HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA ĐIỆN TỬ - VIỄN THÔNG

BÁO CÁO

THIẾT KẾ VI MẠCH SỐ

ĐỀ TÀI: "Thiết kế lõi IP thực hiện mã hóa và giải mã Chinese SM4 theo chế độ CTR"

Nhóm sinh viên:	**LÊ TRỌNG MINH**	**MSV: DT050122**
	THÂN QUANG PHONG	**MSV: DT050124**
	TRẦN VIỆT LINH	**MSV: DT050120**
	TRẦN TRỌNG ÁNH DƯƠNG	**MSV: DT050109**

Giảng viên hướng dẫn: TS. Mai Đức Thọ

Hà Nội, tháng 7 năm 2025

Mục lục

Danh sách hình vẽ

LỜI NÓI ĐẦU

Trong bối cảnh nhu cầu bảo mật thông tin ngày càng tăng cao, việc sử dụng các lõi IP mã hóa phần cứng trở thành một hướng đi thiết thực để tăng tốc độ xử lý và đảm bảo an toàn dữ liệu trong các hệ thống nhúng và FPGA. Với mong muốn tiếp cận và thực hành với các thuật toán mã hóa hiện đại, nhóm chúng em lựa chọn đề tài ***"Thiết kế lõi IP thực hiện mã hóa và giải mã Chinese SM4 theo chế độ CTR"*** *để nghiên cứu và ứng dụng.*

Mục tiêu chính của đề tài là:

- Tìm hiểu, nghiên cứu và phân tích thuật toán Chinese SM4 và chế độ CTR.
- Thiết kế lõi IP thực hiện mã hóa và giải mã SM4 theo chế độ CTR.
- Thực hiện mô phỏng, kiểm tra, tổng hợp trên FPGA và triển khai trên OpenLane.

Cuối cùng, nội dung của đề tài chúng em sẽ thực hiện theo các chương như sau:
Chương 1: Tổng quan, giới thiệu
Chương 2: Thiết kế hệ thống
Chương 3: Thực nghiệm, đánh giá hệ thống

1. TỔNG QUAN, GIỚI THIỆU

1.1 Thuật toán Chinese SM4

1.1.1 Khái niệm

SM4 (SMS4) là một thuật toán mã hóa khối tiêu chuẩn của Trung Quốc, được thiết kế để bảo vệ dữ liệu trong các ứng dụng như mạng không dây WLAN và các hệ thống thông tin mật. SM4 sử dụng mã hóa đối xứng với độ dài khối 128 bit và độ dài khóa 128 bit, hoạt động với cấu trúc 32 vòng lặp.

Hình 1.1: Giới thiệu minh họa về SM4 CTR

1.1.2 Đặc điểm

- Kích thước khối: 128 bit.
- Độ dài khóa: 128 bit.
- Thuật toán hoạt động theo 32 vòng lặp phi tuyến tính, mỗi vòng gồm S-box, phép dịch vòng và XOR.
- Có cấu trúc dễ thực hiện trên phần cứng và phần mềm.
- Khóa giải mã được tạo bằng cách đảo ngược thứ tự các khóa con của quá trình mã hóa.

1.1.3 Tiêu chuẩn hoạt động

- Thuật toán sử dụng các phép toán cơ bản như XOR 32 bit, phép dịch vòng và ánh xạ S-box.
- Mỗi vòng sử dụng một khóa con khác nhau, được sinh ra từ khóa chính thông qua quá trình mở rộng khóa.
- Quá trình mã hóa và giải mã có cấu trúc giống hệt nhau, chỉ khác thứ tự sử dụng khóa con.
- Hỗ trợ các chế độ hoạt động như ECB, CBC, CTR để phù hợp nhiều ứng dụng.

1.1.4 Lợi ích

- Bảo mật cao, được kiểm chứng và sử dụng rộng rãi tại Trung Quốc.
- Hiệu quả tính toán tốt, phù hợp cho cả hệ thống nhúng, FPGA và ASIC.
- Thuật toán đơn giản, dễ cài đặt, hỗ trợ song song hóa ở cấp độ vòng lặp.
- Là tiêu chuẩn mã hóa mở, được quốc gia công nhận, dễ tích hợp vào các ứng dụng bảo mật hiện đại.

