SiFive VCU118 FPGA Getting Started Guide

20G1.05.00

SiFive VCU118 FPGA Getting Started Guide

Proprietary Notice

Release Information

Version	Date	Changes
20G1.03.00	June 15, 2020	Initial release.
20G1.05.00	August 19, 2020	Updated generic Core IP block diagram.

Contents

Chapter 1

Introduction

1.1 About this Document

This document gives necessary information for a user of the SiFive Core IP VCU118 FPGA Eval Kit. To learn more about the functionality of your specific Core IP please read the appropriate Core IP Manual.

This guide will help you download and flash the SiFive Core IP VCU118 FPGA Eval Kit image to an FPGA development board. It will help you install software tools to allow you to write, upload, and debug code on the Eval Kit. It also contains information about what is contained in the MCS file for the SiFive Core IP VCU118 FPGA Eval Kit.

1.2 About this Release

This Eval Kit allows you to prototype and benchmark your target RISC-V software without modifying, integrating, or synthesizing any Verilog code.

This SiFive Core IP VCU118 FPGA Eval Kit includes .mcs and .bit files to program FPGA flash and RAM respectively with Core IP.

This release is intended for evaluation purposes only.

1.3 Evaluation Version Limitations

Version 20G1.03.00 of the SiFive Core IP VCU118 FPGA Eval Kit has the following limitations compared with the fully functional Core IP:

- DTIM is limited in size to 64kB.
- Peripheral Bus, System Bus, and Front Bus are not exported for additional user connections. The evaluation can utilize the peripherals included on the FPGA.
- A fixed number of Local or Global interrupts are exported at the top level.

- Designs without PLIC or CLIC have one added depending on the platform type.
- Designs with APB Debug Interface have JTAG debug interface.
- Refer to your FPGA `core.dts` file for other potential differences.

To target a different FPGA platform or perform synthesis or simulation, you may obtain an Evaluation Version of the Core IP RTL from https://www.sifive.com.

Chapter 2

Required Hardware

The SiFive Core IP VCU118 FPGA Eval Kit requires the following hardware:

2.1 Xilinx Virtex UltraScale+ FPGA VCU118 Evaluation Kit

The Xilinx VCU118 is a Xilinx FPGA development board for makers and hobbyists. The Xilinx VCU118 features the Xilinx XCVU9P-L2FLGA2104E. The board can be purchased directly from Xilinx.

https://www.xilinx.com/products/boards-and-kits/vcu118.html

2.2 Two USB A to Micro-B Cable

Two standard USB Type A Male to Micro-B Male cable can be used to interface with the VCU118. These cables are used to interface between the board and the host machine.

2.3 Olimex ARM-USB-TINY-H Debugger

The Olimex ARM-USB-TINY-H is a hardware JTAG debugger. The Core IP VCU118 FPGA Dev Kit has a standard JTAG debugging interface, and the tools included with the VCU118 have been tested using the Olimex ARM-USB-TINY-H. It can be purchased from Olimex or Digi-Key.

https://www.olimex.com/Products/ARM/JTAG/ARM-USB-TINY-H/

http://www.digikey.com/product-detail/en/olimex-ltd/ARM-USB-TINY-H/1188-1013-ND/3471388

2.4 USB A to B Cable

Any standard USB Type A Male to B Male cable can be used to interface to the Olimex ARM-USB-TINY-H Debugger. Note that the package does not include one. These are available from a variety of sources, including Digi-Key.

http://www.digikey.com/product-detail/en/assmann-wsw-components/AK672-2-1/AE1462-ND/930247

2.5 Male-To-Female Jumper Cables (10)

The connection between the Olimex ARM-USB-TINY-H and SiFive Core IP VCU118 FPGA Eval Kit requires 10 connections. These can be made with Male-to-Female jumper cables. These cables are available from Adafruit in convenient rip-apart ribbon cables:

https://www.adafruit.com/products/826

2.6 ATX Power Supply

The VCU118 is powered by an ATX power supply that is packaged with the kit.

Chapter 3

Board Setup

3.1 Connecting the Power Supply

Connect the 6-pin power cable leading from the 12V power supply to the power connector (J15). The power switch (SW1) will be used to start the board.

3.2 Connecting the USB Interface

Connect the two USB Type A to USB Micro-B cables between the USB-JTAG Connector on the VCU118 (J106/U115) and the host machine. This provides console access to the logic of the Core IP subsystem and the FPGA.

3.3 Connecting the Debugger

The debugger is essential for downloading and debugging code with your SDK. Without the debugger, you can only flash the FPGA image and run the included demo program, you cannot change the software which executes.

Note

The user must set the program counter to the memory location prior to executing.

The GDB command to do this is:

```
thread apply all set $pc=_enter”
```

Connect the Olimex ARM-USB-TINY-H with the USB Type A to B cable to the host machine. Then connect the Olimex ARM-USB-TINY-H debugger to GPIO PMOD header (J52) using the 10 jumper cables. The pinout is as shown in Table 1. Note that the Olimex ARM-USB-TINY-H and the PMOD header on the VCU118 Board have different numbering schemes. Table 2 and Table 3 clarify the different pinouts for the two connectors.

Table 1: *Debugging Connections between Olimex ARM-USB-TINY-H and VCU118 Board's PMOD header J52*

Signal Name	ARM-USB-TINY-H Pin Number	Suggested Jumper Color	VCU118 J52 Pin Number
VREF	1	red	12
VREF	2	brown	6 ("VCC")
nTRST	3	orange	2
TDI	5	yellow	7
TMS/TMSC	7	green	8
TCK/TCKC	9	blue	3
TDO	13	purple	1
GND	14	black	5 ("GND")
nRST	15	grey	9
GND	16	white	11

Table 2: *Debug Connections To the Olimex ARM-USB-TINY-H*

	1 : VREF (red)	2 : VREF (brown)
	3 : nTRST (orange)	4
	5 : TDI (yellow)	6
	7 : TMS/TMSC (green)	8
NOTCH	9 : TCK/TCKC (blue)	10
NOTCH	11	12
	13 : TDO (purple)	14 : GND (black)
	15 : nRST (grey)	16 : GND (white)
	17	18
	19	20
	LED	

Table 3: *Debug Connections to the VCU118 Board J52 PMOD Header*

square pad	1 : TDO (purple)	7 : TDI (yellow)
	2 : nTRST (orange)	8 : TMS/TMSC (green)
	3 : TCK/TCKC (blue)	9 : nRST (grey)
	4	10
“GND”	5 : GND (black)	11 : GND (white)
“VCC”	6 : VREF (brown)	12 : VREF (red)

Chapter 4

FPGA Programming Files

The Xilinx VCU118 configures on power-on from an on-board 16MB Quad-SPI Flash.

To program the VCU118, locate the desired MCS file or BIT file within your Core IP package. The BIT file typically takes around a minute to install, whereas the MCS file will take 20 minutes to install.

Note

Xilinx flashing program Vivado is only supported on Linux and Windows.

The file name is encoded with the Core IP type and version information.

`design-vcu118_primary.mcs and design-vcu118_secondary.mcs`

The user must program both the primary and secondary MCS files.

SiFive Freedom Studio and Xilinx Vivado Design Suite can be used for .mcs flash programming. Only the Xilinx Vivado Design Suite can be used to program a .bit file. A .bit file is raw storage of FPGA information and can be loaded directly onto an FPGA but it does not persist through a power cycle. A .mcs file contains FPGA information, FPGA configuration information, and is stored in non-volitale (flash) memory so it persists through power cycles.

