embeddedworld2019
Exhibition&Conference
...it's a smarter world

How to Build a RISC-V Embedded System In Just 30 Minutes

Drew Barbier
SiFive, Inc.
San Mateo, CA, USA
drew@sifive.com

Cesare Garlati
prpl Foundation
Santa Clara, CA, USA
cesare@prplfoundation.org

***Abstract*—RISC-V is an open ISA (instruction set architecture) enabling a new era of innovation for processor architectures. RISC-V includes open source processor cores, toolchains, simulators and other key supporting components. The RISC-V ecosystem enables a new level of innovation in processor architecture that will be a key driver for the needed gains in performance and power efficiency over the next decade.**

Keywords—RISC-V; Open Source; ISA; FPGA; IDE

The rapid growth of RISC-V has been truly impressive to witness. Originally developed at UC Berkeley as a means to support a graduate student project, the free and open ISA has become widely popular promising to bring the innovation and collaboration of the open source community to the hardware world - and to dramatically disrupt the whole semiconductor industry in the process.

So really the question is: How do I get started with RISC-V?

This paper summarizes the hands-on workshop presented at Embedded World Conference 2019 and shows how to configure a custom RISC-V processor, program it into an FPGA, and write and debug software targeting the custom core.

Many tools are available to develop and customize RISC-V cores. Among them Chisel, the open-source hardware construction language developed at UC Berkeley that supports advanced hardware design using highly parameterized generators and layered domain-specific hardware languages [1].

For this class we decided to use free and open source tools including Sifive's Core Designer [2] to modify the core and Eclipse and OpenOCD to develop and debug the software.

Specific step-by-step flow presented in this class include:

- Configure a fully customized RISC-V Core.
- Program the customized core into an FPGA target.
- Setup an Eclipse software development environment.
- Develop, compile, debug and test a first RISC-V "Hello World" application

I. RISC-V PRIMER

The RISC-V ISA [1] aims to enable "extensibility without fragmentation" in support of the next phase of innovation in computing. The RISC-V base ISA includes less than 50 instructions. These are frozen and will be universally supported by RISC-V processors in perpetuity. These base instructions define a state-of-the-art RISC ISA offering improvements in performance and efficiency over legacy architectures.

The RISC-V ISA also defines additional instruction "extensions" which can optionally be included in a RISC-V implementation. An Example of an ISA extension is the Multiplication extension which provides multiplication and division instructions. In addition to optional standard extensions defined in ISA specification, opcode space is also reserved for custom instructions. This opcode space can be used to build instructions into the core to uniquely fit a given application – be that AI/ML acceleration, Cryptography, or anything else you can imagine. This modular approach to the ISA allows for custom RISC-V implementations which are still compatible with the extensive RISC-V software and tools ecosystem.

In this class, we will configure and download a custom RISC-V core, which includes an FPGA bitstream targeting a Digilent Arty A7-35T FPGA board. The target includes a complete peripheral subsystem which can then be used for software development.

II. CLASS SETUP

A. Core Designer

Core Designer is a graphical tool that allows customization of a RISC-V Core and SoC peripherals from a web interface. Log into Core Designer via SiFive's website, pick a Core Series, and customize it. Once happy with the core configuration, click "Build". This starts a build process in the cloud to configure, build, and verify the requested custom RISC-V core configuration.

