



COMPLETE PROGRAMME GUIDE

RISC-V Verification Training

Standard Parts 1-3 and Advanced Parts 1+-3++ for processor, CPU and SoC verification.

Programme levels	Standard and advanced pathways
Standard programme	Parts 1-3 with 24 connected modules
Advanced programme	Parts 1+-3++; complete programme (+++)
PROGRAMME PURPOSE Develop practical verification capability from CPU and RISC-V foundations through advanced processor, CPU and SoC verification, using structured teaching, open-source workflows and applied exercises.	

Programme overview

As RISC-V adoption expands, verification teams need engineers who can reason across processor architecture, RISC-V ISA behaviour, UVM-based CPU verification and software-driven SoC integration. This programme guide presents two connected levels: a standard three-part pathway and an advanced three-part pathway.

The standard programme supports engineers building CPU and RISC-V verification foundations. The advanced programme is intended for engineers progressing into deeper processor microarchitecture, CPU corner-case verification and complete SoC verification. Participants may select an individual part or follow a connected programme, subject to experience and project objectives.

Programme pathways

Pathway	Coverage
Standard Part 1	CPU and RISC-V Basics - CPU architecture, verification strategies, instruction-stream generation, functional coverage and riscv-dv practice.
Standard Part 2	RISC-V CPU Verification - RISC-V ISA concepts, software toolchains, RARS, open-source UVM and riscv-dv workflows, architectural compliance and CPU microarchitecture.
Standard Part 3	RISC-V SoC Verification - software-driven SoC verification, self-checking tests, assertions, coverage, debug and sign-off.
Advanced Part 1++	Processor Microarchitectures and RISC-V ISA - advanced microarchitecture, selected ISA extensions, custom extensions and CVA6 block-level UVM verification.
Advanced Part 2++	RISC-V CPU Verification - privileged architecture, CSRs, traps, atomics, hazards, advanced instruction-stream generation, coverage and Spike reference-model checking.
Advanced Part 3++	RISC-V SoC Verification - formal, low-power, interconnect, cache coherency, firmware and OS boot, resilience, debug and sign-off.
Complete programme (+++)	The connected advanced pathway comprising Parts 1++, 2++ and 3++.

Naming: each individual advanced part uses "++". The complete three-part advanced programme uses "+++". Alpinum can confirm the appropriate entry point based on the participant's or team's experience, project context and learning objectives.

Standard programme learning outcomes

By the end of the relevant programme scope, participants should be able to:

- Describe current best-practice CPU and CPU-based SoC design verification strategies.
- Explain the principal methodologies, tools and languages used in CPU and CPU-based SoC verification.
- Apply those methodologies, tools and languages to a basic RISC-V CPU design and an SoC design based on RISC-V.
- Analyse a real RISC-V CPU or SoC and propose an appropriate design verification strategy.
- Discuss CPU and CPU-based SoC design verification topics confidently with engineering colleagues.
- Contribute more effectively to RISC-V CPU and SoC verification projects.

Who should attend the standard programme

- Graduate recruits and university placement students.
- Design and verification engineers who want to understand CPU and CPU-based SoC verification.
- Engineers moving into CPU verification, RISC-V verification or CPU integration.
- Engineering managers who need a working understanding of CPU and SoC verification strategies.

Standard programme prerequisites

Some experience with design verification and SystemVerilog using the Universal Verification Methodology (UVM) is beneficial. Contact Alpinum if you are unsure whether the complete standard programme, a focused standard part or an advanced pathway is the best fit.

How the training works

Instructor-led teaching: Live online sessions establish the architecture, methodology and verification flow. On-demand access may also be available through current course listings.

Guided examples: Participants see workflows demonstrated before applying them independently.

Practical exercises: Exercises cover code generation, simulation, coverage, compliance, software-driven SoC testing, debug and sign-off.

Review and feedback: Joint classroom sessions help participants consolidate techniques and relate the exercises to real project work.

Part 1: CPU and RISC-V Basics

PART FOCUS

Establish the CPU architecture and verification foundation before applying it through functional coverage and riscv-dv exercises.

1. Introduction to CPU

Approximately 1 hour of teaching

1.1 CPU architectures and Instruction Set Architectures

- What an Instruction Set Architecture (ISA) is and why ISAs are needed for code and binary portability.
- Introduction to CPU architectures.

