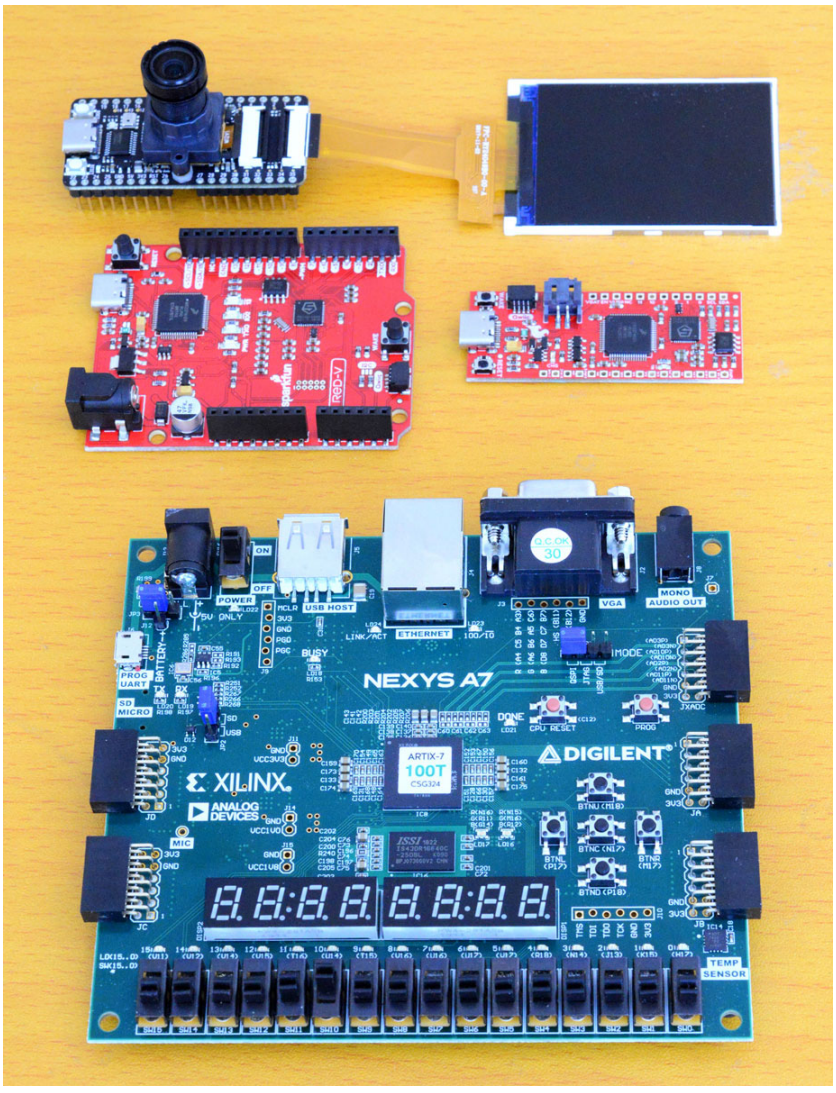
RISC-V指南


NEXYS A7
DIGILENT
XILINX
ANALOG DEVICES
ETHERNET
VGA
USB HOST
AUDIO OUT
TEMP SENSOR

Digi-Key和Imagination Technologies RISC-V指南

目录

1 简介

RISC-V是一种相对较新的计算机技术，正在作为ARM的竞争技术积极推广。

编写本指南的目的是为希望了解RISC-V软件和硬件知识的Digi-Key客户和学员提供简要介绍。

在2020年9月3日举行的RISC-V全球论坛上，Imagination Technologies宣布推出面向RISC-V的“RVfpga：了解计算机架构的完整课程”全球教学项目，这是其计划于2020年11月启动的大学计划的一部分。https://university.imgtec.com上提供了本课程的一些材料。

本文档假设读者之前对RISC-V了解很少或根本不了解，而这正是作者在项目开始时所处的位置。本文档面向可能正在学习计算机架构的大学生，以及希望在教室或在家中扩充知识的电子/计算机工程师。需要Linux的一些基础知识。

1.1 什么是RISC-V?

RISC代表“**R**educed **I**nstruction **S**et **C**omputer”，即精简指令集计算机。此处的**V**代表罗马数字5。因此，RISC-V是指第5代计算机内核系列。它的发音是“Risk Five”。

RISC-V徽标是RISC-V International的注册商标。

图1：RISC-V徽标。图片来源：www.RISC-V.org

有关规范等方面的详细信息，请访问www.RISC-V.org

1.2 动手实验

本指南不是纯粹的理论指南，而是提供了RISC-V动手实验练习，练习中采用了Digi-Key公司提供的三种不同电路板。所述的所有价格均为编写本文时的价格，但可能随着软件的变化而变化：

1. **简单易用、价格合理且充满乐趣。**Seeed Technologies Co Ltd Maix BiT电路板可提供双核RISC-V处理器、摄像头、LCD屏幕和图像识别软件，价格仅为25美元。它采用MicroPython语言编程。（本指南Maix BiT部分的链接）。
2. **中端片上系统（System on a Chip，Soc）。**用C语言或汇编语言在两个SparkFun电路板上进行RISC-V SOC编程，价格分别为30美元和36美元。（本指南SparkFun部分的链接）。
3. **严肃、知识层面要求较高，但价格昂贵。**在Digilent Nexys A7电路板上采用的Xilinx现场可编程门阵列（Field Programmable Gate Array，FPGA）中使用Western Digital SweRV软内核，价格为265美元。但是，它们均可在软件仿真器上运行。（本指南软内核部分的链接）。

好消息是，使用的所有软件都是免费的。

2 RISC-V背景

2.1 历史

RISC-V由David Patterson、Krste Asanovic、Andrew Waterman和Yunsup Lee于2010年在加州大学伯克利分校创建。当时已有Linux开源软件，但是没有等效的开源硬件。为了促进研究，他们创建了自己的指令集架构（Instruction Set Architecture，ISA）。他们在2011年制造了首个RISC-V芯片，并在2014年推出了首款商业产品。

这些年来，还有其他几种RISC实现，包括Microchip PIC、ARM、Atmel AVR、MIPS、SuperH和SPARC。最初的RISC-1实现可追溯到1981年。

2.2 开放式架构和开源

要在诸如处理器或应用特定集成电路（Application Specific Integrated Circuit，ASIC）之类的集成电路中使用ARM®内核，必须先获得许可证。对于最新的内核，这可能需要数百万美元，并且要花费一些时间。对于诸如NXP或Freescale之类的大型跨国公司，这笔费用只是1亿美元总开发费用中的很小一部分。但是，当新的集成电路完成时，每售出一个就必须向ARM支付版税。

RISC-V则有所不同。既没有许可费，也没有版税。这意味着可以随时启动RISC-V实现，并且无需支付任何费用。这对小公司或印度和中国等新兴市场来说是个绝佳的消息。RISC-V架构是固定的，可确保未来产品的向后兼容性。

此外，开源意味着设计可以修改。可以创建特殊指令来提高性能或给黑客的入侵制造麻烦。

RISC-V开发软件（尤其是PlatformIO）是免费的。使用Keil编译器的许可软件（例如ARM MDK）的开销可能很大，而且许可证似乎总是在关键截止日期的前一天到期。

RISC-V内核通常是在Linux中而不是Windows中实现的，这使得内核和软件都是开源的。

2.3 关键参与方

Andes Technology是中国台湾的一家半导体制造商。其产品之一是N22最小RISC-V内核，运行频率为700 MHz，硅面积仅有0.013 mm^2，专为可穿戴设备和物联网（Internet of Things，IoT）应用设计。

Imagination Technologies是一家知识产权（Intellectual Property，IP）公司，专门研究用于手机、平板电脑、计算机和车辆视觉的图形处理单元（Graphic Processing Unit，GPU）。其最著名的产品是PowerVR，这款第10代GPU“A系列”便采用了嵌入式RISC-V控制器。

PlatformIO提供了用于RISC-V开发的免费软件工具。

SiFive由最初的三位RISC-V创造者Krste Asanović、Yunsup Lee和Andrew Waterman创立。它提供RISC-V内核、片上系统（SoC）、IP和开发板。

Western Digital（SanDisk产品的制造商）致力于在未来产品中使用RISC-V，并且已经生产了自己的认证RISC-V内核系列（称为SweRV）。

3 RISC-V基础知识

尽管说RISC-V是计算机内核，更正确的说法应为，RISC-V是ISA。由用户自行制定实现方案。

RISC-V规范由RISC-V.org以两个文档的形式提供：非特权ISA规范（用于基本操作）和特权ISA规范（用于操作系统）。

级别	编码	名称	缩写
0	00	用户/应用	U
1	01	监督器	S
2	10	保留	
3	11	机器	M

表1：RISC-V特权级别。图片来源：RISC-V.org

大多数应用使用的都是用户/应用级别，包括Zephyr操作系统。

3.1 内核命名约定

RISC-V方案可通过16个或32个寄存器以32位、64位或128位实现，并使用规定的命名约定。

名称	功能
RV32I	整数指令集。32位和32个寄存器
RV32E	嵌入式器件的整数指令集。32位和16个寄存器
RV64I	整数指令集。64位和32个寄存器
RV128I	整数指令集。128位和32个寄存器

表2：RISC-V基本ISA。图片来源：RISC-V.org

另外，可选择实现哪些指令。最常见的MAFD也统称为G（通用）。

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

表3：RISC-V标准ISA扩展。图片来源：RISC-V.org

因此，对于使用最少资源的小型SoC，可能会指定RV32EMAB。这意味着*用于嵌入式器件的整数指令集、32位和16个寄存器、整数乘法和除法、原子指令*（以避免与中断发生争用）、没有浮点数学运算，但具有*位操作*。

另一方面，如果需要64位RISC-V实现（包括所有四个通用指令MAFD，以及*位操作*和*用户级别中断*），则称为RV64GBN

3.2 可供下载的内核

现已开发了许多RISC-V内核，大部分是由世界各地的大学完成的。这些内核用多种硬件描述语言（Hardware Description Language，HDL）编写，例如VHDL、Verilog、System Verilog和不知名的Chisel。对于C语言编程人员来说，最容易使用的是Verilog，因为它的语法类似于C语言。System Verilog是超集，也可用于验证。请注意，Verilog是一种硬件描述语言，而不是编程语言。

有关可用RISC-V内核的列表，请参见
https://github.com/riscv/riscv-cores-list

图2给出了一些RISC-V内核的时间线和性能。

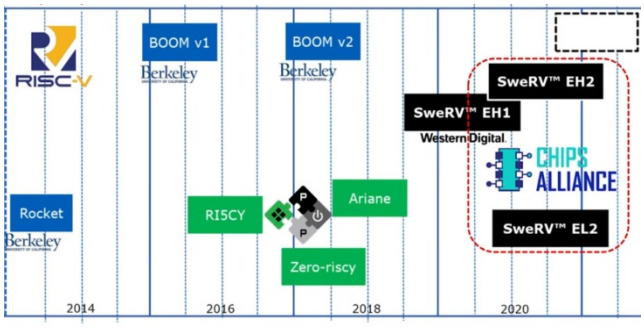


图2：Boom、Rocket、Riscy和SweRV内核。图片来源：Western Digital

Boom是用Chisel编写的Berkeley无序RISC-V处理器。

lowRISC的Rocket内核是具有5级流水线的RV64G变型。

RISCY是由苏黎世联邦理工学院和博洛尼亚大学开发的一个简单内核。

SweRV是Western Digital提供的RISC-V处理器系列（共有3款），非常适合工业应用。

此外，Freedom是SiFive使用的内核。

3.3 与ARM和Intel相比的性能

Western Digital提供的图3给出了ARM、Intel和三款RISC-V内核 – Boom、Rocket和SwerRV的相对性能。CoreMark是嵌入式微处理器基准联盟（Embedded Microprocessor Benchmark Consortium， EEMBC）针对执行指定范围常用操作（例如排序算法、循环冗余校验（Cyclic Redundancy Check， CRC）和状态机等）时实现的性能所设计的基准。

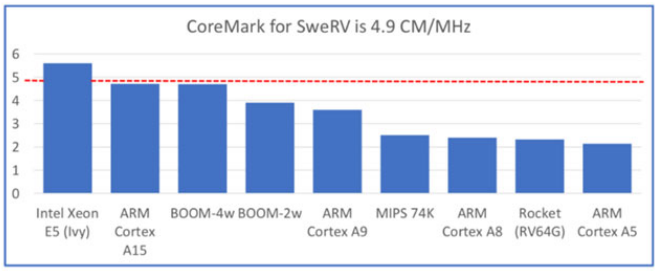


图3：Western Digital CoreMark。图片来源：Western Digital

图中表明，各种RISC-V内核在性能方面可与ARM Cortex A5、ARM Cortex A8、ARM Cortex A9和ARM Cortex A15相媲美。Intel Xeon具有更高的性能。

4 动手实验：使用Seeed Technologies Maix BiT电路板

在四个动手实验板中，此电路板的成本最低且使用最简单。

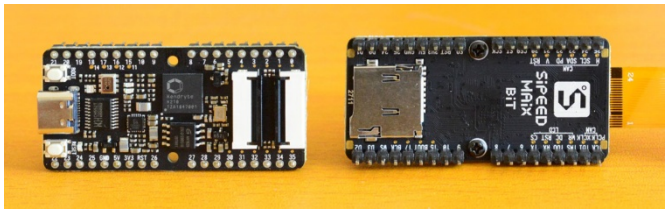


图5：Seeed Technologies Maix BiT电路板

4.1 Maix BiT电路板说明

该电路板专为物联网（IoT）和人工智能（Artificial Intelligence，AI）设计，例如图像识别。它以较低的价格提供了性能非常高的处理器。

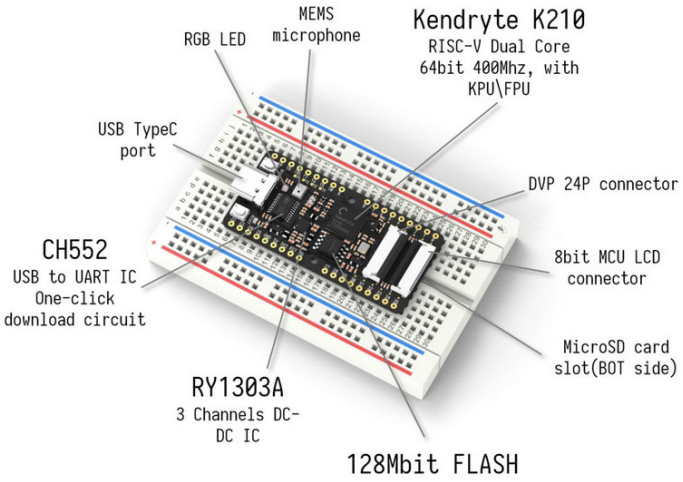


图4：Maix BiT详细信息。图片来源：Seeed Technologies Co Ltd

4.2 基本Maix BiT电路板

可使用基本Maix BiT电路板，即：

Seeed Technology Co. Ltd Sipeed Maix BiT RISC-V AI+IoT，Digi-Key 1597-1714-ND（14.65美元）

4.3 带摄像头和LCD的Maix BiT电路板

但是，对于动手实验，建议使用带摄像头和LCD屏幕的以下套件：

Seeed Technology K210 Sipeed Maix BiT Kit RISC-V AI+IoT Digi-Key 1597-1713-ND（24.21美元）

请注意：Maix BiT电路板未随附USB-A转USB-C线缆，因此需要额外购买：

Seeed Technology USB-C线缆Digi-Key 1597-106990248-ND（2.42美元）

4.3.1 随附的Maix BiT组件

盒中包含一个Maix BiT电路板、一个摄像头和一个LCD显示屏，还提供了一把螺丝刀和3颗螺钉 + 尼龙垫片，但实际上只需要2个螺钉 + 垫片。

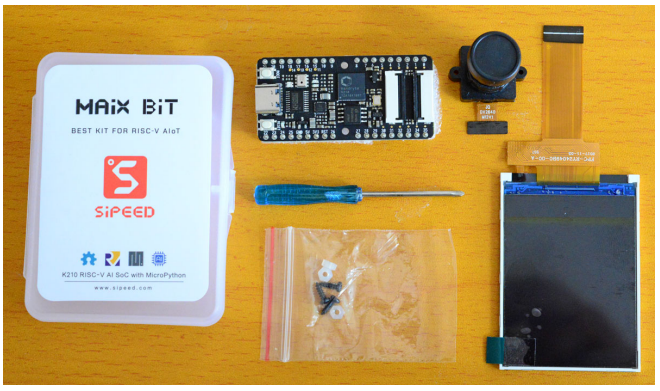


图6：带摄像头、LCD和组件的MaixPy BiT盒和电路板。

4.3.2 组装MaixPy BiT电路板

组装时间：约5分钟。这些组件很小，因此有点复杂。

需要的其他工具：用于在拧紧螺钉时固定尼龙垫片的钳子。

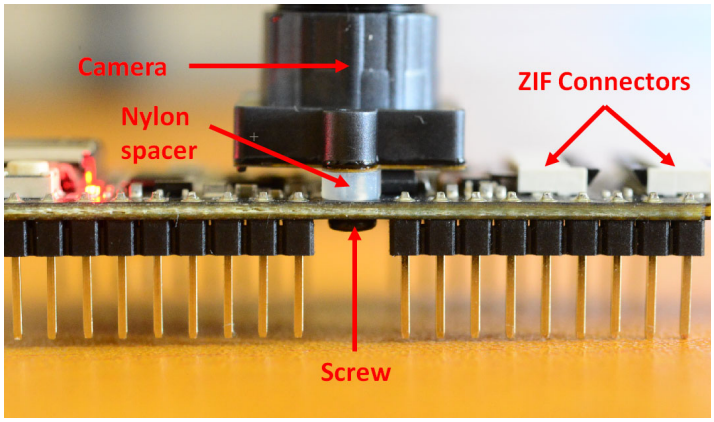


图7：摄像头安装

4.3.3 Maix BiT组装说明

1. 安装两个螺钉和尼龙垫片。
2. 向上推动黑色锁定杆，将摄像头线缆插入电路板中间附近的白色ZIF连接器中。
3. 向下推动黑色锁定杆，将摄像头线缆锁定到位。
4. 需要弯曲摄像头线缆以使摄像头对准螺钉上方。
5. 用螺丝刀拧紧两个螺钉，将摄像头固定到位。
6. 将LCD线缆插入电路板边缘附近的ZIF连接器中。
7. 向下推动黑色锁定杆，将LCD锁定到位。

