Journal of Science and Technology – Smart Systems and Devices
Volume 36, Issue 2, May 2026, 011–018

An FPGA-Based SoC Implementation of the CRYSTALS-Dilithium Post-Quantum Digital Signature Scheme

Tuan-Anh Dang, Nhu-Quynh Luc*
Academy of Cryptography Techniques, Ha Noi, Vietnam
**Corresponding author email: quynhln@actvn.edu.vn*

Abstract

The rapid advancement of quantum computing threatens the long-term security of conventional public-key cryptosystems, motivating the development of practical post-quantum cryptographic solutions. Among the algorithms standardized by the National Institute of Standards and Technology, CRYSTALS-Dilithium has emerged as a leading lattice-based digital signature scheme due to its strong security guarantees and implementation efficiency. However, efficient realization of CRYSTALS-Dilithium on resource-constrained embedded platforms remains challenging. This article presents an FPGA-based System-on-Chip architecture for the CRYSTALS-Dilithium post-quantum digital signature scheme. The proposed design adopts a hardware–software co-design approach in which computationally intensive modules, including matrix expansion, SHAKE-256 hashing, and polynomial vector operations, are implemented in hardware to exploit FPGA parallelism while maintaining reasonable resource utilization. The architecture is implemented on a Xilinx Artix-7 (Basys-3) platform and evaluated using the NIST security level 5 parameter set at an operating frequency of 100 MHz. Experimental results show that the average key generation time is 2653.63 ms, while signing and verification require 2.695 ms and 1.105 ms, respectively. Performance analysis indicates that key generation dominates the total execution time, approximately 99.85%, primarily due to the complexity of the key generation process and the current non-optimized UART data transfer mechanism. These results demonstrate the functional feasibility of the proposed low-cost FPGA SoC architecture and provide a practical baseline for future performance and resource optimization of CRYSTALS-Dilithium accelerators in embedded post-quantum cryptographic systems.

Keywords: CRYSTALS-Dilithium, digital signature scheme, FPGA, post-quantum cryptography, SoC.

1. Introduction

The advent of large-scale quantum computing poses fundamental challenges to the long-term security of widely deployed public-key cryptosystems. Shor's algorithm demonstrates that integer factorization and discrete logarithm problems can be solved efficiently on sufficiently powerful quantum computers [1]. As a result, cryptographic schemes such as RSA and elliptic curve cryptography are considered vulnerable in the post-quantum era. In response to this emerging threat, the National Institute of Standards and Technology (NIST) has conducted a comprehensive standardization process to identify quantum-resistant cryptographic primitives suitable for future secure systems [2].

Among the digital signature candidates, CRYSTALS-Dilithium has been selected by NIST as a primary lattice-based signature scheme [3]. The security of CRYSTALS-Dilithium relies on the hardness of the Module Learning With Errors (MLWE) and Module Short Integer Solution (MSIS) problems [3]. The underlying hardness of LWE and Ring-LWE–type assumptions has been extensively studied and is widely regarded as a strong foundation for post-quantum cryptography [4]. Unlike earlier lattice signatures that depend on discrete Gaussian sampling, CRYSTALS-Dilithium employs rejection sampling and structured polynomial arithmetic, which simplifies constant-time implementations and improves hardware suitability [3]. Architectural optimizations of lattice arithmetic further indicate that CRYSTALS-Dilithium's computational kernels map efficiently onto parallel hardware platforms [5].

Despite these advantages, efficient implementation of CRYSTALS-Dilithium on resource-constrained embedded platforms remains challenging. Core procedures such as matrix expansion and SHAKE-256 hashing introduce substantial computational overhead in practical systems [3]. Benchmarking studies on constrained processors confirm that post-quantum schemes are highly sensitive to platform-level optimization strategies [6]. High-speed implementations on modern CPUs further identify polynomial and transform-based operations as dominant performance factors in CRYSTALS-Dilithium-family primitives [5]. On reconfigurable hardware, prior FPGA-oriented studies have demonstrated the feasibility of accelerating lattice-based digital signatures under optimized architectures [7]. Lightweight accelerator designs have also explored resource-aware implementations targeting cost-sensitive platforms [8]. Unified lattice cryptoprocessor approaches suggest that hardware reuse

p-ISSN 3093-3285
e-ISSN 3093-3315
https://doi.org/10.51316/jst.190.ssad.2026.36.2.2
Received: Mar 1, 2026; Revised: Mar 30;
Accepted: Apr 1, 2026; Online: Apr 15, 2026.

across primitives is possible but requires careful architectural balancing [9]. Nevertheless, the design space of low-cost FPGA-based System-on-Chip (SoC) implementations for CRYSTALS-Dilithium remains less thoroughly explored in the existing literature [10].

