

RTL to GDSII Implementation of UART Communication Over RSA Algorithm using 90nm Technology

Dr. Mallikarjun P Y¹, Dr. Vasudeva G², Dr. Bharathi Gururaj³, Dr. Tripti R Kulkarni⁴, Hemanth M M⁵ and Abhishek H J⁶

¹Professor and HOD, Department of Electronics and Communication Engineering Dayananda Sagar Academy of Technology & Management Bengaluru, India
Email: mallikarjunpy@gmail.com

^{2,4}Assistant Professor, Department of Electronics and Communication Engineering Dayananda Sagar Academy of Technology & Management Bengaluru, India
Email: devan.vasu921@gmail.com, triptikulkarni-ece@dsatm.edu.in

³Department of ECE, KSIT, Bengaluru, India
Email: bharathigururaj@gmail.com

⁵⁻⁶Department of Electronics and Communication Engineering Dayananda Sagar Academy of Technology & Management Bengaluru
Email: hemanthmm08@gmail.com, abhishekhj12@gmail.com

Abstract— In this work, we describe the design and implementation of an 8-bit Universal Asynchronous Receiver-Transmitter (UART) configured for a baud rate of 9600. The primary goal was to create a UART architecture that minimizes both silicon area and power usage, achieving an area of 546.482 μm^2 with 66 instances and a power consumption of 17.299 μW (Post CTS).

The design flow covered all stages from RTL coding to GDSII generation, ensuring reliable performance and smooth integration into larger systems. An important feature of the design is the addition of input/output (I/O) pads, which improve signal integrity and simplify connectivity with external modules.

This is especially beneficial for ensuring robust data transfer in conditions where signal interference may occur. The proposed approach maintains a careful balance between power efficiency and area optimization, making the UART suitable for use in embedded platforms and communication systems.

The implementation was carried out using the 90nm technology library. Synthesis confirmed that the design achieves minimal area utilization while keeping power efficiency at a competitive level. Functional verification was performed through simulations in Xilinx Vivado, while synthesis was handled in Cadence Genus.

Floorplanning was carried out in Cadence Innovus, followed by Design Rule Checking (DRC) and Layout Versus Schematic (LVS). These steps ensured compliance with fabrication standards and made the design tape-out ready.

Keywords— ASCII, UART, RTL, GDSII, DRC, LVS.

I. INTRODUCTION

Universal Asynchronous Receiver-Transmitter (UART) [1] is one of the most commonly used serial interfaces in embedded systems because it is simple, requires few pins, and has strong support from the ecosystem. However, the UART protocol does not offer confidentiality or integrity guarantees [2]. In embedded systems that need security and are networked, plaintext UART links are open to eavesdropping and manipulation [3]. This vulnerability makes it necessary to add cryptographic protection near the physical interface.

This work introduces a secure UART communication subsystem that combines an RSA cryptographic accelerator with the UART data path and control logic. The RSA core is built using synthesizable Verilog HDL and is closely linked with the UART TX/RX pipeline through lightweight protocol wrappers and buffering. This setup allows for encryption and decryption at the link endpoint without requiring software intervention during data transfer. In addition to architectural design, the work progresses the hardware through the complete application-specific integrated circuit (ASIC) physical design flow, from RTL to GDSII, using a 90 nm standard-cell technology [4]. Tasks including place-and-route, clock tree synthesis, post-route optimization, and final ASIC file generation were completed using Cadence Innovus[5]. This was followed by sign-off analyses that included static timing and design rule verification within the tool flow.

Incorporating public-key cryptography into resource-limited embedded systems brings challenges related to area, power, and latency [6]. Unlike symmetric ciphers, RSA uses modular exponentiation with large numbers. This affects choices related to data path width, memory buffering, and clock frequency [7]. Combining the crypto engine with UART also necessitates careful flow-control design to hide cryptographic delays while ensuring protocol compliance and maintaining throughput under baud-rate limits [8].

In this work, we paired an RSA accelerator with a UART controller to enable end-to-end encryption and decryption at the link layer with minimal software overhead. The design was created as a complete ASIC using 90 nm standard-cell technology, and the decisions made are documented across floor planning, power planning, placement, clock tree synthesis, routing, post-route timing closure, and ASIC file generation in Cadence Innovus.