1.2 Chế độ CTR (Counter Mode)

1.2.1 Khái niệm

- Chế độ CTR (Counter Mode) là một chế độ hoạt động của các thuật toán mã hóa khối, trong đó bộ đếm (counter) được mã hóa và kết quả mã hóa này sẽ được XOR với plaintext để tạo thành ciphertext hoặc XOR với ciphertext để tạo ra plaintext khi giải mã.
- Điểm đặc biệt của chế độ CTR là cùng một quá trình được sử dụng cho cả mã hóa và giải mã, giúp giảm độ phức tạp phần cứng khi triển khai lõi IP.

1.2.2 Đặc điểm, cấu trúc hoạt động

Cấu trúc hoạt động:

- Chế độ CTR sử dụng một giá trị bộ đếm (Counter) khác nhau cho mỗi khối dữ liệu. Bộ đếm này sẽ được mã hóa bằng SM4 để tạo ra một khối mã hóa.
- Khối mã hóa này sau đó được XOR với khối dữ liệu gốc (plaintext) để tạo ra ciphertext, hoặc XOR với ciphertext để thu được plaintext.
- Việc sử dụng bộ đếm tăng dần giúp đảm bảo tính độc nhất cho mỗi khối mã hóa.

Ưu điểm:

- Hỗ trợ **mã hóa và giải mã song song** các khối dữ liệu, do bộ đếm có thể được tạo trước, rất phù hợp cho các thiết kế FPGA, ASIC cần tốc độ cao.

- Không bị lan truyền lỗi: nếu xảy ra lỗi bit trong ciphertext, chỉ khối dữ liệu đó bị ảnh hưởng, không lan sang các khối khác.
- Dễ dàng triển khai, tận dụng tối đa tài nguyên phần cứng.
- Đảm bảo tính bảo mật cao nhờ sự thay đổi liên tục của bộ đếm kết hợp với khóa bí mật.

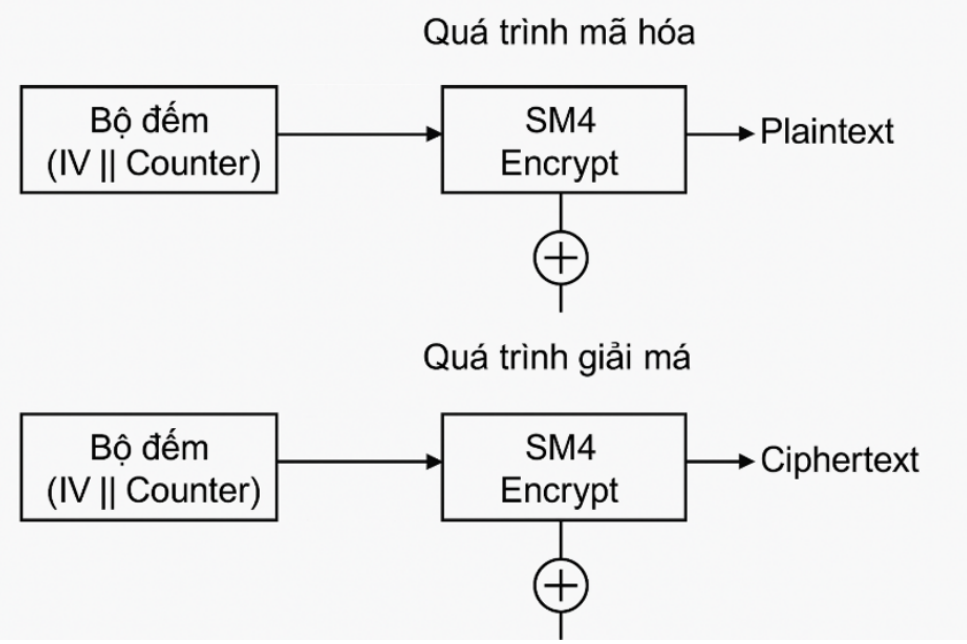
Nhược điểm:

- Bộ đếm cần được quản lý chính xác để tránh trùng lặp, nếu không sẽ ảnh hưởng đến tính bảo mật.
- Không cung cấp tính toàn vẹn dữ liệu, cần kết hợp thêm cơ chế kiểm tra tính toàn vẹn (MAC) khi cần.