1.2 CPU microarchitectures

- A simple CPU block diagram.
- Basic operation of a simple CPU microarchitecture.

2. How do we verify a CPU?

Approximately 2 hours of teaching

2.1 Overall CPU verification strategies

- Compare different approaches to CPU verification and their relative advantages and disadvantages.
- Explain the hierarchical, simulation-based verification approach used during the course.

2.2 Unit-level verification

- How standard SystemVerilog and UVM testbenches can be used for unit-level verification.
- Examples covering fetch and branch prediction, decode and dispatch, integer execution, load/store execution, floating-point execution, instruction caches, data caches, Memory Management Units (MMUs) and Control and Status Registers (CSRs).
- More specialised units may include vector units and reorder buffers.
- Detailed unit-level verification is specific to each design and is therefore outside the scope of this course.

2.3 CPU-level verification

- How binary machine code can be used to verify a CPU.
- How assembler code can be used for CPU verification.
- How instruction-stream generation can be used to produce assembler tests.
- Example of a UVM testbench that runs generated assembler, converted into binary, on the CPU.

2.4 System-level integration

- Introduction to the objectives and techniques used to verify the integration of a CPU into an SoC.
- CPU integration verification is covered in more detail later in the programme.

3. Basic CPU-level verification and tooling

2 hours of teaching plus examples

3.1 Instruction-stream generators

- How instruction-stream generation works.
- Why constrained pseudo-random generation is needed.
- The types of constraints required for CPU verification.

3.2 Open-source riscv-dv tooling

- Overview of riscv-dv as an example of an instruction-stream generator.
- Examples of running riscv-dv.
- riscv-dv is covered in greater detail later in the programme.

3.3 Impact of adding custom instructions

- How adding custom instructions can affect the verification environment and process.
- This topic is explained at a high level because implementation details are specific to each design.

4. CPU microarchitectures for faster code execution

4 hours of teaching

4.1 Reviewing different microarchitecture styles

- Review and comparison of different microarchitecture styles.
- Examples include RISC, CISC and VLIW architectures.
- These architectural approaches are explored in greater detail later in the programme.

5. Verifying CPU microarchitectures

4 hours of teaching plus examples

5.1 Building functional coverage models for a microarchitecture

- Identify the microarchitecture features that need to be verified.
- Define suitable functional coverage points.
- Extend an existing functional coverage model.
- Run tests and review the resulting coverage.

5.2 Generating tests to hit functional coverage points

- Review the existing constraints.
- Extend the constraints to hit coverage points in the expanded functional coverage model.
- Run targeted tests and review their contribution to coverage closure.

6. riscv-dv practical exercises

At least 8 hours of independent study plus at least 1 hour of classroom review

Practical work

- Extend the functional coverage model and constraints file.
- Run tests and work towards coverage closure.
- Complete the exercises independently with access to support and feedback.

6.1 Joint classroom review and feedback

- Jointly review the completed exercises.
- Discuss the results, challenges and possible improvements.

Part 2: RISC-V CPU Verification

PART FOCUS

Apply the foundation to RISC-V software flows, open-source verification tooling, architectural compliance and advanced CPU microarchitecture.

7. Introduction to RISC-V

Approximately 2 hours of teaching

7.1 RISC-V ISA overview

- Understanding RISC-V instruction formats.
- RISC-V ISA modularity.
- RISC-V privilege modes.
- Introduction to the RISC-V memory model.

7.2 Assembly language for RISC-V

- Introduction to RISC-V assembly language.
- Review of example assembler programs.

8. Writing and running RISC-V programs

Approximately 1 hour of teaching plus examples

8.1 RISC-V software toolchains

- Overview of RISC-V software toolchains.
- Flow diagrams illustrating the software toolchain.
- The process for creating binary files from assembler and C code.

8.2 RISC-V Assembler and Runtime Simulator

- Introduction to the RISC-V Assembler and Runtime Simulator (RARS).
- Write and run example assembler programs for the basic RISC-V ISA.
- Run the example programs in RARS.

8.3 Converting C code into RISC-V assembler

- Write and run example C programs for the basic ISA.
- Convert C code into RISC-V assembler.
- Run the programs using RARS.
- Complete practical exercises involving assembler and C programs.

