Digital Security System Design

Vũ Minh Đăng
Toán 2 22-25, VNU-HCM High School for the Gifted

Đặng Bảo Ngọc
Toán 2 22-25, VNU-HCM High School for the Gifted

Trần Thiên Phú
Tự Nhiên 2 22-25, VNU-HCM High School for the Gifted

Instructors: **Dr. Lê Đức Hùng** (University of Science VNUHCM)

Abstract

In contemporary times, driven by escalating security challenges, the domain of cryptography has experienced a profound and expansive evolution. Simultaneously, microcontrollers have found various applications in electronic devices, resulting in the demand for lightweight encryption. The central objective of this research is design and implementation of electronic circuits, specializing in encryption algorithms and lightweight encryption, targeting preconfigured microcontrollers. The methodology employed in this research focuses on designing printed circuit boards, and programming ,encrypting on-chip systems, featuring cryptographic acceleration cores. The anticipated outcomes of this study aim to develop and implement data encryption within limited hardwares such as microcontroller environments. These programmable on-chip systems are to serve as foundational software for the execution of rudimentary data encryption algorithms. This research findings is expected to make contributions to the advancement of secure data encryption. The emphasis lies in the efficient application of lightweight encryption algorithms on hardware platforms which possess limited computational capabilities, finally enhancing the security within microcontroller technology.

Introduction

Electronic devices are increasingly widespread today, so we need to ensure the security and reliability of these devices. According to Sciencedirect Topics, lightweight encryption can operate on less powerful hardware while still ensuring moderate security. Many lightweight encryption algorithms have been used, applied, and designed for specific applications. These algorithms demonstrate diverse hardware and software performance in different situations. This report focuses on the foundational knowledge for applying lightweight encryption on FPGA (Field-Programmable Gate Array) hardware, leveraging the advantages of FPGA in customizing hardware configurations (as mentioned in Electronic Design Help).In addition to that is the process of designing the solar PCB circuit, SoC (System on Chip) along with boolean algebra and cryptography.

Research questions

1. *How to create an integrated circuit PCB (Printed Circuit Board)?*
2. *How to efficiently encrypt, decrypt, and verify data integrity?*
3. *Is encryption on less powerful hardware secure and safe?*

Methodology

Part 1: Printed circuit board (PCB)

1.1 Components

To address the creation of a PCB combined with an integrated chip, we began by designing a low-dropout voltage regulator circuit for charging 18650 batteries. First, we selected the appropriate components:

Designator	Footprint	LibRef	Quantity
C1	CN1	CN1	1
C2, C3	CP1/2.5	CP2	2
C4	CN1	CN1	1
D1	DO-201	1N5819	1
D2	DO-201	1N4007	1
D3	DZ0.5A	DZ	1
IC1	SOP8	SO8	1
IC2	QTO-220	LM317	1
JP1, JP2	HED2	Header 2_1	2
L1	L01	L1	1

LED1, LED2	LED-3	LED	2
Q1	QTO-220	BD139	1
R1	R2W	R2W	1
R2	R1/4W	R1/4W	1
R3, R5, R7	0805_Res	R	3
R4, R6	R1/4W	R1/4W	2
R8	R5W	R5W	1
VR1	VR	VR1	1

1.2 Design

After obtaining the necessary components, we proceeded to create a schematic library for the components using Altium Designer software. The library containing the pinout diagrams can be found in this repository.

Next, we created the schematic diagram for the circuit, and finally, we laid out the components on the PCB board and connected them accordingly. The design file for the charging circuit can be found in this repository.