1.2.3 Sơ đồ hoạt động chế độ CTR

- **Quá trình mã hóa:** Bộ đếm (IV || Counter) → SM4 Encrypt → XOR với Plaintext → Ciphertext.
- **Quá trình giải mã:** Bộ đếm (IV || Counter) → SM4 Encrypt → XOR với Ciphertext → Plaintext.

Cả hai quá trình sử dụng cùng một khối mã hóa SM4, rất tiện cho thiết kế lõi IP trong **Thiết kế Vi mạch số**.



Hình 1.2: Sơ đồ khối hoạt động SM4 chế độ CTR

1.3 Phần mềm thiết kế

1.3.1 Phần mềm Vivado

Hình 1.3: Phần mềm Vivado

Vivado là phần mềm có rất nhiều chức năng, dùng để lập trình và phát triển cho tất cả các loại chip của Xilinx. Nói một cách ngắn gọn, nó hỗ trợ tất cả các khâu của quá trình thiết kế logic sử dụng FPGA.

Vivado Design Suite cung cấp một môi trường để cấu hình, thực thi, kiểm tra và tích hợp IP (Intellectual Property). IP có thể bao gồm logic, các vi xử lý nhúng, các module xử lý tín hiệu số DSP, hoặc các thiết kế thuật toán. Các tính năng chính trong quá trình thiết kế:

- Vivado synthesis
- Vivado implementation
- Vivado timing analysis
- Vivado power analysis
- Bitstream generation

1.3.2 Phần mềm OpenLane

OpenLane là một nền tảng triển khai silicon sáng tạo hỗ trợ các công cụ nguồn mở như Yosys, OpenROAD, Magic, KLayout, cùng với các tiện ích nguồn mở và độc quyền khác. Từ năm 2020, OpenLane đã được sử dụng cho mọi Open MPW và chipIgnite shuttle. OpenLane tích hợp và tóm tắt các bước khác nhau của quá trình triển khai silicon, cho phép người dùng củng cố dự án của họ bằng các tệp cấu hình đơn giản.

Hiện tại, có hai phiên bản OpenLane, mỗi phiên bản được thiết kế riêng cho từng trường hợp sử dụng:

OpenLane 1: Luồng triển khai Silicon nguồn mở đã được thử nghiệm và chứng minh.

OpenLane 1 là một nền tảng triển khai silicon ổn định được xây dựng hoàn toàn trên phần mềm nguồn mở. Nó cung cấp một cài đặt đơn giản dựa trên Docker® và đã cho phép vô số băng ghi âm cho các chương trình chipIgnite của Efabless và Open MPW của Google, hoàn toàn miễn phí.

OpenLane 2: Nền tảng triển khai silicon thế hệ tiếp theo cho các chip tùy chỉnh cao.

OpenLane 2 tái hiện OpenLane không chỉ là một luồng triển khai đơn giản mà còn là một nền tảng triển khai silicon có thể tùy chỉnh hoàn toàn. Nó cho phép người dùng viết các bước hoặc luồng triển khai ASIC hoàn toàn tùy chỉnh.