9. RISC-V practical exercises

At least 4 hours of independent study plus at least 1 hour of classroom review

Practical work

- Write and run code for RISC-V.
- Run RISC-V assembler and C programs using RARS.
- Complete the exercises independently with access to support and feedback.

9.1 Joint classroom review and feedback

- Jointly review the completed exercises.
- Discuss the results, difficulties and lessons learned.

10. Verifying CPU integration in an SoC

Approximately 2 hours of teaching

10.1 System-level integration verification

- Explain how software running on a mini-SoC can be used to verify CPU integration.
- Understand why higher-performance platforms may be needed to run some programs.
- Review examples of compliance suites and benchmarks.

11. Basic RISC-V verification and tooling

2 hours of teaching plus examples

11.1 Open-source riscv-dv tooling

- Detailed overview of riscv-dv.
- Examples of running riscv-dv.
- Review the generated code and how it is executed in simulation.
- Review pass and fail results.
- Review the resulting functional coverage.

11.2 Generating and running a first riscv-dv test

- Run riscv-dv.
- Simulate the generated output.
- Check the results and review the resulting functional coverage.

11.3 Reviewing the riscv-dv functional coverage model

- Review the functional coverage supplied by riscv-dv.
- Understand how the coverage model relates to verification objectives.

11.4 Reviewing the riscv-dv constraints model

- Understand how constraints are defined in riscv-dv.
- Modify selected constraints.
- Review the effect of those changes on generated tests and coverage.

12. riscv-dv practical exercises

At least 8 hours of independent study plus at least 1 hour of classroom review

Practical work

- Run the complete riscv-dv flow.
- Generate tests and simulate the output.
- Review results and functional coverage.
- Complete the exercises independently with access to support and feedback.

12.1 Joint classroom review and feedback

- Jointly review the completed riscv-dv exercises.
- Discuss the results and potential improvements.

13. Architectural compliance

2 hours of teaching plus examples

13.1 Understanding architectural compliance

- What architectural compliance means.
- Differences between RISC-V architectures and their compliance requirements.
- Introduction to architectural compliance suites.
- Detailed review of an example compliance suite.

13.2 RISC-V compliance suites

- Run an example RISC-V compliance suite.
- Review and interpret the results.

14. CPU microarchitectures for faster code execution

4 hours of teaching

14.1 Reviewing different microarchitecture styles

- Compare RISC, CISC and VLIW architectures.
- Understand why RISC architectures, and RISC-V specifically, are used.

14.2 RISC-V registers

- Overview of the RISC-V register set.
- Register addressing modes, including direct, indirect and immediate addressing.
- General-purpose and special-purpose registers.

14.3 Pipelining

- What pipelining is and which performance problems it addresses.
- Verification challenges introduced by pipelining.
- Techniques used to address these challenges, including out-of-order execution, branch prediction and register renaming.

14.4 Additional microarchitectures

- Superscalar processors and how they are designed.
- Multithreaded processors.
- Vector processors and Vector/SIMD instruction-set extensions.
- Multicore processors.

15. Final session

Course review

Review

- Review the principal RISC-V CPU verification concepts.
- Review the practical exercises.
- Discuss how the techniques can be applied to real verification projects.

16. Additional material

Supplementary content

Formal verification

- Introduction to how formal verification can support RISC-V CPU verification.

Part 3: RISC-V SoC Verification

PART FOCUS

Verify a RISC-V CPU within an SoC using software-driven testing, assertions, functional coverage, debug and sign-off evidence.

17. Introduction to SoC verification

Approximately 2 hours of teaching

17.1 SoC verification

- The principal SoC verification tasks and challenges.
- Differences between IP-level and SoC-level verification.
- Metrics and sign-off criteria used in SoC verification.

18. Introduction to the training SoC

Approximately 2 hours of teaching

18.1 The SoC used during the course

- Overview of the architecture of the training SoC.

18.2 The SoC verification environment

- Overview of the testbench.
- Tools and flows used to run tests and simulations.
- Adding checkers using assertions.
- Writing self-checking tests.
- Defining and implementing SoC functional coverage models.

19. SoC practical exercises

At least 4 hours of independent study plus at least 1 hour of classroom review

Practical work

- Review the environment used to run RISC-V code on the SoC.
- Understand the process for updating an existing test or writing a new test.
- Run existing tests.
- Update and run existing tests.
- Write and run new tests.
- Ensure that tests are self-checking.
- Add functional coverage.

