RISC-V Processor Implementation on DE1-SoC FPGA

- # Project Plan: RISC-V Processor Implementation on DE1-SoC FPGA :

Team Size: 6 members

Duration: 12 weeks

Tools: Intel Quartus Prime, ModelSim, GitHub, DE1-SoC FPGA

Key Skills Covered : RTL Design, Verification (UVM/SystemVerilog), FPGA Synthesis, Computer Architecture, Collaboration.

---

1. Team Roles & Responsibilities

Assign roles to ensure market-relevant skills are practiced:

Role	Responsibilities	Skills Developed
Project Lead	Coordinate tasks, manage timelines, and oversee integration.	Project Management, System Integration
RTL Designers (x4)	Design core components (ALU, Control Unit, Register File, Memory/Pipeline).	Verilog/VHDL, Pipelining, RISC-V ISA
Verification Engineer	Develop testbenches, UVM frameworks, and coverage metrics.	SystemVerilog/UVM, Functional Coverage
FPGA Specialist	Synthesize design, handle timing closure, and deploy to DE1-SoC.	Quartus Prime, FPGA Optimization, SignalTap Debugging

---

2. Project Timeline

Phase 1: Research & Planning (Weeks 1-2)

- Define RISC-V subset (RV32I base ISA).
- Set up tools: Quartus, ModelSim, GitHub.
- Study DE1-SoC constraints (clock speed, I/O pins).

Phase 2: Component Design (Weeks 3-6)

- Split RTL tasks:
  - ***ALU :*** Arithmetic/logic operations.
  - ***Control Unit :*** Instruction decoding.
  - ***Register File :*** 32-register design.
  - ***Memory Unit :*** Cache/bus interface.
  - ***Pipeline (Optional)*** : 5-stage pipeline.

+ Weekly code reviews for alignment.

Phase 3: Integration (Weeks 7-8)

- Combine components into a single core.
- Validate data/control paths.

Phase 4: Verification (Weeks 9-10)

- Write SystemVerilog testbenches.
- Develop directed/random tests.
- Track code/functional coverage.

Phase 5: FPGA Deployment (Weeks 11-12)

- Synthesize with Quartus.
- Resolve timing issues.
- Test on DE1-SoC using GPIO (e.g., LEDs for output).

