iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** I **Month of publication:** January 2025

DOI: https://doi.org/10.22214/ijraset.2025.66739

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue I Jan 2025- Available at www.ijraset.com

Optimized High-Speed VLSI Implimentation of the RSA algorithm

Nandini V Munavalli[1], Pragati Suresh Naik[2], Pranamyadevi V Hiremath[3], Ravikant Shri Biradar[4], Prema K N[5]
Department of Electronic and Communication Engineering, Jawaharlal Nehru New College of Engineering

Abstract: ***RSA, one of the most widely adopted public-key cryptographic algorithms, ensures secure communication by leveraging the mathematical properties of modular exponentiation and large prime factorization. However, its computational complexity and high resource demands pose significant challenges for real-time and high-speed applications. This paper addresses these challenges by proposing an optimized Very-Large-Scale Integration (VLSI) design for RSA encryption and decryption, focusing on accelerating the modular exponentiation process, which is the core of RSA computations.***
The design incorporates Montgomery Modular Multiplication to eliminate time-intensive division operations, enabling efficient computation in the modular arithmetic domain. It further integrates techniques such as pipelining, parallel processing, and carry-save adders to reduce critical path delays and enhance throughput. Modular exponentiation is implemented using a scalable iterative approach with the square-and-multiply method, optimized for hardware efficiency.
Hardware prototypes were synthesized and tested using FPGA and ASIC platforms, demonstrating superior performance in terms of speed, area, and power consumption. The proposed architecture achieves high-speed operation while maintaining security and scalability, making it suitable for real-time cryptographic applications such as secure communication, digital signatures, and authentication systems. Comparative analysis with existing implementations highlights significant improvements, establishing the proposed design as a viable solution for next-generation secure hardware accelerators.
Keywords: RSA algorithm, Verilog, FPGA

I. INTRODUCTION

In the modern era of digital communication, ensuring data security has become paramount. The RSA algorithm, named after its inventors Rivest, Shamir, and Adleman, is one of the most widely used public-key cryptographic systems. It is valued for its robustness and ability to secure data transmission over insecure channels. However, the computational complexity of RSA, particularly the exponential operations involved in encryption and decryption, poses significant challenges for real-time applications.

The integration of Very-Large-Scale Integration (VLSI) technology with Field-Programmable Gate Arrays (FPGAs) offers a promising solution to these challenges. VLSI allows the development of highly efficient and compact hardware implementations, while FPGAs provide the flexibility to design, prototype, and optimize such systems rapidly. By leveraging FPGA platforms, researchers can achieve parallelism and high-speed performance, crucial for implementing computationally intensive tasks like RSA encryption and decryption.

This paper explores the implementation of the RSA algorithm on FPGA-based VLSI designs. It focuses on optimizing key parameters such as area, speed, and power consumption to meet the demands of real-time cryptographic applications. The paper also highlights the advantages of using FPGAs for RSA, including hardware level parallelism, reconfigurability, and reduced development cycles compared to traditional ASIC solutions.

The discussion encompasses the modular arithmetic operations fundamental to RSA, such as modular exponentiation and multiplication, which are optimized using FPGA resources. Additionally, various design approaches, including pipelining and parallel processing, are analyzed to enhance system performance. This implementation not only demonstrates the feasibility of FPGA-based RSA systems but also sets a foundation for future research into scalable and efficient cryptographic hardware designs.

Through this work, we aim to provide insights into the practical challenges and solutions associated with implementing RSA on FPGA boards, contributing to the broader field of secure hardware design and cryptographic system development.

©IJRASET: All Rights are Reserved

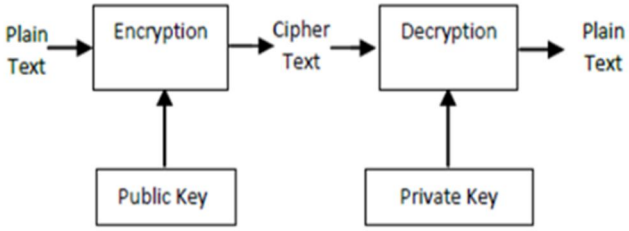


Figure 1: Public key cryptography

II. LITERATURE SURVEY

[1] Sheba Diamond Thabah, Mridupawan Sonowal, Rekib Uddin Ahmed, Prabir Saha*, "Fast and Area Efficient Implementation of RSA Algorithm, ICRTAC 2019.

