Comparative Analysis of Functional Verification Strategies for RISC-V with Potential Use in CV-QKD Systems

Gustavo Santiago Sousa[1,2*], Angelo Gabriel dos Santos[1], Henrique Nunes Teixeira[1,2], Linton Thiago Costa Esteves[1], Wagner Luiz Alves de Oliveira[2], Jés de Jesus Fias Cerqueira[2],Nelson Alves Ferreira Neto[1]

[1]QuIIN – Quantum Industrial Innovation, EMBRAPII CIMATEC Competence Center, SENAI CIMATEC University; [2]Graduate Program in Electrical and Computer Engineering, Federal University of Bahia; Salvador, Bahia, Brazil

With the advancement of quantum computing, traditional cryptographic algorithms have become vulnerable, demanding solutions based on physical principles, such as continuous-variable quantum key distribution (CV-QKD). One of the main challenges for the practical adoption of CV-QKD lies in the digital signal processing (DSP) and post-processing stage, which imposes strict requirements for performance, reliability, and robustness against noise and faults. In this context, architectures based on the open fifth-generation Reduced Instruction Set Computing (RISC-V) standard have proven promising for building application-specific instruction set processor (ASIP) and specialized system-on-chip (SoCs), due to their flexibility, customization capabilities, and growing support ecosystem. However, ensuring the functional reliability of such systems requires the application of rigorous verification processes. This paper presents a comparative study of four complementary functional verification approaches: modular block-based verification, instruction-level transaction-based verification, black-box verification, and cycle-by-cycle comparison. Each approach was evaluated in terms of coverage, implementation complexity, and fault visibility. The study also highlights how these methods complement each other when applied in layered verification environments. The results show that no single approach can efficiently meet all requirements. High-level strategies are better suited for integration testing and broad functional validation, while low-level techniques are essential for localized debugging and precise timing analysis. This reinforces that verification success relies more on strategic integration than on a single optimal method. The findings indicate that adopting hybrid, layered solutions provides better scalability, supports regression testing, and improves both coverage and alignment with the security and performance goals required by CV-QKD systems.

Keywords: CV-QKD. RISC-V. Functional Verification. UVM. Digital Design.

With the ongoing advancement of quantum computing, traditional cryptographic algorithms such as Rivest-Shamir-Adleman (RSA) and elliptic curve cryptography (ECC) are expected to become insecure, requiring new secure communication paradigms such as CV-QKD, which guarantees security based on the laws of quantum physics [1,2]. However, the high-speed DSP and intensive post-processing steps represent a bottleneck in the practical implementation of CV-QKD [3-6].

The use of RISC-V instruction set architecture (ISA) [7], due to its customizable and open-source nature, enables the development of ASIP and SoCs optimized to accelerate these DSP and postprocessing tasks [8-10]. However, to ensure the reliability of such systems, robust functional verification of the RISC-V core is essential, presenting a complex challenge due to the presence of pipelines, complex sequential logic, and temporal dependencies.

To manage this verification complexity, different methodologies exist [11,12], among which the Universal Verification Methodology (UVM) is commonly adopted [13-15]. UVM enables the creation of scalable and reusable verification environments across different levels of abstraction [16,17]. As shown in Figure 1, which illustrates a standard structure UVM testbench environment, each component has a specific function and can be reused or extended according to verification needs. In the RISC-V context, UVM enables hybrid verification strategies, combining internal signal analysis with input-output (black-box) based tests [18-20].

Received on 21 March 2026; revised 26 May 2026.
Address for correspondence: Gustavo Santiago Sousa. Av. Orlando Gomes, 1845 - Piatã, Salvador – BA – Brazil, Zipcode: 41650-010. E-mail: gustavo.sousa@fieb.org.br.