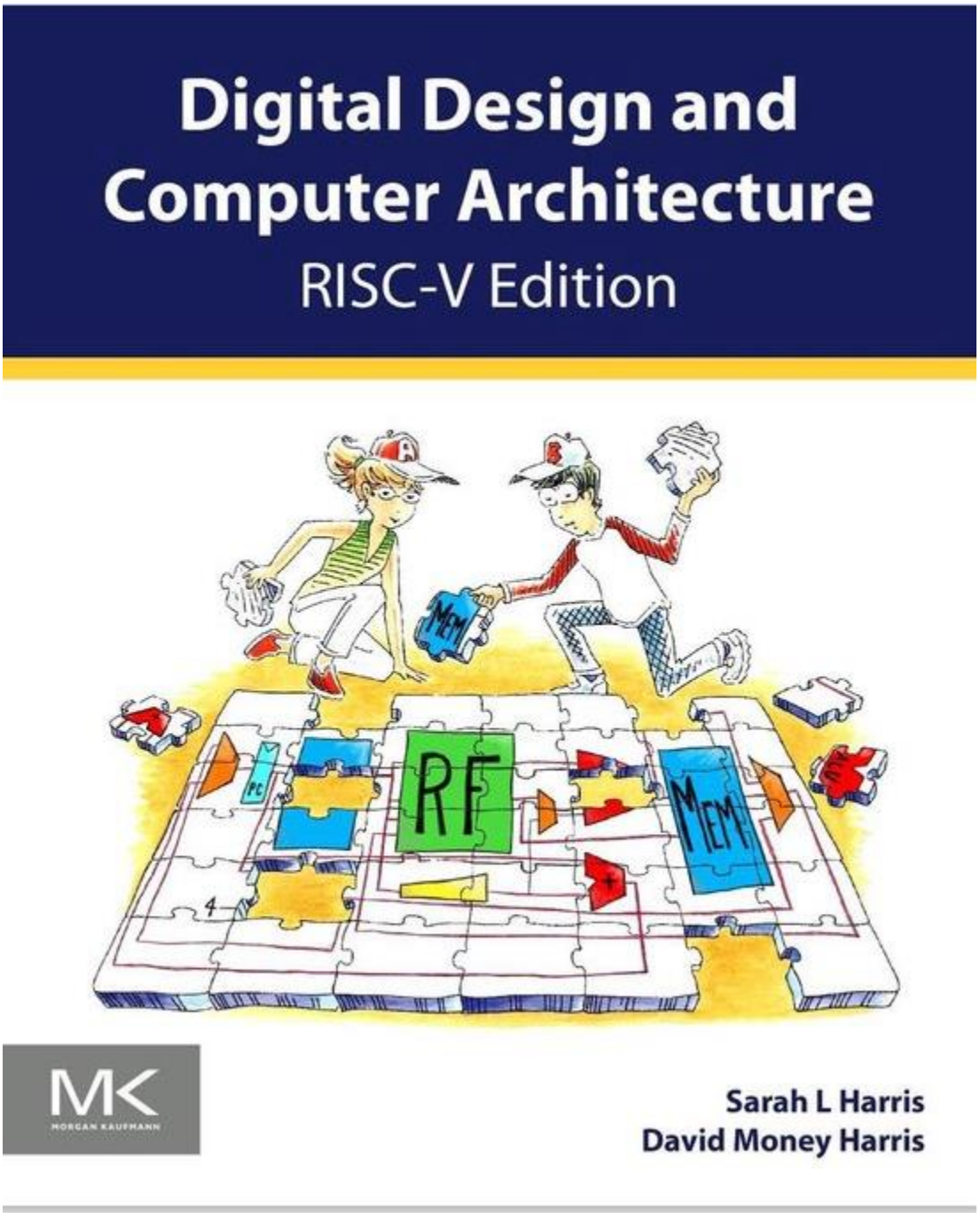
---

3. Learning Resources

A. Foundational Knowledge :

- ***Books***:

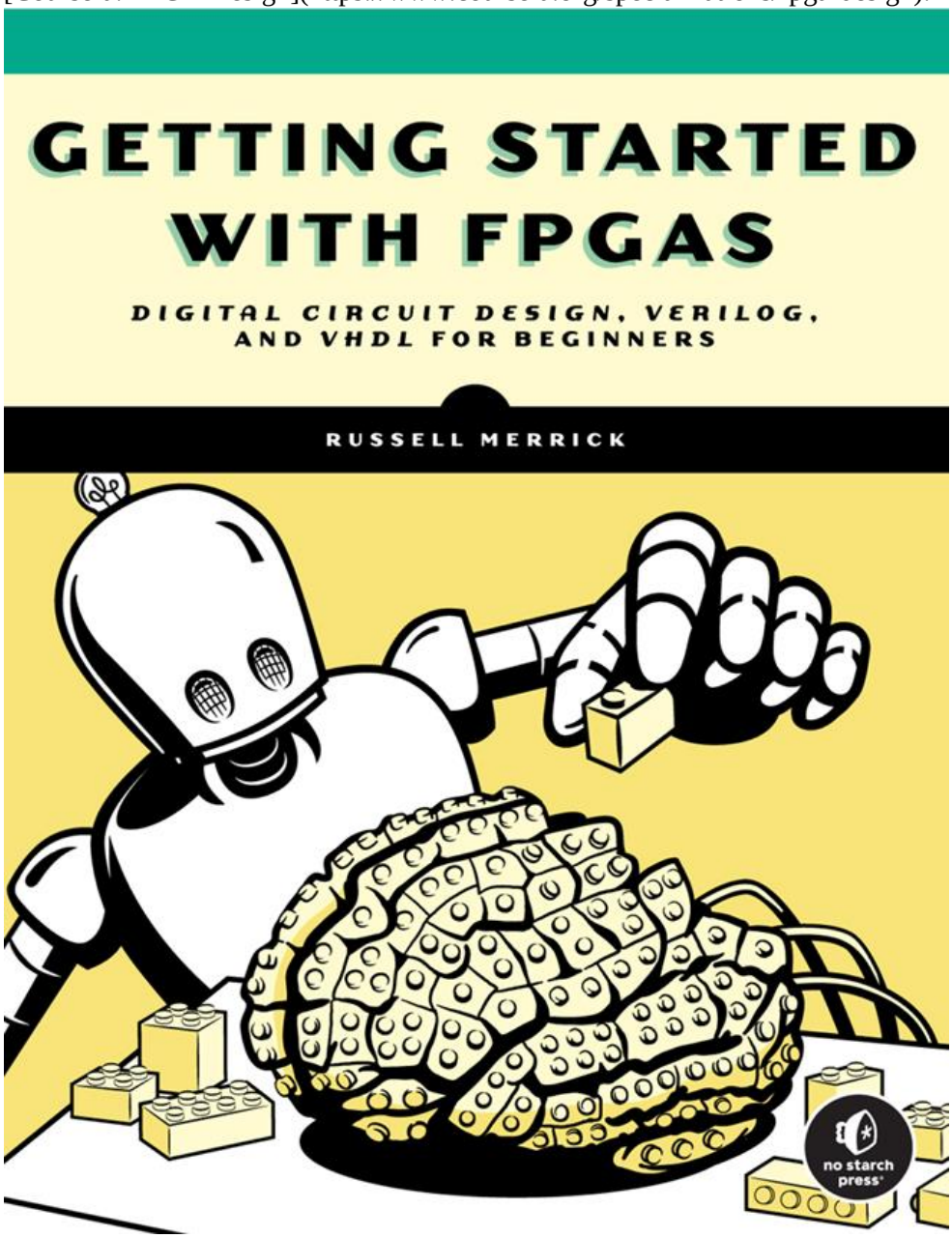
  - *Computer Organization and Design (Patterson & Hennessy)*.



  - RISC-V User-Level ISA Manual ➔ GitHub - riscv/riscv-isa-manual: RISC-V Instruction Set Manual

- ***Courses:***

- [Coursera: FPGA Design](https://www.coursera.org/specializations/fpga-design).



