC-V processors have been used for the hands-on sessions, the Kendryte K210 and the SiFive FE310-G002. This section looks at what these two devices can offer when built into a user's board, either for further development or for a mass-produced product.

9.1 Kendryte K210

This is used on the Seeed Technologies Co Ltd Maix BiT board. It is a very powerful processor at a modest cost.

9.1.1 Kendryte K210 Processor Core

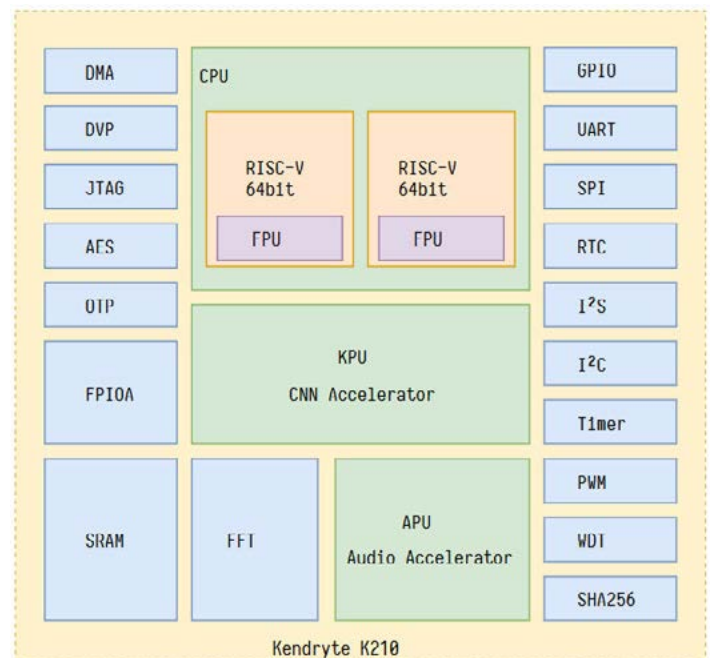


Figure 96: Kendryte K210 Architecture Overview. Image Source: Kendryte K210 Datasheet.

It contains not one but two RISC-V 64bit floating point processors running at 400 MHz. In addition, there are a range of useful internal components and interfaces.

Abbreviation	Description
CNN	Neural Network
DMA	Direct Memory Address
DVP	Digital Video Port
JTAG	Programming and Debug Interface
AES	Advanced Encryption Standard
OTP	One Time Programmable Flash
FPIOA	Field Programmable Input and Output Array (can map GPIO to any pin)
SRAM	Static Random Access Memory
FFT	Fast Fourier Transform
GPIO	General Purpose Input / Output 3V3 and 1V8

Abbreviation	Description
UART	Serial Port
SPI	Serial Peripheral Interface
RTC	Real Time Clock
I2S	Inter-integrated sound. Standard interface for audio codecs
I2C	Inter-integrated Circuit bus for low speed peripherals
PWM	Pulse Width Module
WDT	Watch Dog Timer
SHA256	Secure Hash Algorithm to verify data integrity

Table 8: Kendryte K210 Abbreviations

Having the GPIO being dual 3V3 / 1V8 is very useful. Some radio modules such as LTE/GSM modules support 1V8 operation only. Without dual voltage GPIO, additional level converters would be required at extra cost.

A Field Programmable Input / Output Array (FPIOA) makes the printed circuit board (PCB) layout easier as pins can be grouped for easy routing.

The Kendryte K210 processor is only available in a Ball Grid Array (BGA) package, making it suitable for machine placement only.

9.1.2 Seeed Technologies MaixPy BiT Implementation

Alternatively, for experimentation or low-volume production, the basic MaixPy BiT could easily be fitted to a breadboard or soldered onto another PCB using the mounting pins.

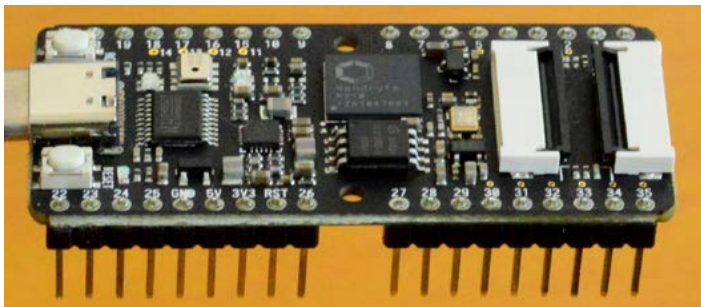


Figure 97: Maix BiT Board with Mounting Pins. Seeed Technologies Maix-I Modules

9.1.3 Maix-I

A problem when implementing any processor on a PCB is that there are quite a large number of additional components required to make the processor work – decoupling capacitors, crystals, DC-DC converters, inductors, reset circuit etc.