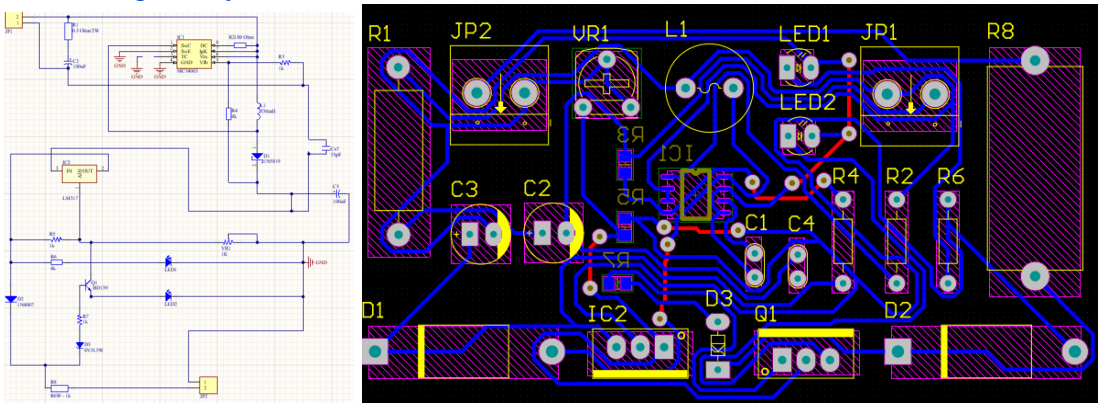

Figure 1.2a. Solar charger schematic and layout

Part 2: Digital design

2.1 Overview

In this section, we delved into researching the software, algorithms, and algebraic operations used in the reconfigurable system-on-chip. Specifically, the hardware used in this system includes the DE10-Standard board, which offers a wide range of features for implementing various circuit designs, from simple circuits to diverse multimedia multi-project applications.

FPGA Device	- Cyclone V SX SoC—5CSXFC6D6F31C6N - 110K LEs, 41509 ALMs - 5,761 Kbits embedded memory - 6 FPGA PLLs and 3 HPS PLLs - 2 Hard Memory Controllers
ARM-Based Hard Processor System (HPS)	- 925 MHz, Dual-Core ARM Cortex-A9 MPCore Processor - 512 KB of Shared L2 Cache - 64 KB of Scratch RAM - Multiport SDRAM Controller with Support for DDR2, DDR3, LPDDR1, and LPDDR2 - 8-Channel Direct Memory Access (DMA) Controller
Configuration and Debug	- Serial Configuration Device – EPCS128 on FPGA - On-Board USB Blaster II (Normal Type B USB Connector)
Memory Device	- 64MB (32Mx16) SDRAM on FPGA - 1GB (2x256Mx16) DDR3 SDRAM on HPS - MicroSD Card Socket on HPS
Communication	- Two USB 2.0 Host Ports (ULPI Interface with USB Type A Connector) on HPS - USB to UART (Micro USB Type B Connector) on HPS - 10/100/1000 Ethernet on HPS - PS/2 Mouse/Keyboard - IR Emitter/Receiver
Connectors	- One 40-pin Expansion Header (Voltage Levels: 3.3V) - One HSMC Connector(Configurable I/O Standards 1.5/1.8/2.5/3.3V) - One 10-Pin ADC Input Header - One LTC Connector (One Serial Peripheral Interface (SPI) Master ,One I2C and One GPIO Interface) on HPS
Display	- 24-bit VGA DAC - 128x64 Dots LCD Module with Backlight on HPS
Audio	- 24-bit CODEC, Line-in, Line-out, and Microphone-In Jacks
Video Input	- TV Decoder (NTSC/PAL/SECAM) and TV-In Connector
ADC	- Sample Rate: 500 KSPS - Channel Number: 8 - Resolution: 12 bits - Analog Input Range : 0 ~ 4.096 V
Switches,	- 5 User Keys (FPGA x4, HPS x1)

Buttons and Indicators	- 10 User Switches (FPGA x10) - 11 User LEDs (FPGA x10 ; HPS x 1) - 2 HPS Reset Buttons (HPS_RST_n and HPS_WARM_RST_n)7-Segment Display x6
Sensors	- G-Sensor onHPS
Power	- 12V DC input

Detailed data of the system used includes connection addresses and functions of each component, which can be found in [this repository.]

2.2 Boolean Algebra Simulation

Boolean algebra is used to build and optimize logic gates in digital circuits. Logic gates such as AND, OR, NOT, XOR, NAND, NOR, and XNOR are constructed by mapping Boolean algebra operations. Therefore, Boolean algebra plays a crucial role in this research.

To understand how semiconductor devices work in the circuit, we use Digital Logic Simulator (DLS) software to perform simulations of both combinational and sequential circuits. This helps us handle binary arithmetic and design electronic and computer circuits.