REMARK: The paper reports a high-frequency, low-latency RSA cryptosystem using a shift-add multiplier and a binary digit-based modular exponentiation circuit. By discarding the most significant bit (MSB), the system achieves higher operating frequencies, The proposed design shows notable improvements in speed and efficiency, validated using both HDL simulation and Xilinx ISE, with ASIC implementation also explored.

[2] Aashish Parihar, Sangeeta Nakhate, "Low latency high throughput Montgomery modular multiplier for RSA cryptosystem", 2022.

REMARK: The proposed Montgomery modular multiplier is designed to achieve both low latency and high throughput, which are essential for efficient RSA cryptosystems in modern communication systems.

[3] Somnath Mondal, Prof Sachin Patkar"Hardware-Software co-implementation of a high performance and light-weight scalable Systolic-Montgomery based modified RSA for portable IoT devices", 2021 International Conference on Emerging Smart Computing and Informatics (ESCI).

REMARK: This paper presents a scalable, low-power RSA cryptosystem using Systolic-Montgomery Multiplication, optimized for IoT devices. Implemented on Xilinx Artix-7 FPGA

[4] Falowo O. Mojisola, Sanjay Misra, C. Falayi Febisola, Olusola Abayomi-Alli, Gokhan Sengul, "An improved random bit-stuffing technique with a modified RSA algorithm for resisting attacks in information security," 2022 THE AUTHORS. Published by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University

REMARK: The paper proposes an enhanced RSA algorithm, named RBMRSA, that aims to improve the security of the classical RSA algorithm against various attacks.

III. OBJECTIVES

1) To generate public and private key.
2) To perform RSA encryption and decryption process.
3) To implement RSA algorithm using FPGA/hardware components.
4) To evaluate the performances and efficiency of the implemented algorithm.

IV. METHODOLOGY

RSA is a public key cryptographic algorithm. The major steps involve in RSA algorithm are

- Key generation
- Encryption process
- Decryption process

1) Key Generation

To generate the public key and private key algorithm 1 is followed. In this algorithm (algorithm 1) $p1$, $p2$ are the selected inputs, and e, d are the outputs which are the two keys required for the RSA algorithm. To generate these two keys, select two large prime numbers of equal length, i.e., $p1$ and $p2$. Compute the modulus number nn and the function $\Psi(n)$.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue I Jan 2025- Available at www.ijraset.com

Choose a positive integer e such that,$1 < e < \Psi(n)$, calculate gcd $(e, \Psi(n)) = 1$. Here, e is called the encryption key or the public key. Then, compute the decryption key or the private key$d$, $1 < d < \Psi(n)$, such that $e.\Psi(n) = 1$. The decryption key d is obtained by taking the multiplicative inverse of the encryption key e.

Algorithm 1: Key Generation

Input:	p_1, p_2
Output:	e, d
1: $n = p_1 * p_2$;	
2: $\Psi(n) = (p_1 - 1)(p_2 - 1)$;	
3: $1 < e < \Psi(n), \gcd(e, \Psi(n)) = 1$;	
4: $e.d \bmod \Psi(n) = 1$;	

2) Encryption Process

The RSA encryption process can be computed through algorithm 2, where modular exponentiation has to be performed by the sender. First, get the public key (e, n) of the recipient and also the plaintext to be sent which is represented as mm. After receiving e, n, m, encrypt the plain text into cipher-text or coded text and then transmit the cipher-text through the channel to the recipient

Algorithm 2: Encryption process

Input:	m, n, e
Output:	$m^e \bmod n$.
1: $c = m^e \bmod n$;	