To address this gap, this article presents an FPGA-based SoC architecture for the CRYSTALS-Dilithium post-quantum digital signature scheme. The proposed design adopts a hardware-software co-design strategy in which computationally intensive kernels, including matrix expansion, SHAKE-256 hashing, and polynomial vector operations, are selectively implemented in hardware while higher-level orchestration is handled in software. The architecture is implemented on a Xilinx Artix-7 (Basys-3) platform and evaluated using the NIST security level 5 parameter set at an operating frequency of 100 MHz. The experimental analysis provides detailed insight into CRYSTALS-Dilithium execution on a low-cost FPGA SoC and identifies key performance bottlenecks that motivate future optimization.

The remainder of this article is organized as follows. Section 2 reviews the background and related work. Section 3 presents the proposed architecture. Section 4 reports the implementation results and performance evaluation. Finally, Section 5 concludes the article.

2. Related Work

The practical deployment of post-quantum cryptography has motivated extensive investigations into efficient implementations of lattice-based digital signature schemes across both software and hardware platforms [2]. CRYSTALS-Dilithium has received particular attention because its security relies on the Module Learning With Errors and Module Short Integer Solution problems, which are widely regarded as resistant to quantum attacks [3]. The algorithm further benefits from the avoidance of discrete Gaussian sampling, making it more amenable to constant-time and hardware-oriented realizations [11].

On the software side, several studies have explored optimized CRYSTALS-Dilithium implementations on embedded processors and general-purpose platforms. The pqm4 benchmarking framework demonstrated the feasibility of running NIST PQC candidates on resource-constrained ARM Cortex-M4 devices while highlighting performance bottlenecks in polynomial arithmetic [6]. Similarly, the Neon-NTT work showed that carefully optimized number-theoretic transform kernels can significantly accelerate CRYSTALS-Dilithium and related schemes on modern CPU architectures [5]. These results underline the importance of arithmetic kernel optimization but do not directly address FPGA-based system integration.

From the hardware perspective, a number of high-performance FPGA implementations of CRYSTALS-Dilithium have been reported in the literature. Gupta *et al.* proposed a lightweight hardware accelerator that achieves efficient CRYSTALS-Dilithium execution on reconfigurable platforms while maintaining moderate resource usage [8]. Mert *et al.* introduced a unified lattice cryptoprocessor capable of supporting both signature and key-exchange primitives and demonstrated competitive throughput on Zynq UltraScale+ devices [9]. Beckwith *et al.* presented a high-performance architecture employing aggressive parallelization and pipelining to achieve sub-millisecond latency on multiple FPGA families [7]. Land *et al.* further optimized deeply pipelined CRYSTALS-Dilithium hardware and confirmed the effectiveness of full-hardware datapaths for minimizing signing and verification latency [10]. Representative CRYSTALS-Dilithium execution times on FPGA platforms reported in the literature are presented and compared with the results achieved in this work, and are further analyzed in detail in the Results and Discussion section.

Although these fully hardware implementations achieve impressive sub-millisecond latency through deep pipelining and extensive parallelism, they typically involve considerable hardware complexity and are commonly evaluated on relatively powerful FPGA platforms [7]. Consequently, the behavior of CRYSTALS-Dilithium in low-cost FPGA-based System-on-Chip environments remains less thoroughly characterized in the open literature. In particular, system-level bottlenecks arising from hardware–software interaction and peripheral communication are often not visible in standalone accelerator studies. Motivated by these observations, this work investigates a hardware–software co-designed CRYSTALS-Dilithium architecture on a low-cost Artix-7 (Basys-3) FPGA SoC platform, aiming to provide practical insight into performance–resource trade-offs in embedded post-quantum deployments. The proposed architecture is described in detail in the following section.

3. Proposed FPGA-Based SoC Architecture

3.1. System Overview

In this work, the authors propose an FPGA-based System-on-Chip (SoC) architecture to accelerate the CRYSTALS-Dilithium post-quantum digital signature scheme on a low-cost embedded platform. The target hardware platform is the Basys-3 development board featuring a Xilinx Artix-7 FPGA operating at 100 MHz. The primary objective is to achieve a practical balance between computational acceleration and hardware resource utilization while maintaining sufficient flexibility for embedded deployment. Fig. 1 illustrates overall system flowchart of the CRYSTALS-Dilithium FPGA-Based SoC architecture that the authors propose.

Throughout this article, the term SoC refers to the proposed FPGA-based system integrating processor-level control, dedicated hardware accelerators, on-chip memory, and a UART communication interface.