II. SYSTEM ARCHITECTURE

A. Overview of Architecture

The proposed system combines a hardware RSA cryptographic engine with a standard UART controller to secure serial communication at the physical interface. As illustrated in the block diagram, the setup consists of RSA encryption and decryption modules, UART transmitter and receiver blocks, control logic, clock and reset circuits, and temporary buffering structures [9]. In operation, plaintext data is first processed by the RSA encryption core and then serialized through the UART transmitter[10]. At the receiving end, the UART receiver deserializes the encrypted data and forwards it to the RSA decryption core, which reconstructs the original plaintext. This integrated flow ensures confidentiality of UART-based communication while fully preserving standard UART signaling and framing.

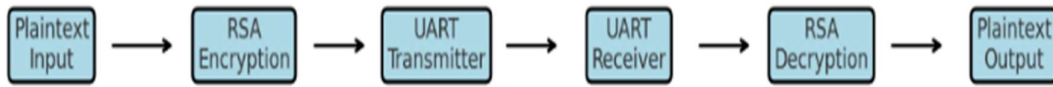


Figure 1: Block Diagram of the Proposed System

B. RSA Algorithm

RSA is an asymmetric cryptographic method based on modular exponentiation with large integer inputs[11]. The hardware setup has three main functions: key storage, encryption, and decryption. Public and private keys are stored in specific registers accessible during setup[12]. The encryption process is:

$$C = M^e \bmod n$$

where M is the plaintext, e is the public exponent, and n is the modulus. The decryption process is:

$$M = C^d \bmod n$$

where d is the private exponent. In hardware, modular exponentiation is achieved through repeated modular multiplication, using an optimized data path for handling inputs[13]. To address resource limits in embedded systems, operand word sizes and pipeline stages were chosen carefully to balance area, delay, and throughput [14].

C. UART Communication Module

UART (Universal Asynchronous Receiver/Transmitter) offers a simple serial link for embedded devices. The UART transmitter takes parallel data, adds start, stop, and parity bits, and shifts bits onto the line at a set baud rate [15]. The receiver handles clock recovery, start-bit detection, and deserialization to form bytes[16]. Error detection features like parity and framing error flags are included in this setup. Although UART is popular for its ease of use and low pin needs, it lacks security, making it prone to interception[17]. This drawback is directly addressed by integrating the RSA core into the data path.

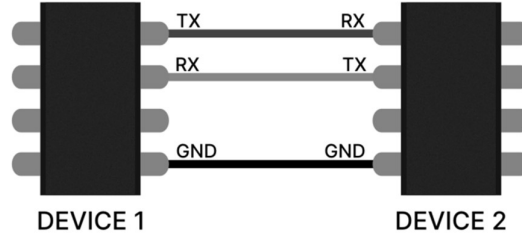


Figure 2: UART Protocol Representation

D. RSA over UART Integration

Integrating RSA with UART ensures that only encrypted data travels through the serial link[18]. In the transmitting process, plaintext input goes to the RSA encryption block first, and its ciphertext output is stored in the UART TX register. On the receiving end, the UART RX block rebuilds the ciphertext, which is then sent to the RSA decryption block to retrieve the plaintext[19]. To manage the computational delay of RSA without halting the UART pipeline, FIFO buffers are used between the RSA and UART modules[20]. The control logic aligns the encryption delay with UART transmission, ensuring compliance with data framing standards[21]. Error-handling methods, such as retransmission, are available at the UART level, while cryptographic accuracy is upheld at the RSA level[22]. This modular setup allows the design to work with standard UART interfaces while embedding strong public-key encryption directly in hardware, eliminating the need for external software-based cryptographic processes during transmission[23,24].

III. METHODOLOGY

The proposed RSA-over-UART architecture was created and implemented using a 90 nm standard-cell library with the Cadence Innovus toolchain. The design flow followed a standard RTL-to-GDSII process, tailored to manage the cryptographic and communication modules. The key steps are outlined below:

A. Design Import and Setup

The RTL description of the RSA encryption/decryption cores and UART transmitter/receiver modules was turned into a gate-level netlist. Technology libraries (.lib) and physical abstracts (.lef) were provided with design constraints (SDC). Multi-Mode Multi-Corner (MMMC) views were created to ensure timing closure under process and voltage changes.