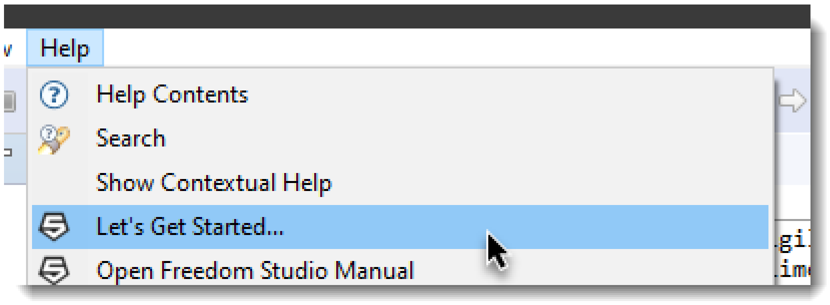
4.1 Using SiFive Freedom Studio

Install the latest Freedom Studio application for your Windows or Linux OS from: https://www.sifive.com/boards

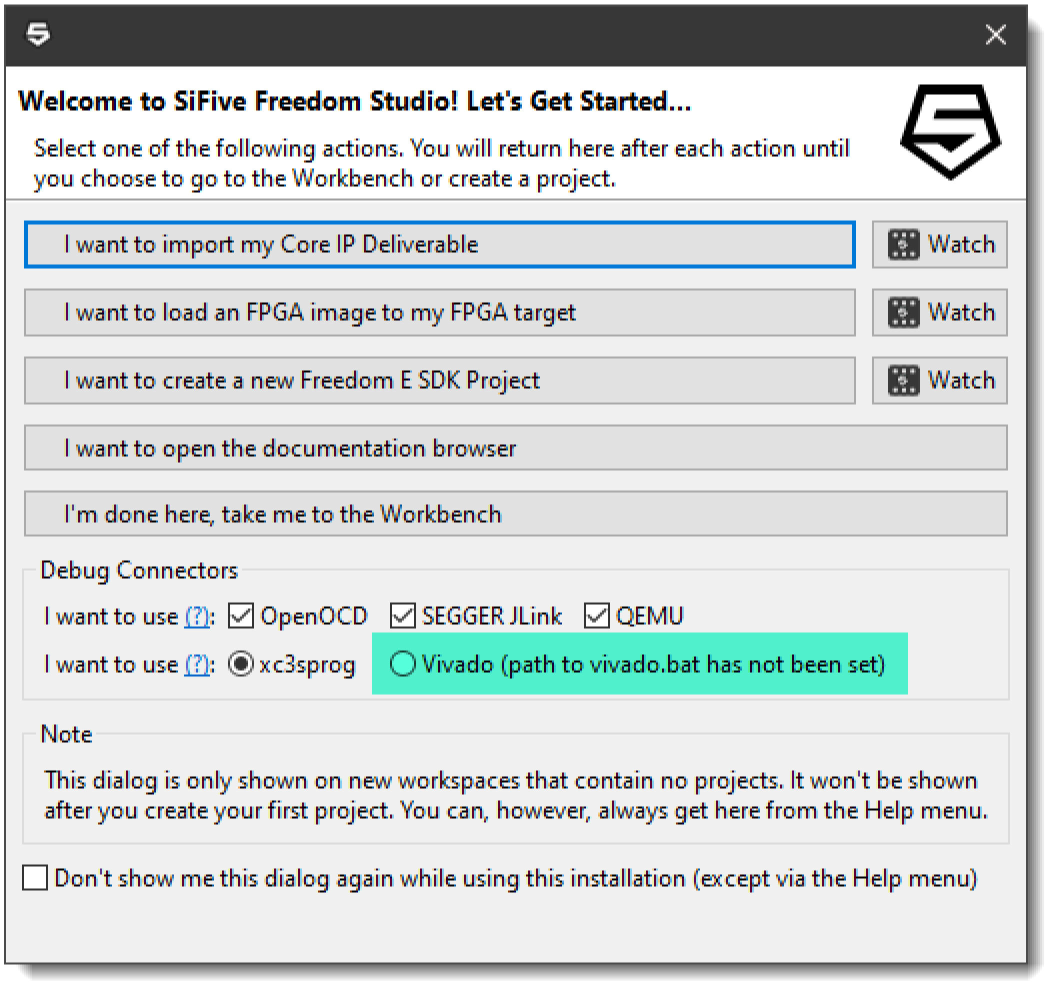
4.2 Windows Programming Example

The following section describes how to program the VCU118 with the Core IP evaluation package using Freedom Studio.

To begin, start Freedom Studio, creating a new workspace if you do not already have one. Freedom Studio will display a Getting Started dialog box if your workspace is empty. If you don't see this dialog, open it from the Help menu:



On the Getting Started Dialogue:



Welcome to SiFive Freedom Studio! Let's Get Started...

Select one of the following actions. You will return here after each action until you choose to go to the Workbench or create a project.

I want to import my Core IP Deliverable
Watch
I want to load an FPGA image to my FPGA target
Watch
I want to create a new Freedom E SDK Project
Watch
I want to open the documentation browser
I'm done here, take me to the Workbench

Debug Connectors

I want to use (?): ☑ OpenOCD ☑ SEGGER JLink ☑ QEMU

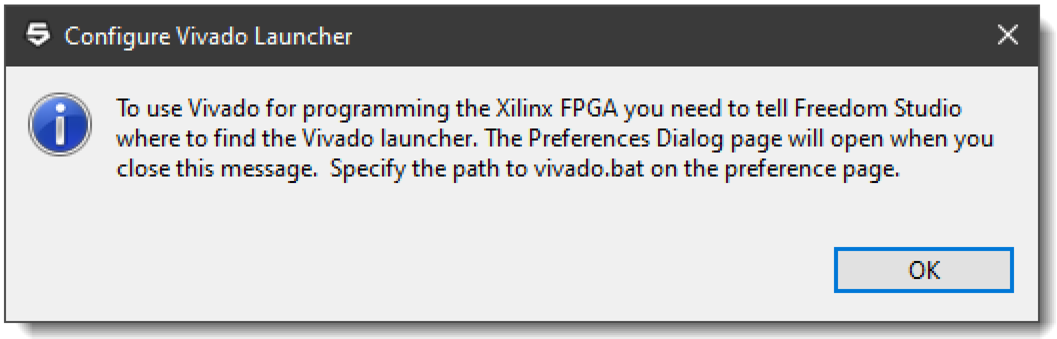
I want to use (?): [✓ xc3sprog] Vivado (path to vivado.bat has not been set)

Note

This dialog is only shown on new workspaces that contain no projects. It won't be shown after you create your first project. You can, however, always get here from the Help menu.