- [Nandland Verilog Tutorials](https://www.nandland.com).

-[Verilog Certification by intel] (Verilog HDL Basics - Intel Learning)

B. Tools & Repositories:

- ***Intel Quartus:*** Use the [DE1-SoC User Manual](https://www.terasic.com.tw) for pin assignments.

- ***GitHub Repos:***

  - [riscv-mini](https://github.com/ucb-bar/riscv-mini) (Scala Chisel, but good for reference).

- [PULP Platform](https://github.com/pulp-platform) (Open-source RISC-V cores).

- ***Simulation :*** Use ModelSim for pre-FPGA testing.

C. Verification :

- Learn UVM via [Verification Academy](https://verificationacademy.com).

- Example testbenches in [EDA Playground](https://www.edaplayground.com).

---

4. FPGA Implementation Guide

Steps for DE1-SoC :

1. Synthesis : Use Quartus to compile RTL.

2. Pin Assignment : Map I/O to DE1-SoC GPIO (e.g., LEDs for debug signals).

3. Timing Analysis : Resolve critical paths with pipeline registers.

4. Programming : Generate .sof file and load via Quartus Programmer.

5. Testing :Write assembly code to validate instructions (e.g., Fibonacci sequence).

---

5. Ensuring Marketable Skills

- ***Designers***: Gain expertise in RTL coding, pipelining, and low-power design.

- ***Verification Engineer:*** Master UVM, coverage-driven validation.

- ***FPGA Specialist:*** Learn synthesis, timing closure, and hardware debugging.

- ***All :*** Collaborate via GitHub (version control, issue tracking).

---

6. Example Deliverables

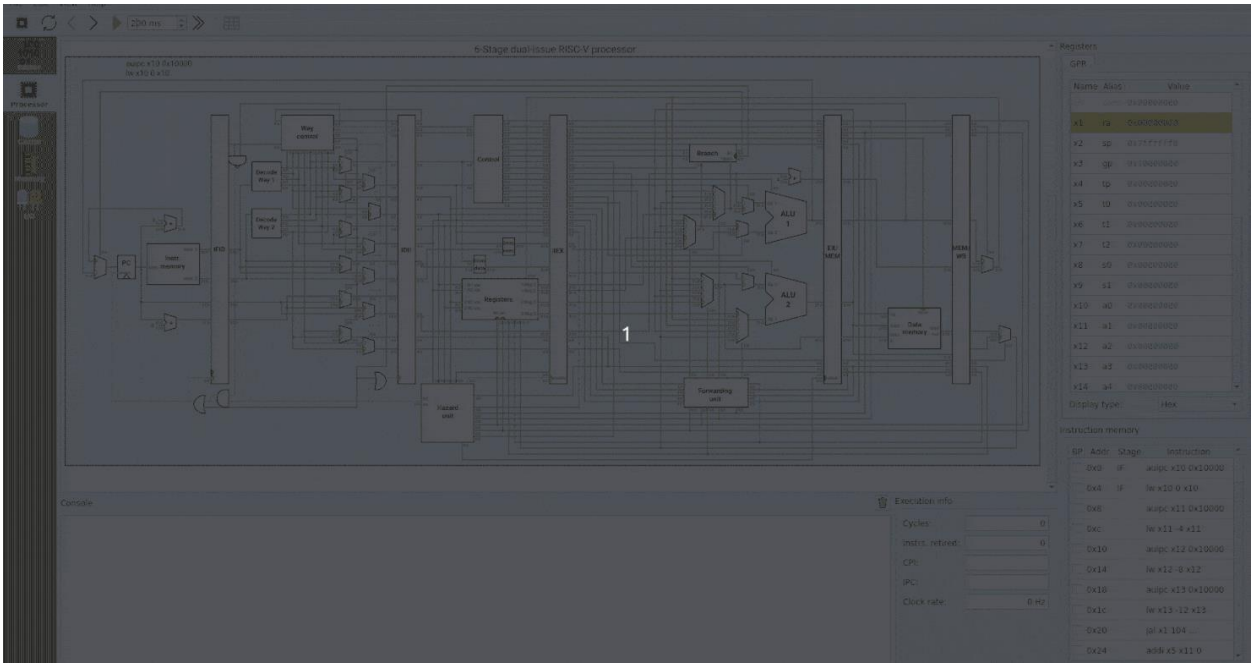
- GitHub repo with RTL, testbenches, and documentation.

- FPGA demo (e.g., LED blinks on DE1-SoC after successful program execution).

- Final report detailing challenges, coverage metrics, and lessons learned.

By following this plan, we will build a functional RISC-V processor while practicing industry-standard workflows. Adjusting timelines as needed, and prioritizing communication to resolve integration issues early!

- I recently came across a project on GitHub that caught my attention. It offers impressive simulations and animations for testing our architecture. I strongly encourage you to complete the tasks on time so we can use this tool for testing, as it will significantly enhance the project's marketability and further expand our knowledge.



[GitHub - mortbopet/Ripes: A graphical processor simulator and assembly editor for the RISC-V ISA]