2.3 1 bit Half Adder

2.3.1 Design

Using logic gates and Boolean algebra, we simulated a 1-bit half-adder circuit in the DLS software with the schematic diagram as follows:

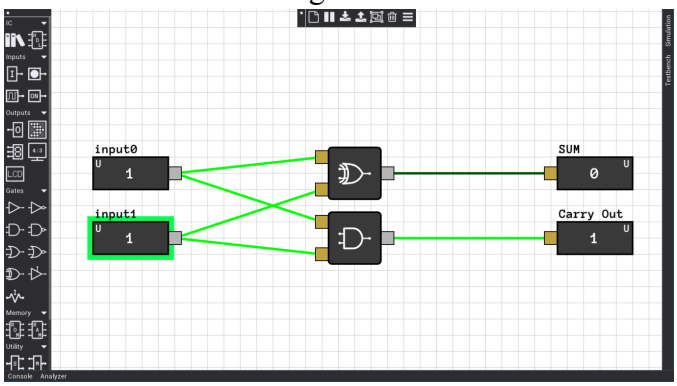

Figure 2.3.1a. Schematic diagram of the 1-bit half-adder circuit

The half-adder circuit performs simple addition using AND and XOR logic gates, yielding a truth table as follows:

A	B	Sum	Carry
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

2.3.2 Theory

Inputs A and B are two 1-bit binary numbers that we will add together. The half-adder circuit produces two results: the output Sum is the result of adding A and B, and the output Carry is the carry bit, if any.

The operation of the half-adder circuit is as follows:

- The output Sum (addition result) is computed using the XOR (exclusive or) operation between A and B. In other words, the Sum result will be 1 if either input A or B is 1, but not both. If both inputs are 1, Sum will be 0.

$$Sum = A \wedge B$$

- The output Carry (carry bit) is computed using the AND operation between A and B. This means that Carry will be 1 only when both inputs (A and B) are 1. If at least one of them is 0, Carry will be 0.

$$Carry = A \,\&\, B$$

2.4 1 bit Full Adder

2.4.1 Design

Using logic gates and Boolean algebra, we simulated a 1-bit full adder circuit in the DLS software with the schematic diagram as follows:

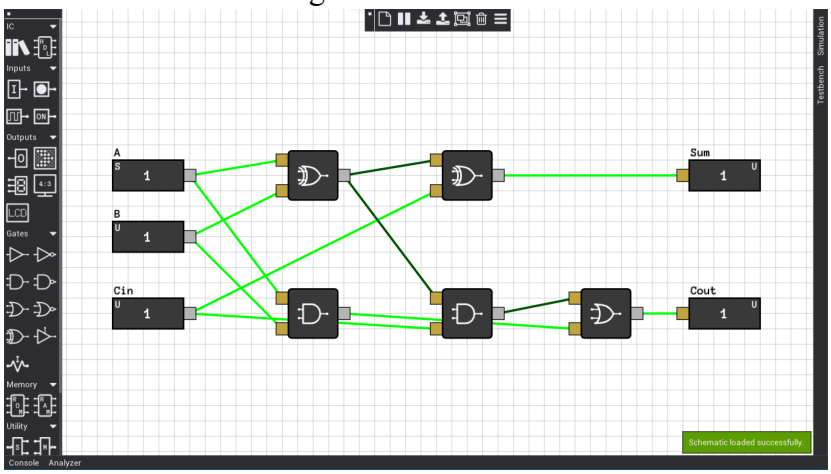

Figure 2.4.1a. Schematic diagram of the 1-bit full adder circuit

The full adder circuit performs simple addition using AND, OR, and XOR logic gates, resulting in a truth table as follows:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1

1	**1**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**

2.4.2 Theory

The full adder circuit has three inputs and two outputs.
Input A holds the value of the first number, input B holds the value of the second number, and input Cin takes the value from the previous carry bit if any.
The two outputs of the full adder circuit are:
The output S is the algebraic sum, the addition result of the two numbers A, B, and Cin. The output Cout carries the value to the next input bit if available.
The operation of the full adder circuit is as follows:

- The Sum (S) is the result of adding the A bit, the B bit, and the previous carry bit (Cin). S is calculated using the XOR gate of the three inputs:

$$S = A \wedge B \wedge Cin$$

- The next carry bit (Cout) is determined using AND and OR gates:

Cout = (A & B) || (Cin & (A ^ B))

2.5 Hardware Description Language

To program the hardware on the DE10 Standard board, we chose the Verilog programming language for the following reasons:

- The DE10 Standard board uses Intel's FPGA and supports Verilog in Intel's development tools like Quartus Prime and ModelSim.
- Verilog is one of the two most popular Hardware Description Languages (HDL) in the electronics industry, alongside VHDL, as noted in What is the future of Verilog in VLSI industry.