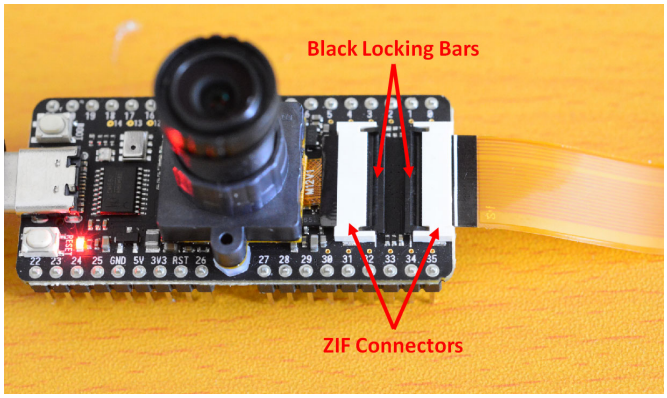


图8：装入ZIF插座的摄像头和LCD

4.4 Maix BiT电路板软件

该电路板预先编程有名为MaixPy的MicroPython解释程序。使用MaixPy IDE软件或只需在串行终端中输入命令即可编写程序。

Windows和Linux均可下载该软件，但Linux需要一些额外的步骤。（本指南Linux安装部分的链接）。

4.5 在Windows上安装MaixPy IDE

这是一个用于编写代码和运行Maix BiT电路板的集成开发环境（Integrated Development Environment，IDE）。

安装时间：约10分钟。

MaixPy IDE软件有点难以找到，因为它的位置看起来是空白的。

转到：`https://maixpy.sipeed.com/en`

此链接将重定向到包含有用信息的MaixPy文档页面。

4.5.1 MaixPy文档网页

单击左侧的“*Install MaixPyIDE(optional)*”（安装MaixPyIDE（可选））。

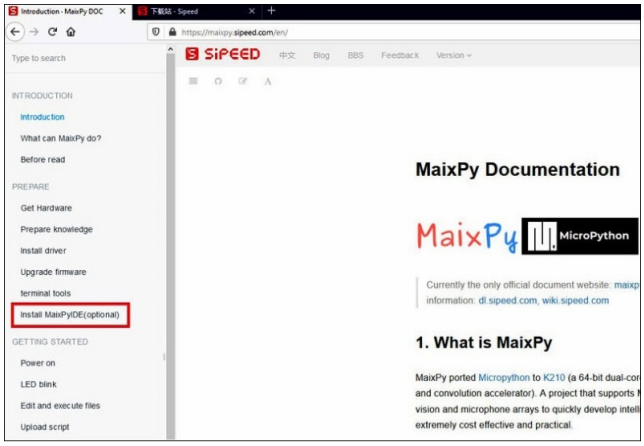


图9：MaixPy文档页面

4.5.2 转到MaixPy下载页面

单击*dl.sipeed.com*

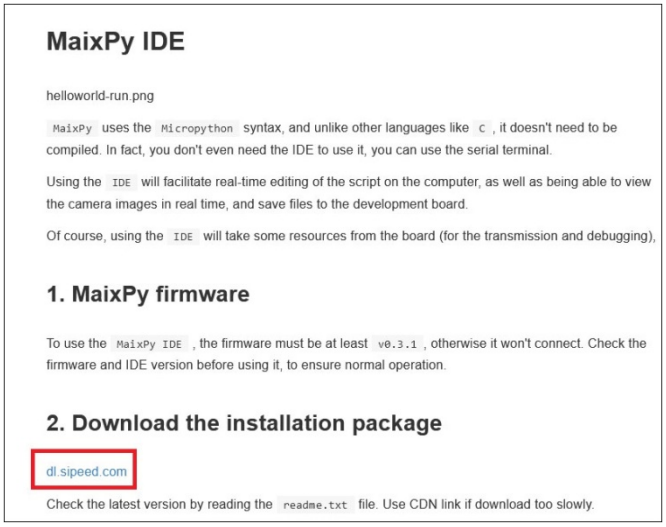


图10：下载安装包

4.5.3 选择目录

单击_

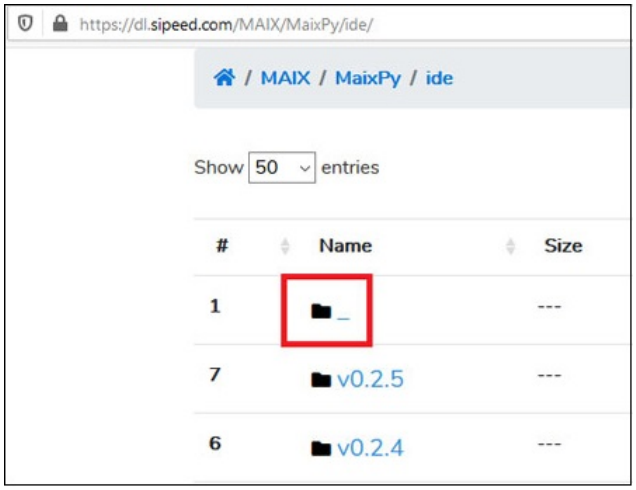


图11：标有_的MaixPy IDE文件夹

4.5.4 选择最新版本

单击最新版本，此处为v0.2.5

图12：选择v0.2.5

4.5.5 选择适用于Windows的MaixPY IDE

单击maixpy-ide-windows-0.2.5.exe。文件大小为85.5 MB。

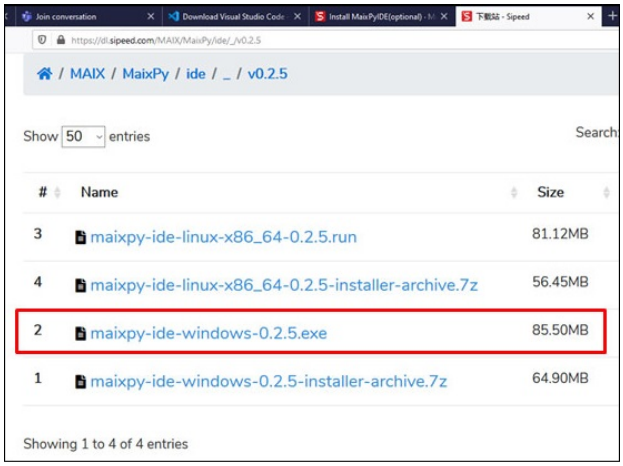


图13：单击适用于Windows的MaixPy IDE

4.5.6 下载MaixPy IDE（Windows）

单击“*Save File*”（保存文件），然后双击maixpy-ide-windows-0.2.5.exe进行安装。

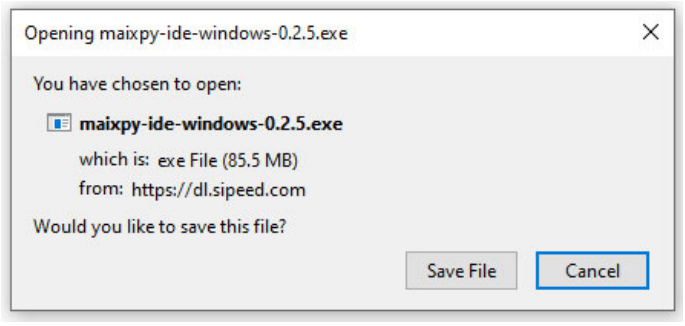


图14：下载maixpy-ide-windows

程序将安装到：

```
C:\Program Files
(x86)\MaixPyIDE\bin\maixpyide.exe
```

4.5.7 MaixPy（Windows）的快捷方式

默认情况下，MaixPy不会在桌面上放置快捷方式图标。要创建MaixPy的快捷方式，请右键单击maixpyide.exe，然后选择“*Create shortcut*”（创建快捷方式），随后快捷方式便可保存在桌面上。

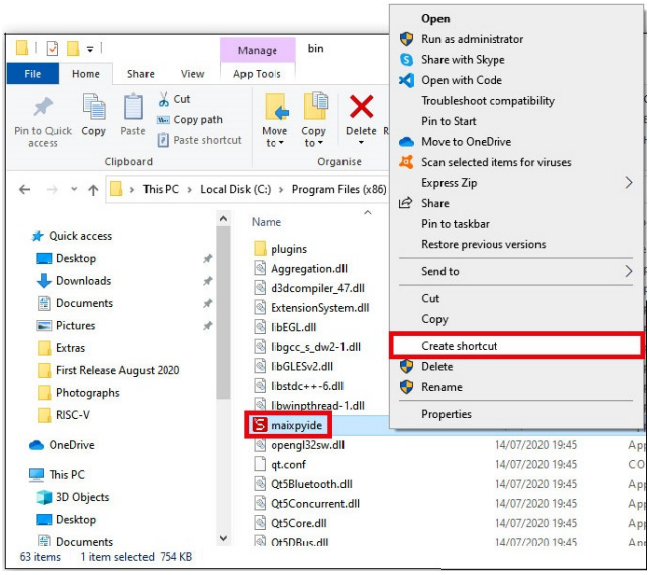


图15：MaixPy IDE的快捷方式

（在MaixPy IDE中运行程序的链接）

4.6 使用Linux安装MaixPy IDE

安装时间：约10分钟。下载的文件大小为287 MB。

还有一个命令行版本可供使用，需要40分钟来下载和安装。

MaixPy IDE软件有点难以找到，因为它的位置看起来是空白的。

转到https://maixpy.sipeed.com/en

此链接将转到包含有用信息的MaixPy文档页面。

4.6.1 MaixPy文档网页

单击“*Install MaixPyIDE (optional)*”（安装MaixPyIDE（可选））

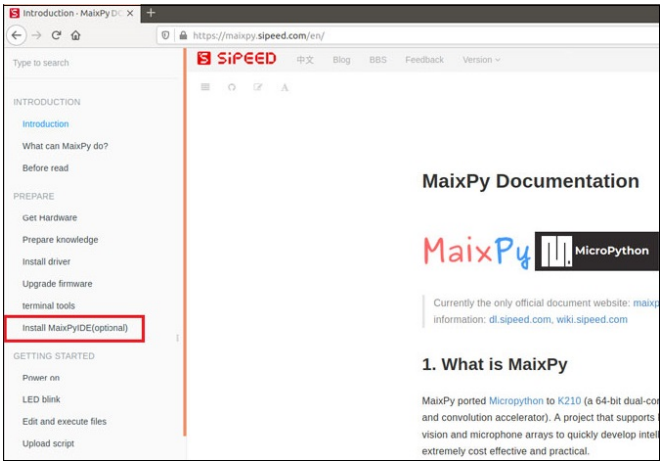


图16：MaixPy文档页面

4.6.2 Seeed下载页面

单击*dl.seeed.com*，转到下载页面。

图17：适用于Linux的MaixPy下载安装包

4.6.3 选择文件夹

单击标有_的文件夹

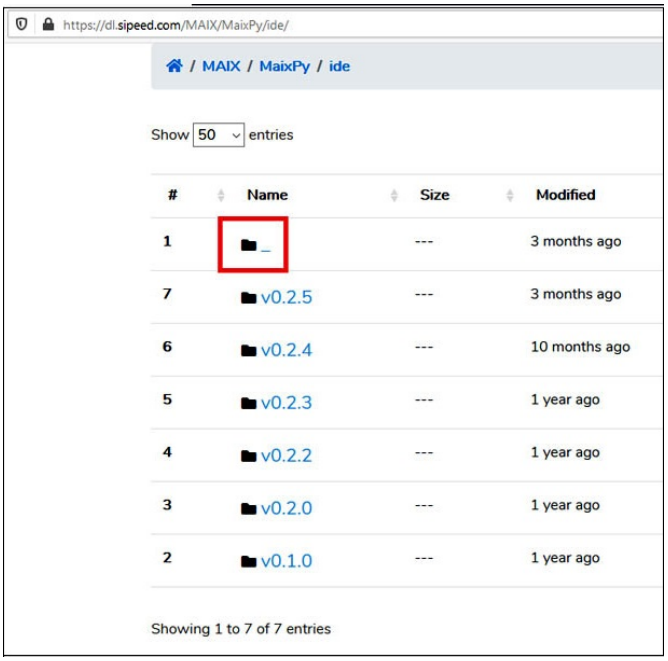


图18：选择标有_的文件夹

4.6.4 选择最新版本

单击*v0.2.5*

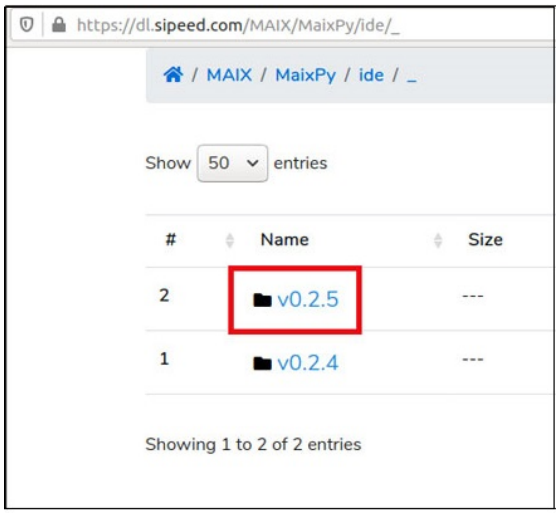


图19：选择v0.2.5

4.6.5 选择文件

单击：

```
maixpy-ide-linux-x86_64.0.2.5.run
```

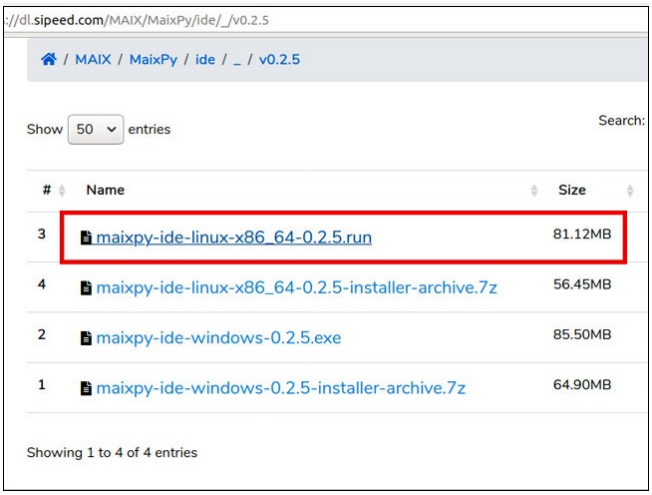


图20：选择Linux运行文件

4.6.6 保存Linux安装文件

单击“*OK*”（确定），保存文件。

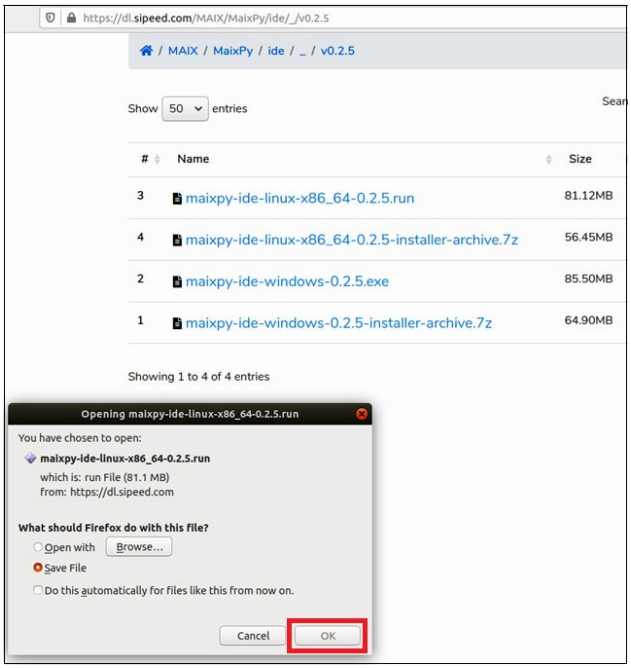


图21：保存Linux安装文件

4.6.7 USB端口访问

默认情况下，USB端口通常不可用，这会阻止MaixPy程序运行。

打开一个*Ubuntu终端* 

并输入：

```
sudo adduser myname dialout
```

其中myname用计算机用户名替换，与/home/myname/中相同

重启计算机，以使更改生效。

如果不执行该步骤，稍后将出现提示。

4.6.8 在Linux中运行下载的文件

打开一个*Ubuntu终端* 

。

为MaixPy提供的指令是为版本0.2.2而不是为版本0.2.5编写的，因此已过时。

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

图22：MaixPy指令

改为输入后来的文件名：

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

```
./maixpy-ide-linux-x86_64-0.2.5.run
```

应出现*MaixPy IDE安装向导*。

4.6.9 MaixPy IDE安装向导

图23：MaixPy安装向导

单击“*Next>*”（下一步>），然后使用默认设置按照向导中的步骤操作。完成后，应显示开始画面。

4.6.10 MaixPy开始画面

如果开始画面未显示来自摄像头的图像（帧缓冲区）和红绿蓝（RGB）信号，请单击画面右侧的 图标。

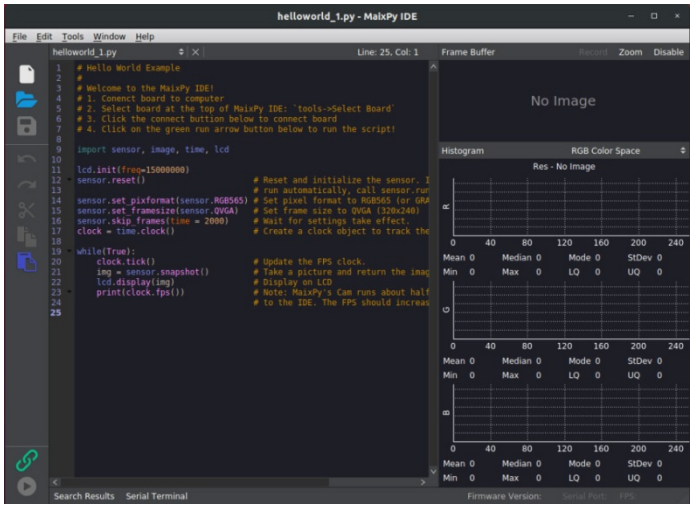


图24：MaixPy Linux IDE画面

4.7 在MaixPy IDE中运行程序

Linux和Windows的过程类似。

4.7.1 插入电路板

将USB线缆插入计算机，LCD上将显示欢迎消息。

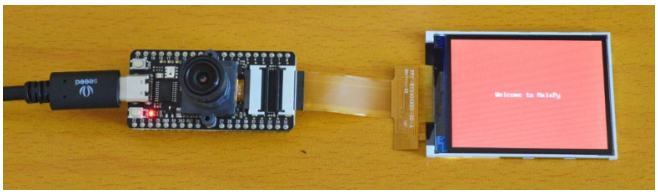


图25：Maix BiT摄像头和LCD操作

4.7.2 启动MaixPy

如果MaixPy IDE尚未运行，则转到Ubuntu搜索工具并输入`MaixPy`。单击*MaixPy IDE*图标。

图26：MaixPy IDE Linux图标

4.7.3 选择程序

首次通电时会默认加载`helloworld_1.py`程序。

4.7.4 选择电路板

在工具栏上，依次选择“*Tools*”（工具）、“*Select Board*”（选择电路板）和“*Sipeed Maix Bit (with Mic)*”（Sipeed Maix Bit（带麦克风））。

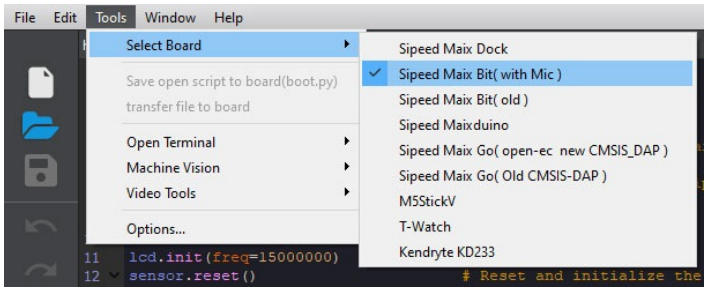


图27：MaixPy IDE选择电路板

4.7.5 选择串行端口

单击画面左下方的绿色图标  以显示“*Connect – MaixPy IDE*”（连接 – MaixPy IDE）菜单。将有多个串行端口可供选择。

选择正确的串行端口后，它会在几秒钟内连接，并且左下方的绿色图标将变为红色。

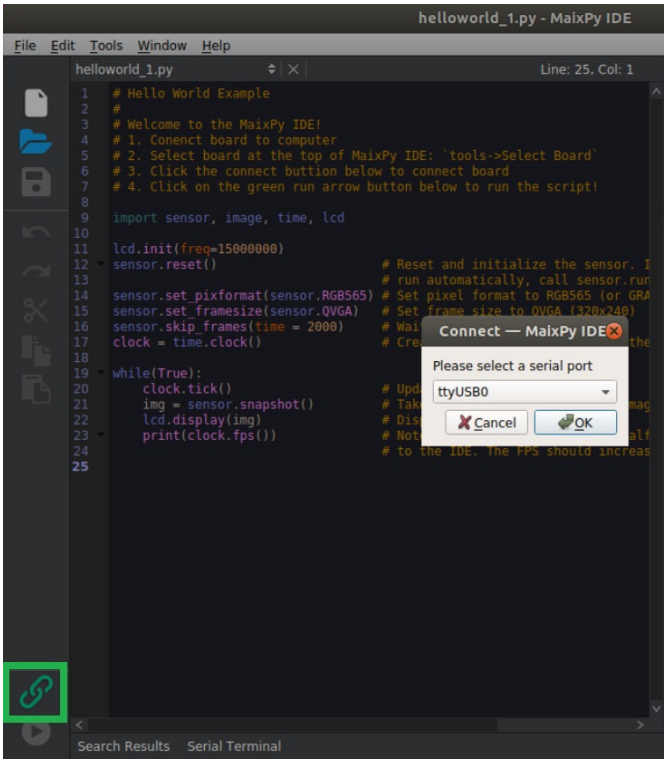


图28：MaixPy串行连接

4.7.6 首次运行时的Linux弹出窗口

如果MaixPy IDE尚无法使用Linux USB端口，则会弹出类似于下面的提醒，但实际计算机用户名不是“richard”。请按照说明操作。

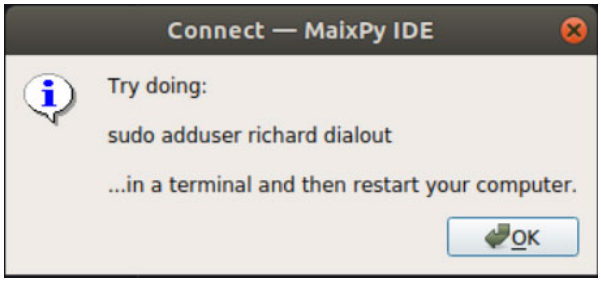


图29：对话框组

4.7.7 运行程序

单击画面左下方的绿色箭头  ，运行程序。

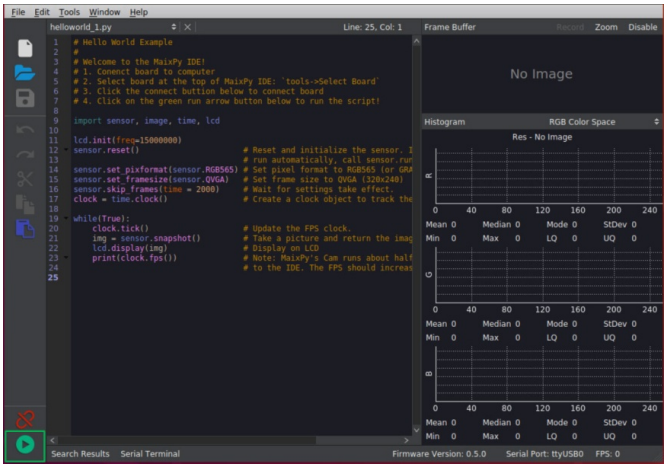


图30：设置程序以准备好运行

4.7.8 Hello World程序运行

helloword_1.py程序现在正在运行。摄像头的图像以及RGB光谱显示在计算机上。图像也显示在LCD屏幕上。

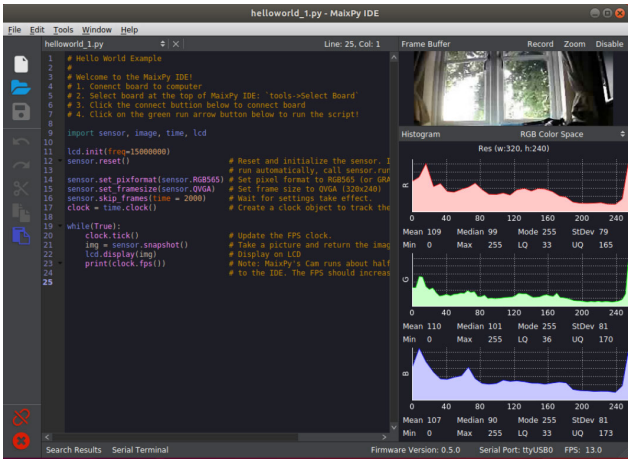


图31：在Linux中运行Hello World应用程序

要停止程序，请单击“*Stop*”（停止）图标  。

4.8 其他MaixPy项目

此外，还提供了其他项目，但是需要下载。在MaixPy IDE中，依次单击“*File*”（文件）、“*More examples*”（更多示例）和“*Examples on github repo*”（github资源库上的示例），转到Github。这适用于Linux和Windows版本。