Two Maix-I modules are available to make life easier for the hardware engineer:

[MAIX-I](#) Digi-Key 1597-1717-ND \$8.91 and

[Maix-I with Wi-Fi](#) Digi-Key 1597-1715-ND \$10.82.

For low volume production or early development boards, these are an attractive option.

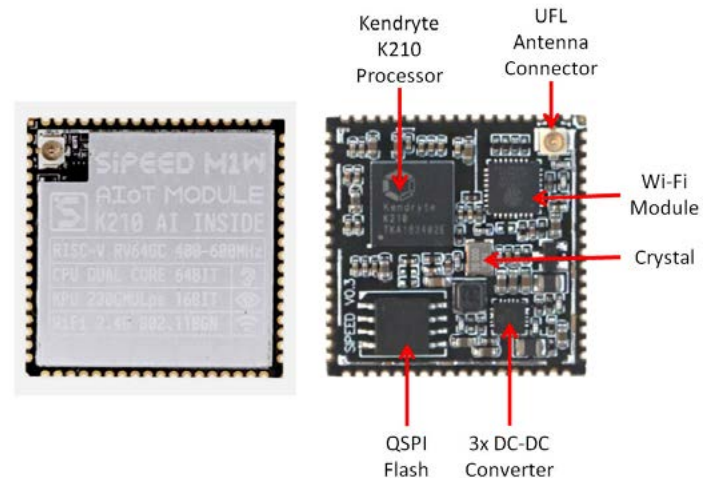


Figure 98: Maix-I Module with Wi-Fi. Image Source: Seeed Technologies Co Ltd (modified).

Both Maix-I modules contain not only the Kendryte K210 processor, but also Quad SPI flash, crystal, DC-DC converters and optionally for little more cost, a Wi-Fi module.

A possible application would be a dedicated graphics processor driven from a main processor via a simple serial port. Because the project can be developed in MicroPython, the software development time would be short.

9.1.4 Visual Studio Code support for MaixPy BiT

The MaixPy BiT uses MicroPython for programming, but C or C++ are also an option.

PlatformIO, as used for the Xilinx FPGA soft core, also supports the Kendryte K210 used on the MaixPy BiT board and Maix-I modules. However, the author has not tried it.

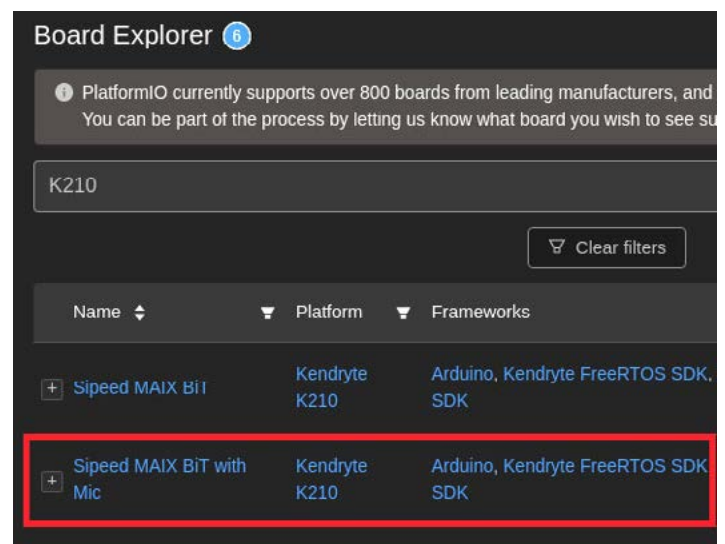


Figure 99: VSC PlatformIO Support for MaixPy BiT

9.2 SiFive

The processor used on the RED-V Redboard is the SiFive Freedom Everywhere FE310-G002.

9.2.1 Freedom Everywhere FE310-G002 RISC-V Pinout

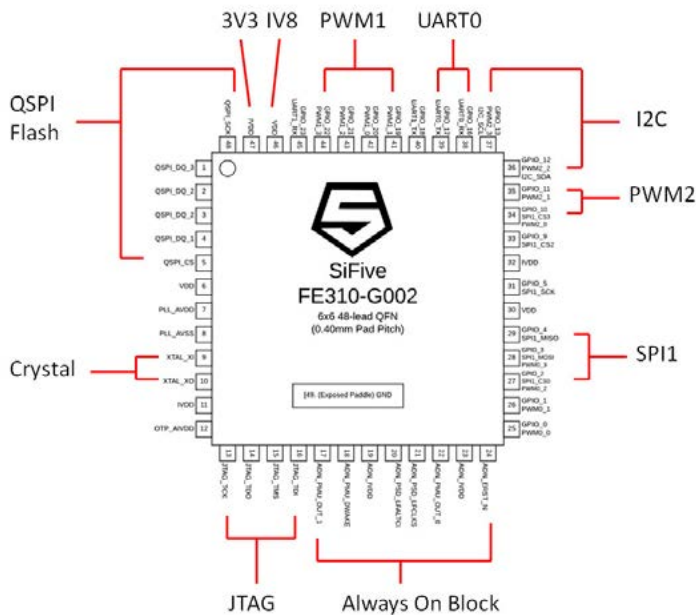


Figure 100: SiFive FE310-G002 Pinout. Image Source: SiFive FE310-G002 datasheet (modified).

