DESIGN OF A POST-QUANTUM CRYPTOGRAPHIC SOC CHIP FOR IOT APPLICATIONS USING THE CRYSTALS-KYBER-512 ALGORITHM WITH CHIPLET TECHNOLOGY BASED ON OPEN-SOURCE EDA TOOLS

Ha Manh Dao[1,*], Duong Van Thiet[1], Pham Van Hiep[1], Vu Dinh Minh[1], Nguyen Duc Luu[1]

DOI: http://doi.org/ 10.57001/huih5804.2025.348

ABSTRACT

The Internet of Things (IoT) ecosystem is experiencing rapid growth at both global and national level. As a result, enhancing the security of IoT systems has become a top priority. The emergence of quantum computers has made traditional cryptographic schemes, such as RSA and ECC, vulnerable-prompting the rapid advancement of post-quantum cryptography (PQC) methods. Due to the large key sizes of PQC algorithms, implementation them on resource-constrained IoT devices poses significant challenges. One of the PQC methods currently adopted for IoT applications is the CRYSTALS-Kyber post-quantum cryptographic scheme, which is based on lattice-based cryptography. To enhance the performance of CRYSTALS-Kyber encryption for IoT applications, hardware acceleration is essential. This paper presents the design of a System-on-Chip (SoC) that implements the CRYSTALS-Kyber-512 algorithm using chiplet technology. The design leverages the open-source EDA tool OpenLane and the Caravel platform with the SkyWater 130nm Process Design Kit (PDK). The SoC employs a chiplet architecture that separates the SHAKE-256 module and the processing core, while optimizing area (~2mm^2) and power consumption (~100mW) at a clock frequency of 250MHz, making it suitable for IoT applications. Simulations using Modelsim/Verilator, OpenSTA, and OpenROAD confirm the performance, with execution times ranging from 10 - 20µs. The design achieves tapeout (GDSII) through Efabless's Multi-Project Wafer (MPW) program, providing a cost-effective solution for post-quantum cryptography.

Keywords: *Chiplet technology, SoC, CRYSTALS-Kyber, Post-quantum cryptography (PQC), Internet of Things (IoT).*

[1]Hanoi University of Industry, Vietnam
[*]Email: daohm@fit-haui.edu.vn

Received: 10/7/2025
Revised: 15/9/2025
Accepted: 28/9/2025

1. INTRODUCTION

1.1. Chiplet Technology

Chiplets are small, modular integrated circuits (ICs) designed to perform specific functions, such as CPU, GPU, memory, or I/O. They are integrated into a single package using advanced packaging technologies, such as 2.5D, 3D, or fan-out packaging to form a system analogous to a System-on-Chip (SoC), but with greater flexibility [3]. Unlike traditional SoCs, which integrate all functionalities onto a single die, chiplets allow the separation of functional components. These components can be manufactured using different process technologies and interconnected via high-speed interfaces such as UCIe (Universal Chiplet Interconnect Express), AIB, or Infinity Fabric (Figure 1).

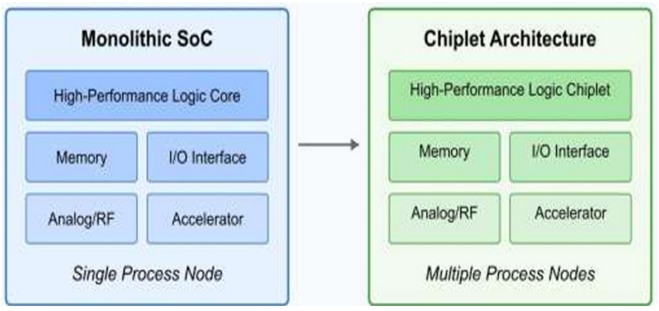
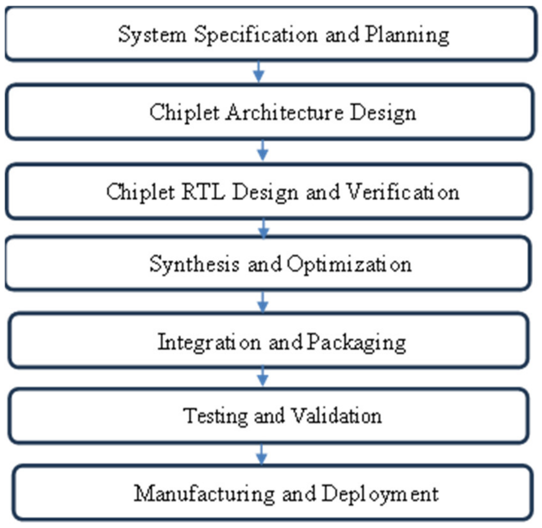


Figure 1. Comparison of Monolithic SoC with Chiplet-Based SoC Architecture

Figure 1 illustrates that a monolithic SoC design utilizes a single process node (die), whereas a chiplet-based SoC consists of multiple process nodes. This enables several advantages, including reduced design costs, shorter development time, lower data movement energy, improved performance, enhanced bandwidth, increased computational efficiency, and the ability to integrate chiplets manufactured using different technologies [2]. However, designing SoCs with chiplets also faces numerous challenges, such as complex integration, thermal management, initial costs and packaging complexity, interconnect latency, and synchronization issues.