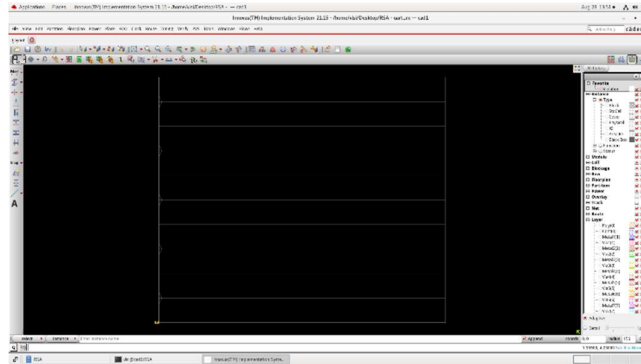


Figure 3: Design Import and Setup

This step lays the groundwork for the physical design. By importing the netlist and libraries, the design environment becomes ready for the next stages. MMC helps the system remain reliable under various operating conditions, such as voltage and temperature changes.

B. Floor planning

A rectangular core area was set up with enough utilization and aspect ratio to fit the cryptographic logic and UART blocks. Pin placement followed the data flow (plaintext input to RSA, then UART TX; and UART RX back to RSA to plaintext output), which kept interconnect overhead low.

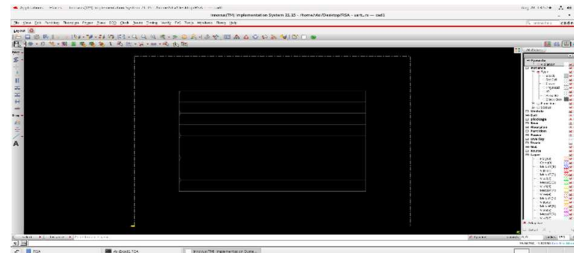


Figure 4: Floorplanning of Design

Floorplanning establishes the physical limits and ensures blocks are positioned to minimize routing complexity and delay. Proper pin alignment enhances performance and reduces wire length.

C. Power Planning

Dedicated VDD and VSS networks were built using core rings and metal stripes. The RSA core, which is computationally heavy, received higher stripe density for better uniformity in IR drop characteristics across the design.

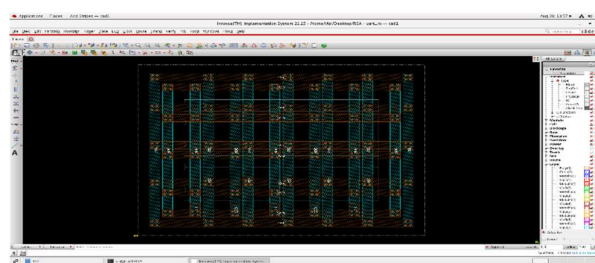


Figure 5: Power Planning with VDD and VSS Distribution

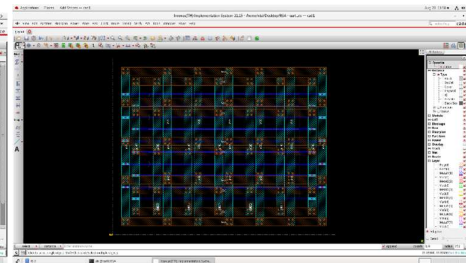


Figure 6: Special Routing for Clock and Power Nets

Power planning ensures a stable supply voltage throughout the chip. Since RSA involves significant computations, strong power distribution helps avoid voltage drops and functional errors.

D. Placement and Pre-CTS Optimization

Standard cells, well taps, and boundary cells were placed automatically. Placement was optimized to balance timing between the RSA datapath and the UART serial interface. Initial setup analysis aimed to cut critical path delays.

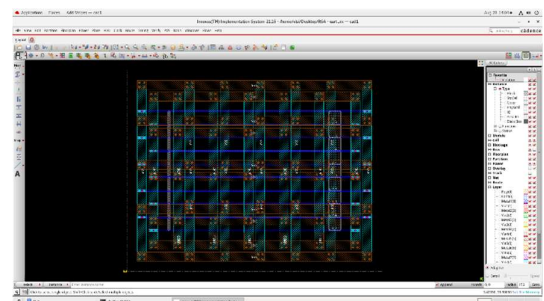


Figure 7: Addition of End Caps and Well Caps

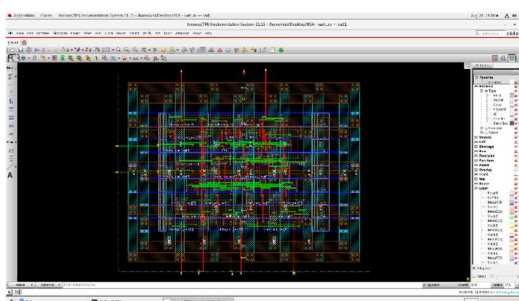


Figure 8: Standard Cell Placement of Design

Placement determines where the logic gates are physically located. An optimized placement reduces congestion and ensures critical paths, such as RSA modular exponentiation, meet timing requirements.

