RISC-V Processor Design 🚀

Building Tiny Vedas

Marco Spaziani Brunella, PhD

Lecture 1

About Your Instructor

- PhD in Electrical Engineering
- Research focus on Computer Architecture and Digital Design
- 6+ Years of undergrad/grad teaching experience on RISC-V
- 10+ Years of experience in the RISC-V industry 👇🏻
  - Mostly Stream Data Processing Accelerators
  - Think AI Accelerators and Network Processing at 100+Gbps

Course Overview

- **Objective**: Design and implement a RISC-V processor (Tiny Vedas)
- **Prerequisites**:
  - None

DISCLAIMER 🚨

- The verbiage in this course is not meant to be a comprehensive guide to RISC-V or processor design
- It is meant to be a quick overview of the concepts and tools we will use
- The goal is to get you up to speed with the basics of processor design and RISC-V

What is a processor?

- A processor is a device that executes instructions upon data
- It is the brain of the computer
- We describe the *microarchitecture* (how it works) in an Hardware Description Language (HDL)
  - Verilog, VHDL, SystemVerilog, etc.
- The capabilities of a processor (or a family of processors) are defined in the Instruction Set Architecture (ISA)

What is RISC-V?

- Open-source Instruction Set Architecture (ISA)
- Created at UC Berkeley in 2010
- Key features:
  - Modular design
  - Simple and clean architecture
  - Free and open standard
  - Extensible instruction set

Why RISC-V?

- **Open Source**: No licensing fees or restrictions
- **Simplicity**: Clean-slate design without legacy baggage
- **Flexibility**: Base ISA + optional extensions
- **Growing Ecosystem**:
  - Industry adoption
  - Available tools and resources
  - Active community

Tools We'll Use

- **HDL**:
  - SystemVerilog
- **Simulation**:
  - Verilator for fast simulation
  - GTKWave for waveform viewing
- **Development Environment**:
  - VS Code (or similar) with HDL extensions
  - Git for version control
- **RISC-V Tools**:
  - RISC-V GNU Toolchain
  - Whisper ISS simulator

Course Structure

1. RISC-V ISA Basics
2. Single-Cycle CPU Design
3. Pipeline Fundamentals
4. Building Tiny Vedas
5. Testing and Verification
6. Performance Optimization

What You'll Learn

- RISC-V instruction set architecture
- Processor design principles
- RTL implementation techniques
- Hardware verification methods
- Performance optimization strategies
- Industry-standard tools and workflows

Accessing the Codebase

- The codebase is available on invitation at this LINK
- We need the email to send you the invitation

Development Environment Setup

- We assume you're using a Linux machine and you've got access to the codebase
  - You can easily replicate the steps on Macos or Windows
- Let's start with the following tools:
  - VS Code/Cursor (I use Cursor)
  - Verilator
  - GTKWave