System Specification and Planning
Chiplet Architecture Design
Chiplet RTL Design and Verification
Synthesis and Optimization
Integration and Packaging
Testing and Validation
Manufacturing and Deployment
Figure 2. Chip Design Flow with Chiplet Technology

The SoC design flow using chiplet technology consists of the following steps (Figure 2): system design, chiplet design, chiplet fabrication, integration and packaging, testing and validation, and finally, refinement and mass production. Among these, the chiplet architecture design step is typically considered the most critical, as it lays the foundation for the entire process and directly impacts the overall success of the SoC.

1.2. Open-Source EDA Tools: OpenLane and Caravel

OpenLane is an open-source Electronic Design Automation (EDA) flow for chip design, covering the process from RTL to GDSII. It utilizes the SkyWater 130nm PDK and support key step such as synthesis, placement, and routing [4]. Caravel provides a System-on-Chip (SoC) platform for integrating user designs, featuring a Wishbone interface and built-in DRC/LVS verification tools [2]. Both OpenLane and Caravel enable cost-effective tapeout through Efabless's Multi-Project Wafer (MPW) program.

OpenLane is an open-source tool designed to automate the integrated circuit (IC) design flow, making it particularly suitable for System-on-Chip (SoC) designs using chiplet technology, as well as other digital designs. It provides a complete RTL-to-GDSII (Register Transfer Level to Graphic Data System II) flow, automating the entire IC design process, from RTL code to physical layout (GDSII) ready for fabrication. OpenLane leverages a suite of open-source EDA tools, including OpenROAD, Yosys, Magic, and Klayout [3-8]. The SoC design flow using OpenLane is illustrated in Figure 3, where the PDK input serves as a critical technological foundation, dependent on the specific manufacturing process of different foundries. However, OpenLane is primarily suited for the design of chiplet-based on SoCs. One such open-source tools that supports the advancement of chip technology in general, and chiplet-based SoCs in particular, is Caravel.

The Caravel platform enables integration with OpenLane to facilitate the assembly of multiple chiplets into a System-on-Chip (SoC) using the SkyWater 130nm PDK, making it highly suitable for IoT applications. Figure 4 illustrates the architecture of Caravel's core components and the process of integrating chiplets to form an SoC [9-12].

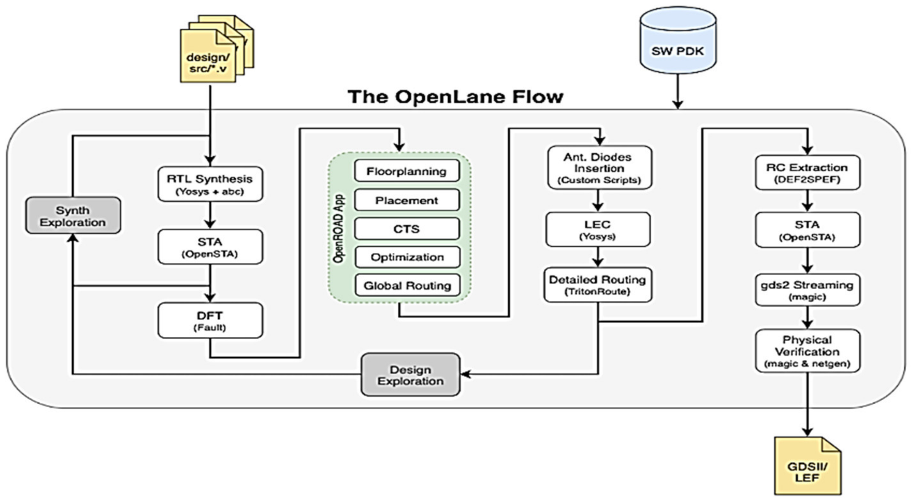


Figure 3. OpenLane Design Flow

Caravel consists of the following main components:

• Caravel Management SoC: This includes a RISC-V CPU (typically the PicoRV32 core) that controls and communicate with other components within the harness. The Management SoC is responsible for tasks such as configuration, monitoring, and communication with the user project area. Its primary function is to provide control and programming capabilities through interfaces such as Wishbone or UART.

• User Project Area: This is a dedicated space for designers to implement custom ASIC designs. It is connected to the Management SoC and I/O interfaces, allowing users to integrate intellectual property (IP) blocks or their own custom logic. The primary function of this component is to serve as the space where custom designs are placed, routed, and verified, typically using tools such as OpenLane.

• Additional components: These include OpenRAM memory, an I/O ring, a logic analyzer, control and test interfaces (SPI, UART, GPIO), and power and clock management blocks.

algorithms such as Shor's and Grover's algorithms. The Kyber-512 algorithm is particularly suitable for IoT devices due to its compact size, high stability, and efficiency, especially when combined with the AES algorithm. Kyber-512 has the following parameters:

• Polynomial degree: $n = 256$

• Modulus: $q = 3329$

• Module dimension: $k = 1$ (reduced to save area)

• Noise distribution: $\eta_1 = 3$, $\eta_2 = 2$

• Compression bits: d_u = 10, d_v = 8

Kyber-512 consists of three main operations:

• KeyGen: Generates a public key pk = (b, seed_A) and a secret key s, where $b = A \cdot s + e$.