2.6 Hardware Programming

In this section, we use the Verilog language to program and load the source code into the FPGA system through the Quartus software. The source code that has been loaded into the system can be found on the research log page. The table below describes the corresponding locations of each component on the DE10 Standard board.

CLOCK_50	PIN_AF14	LEDR[2]	PIN_AC23	HEX1[2]	PIN_AE16	HEX3[5]	PIN_AA19
SW[0]	PIN_AB30	LEDR[3]	PIN_AD24	HEX1[3]	PIN_AD17	HEX3[6]	PIN_AD20
SW[1]	PIN_Y27	LEDR[4]	PIN_AG25	HEX1[4]	PIN_AE18	HEX4[0]	PIN_AD21
SW[2]	PIN_AB28	LEDR[5]	PIN_AF25	HEX1[5]	PIN_AE17	HEX4[1]	PIN_AG22
SW[3]	PIN_AC30	LEDR[6]	PIN_AE24	HEX1[6]	PIN_V17	HEX4[2]	PIN_AE22
SW[4]	PIN_W25	LEDR[7]	PIN_AF24	HEX2[0]	PIN_AA21	HEX4[3]	PIN_AE23
SW[5]	PIN_V25	LEDR[8]	PIN_AB22	HEX2[1]	PIN_AB17	HEX4[4]	PIN_AG23
SW[6]	PIN_AC28	LEDR[9]	PIN_AC22	HEX2[2]	PIN_AA18	HEX4[5]	PIN_AF23

SW[7]	PIN_AD30	HEX0[0]	PIN_W17	HEX2[3]	PIN_Y17	HEX4[6]	PIN_AH22
SW[8]	PIN_AC29	HEX0[1]	PIN_V18	HEX2[4]	PIN_Y18	HEX5[0]	PIN_AF21
SW[9]	PIN_AA30	HEX0[2]	PIN_AG17	HEX2[5]	PIN_AF18	HEX5[1]	PIN_AG21
KEY[0]	PIN_AJ4	HEX0[3]	PIN_AG16	HEX2[6]	PIN_W16	HEX5[2]	PIN_AF20
KEY[1]	PIN_AK4	HEX0[4]	PIN_AH17	HEX3[0]	PIN_Y19	HEX5[3]	PIN_AG20
KEY[2]	PIN_AA14	HEX0[5]	PIN_AG18	HEX3[1]	PIN_W19	HEX5[4]	PIN_AE19
KEY[3]	PIN_AA15	HEX0[6]	PIN_AH18	HEX3[2]	PIN_AD19	HEX5[5]	PIN_AF19
LEDR[0]	PIN_AA24	HEX1[0]	PIN_AF16	HEX3[3]	PIN_AA20	HEX5[6]	PIN_AB21
LEDR[1]	PIN_AB23	HEX1[1]	PIN_V16	HEX3[4]	PIN_AC20		

2.7 Signal Waveform Simulation

In this section, we perform algorithm simulations using the Verilog language and ModelSim software. The simulations below involve designing and testing the functionality of circuits and algorithms on the DE10 Standard FPGA. The source code for these circuits can be found in the research log.

- Multiplexer (MUX) circuit is designed to assign the output based on the control signal, following the principles of a multiplexing circuit. It has a simple structure and operates as a channel selector.
- Sequential LED shifter circuit is designed to shift LEDs from left to right on the DE10 board when a designated button is pressed. The LED shifting speed aligns with the DE10 Standard board's clock signal. The circuit employs a counter to count pulses and a register to store the LED states.
- Greatest Common Divisor (GCD) circuit is designed to calculate the greatest common divisor of two 16-bit integers. It uses clock signals, reset signals, and control signals to perform the computation.

Part 3: Cryptography

3.1 Algorithms

In this research, we focus on various encryption methods, including the Caesar cipher, ChaCha, AES, and PRINCE.

3.1.1 Caesar Cipher

The Caesar cipher involves shifting the letters in the original message by a fixed number of positions with a specific step.

Advantages	Disadvantages
- Easy to implement. - Suitable for encrypting non-sensitive information.	- Vulnerable to attacks using brute force to try all possible shifts. - Does not provide high-level security assurance.

3.1.2 Symmetric-key Algorithms

Symmetric-key encryption uses the same key for both encrypting and decrypting data. This key is known as a symmetric secret key.

Applications	Examples
- Encrypting data and messages to maintain confidentiality. - Providing security for online connections, such as through HTTPS.	- Serpent, Data Encryption Standard (DES), Advanced Encryption Standard (AES), Salsa, ChaCha20

3.1.3 Asymmetric-key Algorithms

Asymmetric-key encryption uses a pair of public and private keys for access authorization from different parties.

