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

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

[1] *QuIIN – Quantum Industrial Innovation, Centro de Competência Embrapii Cimatec, SENAI CIMATEC, Salvador, BA, Brazil*

[2] *Graduate Program in Electrical and Computer Engineering, Federal University of Bahia, 40210-630, Salvador, Brazil*

[*] *Corresponding author: gustavo.sousa@fbter.org.br*

Abstract: 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.

1. Introduction

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, 4, 5, 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 post-processing tasks [8, 9, 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, 14, 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, 19, 20].

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 post-processing. The ultimate goal is to establish a robust verification methodology that ensures functional reliability in future high-performance, real-time systems where any flaw could compromise cryptographic integrity.

The structure of this paper is organized as follows: Section 2 describes the four verification approaches adopted in this study. Section 3 presents the results obtained and the main observations arising from their application. Section 4 discusses the relationship between the level of abstraction and verification visibility, as well as the relevance of complementary strategies such as regression testing. Finally, Section 5 summarizes the conclusions.

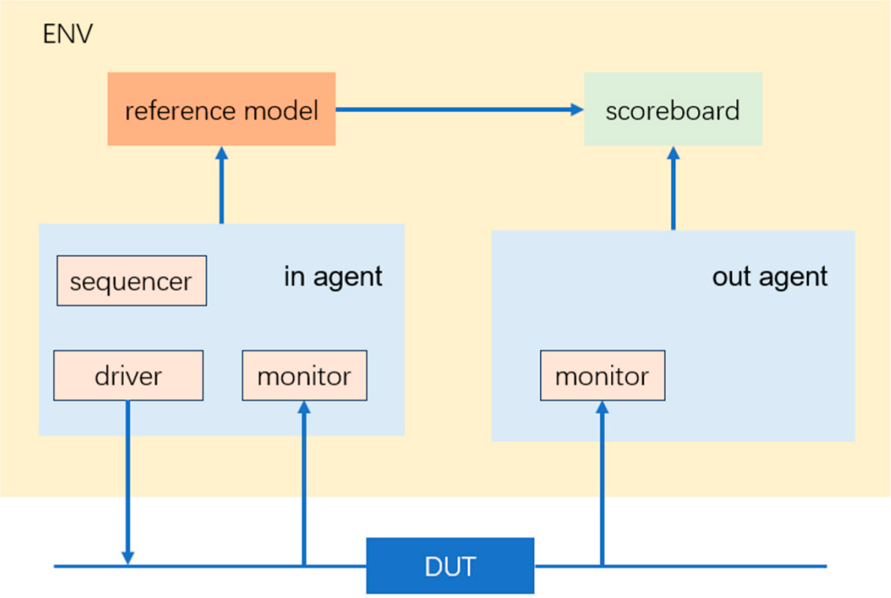
2. Methodology

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.

2.1. Block-Level Verification

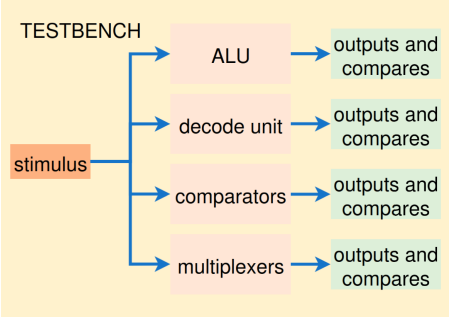
This technique involved the isolated verification of individual processor components such as the arithmetic logic unit (ALU), decode unit, comparators, and multiplexers [22]. Stimuli were applied directly to the inputs of these blocks, and the outputs were monitored to verify compliance with the expected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

Figure 1: Standard UVM environment setup. Source: [21]



with internal SystemVerilog assertions (see Figure 2).