19.1 Joint classroom review and feedback

- Jointly review the completed SoC exercises.
- Discuss the results and possible improvements.

20. SoC practical debug exercises

At least 2 hours of independent study, including classroom review

Debug workflow

- Understand the bug-finding, triage and debug workflow.
- Run a test on an SoC containing a known bug.
- Triage and debug the known bug.
- Review the debug process and findings.

21. How do we verify an SoC CPU?

Approximately 3 hours of teaching

21.1 Overall SoC verification strategies

- Compare different approaches to SoC verification and their relative advantages and disadvantages.
- Explain the SoC verification approach used during the course.

21.2 Verification of specific SoC features

- SoC integration verification.
- SoC reset and boot, including power-on sequence verification.
- SoC clock-control and clock-gating verification.
- SoC power-control verification.
- SoC interrupt verification.
- SoC system-control verification.
- SoC use-case verification.

22. SoC verification and sign-off practical exercises

At least 4 hours of independent study plus at least 1 hour of classroom review

Practical work

- Write and run tests for a range of SoC features.
- Collect functional coverage and sign-off metrics.
- Prepare an SoC sign-off report.

22.1 Joint classroom review and feedback

- Jointly review the completed exercises.
- Discuss the collected evidence and sign-off results.

23. Final session

Course review

Review

- Review the principal RISC-V SoC verification concepts.
- Review the practical exercises.
- Discuss how the techniques can be applied to real SoC verification projects.

24. Additional material

Supplementary content

Formal verification

- Additional guidance on how formal verification can support RISC-V SoC verification.

ADVANCED PROGRAMME PARTS 1++ TO 3++

Advanced programme overview

The advanced series extends Alpinum's standard three-part RISC-V verification programme. Participants may select an individual advanced part or follow the complete +++ pathway from architecture and ISA through CPU and SoC verification.

Advanced part	Focus	Coverage
Part 1++	Processor Microarchitectures & RISC-V ISA - Advanced	Microarchitectures, six selected RISC-V ISA extensions, custom extensions and CVA6 block-level UVM verification.
Part 2++	RISC-V CPU Verification - Advanced	Privileged architecture, CSRs, traps, atomics, pipeline hazards, advanced ISGs and Spike-referenced UVM checking.
Part 3++	RISC-V SoC Verification - Advanced	Fourteen advanced modules covering sign-off, formal, UPF, AXI, coherency, firmware/OS boot, resilience and future trends.

How the advanced series fits together

- Part 1++ establishes the advanced processor microarchitecture and ISA foundation, then applies it to block-level UVM verification of selected CVA6 core blocks.
- Part 2++ targets high-risk CPU corner cases using advanced instruction-stream generation, functional coverage and a UVM/Spike reference-model methodology.
- Part 3++ extends the work to complete RISC-V subsystem and SoC verification across formal, low-power, interconnect, memory, software-driven and sign-off concerns.

Delivery and access

The advanced parts are available through on-demand recordings, with live instructor-led delivery offered subject to current availability. Dates, access periods, delivery arrangements and pricing are published with the active course listing. Eligible university students may request reduced-price access to selected courses.

Who should attend

- Verification engineers progressing beyond introductory CPU and RISC-V verification.
- CPU, subsystem and SoC engineers responsible for coverage closure, debug or sign-off.
- Engineers working with UVM, instruction stream generators, reference models, formal methods or software-driven verification.
- Engineering leads who need a connected view of advanced RISC-V CPU and SoC verification strategy.

Advanced programme prerequisites

Participants should have completed the relevant standard training or have equivalent practical experience in SystemVerilog, UVM, processor or CPU verification, or SoC verification. Alpinum can confirm the appropriate advanced entry point against the participant's experience and project objectives.

PART 1++

Processor Microarchitectures & RISC-V ISA - Advanced

Advanced architecture and ISA depth, followed by selected CVA6 block-level UVM verification.

Course focus

Part 1++ covers processor microarchitecture design in depth: single-cycle, multi-cycle, pipelined and out-of-order execution, including hazard analysis, forwarding, and critical-path and performance evaluation. It then covers six selected RISC-V ISA extensions M, A, F, C, Bitmanip and V down to the underlying hardware algorithms.