Applications	Examples
- Digital signature for verifying the data source. - Secure database storage. - Secure communication and protection of personal information.	- RSA, Elliptic Curve Digital Signature Algorithm (ECDSA), Ed25519

3.1.4 Hash Functions

Hash functions perform one-way encryption by converting data into a fixed character string.

Applications	Examples
- Checking and verifying the integrity of files and data. - Secure password storage. - Generating digital signatures and verifying data integrity.	- MD5, SHA-256, SHA3-256, SHA3-512

3.2 Configuring the System on a Chip (SoC)

In this section, we integrated hardware and software on a System-on-Chip (SoC) platform to provide features and functions on a single board. The process of designing and implementing an SoC system was performed using Quartus Prime and Platform Designer. This system includes a Nios II CPU, RAM, JTAG/UART interface, output LEDs, and an encryption accelerator using ChaCha.

3.2.1 SoC System Structure

We built the SoC system with the following key components:

- Nios II CPU capable of system control and management.
- RAM for data and program storage for the CPU.
- JTAG/UART interface for programming, debugging, and interacting with the system.
- Output LEDs for displaying system information and results.
- ChaCha encryption accelerator for data encryption.

3.2.2 Verilog Code for the SoC System

The Verilog source code that describes the SoC system is a module used to establish connections and utilize the components within the SoC system.
After configuring the FPGA, we used encryption algorithm source code for the Nios II CPU. The source code for encryption algorithms, including ChaCha, AES, and PRINCE, implemented on the DE10 Standard board, can be found in this research log.

- The ChaCha encryption accelerator is used to perform data encryption, including configuring the number of algorithm rounds. Then, data is input and encrypted. Finally, the actual encryption result is compared to the theoretical value to verify the accuracy of the encryption.
- PRINCE lightweight encryption is a type of symmetric encryption designed for improved performance by reducing encryption complexity.

Results

1. Efficiency

To simulate encryption algorithms, especially to evaluate the efficiency of lightweight encryption on constrained hardware, configuring and setting up the SoC system appropriately will provide results regarding speed and accuracy. This allows for optimization in terms of energy consumption, computer resources, and security.
In general, each encryption algorithm used and simulated in this research has its own speed, security, and efficiency characteristics. The Caesar encryption algorithm has fast processing speed but does not guarantee security and is relatively easy to break due to the algorithm's characteristics. The AES encryption algorithm provides high security but is computationally intensive and challenging to break due to its block-based encryption method. Meanwhile, the lightweight encryption algorithm PRINCE is a resource-efficient approach for weaker hardware in smart electronic devices while still ensuring reasonable reliability.

2. Cost-effectiveness

After this internship, we have completed a self-designed PCB board for a voltage-reducing and self-cut-off charger for 18650 batteries, with a cost of approximately 50,000 VND per board. Detailed component costs can be found in this research log.
Additionally, our team used the Terasic DE10 Standard board to build the SoC system for simulating encryption algorithms. The price for this product is approximately $790 (refer to DE10-STANDARD DEV KIT P0493 Terasic).

Discussion

1. Printed circuit board (PCB)

The voltage-reducing and load-cutting charging circuit for 18650 rechargeable batteries has helped charge the battery and protect it from overheating or overloading. In the circuit design, we used the 1N5819 diode and LM317 voltage regulator, where the diode controls the current and prevents reverse flow, and the LM317 circuit allows for adjustable output voltage based on a custom signal. Therefore, the 18650 rechargeable battery can adapt to various input voltage levels, preventing damage or hazards.

2. Digital design

2.1 1 bit Half Adder

A 1-bit half adder has a simple structure with basic logic gates like AND and XOR. Therefore, it is easy to implement and resource-efficient. Additionally, 1-bit half adders play a crucial role in

building processors, memory units, and other digital logic circuits. Furthermore, 1-bit half adders serve as building blocks for full adders, enabling the addition of multi-bit binary numbers.

2.2 1 bit Full Adder

A 1-bit full adder can be used to alter values in encryption through bitwise addition and in substitution-permutation networks (S-boxes). This adder can also be employed to rearrange bits in input data, creating asymmetry in the P-box substitution network. In one-way encryption, a 1-bit full adder can be used to create irreversible transformations, making it harder to break the encryption. In symmetric encryption, a 1-bit full adder can be used in data transformation operations, such as performing XOR operations between character strings (keys) and original data to generate encrypted data.