☐ Don't show me this dialog again while using this installation (except via the Help menu)

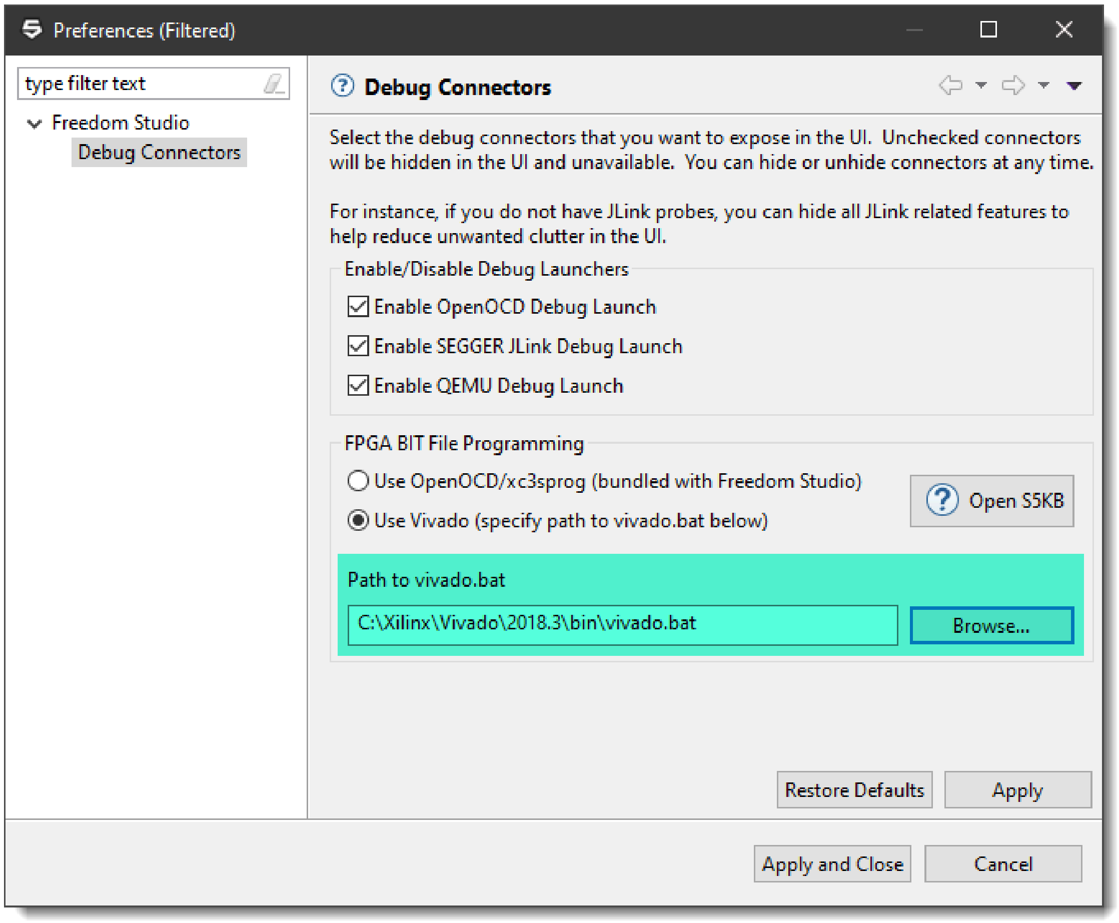
Click the "Vivado" radio button. When you do, a new message box will appear telling you that Freedom Studio needs to know where to find the Vivado launcher.



Configure Vivado Launcher

To use Vivado for programming the Xilinx FPGA you need to tell Freedom Studio where to find the Vivado launcher. The Preferences Dialog page will open when you close this message. Specify the path to vivado.bat on the preference page.

Press the OK button to dismiss this message box. The Preference Dialog will then open and you can specify the path to vivado.bat (on Windows) or vivado (on Linux). Use the "Browse" button to locate the appropriate launcher.


Preferences (Filtered)

type filter text

Freedom Studio

Debug Connectors

Debug Connectors

Select the debug connectors that you want to expose in the UI. Unchecked connectors will be hidden in the UI and unavailable. You can hide or unhide connectors at any time.

For instance, if you do not have JLink probes, you can hide all JLink related features to help reduce unwanted clutter in the UI.

Enable/Disable Debug Launchers

- [x] Enable OpenOCD Debug Launch
- [x] Enable SEGGER JLink Debug Launch
- [x] Enable QEMU Debug Launch

FPGA BIT File Programming

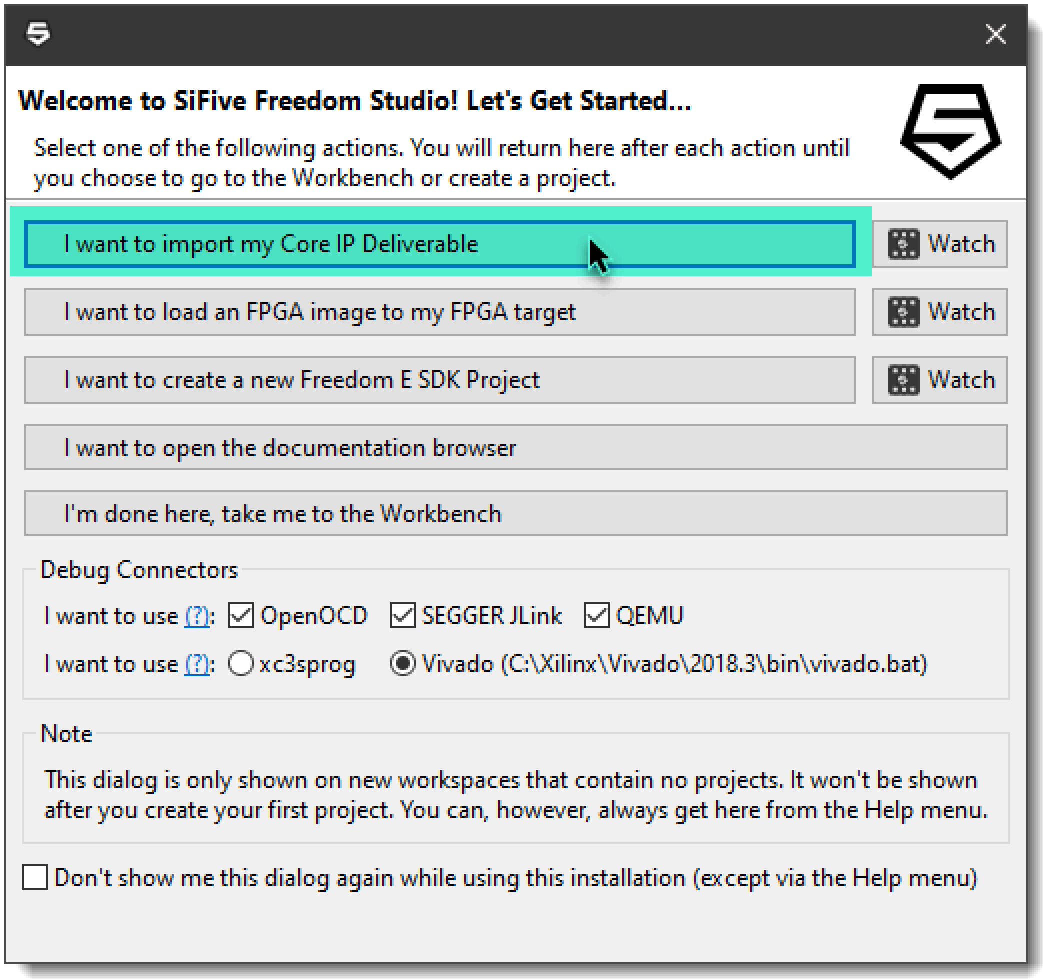
○ Use OpenOCD/xc3sprog (bundled with Freedom Studio)

● Use Vivado (specify path to vivado.bat below)

Path to vivado.bat: C:\Xilinx\Vivado\2018.3\bin\vivado.bat

Click the “Apply and Close” button and go back to the Getting Started Dialog. This configuration only needs to be done once.

On the Getting Started Dialog, press the "I want to import my Core IP Deliverable" button.