Core technical coverage

- Hardware algorithms behind the M extension: shift-and-add multiplication, Radix-4 Booth multiplication, and restoring and Radix-4 division.
- Floating-point arithmetic, the RISC-V C compressed instruction format, and the Bitmanip extension family.
- The RISC-V V vector extension: vector registers, vl/SEW/LMUL, masking, tail policies, and unit-stride, strided and indexed memory operations.
- How standard and custom extensions are named, ordered and encoded, with worked custom-extension design examples.
- Out-of-order execution concepts including dynamic scheduling, Tomasulo's algorithm, register renaming and speculative execution.

CVA6 block-level UVM verification

The advanced part also includes additional lessons and slides covering block-level UVM verification of the CVA6 Multiplier, Serial Integer Divider and Compressed Decoder.

- Detailed design overviews for each block.
- Block interfaces and functionality.
- Verification plans and execution examples.
- Hands-on practical exercises demonstrating complete block verification.

PART 1++

Learning outcomes and agenda

The architectural and ISA foundation for the CPU- and SoC-level work in Parts 2++ and 3++.

What you will gain

- Deep coverage of single-cycle, multi-cycle, pipelined and out-of-order processor microarchitectures and six selected RISC-V ISA extensions: M, A, F, C, B and V.
- Understanding of multiplication and division algorithms, floating-point arithmetic, compressed instruction decoding, bit manipulation and the RVV vector execution model.
- Hands-on grounding in dynamic scheduling, Tomasulo's algorithm, register renaming and speculative execution.
- A strong architecture and ISA foundation that carries into Parts 2++ and 3++.
- Verification IP (VIP) guidance relevant to your design.

Agenda

- Processor microarchitectures and RISC-V ISA extensions M, A, F, C, B and V.
- RISC-V ISA extensions in depth, from motivation and C-code examples through hardware algorithms.
- Standard versus custom extensions: naming and ordering rules, encoding and opcode space, with worked design examples.
- Knowledge-check quizzes and review materials, with instructor-led review where included in the current delivery.

PART 2++

RISC-V CPU Verification - Advanced

Corner-case CPU verification using advanced ISGs, functional coverage, UVM and the Spike ISS reference model.

Course focus

Part 2++ takes a deep, corner-case-level look at four high-risk areas in RISC-V CPU verification: privileged architecture and CSR verification; exception and interrupt verification with pipeline interaction; atomic instruction LR/SC and AMO corner cases; and pipeline hazard, forwarding and flush verification.

Participants see how instruction stream generators target these behaviours across privilege transitions, traps, atomics and hazards, and how a UVM environment verifies CSR correctness cycle by cycle against the Spike ISS golden reference model.

Hands-on methodology

- Design ISG sequences in online, offline, cycle-accurate and retire-accurate modes to target privileged transitions, traps and hazard corner cases.
- Write realistic SystemVerilog/UVM-style ISG constraints and build functional coverage models, including cross-coverage, for CSR access, atomics and hazards.
- Verify CSR correctness with a UVM environment comprising the DUT, ISG-driven instruction stream, monitor and scoreboard, checked against the Spike ISS reference model.

What you will gain

- Advanced ISG-based test generation for RISC-V CPUs.
- In-depth verification coverage of privileged architecture and CSR behaviour, exceptions and interrupts, atomic corner cases, and pipeline hazard, forwarding and flush scenarios.
- A working UVM CSR verification methodology checked cycle by cycle against Spike.
- Verification IP (VIP) guidance relevant to your RISC-V CPU design.

PART 2++

Agenda

A practical deep dive into advanced RISC-V CPU verification risks and evidence.

- Privileged architecture and CSR verification; exception and interrupt verification; atomic LR/SC and AMO corner cases; and pipeline hazard, forwarding and flush verification.
- Advanced ISG techniques: online and offline and cycle-accurate and retire-accurate generation, plus realistic constraint design.
- Corner-case deep dives: trap flow and privilege transitions, CSR field and instruction hazards, LR/SC and AMO failure modes, and RAW, load-use and store hazard verification.
- Functional coverage and constraints modelling.
- Guided exercises and review materials, with instructor-led review where included in the current delivery.

Project relevance

This advanced part is designed to build project-ready expertise for demanding RISC-V CPU verification environments, with emphasis on targeted stimulus, reference-model checking, functional coverage and evidence-driven debug.