J Bioeng. Tech. Health 2026;9(6):579-585

This work presents an initial-stage comparative study of four verification approaches applied to a basic RISC-V implementation, aiming to evaluate the trade-offs between abstraction level, verification visibility, and methodological complexity. The approaches investigated — ranging from modular testing to clock-cycle-accurate comparison — provide foundational insights into their respective strengths and limitations. While this study serves as an early investigation, its conclusions are designed to inform and guide the selection of optimal verification strategies for the development of more complex, security-dedicated ASIPs and SoCs for CV-QKD DSP and postprocessing. The ultimate goal is to establish a robust verification methodology that ensures functional reliability in future high-performance, realtime systems where any flaw could compromise cryptographic integrity.

Materials and Methods

During the development of functional verification, various strategies were explored, each operating at a different level of abstraction. The approaches evaluated included modular block-based verification, instruction-level transaction-based verification, black-box verification, and cycle-by-cycle verification. These strategies support the development of unit testing within the context of hardware verification, enabling their application to both individual components and complete system implementations through the isolated analysis of specific design elements. The following sections provide a detailed description of each strategy.

Block-Level

Verification This technique involved the isolated verification of individual processor components such as the arithmetic logic unit (ALU), decode unit, comparators, and multiplexers [22]. Stimuli were applied directly to the inputs of these blocks, and the outputs were monitored to verify compliance with the expected behavior. Tests were implemented using simple testbenches and could be complemented with internal SystemVerilog assertions (Figure 2).

Transaction-Based Verification

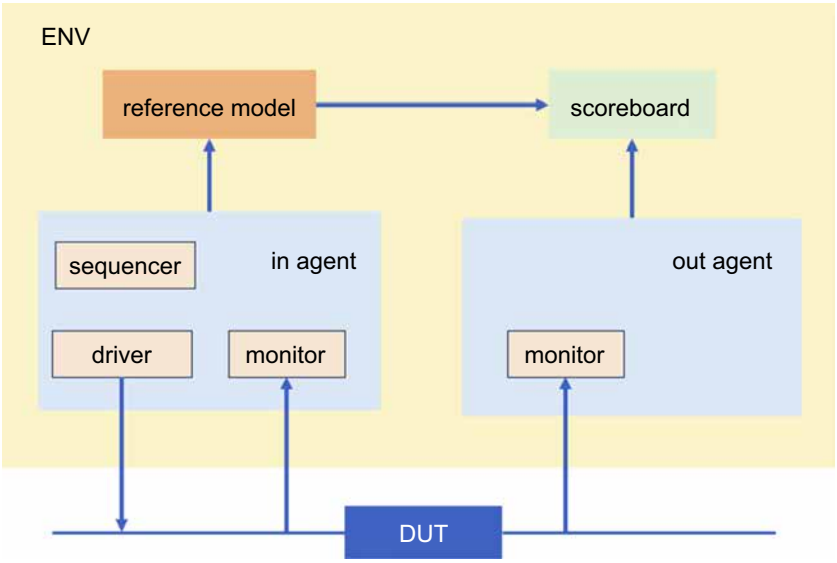
This approach focused on verifying individual instructions in isolation. In this setup, a single instruction was sent to both the processor and the reference model over a specific number of clock cycles, depending on the processor's latency. The reference model processed the instruction instantaneously, while a monitor captured the processor’s response after the natural pipeline latency. The output signals were then compared against the results produced by the reference model (Figure 3).

Black-Box Verification

In this strategy, instruction sequences were composed of LOAD operations, intermediate instruction set operations, and concluding STORE instructions [23]. Registers were initialized with known values, and the final data written to memory was used for verification. No internal signals were monitored; evaluation was based solely on observable results after program execution. This approach is particularly useful in scenarios where internal signals – such as ALU outputs or intermediate data – are not externally accessible. This is especially relevant in the context of RISC-V, where there is no official hardware implementation, only an ISA specification [7].

Different cores may expose or conceal internal signals in various ways, as discussed in Oleksiak and colleagues [18]. Verification at this level of abstraction is therefore highly reusable across different RISC-Vcompliant implementations, representing the most generic form of functional verification. This approach is also inherently robust to pipeline and timing variations, as checking is performed only after execution has completed (Figure 4).