Welcome to SiFive Freedom Studio! Let's Get Started...

Select one of the following actions. You will return here after each action until you choose to go to the Workbench or create a project.

I want to import my Core IP Deliverable
Watch
I want to load an FPGA image to my FPGA target
Watch
I want to create a new Freedom E SDK Project
Watch
I want to open the documentation browser
I'm done here, take me to the Workbench

Debug Connectors

I want to use (?): [✓] OpenOCD [✓] SEGGER JLink [✓] QEMU

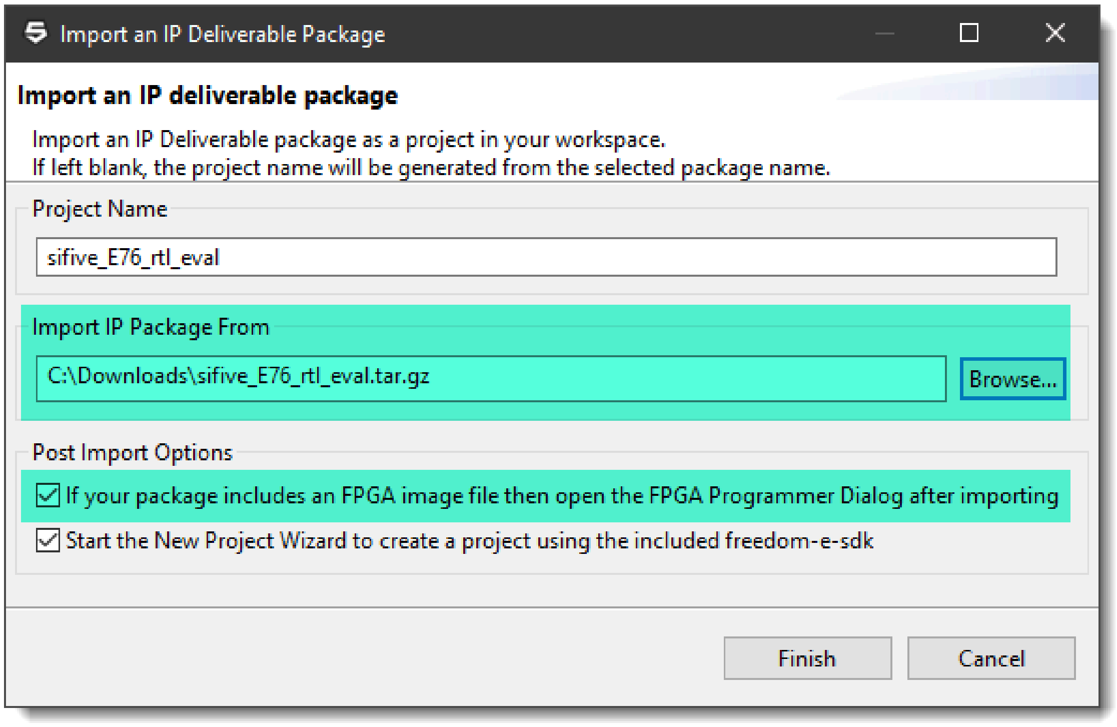
I want to use (?): ○ xc3sprog ● Vivado (C:\Xilinx\Vivado\2018.3\bin\vivado.bat)

Note

This dialog is only shown on new workspaces that contain no projects. It won't be shown after you create your first project. You can, however, always get here from the Help menu.

☐ Don't show me this dialog again while using this installation (except via the Help menu)

This opens the Import Wizard:


Import an IP Deliverable Package

Import an IP deliverable package

Import an IP Deliverable package as a project in your workspace.
If left blank, the project name will be generated from the selected package name.

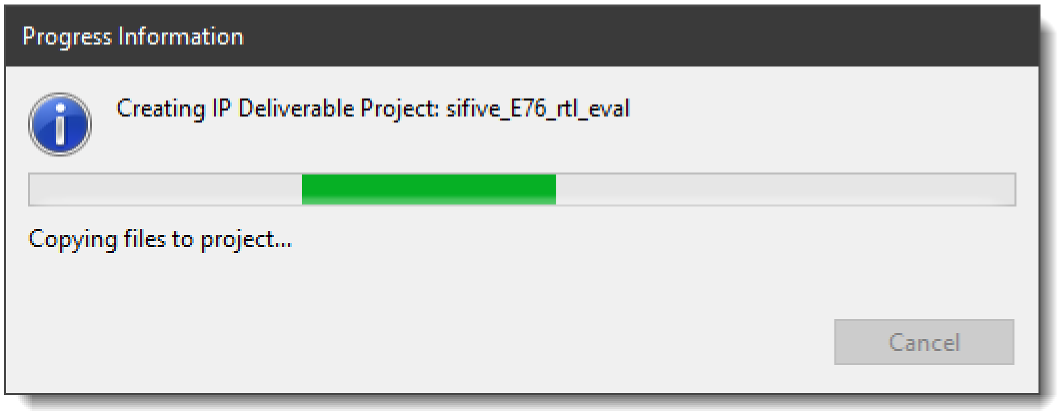
Project Name: sifive_E76_rtl_eval

Import IP Package From: C:\Downloads\sifive_E76_rtl_eval.tar.gz

Post Import Options

- ☑ If your package includes an FPGA image file then open the FPGA Programmer Dialog after importing
- ☑ Start the New Project Wizard to create a project using the included freedom-e-sdk

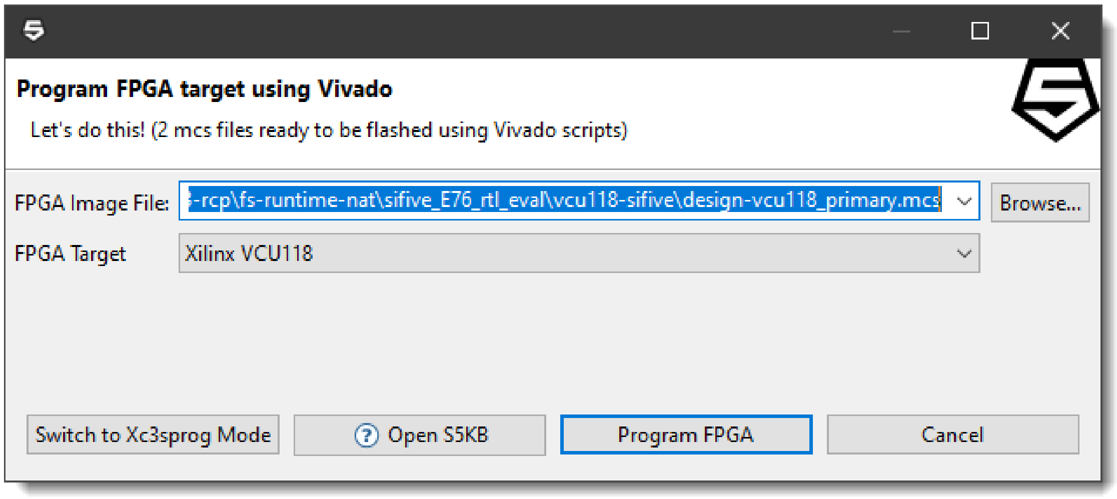
Use the Browse button to locate your IP package. The project name will be generated from your IP package, but you can also specify a different name if desired. Leave the highlighted check-box checked. You can also leave the New Project Wizard checkbox checked if you want to cre-ate a software example project after flashing the core to the target. Press the Finish button to start the import. You’ll see a progress dialog as the IP project is imported.