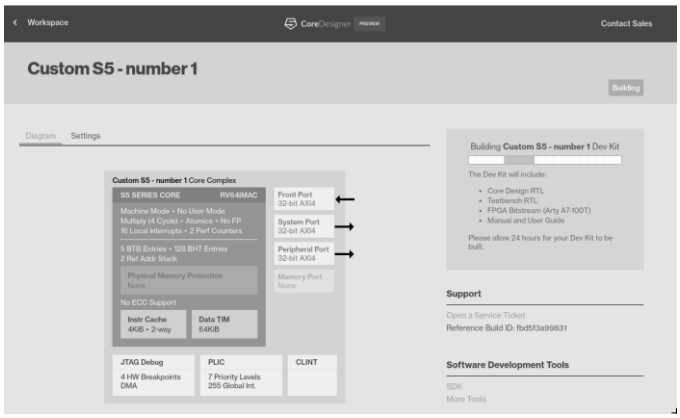

Fig. 1 Configuring a softcore via Core Designer

B. Program the FPGA

After the build completes, a download which contains RTL, an RTL testbench, FPGA bitstream, and Documentation is made available through the web interface. The RTL deliverable can be used for synthesis and simulation-based evaluations. The FPGA bitstream can be used for software development.

In this class we will program the FPGA bitstream to the Arty board using Xilinx's Vivado toolchain [4] and a micro USB connector.

C. Download and Install Freedom Studio (Eclipse)

Freedom Studio is the Sifive's repackaging of the open source Eclipse IDE with built in support for the RISC-V GNU toolchain. We'll use Freedom Studio for the whole software development cycle to write, compile, and debug a target firmware image. Freedom Studio supports all major Operating Systems, includes a pre-built GCC toolchain, OpenOCD debugger, and examples for 32-bit and 64-bit cores. In this example we download and install Freedom Studio binaries as described in the Getting Started Guide. [3]

D. Open Example Project

Open Freedom Studio and import the "coreplexip_welcome" application by navigating to File -> Import -> General -> Dev Kit Examples into Workspace; click the Browse button and then the E31.zip file.

Import all items.

E. Open and Build Example Project

Navigate and expand to the coreplexip_welcome item in the Project Explorer, expand coreplexip_welcome.c and double click on main(void). Here you can see this program completes the following tasks:

1. Set up the UART to 115,200 8N1
2. Print a Welcome Message
3. Operate a loop which causes LED LD1 to rotate through the color pallet.

Select the coreplexip_welcome project again and click the Build icon at the top. This compiles and links the program for this target board to produce an ELF output file.

F. Upload and Run Example Project over OpenOCD

Click the Run Icon, this will cause the built program to get uploaded to the board via open OCD. Click on the Terminal tab next to the Outline box in the bottom right, click the terminal picture to Open a Terminal, select Serial, 115,200 8N1 and you will see the SiFive welcome message in the terminal.

G. Modify and Debug the Example Project

Open the coreplexip_welcome.c -> main() tab again and scroll to the welcome message you saw; make a modification that you will recognize such as adding "Welcome to Embedded World 2019". Save this file, build and run again. You should see this modified message displayed on the terminal screen.

To debug, stop the running program and this time instead of clicking Run click the bug icon. You will see the upload progress in the same manner, but then the program will stop at the first line of main.c. This allows you to debug by skipping instructions, inspecting registers, variables and pending instructions and setting breakpoints.

Note: in the second part of this class we show how to secure this application with MultiZone Security, the first Trusted Execution Environment for RISC-V [6] – see Conference program "How to Secure a RISC-V Embedded System in Just 30 Minutes".

Note: a number of Xilinx FPGA boards will be given away to the participants of these classes.

REFERENCES

[1] Chisel - Constructing Hardware in a Scala Embedded Language https://chisel.eecs.berkeley.edu/

[2] RISC-V Foundation. "RISC-V Instruction Set Manual Volume I: User Level ISA Document Version 2.2". [Online]: https://riscv.org/specifications/

[3] SiFive, Inc. "Core Designer". 2019 [Online] https://www.sifive.com/core-designer

[4] SiFive, Inc. "Freedom Studio". 2018. [Online] https://www.sifive.com/boards

[5] Xilinx, Inc. "Xilinx Vivado Design Suite." 2019. [Online]: https://www.xilinx.com/products/design-tools/vivado.html

[6] Hex Five Security Inc. "MultiZone Security" 2018. [Online]: https://hex-five.com/