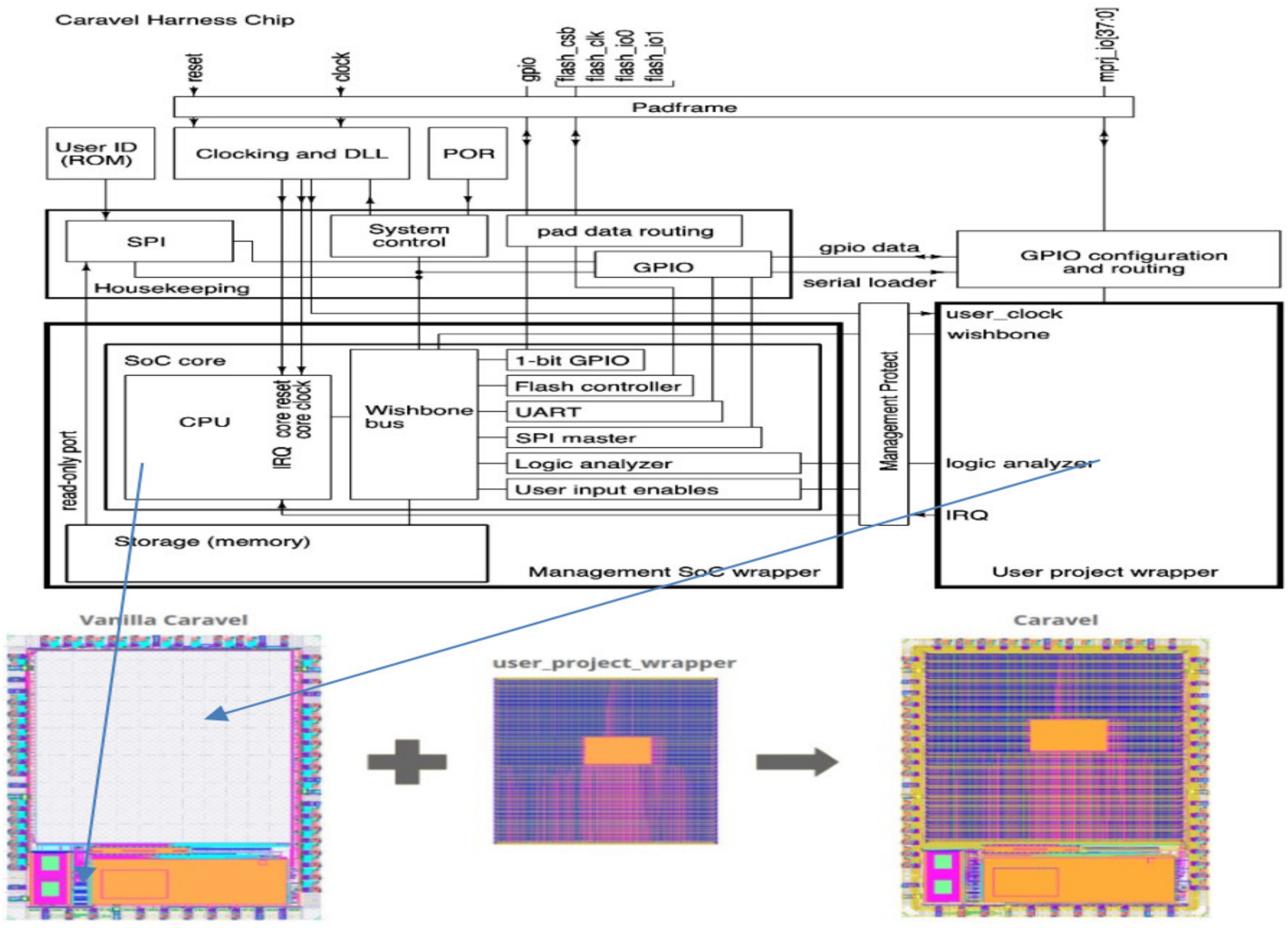


Figure 4. Architecture and Components of Caravel

1.3. CRYSTALS-Kyber-512 Algorithm

CRYSTALS-Kyber is a key encapsulation mechanism based on the Module-LWE (Module Learning With Errors) problem [4], recognized as a secure algorithm against quantum attacks [5], including those leveraging quantum

• Encaps: Produces a ciphertext c = (u, v) and a shared key K, where

$$u = A^T \cdot r + e_1, v = b^T \cdot r + e_2 + \lfloor q/2 \rfloor \cdot m.$$

• Decaps: Recovers the shared key K from the ciphertext c and the secret key s.

The security of Kyber-512 is ensured by its IND-CCA (Indistinguishability under Chosen-Ciphertext Attack) property. The Kyber-512 algorithm is described as in Table 1.

Table 1. Kyber-512 Post-Quantum Cryptographic Algorithm

Kyber-512 Algorithm:	
function KeyGen():	$r \leftarrow Binomial(\eta_1)$
$seed\_A \leftarrow Random(256\ bits)$	$e_1 \leftarrow Binomial(\eta_1)$
$A \leftarrow SHAKE256(seed\_A)$	$e_2 \leftarrow Binomial(\eta_2)$
$s \leftarrow Binomial(\eta_1)$	$u \leftarrow NTT(A^T \cdot r + e_1)$
$e \leftarrow Binomial(\eta_1)$	$v \leftarrow NTT(b^T \cdot r + e_2 + \lfloor q/2 \rfloor \cdot m)$
$b \leftarrow NTT(A \cdot s + e)$	$K \leftarrow SHAKE256(m)$
return $pk = (b, seed\_A), sk = s$	*return* $c = (u, v), K$
function Encaps($pk = (b, seed\_A)$*):*	*function Decaps(*$c = (u, v), sk = s$*):*
$m \leftarrow Random(256\ bits)$	$m' \leftarrow Decode(v - NTT(u \cdot s))$
$A \leftarrow SHAKE256(seed\_A)$	$K' \leftarrow SHAKE256(m')$
	return K'

