POLITECNICO DI TORINO
Repository ISTITUZIONALE

# Case Study: Integrating RISC-V Zicfiss Shadow Stack Support for Control-Flow Integrity on the CVA6 Processor

*Original*
Case Study: Integrating RISC-V Zicfiss Shadow Stack Support for Control-Flow Integrity on the CVA6 Processor / Farnaghinejad, B., Sanchez, E.. - ELETTRONICO. - (2026), pp. 1-2. (27th IEEE Latin American Test Symposium (LATS 2026) Florianópolis (BRA) 17-20 March 2026) [10.1109/lats70329.2026.11480312].

*Availability:*
This version is available at: 11583/3010131 since: 2026-04-21T08:48:38Z

*Publisher:*
IEEE

*Published*
DOI:10.1109/lats70329.2026.11480312

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

13 October 2026

# Case Study: Integrating RISC-V Zicfiss Shadow Stack Support for Control Flow Integrity on the CVA6 Processor

Behnam Farnaghinejad and Ernesto Sanchez
Department of Control and Computer Engineering, Politecnico di Torino, Torino, Italy
Email: {behnam.farnaghinejad, ernesto.sanchez}@polito.it

***Abstract*—Control-flow integrity (CFI) is a fundamental defense against control-flow hijacking attacks that corrupt return addresses, and hardware-supported shadow stacks have been adopted in several mainstream processor architectures to provide backward-edge protection. The RISC-V Zicfiss extension introduces analogous CFI support into the RISC-V ecosystem, yet its practical integration into RISC-V processors remains largely unexplored. This paper presents a case study on integrating Zicfiss shadow stack support into the CVA6 processor, an open-source application-class RISC-V core representative of systems executing rich operating systems and user applications. The extension's architectural behavior is implemented and validated across privilege levels, establishing a correct and specification-aligned baseline for enforcing backward-edge CFI. This work enables a systematic study of control-flow protection mechanisms in advanced RISC-V designs. It provides a foundation for future investigations into robustness and hardening against physical and microarchitectural attack vectors.**

***Index Terms*—RISC-V, Zicfiss, shadow stack, control-flow integrity, CVA6, hardware security**

## I. Introduction

Control-flow integrity (CFI) is a well-established defense against control-flow hijacking attacks that manipulate return addresses or indirect control transfers [1]. In particular, violations of backward-edge control flow, where a function returns to an unintended address, can lead to arbitrary code execution.

Hardware-assisted shadow stacks provide an effective mechanism for enforcing backward-edge CFI by maintaining a protected copy of control-flow return state. In RISC-V, architectural support for such mechanisms has been introduced through the *Zicfiss* extension [2].

Despite its specification, the practical integration of Zicfiss into RISC-V processors remains largely unexplored. This paper presents a case study on integrating Zicfiss shadow stack support into the CVA6 processor, an open-source 64-bit RISC-V core capable of running complete operating systems [3]. The implementation is validated across machine, supervisor, and user execution contexts under controlled enablement conditions, establishing a functional baseline for backward-edge CFI on a representative application-class RISC-V design.

This work was funded by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

## II. Background

Backward-edge control-flow integrity aims at ensuring that functions return only to legitimate call sites, mitigating attacks that corrupt return addresses stored on the software stack [1]. Shadow stacks enforce this property by maintaining a protected copy of return addresses that is checked on function return.

The RISC-V Zicfiss extension introduces architectural support for backward-edge CFI via dedicated shadow-stack instructions and architectural state [2]. Zicfiss is currently specified as a draft extension in the RISC-V Privileged Architecture and defines privilege-aware enablement and exception signaling via dedicated Control and Status Registers (CSRs) used for control-flow violations, while leaving microarchitectural implementation choices to processor designers. CVA6 is an open-source, 64-bit application-class RISC-V core that implements the privileged ISA and supports complete operating systems such as Linux, making it a representative target for evaluating Zicfiss integration [3].

## III. Methodology

Zicfiss support is integrated into the CVA6 core by extending call and return control logic to maintain a commit-stage shadow-stack controller backed by an internal memory structure, consistent with the RISC-V privileged architecture specification [2]. The integration preserves compatibility with standard execution when the extension is disabled. In the validation setup, Zicfiss execution in U-mode is explicitly enabled by setting the Shadow Stack Enable (SSE) bit of the `senvcfg` CSR from supervisor mode, as required by the specification's privilege-gated activation model.

Figure 1 provides a software-level view of Zicfiss usage, illustrating the interaction between application code, system software responsibilities, and hardware enforcement. Control-flow violations detected by shadow stack checks raise architectural exceptions that are handled by system software.