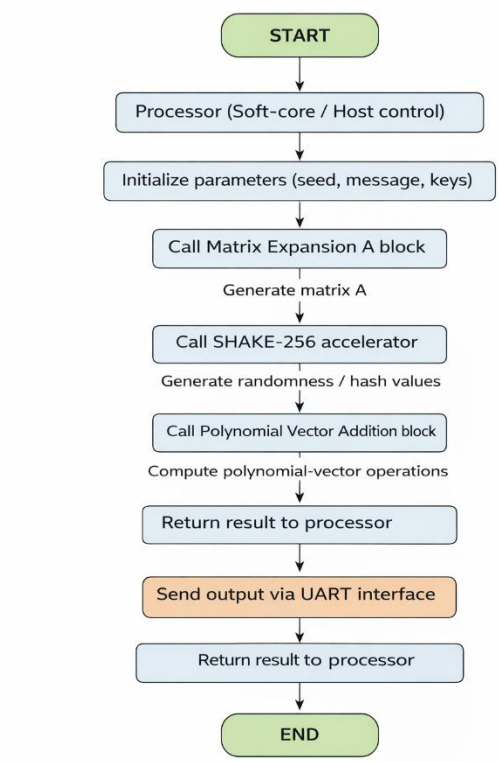

Fig. 1. Overall system flowchart of CRYSTALS-Dilithium FPGA-based SoC architecture

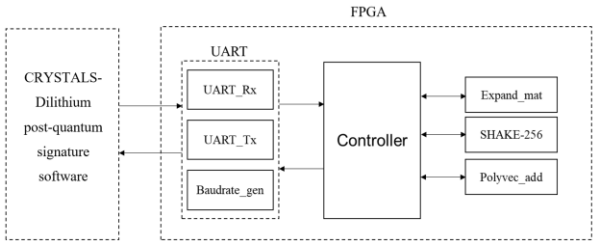

Fig. 2. Block diagram of the proposed design

Fig. 2 illustrates the top-level architecture of the proposed system. The authors adopt a hardware-software co-design paradigm in which only the most computationally intensive kernels of the CRYSTALS-Dilithium algorithm are implemented in programmable logic, while higher-level control and protocol operations are executed in software. This partitioning significantly reduces implementation complexity compared with full-hardware realizations while still enabling meaningful acceleration.

The overall system consists of the following main components: (1) an host-side software controller for system control; (2) dedicated hardware accelerator modules for critical CRYSTALS-Dilithium kernels; (3) on-chip memory for intermediate data storage; (4) UART communication interface for external interaction and debugging.

3.2. Hardware-Software Partitioning Strategy

Efficient mapping of CRYSTALS-Dilithium onto a resource-constrained FPGA requires careful identification of computational bottlenecks. Based on algorithm analysis, the authors select three major kernels for hardware acceleration: (1) matrix expansion (ExpandA); (2) SHAKE-256 hashing; (3) polynomial vector addition. These operations dominate the arithmetic workload and exhibit strong data-level parallelism, making them suitable for FPGA implementation.

All remaining tasks-including protocol sequencing, parameter handling, and non-critical arithmetic-are retained in software. This hybrid strategy reduces logic utilization pressure on the Artix-7 device while preserving the ability to accelerate the dominant computational stages. For the NIST security level 5 parameter set used in this work, the main Dilithium parameters include polynomial degree $n = 256$ and modulus $q = 8380417$, which determine the datapath width and memory requirements of the hardware accelerators [3].

3.3. Hardware Accelerator Modules

3.3.1. Matrix expansion module (ExpandA)

The matrix expansion module generates the public matrix elements required by CRYSTALS-Dilithium using the seed value ρ. The process relies on SHAKE-128-based pseudorandom generation followed by rejection sampling to obtain coefficients in the finite field modulo q. Fig. 3a illustrates the signal diagram of the matrix expansion module.

The matrix expansion module interfaces with the system through a set of control, input, and output signals. The block is synchronized by the ap_clk signal and reset via ap_rst, while the ap_ctrl interface manages the start, ready, done, and idle handshaking states. The primary input to the module is the 32-byte seed ρ, which is sequentially provided through the rho_q0[7:0] data bus and accessed using the rho_address0[4:0] address signal under the control of rho_ce0. After processing, the generated matrix coefficients are written to the external mat memory through the mat_d0[31:0] output data bus, with mat_address0[12:0] specifying the write location and mat_ce0 and mat_we0 controlling memory enable and write operations, respectively. This interface structure ensures correct data flow and synchronization between the matrix expansion accelerator and the overall SoC system. In the proposed design, the authors implement ExpandA as an FSM-controlled hardware block. Candidate coefficients are extracted from the SHAKE output stream and accepted only if $a_i < q$, where $q = 8380417$. Otherwise, the sample is discarded and regeneration continues. This rejection-sampling mechanism ensures correctness while maintaining hardware efficiency. Internal buffering and pipelined control are employed to improve

throughput and reduce memory access stalls. The module interfaces with the processor via memory-mapped registers for flexible invocation.