3. Cryptography

While the PRINCE encryption algorithm has made improvements in terms of speed, performance, and security, we still observe some potential issues and risks.

- Although PRINCE encryption has improved in terms of asymmetry, the algorithm can still be broken or subjected to brute-force attacks.
- PRINCE encryption uses relatively small bit and byte sizes, so this type of encryption may not be suitable for applications that require encryption with a larger number of bits or bytes.

The lightweight PRINCE encryption has applications in various fields:

- PRINCE encryption can be used in IoT devices with limited resources to ensure security and performance.
- IoT Devices: PRINCE encryption can be used in IoT devices with limited resources to ensure security and performance.

Conclusion

The application of encryption in electronic devices has become widespread, leading to increased requirements for lightweight encryption security. Therefore, the development of electronic circuits and the application of encryption, especially lightweight encryption, on pre-configured microcontrollers are essential.

We have successfully applied chip-based systems with programmable capabilities and integrated encryption accelerators. The results of this research will support the development of hardware for data encryption in microcontrollers and programmable chip-based systems. This will lay the foundation for enhancing the security of microcontrollers and advancing the development of lightweight encryption algorithms on resource-constrained hardware.

In the field of smart electronic devices, security and reliability are becoming crucial concerns. Lightweight encryption is a solution capable of operating on resource-constrained hardware while ensuring security. Many lightweight encryption algorithms, such as PRINCE, have been applied and optimized for applications using less powerful hardware, providing promising results with adequate security.

Acknowledgment

We would like to express our sincere gratitude to all those who have contributed to the completion of this project.

Firstly, we truly value the guidance and aid given to us during our project by our instructors, Mr. Lê Đức Hùng. His insightful feedback and suggestions helped to shape the project's scope. Secondly, we would also like to express gratitude to Mr. Thái Hồng Hải and Mr. Mã Khải Minh, our mentors, for their unwavering assistance whenever our project faced technical issues. Additionally, we would like to thank Deslab - University of Science for giving us the supplies, environment, and resources that were crucial to the completion of our project. Lastly, we wish to acknowledge the PIISE coordination team for their continuous supervision and assistance during our participation in the program, as well as the PIISE organizing team for developing such a valuable learning and experience program.

Without the contributions and assistance of all those mentioned above, this project would not have been possible. Thank you all for your valuable contributions and support.

References

Terasic Inc (2018). DE10-Standard Development Kit Specifications. Retrieved from Terasic - DE Boards - Cyclone - DE10-Standard

Joachim Strömbergson (2019). The Prince lightweight block cipher in Verilog. Retrieved from secworks/prince: The Prince lightweight block cipher in Verilog. (github.com)

Joachim Strömbergson (2019). Verilog 2001 implementation of the ChaCha stream cipher. Retrieved from secworks/chacha: Verilog 2001 implementation of the ChaCha stream cipher. (github.com)

Joachim Strömbergson (2019). Verilog implementation of the symmetric block cipher AES (NIST FIPS 197). Retrieved from secworks/aes: Verilog implementation of the symmetric block cipher AES (Advanced Encryption Standard) as specified in NIST FIPS 197. This implementation supports 128 and 256 bit keys. (github.com)

Without the contributions and assistance of all those mentioned above, this project would not have been possible. Thank you all for your valuable contributions and support.

Appendix

This research project is carried out within the PTNK Initiative in Interdisciplinary Science and Engineering (PIISE) Summer Research Internship Program under the topic Information Technology at Deslab - University of Science from 27/07/2023 - 29/10/2023.

This research project has achieved significant milestones, including:

Time period	Milestone
24/07/2023 - 20/08/2023	Implementing PCB Design
21/08/2023 - 01/10/2023	FPGA and Encryption Approach
02/10/2023 - 29/10/2023	Building a SoC System

With the aim of sharing knowledge, aligned with the spirit of continuous learning and accumulation in science, we would like to share the tools, knowledge, and useful keywords that have been beneficial to us during the course of this project, including:

- Altium
- Quartus Prime
- Mật mã hóa nhẹ
- Chacha20
- AES
- PRINCE
- Caesar

At the end of the program, we have gained valuable experiences for ourselves, and we are ready to continue our path of continuous learning and research. We hope that the results we share will make a meaningful contribution to the community and will be continuously built upon.