E. Clock Tree Synthesis (CTS)

A balanced clock distribution network was built to reduce skew and insertion delay. Clock buffers and inverters were added strategically to satisfy setup and hold requirements for RSA's modular exponentiation unit and the UART shift registers.

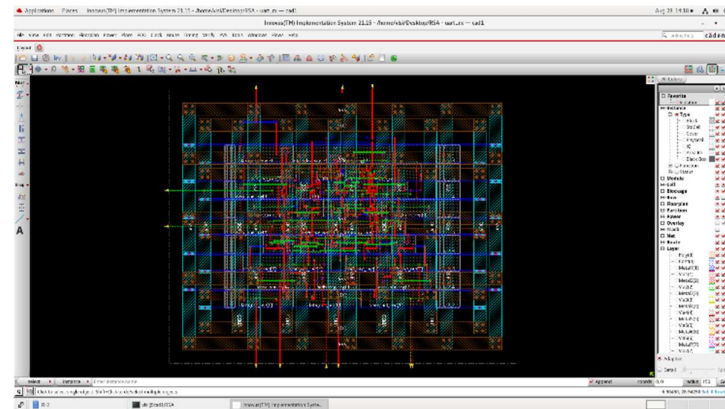


Figure 9: Routing of Interconnects

CTS makes sure the clock signal reaches all sequential elements at the same time. This prevents timing issues due to skew and keeps synchronization correct between RSA and UART functions.

F. Routing

Global and detailed routing was done using algorithms designed to handle congestion and timing. Signal integrity checks were activated to limit crosstalk on the RSA datapath and UART high-fanout nets (TX/RX). Routing connects all placed cells physically. Timing-driven routing makes sure that paths crucial for performance meet timing, while signal integrity checks stop interference that could damage encrypted data.

F. Post-Route Optimization and Verification

Static Timing Analysis (STA) confirmed that setup and hold timing were satisfied under worst- and best-case scenarios. Design Rule Check (DRC) and Layout vs. Schematic (LVS) verification ensured manufacturing compliance and logical consistency.

This step checks that the design is both timing-correct and manufacturable. DRC guarantees compliance with fabrication rules, while LVS ensures the layout matches the intended circuit functionality.

G. Back-Annotation and Functional Verification

The routed netlist with SDF timing information was re-simulated to verify that secure data transmission over UART stayed functionally correct under realistic delays.

Back-annotation confirms that the chip will work as expected after fabrication. It ensures that RSA-encrypted UART communication is not disrupted by real-world propagation delays.

H. Final GDSII Generation

The clean layout database was converted into GDSII format, completing the ASIC implementation. This file represents the final manufacturable layout of the secure RSA-over-UART system.

GDSII is the standard file for chip manufacturing. Creating this file indicates that the design is ready for tape-out and production.

I. ASCII File Generation

The verified UART_RX design was exported into an ASCII-compatible format (.enc), which represents the physical layout of the circuit. This file is tape-out ready, meaning it can be directly used for fabrication after passing all design rule, LVS, ERC, timing, area, and power checks. The .enc file ensures the optimized and verified design can seamlessly transition from the design stage to manufacturing.



Figure 10: Generated ASCII File

IV. RESULTS AND ANALYSIS

A. Physical Verification (ERC & LVS Reports)

ERC Summary No electrical rule violations were found. Checks for substrate and well contacts (psubstrate, nwell_conn, Nburied) reported zero errors.

LVS Report The design matched successfully with the schematic. No shorts, opens, or device mismatches were detected. This confirms that the layout is functionally identical to the schematic netlist, ensuring it is correct for fabrication.

B. Timing Analysis

TABLE I: TIMING ANALYSIS RESULTS AT DIFFERENT IMPLEMENTATION STAGES

Stage	Setup Slack	Hold Slack
Pre-CTS	0.000 ns	0.000 ns
Post-CTS	0.000 ns	0.000 ns
Post-Route	0.000 ns	0.000 ns

At all implementation stages (Pre-CTS, Post-CTS, Post-Route), both setup and hold slacks remain at 0.000 ns. This shows that the design meets timing requirements without violations throughout the process.