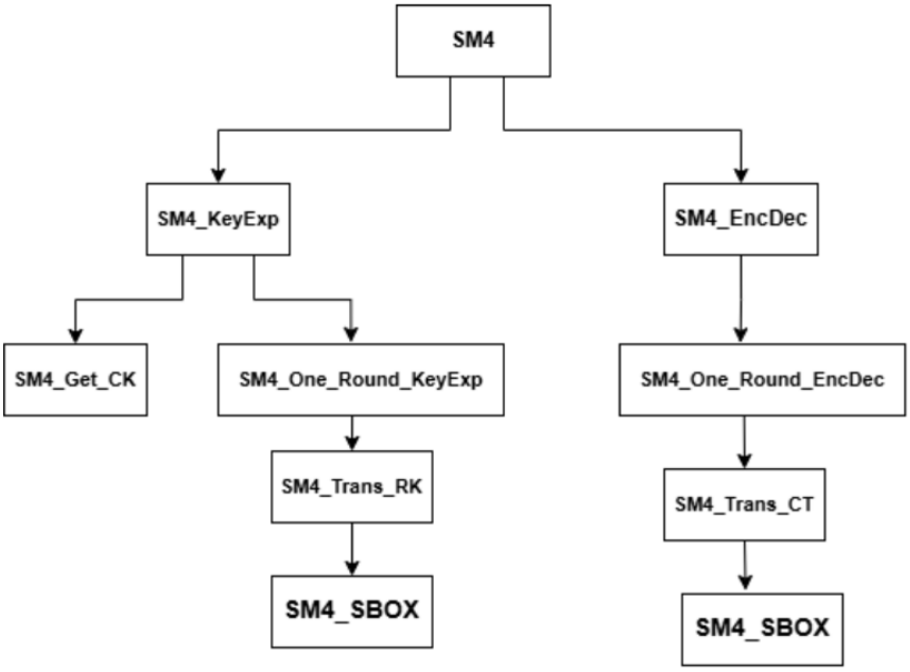
Tính năng và lợi ích:

- **Tính ổn định và linh hoạt:** OpenLane 2 giới thiệu một cơ sở hạ tầng ổn định hỗ trợ việc tạo ra nhiều luồng tùy chỉnh.
- **Trải nghiệm người dùng được nâng cao:** Người dùng được hưởng lợi từ việc xác thực cấu hình hoàn chỉnh, xử lý lỗi linh hoạt hơn và tính linh hoạt để tiếp tục quy trình làm việc từ các giai đoạn cụ thể.
- **Khả năng tùy chỉnh là cốt lõi:** Với OpenLane 2, người dùng có thể viết các bước riêng của mình bằng Python, tạo các luồng phức tạp với khả năng ra quyết định và thậm chí thử nghiệm trong Python Notebooks.