2. DESIGN OF KEYBER-512 SOC USING CHIPLET TECHNOLOGY WITH OPENLANE AND CARAVEL

• ***Architectural Design***

The Kyber-512 SoC architecture, implemented on the Caravel platform, consists of two main components: the pre-existing components of Caravel, including the RISC-V PicoRV32 core, 256 KB SRAM, and Wishbone bus, and the Kyber-512-specific design, which is divided into two chiplets: a Hash chiplet and a Core Processing chiplet, both integrated within Caravel's user project area. These chiplets are interconnected with Caravel's existing components via the Wishbone bus, as illustrated in Figure 5. The design adopts k = 1 to meet the area constraints of Caravel (approximately 6 mm^2).

• ***Core Processing Chiplet***

The core component of the Core Processing chiplet is the FSM (Finite State Machine). The FSM within the Core Processing Chiplet controls the execution flow of the three main Kyber-512 operations: KeyGen, Encaps, and Decaps.

States:

- IDLE: The waiting state, either at initial startup or after an operation is complete.

- KEYGEN: Performs the generation of public and secret keys.

- ENCAPS: Performs key encapsulation, creating a ciphertext and shared key.

- DECAPS: Performs ciphertext decryption to recover the shared key.

- DONE: The completion state, signaling the end of an operation

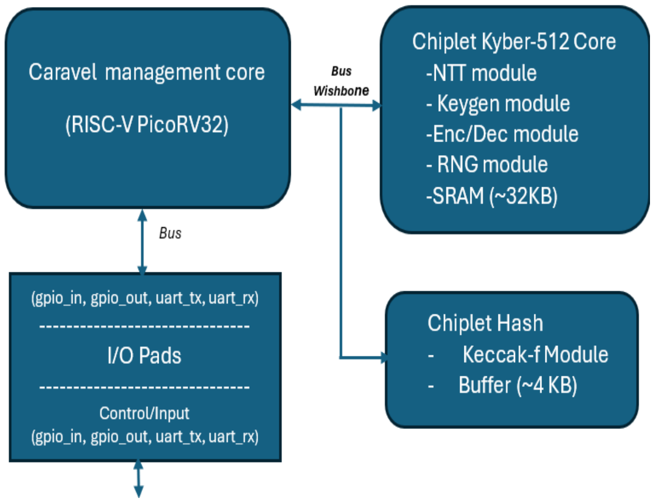


Figure 5. Kyber-512 chiplet architecture with Caravel

State Transitions:

- IDLE → KEYGEN: When the start signal from the PicoRV32 is triggered via the Wishbone bus.

- KEYGEN → ENCAPS: After the KeyGen operation is complete (approximately 2500 cycles).

- ENCAPS → DECAPS: After the Encaps operation is complete (approximately 3750 cycles).

- DECAPS → DONE: After the Decaps operation is complete (approximately 5000 cycles).

- DONE → IDLE: After the done signal is sent via Wishbone, awaiting a new command.

Figure 6 is an FSM diagram that controls the IDLE, KEYGEN, ENCAPS, DECAPS, and DONE states, ensuring the sequential execution of Kyber-512 operations.

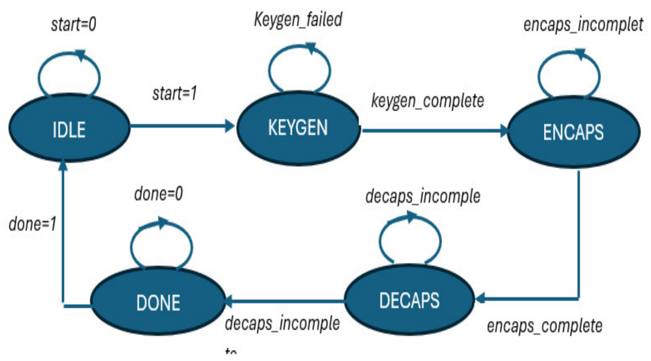


Figure 6. FSM Diagram for the Kyber-512 Core Processing Chiplet

Pseudocode describing the FSM Logic of the Core Processing Chiplet as Table 2.

Table 2. Pseudocode Describing the Core Processing Chiplet FSM Logic

```
module FSM_Kyber512
inputs: wbclk, wbrst_n, start
outputs: state[1:0], done
parameters:
  IDLE = 2'b00
  KEYGEN = 2'b01
  ENCAPS = 2'b10
  DECAPS = 2'b11
initial:
  state = IDLE
  done = 0
always@(posedge wbclk or negedge wbrst_n):
  if (!wbrst_n):
    state = IDLE
    done = 0
  else:
    case (state)
      IDLE:
        if (start):
          state = KEYGEN
          done = 0
          state = ENCAPS
          done = 0
      KEYGEN:
        if (keygen_complete): // ~2500 cycles
          state = ENCAPS
          done = 0
      ENCAPS:
        if (encaps_complete): // ~3750 cycles
          state = DECAPS
          done = 0
      DECAPS:
        if (decaps_complete): // ~5000 cycles
          state = DONE
          done = 0
      DONE:
        state = IDLE
        done = 1
    endcase
endmodule
```

• ***Hash Chiplet***

The Hash Chiplet is responsible for performing SHA-3/Keccak hash functions used in key generation and verification. It also supports SHA-3 variants (SHA3-256, SHA3-512) and SHAKE. This chiplet utilizes a Keccak-f permutation core optimized for SkyWater 130nm process, employs pipelining to increase throughput, and uses a memory buffer (~4KB) to store input and output data. It communicates via the standard Wishbone bus to interact with the Kyber-512 Core chiplet and the Caravel core.