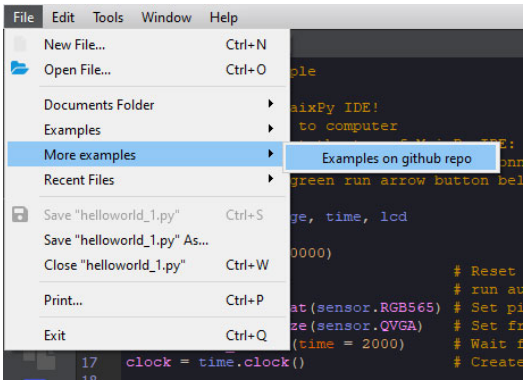


图32：更多MaixPy示例

4.8.1 下载MaixPy示例

下载MaixPy_scripts-master.zip并保存，然后解压缩文件。

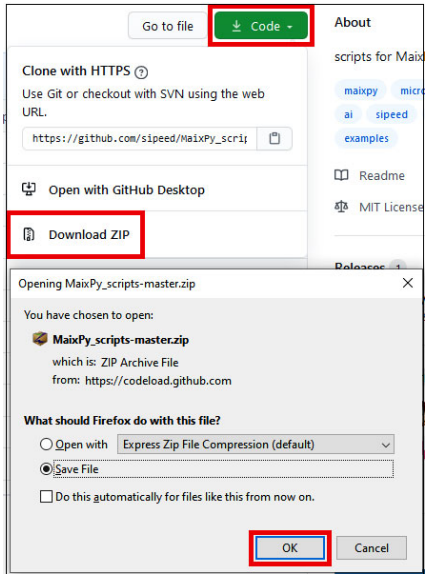


图33：MaixPy示例

4.9 人脸识别项目

所需时间：约15分钟。这涉及到下载和安装另外两个程序。

MaixPy的其他示例在machine_vision目录中提供了人脸识别程序demo_find_face.py。将其加载到MaixPy IDE中。

4.9.1 所需的附加文件

该项目需要在运行之前将名为face_model_at_0x300000.kfpkg的附加模型加载到MaixPy BiT电路板上。为此，需要kflash_gui工具。

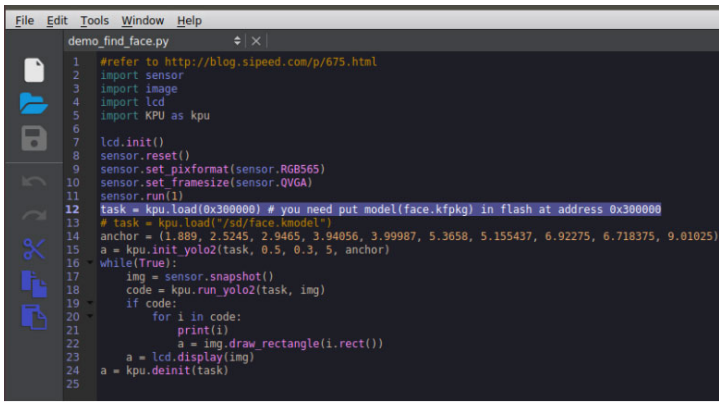


图34：人脸识别程序附加文件

请注意：在早期版本的kflash_gui上，默认下载位置为0x00000。这意味着可能将face_model_at_0x300000.kfpkg下载到地址0x00000，而不是正确的地址0x300000。这会改写MicroPython解释程序，而电路板根本不会运行。因此，有必要对电路板进行重新编程。

4.10 更新Maix BiT闪存

所需时间：约10分钟。

kflash_gui工具可在Windows和Linux上运行。使用最新软件可以迅速而轻松地烧写电路板。

4.10.1 下载kflash_gui

可从以下位置下载kflash_gui：
https://github.com/sipeed/kflash_gui/releases.

此处使用的是kflash_gui_v1.5.3_windows.7z。更高版本会导致Avast Virus检查程序出现问题。

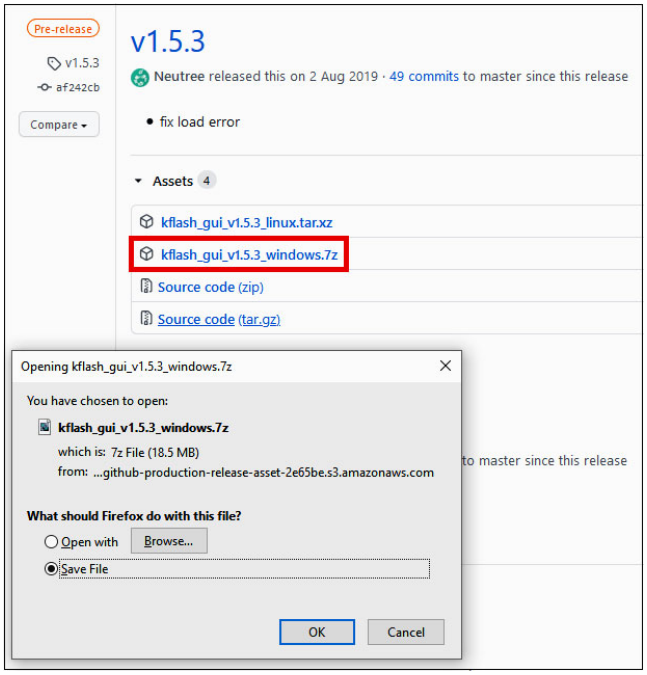


图35：下载kflash GUI

解压缩.7z文件。如果计算机不支持.7z文件，可从以下网址在线获取免费工具，例如Express Zip：

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

要运行该程序，转到解压后的文件，然后双击 kflash_gui图标。

请注意：kflash_gui安装不会在桌面上放置快捷方式。如果需要快捷方式，则需要手动创建。

4.10.2 下载人脸模型

从https://github.com/sipeed/MaixPy/releases下载face_model_at_0x300000.kfkpg并保存文件。

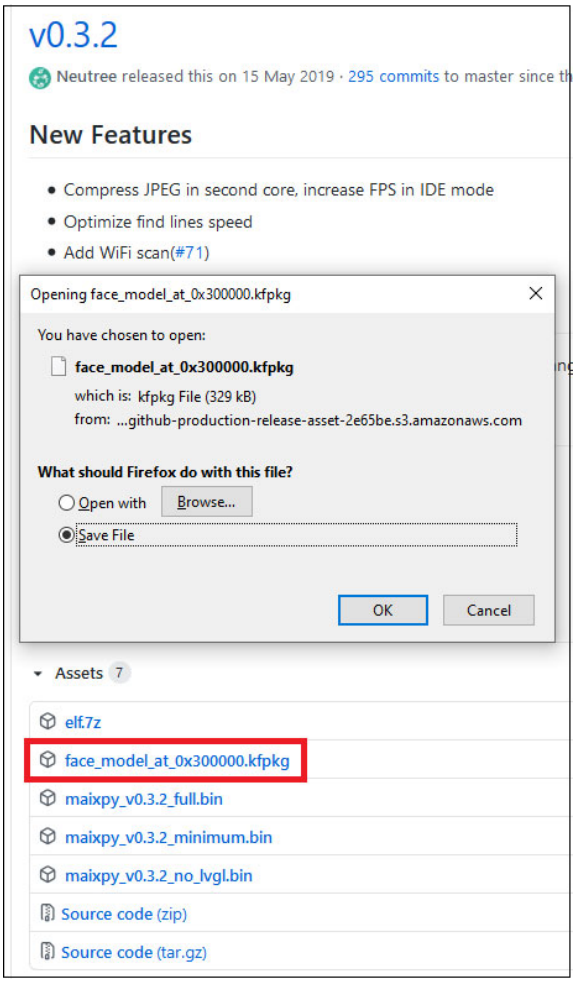


图36：下载人脸模型

4.10.3 使用kflash GUI编程人脸模型

单击 kflash_gui图标。单击“*Open File*”（打开文件），然后选择face_model_at_0x300000.kfpkg。依次选择“*Sipeed Maix Bit (with Mic)*”（*Sipeed Maix Bit（带麦克风）*）*电路板*和串行*端口*，然后单击“*Download*”（下载）。

下载需要大约20秒。

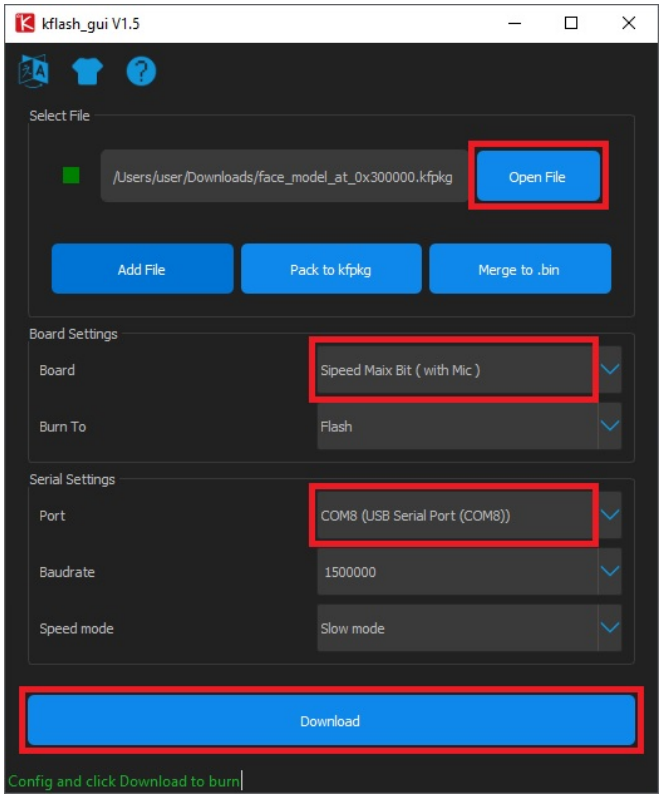


图37：kflash GUI

4.11 运行人脸识别程序

返回MaixPy IDE，然后运行machine_vision目录中的demo_find_face.py。

识别到人脸后，LCD和计算机屏幕上将绘制一个矩形。

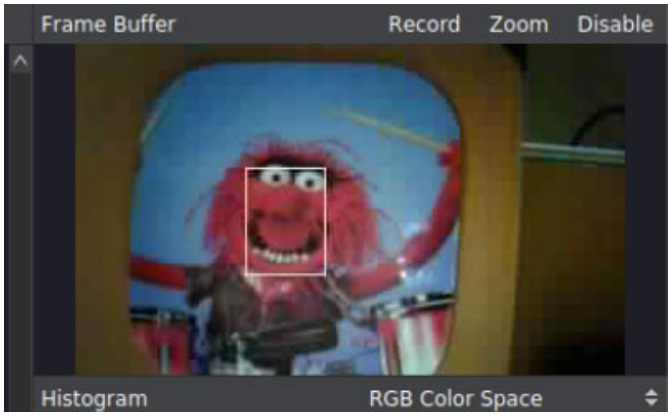


图38：人脸识别示例

4.12 重新烧录Maix BiT电路板主软件

kflash_gui还可用于在Maix BiT电路板上重新编程MicroPython解释程序。固件更新十分频繁，但并非全部都与MaixPy IDE兼容。

4.12.1 下载最新软件

转到：

https://dl.sipeed.com/MAIX/MaixPy/release/master。

其中包含MaixPy版本的列表。单击最新版本。

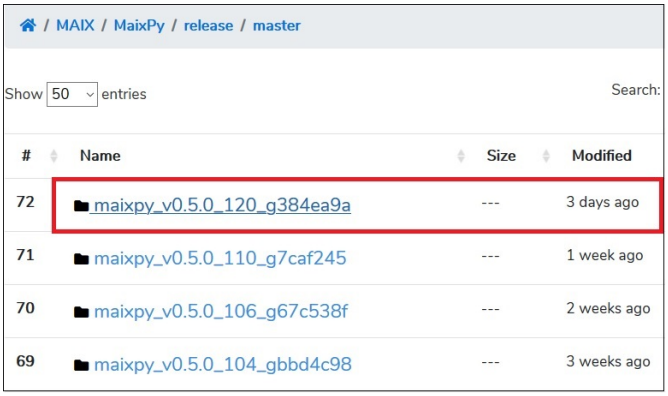


图39：MaixPy版本主索引

4.12.2 选择最新MaixPy

单击以_minimum_with_ide_support.bin结尾的文件并保存。

文件具有IDE支持至关重要。

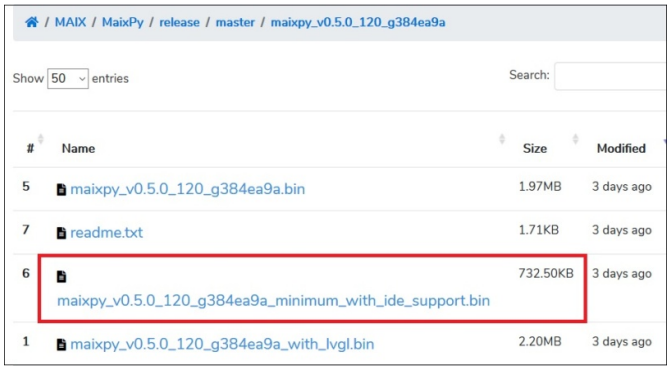


图40：Maix BiT主软件下载

4.12.3 编程Maix BiT电路板

打开kflash_gui。使用“*Open File*”（打开文件）进行选择：

```
maixpy_v0.5.0_120_g3943a9a_minimum_with_ide_support.bin
```

依次选择*电路板*和串行*端口*，然后单击“*Download*”（下载）。

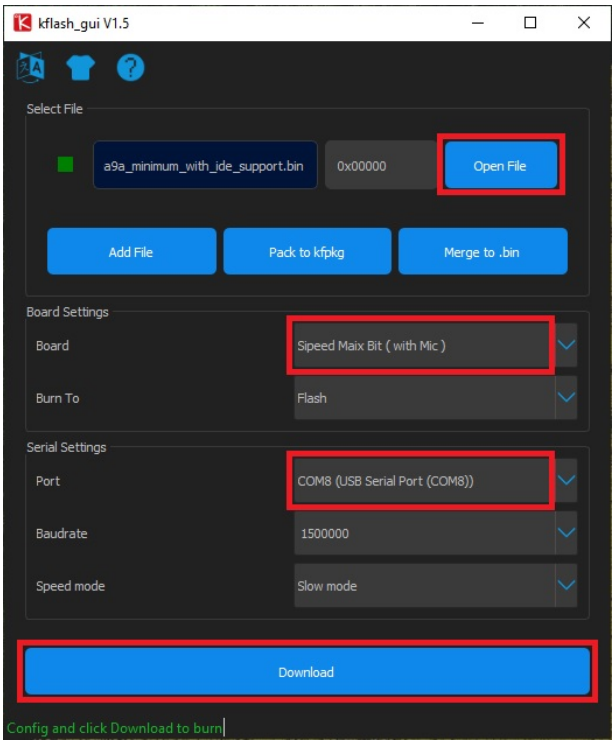


图41：重新编程Maix BiT闪存

4.13 在串行终端上运行MaixPy

所需时间：约5分钟。

除了使用MaixPy IDE外，程序也可以在串行终端上运行。这种方法快速轻松，对于检查Maix BiT电路板配置和软件版本十分有用。它也可以用作学习MicroPython的工具。

4.13.1 设置串行终端

如果尚未安装，请下载并安装任何免费的串行终端程序，例如Putty或TeraTerm。

将串行端口速度设置为115200波特。在另一台计算机上，端口COM8可能有所不同。

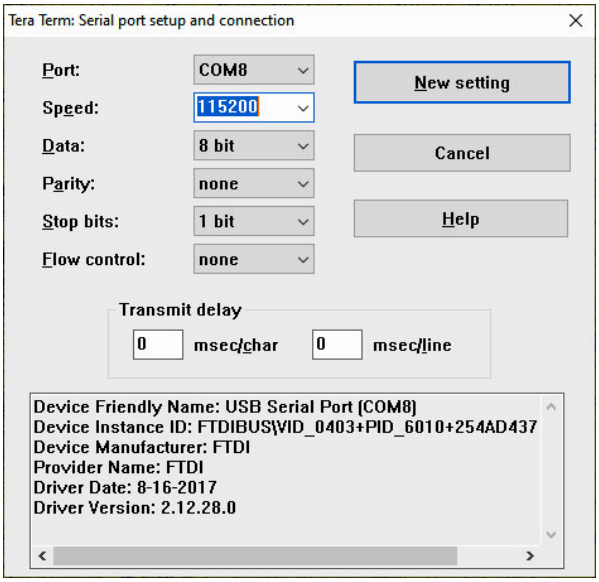


图42：串行终端配置

4.13.2 按下并释放复位按钮

按下并释放MaixPy BiT电路板上的*复位按钮*。

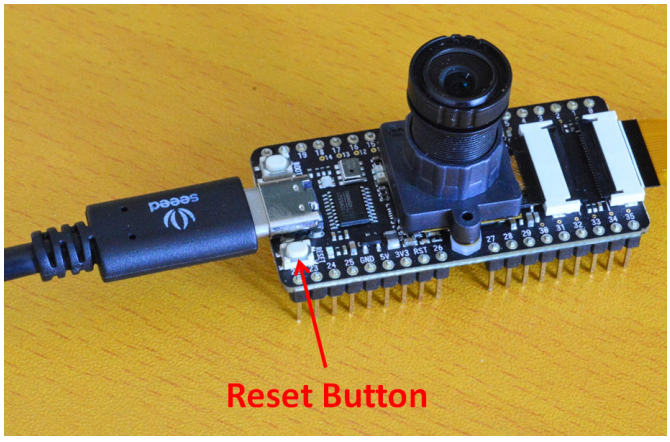


图43：Maix BiT电路板复位按钮

4.13.3 在串行终端上运行MaixPy

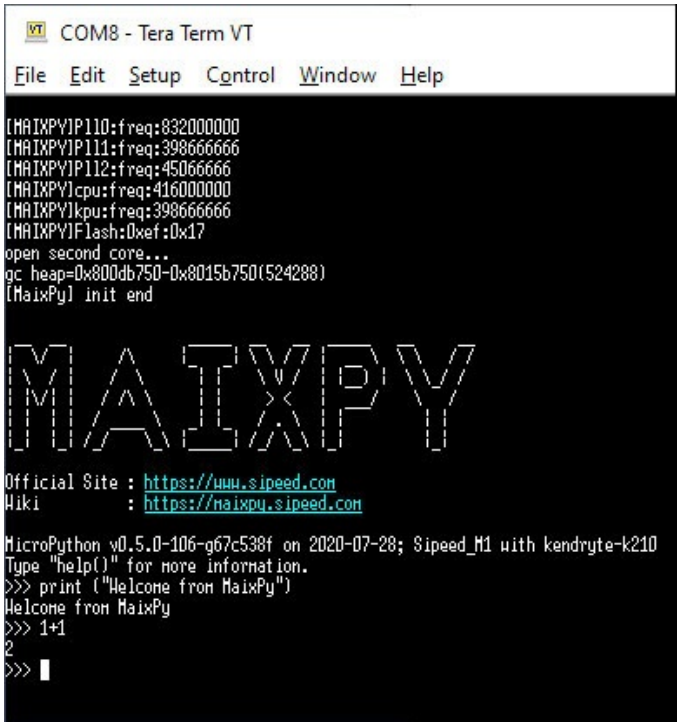


图44：串行终端上的MaixPy

输入print（“Welcome from MaixPy”），响应为“Welcome from MaixPy”。

输入1+1，响应为2。

5 动手实验：在SparkFun RED-V电路板上使用SiFive SoC

这是中档选项，其中两个电路板设计为在FreedomStudio IDE中使用C语言编程。

5.1 SparkFun视频

Digi-Key网站上提供了Shawn Hymel制作的一个精彩短片，展示了如何使用这些电路板。在购买电路板、安装软件之前，或稍后等待软件下载时，这部短片非常值得一看。