2. THIẾT KẾ HỆ THỐNG

2.1 Mô hình tổng thể

2.1.1 Sơ đồ khối kết nối các module



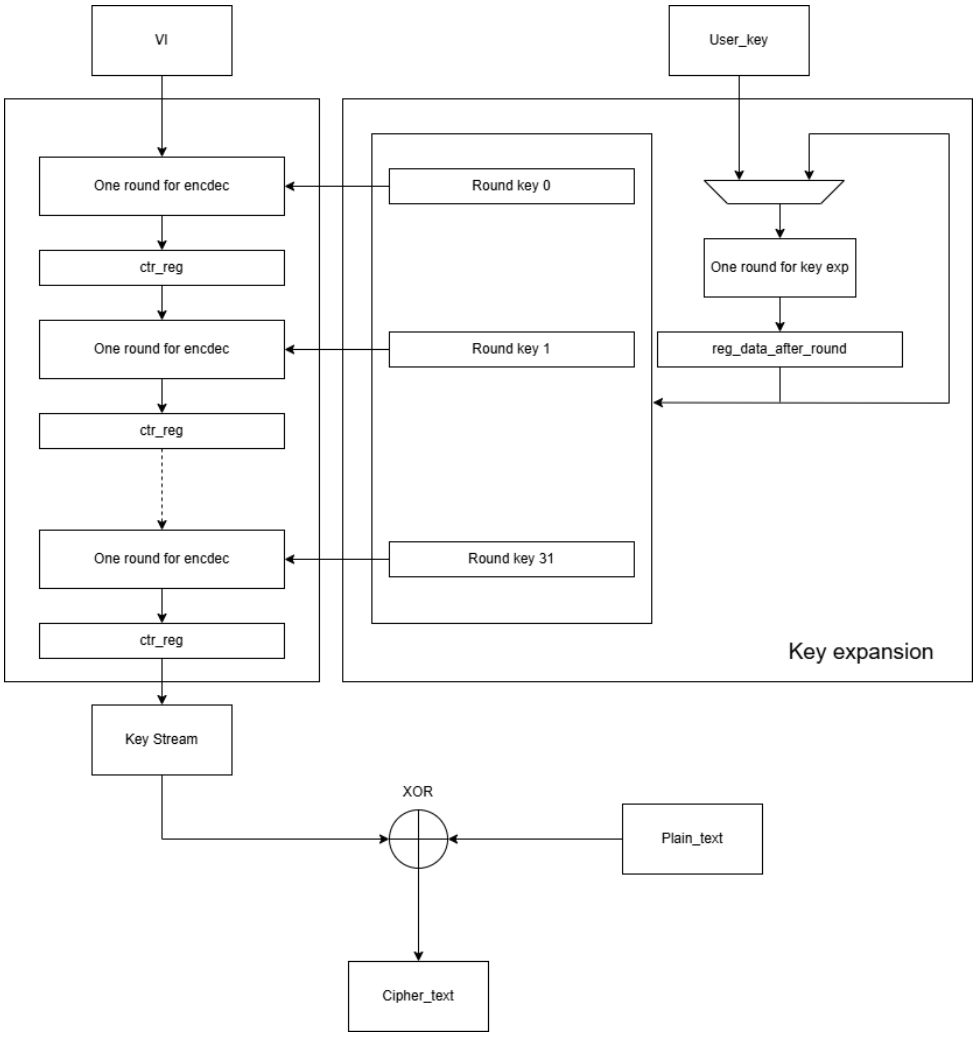
Hình 2.1: Sơ đồ khối

Các module thành phần:

- **SM4**: Module chính, điều phối toàn bộ quá trình mã hóa SM4 trong chế độ CTR.
- **SM4_KeyExp**: Module mở rộng khóa, tạo ra 32 khóa vòng (round keys) từ khóa đầu vào.
- **SM4_EncDec**: Module thực hiện mã hóa hoặc giải mã dữ liệu (trong chế độ CTR, chỉ mã hóa được sử dụng).
- **SM4_SBOX**: Module thực hiện phép thay thế phi tuyến (S-box) cho cả mở rộng khóa và mã hóa/giải mã.
- **SM4_Get_CK**: Module cung cấp các hằng số khóa (CK) cho mỗi vòng của quá trình mở rộng khóa.

- **SM4_Trans_RK**: Module thực hiện phép biến đổi phi tuyến trong mở rộng khóa.
- **SM4_Trans_CT**: Module thực hiện phép biến đổi phi tuyến trong mã hóa/giải mã.
- **SM4_One_Round_KeyExp**: Module thực hiện một vòng của quá trình mở rộng khóa.
- **SM4_One_Round_EncDec**: Module thực hiện một vòng của quá trình mã hóa/giải mã.

2.1.2 Kiến trúc mã hoá và giải mã



Hình 2.2: Kiến trúc mã hoá

VI - Vector khởi tạo

- Là giá trị khởi đầu cho bộ đếm CTR.
- Được tăng dần theo từng vòng mã hóa tiếp theo.