PART 3++

RISC-V SoC Verification - Advanced

Fourteen connected modules for advanced subsystem and SoC verification, closure and sign-off.

Course focus

Part 3++ builds on Parts 1++ and 2++. Participants learn to verify a complete RISC-V subsystem or SoC across fourteen advanced modules.

1. Introduction to Advanced RISC-V SoC Verification.
2. Advanced SoC Verification Planning, Closure and Signoff.
3. Berkeley Out-of-Order Machine (BOOM).
4. ASIC Verification: A Resource-Limited Search for Bugs.
5. Cross-Subsystem and System-Level Integration and Scenario-Driven SoC Verification.
6. Formal Applications at SoC Level.
7. Formal Connectivity in SoC Integration.
8. Power-Aware SoC Verification and UPF-Based Validation.
9. Formal Verification for UPF and Low-Power.
10. Advanced SoC-Level Interconnect Verification: Multi-Master AXI and Memory System Interaction.
11. Memory System and Cache Coherency Verification in RISC-V SoCs.
12. Firmware-Driven, OS Boot and Workload-Based Verification.
13. Performance, Stress and Resilience Verification.
14. Additional Future Verification Trends: digital twins, in-field/runtime verification, and quantum and post-quantum hardware verification.

PART 3++

Learning outcomes and agenda

Advanced SoC verification strategy spanning formal, low-power, interconnect, memory, software and resilience.

What you will gain

- End-to-end SoC verification strategies for RISC-V designs.
- Fourteen in-depth modules spanning formal methods, UPF and low-power validation, multi-master AXI, cache coherency, firmware and OS-boot verification, resilience and fault injection, and emerging trends.
- Hands-on exercises for advanced SoC verification, coverage and sign-off.
- Practical guidance for debugging and triaging SoC issues.
- Verification IP (VIP) guidance relevant to your RISC-V SoC design.

Agenda

- Modules 1-4: advanced SoC verification foundations, planning and sign-off, BOOM out-of-order architecture, and ASIC bug-hunting strategy.
- Modules 5-9: scenario-driven system integration, formal SoC and connectivity verification, and power-aware and UPF verification.
- Modules 10-12: multi-master interconnect, cache coherency, and firmware and OS-boot verification.
- Modules 13-14: performance, stress and resilience verification, and emerging verification trends.
- Review materials and feedback options, as specified in the current delivery.

Choose an advanced pathway

Select the advanced part that matches your current engineering objective, or follow the complete +++ programme for a connected route from architecture and ISA to CPU and SoC verification.

Part 1++	Processor Microarchitectures & RISC-V ISA - Advanced
Part 2++	RISC-V CPU Verification - Advanced
Part 3++	RISC-V SoC Verification - Advanced
Complete +++	All three advanced parts as one connected programme

Enquiries and current access

Visit alpinumconsulting.com or contact the Alpinum training team for current availability, delivery options, access periods and programme selection guidance.

Important course information

- Course delivery, dates, pricing and access periods may vary by cohort or on-demand listing.
- The advanced parts are intended to extend the standard RISC-V verification programme and assume relevant prior knowledge or equivalent experience.
- Alpinum can advise whether an individual ++ part or the complete +++ programme is the best fit for your objectives.

Current advanced access

Use the current listings below to confirm access, delivery arrangements and pricing:

- [Part 1++ current access](#)
- [Part 2++ current access](#)
- [Part 3++ current access](#)
- [Complete Advanced Programme \(+++\)](#)

University access

Eligible university students may request reduced-price access to selected Alpinum semiconductor training programmes using a valid academic email address. [View the University Student Access Programme](#)

Discuss the right programme scope

To recommend the most suitable starting point, Alpinum will consider:

- Your current experience with SystemVerilog, UVM, processor architecture, CPU verification and SoC verification.
- Whether your priority is architecture and ISA depth, CPU corner-case verification, or complete SoC verification.
- Whether you need an individual part, a connected programme, or a tailored team pathway and delivery arrangement.

NEXT STEP

Contact Alpinum with your experience, team requirements, project context and learning objectives. We will help identify the most appropriate standard part, advanced part or complete programme.

[View RISC-V Verification Training](#)

Complete Programme Guide | Version 2026