Building commercially relevant open source silicon: The many aspects of Ibex

G. Chadwick[1], A. Kurth[1], M. van der Maas[1], H. Callahan[1], lowRISC contributors*

[1]lowRISC C.I.C.

Abstract

Ibex [1] is an open source RV32IMCB CPU with a 2 or 3 stage pipeline (configurable at synthesis) targeting embedded and security applications. It started life as zero-riscy [2] from ETH Zürich who contributed it to lowRISC [3], and its development is now part of the OpenTitan [4] project.

lowRISC's aim with Ibex (as with all of the IP it maintains) is not only to make high quality RTL freely available under a permissive licence, but to ensure it meets the needs of commercial users with the goal of enabling wide-scale adoption of open source silicon designs. To achieve this, it is not enough to simply make the RTL available. The code needs to meet high quality standards, be in a form that is usable and familiar for industry, and to be credible. A core part of credibility is achieving full verification up to the standard required for commercial tape outs. Crucially the verification framework itself as well as test and coverage plans are fully open source so they can be properly scrutinised. Regular regression results are published to demonstrate IP maturity and underscore our long-term commitment to it.

In this proposal we present the collaborative engineering lowRISC and the community have done on Ibex. There is an overview of the design and configuration options; details of the full UVM based testbench, test suite and coverage plan; and a discussion on lowRISC's quality standards.

Ibex Design Overview

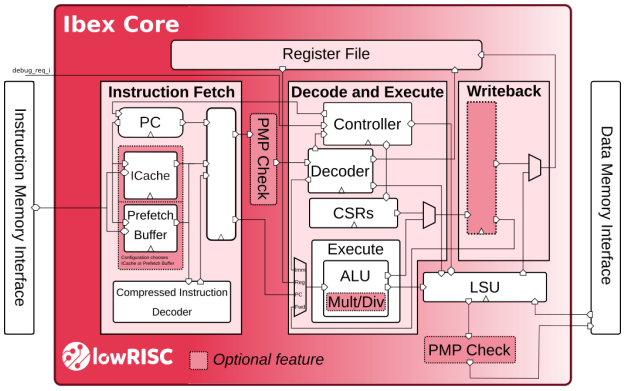

Figure 1: *The Ibex pipeline*

Ibex's three pipeline stages (see Fig. 1) are instruction fetch (with optional instruction cache), decode and execute, and writeback. The pipeline register between the second and the third stage is optional. Machine mode and user mode are implemented as well as a configurable number of PMP regions and the optional Smepmp extension [5]. The 'B' bit manipulation [6] has multiple configurations with different sub-extensions. The M extension with a hardware multiplier and divider are optional with various multiplier implementations trading off size and performance, up to a single cycle implementation.

As the CPU employed by OpenTitan [4], Ibex has a number of optional security hardening features to mitigate potential attacks. For example, a dual-core lockstep configuration where two cores are instantiated is supported. Both see the same inputs, and their outputs are cross checked so that a security alert can be triggered if they differ. The register file content, ICache content, and memory transaction data are protected with an ECC. The ICache content is scrambled using the PRINCE cipher [7].

Ibex is written in SystemVerilog following the lowRISC style guide [8]. This style guide aims to produce neat, consistent, readable RTL that will work as intended across a broad range of EDA tools.

Verifying Ibex

The primary Ibex DV environment is a UVM based testbench written in SystemVerilog. It runs programs that are randomly generated by RISCV-DV [9] along with UVM sequences that control timing on the external interfaces, generate bus errors, and stimulate interrupts and debug requests. As is usual for UVM test benches all sequence behaviour is configurable and controlled by constrained randomization.

Checking is performed through co-simulation, where Ibex is run in lockstep with an instruction set simulator (ISS). Execution of the ISS and Ibex are cross-checked (including data side memory accesses) and an error signalled if there are any discrepancies. The method employed allows full use of interrupts and debug requests and stimulation of bus errors.

Spike [10] is employed as the ISS used for co-simulation. Some small modifications are needed to integrate it into the verification environment. The

*Many, beyond the named authors, have contributed to Ibex see https://github.com/lowRISC/ibex/blob/master/CREDITS.md

co-simulation interface is designed to be generic to allow the use of alternate ISS programs, such as the RISCV Sail model [11].