The FE310-G002 is available in a 48 pin QFN Package and runs up to 320 MHz.

9.2.2 FE310-G002 GPIO Pins Multiplexed Functions

Name	Pin	GPIO	PWM	SPI	UART	I ² C
GPIO.0	25	0 I/O	PWM0.0 0			
GPIO.1	26	1 I/O	PWM0.1 0			
GPIO.2	27	2 I/O	PWM0.2 0	SPI1.SS0		
GPIO.3	28	3 I/O	PWM0.3 0	SPI1.MOSI		
GPIO.4	29	4 I/O		SPI1.MISO		
GPIO.5	31	5 I/O		SPI1.SCK		
GPIO.9	33	9 I/O		SPI1.SS2		
GPIO.10	34	10 I/O	PWM2.0 0	SPI1.SS3		
GPIO.11	35	11 I/O	PWM2.1 0			
GPIO.12	36	12 I/O	PWM2.2 0			I2C0.SDA
GPIO.13	37	13 I/O	PWM2.3 0			I2C0.SCL
GPIO.16	38	16 I/O			UART0.RX I	
GPIO.17	39	17 I/O			UART0.TX 0	
GPIO.18	40	18 I/O			UART1.RX I	
GPIO.19	41	19 I/O	PWM1.1 0			
GPIO.20	42	20 I/O	PWM1.0 0			
GPIO.21	43	21 I/O	PWM1.2 0			
GPIO.22	44	22 I/O	PWM1.3 0			
GPIO.23	45	23 I/O			UART1.RX I	

Table 9: SiFive Port Pins Secondary Functions. Image Source: SiFive FE310-G002 datasheet.

The GPIO pins are multiplexed, which means that there can be up to 3xPWM, 1xSPI, 1xI2C and 2xUART. Note that there is no Analog to Digital Converter (ADC). Because it is a QFN48 package, it is difficult to solder (or de-solder) by hand, so it is best-suited to machine placement.

9.2.3 SparkFun Electronics RED-V SiFive Thing Plus Implementation

For experimentation or low-volume production, the RED-V SIFIVE RISC-V THING PLUS could also be used. The cost for the 1568-DEV-15799-ND is \$29.95. Connecting pins would need to be added. Schematics are available on the SparkFun website.

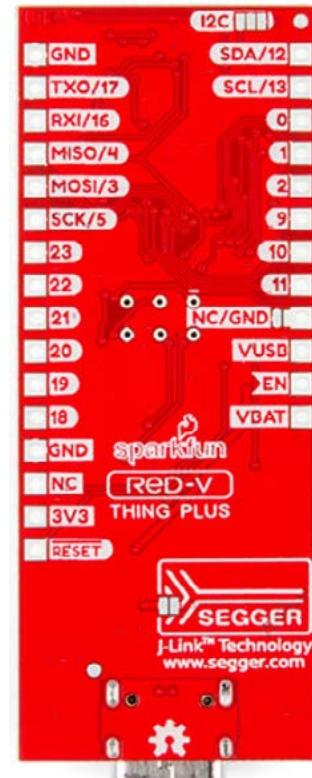


Figure 101: SparkFun Electronics SiFive Thing Plus Bottom View. Image Source: SparkFun Electronics

10 More Information on RISC-V

For more information on RISC-V, the proceedings of the RISC-V Summits are available on YouTube and the RISC-V.org website. Both the inventors of RISC-V David Patterson and Krste Asanovic are entertaining speakers. There are also videos of the Maix Bit, SparkFun and Digilent boards on YouTube.

For those readers wishing to build on and gain further knowledge after running the SweRV core in the FPGA described here, the Imagination Technologies "RVfpga: The Complete Course in Understanding Computer Architecture" to be released in November 2020 may well be of interest. This consists of 20 Laboratories in 4 parts, some of Part 1 having been covered in this guide:

Part 1: Vivado Project and Programming – Vivado and Verilator, C programming, Whisper, RISC-V Application Binary Interface (ABI), procedure calling convention, merging C and assembly code.

Part 2: I/O Systems – driving the 7-segment display on the Digilent Nexys A7 board, interrupts, timers and serial buses (SPI, I2C and UART).

This will be followed in Q3 2021 by:

Part 3: RISC-V Core – understanding the core structure, pipeline, hazards and implementing new instructions

Part 4: RISC-V Memory Systems – understanding the cache controller, cache hits and misses, modifying the cache, understanding ICCM (instruction closely coupled memory) and DCCM (data closely coupled memory).

