ECE 587 – Hardware/Software Co-Design
Lecture 21 RISC-V and Chipyard

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

March 30, 2026

Reading Assignment

- This lecture: RISC-V and Chipyard
- Next lecture: Systolic Array

Outline

Exploring Ideas for Hardware/Software Co-Design

- What if we want to build something like a Raspberry Pi but possibly for a target application?
  - A general-purpose processor to run software.
  - Memory system and standard I/O peripherals.
  - Hardware accelerators for tasks like machine learning.
- What about software?
  - Can we boot Linux and run familiar software stacks?
- What about hardware?
  - Can we just focus on RTL designs for CPU and accelerators?
  - Can we actually manufacture the chip?

Open Source Designs

- Open source software gives us a lot of freedom on the software stack.
  - Operating system and databases.
  - Languages and compilers.
  - Libraries and application software.
- To have the same level of freedom in hardware design and manufacturing, we need similar practices for
  - Instruction set standard.
  - RTL implementations for CPUs and accelerators.
  - RTL implementations for other SoC components.
  - EDA tools.
  - Chip fabrication process.
- While many of these items still require licensing and signing of non-disclosure agreements (NDA), in this lecture we discuss efforts toward open-source hardware designs.

Outline

RISC-V Instruction Set

- RISC-V is an open standard instruction set architecture (ISA).
  - Based on established reduced instruction set computer (RISC) principles that trades off code size for hardware simplicity.
- Developed at the University of California, Berkeley since 2010.
- Features
  - Open standard: no licensing fees, no non-disclosure agreements (NDA) to sign
  - Modular design: enable selection of extensions for custom processor designs
  - Supports a wide range of devices, from tiny microcontrollers to high-performance computing systems.
- A growing ecosystem of developers, tools, and resources supports RISC-V development.

RISC-V Architecture

- 32 registers including a dedicated zero register.
  - 16 for embedded variants
- Memory access through load and store instructions.
- ISA base: core of RISC-V all implementations must support
  - Weak Memory Ordering
  - 32/64/128-bit Base Integer Instruction Set with embedded variants.
- ISA extensions: optional modules
  - Integer Multiplication and Division
  - Atomic Instructions
  - Single-Precision Floating-Point
  - Double-Precision Floating-Point
  - And so on.

RISC-V Implementations

- Unlike RISC-V ISA that is open standard, implementations of RISC-V processors can choose to be open-source or not.
  - From microarchitecture designs to chip tapeouts.
- Rocket Chip: an open-source System-on-Chip design generator
  - In-order RISC-V core: Rocket
  - Out-of-order RISC-V core: BOOM
  - Supports the integration of custom accelerators
- SiFive: one of the first companies providing commercial RISC-V processor IP and silicon chips.
- And many other groups and companies with increasing adoption in commercial products.

Outline

Chipyard

- Chipyard is an integrated development framework for System-on-Chip (SoC) using RISC-V cores and accelerators.
  - Developed by the University of California, Berkeley.
- Aims to streamline the process of open-source RISC-V system design, simulation, and prototyping.
  - Targeted at small teams, e.g. graduate students and startups.
  - Instead of requiring a large team or initial investments in commercial EDA tools and IP cores to operate.

Chipyard's Ecosystem

- Designed to be compatible with different RISC-V tools and libraries, promoting interoperability within the RISC-V ecosystem.
- Supports adding custom extensions and accelerators, allowing users to tailor their chip designs for specific applications.
- With a growing community of developers and researchers contributing to the framework, enhancing its capabilities, and sharing their designs.
- However, as with any rapidly evolving open-source software, be prepared to troubleshoot issues that arise from incompatible versions of libraries and packages.

Chipyard Workflow

- Setup system for software development, e.g. to support C++ and JVM languages.
- Download and build hardware design tools to support languages like Verilog and Chisel.
- Download and build RTL generators for RISC-V cores and accelerators like Rocket Chip and Gemmini.
- Provide scripts to generate and simulate hardware designs that designers need to run after they update their designs every time.
- Download and build additional tools and libraries for cross-compilation, software simulation, FPGA-accelerated emulation, and VLSI design toward actual chip tapeouts.

SoC Design

- RISC-V Cores
  - Rocket Chip In-Order core
  - SonicBOOM Out-of-Order Superscalar core
- Accelerators
  - Hwacha Vector Accelerator
  - sha3 accelerator
  - NVDLA (NVIDIA Deep Learning Accelerator)
  - Machine learning accelerators like Gemmini
- Peripherals and other IP cores
  - TileLink for on-chip interconnect with cache coherence transactions
  - L2 Cache, JTAG, SPI, I2C, UART, Disk, Ethernet NIC, etc.
- Designers use Chisel, a hardware description library embedded in Scala, to specify their SoC designs.
  - Integrate above pre-built components with custom-built components.
  - Verilog implementations are then generated from Chisel.

Simulation and Emulation

- Preparation: generate RISC-V binaries.
  - Use a cross-compiler to compile C/C++ source code into RISC-V programs.
  - Link with a pre-compiled RISC-V baremetal library to run directly on RISC-V processors directly, or
  - Run in an operating system (OS) with a RISC-V port like Linux.
- Spike RISC-V ISA Simulator
  - Without any hardware implementation of RISC-V processors.
  - Implement functional model of RISC-V instructions.
  - Support accelerators via additional functional models.
  - Execute baremetal RISC-V programs or boot the RISC-V OS.
  - Cannot be used for performance evaluation since there is no hardware implementation.

Simulation and Emulation (Cont.)

- Verilator: open-source Verilog/SystemVerilog simulator
  - Simulate a RISC-V SoC implementation in Verilog generated by Chipyard.
  - Provide cycle-accurate performance estimation for baremetal RISC-V programs.
  - Could be too slow if need to boot a RISC-V OS.
- FireSim: open-source FPGA-accelerated simulation platform
  - Emulate RISC-V SoCs on large FPGA systems by automatically transforming and instrumenting Verilog implementations.
  - Make use of Amazon Web Services (AWS) EC2 F1 instances for cost saving.
  - Provide fast pre-silicon verification and performance validation of RISC-V processors at 10s-100s MHz.

Open-Source VLSI Design

- However, chip design and fabrication are still dominated by commercial tools and proprietary technologies.
  - Commercial EDA tools are needed to transform a Verilog design into GDSII layout for fabrication.
  - EDA tools need access to libraries, in particular process design kit (PDK) that defines manufacturing rules, which may require additional licensing and signing of NDAs.
- OpenROAD: open-source RTL to GDSII flow
  - With open-source tools for logic synthesis, place-and-route, as well as design-rule-checking (DRC) and GDSII generation.
  - A number of research and commercial applications simplify configuration and scripting of OpenROAD tools.
- SKY130 and GF180MCU: open-source 130nm/180nm PDKs
  - A collaborative effort between Google, GlobalFoundries, SkyWater Technology, and Efabless since 2020.