Progress Information

Creating IP Deliverable Project: sifive_E76_rtl_eval

Copying files to project...

After the IP package is imported, the Programming Dialog will be displayed. The programmer will default to the Arty 100T FPGA image if present in the package. If this is true for your pack-age, the use the FPGA Image File drop down to select the VCU118_primary.mcs file. The FPGA Target dropdown should automatically switch to the Xilinx VCU118. If it does not, do so manually.



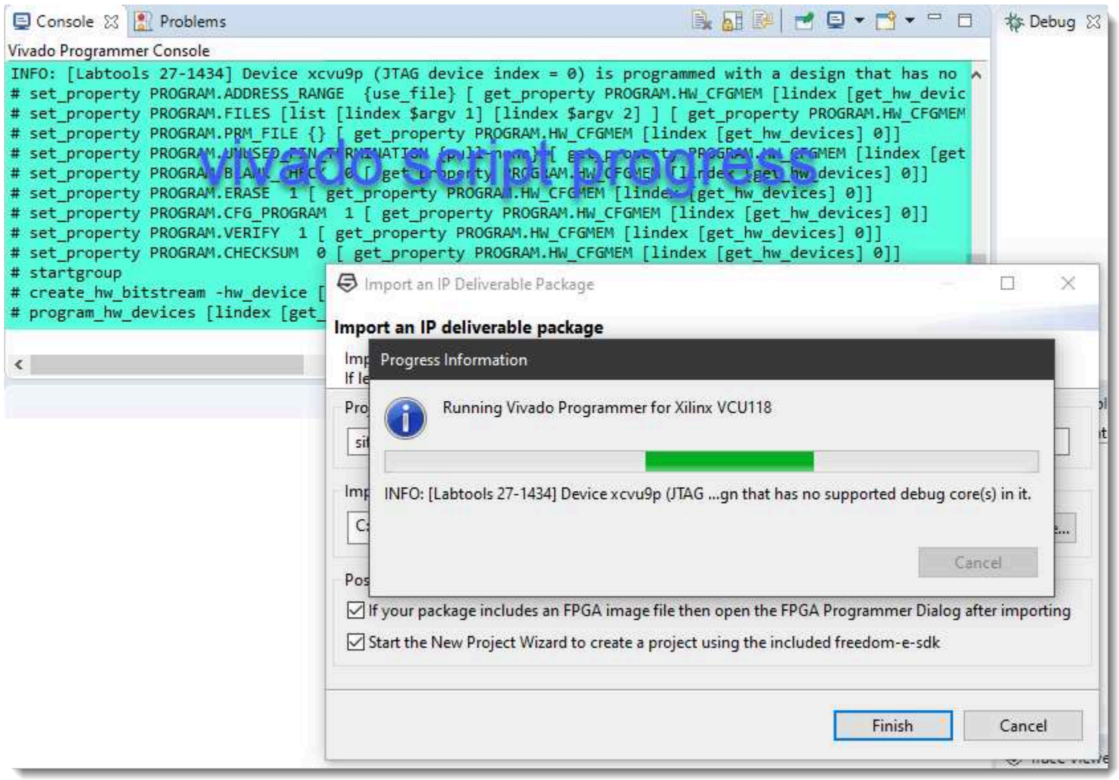
Program FPGA target using Vivado

Let's do this! (2 mcs files ready to be flashed using Vivado scripts)

FPGA Image File: -rcp\fs-runtime-nat\sifive_E76_rtl_eval\vcu118-sifive\design-vcu118_primary.mcs

FPGA Target: Xilinx VCU118

When you are ready for a cup of coffee (or two) press the "Program FPGA" button. Programming the MCS images can take up to 20 minutes to complete. A progress dialog will be shown and you will also see output on the Freedom Studio console:



Vivado Programmer Console

```
INFO: [Labtools 27-1434] Device xcvu9p (JTAG device index = 0) is programmed with a design that has no
# set_property PROGRAM.ADDRESS_RANGE  {use_file} [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devic
# set_property PROGRAM.FILES [list [lindex $argv 1] [lindex $argv 2] ] [ get_property PROGRAM.HW_CFGMEM
# set_property PROGRAM.PRM_FILE {} [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]
# set_property PROGRAM.UNUSED_PIN_TERMINATION {pull-none} [ get_property PROGRAM.HW_CFGMEM [lindex [get
# set_property PROGRAM.BLANK_CHECK  0 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]
# set_property PROGRAM.ERASE  1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]
# set_property PROGRAM.CFG_PROGRAM  1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]
# set_property PROGRAM.VERIFY  1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]
# set_property PROGRAM.CHECKSUM  0 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]
# startgroup
# create_hw_bitstream -hw_device [
# program_hw_devices [lindex [get_
```

Import an IP Deliverable Package

Import an IP deliverable package

Progress Information

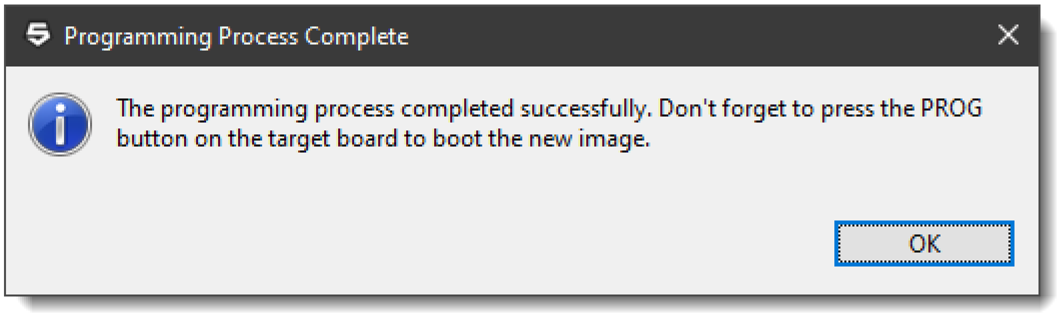
Running Vivado Programmer for Xilinx VCU118

INFO: [Labtools 27-1434] Device xcvu9p (JTAG ...gn that has no supported debug core(s) in it.

☑ If your package includes an FPGA image file then open the FPGA Programmer Dialog after importing

☑ Start the New Project Wizard to create a project using the included freedom-e-sdk