3) Decryption Process

The RSA decryption process involves the recipient using their private key d and the modulus n to decrypt the ciphertext C and retrieve the original plaintext message. The decryption is carried out through modular exponentiation, where the recipient computes m= C^d \mod n, with m representing the plaintext message. The ciphertext CCC is raised to the power of the private key d, and the result is taken modulo n, which effectively reverses the encryption process. This process relies on the recipient's private key, which is kept confidential, ensuring that only they can decrypt the message. Modular exponentiation techniques, such as the **square-and-multiply** algorithm, are typically used to efficiently perform the decryption, even with large values of d and n. Thus, the RSA decryption process guarantees that only the intended recipient, who possesses the correct private key, can successfully decrypt and access the original plaintext message.

V. IMPLEMENT RSA ALGORITHM USING FPGA

To implement the RSA algorithm on the Spartan-6 FPGA, the first step is to understand the core operations of RSA: key generation, encryption, and decryption, all of which rely on modular arithmetic. The RSA algorithm involves the use of modular exponentiation, which can be computationally intensive, particularly for large key sizes. The Spartan-6 FPGA provides a suitable platform due to its DSP slices, logic cells, and block RAM, all of which can be used to optimize modular multiplication and exponentiation operations.

Key Operations and FPGA Implementation:

1) Modular Multiplication:

The RSA algorithm requires modular multiplication, which is efficiently implemented on the Spartan-6 using **Montgomery Multiplication**. This method avoids division and reduces latency by performing multiplication in a transformed domain. FPGA's DSP slices are well-suited for these high-speed arithmetic operations, ensuring that large numbers (such as those used in RSA) are handled efficiently. By pipelining the multiplication stages, the throughput can be enhanced, reducing the overall computation time.

2) Modular Exponentiation:

RSA's encryption and decryption processes require modular exponentiation, which can be implemented using the square-and-multiply algorithm. The Spartan-6 FPGA's parallel processing capabilities allow this algorithm to be pipelined, significantly improving speed. The exponentiation algorithm works by iterating through the bits of the exponent, squaring and multiplying modularly based on the bit values. Using FPGA resources such as the LUTs and DSP slices, the implementation can process each step of exponentiation in parallel.

©IJRASET: All Rights are Reserved

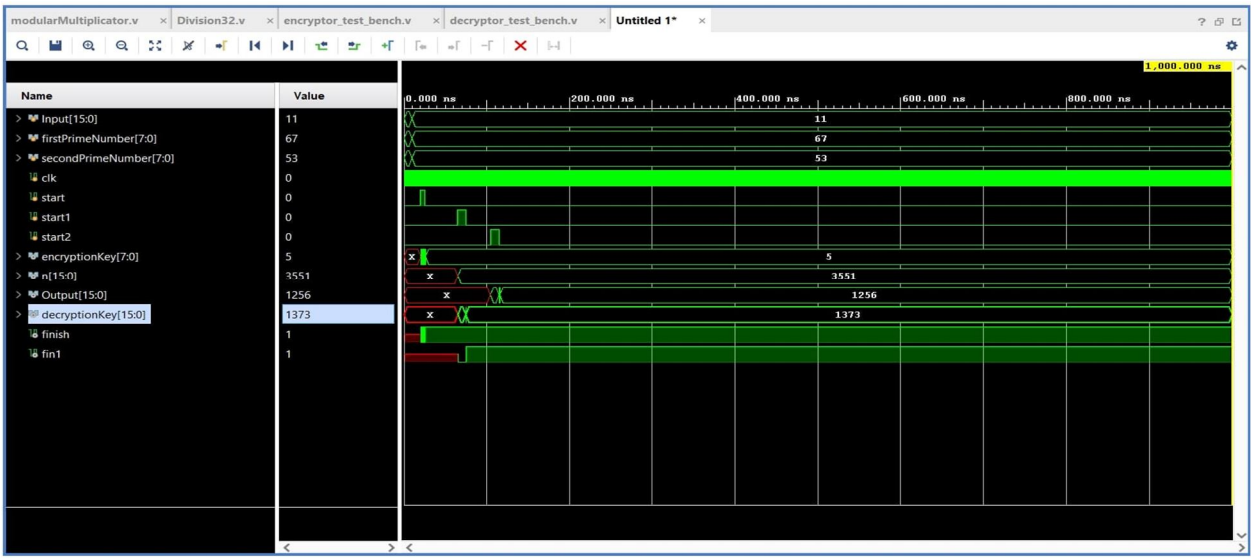
3) Control Logic and Finite State Machine (FSM):