3.3.2. SHAKE-256 dardware core

Hashing based on SHAKE-256 constitutes a significant computational component of CRYSTALS-Dilithium. To accelerate these operations, the authors implement a dedicated Keccak-based SHAKE-256 core in programmable logic. The core adopts an iterative permutation architecture to balance area and performance on the Artix-7 device. The permutation operates on a 1600-bit internal state and executes the standard Keccak round function sequentially. An iterative (non-unrolled) permutation is chosen to reduce LUT and register usage on the target FPGA while maintaining sufficient throughput for the 100 MHz operating point. Fig. 3b illustrates the signal diagram of the SHAKE-256 hardware module.

The SHAKE-256 block implements an extendable-output function (XOF) based on the Keccak sponge construction and is used throughout the CRYSTALS-Dilithium workflow. The module operates under the ap_ctrl handshaking interface and is synchronized by ap_clk and ap_rst signals. Input data are provided through the input_r[7:0] port with the corresponding input length inlen[63:0], while the required output length is specified via outlen[63:0]. When the input size exceeds the SHAKE-256 rate (136 bytes), the message is processed in multiple absorb blocks, and similarly, multiple Keccak-f[1600] permutations are invoked during the squeeze phase when the requested output length is larger than one rate block. Internally, the operation follows the standard sponge procedure. During the absorb phase, the Keccak state is initialized and the input blocks are XORed into the state before applying the Keccak-f[1600] permutation consisting of the Theta, Rho, Pi, Chi, and Iota steps. After padding with the domain separator, the module enters the squeeze phase, where output bytes are iteratively extracted and delivered through output_r[7:0], with validity indicated by output_r_ap_vld. The SHAKE-256 core is utilized in key generation, signing, and verification, serving roles in secret expansion, matrix generation, and message hashing within the CRYSTALS-Dilithium scheme. The hardware SHAKE module supports streaming input from the processor and produces extendable-output results required during signing and verification. Offloading SHAKE-256 to hardware significantly reduces the hashing burden on the software layer.

3.3.3. Polynomial vector addition hardware core

Polynomial vector addition is another frequently executed operation in CRYSTALS-Dilithium. The proposed accelerator performs coefficient-wise vector addition using parallel arithmetic units and shared memory buffers. The datapath width is conFig. d to support coefficients modulo $q = 8380417$, ensuring compatibility with the CRYSTALS-Dilithium parameter set while avoiding excessive resource usage. The unit communicates with the processor through memory-mapped control registers, enabling flexible scheduling within the signature generation and verification procedures. Fig. 3c illustrates the signal diagram of the polynomial vector addition hardware module. The polynomial vector addition module operates under the standard ap_ctrl handshake protocol, including the control signals ap_ctrl_0_start, ap_ctrl_0_done, and ap_ctrl_0_idle/ap_ctrl_0_ready. When the system is reset (ap_rst_0 = '1'), the module enters the idle state with ap_ctrl_0_idle = '1', ap_ctrl_0_ready = '1', and ap_ctrl_0_done = '0', indicating readiness for a new transaction. A new processing session is initiated when the external controller asserts ap_ctrl_0_start for one clock cycle, after which the module deasserts the idle and ready signals and begins computation.

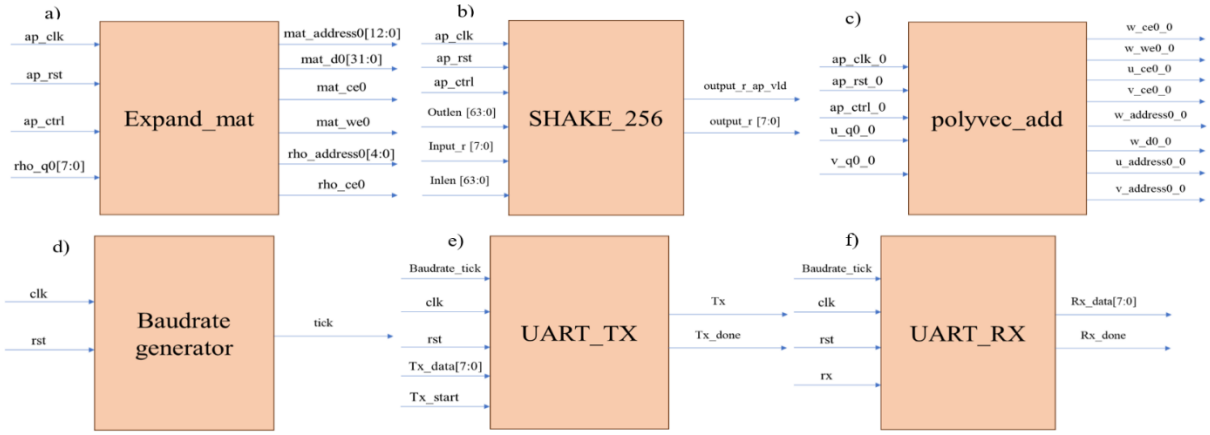

Fig. 3. Signal diagram of CRYSTALS-Dilithium: a) matrix expansion module; b) SHAKE-256 hardware module; c) polynomial vector addition hardware module; d) baud rate generator module; e) UART transmitter module; f) UART receiver module