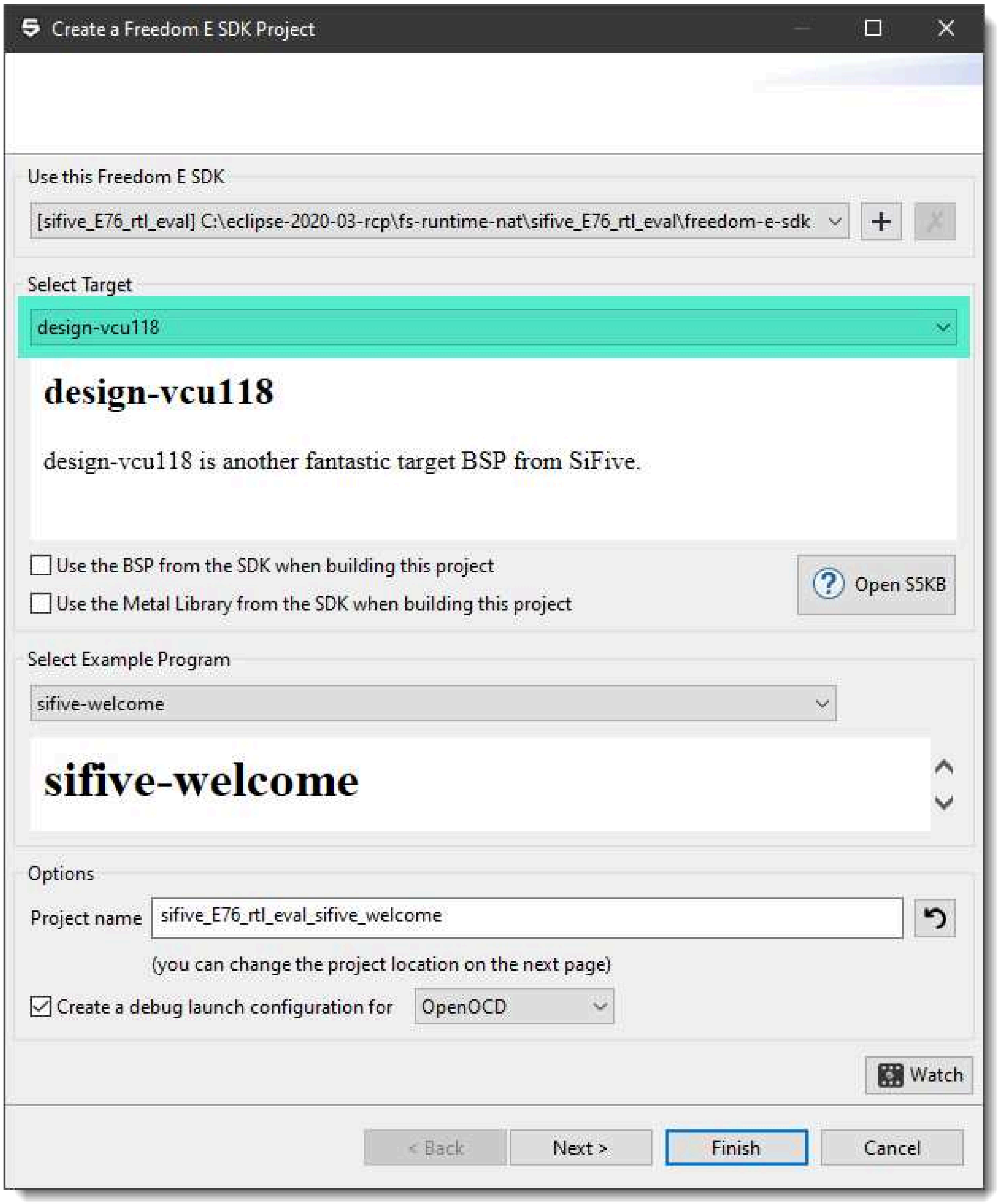
When programming is complete you will be prompted to press the PROG button on the target to load the new core into the FPGA from flash memory.



The programming process completed successfully. Don't forget to press the PROG button on the target board to boot the new image.

Your VCU118 target board is now programmed with the core from your IP package.

If the "New Project Wizard" checkbox was checked, you’ll now see the New Project Wizard dialog. Make sure to select "design-vcu118" from the Select Target dropdown.



Create a Freedom E SDK Project

Use this Freedom E SDK

[sifive_E76_rtl_eval] C:\eclipse-2020-03-rcp\fs-runtime-nat\sifive_E76_rtl_eval\freedom-e-sdk

Select Target

design-vcu118

design-vcu118

design-vcu118 is another fantastic target BSP from SiFive.

☐ Use the BSP from the SDK when building this project

☐ Use the Metal Library from the SDK when building this project

Select Example Program

sifive-welcome

sifive-welcome

Options

Project name: sifive_E76_rtl_eval_sifive_welcome

(you can change the project location on the next page)

☑ Create a debug launch configuration for: OpenOCD

The default settings will create example project and debug launch configuration for the "sifive_welcome" example. You can choose a different example if you wish. Pressing Finish will create a new example software project and a debug launch configuration. The debug launch configuration is fully configured, requiring no changes. If you have connected the Olimex probe to the target, you can press the "Debug" button and start the debugger. See the Freedom Studio User Guide for more information.

4.3 Using Xilinx Vivado Design Suite

The Xilinx Vivado Design Suite is used flash programming.

Both the Vivado Lab Edition and WebPACK Edition 2018.2 support Artix-7 devices and VCU118 free of charge.

4.3.1 Programming the VCU118 Flash

To program the VCU118 with Vivado take the following steps:

1. Launch Vivado
2. Open Hardware Manager
3. Open target board
4. Right click on the FPGA device and select "Add Configuration Memory Device"
5. Select the following SPI flash parameters:

Name	mt25qu01g-spi-x1_x2_x4_x8
Part	mt25qu01g
Manufacturer	Micron
Alias	-
Family	mt25qu
Type	spi
Density	1024
Width	x1_x2_x4_x8

6. Click OK to "Do you want to program the configuration memory device now?"
7. Add the MCS file
8. Select OK
9. Once the programming completes in Vivado, press the "PROG" Button on the VCU118 Board to load the image into the FPGA.

Chapter 5

Boot and Run

5.1 Serial Setup

Using a terminal emulator such as GNU screen on Linux or a terminal on Windows, open a console connection from the host computer to the VCU118.

Set the following parameters:

Speed	115200
Parity	None
Data bits	8
Stop bits	1
Hardware Flow	None

For example, on Linux using GNU Screen:

```
sudo screen /dev/ttyUSB2 115200
```

You can use `Ctrl-a k` to “kill” (exit) the running screen session.

Depending on your setup, you may need additional drivers or permissions to communicate over the USB port.

If you are running on Ubuntu-style Linux, the below is an example of steps you may need to follow to access your dev kit without `sudo` permissions:

1. With your board’s debug interface connected, make sure your device shows up with the `lsusb` command:

   ```
   > lsusb
   ```

```
...
Bus XXX Device XXX: ID 0403:6011 Future Technology Devices International,
Ltd FT2232C Dual USB-UART/FIFO IC
```

2. Set the udev rules to allow the device to be accessed by the `plugdev` group:

   ```
   > sudo vi /etc/udev/rules.d/99-openocd.rules
   ```

   Add the following lines and save the file (if they are not already there):

   ```
   # These are for the HiFive1 Board
   SUBSYSTEM=="usb", ATTR{idVendor}=="0403",
     ATTR{idProduct}=="6011", MODE="664", GROUP="plugdev"

   SUBSYSTEM=="tty", ATTRS{idVendor}=="0403",
     ATTRS{idProduct}=="6010", MODE="664", GROUP="plugdev"

   # These are for the Olimex Debugger for use with CoreIP VCU118 Devkit

   SUBSYSTEM=="usb", ATTR{idVendor}=="15ba",
     ATTR{idProduct}=="002a", MODE="664", GROUP="plugdev"

   SUBSYSTEM=="tty", ATTRS{idVendor}=="15ba",
     ATTRS{idProduct}=="002a", MODE="664", GROUP="plugdev"
   ```

3. See if your board shows up as a serial device belonging to the `plugdev` group:

   ```
   > ls /dev/ttyUSB*
   /dev/ttyUSB0  /dev/ttyUSB1  /dev/ttyUSB2  /dev/ttyUSB3
   ```

   (If you have other serial devices or multiple boards attached, you may have more devices listed). For serial communication with the UART, you will always want to select the higher number of the pair, in this example `/dev/ttyUSB2`.