A key part of the FPGA design is the control unit that manages the RSA algorithm's execution. This is achieved using a finite state machine (FSM) that controls the flow of operations. The FSM governs the fetching of inputs, the execution of modular arithmetic, and the storage of intermediate and final results. It ensures that the encryption or decryption process proceeds step-by-step, while managing the flow of data between different modules of the FPGA.

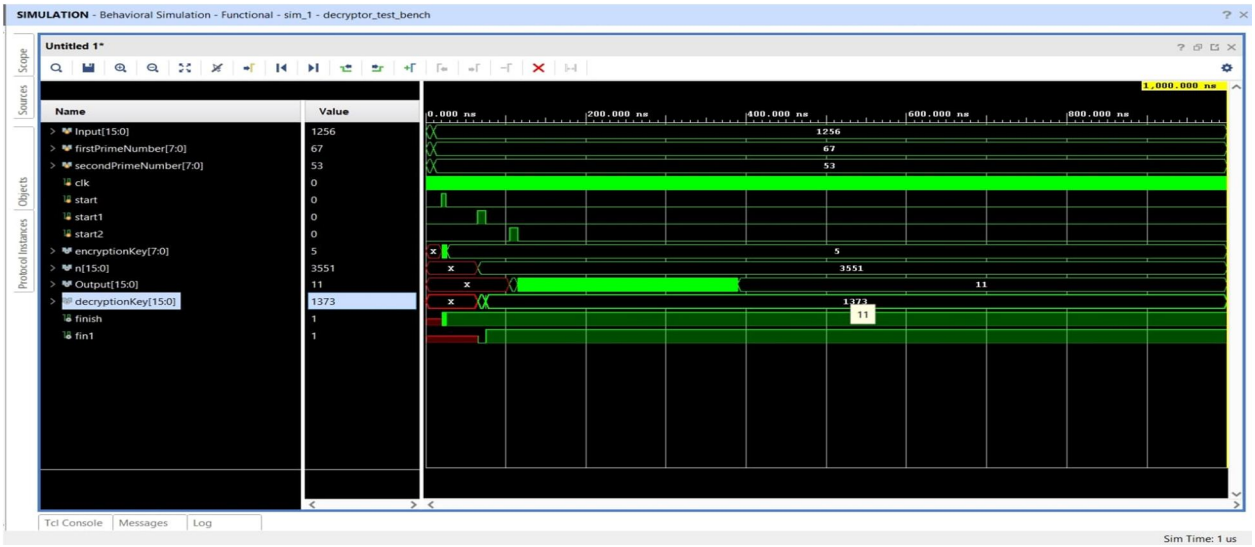
4) Synthesis and Testing:

After implementing the design in Verilog or VHDL, the next step is synthesizing the design using Xilinx's Vivado or ISE tools. These tools generate the bitstream file, which is then loaded onto the Spartan-6 FPGA for testing. Simulations are conducted to verify the correctness of the RSA algorithm and its functionality under various conditions. Performance analysis, including throughput, latency, and resource utilization, is performed to ensure the design meets the required specifications.

VI. RESULTS AND ANALYSIS



The image shows a timing simulation for a hardware design, likely for RSA encryption and decryption, based on the signals and values displayed in the waveform.



The image shows a timing simulation for a hardware design, likely for RSA encryption and decryption, based on the signals and values displayed in the waveform.

VII. CONCLUSION

Implementing the RSA algorithm on the Spartan-6 FPGA presents a powerful solution for high-speed cryptographic operations, leveraging the FPGA's inherent parallelism, reconfigurability, and hardware acceleration capabilities. By utilizing Spartan-6's DSP slices, Block RAM, and logic cells, the design effectively handles the computationally intensive tasks of modular multiplication and modular exponentiation, which are fundamental to the RSA algorithm.