Figure 1. Standard UVM environment setup.



Source: Liu and colleagues [21].

Figure 2. Block-level verification strategy.

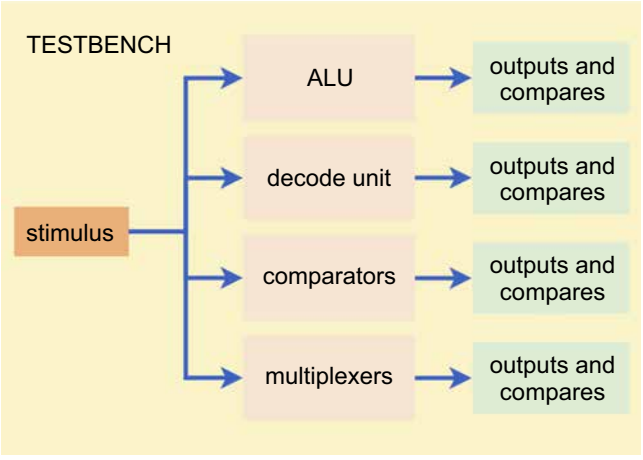


Figure 3. Transaction-based verification at the instruction level.

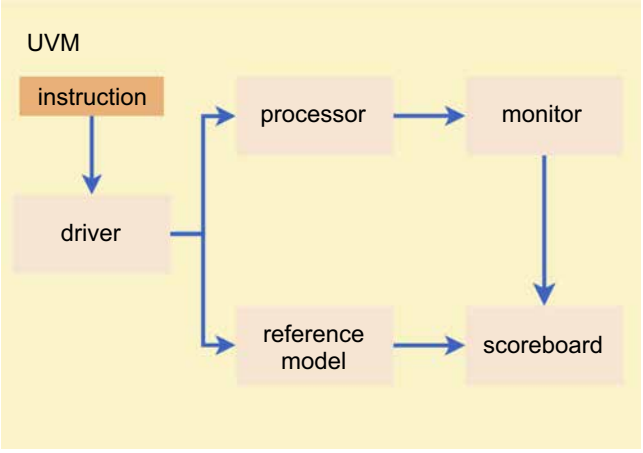
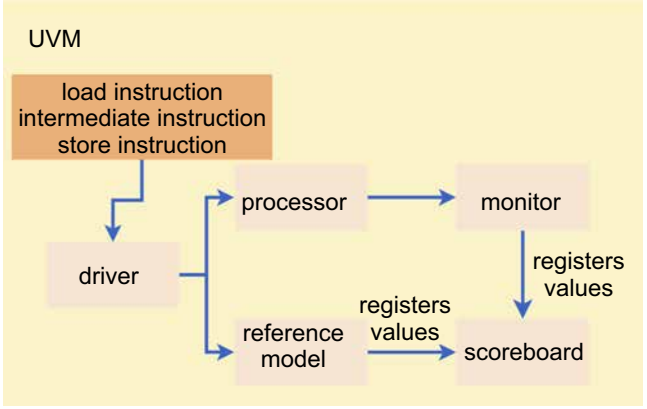


Figure 4. Black-box verification using memory output comparison.



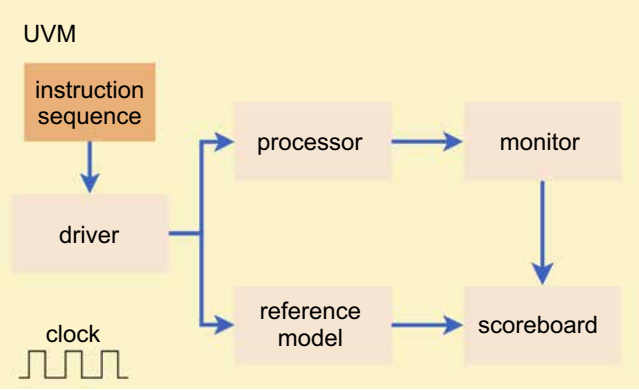
Cycle Accurate Verification (Clock Level)

This approach combined elements from the two previous strategies in a hybrid method. Instruction blocks were composed of LOAD operations, randomly selected arithmetic or logical instructions, and concluded with STORE operations, similar to the black-box strategy. However, in this version, internal signals – such as memory addresses, data to be written, and the instruction address (programCounter) – were monitored and compared on a cycle-by-cycle basis. These signals were also used as analysis parameters in Schiavone and colleagues [19]. To support this, the reference model was extended to simulate a

complete processor, capable of handling stalls, forwarding, and other typical pipeline behaviors.