   ```
   > ls -l /dev/ttyUSB2
   crw-rw-r-- 1 root plugdev 188, 1 Nov 28 00:00 /dev/ttyUSB1
   ```

4. Add yourself to the `plugdev` group to eliminate the need to sudo for access to the device. You can use the `whoami` command to determine your user name.

   ```
   > whoami
     your_user_name
   > sudo usermod -a -G plugdev your_user_name
   ```

5. Log out and log back in, then check that you’re now a member of the plugdev group:

   ```
   > groups
   ... plugdev ...
   ```

Now you should be able to access the serial (UART) and debug interface without sudo permissions.

5.1.1 Reset and Boot

By default, the core will reset to address `0x0`. The user will need to connect to the debugger, as outlined in Section 3.3.

5.1.2 Load a Program

You can change the program which the VCU118 runs by using the debug/programming interface to flash a new compiled program into the DTIM.

Chapter 6

Software Development Flow

SiFive supports several methods of obtaining the software development toolchain. Freedom Studio is an Eclipse-Based IDE which bundles everything you need into one download. You can also compile the source yourself and run command line tools with the Freedom E SDK.

These different development versions will all install the same set of tools, but the versions, install paths and associated software libraries and examples are different for each.

6.1 Supported Platforms

Freedom Studio is supported on Linux and Windows.

6.2 Software Development Using Freedom Studio IDE

SiFive recommends software development for the VCU118 with the Eclipse-based Freedom Studio IDE. Freedom Studio is supported for Windows, macOS, and Linux. When using this method, the precompiled tools and drivers are automatically installed, you do not need to download or install it seperately to get tools and example code.

You can obtain Freedom Studio from the SiFive website:

https://www.sifive.com/boards

More information on how to use it can be found in the Freedom Studio Manual:

https://www.sifive.com/documentation/tools/freedom-studio-manual

6.3 Software Development Using Freedom E SDK Command Line Tools

Freedom-E-SDK is a public github repository, maintained by SiFive Inc, that makes it easy to get started developing software for SiFive's Freedom and RISC-V Core IP Platforms. The SDK supports a wide array of SiFive Core IPs, SoCs and Emulation environments.

https://github.com/sifive/freedom-e-sdk

This section describes how to set up the toolchain and configure the SDK. The section also walks through building an example program and executing it in the RTL testbench included in a SiFive Core IP deliverable.

Note

SiFive Core IP deliverables downloaded from the SiFive site already include a ready-to-use version of Freedom E SDK. Installation from github is not required.

In addition, the section will walk through how to import custom BSPs and build a program using the custom BSP target.

6.3.1 Setting Up Freedom-E-SDK

Prerequisites

To use this SDK, you will need the following software available on your machine:

- Internet Connection
- GNU Make
- Git

Toolchain Prerequisites

To build examples and programs, you will need the following software installed on your machine:

- RISC-V GNU Toolchain
- RISC-V OpenOCD (for use with development board and FPGA targets)

Pre-built versions of these softwares can be found on the SiFive Website.

https://www.sifive.com/boards

The pre-built tools have been carefully packaged to support both RISC-V 32bit & 64bit ISAs and work on Linux, macOS, and Windows hosts.

Download the toolchain your platform, and unpack it to your desired location. Then, use the RISCV_PATH and RISCV_OPENOCD_PATH variables when using the tools.

For example,

```
    > cp openocd-<date>-<platform>.tar.gz /my/desired/location/
    > cp riscv64-unknown-elf-gcc-<date>-<platform>.tar.gz /my/desired/location
    > cd /my/desired/location
    > tar -xvf openocd-<date>-<platform>.tar.gz
    > tar -xvf riscv64-unknown-elf-gcc-<date>-<platform>.tar.gz
    > export RISCV_OPENOCD_PATH=/my/desired/location/openocd
    > export RISCV_PATH=/my/desired/location/riscv64-unknown-elf-gcc-<date>-<version>
```

If you are configuring a toolchain included with a SiFive Core IP deliverables package, set the following environment variables:

```
> export RISCV_OPENOCD_PATH=/my/desired/location/openocd
> export RISCV_PATH=/my/desired/location/riscv64-unknown-elf-gcc-<date>-<version>
```

6.3.2 Cloning the Repository

The Freedom-E-SDK repository can be cloned by running the following commands:

```
    > git clone --recursive https://github.com/sifive/freedom-e-sdk.git
    > cd freedom-e-sdk
```

The recursive option is required to clone the git submodules included in the repository. If at first you omit the recursive option, you can achieve the same effect by updating submodules using the command:

```
    > git submodule update --init --recursive
```

6.4 Freedom E SDK VCU118 BSP

The Freedom Metal Compatibility Library layer uses the board support package files to provide the hardware abstraction layer. These BSP files can be found under the bsp folder in Freedom-E-SDK and are encapsulated entirely within each target directory. The supported targets are:

- SiFive Freedom E310 Arty
- freedom-e310-arty

SiFive CoreIP FPGA targets

- design-arty
- design-vcu118

For example, the board support files may consist of the following:

- bsp/design-xx
  - core.dts: The DeviceTree description of the target. This file is used to parameterize the Freedom Metal library to the target device. It is included as reference so that users of Freedom Metal are aware of what features and peripherals are available on the target.
  - design.dts: The DeviceTree Overlay for describing the chosen node, and testrams.
  - design.svd: CMSIS SVD XML file used for describing the design peripherals, their registers register-fields in the device with offset information on where they are all mapped into the memory space.
  - metal.default.lds
  - metal.freertos.lds
  - metal.scratchpad.lds
  - metal.ramrodata.lds
  - metal.h
  - metal-inline.h
  - metal-platform.h
  - settings.mk: Used to set `-march` and `-mabi` arguments to the RISC-V GNU Toolchain as well as configure target-specific build steps.
  - openocd.cfg: Used to configure OpenOCD for flashing and debugging the target device.

6.5 Example Programs

Some example programs can be found under software:

`***hello***`

Prints "Hello, World!" to stdout, if a serial device is present on the target.

`***return-pass***`

Returns status code 0 indicating program success.

`***return-fail***`

Returns status code 1 indicating program failure.

`***example-itim****`

Demonstrates how to statically link application code into the Instruction Tightly Integrated Memory (ITIM) if an ITIM is present on the target.

`***software-interrupt***`

Demonstrates how to register a handler for and trigger a software interrupt

`***timer-interrupt***`

Demonstrates how to register a handler for and trigger a timer interrupt

`***local-interrupt***`

Demonstrates how to register a handler for and trigger a local interrupt

`***example-pmp***`

Demonstrates how to configure a Physical Memory Protection (PMP) region

6.6 Using the Freedom E SDK

The example programs within freedom-e-sdk rely on built binaries from the freedom-devicetree-tools repository, which is included in the IP delivery .tar. These binaries are required to be built and added to your PATH before any freedom-e-sdk software examples can be compiled. To build the freedom-devicetree-tools binaries, follow these steps:

```
cd /path/to/tarball/freedom-e-sdk/freedom-devicetree-tools
```

```
./configure
```

```
make
```

Then add /path/to/tarball/freedom-e-sdk/freedom-devicetree-tools to your PATH.

Alternatively, the `freedom-devicetree-tools` repo can be copied into a location outside of the IP delivery .tar file and PATH updated with that location, so it is decoupled from the specific design you are working with.

6.6.1 Building an Example

To compile a bare-metal RISC-V program:

```
make PROGRAM=timer-interrupt TARGET=design-vcu118 software
```

The square brackets in the above command indicate optional parameters for the Make invocation.