The use of Montgomery Multiplication optimizes modular arithmetic by avoiding costly division operations, significantly improving the efficiency of modular multiplication on the FPGA. Additionally, implementing pipelined modular exponentiation using the square-and-multiply algorithm accelerates encryption and decryption processes. Spartan-6's resources allow for efficient handling of large RSA key sizes, such as 1024-bit and 2048-bit keys, ensuring scalability for real-world applications.

By employing a Finite State Machine (FSM) for control logic, the FPGA efficiently manages the step-by-step execution of RSA operations, ensuring that key generation, encryption, and decryption are carried out seamlessly and in the correct sequence. The design also efficiently utilizes Spartan-6's Block RAM for storing large intermediate values, which is crucial for handling the large numbers involved in RSA.

Simulation and testing using Xilinx Vivado or ISE tools verify the correctness of the design, while performance analysis—measuring throughput, latency, resource utilization, and power consumption—ensures the FPGA implementation meets the required performance benchmarks.

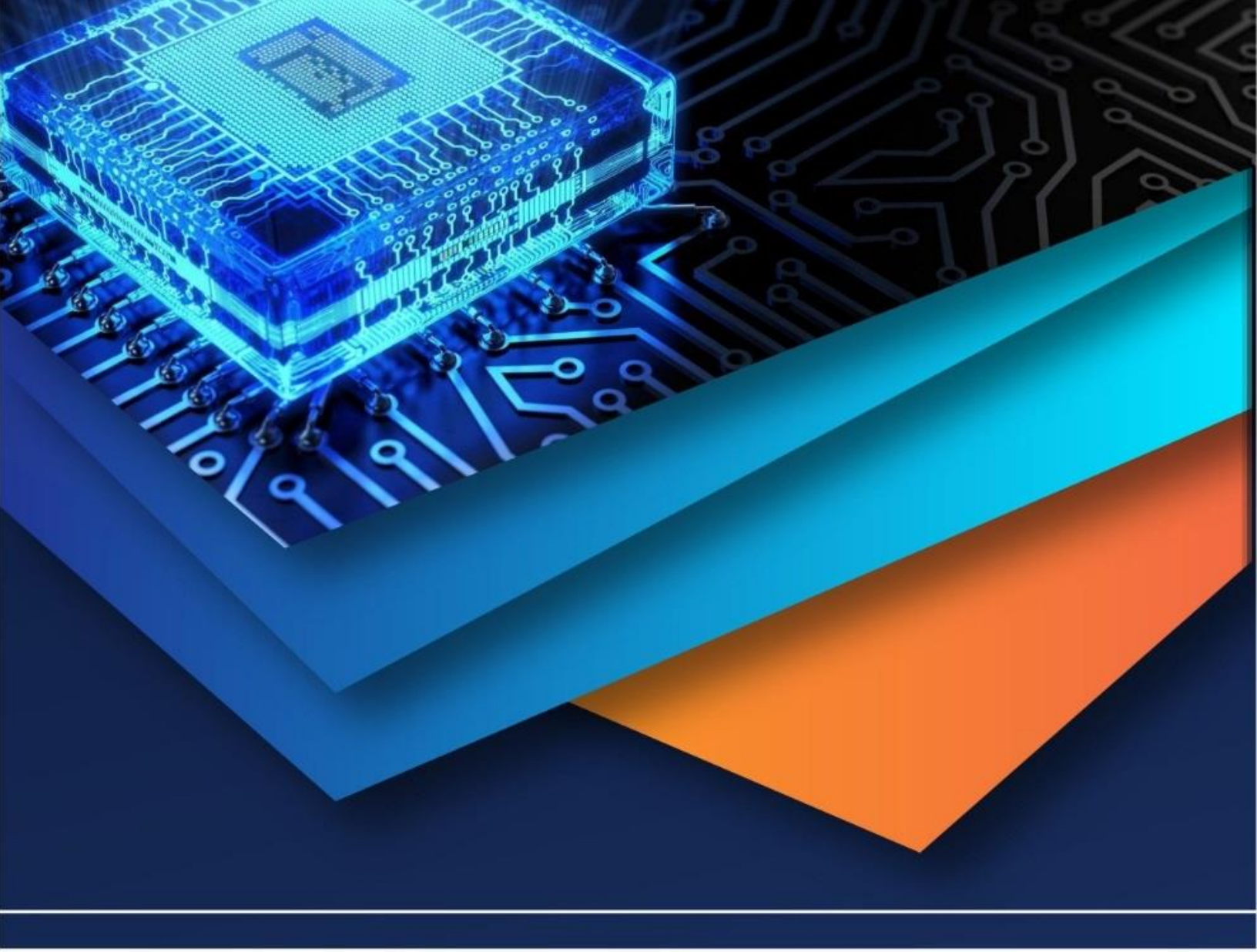
In conclusion, implementing RSA on the Spartan-6 FPGA offers a high-speed, energy-efficient, and scalable solution for cryptographic applications. This approach provides a substantial performance improvement over software-based RSA implementations, making it ideal for applications in secure communication, digital signatures, and real-time authentication systems. The Spartan-6 FPGA's versatility and resources make it a suitable platform for efficiently deploying RSA in embedded and high-performance environments.