During execution, the module sequentially generates read addresses for the input memories u and v via u_address0_0 and v_address0_0, while enabling memory access using u_ce0_0 and v_ce0_0. The corresponding 32-bit coefficients are supplied through u_q0_0 and v_q0_0, and the internal datapath performs parallel coefficient-wise addition. The results are written to the output memory w using the interface signals w_address0_0, w_d0_0, w_ce0_0, and w_we0_0. After all elements are processed, the module asserts ap_ctrl_0_done for one clock cycle and returns to the ready state for the next operation. Functionally, the block performs pure linear addition between corresponding polynomial coefficients of the two input vectors to produce the resulting polynomial vector.

All hardware accelerators are integrated through a centralized controller and memory-mapped interconnect. The embedded processor orchestrates the CRYSTALS-Dilithium workflow and invokes hardware kernels when acceleration is required. UART is employed as the primary external communication interface on the Basys-3 platform. While this approach simplifies system bring-up, UART-based transfers may introduce non-negligible latency for large data exchanges. The quantitative impact of this communication overhead is analyzed in Section 4.

3.3.4. Other modules

Fig. 3d, Fig. 3e, Fig. 3f respectively depict the baud rate generator, UART transmitter, and UART receiver modules. The baudrate generator provides precise timing pulses required for reliable UART serial communication. The module operates using the system clock clk and an asynchronous reset rst. When reset is asserted, the internal counter is cleared to zero and the tick output is forced low, ensuring a deterministic startup state. After reset is released, the counter increments on each rising edge of the system clock. When the counter reaches the threshold value $\mathrm{BAUD\_PERIOD} - 1$, where $\mathrm{BAUD\_PERIOD} = \frac{\mathrm{CLOCK\_FREQ}}{\mathrm{BAUD\_RATE}}$, the module generates a single-cycle tick pulse and resets the counter. Otherwise, the counter continues incrementing while tick remains low. For example, with a 100 MHz system clock and a target baud rate of 9600 bps, the baud period is approximately 10,416 clock cycles. These periodic tick pulses serve as the timing reference for UART transmission and reception, ensuring accurate bit-level synchronization.

The UART_tx module implements serial data transmission using a finite state machine (FSM) with four states: IDLE, START_BIT, DATA_BITS, and STOP_BIT. In the idle state, the tx line remains high while the module waits for the tx_start signal. Once asserted, the 8-bit input tx_data is loaded into a shift register and transmission begins with the start bit. During the DATA_BITS state, each bit is serially transmitted from LSB to MSB on every baudrate_tick pulse. After all eight bits are sent, the module enters the STOP_BIT state, where the line is driven high and tx_done is asserted to indicate completion. The FSM then returns to the idle state, ready for the next transmission. The entire process is precisely synchronized by the baudrate tick to ensure accurate UART timing.

The UART_rx module implements serial data reception using a FSM with four states: IDLE, START_BIT, DATA_BITS, and STOP_BIT. In the idle state, the module monitors the rx line, which normally remains high. When a low level is detected, indicating the start bit, the FSM transitions to START_BIT and waits for a baudrate_tick to confirm synchronization. The receiver then enters the DATA_BITS state, where incoming bits are sampled from rx and shifted into an internal register on each baudrate tick until all eight bits are collected. Afterward, the module proceeds to STOP_BIT to verify the stop condition and asserts rx_done to indicate successful reception, with the recovered byte available at rx_data. The FSM finally returns to the idle state, ready for the next frame. The entire reception process is synchronized by the baudrate tick to ensure correct UART timing.

The proposed architecture prioritizes practical deployability on entry-level FPGA hardware rather than maximum raw throughput. The design choices made by the authors aim to balance performance acceleration and hardware cost on low-end FPGA platforms. The prototype is intended primarily as a low-cost embedded demonstrator for feasibility study and educational or experimental evaluation, rather than as a production-grade high-throughput signing engine. On-chip memory was selected in this prototype to simplify the datapath and avoid additional external-memory interface complexity on the Basys-3 platform. This choice is prototype-driven and should not be interpreted as a universal preference over off-chip memory in all Dilithium implementations. Preliminary profiling indicates that key generation is likely to remain the dominant execution stage in CRYSTALS-Dilithium implementations on resource-constrained devices. This observation suggests that future optimization should focus on both deeper hardware acceleration and improved data movement efficiency. Based on this architecture, the next section presents the implementation results and performance evaluation.