To compile a bare-metal RISC-V program for a target included with your tarball:

```
make PROGRAM=timer-interrupt TARGET=design-vcu118 software
```

6.6.2 Uploading to the Target Board

```
make [PROGRAM=hello] [TARGET=design-vcu118] upload
```

6.6.3 Debugging a Target Program

```
make [PROGRAM=hello] [TARGET=design-vcu118] debug
```

6.6.4 Cleaning a Target Program Build Directory

```
make [PROGRAM=hello] [TARGET=design-vcu118] clean
```

6.6.5 Create a Standalone Project

You can export a program to a standalone project directory using the standalone target. The resulting project will be locked to a specific TARGET.

STANDALONE_DEST is a required argument to provide the desired project location.

```
make standalone [PROGRAM=hello] [TARGET=design-vcu118]
  STANDALONE_DEST=/path/to/desired/location
```

Once the standalone project is created, it can be compiled simply by typing make.

```
cd /path/to/desired/location standalone
make
```

Run make help for more commands.

Chapter 7

Core IP VCU118 MCS Image Contents

Figure 1 shows a block diagram of the Core IP VCU118.

The evaluation kit includes an evaluation Core IP along with additional peripherals and I/Os to allow software prototyping.

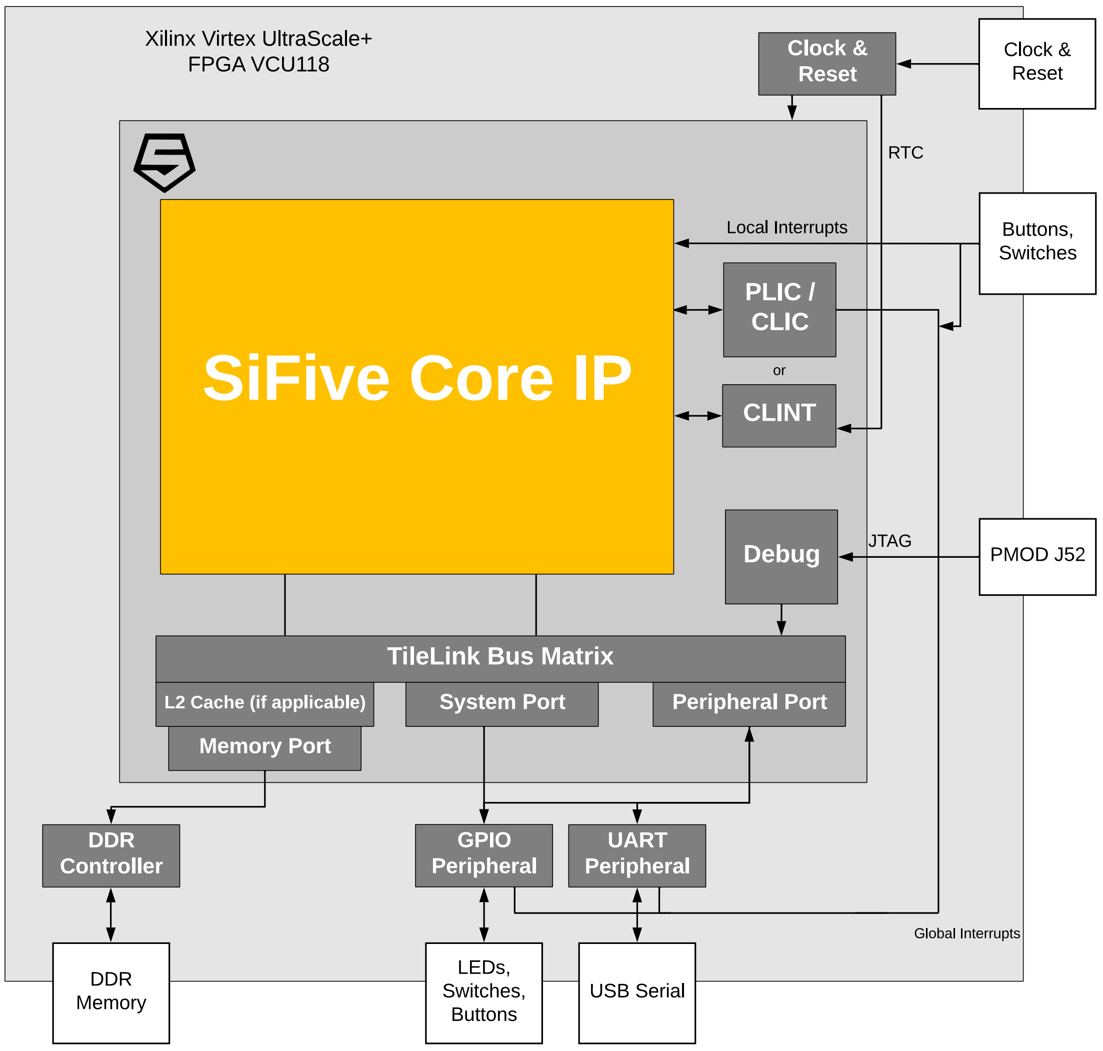


Figure 1: *Core IP VCU118 Block Diagram*

7.1 VCU118 Memory Map

The FPGA design on the VCU118 has an evaluation version of the SiFive Core IP Core IP, as well as peripheral devices which are not included with the Core IP deliverable. These devices allow you to perform basic I/O to prototype and benchmark some basic applications.

Please refer to the Device Tree file, (`core.dts`) for details of the Memory Map and interrupts.

7.2 VCU118 Clock and Reset

The VCU118 has a 300MHZ input to the FPGA. This is used to derive the Core IP's `io_coreClock` at 32.5 MHz, and the `clock` (peripheral clock) at 32.5 MHz. The `io_rtcToggle` is driven at approximately 32 kHz.

The system reset driven by the Reset Button on the evaluation board is combined with the external debugger's `SRST_n` pin as a full system reset for the VCU118. This is combined with the `io_ndreset` to drive the reset input to the Core IP.

7.3 VCU118 Pinout

The peripherals perform I/O functionalities and are also used to demonstrate the use of Global Interrupts. The peripheral devices are connected to hardware on the VCU118 development board as described in Table 4.

In addition, some board I/Os are configured as Local Interrupt sources. The mapping between hardware on the VCU118 and Local Interrupt sources are in the `core.dts` file in the deliverable.

Table 4: *VCU118 GPIO Offset to Board Pin Number*

Peripheral	Peripheral Offset	Connections
UART	UART TX/RX	To USB Serial
	SWITCH 0	Direct Global Interrupts
	SWITCH 1	
	SWITCH 2	
	SWITCH 3	
GPIO	GPIO[0] (Output Only)	LED 0
	GPIO[1] (Output Only)	LED 1
	GPIO[2] (Output Only)	LED 2
	GPIO[3] (Input Only)	SWITCH 3
	GPIO[4] (Input Only)	BUTTON 0
	GPIO[5] (Input Only)	BUTTON 1
	GPIO[6] (Input Only)	BUTTON 2
	GPIO[7] (Input Only)	BUTTON 3

The user may locate the local interrupt mapping by looking at the `core.dts` file in the deliverable.

Chapter 8

For More Information

Additional information, the latest version of this guide, and supporting files can be found at https://www.sifive.com.

More information about RISC-V in general is available at

http://riscv.org

SiFive thoughts, ideas, and news at

https://www.sifive.com/blog/

Webinars at

https://info.sifive.com/risc-v-webinar