C. Area Analysis

TABLE II: AREA ANALYSIS RESULTS ACROSS DESIGN STAGES

Stage	Total Area	Instances	Change
Pre-CTS	557.835 μm^2	66	—
Post-CTS	546.482 μm^2	66	-11.353 μm^2 (-2.04%)
Post-Route	546.482 μm^2	66	negligible change

The total cell area decreased from 557.835 μm^2 (Pre-CTS) to 546.482 μm^2 (Post-CTS), showing a 2.04% improvement in area use. There is no change in the number of instances (66), confirming effective placement and optimization without cell duplication.

D. Power Analysis

Pre-CTS vs Post- CTS

TABLE III: POWER ANALYSIS – PRE-CTS VS POST-CTS

Power Component	Pre-CTS	Post-CTS	Change
Internal Power	6.656 μW	11.944 μW	+79.44%
Switching Power	1.139 μW	2.477 μW	+117.49%
Leakage Power	3.115 μW	2.878 μW	-7.61%
Total Power	10.910 μW	17.299 μW	+58.56%

An increase in dynamic power (internal plus switching) is observed after CTS due to clock tree insertion. Leakage power decreases slightly, improving static efficiency.

TABLE IV: POWER ANALYSIS – POST-CTS VS POST-ROUTE

Power Component	Post-CTS	Post-Route	Change
Internal Power	11.94408 μ W	11.94411 μ W	+0.00025%
Switching Power	2.47675 μ W	2.47609 μ W	-0.02665%
Leakage Power	2.87832 μ W	2.87832 μ W	0.00%
Total Power	17.29915 μ W	17.29852 μ W	-0.00364%

Power stays stable after routing, with small changes in the nW range. This shows that routing has not added extra capacitance or leakage problems.

The ERC and LVS reports confirm that the layout is free from connectivity and structural errors, ensuring reliability for fabrication. Timing analysis validates that setup and hold constraints are satisfied across all design stages, while a 2.04% reduction in area demonstrates optimization efficiency. Furthermore, power consumption remains stable after routing, highlighting the effectiveness of power optimization.

V. CONCLUSIONS

This work adopted a systematic approach covering all stages of the design flow—RTL development, verification, synthesis, and physical implementation—to ensure a reliable and efficient outcome. During the RTL phase, the specifications were converted into Verilog code. Functional verification played a critical role in validating the correctness of the design. The physical design stage provided insights into performance metrics and guided optimization of the existing architecture. The implemented design achieved a reasonable setup slack along with the reported power values and area values. Overall, the ASIC implementation cycle highlights the significance of every step, from RTL modeling to GDSII generation. Future enhancements will focus on optimizing the RTL through techniques such as pipelining, enabling the design to operate at higher frequencies. Additionally, further backend optimizations are planned to achieve improved performance, power efficiency, and area utilization.