Vòng lặp sinh khóa dòng - Key Stream Generation

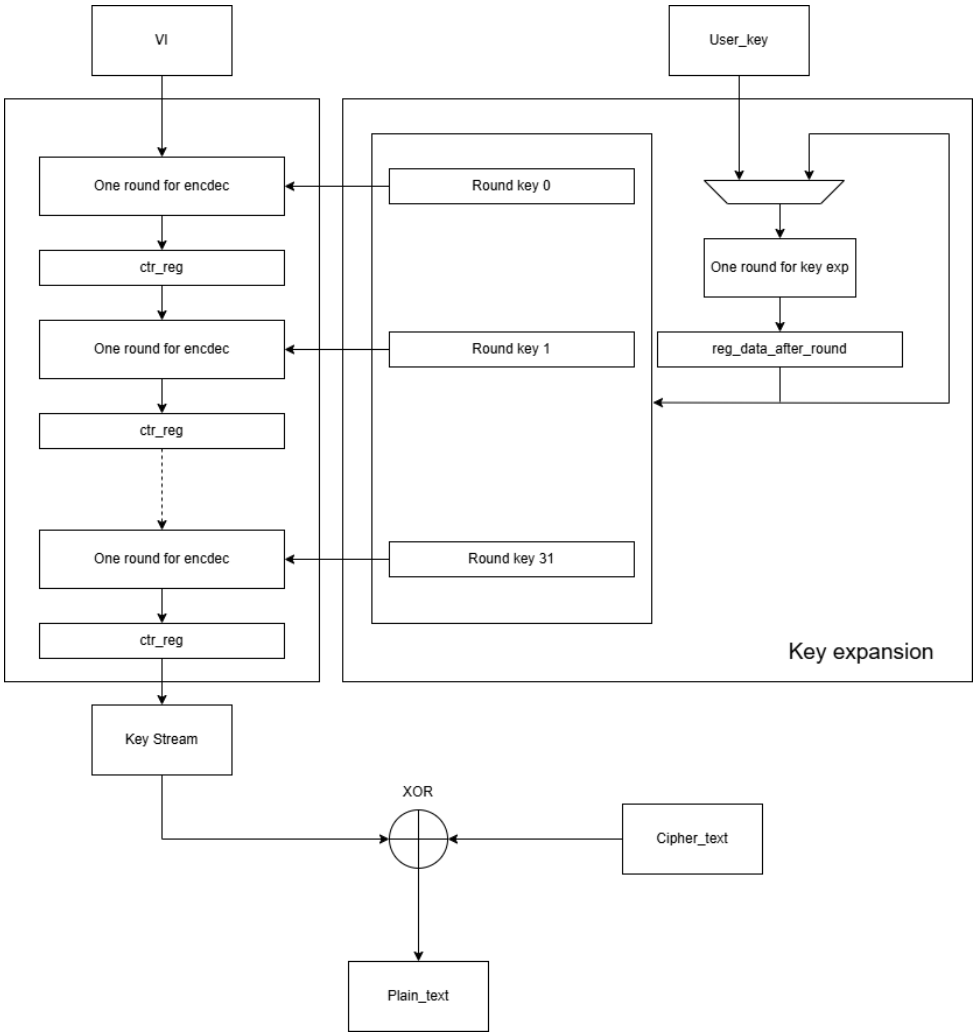
- Mỗi giá trị của bộ đếm `ctr_reg` được đưa vào khối mã hóa.
- Khối mã hóa gồm nhiều vòng xử lý, mỗi vòng sử dụng một *round key* đã được sinh ra từ quá trình mở rộng khóa.
- Kết quả đầu ra là một khối dữ liệu được gọi là **Key Stream**.

Khối mở rộng khóa - Key Expansion

- Nhận khóa người dùng đầu vào.
- Trải qua một vòng xử lý để sinh ra từng *round key*.
- Tổng cộng sinh ra 32 *round key* từ `rk0` đến `rk31`.

XOR với bản mã để giải mã

- Bản mã (*cipher text*) sẽ được thực hiện phép XOR với **Key Stream**.
- Kết quả thu được là bản rõ (*plain text*).



Hình 2.3: Kiến trúc giải mã

VI - Vector khởi tạo

- Khởi tạo bộ đếm ban đầu tương tự như kiến trúc mã hoá.

Sinh khóa dòng - Key Stream Generation

- Giá trị của bộ đếm được đưa vào vòng mã hóa nhiều tầng.
- Mỗi vòng sử dụng các *round key* đã được sinh ra từ khối mở rộng khóa.
- Kết quả tạo thành dòng khóa mã hóa - **Key Stream**.