Although this approach is not fully generic across all RISC-V implementations, it enables precise cycle-by-cycle comparison of input and output signals for this specific processor. This strategy offers a balanced trade-off between observability and practicality, facilitating error identification and providing deeper insights into processor behavior (Figure 5).

Figure 5. Cycle-accurate verification with internal signal monitoring.



Results and Discussion

Throughout the verification of the RISC-V processor, each of the previously described approaches was applied at different stages of the project. This section presents the main results observed for each strategy, along with the practical challenges encountered during their implementation.

Modular Verification

Modular verification proved to be highly effective in identifying localized faults and providing an initial validation of the core modules before full processor integration. It exhibited low implementation complexity, reduced development time, and facilitated debugging. However, it does not cover the integration between components, timing effects, hazards, or complex sequential behavior typical of processor pipelines, thus limiting its applicability in global behavior testing.

Instruction-Level Transactional Verification

The transactional approach enabled effective analysis of simple instructions by comparing the output signals of the device under test (DUT) with those of the reference model, and it facilitated the detection of decoding errors. However, it presented difficulties with instructions that affect control flow or require forwarding, such as BEQ and JAL. The absence of mechanisms to indicate execution completion required the manual insertion of NOP instructions, making the approach timingsensitive and unstable in complex scenarios. Although initially promising due to its ability to perform targeted validations with good granularity, the approach proved unreliable in tests involving execution dynamics, such as branches, stalls, or paths with forwarding, compromising its effectiveness in realistic environments.

Black-Box Verification

The method proved simple and robust for basic programs. Since it did not rely on the internal timing of the DUT, it avoided synchronization issues. However, its low observability hindered fault diagnosis, requiring artificial mechanisms to identify the end of program execution. While advantageous in eliminating timing-related concerns, this approach is not suitable for validating temporal aspects of the design – such as the number of cycles required to execute a program or the behavior under conditions like forwarding and stalls. In such cases, complementary verification methods are essential.

Cycle-by-Cycle Verification

The most fine-grained approach provided the highest functional coverage among the methods, enabling the detection of subtle mismatches. However, it was extremely sensitive

to desynchronization between the DUT and the model, requiring strict alignment of signals such as clock and reset. The complexity of the reference model and the comparison infrastructure increased significantly, including the need to simulate pipelines, stalls, forwarding, and other timing effects. This approach proved viable only in scenarios where synchronization could be maintained with precision.

Comparative Analysis of Verification Methods

To summarize the advantages and limitations of each verification approach, Table 1 presents a comparative analysis based on key evaluation criteria. The analysis shows that no single method is optimal for all verification needs. Modular verification is most effective during the early stages of development, while black-box techniques offer the highest reusability. For comprehensive validation, cycle-by-cycle verification provides the most detailed insights, despite its implementation complexity. An effective verification strategy should combine these methods according to the specific requirements and phase of the project.

Discussion

This section examines the key considerations and trade-offs involved in the verification of digital systems. Different strategies offer varying levels of coverage, complexity, and maintainability. By analyzing specific aspects such as levels of abstraction and regression testing, we aim to highlight practical insights and common challenges encountered during functional verification.