VIII. FUTURE SCOPE

The future scope of implementing the RSA algorithm on Spartan-6 FPGA is vast and holds significant potential for enhancing cryptographic systems. One key direction is supporting larger RSA key sizes (4096-bit, 8192-bit, etc.), which would require additional FPGA resources such as more Block RAM and high-performance arithmetic units. Another area is the parallelization of RSA encryption and decryption, allowing multiple operations to be performed simultaneously, which would improve throughput in high-demand environments. Additionally, integrating RSA with symmetric cryptographic algorithms like AES for hybrid systems could optimize both speed and security in real-time applications. Preparing for quantum threats is another important future focus, where post-quantum cryptographic algorithms could be implemented on FPGA to safeguard RSA against quantum attacks. Furthermore, the optimization of RSA for embedded systems, with an emphasis on low-latency and low-power designs, will be crucial for IoT devices and mobile applications. Power optimization techniques, such as dynamic power management and voltage scaling, can be explored to minimize power consumption while maintaining high-speed performance. Distributed RSA implementations across multiple FPGA devices can also be developed for large-scale systems, enhancing performance and scalability. Integrating FPGA-based RSA into Hardware Security Modules (HSMs) would also be a promising future application, offering dedicated cryptographic processors for secure key management. Additionally, FPGA-based RSA implementations can support the growing demand for secure blockchain systems and modern cryptographic protocols like TLS 1.3. Overall, the future of RSA on FPGA offers immense potential for improving performance, scalability, and security in a variety of applications, from secure communications to cryptographic key management and beyond.

REFERENCES

[1] Bansal, A., & Soni, R. (2011). "A High-Speed VLSI Architecture for RSA Cryptosystem." International Journal of Computer Applications, 24(3), 1-4.
[2] Sinha, P., & Chattopadhyay, S. (2017). "Efficient FPGA Implementation of RSA Algorithm with High-Speed Modular Exponentiation." International Journal of Computer Science and Information Security, 15(9), 352-358.
[3] Chin, K. M., & Wong, H. K. (2015). "High-Speed and Low-Power VLSI Architecture for RSA Cryptosystem." Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, 80(5), 1146-1156.
[4] Sarkar, P., & Ghosh, S. (2010). "Efficient VLSI Design for RSA Cryptosystem Using Hardware Optimization Techniques." Proceedings of the IEEE International Conference on Electronics and Communication Engineering (ICECE), 1-6.
[5] Zhang, X., Li, Y., & Li, X. (2018). "A High-Speed FPGA-Based Implementation of RSA Cryptosystem for Real-Time Applications." Proceedings of the 2018 IEEE International Conference on Application of Electronics (ICAE), 1-4.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)