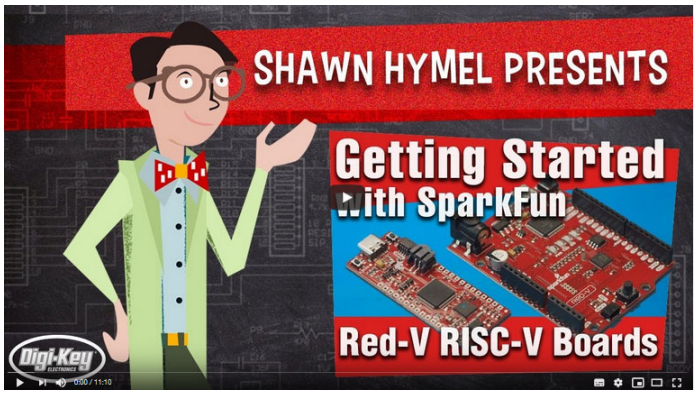


图45：RED-V入门。来源：Digi-Key网站。

Digi-Key网站的Red Board和RED-V Thing Plus页面上也提供了视频的链接。

5.2 硬件

基于SiFive Freedom Everywhere RISC-V SoC的SparkFun Electronics提供两个电路板。两个电路板均兼容SiFive1 Rev B电路板和软件：

小型SparkFun RED-V SIFIVE RISC-V THING PLUS - 1568-DEV-15799-ND（29.95美元）

稍大的SparkFun Electronics FE310 Red Board - 1568-DEV-15594-ND 39.95（39.95美元）

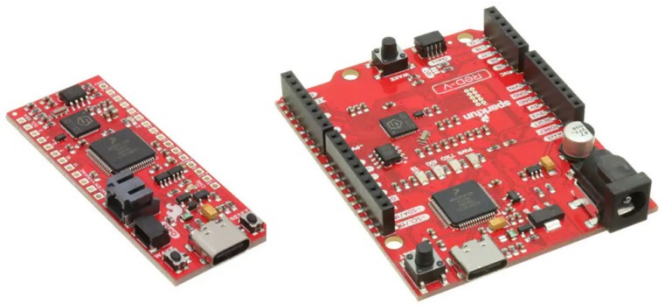


图46：SparkFun RED-V Thing Plus（左侧）和Red Board（右侧）。图片来源：SparkFun Electronics

注： Thing Plus和Redboard都没有USB-C转USB-A线缆，因此需要自行购买，例如：

Seeed Technology USB-C线缆Digi-Key 1597-106990248-ND（2.42美元）

SiFive提供适用于Windows和Linux的软件。遗憾的是，作者无法让Linux版本运行，因此这里只考虑Windows版本。

5.3 SiFive Freedom Studio软件安装（Windows）

安装需要约20分钟，下载文件的大小为1.3 GB。

转到www.SiFive.com/software，将页面向下滚动到Freedom Studio部分。

单击Windows，然后保存.zip文件。

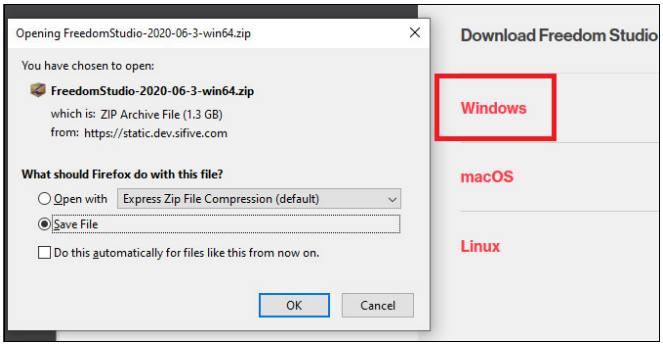


图47：SiFive Freedom Studio（Windows）

5.3.1 Freedom Studio文件位置

解压文件。导航到安装目录，例如：

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

双击运行 FreedomStudio。

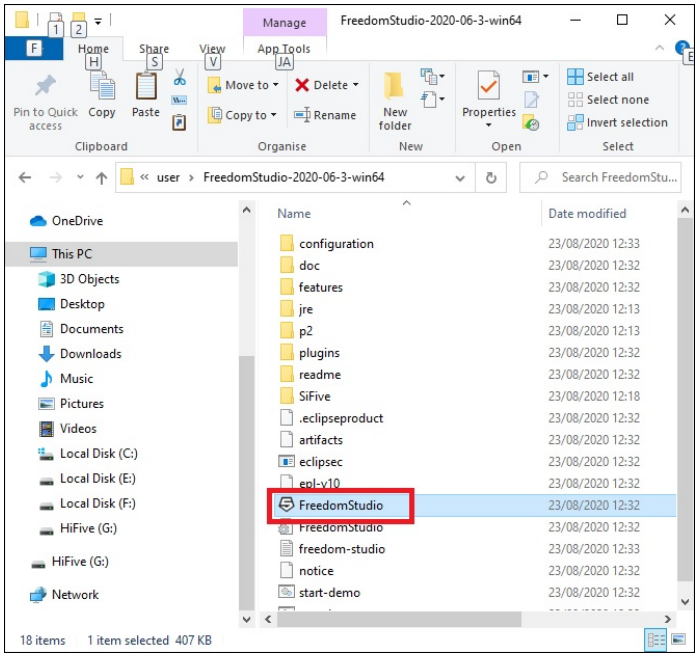


图48：FreedomStudio文件位置

5.3.2 Freedom Studio启动画面

应出现Freedom Studio图标。如果需要，可将快捷方式添加到桌面。

图49：Freedom Studio启动画面

5.4 创建一个新的软件项目

如果尚未运行，请启动Freedom Studio。

5.4.1 插入电路板

建议在创建程序之前插入电路板，因为它会自动设置调试配置。使用USB-C转USB-A线缆。可使用任一SparkFun电路板。

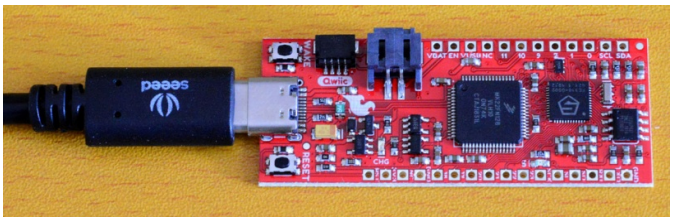


图50：用USB-C线缆连接Thing Plus Board

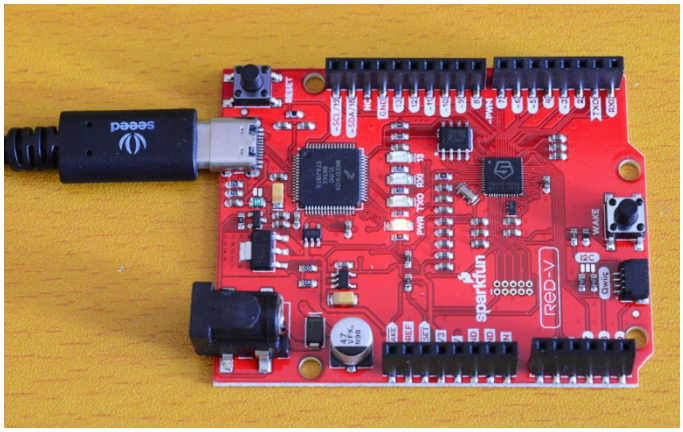


图51：用USB-C线缆连接Red Board

5.4.2 创建新的Freedom E SDK软件项目

转到画面顶部的“*SiFive Tools*”（SiFive工具），单击“*Create a new Freedom E SDK Software Project*”（创建一个新的Freedom E SDK软件项目）。

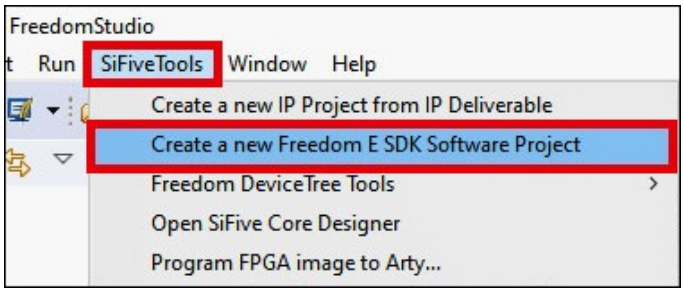


图52：创建新的Freedom E SDK软件项目

5.4.3 电路板选择

支持多种电路板。从下拉菜单中选择*siFive-hifive1-revb*。它同时兼容Red Board和Thing Plus Board。

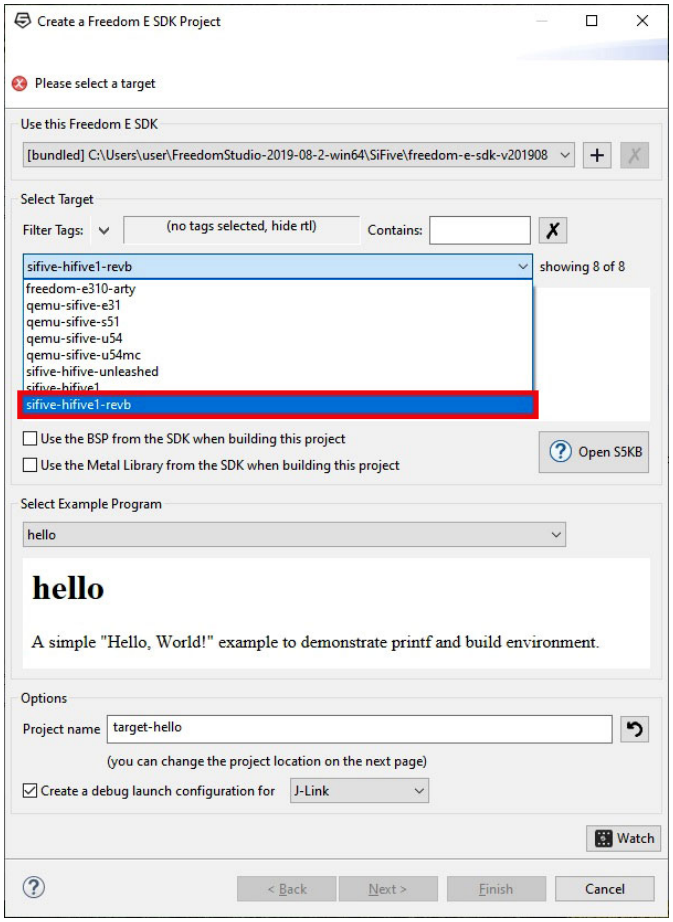


图53：SiFive电路板选择

5.4.4 示例程序选择

从“*Select Example Program*”（选择示例程序）下拉菜单中，可选择一系列示例项目。

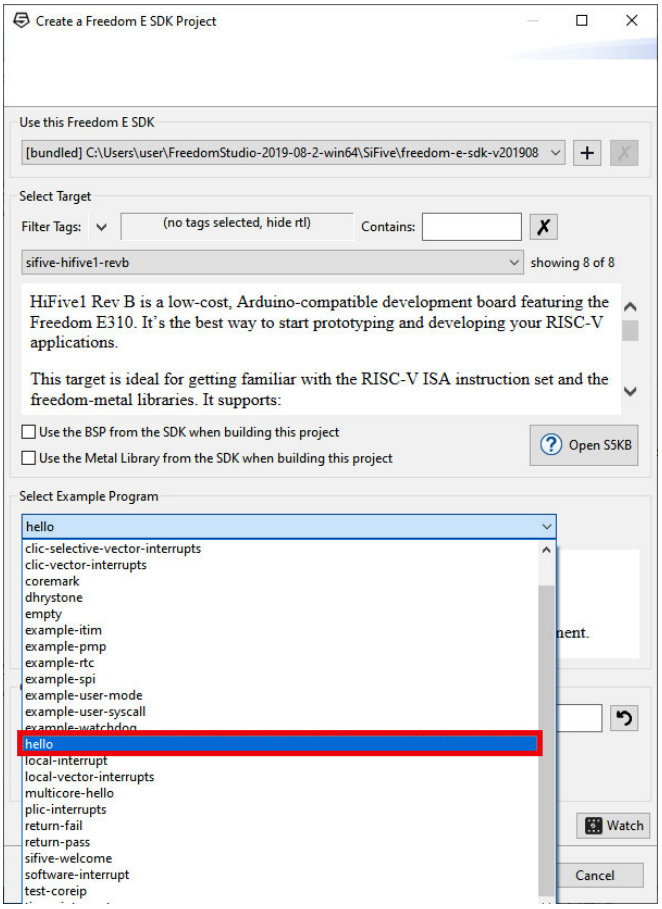


图54：SiFive示例项目

这些示例项目是其他项目的绝佳起点。目前，请继续使用*hello*。

5.4.5 调试启动配置

在窗口底部，已自动设置“*Create a debug launch configuration for J-Link*”（创建J-Link的调试启动配置）。原因是电路板已插入。单击“*Finish*”（完成）。项目将自动编译。

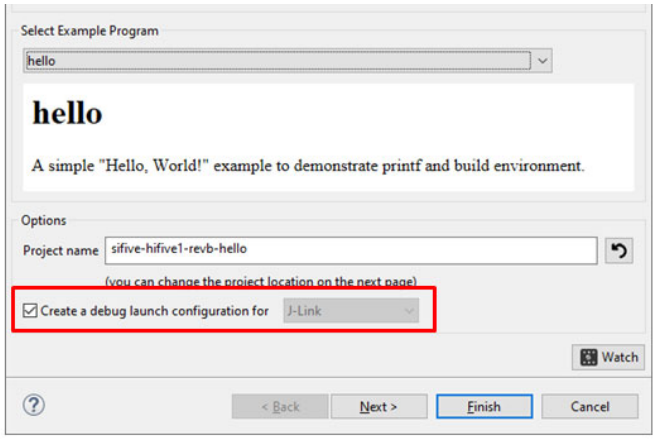


图55：选择示例程序和调试启动配置

5.4.6 检查.elf文件

“*Edit Configuration*”（编辑配置）窗口将自动弹出。检查是否已创建可执行且可链接的文件`hello.elf`。这是编程电路板的必要步骤。

可通过单击“*Debug*”（调试）直接调试，但目前需要先单击“*Close*”（关闭），因为还有一些步骤需要设置。

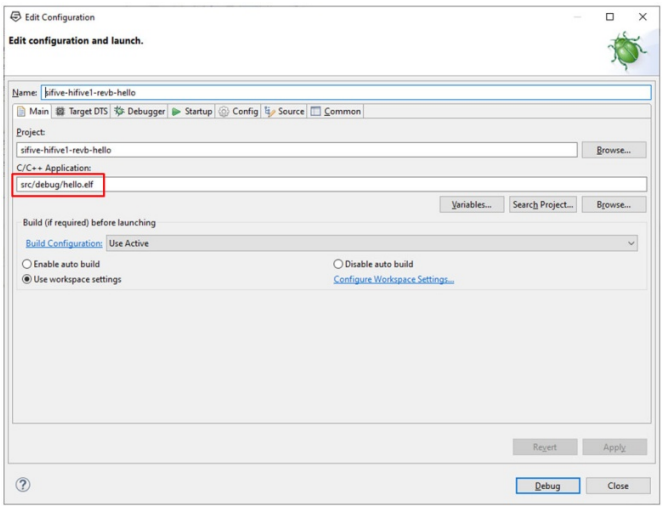


图56：编辑配置菜单

5.4.7 控制台和源代码视图

“*Console*”（控制台）窗口提供有关编译的信息 – 所使用的编译器、代码大小和编译时间。

第一次编译需要一段时间，因为它需要编译项目中的所有文件。后续编译仅重新编译实际发生更改的文件，因此速度要快得多。

`hello.c`窗口提供C代码。

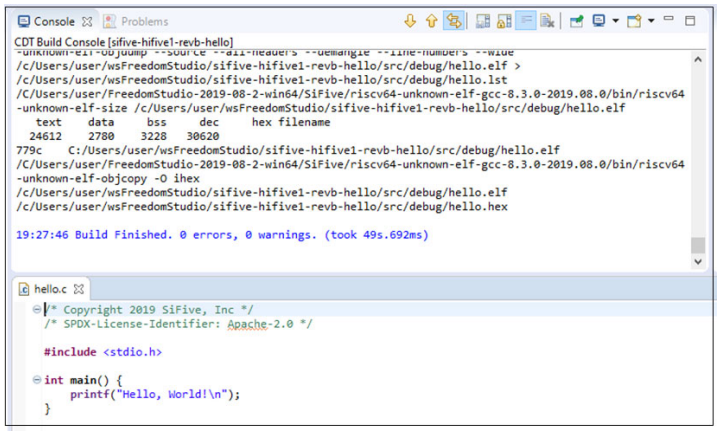


图57：控制台和源代码窗口

5.4.8 修改`hello.c`源代码

如果按原样调试`hello.c`程序，则不会看到很多内容，因为它只执行终止操作。因此，将图58中所示的`hello.c`修改为显示`Hello, World!` 10次。

图58：修改后的hello.c文件

可通过单击工具栏上的“*Build*”（编译）图标  来生成程序以检查语法，但当运行“*Debug*”（调试）时，该操作会自动完成。

5.4.9 调试程序

要开始调试程序，请单击工具栏上的“*Debug*”（调试）图标  或按下键盘上的F11键，也可先单击工具栏上的“*Run*”（运行）再单击“*Debug*”（调试）。

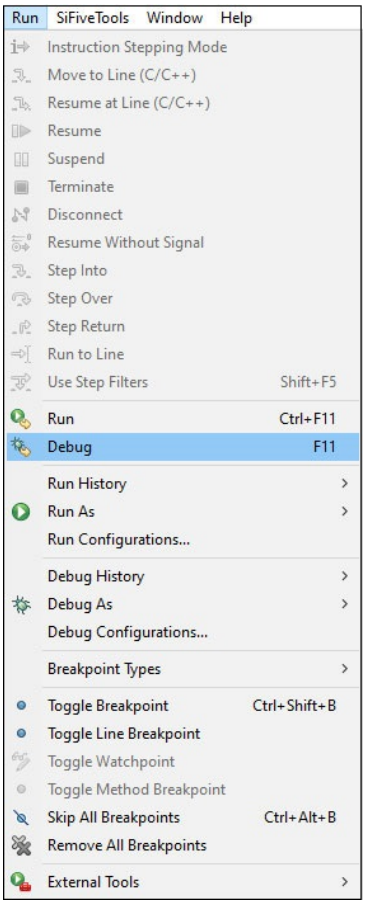


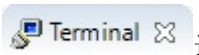
图59：开始调试

5.4.10 调试器在main()处停止

调试器将在main()开始处停止。

图60：调试器在main()处停止

5.4.11 设置用于printf的串行终端

单击“*Terminal*”（终端） 

选项卡，然后单击画面右侧的“*Open a Terminal*”（打开终端） 图标。

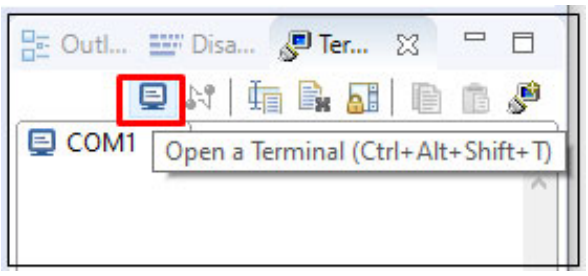


图61：“Open a Terminal”（打开终端）图标

5.4.12 启动终端

将串行终端设置为115200波特。有多个*串行端口*，因此可能需要进行多次尝试。

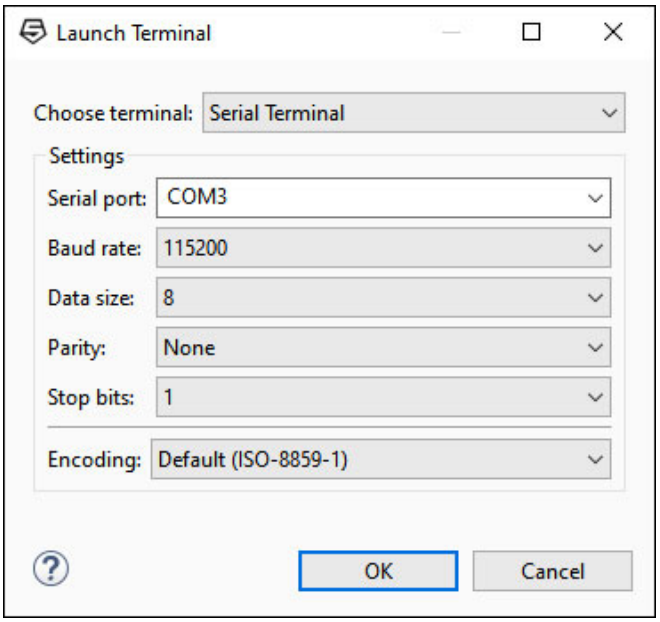


图62：串行终端设置

5.4.13 调试输出

要单步执行代码，请单击工具栏上的“*Step Over*”（单步跳过）  图标，或多次按下键盘上的F6功能键，以便使Hello, World!出现十次。

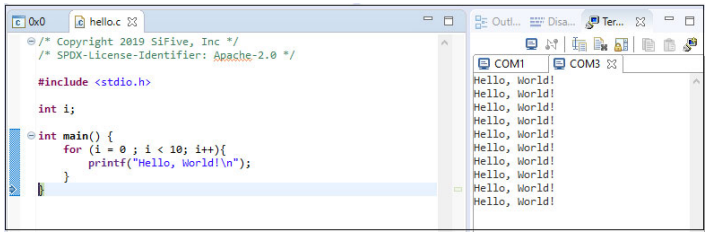
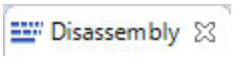


