RISC-V Unified Database [1]

Derek R. Hower[1], Afonso Oliveira[2]

[1]Qualcomm Technologies, Inc.
[2]Synopsys, Inc.

Abstract

*This work presents the RISC-V Unified Database (UDB), a step towards a unified, machine-readable source of truth for the RISC-V Specification. Currently, the RISC-V specification (including both ratified and de-facto parts) is scattered across multiple repositories and cloud storage files. Little of it is machine-readable, and in many cases information is duplicated in error-prone ways. This presents a barrier to further RISC-V growth as it is increasingly difficult to understand (and especially **verify**) the complex and growing specification. By gathering the specification in a single database, UDB provides the means to quickly find information and to generate artifacts directly from an authority. Towards that end, UDB currently includes over ten generators, including ones that produce ISA manuals, instruction and CSR indices, and an Instruction Set Simulator (ISS) – all from the same source. Though tremendous progress has been made in the last year, UDB is still a work in progress, and we are actively looking for increased community participation.*

Introduction

RISC-V has undergone an extraordinary transformation, evolving from an academic project with five foundational extensions [2] into a global industry standard featuring *nearly 200* ratified extensions [3]. This rapid growth has exposed challenges in the ecosystem's infrastructure. Namely, the standard is scattered across multiple repositories and cloud storage files, has inconsistent style, and is not machine-readable – making interpretation and verification labor-intensive and error-prone. This puts RISC-V at risk of fragmentation from vendor incompatibility.

To address these systemic challenges, the RISC-V Unified Database (UDB) contributes:

- A unified, machine-(and human-)readable source-of-truth
- Cross-validation against established resources to ensure data correctness
- A framework to generate artifacts directly from the source-of-truth
- A mechanism to *overlay* customization on the standard – true to RISC-V's core – that allows vendors to use the tool to generate artifacts for their specific implementation.

UDB's generated-from-the-source artifacts include, but are not limited to, a complete ISA manual enhanced with an alphabetical list of all instructions, CSRs, and extensions, a fast (comparable to Spike [4]) Instruction Set Simulator (ISS), and documentation for commercial RISC-V custom extensions [5] using the above-mentioned overlay mechanism.

UDB is gaining traction among industry leaders and early adopters, including Qualcomm, Synopsys, and the RISC-V's Certification Steering Committee (CSC), each of which leverage UDB's capabilities to generate tailored outputs with reduced effort and better results compared to previous processes. As a result, UDB is well-positioned to serve as the foundation for the next generation of the RISC-V specification ecosystem.

Challenges with RISC-V Specification Ecosystem

The RISC-V specification ecosystem currently relies on multiple disconnected repositories and specification formats. Additionally, key documents, such as the ISA and assembly manuals, are written in AsciiDoc – a format not meant for structured data capture – forcing reliance on copy-pasting or rewriting for derivative document generation and precluding data verification.

The intricate relationships between instructions, extensions, Control and Status Registers (CSRs), architectural parameters (most of which are unnamed), and profiles further amplify the complexity of this challenges. Instructions belong to one (or more) extensions, yet tracking which instructions are introduced, modified, or forbidden across different extensions is difficult. CSR interactions with specific extensions and privilege levels are often unclear. Profiles group extensions into structured subsets of the ISA, but the lack of consistent cross-referencing makes it difficult to determine which components apply to a given subset.

Version management further compounds this complexity. A typical CPU design incorporates multiple extensions, each evolving independently as new versions are ratified. For example, a current design may specify `RV32IM_Zba1p0_Zbb1p0_Zbs1p0_Zicsr2p0`, but as the ecosystem progresses beyond "1.0" and "2.0" versions, maintaining consistency across implementa-

tions and documentation will become more difficult.

All of this creates challenges for entire ecosystem. For example, it's difficult for vendor documentation to stay up-to-date with the standard given the reliance on copy/paste. Tool developers (*e.g.*, compilers) must create and maintain their own machine-readable representations. And implementations struggle to verify a core against documentation that is not easily understood or referenced.