The Core Trade-off: Abstraction, Coverage, and Complexity

The results reveal a clear trade-off between coverage and complexity, which is directly influenced by the chosen level of abstraction. Low-level techniques, such as cycle-by-cycle verification, provide detailed visibility but are sensitive to small design changes and can be difficult to maintain. In contrast, block-level tests and high-level methods like black-box verification are simpler and more robust to internal modifications, but they offer limited coverage, making it more challenging to locate and diagnose functional failures. Validating modules such as ALUs or decoders in isolation enables early bug detection and serves as a foundation for more complex validations. Ultimately, the appropriate level of abstraction depends on the project phase, available resources, and the types of errors being targeted. Hybrid strategies that combine multiple levels often provide the best balance between verification effort and coverage.

Functional Verification Is Not Complete Verification

Even when a design passes directed tests and incorporates techniques such as random testing, assertions, and coverage analysis, it may still contain latent bugs. Functional verification, although comprehensive in scope, is inherently

Table 1. Comparison of verification methods.

Criteria	Modular	Transactional	Black-Box	Cycle
Implementation Complexity	Low	Medium	Low	High
Level of Debugging	High	Medium	Low	Medium
Timing Sensibility	None	High	None	Very High
Reusability	Low	Medium	High	Low
Reference Model	Simple	Medium	Simple	Complex

limited by the scenarios it can realistically cover. Rare corner cases or complex interactions may go undetected. Therefore, achieving complete verification requires combining multiple strategies – including formal methods and runtime validation – to minimize the risk of undetected failures and increase overall design confidence.

The Importance of Regression Testing

Design changes can introduce regressions in previously stable functionalities. A comprehensive set of regression tests is essential to ensure system integrity over time. Automation and continuous integration are key to detecting new bugs efficiently, as demonstrated by [24], where cron jobs and Bash scripts were used to run regressions automatically.

Conclusion

This study presented a comparative analysis of four functional verification strategies for a basic RISC-V core, demonstrating that no single approach is universally optimal. Instead, effective verification relies on a multi-layered strategy that combines high-level functional tests with lowlevel, cycle-accurate debugging, with the choice of technique involving a clear trade-off between abstraction, visibility, and complexity.

For security-critical applications such as CV-QKD DSP and post-processing, this rigorous, multifaceted verification is a fundamental requirement to ensure reliability. The insights and guidelines derived from this work provide a practical foundation for selecting and integrating verification strategies in the development of more complex, dedicated RISC-V ASIPs and SoCs, where the intelligent combination of these complementary techniques will be critical to success.

Acknowledgement

This work was funded by the HW DSP project: Development and Prototyping of a Multicore SoC with Dedicated Accelerators and a DSP RISC-V Processor with Dedicated Instructions for CV-QKD Applications on FPGA, supported by QuIIN – Quantum Industrial Innovation, the EMBRAPII CIMATEC Competence Center in Quantum Technologies, with financial resources from the MCTI IoT/Industry 4.0 Priority Program (PPI), through Cooperation Agreement No. 053/2023 established with EMBRAPII. The authors also acknowledge CAPES (Coordination for the Improvement of Higher Education Personnel) for the financial support provided under Finance Code 001.

References

1. da Silva VL, Dias MA, Ferreira Neto NA, Tacla AB. From coherent communications to quantum security: modern techniques in CV-QKD. In: 2024 SBFoton International Optics and Photonics Conference (SBFoton IOPC); 2024. p. 1-5.
2. Motaharifar M, Hasani M, Kaatuzian H. A survey on continuous variable quantum key distribution for secure data transmission: toward the future of secured quantum-networks. Quantum Inf Comput. 2025;25(2):175-194.
3. Pirandola S, Andersen UL, Banchi L, Berta M, Bunandar D, Colbeck R, et al. Advances in quantum cryptography. Adv Opt Photonics. 2020;12(4):1012-1236.
4. Yang S, Lu Z, Li Y. High-speed post-processing in continuous-variable quantum key distribution based on FPGA implementation. J Lightwave Technol. 2020;38(15):3935-3941.
5. Luo Y, Cheng X, Mao H, Li Q. An overview of postprocessing in quantum key distribution. Mathematics. 2024;12(14).
6. Galvão LQ, Sousa DJG, Dias MA, Ferreira Neto NA. Neural network for excess noise estimation in continuous-variable quantum key distribution under composable finite-size security. 2025.
7. RISC-V Foundation. RISC-V Foundation website [Internet]. [cited 2025 Jul]. Available from: https://riscv.org
8. Cui E, Li T, Wei Q. RISC-V instruction set architecture extensions: a survey. IEEE Access. 2023;11:24696-24711.
9. Hepola K, Multanen J, Jääskeläinen P. OpenASIP 2.0: co-design toolset for RISC-V application-specific instruction-set processors. In: 2022 IEEE 33rd International Conference on Application-Specific Systems, Architectures and Processors (ASAP); 2022. p. 161-165.
10. Lee D, Kohlbrenner D, Shinde S, Asanović K, Song D. Keystone: an open framework for architecting trusted execution environments. In: Proceedings of the Fifteenth