CVA6 is an out-of-order core that may speculatively execute control-flow instructions and later squash them on misprediction or exception. To preserve architectural correctness, shadow-stack updates are tied to the same retirement/commit conditions that make the corresponding call/return architecturally visible. Concretely, speculative effects of shadow-stack operations are not allowed to survive kills or flushes: a shadow-stack push is applied only when the associated call retires,

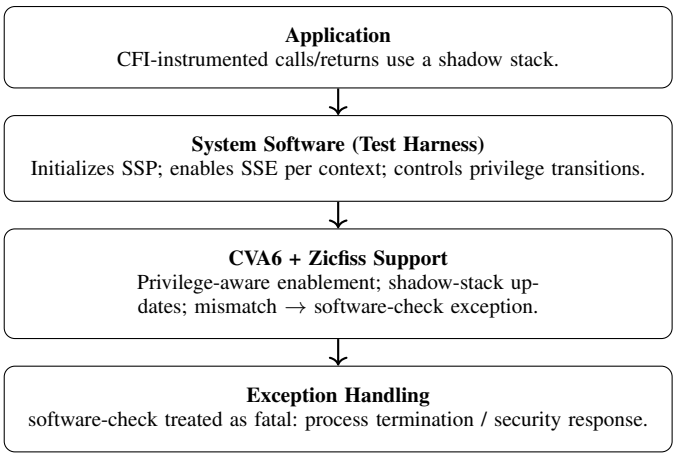


Fig. 1: Software–hardware interaction for Zicfiss-based backward-edge CFI in the evaluated CVA6 setup.

and a shadow-stack pop/check is finalized only when the corresponding return retires. If an instruction is killed (e.g., due to branch misprediction, exception, or pipeline flush), any pending shadow-stack effect is discarded to maintain consistency between the architectural and shadow stacks.

The scope of this work is limited to architectural correctness and privilege-aware behavior of the Zicfiss extension. Memory-system features such as shadow stack page types in the MMU, PMP/PMA enforcement, and interaction with a full operating system are not evaluated at this stage. This deliberate scoping allows isolation of core architectural behavior before exploring system-level integration and security hardening.

## IV. Experimental Results

This section presents the experimental results obtained from evaluating the Zicfiss integration on the CVA6 processor. The evaluation focuses on architectural correctness and control-flow integrity enforcement, rather than performance or attack resilience, which are outside the scope of this case study.

The integrated design is evaluated using RTL simulation of the modified CVA6 core. A dedicated bare-metal test suite is executed to exercise Zicfiss functionality under controlled conditions. The tests are designed to observe architectural state changes and exception behavior directly, enabling precise validation of shadow stack semantics.

The experimental setup covers execution in machine, supervisor, and user modes, including transitions between privilege levels. System software responsibilities, such as shadow stack initialization and Shadow Stack Pointer (SSP) management, are emulated within the test framework to reflect realistic usage scenarios. No operating system or virtual memory support is enabled in this evaluation, allowing isolation of architectural behavior.

The test suite validates the correct behavior of the shadow stack mechanism under normal execution and error conditions. Correct backward-edge control-flow enforcement is confirmed by verifying that return address mismatches are detected and reported through architectural exceptions. Additional tests confirm that shadow stack functionality is enabled or disabled according to privilege-level configuration and that incorrect usage results in precise and expected exception signaling.

TABLE I: Functional coverage summary of the test suite. Virtualized (VS/VU) CSR cases are not evaluated.

| Validated property | Status |
|---|---|
| Shadow stack enable/disable behavior | ✓ |
| Privilege-aware operation (M/S/U) | ✓ |
| SSP CSR accessibility rules (M/S/U) | ✓ (partial) |
| Backward-edge CFI mismatch detection | ✓ |
| Exception behavior and reporting | ✓ |
| Alignment checks for SSP | ✓ |
| MMU SS-page protections (pte.xwr=010) | ✗ (future) |
| PMP/PMA SS enforcement rules | ✗ (future) |
| Fault-injection robustness evaluation | ✗ (future) |

Table I summarizes the functional coverage achieved by the test suite. The results demonstrate correct shadow stack pointer handling, privilege-aware enforcement, and consistent exception behavior across all supported execution modes. These findings establish a functional baseline consistent with the evaluated subset of the Zicfiss specification on an application-class RISC-V core.

Enabling the Zicfiss shadow-stack extension inserts one `SSPUSH` on each call and one `SSPOPCHK` on each return, introducing two additional instructions per protected call/return pair. Area overhead was evaluated on an FPGA for a shadow-stack depth of 256 entries (64 bits each). The shadow-stack controller accounts for 4,374 LUTs and 16,384 FFs within the complete core implementation (54,544 LUTs and 46,651 FFs total), corresponding to approximately 8.0% of LUTs and 35.1% of FFs. No change in the post-synthesis critical path was observed after integration.

## V. Conclusion and Future Work

This paper presented a case study on integrating the RISC-V Zicfiss shadow stack extension into the CVA6 application-class processor. The integration and functional validation demonstrate that backward-edge control-flow integrity can be correctly enforced across privilege levels on a complex RISC-V core, establishing a functional baseline consistent with the evaluated subset of the Zicfiss specification.

Future work will focus on extending the evaluation to include operating system support, virtual memory protections for shadow stack pages, and performance and area characterization on FPGA and ASIC targets. In addition, robustness of Zicfiss-based control-flow enforcement under physical and microarchitectural fault models will be investigated, together with potential hardening techniques to strengthen control-flow integrity guarantees in adversarial environments.

## References

[1] "Control-Flow Integrity: Precision, Security, and Performance: ACM Computing Surveys: Vol 50, No 1," ACM Computing Surveys (CSUR), Accessed: Feb. 06, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3054924

[2] "The RISC-V Instruction Set Manual, Volume II: Privileged Architecture," RISC-V International, Document Version 20260205.

[3] F. Zaruba and L. Benini, The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. (Jul. 2019). doi: 10.1109/TVLSI.2019.2926114.