UDB Source-of-Truth

UDB structures information into extensions, instructions, CSRs, architectural parameters, profiles, and certificates with unambiguous attributes and relationships. Moreover, UDB allows users to create overlays that enable customizations to the base specification, but leave the base specification intact.

UDB data is represented in YAML using a strict schema enforced by JSON Schema. YAML is machine-readable, friendly for human consumption, and is supported by every mainstream programming language from C to Python. Each component is structured with attributes that define its properties and relationships while maintaining modularity to allow entities to be referenced, inherited, or extended without duplication. For example, instructions *refer* to defining extensions, profiles *include* ISA extensions, and architectural parameters highlight implementation choices. The schema supports versioning, dependency resolution, and formal descriptions, making it suitable for generating structured documentation, generating and configuring ISSs, and integrating with tools.

Table 1 compares UDB with the current approach. While profile manuals and opcode databases existed before, they were either incomplete or manually maintained. The instruction and CSR indices, which provide structured cross-referencing, were entirely absent in previous documentation efforts but are now being actively developed within UDB. By unifying these outputs, UDB enhances clarity, accessibility, and long-term maintainability across the ecosystem.

Data Status and Verification. UDB currently includes encoding data for all ratified RISC-V instructions found in `riscv-opcodes` and the *GNU* and *LLVM* toolchains. Formal execution semantics exist for the many extensions, though some notable ones, like Vector (V) and Hypervisor (H), are missing, incomplete, or unverified. The UDB community is working to fill these gaps, and funding has been identified.

UDB cross-verifies its data against existing defacto sources-of-truth. UDB's encoding data is verified against `riscv-opcodes`. Paired with an existing UDB generator that produces the same outputs as `riscv-opcodes`, UDB is technically ready to supplant the function of `riscv-opcodes` should the community decide to move that direction.

Artifact	Current	UDB
ISA Manuals	Prose	Prose + Appendices
ISS / Formal model	Incomplete	Generator complete, incomplete semantics
Opcode Database	Incomplete	Complete (superset of `riscv-opcodes`)
Profile Manuals	Prose	Prose + Appendices
CSC Documents	Using UDB	-
Instruction Index	Non-existent	Complete mnemonics/encoding, Incomplete semantics/descriptions
CSR Index	Non-existent	Incomplete

Table 1: *Comparison of current RISC-V artifacts versus UDB*

UDB instruction data is also validated against the information in *LLVM*'s instruction tables. We hope to expand that work soon to include LLVM-compatible disassembly information and relocation data. Ultimately, we believe that UDB could serve as a source-of-truth for generated LLVM artifacts.

Future Work / Call to Action!

UDB has been successfully bootstrapped and shows a clear path forward, though challenges remain. The community is working hard to fill data gaps and enhance the infrastructure. There is also an effort underway to link UDB data to normative text in the prose manuals, providing an auditable trail for the data sources. We strive to advance UDB to the point that it could eventually serve as the official source of truth for the RISC-V specification and (generated) artifacts. The UDB community, like RISC-V as a whole, strives to be open and collaborative, and we welcome more contributors to our BSD-3-licensed project.

References

[1] Hower, Derek, et. al. *The RISC-V Unified Database*. URL: `https://github.com/riscv-software-src/riscv-unified-db`.

[2] Andrew Waterman et al. "The RISC-V instruction set manual, volume i: Base user-level isa". In: *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62* 116 (2011), pp. 1–32.

[3] RISC-V International. *The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA*. Version 20240411.

[4] RISC-V Software Source. *Spike RISC-V ISA Simulator*. 2024. URL: `https://github.com/riscv-software-src/riscv-isa-sim`.

[5] Albert Yosher and Derek Hower. *Qualcomm uc extensions*. 2025. URL: `https://github.com/quic/riscv-unified-db/releases/tag/Xqci-0.6`.