图63：COM3上的调试输出

printf消息在同一程序的窗口中显示，这对于调试和代码覆盖非常有用。这比必须连接到单独的终端容易得多。

5.4.14 反汇编窗口

图64给出了稍作修改的hello.c程序版本，用于与ARM®进行比较。

要查看组成C代码的汇编语言指令，请单击“*Disassembly*”（反汇编）选项卡 

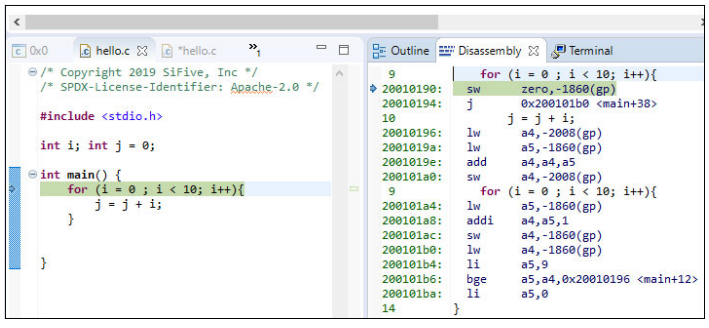


图64：反汇编窗口

5.4.15 与ARM Cortex M0的比较

图65给出了等效的ARM® Cortex® M0+反汇编代码。

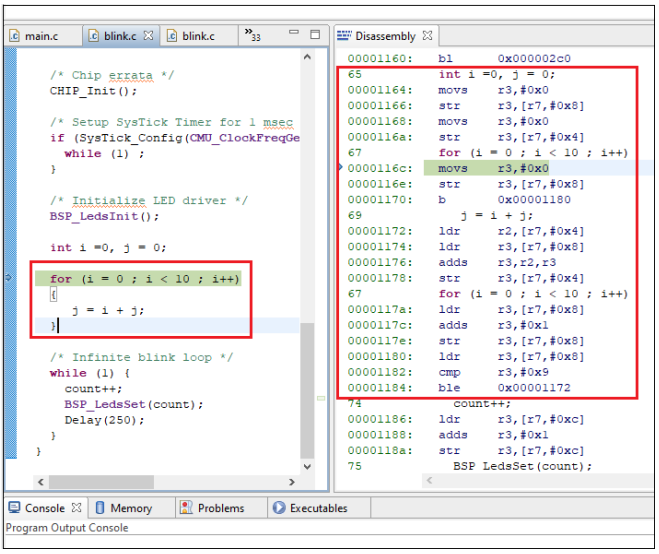


图65：与ARM Cortex M0的比较

它表明，在这种情况下，RISC-V所需的指令少于ARM。

5.4.16 停止程序

要停止程序并从头开始再次运行程序，请单击*工具栏*上或“*Console*”（控制台）窗口中的“*Terminate*”（终止）  图标。

要再次开始调试程序，请单击工具栏上的“*Debug*”（调试）图标  ，或按下键盘上的F11键。

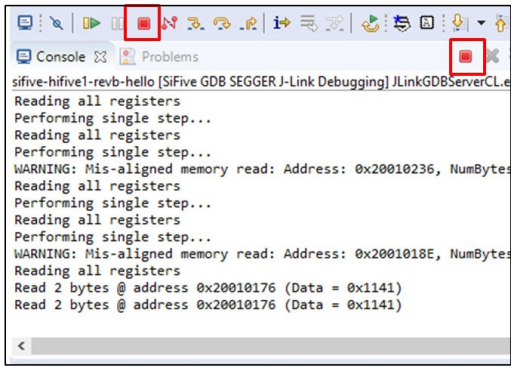


图66：终止程序

5.4.17 关闭并重新打开现有项目

当现有的Freedom Studio项目关闭并再次打开时，会有一个小麻烦。关闭Freedom Studio，然后再次将其打开。单击项目，然后单击“*Debug*”（调试）图标。程序无法调试，并显示错误消息：

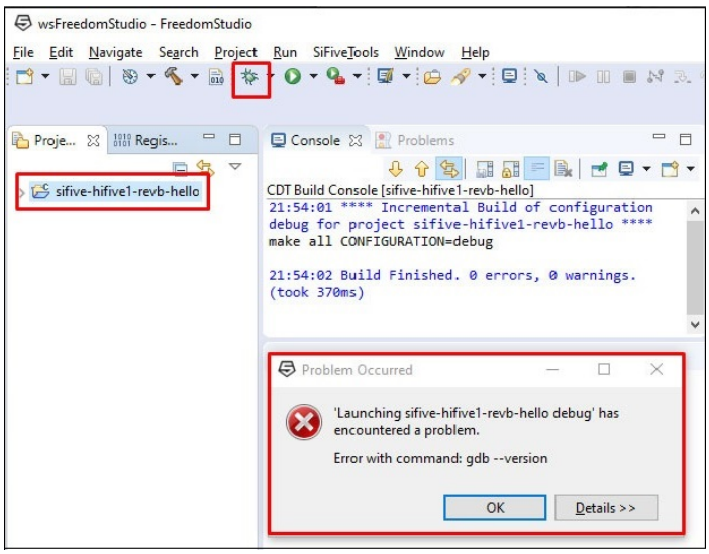


图67：重新打开项目时出现问题

5.4.18 选择调试配置

从画面顶部的*工具栏*中，选择“*Run*”（运行），然后选择“*Debug Configurations*”（调试配置）。

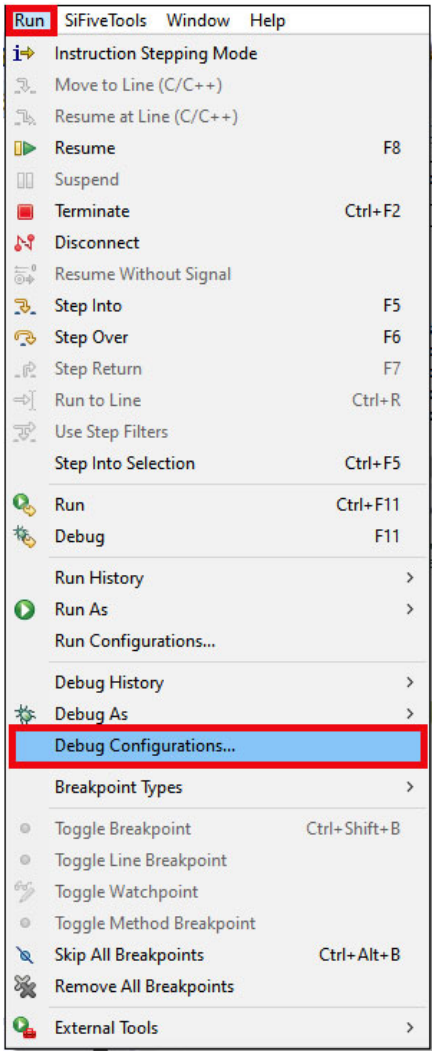


图68：选择调试配置

5.4.19 创建、管理和运行配置

展开“*Sifive GDB SEGGER J-Link Debugging*”（Sifive GDB SEGGER J-Link调试）选项卡并单击要调试的项目，此处为`sifive-hifive-1-revb-hello`。单击“*Debug*”（调试）。现在应该可以调试程序了。

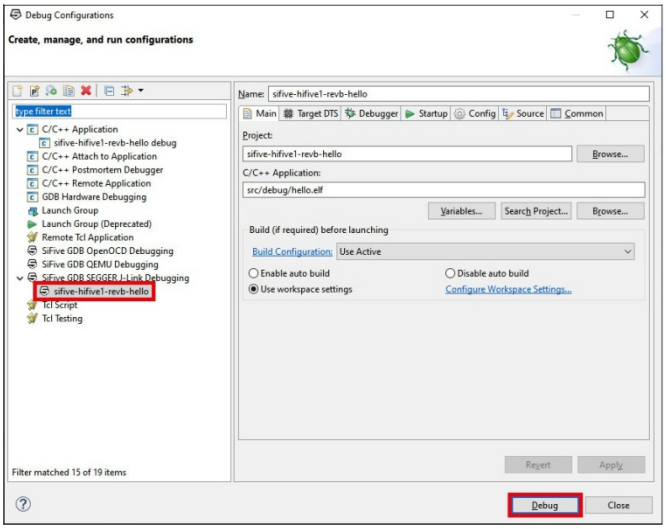


图69：选择要调试的项目

6 动手实验：使用Digilent Nexys A7实现软内核

这是指南中对技术要求最高的部分，涉及将RISC-V内核嵌入到FPGA中，非常适合详细研究计算机架构，但对于某些读者而言可能过于复杂。

6.1 Imagination Technologies大学计划提供的支持

在FPGA上实现软内核是一项艰巨的任务。幸运的是，Imagination Technologies大学计划正在全球范围内发布如何在Xilinx Artix-7 FPGA上实现RISC-V内核的教材，也就是众所周知的“RVfpga：了解计算机架构的完整课程”。这包括一整套的实验（20个）、文档、示例和带解决方案的练习。所有必需软件都可以免费下载。这为用户免去了大量的工作和棘手问题。这里提供的材料版本占整个课程的10%。

要获得RVfpga材料，首先需要在以下网站上注册：

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

填写注册表单的过程可能有些长，因为这是专为大学而设计的。但是，RVfpga材料可供大学以外的人员使用，因此，在要求填写与大学相关的字段时，可以输入来自商业或其他组织的等效内容。注册所需的时间少于10分钟，因为并非所有字段都必须填写。电话号码中不允许有空格。

注册后，登录并请求从“Teaching Resources”（教学资源）页面下载材料：

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

下载材料相当于大学三个学期的本科学习课程，涵盖了20个实验，而且全部免费。此外，还有相应的论坛可供探讨问题和回答问题。

2021年第一季度将开设硕士级“创建SoC”课程。

6.2 软内核选项

假设项目对成本不太敏感，那么在FPGA中使用软内核可以为设计人员提供比使用标准处理器更大的自由。可以从www.opencores.org下载多种现成的外设。快速傅立叶变换（Fast Fourier Transform，FFT）、数字滤波器和复杂加密算法等算法可以在硬件而非软件中实现，这有助于提高处理速度。由于几乎可以将外设连接到任何引脚，因此PCB布线要容易得多。

6.3 运行程序所需的电路板

如果仅用于软件仿真和软件调试，则不需要电路板。

使用的电路板包括：

Digilent Nexys A7 Nexys A7 ECE FPGA Trainer Board
Digi-Key 1286-1081-ND（265.00美元）

也可以使用它的前一代产品（现在已经过时的Nexys4 DDR电路板），因为它们非常相似。

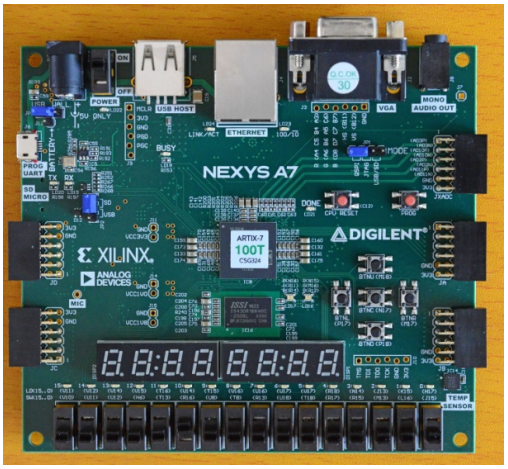


图70：Digilent Nexys A7电路板

6.4 所需软件

需要以下程序：

程序	用途
Xilinx Vivado 2019.2	编译构成RISC-V内核的Verilog .v和System Verilog .sv文件。还会将生成的.bit文件下载到硬件
Visual Studio Code（VSCode）	用于C语言和汇编语言软件开发的IDE
PlatformIO	VSCode的插件，可为RISC-V处理器提供支持
Open OCD	开源片上调试器
RISC-V Toolchain	编译RISC-V项目所需的编译器和其他文件
Verilator	用于Verilog .v和System Verilog .sv文件的硬件仿真器和处理器
Whisper	Western Digital的指令集仿真器（Instruction Set Simulator，ISS）。允许在计算机上运行和调试代码，无需硬件
Digilent Board Files	Digilent Nexys A7电路板的特定配置文件
GTKWave	波形查看器，用于查看底层系统时序
RVfpga	软件内核、代码示例和Verilog/System Verilog项目
RVfpga Labs	20个实验

表4：所需程序

目前，所有程序仅在Linux上运行，但Windows和Mac支持正在开发中。完整的下载和安装时间约为4小时。

6.5 运行RISC-V实现

要在Xilinx FPGA中的RISC-V上运行代码，需要执行两项任务：

1. 使用.bit文件配置FPGA硬件。
2. 编译代码并将.elf文件下载到FPGA中。

6.5.1 FPGA硬件和软件组件之间的关系

图71说明了如何使用Xilinx Vivado处理Verilog .v和System Verilog .sv文件并将其转换为可编程到FPGA中的.bit文件。

C代码或汇编代码使用PlatformIO进行编译，.elf文件下载到支持其运行和调试的FPGA中。

通过单个共享USB端口与Digilent Nexys A7电路板通信。不能同时使用Vivado和Visual Studio Code与电路板通信。

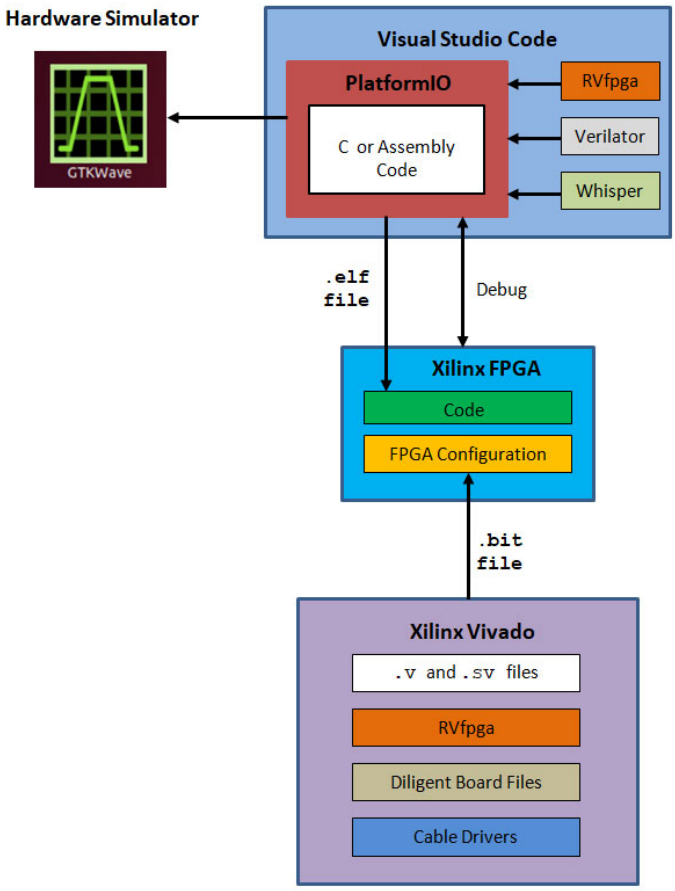


图71：Visual Studio、FPGA和Xilinx Vivado的关系

6.6 Western Digital SweRV内核

尽管有各种各样的RISC-V内核可用，但其中很多是学术研究的一部分，并未经过完全认证。并不是说这些内核不能正常工作，但是对于工业产品，RISC-V内核必须经过充分验证和认证，以避免潜在的高昂产品责任成本。

Western Digital（2019年收入165.7亿美元）开发了一个系列共三款SweRV内核（分别为EH1、EH2和EL2），经认证可用于商业产品。

6.6.1 SweRV内核列表

内核名称	流水线阶段	线程	尺寸（mm）@TSMC	CoreMark /MHz
EH1	9	单	0.110@28 nm	4.9
EH2	9	双	0.067@16 nm	6.3
EL2	4	单	0.023@16 nm	3.6

表5：Western Digital SweRV内核。图片来源：Western Digital

此处的尺寸是指从台湾积体电路制造股份有限公司（Taiwan Semiconductor Manufacturing Company，TSMC）获得的数据。

这三个内核均为RV32IMC，这意味着整数指令集、32位和32个寄存器、整数乘法和除法以及压缩指令（16位）。

6.6.2 SweRV内核之间的区别

EH1和EH2内核均具有完整的流水线功能，区别是EH1支持单线程，而EH2支持双线程。EH2的规范更高（6.3 CoreMark/MHz），但架构也更复杂。

如果尺寸和成本十分重要，则EL2内核是一个不错的选择，但性能会下降到3.6 CoreMark/MHz。它使用具有4级而不是9级的简化流水线。

EH1内核已经在Digilent Nexys A7电路板上实现并经过了测试，因此已被选择用于本指南的动手实验部分。未来计划添加性能更高的EH2内核和资源较少的EL2内核。

6.6.3 SweRV EH1内核详细信息

SweRV EH1内核的框图如图72所示：

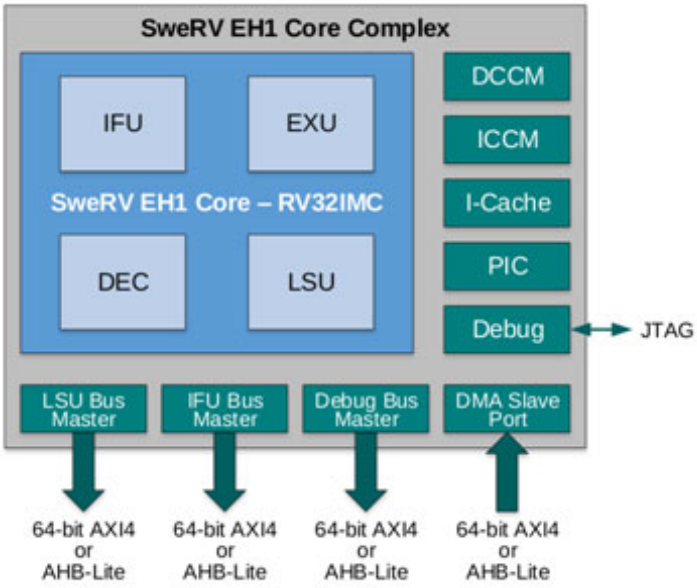


图72：SwerRV EH1内核组合。图片来源：Western Digital

内核分为4个单元，分别为取指单元（IFU）、执行单元（EXU）、解码单元（DEC）和装入/存储单元（LSU）。

6.6.4 SweRV EH1的各个组件

缩写	说明
IFU	取指单元
EXU	执行单元
DEC	解码单元
LSU	装入存储单元
DCCM	紧密耦合的数据存储器
ICCM	紧密耦合的指令存储器
I-Cache	指令高速缓存
PIC	可编程中断控制器
AXI	高级可扩展接口
AHB	高级高性能总线

表6：SwerRV EH1组件

AXI4和AHB-Lite是两种行业标准互连总线。JTAG用于代码下载和调试。

6.7 SweRVolf内核

就其本身而言，SweRV EH1只是一个基本的计算机内核，它需要其他组件才能具有实际用途。Olof Kindgren为SweRV编写了一组包装程序，称为SweRVolf。RVfpga的作者使用一些有用的附加外设扩展了SweRVolf，并重新组织了HDL源。