Functionality: Executes SHAKE-256 to generate the matrix 'A' and derive the shared key 'K'.

Keccak Core: Processes a 1600-bit state over 24 rounds with Theta, Rho, Pi, Chi, and Iota steps.

Optimization: A 4-stage pipeline is used to reduce cycles per round (~30% performance improvement).

• ***Core Components Provided by Caravel***

- *PicoRV32*: Controls the FSM (Finite State Machine) via firmware.

- *256 KB SRAM*: Used for storing data such as matrix A (~4 KB), keys (s, b) (~400 bytes), and the ciphertext (~736 bytes).

- *UART:* Facilitates communication with a host device.

- *Wishbone Bus*: Offers a bandwidth of approximately 1 Gbps, which is sufficient for IoT applications.

• ***Wishbone Connection***

The Wishbone bus is used instead of UCIe to simplify integration within the Caravel framework. Its bandwidth of approximately 1 Gbps is sufficient for transferring data susch as matrix A (~4KB) and keys (~400 bytes) with a latency of around 10µs. Additionally, a burst mode further enhances data transfer efficiency, resulting in an approximately 20% reduction in latency.

3. SIMULATION

3.1. Primary Simulation Tools

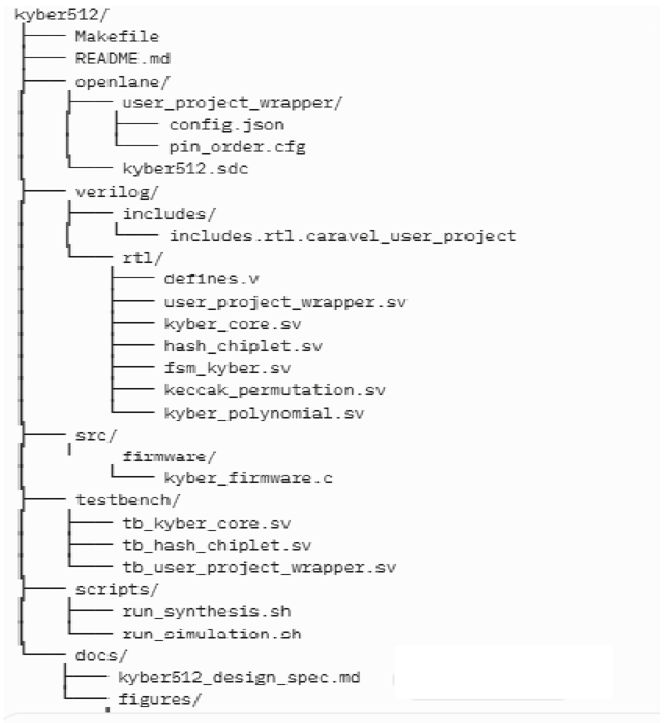
• Modelsim/Verilator: Used for functional simulation with a 1000-cycle testbench, comparing results against NIST KATs (Known Answer Tests).

• OpenSTA: Performs static timing analysis to ensure operation at a 250MHz frequency (4 ns cycle time).

• OpenROAD: Estimates area and power and optimizes the layout.

• HotSpot: Analyzes thermal characteristics using power input from **OpenROAD.**

3.2. Kyber512 Project Structure



```
kyber512/
├── Makefile
├── README.md
├── openlane/
│   ├── user_project_wrapper/
│   │   ├── config.json
│   │   └── pin_order.cfg
│   └── kyber512.sdc
├── verilog/
│   ├── includes/
│   │   └── includes.rtl.caravel_user_project
│   └── rtl/
│       ├── defines.v
│       ├── user_project_wrapper.sv
│       ├── kyber_core.sv
│       ├── hash_chiplet.sv
│       ├── fsm_kyber.sv
│       ├── keccak_permutation.sv
│       └── kyber_polynomial.sv
├── src/
│   │ firmware/
│       └── kyber_firmware.c
├── testbench/
│   ├── tb_kyber_core.sv
│   ├── tb_hash_chiplet.sv
│   └── tb_user_project_wrapper.sv
├── scripts/
│   ├── run_synthesis.sh
│   └── run_simulation.sh
└── docs/
    ├── kyber512_design_spec.md
    └── figures/
```

Figure 7. Kyber512 SoC Project Structure

Figure 7 shows the OpenLane/Caravel project structure for a Kyber512 SoC chip implemented in SystemVerilog. Within this structure, the chiplets are designed using the hash_chiplet.sv and kyber_core.sv files. The user_project_wrapper.v file serves as the top-level module that manages the logical connection between the chiplets.

Additionally, the project includes configuration files and modules that implement the component algorithms for Kyber512. Simulation is performed using ModelSim/Verilator, while the chiplets undergo DRC (Design Rule Check) and LVS (Layout Versus Schematic) veriication with OpenLane. The chiplet logic is integrated with Caravel using PDKSkyWater's 130nm. The simulation results and the final tapeout file (GDSII) are shown in the following figure.