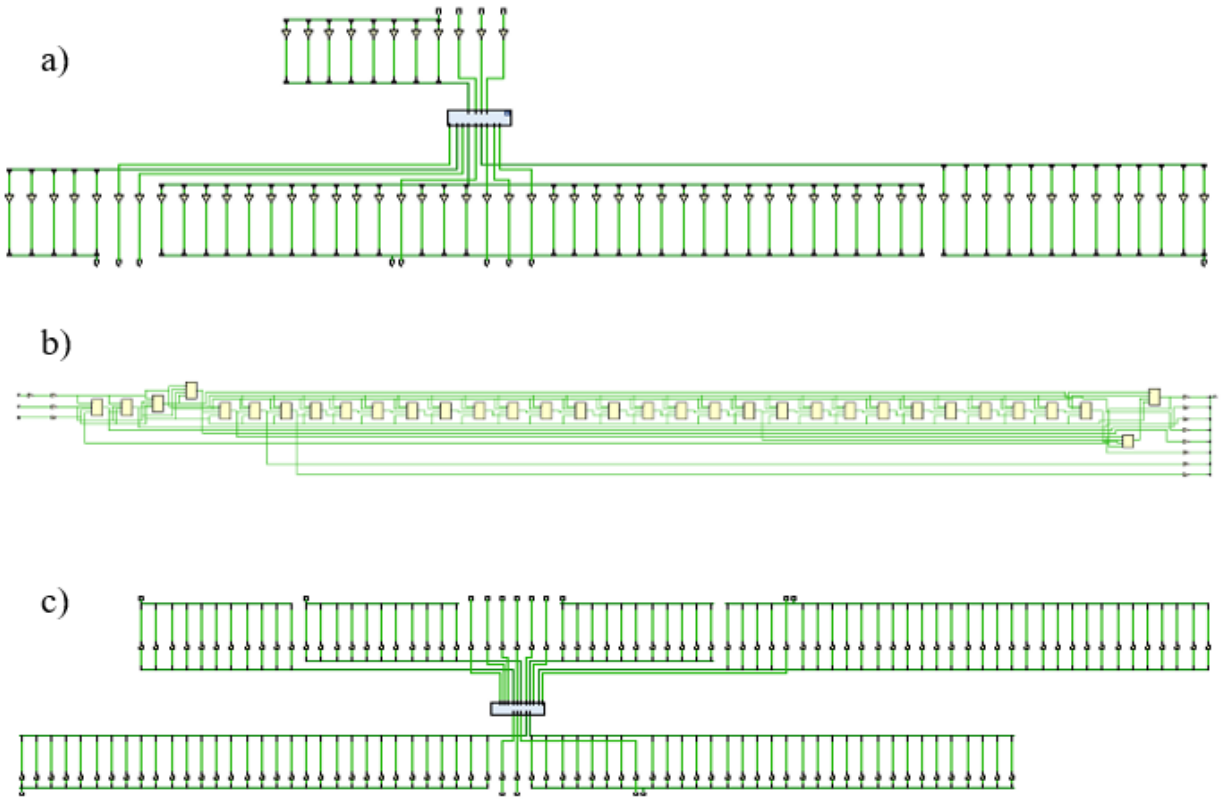

Fig. 4. RTL views of the proposed hardware accelerators: (a) ExpandA; (b) SHAKE-256; (c) polynomial vector addition

4. Results and Discussion

The proposed FPGA-based SoC was implemented on the Basys-3 development board featuring a Xilinx Artix-7 FPGA and evaluated at an operating frequency of 100 MHz. The experiments target the CRYSTALS-Dilithium parameter set corresponding to NIST security level 5.

Table 1. Performance comparison with existing FPGA-based CRYSTALS-Dilithium implementations

Platform	Frequency (MHz)	KeyGen (ms)	Sign (ms)	Verify (ms)
Artix-7 Dilithium-II [7]	163	0.39	0.70	0.42
Artix-7 Dilithium-V [7]	116	0.12	0.21	0.13
Kintex-7 [7]	173	0.08	0.14	0.08
Zynq UltraScale+[8]	391	0.16	0.29	0.17
Zynq UltraScale+[9]	200	0.19	0.35	0.23
Virtex UltraScale+ [7]	256	0.05	0.10	0.06
Artix-7 Dilithium-V (this work)	100	2653.63	2.695	1.105

Performance evaluation focuses on the three principal operations of the scheme: (1) key generation, (2) signature generation, and (3) signature verification.