European Conference on Computer Systems (EuroSys '20); 2020.
11. Tain C, Patil S, Al-Asaad H. Survey of verification of RISC-V processors. J Electron Test. 2025;41:111-138.
12. Weingarten L, Datta K, Kole A, Drechsler R. Complete and efficient verification for a RISC-V processor using formal verification. In: 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE); 2024. p. 1-6.
13. Adel A, Saad D, Abd El Mawgoed M, Sharshar M, Ahmed Z, Ibrahim H, et al. Implementation and functional verification of RISC-V core for secure IoT applications. In: 2021 International Conference on Microelectronics (ICM); 2021. p. 254-257.
14. Wang J, Tan N, Zhou Y, Li T, Xia J. A UVM verification platform for RISC-V SoC from module to system level. In: 2020 IEEE 5th International Conference on Integrated Circuits and Microsystems (ICICM); 2020. p. 242-246.
15. Jimenez V, Rodriguez M, Dominguez M, Sans J, Diaz I, Valente L, et al. Functional verification of a RISC-V vector accelerator. IEEE Des Test. 2023;40(3):36-44.
16. Harshitha NB, Praveen Kumar YG, Kurian MZ. An introduction to universal verification methodology for the digital design of integrated circuits (ICs): a review. In: 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS); 2021. p. 1710-1713.
17. IEEE. IEEE Standard for Universal Verification Methodology Language Reference Manual. IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017). 2020. p. 1-458.
18. Oleksiak A, Cieślak S, Marcinek K, Pleskacz WA. Design and verification environment for RISC-V processor cores. In: 2019 MIXDES – 26th International Conference Mixed Design of Integrated Circuits and Systems; 2019. p. 206-209.
19. Schiavone PD, Sánchez E, Ruospo A, Minervini F, Zaruba F, Haugou G, et al. An open-source verification framework for open-source cores: a RISC-V case study. In: Proceedings of the 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC); 2018. p. 43-48.
20. Adel A, Saad D, El Mawgoed MA, Sharshar M, Ahmed Z, Ibrahim H, et al. Implementation and functional verification of RISC-V core for secure IoT applications. In: 2021 International Conference on Microelectronics (ICM); 2021. p. 254-257.
21. Liu C, Xu X, Chen Z, Wang B. A universal-verification-methodology-based testbench for the coverage-driven functional verification of an instruction cache controller. Electronics. 2023;12(18).
22. Hegazy A, El-Ashry S, El-Kharashi MW, Taher M. Verification of an ARM Cortex-M3 based SoC using UVM. In: 2023 10th International Conference on Signal Processing and Integrated Networks (SPIN); 2023. p. 778-783.
23. Piccolboni L, Pravadelli G. Stimuli generation through invariant mining for black-box verification. In: 2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC); 2016. p. 1-6.
24. Molina-Robles R, Solera-Bolanos E, García-Ramírez R, Chacón-Rodríguez A, Arnaud A, Rimolo-Donadio R. A compact functional verification flow for a RISC-V 32I based core. In: 2020 IEEE 3rd Conference on PhD Research in Microelectronics and Electronics in Latin America (PRIME-LA); 2020. p. 1-4.