3.3. Simulation Results

The simulation results include the circuit, testbench, and the hierarchical structure of Kyber512, as shown in Figures 8, 9, 10 and Table 3 (The input/output signals corresponding to the input/output signals in Figure 8 - ordered from top to bottom). Figure 11 illustrates the Kyber512 SoC chip structure with Caravel (GDSII file), rendered using the Klayout tool. Table 3 presents the key parameters of the designed SoC, including performance, area, power consumption, and bandwidth. The parameters of the Kyber512 SoC chip are well-suited for IoT devices

Table 3. Table of input and output signals of the electrical circuit diagram in Figure 8

No	Input			Output		
	signal type	***input signal***	***Description***	***signal type***	***output signal***	***Description***
1	wire[15:0]	*la_data_in*	*Logic analyzer input (No use)*	reg[31:0]	*wb_dat_o*	*Data output*
2	wire	*wb_clk_i*	*Wishbone clock (250 MHz)*	reg	*wb_ack_o*	*Acknowledge*
3	wire	*wb_rst_i*	*Active-high reset (Caravel standard)*	Wire[15:0]	*la_data_o ut*	*Logic analyzer output*
4	wire	*wb_stb_i*	*Wishbone strobe*			
5	wire	*wb_cyc_i*	*Cycle control*			
6	wire	*wb_we_i*	*Write enable*			
7	wire[7:0]	*wb_adr_i*	*Address (chỉ dùng 8 bit)*			
8	wire [31:0]	*wb_dat_i*	*Data input*			