Table 1 summarizes the execution time of the implemented prototype and compares it with representative FPGA-based CRYSTALS-Dilithium implementations reported in the literature. As shown in Table 1, the proposed design achieves millisecond-level latency for both signing and verification while operating at a moderate clock frequency. However, key generation remains significantly slower than fully optimized FPGA implementations. From Table 1, it is evident that the signing and verification stages of the proposed architecture are reasonably efficient for an entry-level FPGA platform. This comparison should not be interpreted as a like-for-like comparison with fully optimized RTL accelerators. Prior works largely report deeply pipelined hardware cores on more aggressive architectures, whereas the present results correspond to a prototype-level end-to-end co-design measurement that includes communication and software-control overhead. The significantly higher latency of key generation is mainly attributed to the computational complexity of this phase and the current reliance on UART-based data transfer for large intermediate data. This observation is consistent with the design trade-offs

discussed in Section III, where the authors prioritize low-cost deployment and selective hardware acceleration. Beyond timing performance, hardware utilization is an important factor for embedded deployment. Table 2 summarizes the post-implementation resource consumption on the target Artix-7 device. The results indicate moderate utilization of LUTs, flip-flops, and BRAM resources, leaving sufficient headroom for further architectural enhancements or additional accelerators.

Table 2. Resource utilization of the proposed FPGA-based SoC

Resource	Used	Available	Utilization
LUT	6780	20800	32.60%
FF	4034	41600	9.70%
BRAM	10	50	20.00%

To further verify the correctness of the hardware implementation, RTL analyses were generated for the major accelerator modules in the proposed design. Fig. 4 (a)–(c) present the synthesized RTL views of (a) the ExpandA module, (b) the SHAKE-256 core, and (c) the polynomial vector addition unit, respectively. These RTL results confirm proper structural integration of the accelerators within the FPGA fabric.

In addition to structural validation, functional verification was performed through post-simulation analysis. Fig. 5 illustrates the simulation waveform of the polynomial vector addition module, confirming correct arithmetic behavior and timing consistency of the implemented datapath.

Fig. 5. Post-simulation waveform verifying the functionality of the polynomial vector addition accelerator

As observed in the control signals of the matrix expansion block operate correctly, where the start, ready, and done handshaking signals follow the expected timing behavior. The generated output data transitions from undefined states to valid values once the computation is completed, confirming the correct implementation of the Expand_mat datapath. Similarly, Fig. 5 shows the timing waveform of the SHAKE-256 module, in which the internal state and output stream evolve consistently with the input seed and control sequence. The sequential updates of the data buses and valid signals demonstrate that the hashing core functions reliably under the specified clock conditions. Together, these simulation results verify the functional correctness and proper synchronization of the proposed hardware accelerators prior to full system integration.

Fig. 6 shows the estimated power consumption of the implemented design. The total power is approximately 0.307 W, with dynamic power accounting for the majority of the consumption. This relatively low power profile indicates that the proposed SoC is suitable for low-cost and energy-constrained embedded post-quantum cryptographic applications.

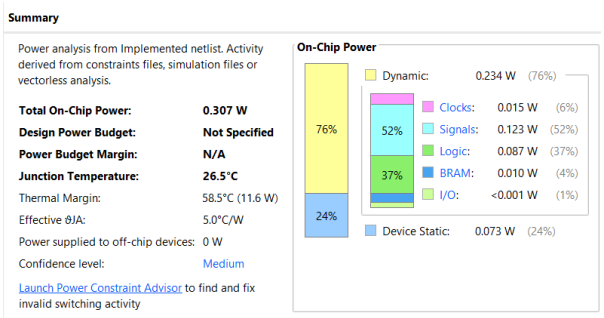

Fig. 6. Estimated power consumption of the proposed FPGA-based SoC

Fig. 7 illustrates the graphical user interface of the proposed CRYSTALS-Dilithium digital signature application implemented on the FPGA-based SoC platform. The interface enables key generation, message signing, and signature verification through a serial communication link (COM11) between the host computer and the hardware accelerator. As shown in the Fig. 7, the system successfully generates the public key (PK) and secret key (SK), and the verification dialog confirms that the produced signature is valid. This result demonstrates the correct functional integration of the hardware accelerators with the software control layer and validates the end-to-end operation of the proposed CRYSTALS-Dilithium implementation.

Fig. 7. Graphical interface of the FPGA-based CRYSTALS-Dilithium signature system showing successful key generation and signature verification

The reported end-to-end timings include hardware-kernel execution together with host-side control, UART communication, and accelerator invocation overhead. A cycle-accurate decomposition of these components is not yet available in the current prototype; therefore, the discussion of bottlenecks should be interpreted qualitatively rather than as a fine-grained profiling result.