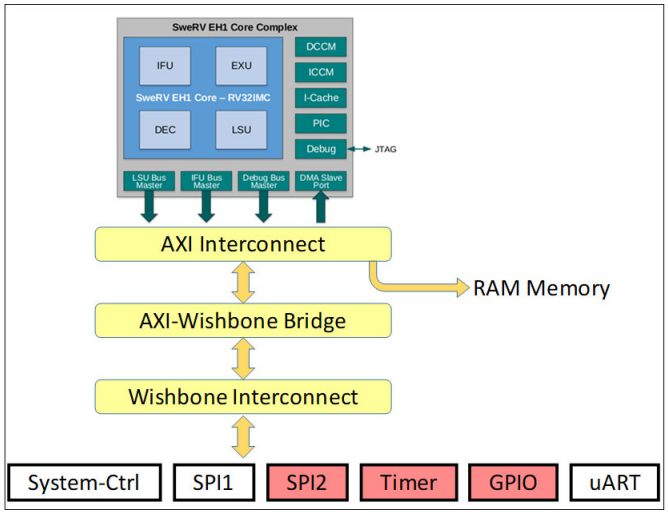


图73：SwerRVolf内核。图片来源：Imagination Technologies UP

6.7.1 扩展SweRVolf组件

扩展SweRVolf实现分为以下组件：

- Wishbone互连：Opencores互连，比AXI更简单。
- axi2wb：AXI转Wishbone总线转换器。
- GPIO：来自Opencores的通用输入输出，具有64个输入/输出端口。
- 定时器：来自Opencores
- SPI：开源串行外设接口控制器（从 https://opencores.org/projects/simple_spi 获取）。
- UART：开源UART控制器（从 https://opencores.org/projects/uart16550 获取）。
- 引导ROM：引导ROM包含第一阶段的自举程序。系统复位后，SweRV将开始从该区域获取初始指令。
- RAM：SweRVolf内核不包含存储器控制器，但保留了其存储器映射的前128 MB并且支持访问AXI总线，以便用户可以包含RAM存储器。
- SPI闪存：也可以使用上一部分中介绍的SPI控制器包含SPI闪存（或四路SPI）。

6.7.2 SweRVolf存储器映射

所有组件的存储器映射如下：

内核映射	存储器地址范围
引导ROM	0x80000000 - 0x80000FFF
系统控制器	0x80001000 - 0x8000103F
SPI1	0x80001040 - 0x8000107F
SPI2	0x80001100 - 0x8000113F
定时器	0x80001200 - 0x8000123F
GPIO	0x80001400 - 0x8000143F
UART	0x80002000 - 0x80002FFF

表7：SweRVolf存储器映射

6.8 SwerRVolf Verilog和System Verilog文件

图74的文件层级以Verilog .v和System Verilog .sv格式显示。这些均由RVfpga软件包提供。

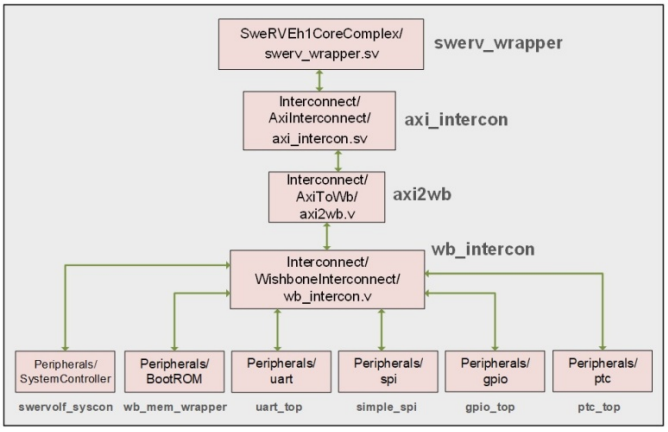


图74：SweRVolf文件结构。图片来源：Imagination Technologies UP

6.9 Nexys A7的SwerRVolf层级

SweRVolf项目的所有组件都在RVfpga软件包中提供，可以下载。

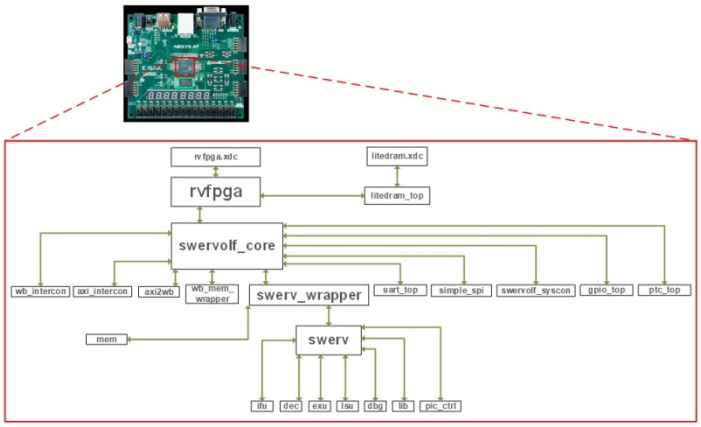


图75：Nexys A7中的SweRVolf项目层级。图片来源：Imagination Technologies UP

图76：一些经过认证的开源内核。来源：www.opencores.org

6.10 增强SweRVolf实现

可以通过添加其他外设来扩展原始SweRVolf实现。

6.10.1 Opencores提供的外设

SweRVolf是一种开放式规范，所有源代码都可以从Github下载。这意味着可以对其进行修改，而且可以添加新外设。`www.opencores.org/projects`提供了各种各样的项目。图76给出了一些经过认证的开源内核的示例：

6.10.2 实验中添加的外设

在实验中，为Nexys A7电路板添加了以下外设：

- 5个按钮（通过新GPIO实现）
- 三色LED（使用PWM）
- 温度传感器（使用I²C）
- 7段显示屏驱动器
- SPI加速计驱动器

7 软内核软件安装

7.1 下载RVfpga

在安装用于软内核开发的软件之前，需要下载RVfgpa软件包，因为它包含所需的许多文件。有关详细信息，请参见第6.1节。

下载所有文件。RVfpga软件包的理想安装位置是主目录，例如/home/myusername/RVfpga，其中“myusername”是实际的计算机名。当一个示例如下时：

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

只需输入：

```
~/RVfpga/
```

请确保所有文件都具有读/写权限。

以下示例使用了RVfpga的当前版本，可能会有所变化。建议参考Imagination Technologies“RVfpga：入门指南”来了解最新信息。

7.2 关于Xilinx Vivado

这是到目前为止最耗时的操作，因为在Ubuntu 18.04 Linux中需要设置多个程序。Windows和Mac支持目前正在开发。Xilinx下载和安装需要约2.5小时。

选择Xilinx有以下几个原因：

1. Xilinx处于FPGA市场的高端，具有极为丰富的高规格产品。
2. 从职业角度和市场技能来看，FPGA技术对Xilinx的需求最为广泛。只需快速搜索招聘网站即可确认这一点。
3. SweRVolf是使用Xilinx编写的。
4. SiFive也为其内核使用Xilinx。

7.3 有关在Linux中安装Xilinx Vivado的重要信息

在执行Xilinx下载之前（如果尚未完成），需要在计算机上安装Ubuntu 18.04。Ubuntu分区至少需要100 GB。**不要使用Ubuntu 20.04**。

Xilinx指定了哪些版本的Ubuntu与其软件兼容，而Ubuntu 20.04目前不在兼容版本之列。

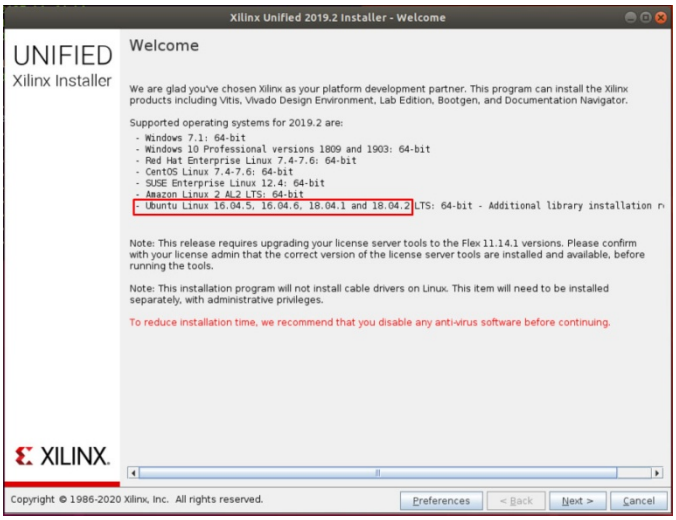


图77：Xilinx统一安装程序兼容性

不要像作者一样下载Ubuntu20.04，结果在2个半小时后才发现它不兼容且无法运行。该过程将需要从头开始，删除所有内容，然后安装正确的版本Ubuntu 18.04。

7.3.1 Xilinx Vivado系统要求

Xilinx Vivado的大小为16 GB（因此下载时间较长），并且需要56 GB的安装空间（因此分区大小为100 GB）。如果磁盘空间不足，将产生错误。

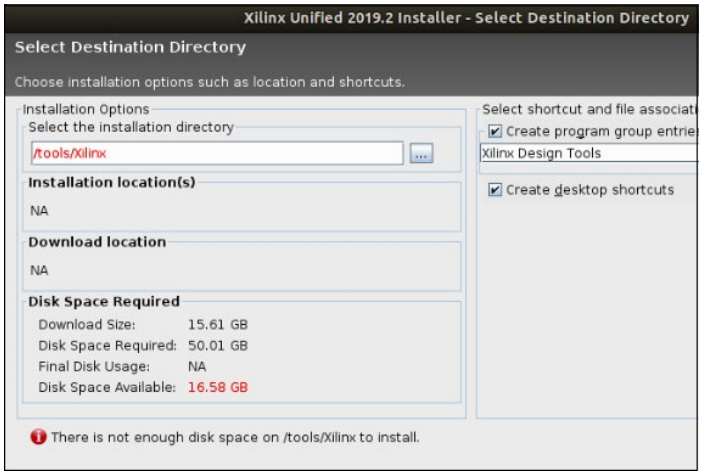


图78：Xilinx Vivado系统要求

7.4 在Linux中安装Xilinx Vivado

该程序用于为软内核修改和编译Verilog .v和System Verilog .sv文件，并将其下载到Digilent Nexys A7电路板。

如果没有Digilent Nexys A7电路板，或者该电路板仅用于运行计算机仿真，则可以在以后安装Vivado。

7.4.1 下载Xilinx软件

要下载Xilinx软件，需要创建一个Xilinx帐户（如果尚不存在）。

转到https://www.xilinx.com/support/download.html，选择2019.2版本进行下载

单击“Xilinx Unified Installer 2019.2: Linux Self Extracting Web Installer”（Xilinx统一安装程序2019.2：Linux自提取Web安装程序）进行下载。

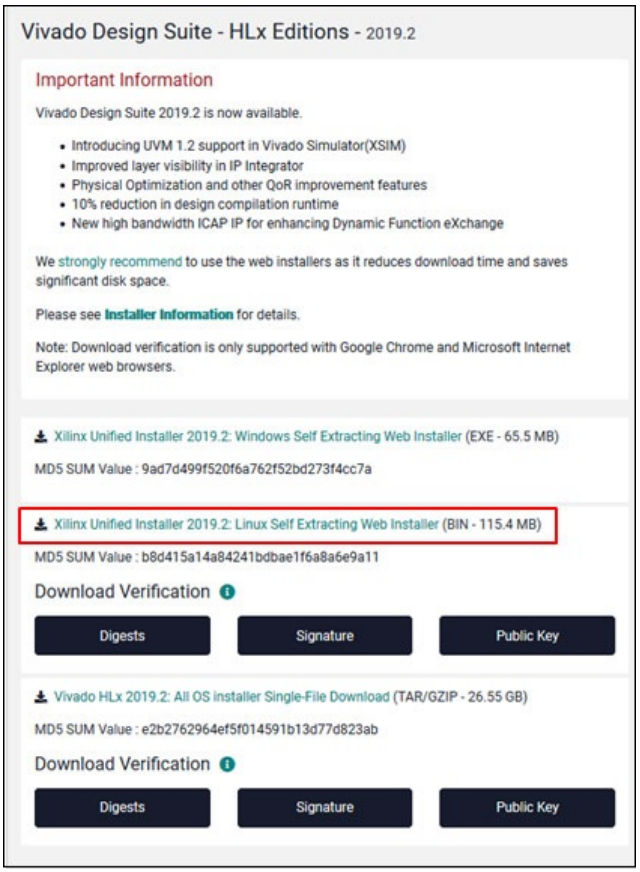


图79：Xilinx统一安装程序2019.2

7.4.2 更改二进制文件的权限

当Xilinx_Unified_2019.2_1106_2127_Lin64.bin文件下载到/Downloads目录后，它是无法执行的。

使用“*Ubuntu Files*”（Ubuntu文件）  工具，右键单击文件名称，然后选择“*Properties-Permissions*”（属性-权限），使其变为可执行状态。单击“*Allow executing file as program*”（允许将文件作为程序执行）。

7.4.3 运行二进制文件

打开一个*Ubuntu终端*  ，输入以下内容将其设置为根：

```
sudo su
```

返回到“*Ubuntu Files*”（Ubuntu文件）  工具，将以下文件拖放到“*Terminal*”（终端）窗口：

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

在*Ubuntu终端*中按下回车键，运行二进制文件。

7.4.4 选择要安装的Vivado产品

在“*Select Product to Install*”（选择要安装的产品）窗口中，选择*Vivado*而不是*Vitis*。

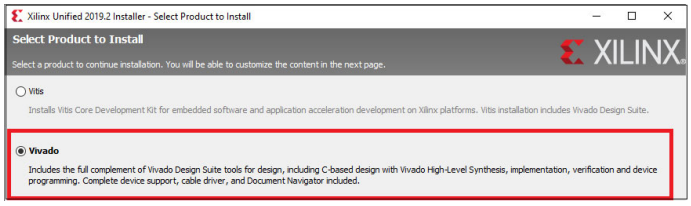


图80：选择Vivado

7.4.5 选择WebPACK

在“*Select Edition to Install*”（选择要安装的版本）窗口中，选择*Vivado HL WebPACK*，这是免费版本。

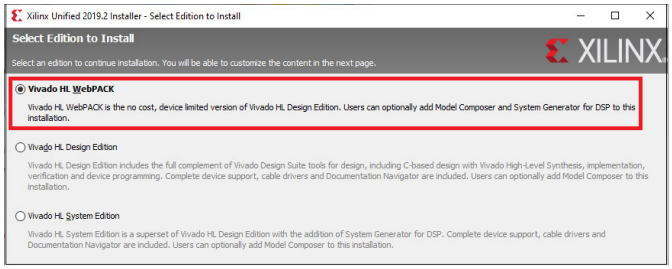


图81：选择要安装的版本

按照其余步骤完成安装，并使用默认设置。

7.4.6 安装Digilent Nexys A7电路板的电缆驱动程序

打开一个*Ubuntu终端*  并导航到：

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

然后输入：

```
sudo ./install_drivers
```

7.4.7 安装电路板文件

转到https://github.com/Digilent/vivado-boards。

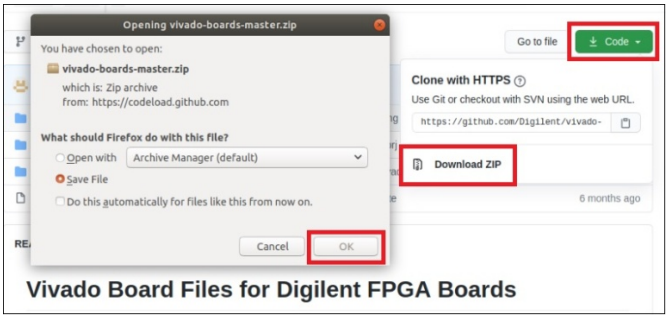


图82：下载Vivado Boards Master

下载.zip文件并使用“*Ubuntu Files*”（Ubuntu文件）工具将其解压。

在*Ubuntu终端*中，导航到：

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

需要将此处的文件复制到Vivado board_files目录中。输入：

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

重要的电路板文件是nexys-A7-100t。

7.4.8 在Linux下运行Vivado

打开一个*Ubuntu终端*。

导航到目录：

/tools/Xilinx/Vivado/2019.2，然后输入：

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

7.5 安装Visual Studio Code和PlatformIO

所需时间：约20分钟。

Visual Studio Code（VSCode）是用于软件开发和调试的IDE。

PlatformIO是VSCode的插件，可为RISC-V和许多其他处理器提供支持。

7.5.1 下载Visual Studio Code

转到 https://code.visualstudio.com/Download 。单击.deb下载文件。文件大小目前为60.5 MB

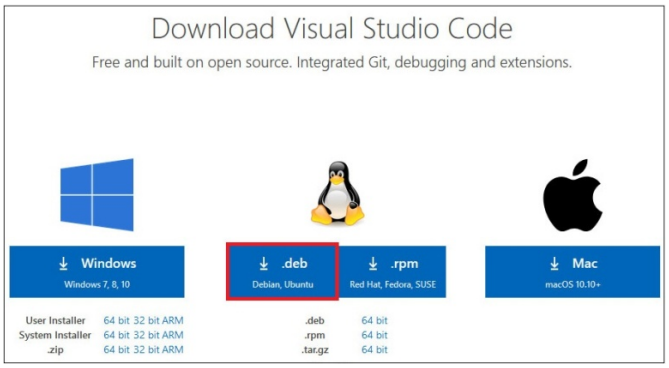


图83：下载Visual Studio Code

7.5.2 提取Visual Studio Code

打开一个*Ubuntu终端*并输入：

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

要运行VSCode，输入：

```
code
```

7.5.3 安装PlatformIO

PlatformIO为RISC-V提供特定支持。

在*Ubuntu终端*中输入以下内容以加载一些必要的实用程序：

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

如果VSCode尚未运行，则在搜索菜单中输入`Visual Studio Code`并选中它来对其进行定位。或者，单击*Visual Studio Code*图标。

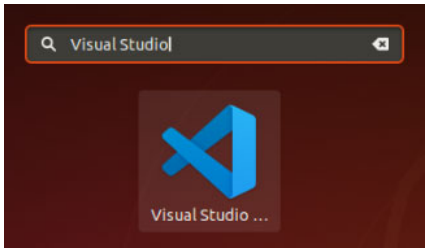


图84：Visual Studio Code图标

7.5.4 VS Code中的PlatformIO扩展

在VSCode中，单击“*Extensions*”（扩展）图标 。

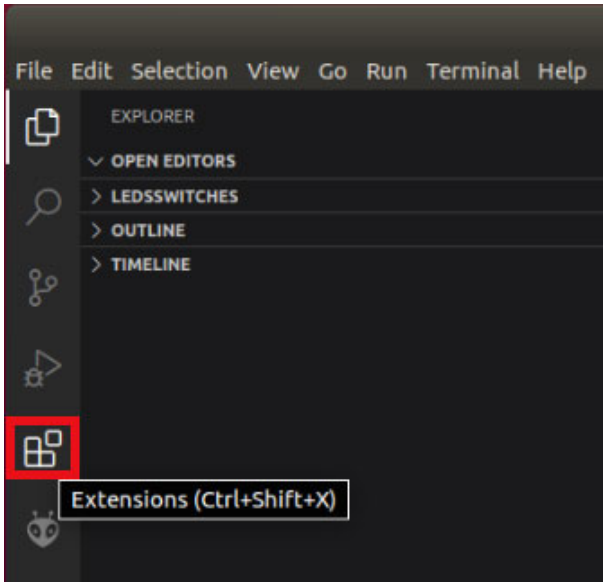


图85：VSCode扩展栏

7.5.5 安装PlatformIO扩展功能

在搜索框中输入`platformio`。转到*PlatformIO IDE*，然后单击“*Install*”（安装）按钮。

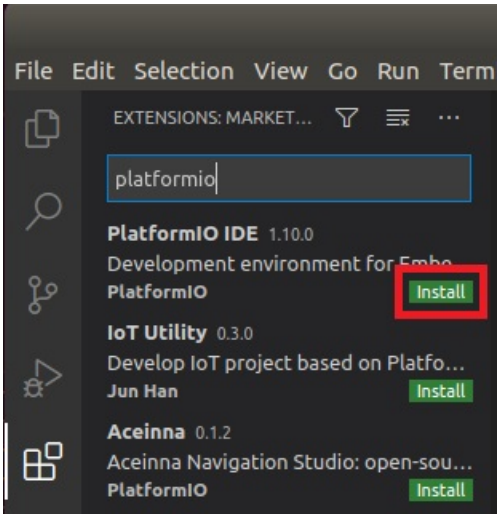


图86：PlatformIO IDE扩展

7.5.6 重新载入PlatformIO

底部的“*OUTPUT*”（输出）窗口提供有关安装进度的信息。

完成后，单击“*Reload Now*”（立即重新载入），这样PlatformIO内核便会安装到VSCode中。

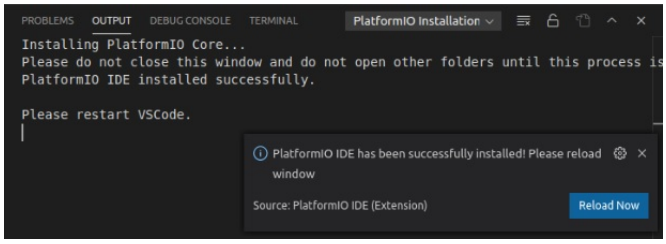


图87：安装PlatformIO后立即重新载入

7.5.7 安装Chips Alliance软件包

下一步是选择处理器目标。本例中为SweRVolf，它是已下载RVfpga软件包的一部分。

通过单击Platformio  按钮（位于左侧栏上）查看“*Quick Access*”（快速访问）菜单，然后单击“*PlatformIO Core CLI*”（PlatformIO内核CLI）选项。