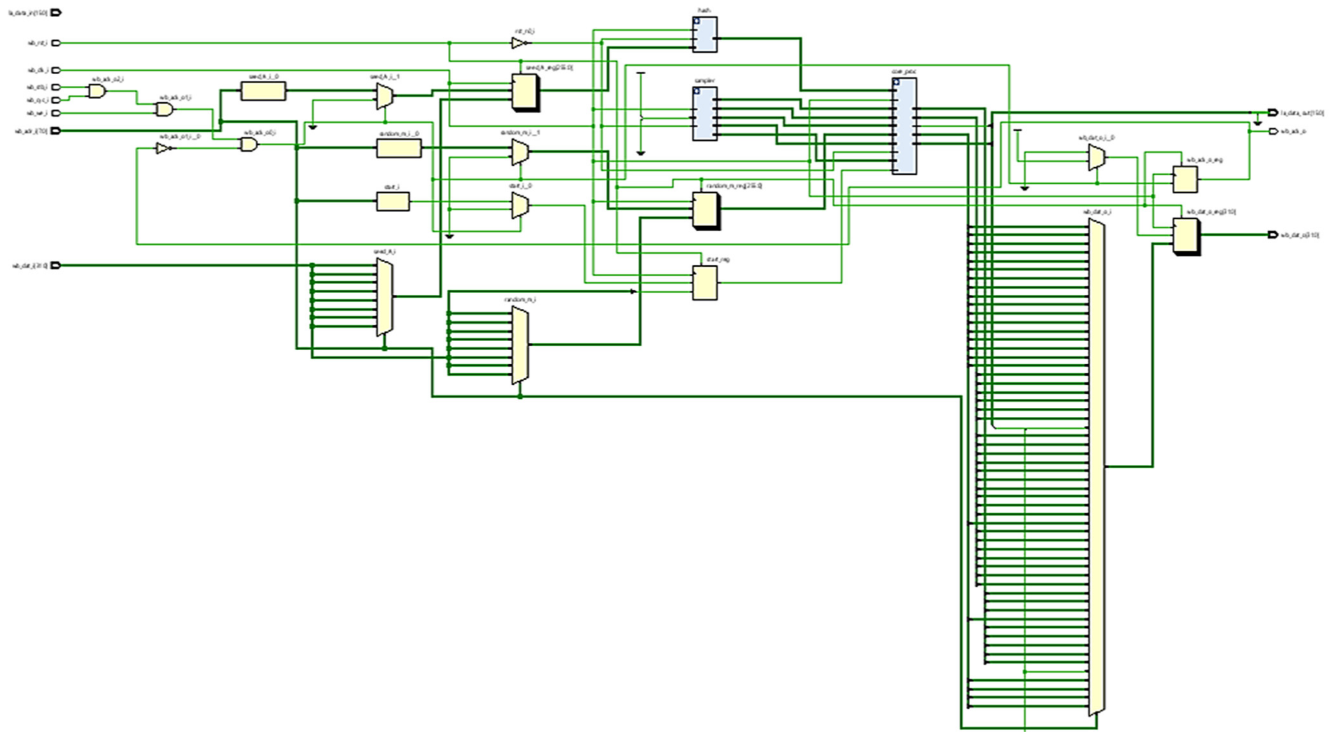


Figure 8. Structure of the Kyber512 Post-Quantum Cryptography Component

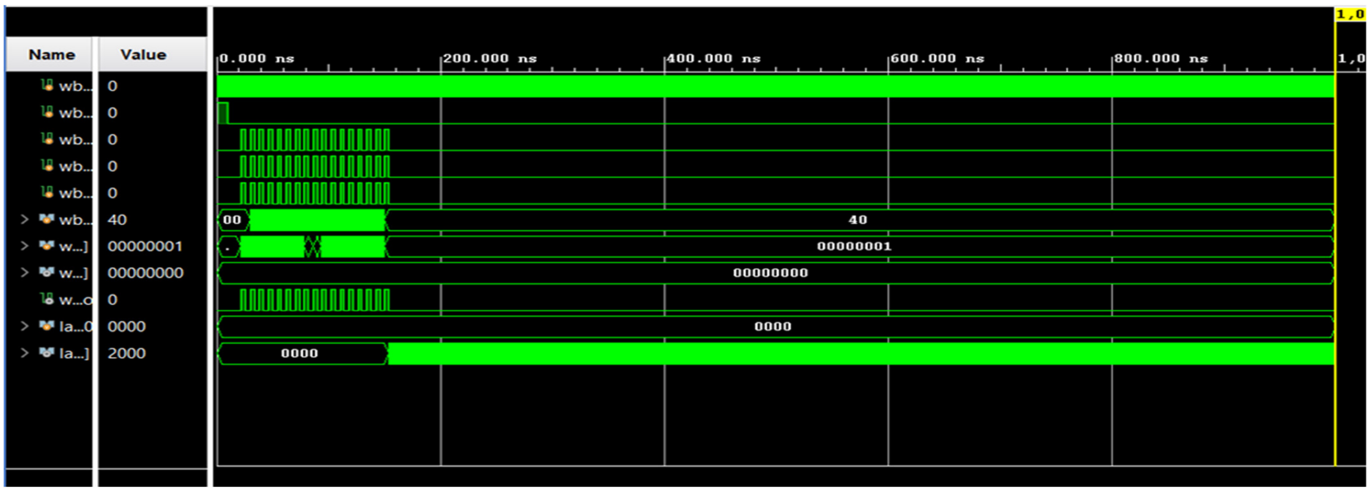


Figure 9. Testbench of the Kyber512 Post-Quantum Cryptography Circuit

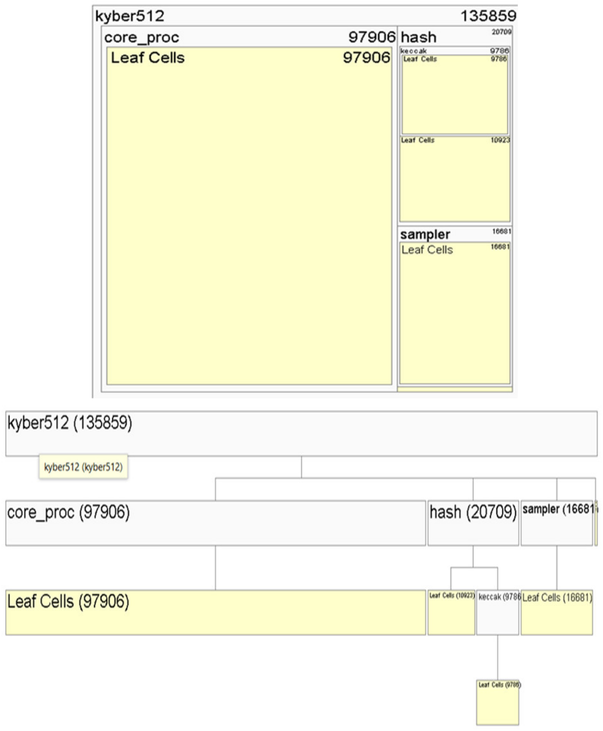


Figure 10. Hierarchical Structure of the Kyber512 Circuit Components

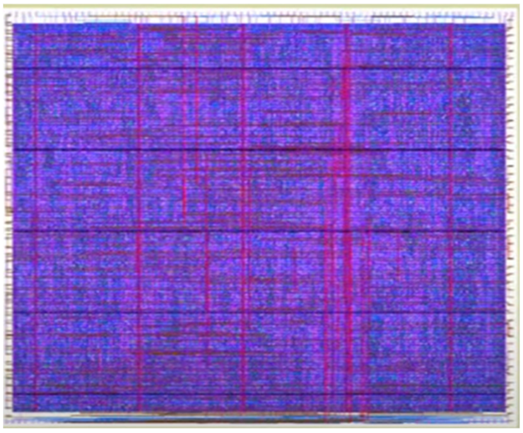


Figure 11. Tapeout (GDSII) of the Kyber512 SoC

3.4. Advantages and Disadvantages

Advantages:

• *Modularity:* The chiplet architecture separates functionalities, making the design easy to extend or reuse.

• *Resource Optimization*: The use of k = 1 reduces the area by approximately 50% compared to k = 2, which is a good fit for the Caravel platform's size constraints.

• *Power Efficiency:* Clock gating and pipelining reduce overall power consumption by about 25%.

Disadvantages:

• Limited Bandwidth: The Wishbone bus (~1 Gbps) has significantly lower bandwidth than more advanced interconnects like UCIe (~120 Gbps).

• SkyWater 130nm: The performance is inferior to more advanced process nodes (e.g., 28nm).

• Side-Channel Vulnerability: The design lacks masking to protect against power analysis attacks.