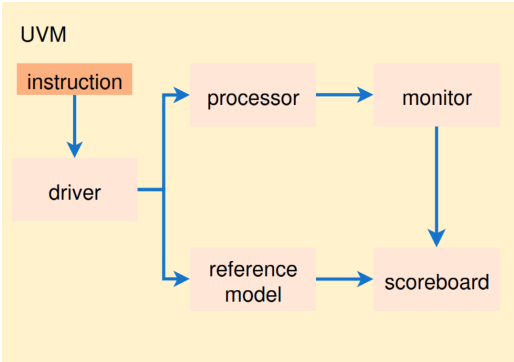
Figure 2: Block-level verification strategy.



2.2. 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 every four clock cycles. 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).

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



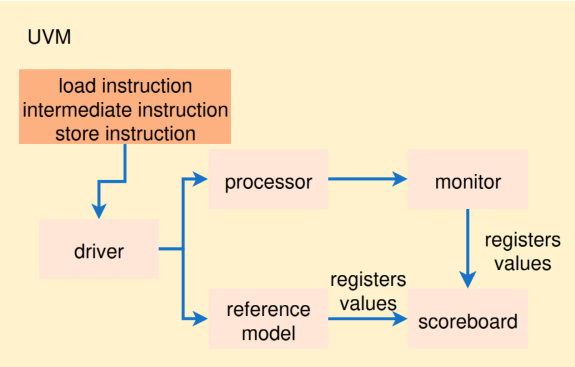
2.3. Black-Box Verification

In this strategy [23], instruction sequences were composed of `LOAD` operations, intermediate instruction set operations, and concluding `STORE` instructions. 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 in-

ternal 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 [18]. Verification at this level of abstraction is therefore highly reusable across different RISC-V-compliant 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 4: Black-box verification using memory output comparison.

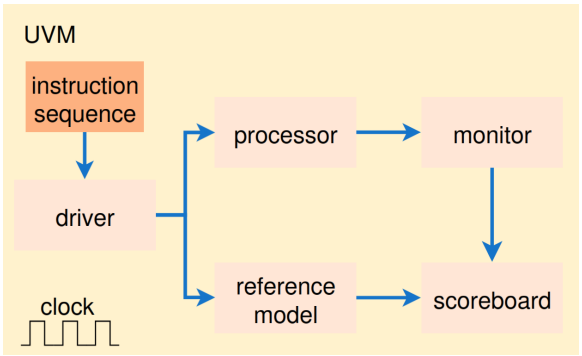


2.4. Cycle Accurate Verification (Clock Level)

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 [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 (see Figure 5).

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



3. Results and Observations

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 en-

countered during their implementation.

3.1. 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.

3.2. 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 timing-sensitive 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.

3.3. 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.

3.4. 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.

3.5. 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.

4. 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.

4.1. 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.

4.2. 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 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.

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

4.3. 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.

5. 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 low-level, 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, multi-faceted 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 fully funded by the project *HW DSP: Development and Prototyping of Multicore SoC with Dedicated Accelerators and RISC-V DSP*, supported by QuIIN – Quantum Industrial Innovation, the EMBRAPII CIMATEC Competence Center in Quantum Technologies. Financial resources were provided by the PPI IoT/Industry 4.0 program of the Brazilian Ministry of Science, Technology and Innovation (MCTI), under grant number 053/2023, in partnership with EMBRAPII.

References

[1] V. L. da Silva, M. A. Dias, N. A. F. Neto, and A. B. Tacla. From coherent communications to quantum security: Modern techniques in cv-qkd. In *2024 SBFoton International Optics and Photonics Conference (SBFoton IOPC)*, pages 1–5, 2024.

[2] M. Motaharifar, M. Hasani, and H. Kaatuzian. A survey on continuous variable quantum key distribution for secure data transmission: Toward the future of secured quantum-networks. *Quantum Information & Computation*, 25(2):175–194, 2025.

[3] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden. Advances in quantum cryptography. *Adv. Opt. Photon.*, 12(4):1012–1236, Dec 2020.