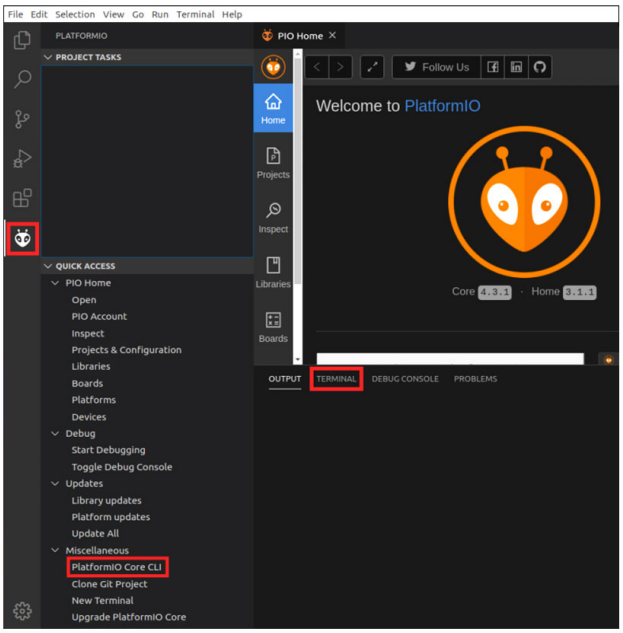


图88：选择PlatformIO内核目标

在底部窗口中，从“*OUTPUT*”（输出）窗口更改为“*TERMINAL*”（终端）窗口。这允许在VSCode中输入命令。

7.5.8 安装Chips Alliance平台

通过在“*TERMINAL*”（终端）上运行以下命令来安装Chips Alliance平台：

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

其中[RVfpgaPath]是RVfpga文件的保存位置。例如，它可能位于/home/myname/中，其中myname是实际的计算机名。

该软件包中包含预编译的RISC-V工具链、用于RISC-V的OpenOCD调试器、Verilator、Whisper、JavaScript和Python脚本。此外，还提供了RVfpga位文件和示例代码。

7.5.9 SweRVolf是否已加载的可选检查

要确认swervolf_nexys是否已加载，在“*TERMINAL*”（终端）窗口中输入：

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

然后向下滚动到swervolf_nexys

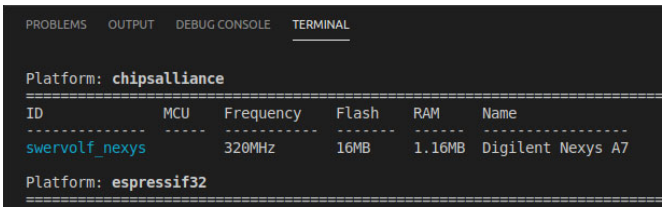


图89：检查swervof_nexys是否已加载

7.5.10 关闭并重新打开VSCode

激活PlatformIO后，请关闭并重新打开VSCode。

注：通常情况下，.platformio文件为隐藏状态。

要在*Ubuntu终端*  中搜索.platformio，输入：

```
ls -a
```

这会使隐藏的文件可见。

或者，使用“*Ubuntu Files*”（Ubuntu文件）工具 显示隐藏文件。

7.6 安装Verilator和GTKWave

所需时间：约20分钟。

Verilator是适用于PlatformIO使用的Verilog和System Verilog文件的硬件仿真器。

要安装将在Linux中本机运行的Verilator，需打开一个*Ubuntu终端* 。

首先安装默认情况下未安装的必要实用程序：

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

安装Verilator程序

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

最后安装波形查看器

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

图90：GTK Wave图标

7.6.1 典型Verilator波形

Verilator用作硬件仿真器时十分复杂，超出了本文的范围。“RVfpga：入门指南”中给出了完整的详细信息。但是，时钟和取指单元（Instruction Fetch Unit，IFU）的典型波形为：

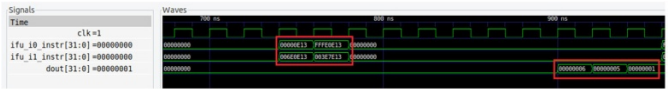


图91：典型Verilator波形。

7.7 使用Whisper在PlatformIO中运行C程序

Western Digital有一个名为Whisper的指令集仿真器（ISS），其开发用于在没有任何硬件的情况下对SweRV RISC-V内核进行软件验证。它是RVfpga下载包的一部分。

7.7.1 打开文件夹

打开VSCode并进入PlatformIO。

在PlatformIO的顶部菜单栏上，单击“*File*”（文件），然后单击“*Open Folder*”（打开文件夹）。无需实际打开文件。

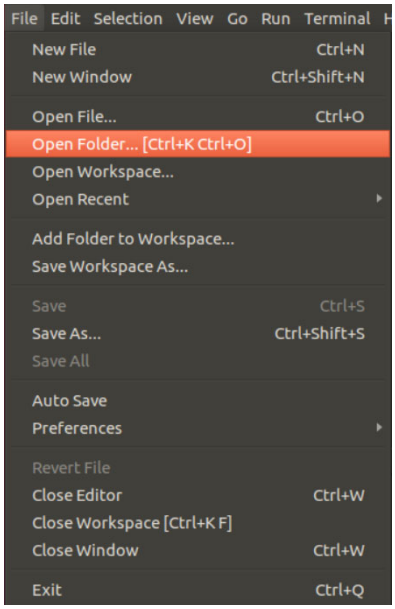


图92：打开仿真文件的文件夹

7.7.2 Vector Sorting Whisper示例代码

选择目录VectorSorting，然后单击“*OK*”（确定）。

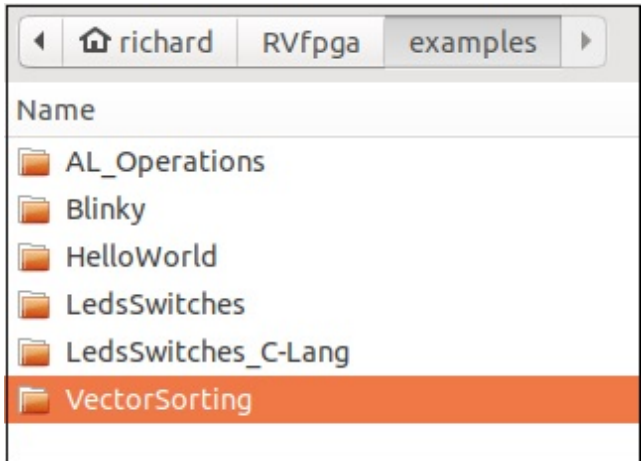


图93：VectorSorting文件夹

7.7.3 修改platformio.ini

要使用Whisper，必须修改platformio.ini文件。打开该文件。

取消注释行：

```
debug_tool = whisper
```

然后保存它。

重要信息：确保debug_tool与上面的字符对齐，并且前面没有空格。图94给出了存在意外空格时产生的错误。

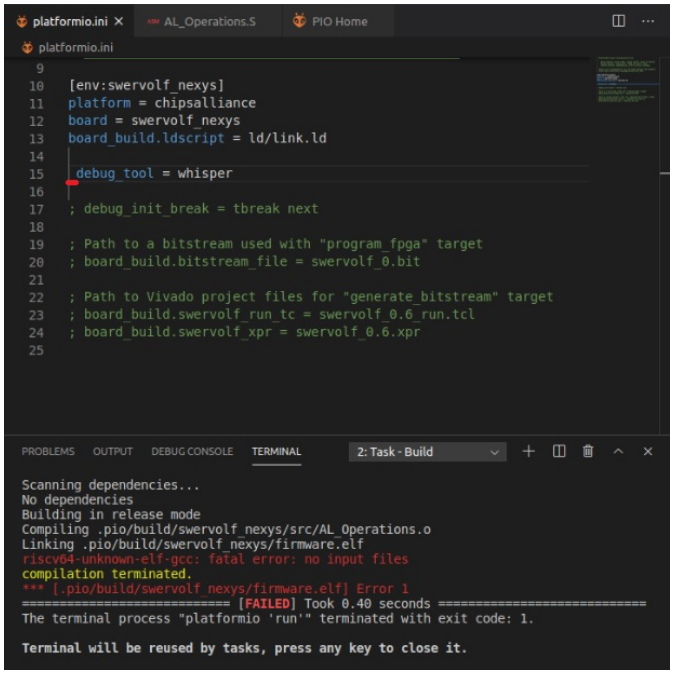


图94：错误的platformio.ini设置

7.7.4 使用Whisper运行仿真

通过单击第10行左侧放置一个断点。

通过依次单击  和 

来启动调试器。

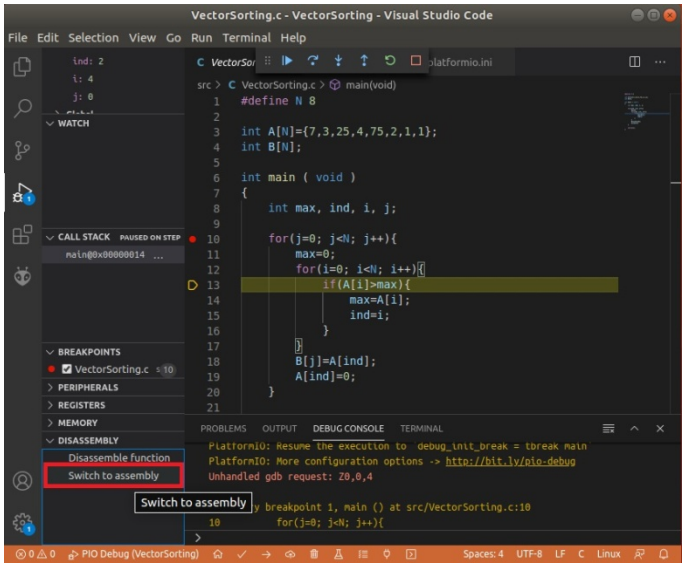


图95：显示程序RISC-V指令和第10行的断点

现在可以使用键盘功能键F10单步执行代码。

单击画面左下角的“*Disassembly*”（反汇编）可显示RISC-V指令。

7.7.5 查看RISC-V汇编语言

C编译器生成的汇编语言如图96所示。

要返回到C源代码，单击“*Switch to code*”（切换到代码）。

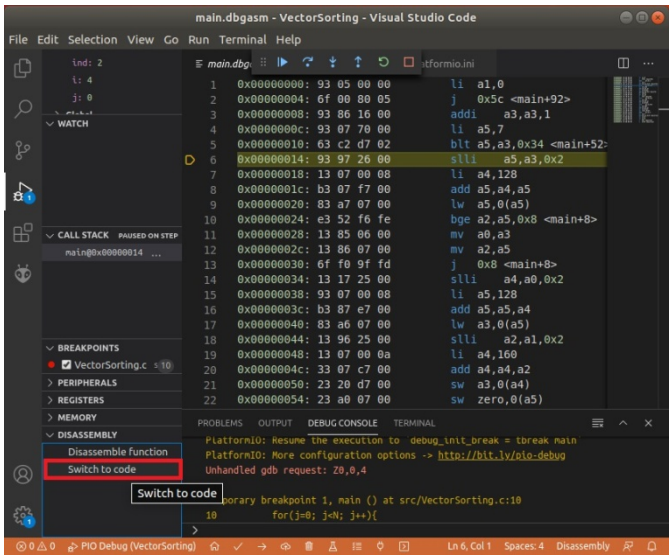


图96：RISC-V汇编语言视图

有关调试的更多详细信息，请参见“RVfpga：入门指南”。

8 编译SweRVolf内核并将其下载到FPGA

所需时间：约5分钟

8.1 编译SweRVolf内核

好消息是不需要该部分，这要归功于马德里康普顿斯大学的RVfpga团队。作为RVfgpa软件包的一部分，Xilinx Vivado项目提供了`RVfpga.bit`文件，该文件可直接下载到Digilent Nexys A7硬件中。