Functional coverage is seperated into two categories:

Architectural Coverage Provided by RISCV-DV. This examines executed instruction traces and gathers coverage relating to instructions executed, including various corner cases of each instruction.

Micro-architectural Coverage Gathered by the testbench by probing the internals of Ibex. This covers micro-architectural behaviours such as different stalls and hazards, interrupt and debug behaviour and more. A separate set of covergroups gather metrics over PMP behaviour including seeing all possible configurations and combinations of success and failure plus interesting corner cases (e.g. accesses crossing PMP region boundaries).

For the 'opentitan' configuration of Ibex (3 stage, with security hardening; bit-manipulation; 16 PMP regions; and single cycle multiplier) a high degree of verification has been achieved, with 94% code coverage and 93% functional coverage across a regression run of 1380 tests with a pass rate of 94% at the time of writing. All failing tests are due to verification environment issues and not RTL bugs. There are no known RTL bugs.

Setting the Quality Bar

High standards are set for Ibex. To meet these criteria, lowRISC employs a broad range of industry-standard techniques. Code review is employed for all contributions which helps enforce the style guide and maintain consistency. Continuous integration (CI) testing must be passed before a contribution is accepted, this runs a variety of quick tests including a full lint check which prevents broken code from being merged. Nightly regressions, using the full DV test suite, are run and monitored. Test failures are triaged and fixed as part of day to day work to maintain a high pass rate. Coverage is monitored and work done to tweak tests and introduce new ones to achieve the desired verification closure quality.

Documentation and 'out of the box' experience are also crucial. Ibex is extensively documented, covering core integration along with micro-architectural, implementation and verification details. A Verilator [12] simulation demonstrating use of Ibex is provided. The Ibex Demo System [13] provides a fuller example that can be run on an FPGA board.

Making the Most of Open Source

A traditional user of commercially available IP may be tempted to treat Ibex in the same manner; an opaque box that implements a required set of features that is acquired and integrated into a large design, the only difference being Ibex is obtained entirely free of charge.

The open source nature of development can offer users so much more. A user can modify RTL to suit their needs, adding and removing functionality as needed. As the testbench and all DV collateral is available, such changes can be verified quickly by the user.

Users can contribute changes upstream, allowing them to introduce new features without needing to maintain them long-term. By working with lowRISC and the community, democratised innovation and maintenance possibilities become available.

Lacking the commercial need to sell updated versions of IP, users can quickly benefit from new features, updates to the latest RISC-V specifications and bug fixes rather than waiting for yearly (or longer) releases. With no incentive to sell different configurations as different products users can simply choose amongst the many configuration options that suit their needs and easily experiment with others.

Author Biographies

G. Chadwick is Digital Design Lead at lowRISC and holds a Ph.D. from Cambridge University.

A. Kurth is a Senior Engineer at lowRISC and holds a Doctor of Science from ETH Zurich.

M. van der Maas is a Senior Engineer at lowRISC and holds a Ph.D. from Cambridge University.

H. Callahan is an Engineer at lowRISC and holds an M.Eng. from Oxford University.

References

[1] *Ibex*. URL: https://github.com/lowrisc/ibex.

[2] P.D. Schiavone et al. "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications". In: *PATMOS 2017*.

[3] *lowRISC*. URL: https://lowrisc.org.

[4] *OpenTitan*. URL: https://opentitan.org.

[5] *Smepmp extension*. Version 1.0 12/2021. RISC-V International.

[6] *RISC-V Bit-Manipulation ISA-extensions*. Version 1.0.0-38-g865e7a7. RISC-V International.

[7] Julia Borghoff et al. "PRINCE A Low-Latency Block Cipher for Pervasive Computing Applications". In: *Advances in Cryptology – ASIACRYPT 2012*. 2012, pp. 208–225.

[8] *lowRISC style guides*. URL: https://github.com/lowRISC/style-guides.

[9] *RISCV-DV*. URL: https://github.com/chipsalliance/riscv-dv.

[10] *Spike*. URL: https://github.com/riscv-software-src/riscv-isa-sim.

[11] *Sail RISC-V Model*. URL: https://github.com/riscv/sail-riscv.

[12] *Verilator*. URL: https://www.veripool.org/verilator/.

[13] *A demo system for Ibex*. URL: https://github.com/lowRISC/ibex-demo-system.