REFERENCES

- [1] P. Bhatt, D. Joshi, and S. Jadhav, "RTL to GDS Implementation and Verification of UART using UVM and OpenROAD," in Proc. 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2024, pp. 0713–0720, doi:10.1109/CCWC60891.2024.10427771.
- [2] S. Kumar and A. Singh, "Universal Asynchronous Receiver and Transmitter Implementation on 40 nm Technology," Int. J. Electronics Communication and Computer Engineering, vol. 9, no. 2, pp. 54–58, Mar. 2018.
- [3] B. Venkataramana, R. D. Ramesh, and V. G. Kumar, "ASIC Implementation of Universal Asynchronous Receiver and Transmitter using 45 nm Technology," Int. J. of Engineering and Technology (IJET), vol. 5, no. 3, Jun.–Jul. 2013.
- [4] M. A. Shakil and H. Patel, "Hardware Implementation of RSA Algorithm on FPGA for Secure Communication," Microprocessors and Microsystems, vol. 88, pp. 104–116, Jan. 2022, doi:10.1016/j.micpro.2021.104116.
- [5] H. El-Sawy, A. Abdelrahman, and F. M. Salem, "Efficient RSA Cryptoprocessor for Secure Embedded Systems," IEEE Access, vol. 11, pp. 14582–14593, Feb. 2023.
- [6] Y. Lee and H. Kim, "FPGA Based Low-Power UART Design with Enhanced Security," in Proc. 2023 IEEE Int. Symp. on Circuits and Systems (ISCAS), Monterey, CA, USA, 2023, pp. 1–5.
- [7] S. Banerjee and K. R. Shyam, "ASIC Implementation of Cryptographic Algorithms for IoT Devices," in Proc. 2021 Design Automation Conf. (DAC), San Francisco, CA, USA, 2021, pp. 1321–1326.
- [8] 28th International Symposium on VLSI Design and Test (VDATE 2024), "Design and ASIC Implementation of Area Efficient UART Core in SCL 180nm Technology," 2024.
- [9] X. Zhang and W. Wang, "Low Power Physical Design Flow for Cryptographic Accelerators," Integration, the VLSI Journal, vol. 85, pp. 21–30, May 2022.
- [10] Gupta and R. Mehta, "FPGA Implementation of Modified UART Protocol with Error Detection," in Proc. 2023 IEEE Int. Midwest Symp. Circuits and Systems (MWSCAS), Windsor, Canada, 2023, pp. 219–223.
- [11] K. V. Prasad and N. Gupta, "Power and Area Optimized Physical Design of UART Controller in 65 nm Technology," Int. J. of Electronics and Electrical Engineering, vol. 12, no. 3, pp. 97–103, Sep. 2020.
- [12] M. Kumar and S. Chatterjee, "FPGA Based RSA Implementation for Data Security in IoT," Int. J. of Reconfigurable Computing, vol. 2022, Article ID 889732, pp. 1–9, 2022.
- [13] J. Singh and P. Patel, "Physical Design Challenges in 28 nm Technology Node," in Proc. 2021 IEEE Computer Society Annual Symp. on VLSI (ISVLSI), Tampa, FL, USA, 2021, pp. 219–224.
- [14] Bose and S. Ray, "FPGA Based Implementation of RSA Cryptosystem Using Montgomery Multiplication," Int. J. of Electronics and Communication Engineering, vol. 105, pp. 178–185, Nov. 2019.

- [15] S. L. Panda and M. Mohanty, "Synthesis and Timing Closure for UART Design in 90 nm Technology," in Proc. 2020 IEEE Int. Symp. on Electronic System Design (ISED), Bhubaneswar, India, 2020, pp. 125–130.
- [16] P. Choudhury and V. Narayanan, "RTL to GDSII Physical Design Flow for Secure Communication Systems," IEEE Trans. on Semiconductor Manufacturing, vol. 36, no. 1, pp. 33–44, Feb. 2023.
- [17] R. Sharma and A. Jain, "Comparative Study of UART Implementations on FPGA and ASIC," Int. J. of VLSI Design & Communication Systems, vol. 13, no. 2, pp. 45–52, Apr. 2022.
- [18] L. Chen and Y. Wu, "ASIC-Based Cryptographic Hardware for Secure IoT Applications," IEEE Internet of Things Journal, vol. 9, no. 12, pp. 9751–9762, Jun. 2022.
- [19] M. Al-Haj and K. Salah, "Secure FPGA Based Hardware Implementation of RSA Algorithm," in Proc. 2021 Int. Conf. on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 2021, pp. 441–445.
- [20] N. P. Desai and P. R. Patel, "Timing Analysis and Physical Verification in 45 nm UART Design," Int. J. of Engineering Research & Technology, vol. 11, no. 6, pp. 122–128, Jun. 2022.
- [21] K. S. Rao and D. K. Sharma, "Physical Design Automation Tools for VLSI Implementation: A Review," Microelectronics Journal, vol. 112, pp. 105–118, Jan. 2021.
- [22] Y. Wang, Z. Li, and H. Xu, "Energy Efficient Hardware Implementation of RSA Algorithm for Wireless Networks," IEEE Access, vol. 10, pp. 68123–68133, Jul. 2022.
- [23] R. Kumar and M. Thomas, "Implementation and Analysis of UART Protocol on FPGA with Security Extensions," in Proc. 2020 IEEE Int. Conf. on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 2020, pp. 542–547.
- [24] S. A. Sheikh and H. B. Patel, "Design and Verification of Cryptographic Cores Using OpenROAD Physical Design Flow," ACM Trans. on Reconfigurable Technology and Systems, vol. 16, no. 3, pp. 1–18, Aug. 2023.