8.2 连接Nexys A7电路板

使用随附的USB线缆插入Nexys4 A7电路板。将电源开关滑动到开启位置，因为出厂默认设置为关闭。

8.3 将内核下载到Digilent Nexys A7中

这需要先下载RVfpga软件包，因为其中包含`RVfpga.bit`文件。请参见第6.1节。

8.3.1 在Vivado中打开硬件管理器

按第7.4.8节所示打开Vivado。

单击“*Open Hardware Manager >*”（打开硬件管理器 >）

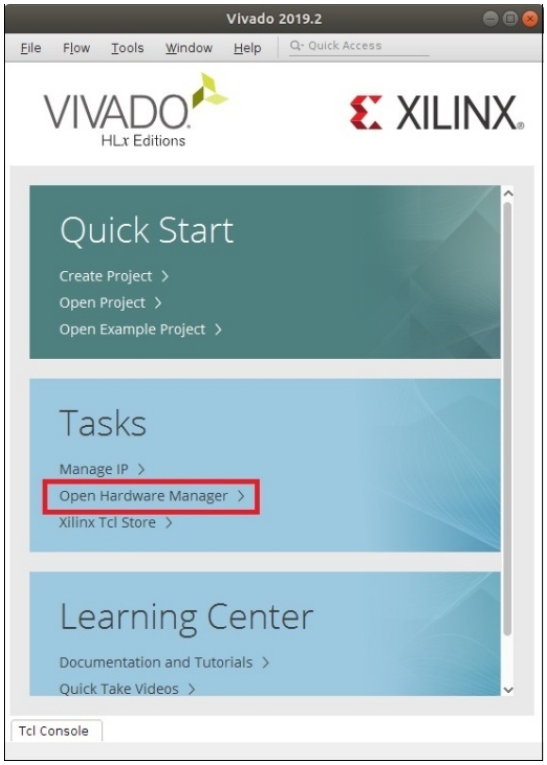


图97：打开Vivado硬件管理器

8.3.2 打开硬件目标

硬件管理器将自动检测到硬件未打开。单击“*Open Target*”（打开目标），然后单击“*Auto-detect*”（自动检测）。

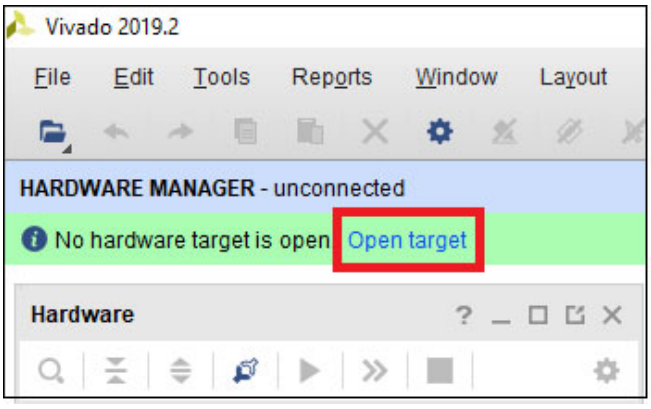


图98：打开硬件目标

8.3.3 编程FPGA配置

单击“*Program device*”（编程器件）。

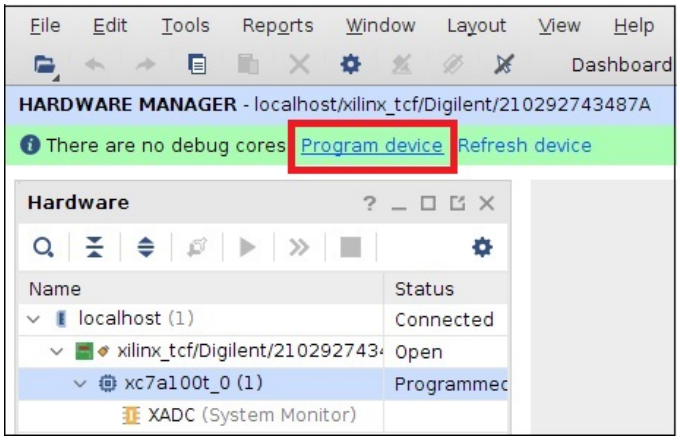


图99：编程FPGA配置

8.3.4 选择比特流文件

对于比特流文件，请选择：

```
[RvfpgaPath]/RVfgpa/src/RVfpga.bit
```

“*Debug probes file*”（调试探头文件）可以留空。单击“*Program*”（编程）。电路板应在大约10秒钟内编程。

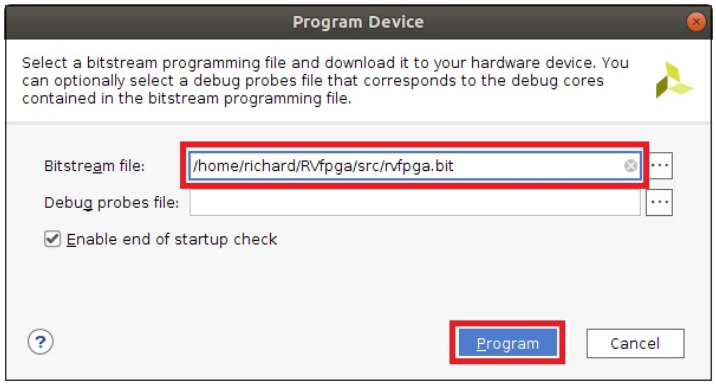


图100：下载比特流文件

8.3.5 关闭Vivado

这是重要的一步。退出Vivado，以便使USB端口空闲，否则无法使用VSCode下载和运行程序。

单击右侧的“*Close Hardware Manager*”（关闭硬件管理器）图标 。

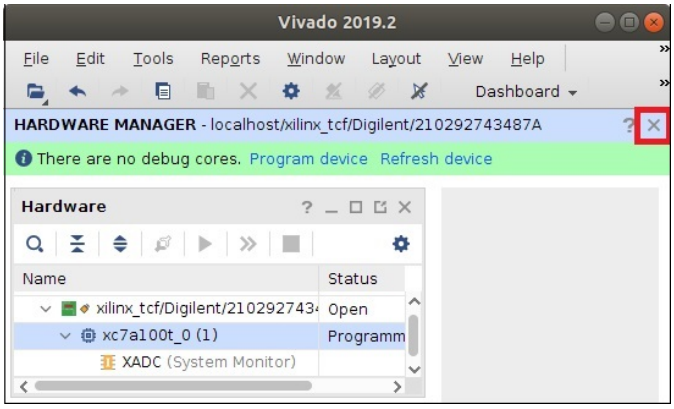


图101：退出Vivado

8.4 在NexysA7电路板上运行程序

所需时间：约5分钟。

8.4.1 打开要运行的程序

在搜索栏中输入`Visual Studio Code`并单击*VSCode* 图标，打开VSCode。

在顶部菜单栏上，单击“*File*”（文件），然后单击“*Open Folder*”（打开文件夹）。

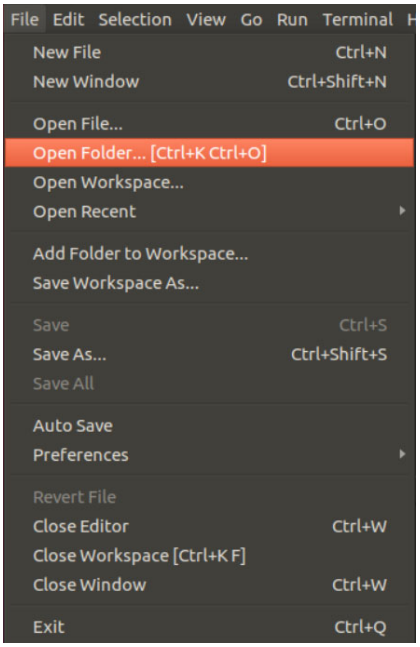


图102：打开文件夹

无需打开文件。

8.4.2 选择程序LedsSwitches

导航到RVfpga/examples，然后单击LedsSwitches。其他示例可以稍后再试。

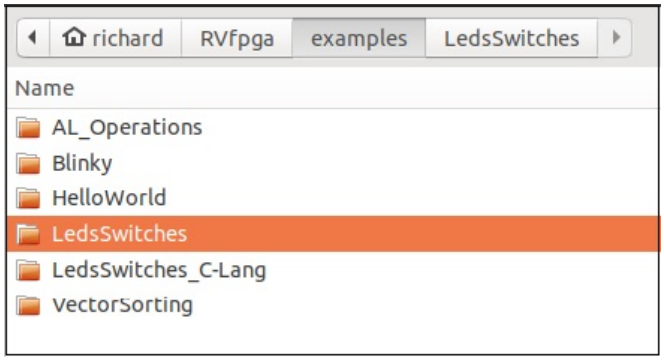


图103：LED和开关程序

8.4.3 打开LED和开关程序

这是一个最小的汇编语言程序，它会持续读取16个开关，然后将值输出到16个LED。RVfpga示例中还有一个C代码版本。

展开*>src*，然后单击LedsSwitches.S，以便使汇编代码可见。

单击画面左下方的“*PlatformIO Build*”（PlatformIO编译）图标 。

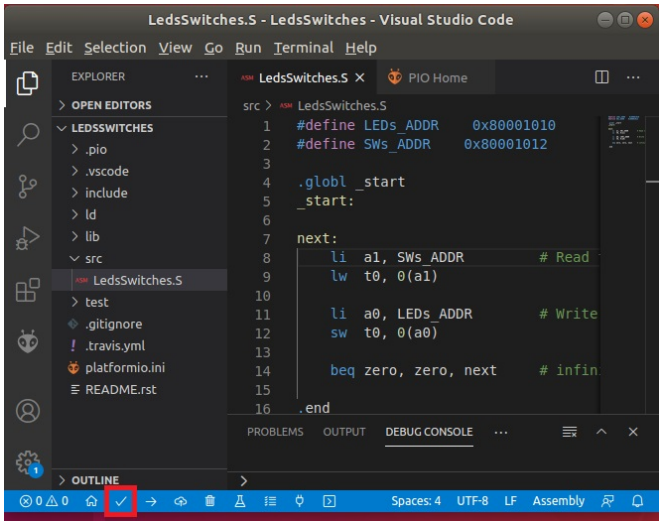


图104：编译LedsSwitches程序

8.4.4 运行LedsSwitches程序

从工具栏中选择“*Run*”（运行），然后选择“*Start Debugging*”（开始调试），或者按下键盘功能键F5。程序将持续运行。

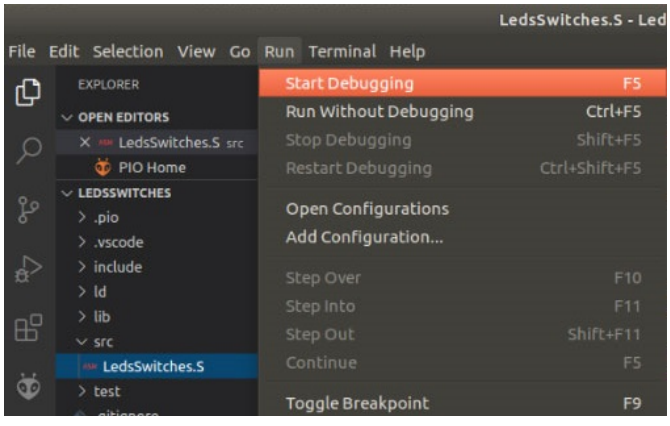


图105：开始调试LED和开关程序

8.4.5 运行RISC-V内核的Nexys A7电路板 - 由开关控制的LED

操作电路板底部的滑动开关。绿色LED可以单独点亮和熄灭。

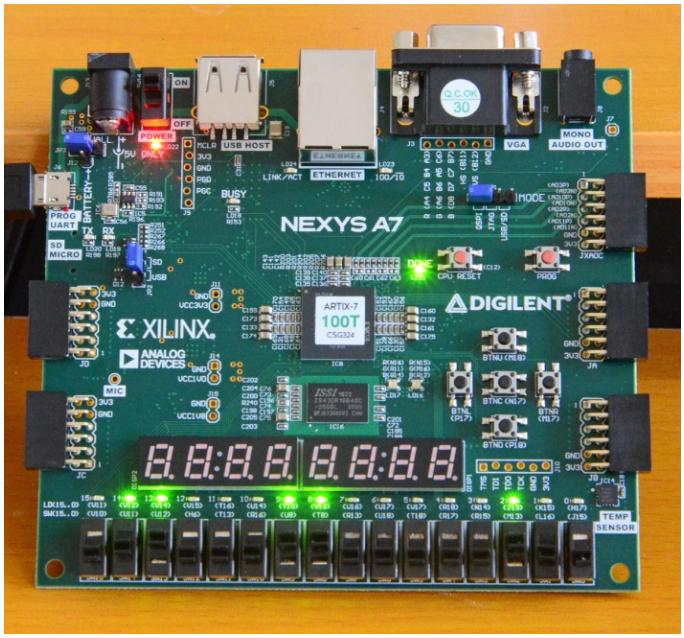


图106：运行RISC-V内核的Nexys A7电路板

请注意：在最新版本的RVfpga.bit上，7段显示屏显示0000 0000。为了清楚起见，这里给出了一个简单的版本。

9 使用RISC-V进行产品开发

在本指南中，我们使用Kendryte K210和SiFive FE310-G002这两款SoC RISC-V处理器完成动手实验。本章介绍这两款器件在安装到用户电路板上时所能提供的功能，这些功能可用于进一步开发或者批量生产的产品。

9.1 Kendryte K210

这款处理器在Seeed Technologies Co Ltd Maix BiT电路板上使用。它是一款功能强大的处理器，价格适中。

9.1.1 Kendryte K210处理器内核

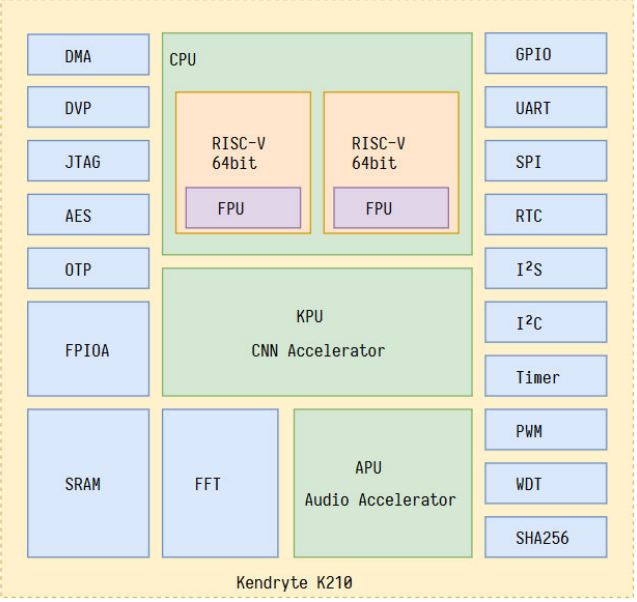


图107：Kendryte K210架构概述。图片来源：Kendryte K210数据手册。

它包含两个（而非一个）以400 MHz运行的RISC-V 64位浮点处理器。此外，还有一系列有用的内部组件和接口。

缩写	说明
CNN	神经网络
DMA	直接存储器地址
DVP	数字视频端口
JTAG	编程和调试接口
AES	高级加密标准
OTP	一次性可编程闪存
FPIOA	现场可编程输入和输出阵列（可将GPIO映射到任何引脚）
SRAM	静态随机存取存储器
FFT	快速傅立叶变换
GPIO	通用输入/输出3V3和1V8
UART	串行端口
SPI	串行外设接口
RTC	实时时钟
I2S	集成电路内置音频总线。音频编解码器的标准接口
I2C	用于低速外设的集成电路间总线
PWM	脉冲宽度模块
WDT	看门狗定时器
SHA256	用于验证数据完整性的安全哈希算法

表8：Kendryte K210缩写

使GPIO成为双路3V3/1V8非常有用。某些无线电模块（例如LTE/GSM模块）仅支持1V8操作。如果没有双电压GPIO，将需要额外的电平转换器，但这需要额外的成本。

现场可编程输入/输出阵列（Field Programmable Input/Output Array，FPIOA）使印刷电路板（Printed Circuit Board，PCB）的布局更加简单，因为可以对引脚进行分组来简化布线。

Kendryte K210处理器仅提供球栅阵列（Ball Grid Array，BGA）封装，这使其只适用于机器布局。

9.1.2 Seeed Technologies MaixPy BiT实现

另外，对于实验或小批量生产，基本的MaixPy BiT可以轻松安装到面包板上或者使用安装引脚焊接到另一个PCB上。

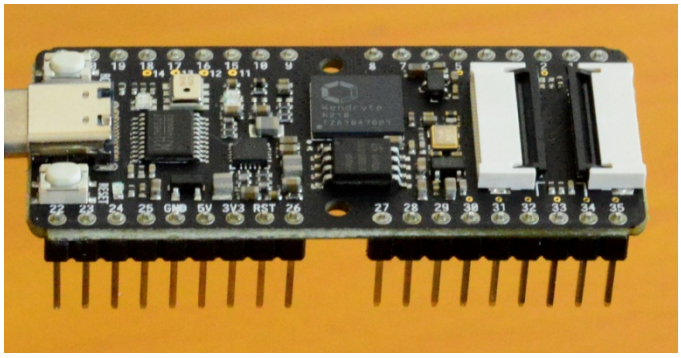


图108：带安装引脚的Maix BiT电路板。Seeed Technologies Maix-I模块

9.1.3 Maix-I

在PCB上实现任意处理器时的一个问题是，要使处理器正常工作，需要很多附加组件 – 去耦电容、晶振、直流-直流转换器、电感和复位电路等。

为了减轻硬件工程师的工作负担，提供了两个Maix-I模块：

MAIX-I Digi-Key 1597-1717-ND（8.91美元）和

Maix-I with Wi-Fi Digi-Key 1597-1715-ND（10.82美元）。

对于小批量生产或早期开发板，这两个模块极具吸引力。

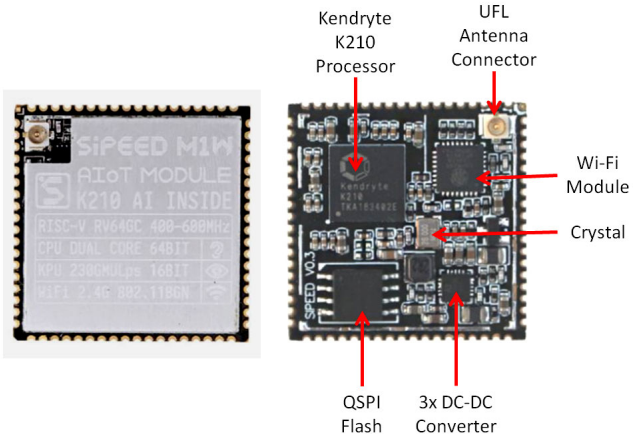


图109：带Wi-Fi的Maix-I模块。图片来源：Seeed Technologies Co Ltd（已修改）。

两个Maix-I模块不仅包含Kendryte K210处理器，还包含四路SPI闪存、晶振、直流-直流转换器以及可选的Wi-Fi模块（价格稍高）。

可能的应用是由主处理器通过一个简单的串行端口驱动的专用图形处理器。由于可以使用MicroPython开发项目，因此软件开发时间会很短。

9.1.4 对MaixPy BiT的Visual Studio Code支持

MaixPy BiT使用MicroPython进行编程，但也可以选择C或C++。

此外，Xilinx FPGA软内核使用的PlatformIO还支持MaixPy BiT电路板和Maix-I模块上使用的Kendryte K210。但是，作者还未尝试过。

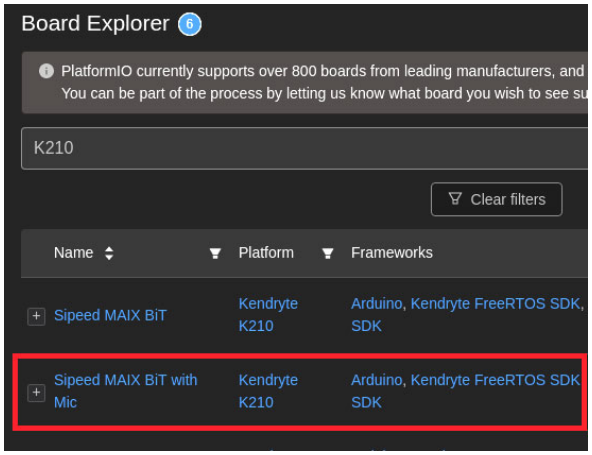


图110：对MaixPy BiT的VSC PlatformIO支持

9.2 SiFive

RED-V Redboard上使用的处理器是SiFive Freedom Everywhere FE310-G002。

9.2.1 Freedom Everywhere FE310-G002 RISC-V引脚排列

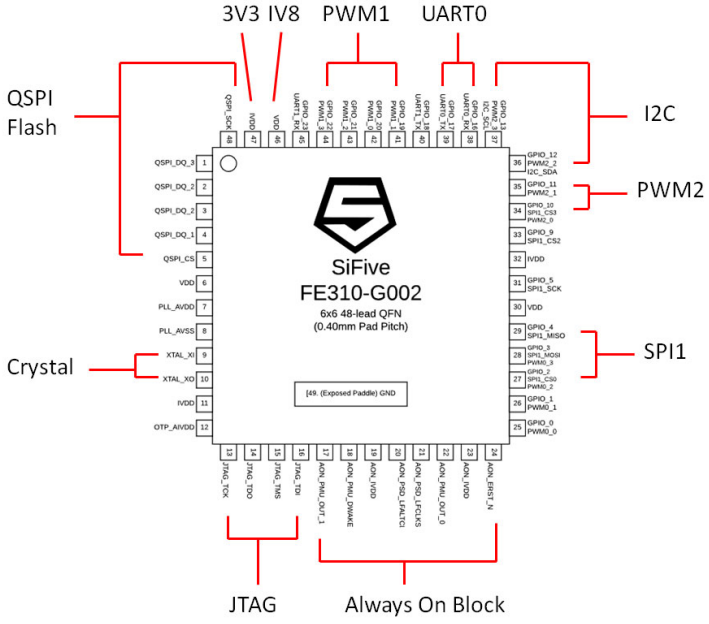


图111：SiFive FE310-G002引脚排列。图片来源：SiFive FE310-G002数据手册（已修改）。

FE310-G002采用48引脚QFN封装，运行频率最高为320 MHz。

9.2.2 FE310-G002 GPIO引脚多路复用功能

Name	Pin	GPIO	PWM	SPI	UART	i2C
GPIO_0	25	0 I/O	PWM0_0 O			
GPIO_1	26	1 I/O	PWM0_1 O			
GPIO_2	27	2 I/O	PWM0_2 O	SPI1_SS0		
GPIO_3	28	3 I/O	PWM0_3 O	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 O	SPI1_SS3		
GPIO_11	35	11 I/O	PWM2_1 O			
GPIO_12	36	12 I/O	PWM2_2 O			I2C0_SDA
GPIO_13	37	13 I/O	PWM2_3 O			I2C0_SCL
GPIO_16	38	16 I/O			UART0_RX I	
GPIO_17	39	17 I/O			UART0_TX O	
GPIO_18	40	18 I/O			UART1_TX O	
GPIO_19	41	19 I/O	PWM1_1 O			
GPIO_20	42	20 I/O	PWM1_0 O			
GPIO_21	43	21 I/O	PWM1_2 O			
GPIO_22	44	22 I/O	PWM1_3 O			
GPIO_23	45	23 I/O			UART1_RX I	

表9：SiFive端口引脚辅助功能。图片来源：SiFive FE310-G002数据手册。

GPIO引脚是多路复用的，这意味着最多可以有3个PWM、1个SPI、1个I2C和2个UART。请注意，没有模数转换器（Analog to Digital Converter，ADC）。由于它采用QFN48封装，难以手工焊接（或脱焊），因此最适合机器布局。

9.2.3 SparkFun Electronics RED-V SiFive Thing Plus实现

对于实验或小批量生产，也可以使用RED-V SIFIVE RISC-V THING PLUS。1568-DEV-15799-ND的成本为29.95美元。需要增加连接引脚。SparkFun网站上提供了原理图。

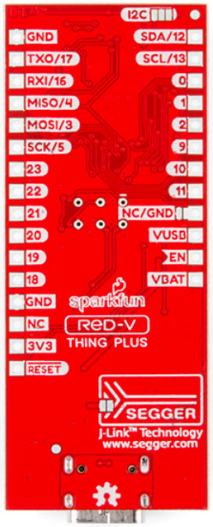


图112：SparkFun Electronics SiFive Thing Plus仰视图。图片来源：SparkFun Electronics

10 有关RISC-V的更多信息

有关RISC-V的更多信息，请参见YouTube和RISC-V.org网站上的RISC-V峰会议程。RISC-V的发明者David Patterson和Krste Asanovic都是幽默的演讲者。YouTube上还提供了Maix BiT、SparkFun和Digilent电路板的视频。

对于那些希望在本文介绍的FPGA中运行SweRV核心后巩固并获得更多知识的读者，请关注将于2020年11月发布的Imagination Technologies“RVfpga：了解计算机架构的完整课程”。它由20个实验组成，分为4个部分，本指南涵盖了第1部分的一些内容：

第1部分：Vivado项目和编程 – Vivado和Verilator、C编程、Whisper、RISC-V应用程序二进制接口（Application Binary Interface，ABI）、过程调用约定、合并C和汇编代码。

第2部分：I/O系统 – 驱动Digilent Nexys A7电路板上的7段显示屏、中断、定时器和串行总线（SPI、I2C和UART）。

后续课程将于2021年第3季度发布：

第3部分：RISC-V内核 – 了解内核结构、流水线、风险和实现新指令

第4部分：RISC-V存储器系统 – 了解高速缓存控制器、高速缓存命中和未命中、修改高速缓存，了解ICCM（指令紧密耦合存储器）和DCCM（数据紧密耦合存储器）。

有关RISC-V的两本相关参考书的详细信息，另请参见“参考资料”部分。

11 结论

对比RISC-V的不同电路板后，MaixPy BiT被证明最容易使用且充满最多乐趣。有意思的是，成本最低的电路板（带摄像头和LCD）能够以双核RISC-V处理器的形式提供最强大的计算能力，而价格却非常适中。它是学习机器视觉和面部识别的理想工具。由于它使用MicroPython编程，因此即使是没有编程背景的人也可轻松上手。

SparkFun Electronics RED-V Thing Plus和Red Board非常适合业余爱好者，或者希望学习如何使用C语言或汇编语言进行RISC-V编程以期自行开发产品的电子/软件工程师。SiFive的软件工具易于使用，其中有许多有用的示例，经过修改便可用于其他应用。SiFive的文档也很易于阅读。适合ARM或TI处理器的Eclipse工具的用户对此应该不会感到困难。

在本RISC-V指南中，只能简要介绍如何在Xilinx FPGA中实现软内核。本指南面向的是学习计算机架构的学生和从事大型项目的工程师。这是一项复杂的业务，建议不要在没有详细说明的情况下进行。即使以前从未使用过软内核，但只要认真遵循Imagination Technologies“RVfpga：入门指南”并使用RVfpga软件包，便可于一天内在Nexys A7电路板上成功运行RISC-V内核。在四款电路板中，这是迄今为止技术性最强的一种。对于想要发挥自己智力的读者，可以参加后续活动Imagination Technologies“RVfpga：了解计算机架构的完整课程”。

我们在RISC-V上投入了大量精力，唯有时间才能证明它是ARM的有力竞争对手。开源ISA且没有版税无疑为RISC-V带来了竞争优势，因为它能够提供一些非常强大的低成本处理器。

12 致谢

非常感谢马德里大学的Daniel chaver Martinez教授允许作者使用他的RISC-V教学资料，还要感谢教授的学生Daniel Gonzalez允许作者使用他的硕士论文。没有上述授权，将无法编写本指南。

最后，特别感谢Imagination Technologies的Robert Owen构思了本指南并最终实现。

13 作者简介

Richard Sikora担任硬件和软件工程师已逾30年，从事过30多种不同处理器和数字信号处理器（Digital Signal Processors，DSP）的相关工作。多年来，他设计了称量机、过程控制仪表、飞机点火装置、自动售货机、智能卡读卡器以及烟雾和一氧化碳监视器等设备。此外，他还为Texas Instruments DSP和Matlab编制过多个教学工具包。

14 参考资料

嵌入式微处理器基准联盟
https://www.eembc.org/coremark/

RISC-V徽标https://riscv.org/RISC-V-branding-guidelines-and-materials/

RISC-V规范https://riscv.org/technical/specifications/

其他内核https://chipsalliance.org

Kendryte K210数据手册https://s3.cn-north-1.amazonaws.com.cn/dl.kendryte.com/documents/kendryte_datasheet_20181011163248_en.pdf

Wishbone总线
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/

SweRV内核下载地址：
https://github.com/chipsalliance/Cores-SweRV

SweRV编程器参考手册
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，《适合工业物联网的专用RISC-V片上系统的FPGA实现》，硕士学位论文，马德里康普顿斯大学，2020年7月。

数字设计和计算机架构，Sarah Harris和David Money Harris，第二版（RISC-V版本将于2021年发布）
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&sprefix=digital+design+and+%2Caps%2C219&sr=8-1

RISC-V Reader，David Patterson和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