[4] S. Yang, Z. Lu, and Y. Li. High-speed post-processing in continuous-variable quantum key distribution based on fpga implementation. *Journal of Lightwave Technology*, 38(15):3935–3941, 2020.

[5] Y. Luo, X. Cheng, H. Mao, and Q. Li. An overview of postprocessing in quantum key distribution. *Mathematics*, 12(14), 2024.

[6] Lucas Q. Galvão, Davi Juvêncio G. de Sousa, Micael Andrade Dias, and Nelson Alves Ferreira Neto. 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. `https://riscv.org`. Accessed: July 2025.

[8] Enfang Cui, Tianzheng Li, and Qian Wei. Risc-v instruction set architecture extensions: A survey. *IEEE Access*, 11:24696–24711, 2023.

[9] Kari Hepola, Joonas Multanen, and Pekka Jääskeläinen. 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)*, pages 161–165, 2022.

[10] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song. Keystone: an open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, New York, NY, USA, 2020.

[11] C. Tain, S. Patil, and H. Al-Asaad. Survey of verification of risc-v processors. *Journal of Electronic Testing*, 41:111–138, 2025.

[12] L. Weingarten, K. Datta, A. Kole, and R. Drechsler. Complete and efficient verification for a risc-v processor using formal verification. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2024.

[13] Abdelrahman Adel, Dina Saad, Mahmoud Abd El Mawgoed, Mohamed Sharshar, Zyad Ahmed, Hala Ibrahim, and H. Mostafa. Implementation and functional verification of risc-v core for secure iot applications. *2021 International Conference on Microelectronics (ICM)*, pages 254–257, 2021.

[14] Jiayi Wang, Nianxiong Tan, Yangfan Zhou, Ting Li, and Junhu Xia. 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)*, pages 242–246, 2020.

[15] Victor Jimenez, Mario Rodriguez, Marc Dominguez, Josep Sans, Ivan Diaz, Luca Valente, Vito Luca Guglielmi, Josue V. Quiroga, R. Ignacio Genovese, Nehir Sonmez, Oscar Palomar, and Miquel Moreto. Functional verification of a risc-v vector accelerator. *IEEE Design & Test*, 40(3):36–44, 2023.

[16] N. B. Harshitha, Y. G. Praveen Kumar, and M. Z. Kurian. An introduction to universal verification methodology for the digital design of integrated circuits (ic's): A review. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pages 1710–1713, 2021.

[17] Ieee standard for universal verification methodology language reference manual. *IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017)*, pages 1–458, 2020.

[18] A. Oleksiak, S. Cieślak, K. Marcinek, and W. A. Pleskacz. Design and verification environment for risc-v processor cores. In *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"*, pages 206–209, 2019.

[19] P. D. Schiavone, E. Sánchez, A. Ruospo, F. Minervini, F. Zaruba, G. Haugou, and L. Benini. 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)*, pages 43–48. IEEE, 2018.

[20] A. Adel, D. Saad, M. A. El Mawgoed, M. Sharshar, Z. Ahmed, H. Ibrahim, and H. Mostafa. Implementation and functional verification of risc-v core for secure iot applications. In *2021 International Conference on Microelectronics (ICM)*, pages 254–257, 2021.

[21] C. Liu, X. Xu, Z. Chen, and B. Wang. A universal-verification-methodology-based testbench for the coverage-driven functional verification of an instruction cache controller. *Electronics*, 12(18), 2023.

[22] A. Hegazy, S. El-Ashry, M. W. El-Kharashi, and M. Taher. Verification of an arm cortex-m3 based soc using uvm. In *2023 10th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 778–783, 2023.

[23] L. Piccolboni and G. Pravadelli. Stimuli generation through invariant mining for black-box verification. In *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, 2016.

[24] R. Molina-Robles, E. Solera-Bolanos, R. García-Ramírez, A. Chacón-Rodríguez, A. Arnaud, and R. Rimolo-Donadio. 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)*, pages 1–4. IEEE, 2020.