Details of two relevant books on RISC-V are also given in the References section.

11 Conclusion

Having looked at the different boards for RISC-V, the MaixPy BiT proved to be the easiest to use and the most fun. It is interesting that the lowest-cost board offers the most powerful computing power in the form of a dual-core RISC-V processor with camera and LCD at a very modest price. It is a really good tool for learning about machine vision and face recognition. Because it is programmed using MicroPython, it is easy to use, even for those without a programming background.

The SparkFun Electronics RED-V Thing Plus and Red Board fit well into the hobbyist category or for those electronics/software engineers wanting to learn RISC-V programming in C or assembly language, possibly with a view to developing their own product. The software tools from SiFive are easy to use and there are lots of useful examples that can be adapted to other applications. Documentation from SiFive is also easy to read. Users of Eclipse tools for ARM or TI processors should have no difficulty with this.

In this guide to RISC-V, it has only been possible to give a short introduction to implementing a soft core in a Xilinx FPGA. This is aimed at students studying computer architecture and engineers working on major projects. It is a complex business and is not recommended without detailed instructions. Having never used a soft core before, but by carefully following the Imagination Technologies “RVfpga: Getting Started Guide” and using the RVfpga package, it was possible to successfully run a RISC-V core on a Nexys A7 board in a day. Of the four boards, this is by far the most technical. As a follow on activity for those readers who wish to stretch themselves intellectually, there is the Imagination Technologies “RVfpga: Complete Course in Understanding Computer Architecture”.

A lot of effort has been put into RISC-V and only time will tell how effective it will be as a competitor to ARM. The open source ISA and no royalties most certainly give RISC-V a competitive edge, with some very powerful processors available at a low cost.

12 Acknowledgements

Many thanks to Professor Daniel Chaver Martinez at Madrid University who allowed the author to use his RISC-V teaching material and his student Daniel Gonzalez who allowed the author to use his Master's Thesis. Without these, it would not have been possible to write this guide.

Finally, special thanks to Robert Owen of Imagination Technologies who conceived the idea for this guide and made it happen.

13 Author Profile

Richard Sikora has been a hardware and software engineer for more than 30 years, having worked on more than 30 different processors and digital signal processors (DSPs). Over the years he has designed weighing machines, process control instrumentation, aircraft ignition, vending machines, smartcard readers, smoke and carbon monoxide monitors. He is also the author of several teaching kits for Texas Instruments DSPs and Matlab.

14 References

Embedded Microprocessor Benchmark Consortium

<https://www.eembc.org/coremark/>

RISC-V Logo

<https://riscv.org/about/risc-v-branding-guidelines/>

RISC-V Specifications

<https://riscv.org/technical/specifications/>

Other cores

<https://chipsalliance.org>

Kendryte K210 Datasheet

https://s3.cn-north-1.amazonaws.com.cn/dl.kendryte.com/documents/kendryte_datasheet_20181011163248_en.pdf

Wishbone Bus

<http://indico.ictp.it/event/a11204/session/35/contribution/22/material/0/0.pdf>

<https://venturebeat.com/2019/12/11/risc-v-grows-globally-as-an-alternative-to-arm-and-its-license-fees/>

The SweRV core download:

<https://github.com/chipsalliance/Cores-SweRV>

The SweRV Programmer's Reference Manual

https://github.com/chipsalliance/Cores-SweRV/blob/master/docs/RISC-V_SweRV_EH1_PRM.pdf

SweRVolf:

<https://github.com/chipsalliance/Cores-SweRVolf>

PlatformIO:

<https://github.com/platformio/platformio-core>

Verilator:

<https://github.com/verilator/verilator>

Whisper:

<https://github.com/westerndigitalcorporation/swerv-ISS>

DANIEL LEÓN GONZÁLEZ, FPGA IMPLEMENTATION OF AN AD-HOC RISC-V SYSTEM-ON-CHIP FOR INDUSTRIAL IOT, Master's Degree Thesis, Universidad Complutense, Madrid, July 2020.

Digital Design & Computer Architecture, Sarah Harris & David Money Harris, second edition (RISC-V Edition due 2021)

https://www.amazon.com/Digital-Design-Computer-Architecture-Harris/dp/0123944244/ref=sr_1_1?crid=1232QZSYQ20GW&dchild=1&keywords=digital+design+and+computer+architecture+2nd+edition&qid=1597397347&srefix=digital+design+and+%2Caps%2C219&sr=8-1

The RISC-V Reader, David Patterson & Andrew Waterman

https://www.amazon.com/RISC-V-Reader-Open-Architecture-Atlas/dp/0999249118/ref=sr_1_3?dchild=1&keywords=Patterson+RISC-V&qid=1597397389&sr=8-3