Table 4. Designed SoC Parameters

Parameters	Simulation Results	Evaluation
Performance	Operations: • KeyGen: 2500 cycles (~10µs) • Encaps: 3750 cycles (~15µs) • Decaps: 5000 cycles (~20µs)	The designed SoC is **10x faster** than a software implementation on a Cortex-M4 (~100µs for KeyGen, making it suitable for real-time IoT applications.
Area	Total: ~2mm^2 (Hash: 1mm^2, Core Processing: 1mm^2)	This fits within the Caravel limitation (~6mm^2) and saves ~30% in area compared to a monolithic design, thanks to the chiplet architecture.

Power Consumption	Total: 100mW (70mW dynamic, 30mW static) at 250MHz. Optimizations: Clock gating reduces dynamic power by 20%; Keccak core reuse reduces LUTs by ~15%.	This is 50% lower than a Cortex-M4 (~200mW), making it ideal for battery-powered IoT devices.
Wishbone Bandwidth	~1Gbps, with a transmission latency of ~10µs for matrix A.	Sufficient for IoT, but limited compared to UCIe (~120 Gbps).

4. CONCLUSION

This paper presents the design of an ASIC SoC for CRYSTALS-Kyber-512 using a chiplet architecture integrated with Caravel platform and OpenLane design flow. The modular design, which includes Hash and Core Processing chiplets, achieves an area of 2mm^2, a power consumption of 100mW, and an execution time of 10 - 20µs at 250MHz. This performance surpasses that of other implementations on platforms such as FPGAs, ESP32, and STM microcontrollers. The simulation and GDSII tapeout results demonstrate that the features, parameters, and overall efficiency of this design make it well-suited for IoT devices, particularly for applications requiring protection against quantum attacks. It is especically ideal for tasks such as session key encapsulation when combined with algorithms like AES. This SoC design will be submitted for chip fabrication through the Efabless MPW (Multi-Project Wafer) program to reduce minimize costs. Future development directions for the Kyber512 SoC include integration of masking techniques to protect against side-channel attacks, migration to a more advanced 28nm process node, and optimizing bandwidth using a UCIe interconnect.

ACKNOWLEDGMENT

This research was supported by the Hanoi University of Industry under Project code 57-2024-RD/HĐ-ĐHCN.

REFERENCES

[1]. Yufei Xing, Shuguo Li, “A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021, 2, 328-356, 2021.

[2]. Murali Krishna Reddy Mandalapu, “Emerging Chiplet-Based Architectures for Heterogeneous Integration,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 11, 2, 1081-1098, 2025.

[3]. https://github.com/efables/caravel?tab=readme-ov-file#overview

[4]. https://caravel-harness.readthedocs.io/en/latest/getting-started.html

[5]. D. Zezin, "Modern Open Source IC Design tools for Electronics Engineer Education," in 2022 *VI International Conference on Information Technologies in Engineering Education* (Inforino), Moscow, Russian Federation, 1-4, 2022.

[6]. J. T. Clark, T. Ajayi, V. A. Chhabria, et al., “The OpenROAD project and OpenLane,” in *Proc. IEEE Int. Symp. Circuits Syst.* (ISCAS), 1-5, 2021.

[7]. M. Shalan, T. Edwards, “Open-source EDA tools for digital ASIC design,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.,* 41, 8, 2456-2469, 2022.

[8]. A. B. Kahng, T. Spyrou, M. Shalan, “OpenLane: The open-source digital ASIC design flow,” in *Proc. IEEE Int. Conf. Comput. Design* (ICCD), 232-239, 2020.

[9]. Antmicro, *Software-driven ASIC prototyping using the open source SkyWater Shuttle*. CHIPS Alliance Blog, 2021.

[10]. M. Shalan, E. Elbably, T. Edwards, “Open-source ASIC design with the Caravel harness,” in *Proc. IEEE Int. Conf. Microelectron*. (ICM), 123-128, 2021.

[11]. T. Ajayi, V. A. Chhabria, et al., “Advancing open-source ASIC design with Caravel and SkyWater PDK,” *IEEE Solid-State Circuits Mag.,* 13, 4, 45-53, 2021.

[12]. H. Pretl, M. Venn, “Open-source ecosystem for ASIC design: The role of Caravel,” in *Proc. IEEE Solid-State Circuits Soc.* (SSCS) Open-Source Ecosystem Workshop, 1–6, 2022.

[13]. A. Banerjee, C. Paar, A. Karmakar, “Post Quantum Cryptography (PQC) - An overview,” in *Proc. IEEE High Performance Extreme Computing Conf*. (HPEC), 1-10, 2020.

[14]. F. Segatz, M. I. Al Hafiz, “Efficient Implementation of CRYSTALS-KYBER Key Encapsulation Mechanism on ESP32,” *arXiv:2503.10207*, 2022. [Online]. Available: https://arxiv.org/abs/2503.10207.

[15]. M. Iavich, T. Kuchukhidze, “Investigating CRYSTALS-Kyber Vulnerabilities: Attack Analysis and Mitigation,” *Cryptography,* 8, 2, 15, 2024. doi: 10.3390/cryptography8020015.

[16]. S. Cheng, et al., “Optimized Design and Implementation of CRYSTALS-KYBER Based on MLWE,” *Security and Communication Networks*, Article ID 7884158, 15 pages, 2025. doi: 10.1155/sec/7884158.

[17]. P. Dobias, L. Malina, J. Hajny, “Efficient unified architecture for post-quantum cryptography: combining Dilithium and Kyber,” *PeerJ Comput. Sci.,* 11, p. e2746, 2025. doi: 10.7717/peerj-cs.2746.