Overall, the experimental results demonstrate that the proposed FPGA-based SoC provides a practical baseline for accelerating CRYSTALS-Dilithium on resource-constrained platforms. While the current implementation already achieves efficient signing and verification performance with moderate hardware cost, further optimization-particularly for the key-generation stage and communication subsystem-remains an important direction for future work.

5. Conclusion

This work presented an FPGA-based System-on-Chip architecture for accelerating the CRYSTALS-Dilithium post-quantum digital signature scheme on a low-cost Artix-7 (Basys-3) platform. Using a hardware-software co-design approach, the authors offloaded key computational kernels to dedicated accelerators while maintaining resource efficiency for embedded environments. Operating at 100 MHz with the NIST security level 5 parameter set, the implementation achieved an average key generation time of 2653.63 ms, while signing and verification required 2.695 ms and 1.105 ms, respectively. The results indicate that key generation is the primary performance bottleneck due to algorithmic complexity and UART communication overhead. Overall, the proposed design provides a practical baseline for embedded post-quantum cryptographic systems, and future work will focus on deeper pipelining and higher-throughput interfaces to further improve performance and scalability.

Acknowledgments

The authors acknowledge the laboratory of Department of Cryptography and the Academy of Cryptography Techniques for supporting this work.

References

[1] P. W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, pp. 124–134, Nov. 1994. https://doi.org/10.1109/SFCS.1994.365700

[2] J. Lichtinger, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, Status report on the third round of the NIST post-quantum cryptography standardization process, National Institute of Standards and Technology Interagency or Internal Report NIST IR 8413-upd1, Jul. 2022. https://doi.org/10.6028/NIST.IR.8413-upd1

[3] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, CRYSTALS-Dilithium: A lattice-based digital signature scheme, IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2018, no. 1, pp. 238–268, Feb. 2018. https://doi.org/10.13154/tches.v2018.i1.238-268

[4] D. Balbas, The hardness of learning with errors and ring-learning with errors: A survey, Cryptology ePrint Archive, Paper 2021/1358, 2021. [Online]. Available: https://eprint.iacr.org/2021/1358.pdf

[5] H. Becker, V. Hwang, M. J. Kannwischer, B.-Y. Yang and S.-Y.Yang, Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1, Cryptology ePrint Archive, Paper 2021/986, 2021. [Online]. Available: https://eprint.iacr.org/2021/986.pdf

[6] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4, in Proceedings of the third NIST Post-Quantum Crytography Standardization Conference, Virtual Conference, Jun. 7–9, 2021.

[7] L. Beckwith, D. T. Nguyen, and K. Gaj, High-performance hardware implementation of lattice-based digital signatures, Cryptology ePrint Archive, Report, 2022.

[8] N. Gupta, A. Jati, A. Chattopadhyay, and G. Jha, Lightweight hardware accelerator for post-quantum digital signature CRYSTALS-Dilithium, Cryptology ePrint Archive, Report, 2022.

[9] A. C. Mert, D. Jacquemin, A. Das, D. Matthews, S. Ghosh, and S. S. Roy, A unified cryptoprocessor for lattice-based signature and key exchange, Cryptology ePrint Archive, Report, 2021.

[10] G. Land, P. Sasdrich, and T. Guneysu, A hard crystal-implementing dilithium on reconfigurable hardware, Cryptology ePrint Archive, Report 2021.

[11] V. Lyubashevsky, Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures, in Advances in Cryptology - ASIACRYPT, Tokyo, Japan, pp. 598–616, 2009.

[12] D. J. Bernstein, J. Buchmann, and E. Dahmen, Post-Quantum Cryptography, Berlin, Germany, Springer, 2008.

[13] M. E. Sabani, I. K. Savvas, D. Poulakis, and G. Garani, Evaluation and comparison of lattice-based cryptosystems for a secure quantum computing era, pp. 1–25, 2023.

[14] A. Durmus and V. Lyubashevsky, Lattice signatures and bimodal Gaussians, in Advances in CRYPTO, Santa Barbara, CA, USA, 2013.

[15] C. Bonte, I. Iliashenko, J. Park, H. V. Pereira, and N. Smart, FINAL: Faster FHE instantiated with NTRU and LWE, Cryptology ePrint Archive, Report, 2022.

[16] P. Schwabe, B. Westerbaan, CRYSTALS-Kyber, National Institute of Standards and Technology Technical Report, 2020.

[17] J. Bradbury and B. Hess, Fast quantum-safe cryptography on IBM Z, in Proceedings of the third NIST Post-Quantum Crytography Standardization Conference, Virtual Conference, Jun. 7–9, 2021.