Mở rộng khóa - Key Expansion

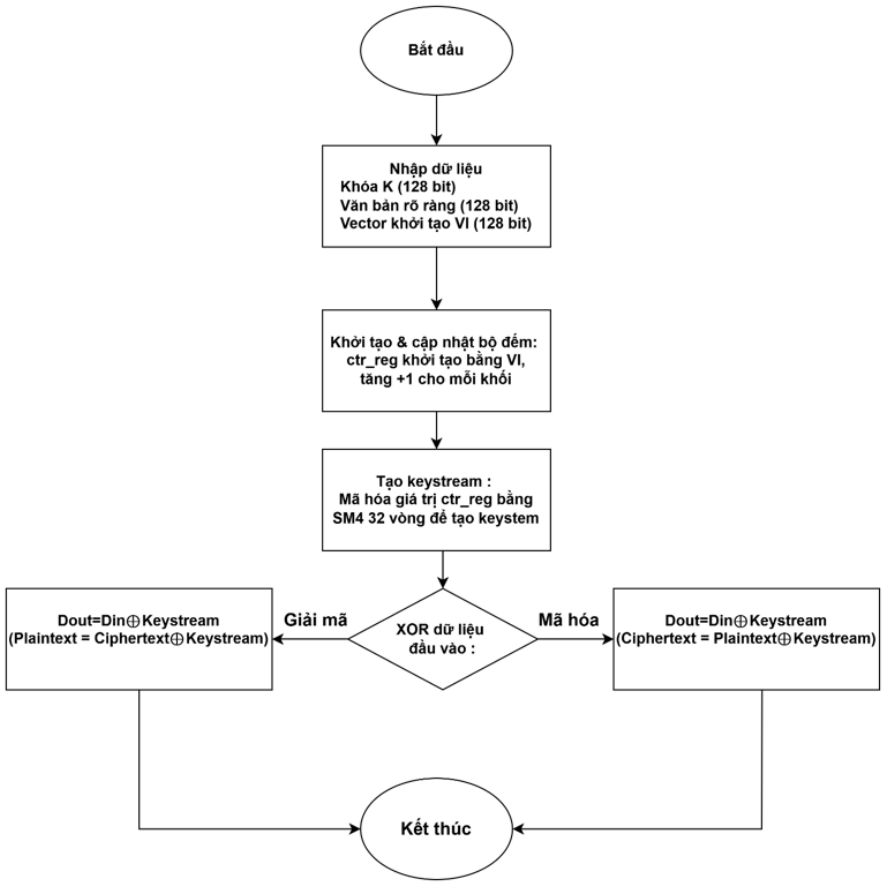
- Tương tự kiến trúc mã hoá: từ khóa người dùng sinh ra 32 *round key*.

XOR với bản rõ để mã hóa

- Dữ liệu bản rõ (*plain text*) được XOR với **Key Stream**.

- Kết quả là bản mã (*cipher text*).

2.1.3 Lưu đồ thuật toán



Hình 2.4: Lưu đồ thuật toán mã hóa và giải mã SM4 chế độ CTR

Các bước hoạt động của thuật toán:

1. Bắt đầu

Khởi động hệ thống, cấp xung **CLK**, đảm bảo **RSTn = 1** để sẵn sàng hoạt động.

2. Nhập dữ liệu

- **Khóa K (128-bit):** Khóa bí mật dùng trong SM4.
- **Văn bản rõ ràng Din (128-bit):**
  - Là **plaintext** khi **mã hóa**.
  - Là **ciphertext** khi **giải mã**.
- **Vector khởi tạo VI (128-bit):** Dùng để khởi tạo bộ đếm ban đầu trong CTR.

3. Khởi tạo & cập nhật bộ đếm

- Đăng ký **ctr_reg** khởi tạo bằng **VI**.

- Sau mỗi khối xử lý xong, **ctr_reg = ctr_reg + 1** đảm bảo **mỗi khối dữ liệu có một giá trị bộ đếm duy nhất**.

4. Tạo keystream

- Mã hóa giá trị **ctr_reg** bằng **thuật toán SM4 32 vòng** để tạo ra **keystream**.
- Keystream được sinh ra **có cùng kích thước 128-bit** với Din.

5. XOR dữ liệu đầu vào
Khi MÃ HÓA:

- Din = **Plaintext**.
- Thực hiện phép: Ciphertext = Keystream $\oplus$ Plaintext
- Kết quả **Dout** là **Ciphertext**.

Khi GIẢI MÃ:

- Din = **Ciphertext**.
- Thực hiện phép: Plaintext = Keystream $\oplus$ Ciphertext
- Kết quả **Dout** là **Plaintext**.

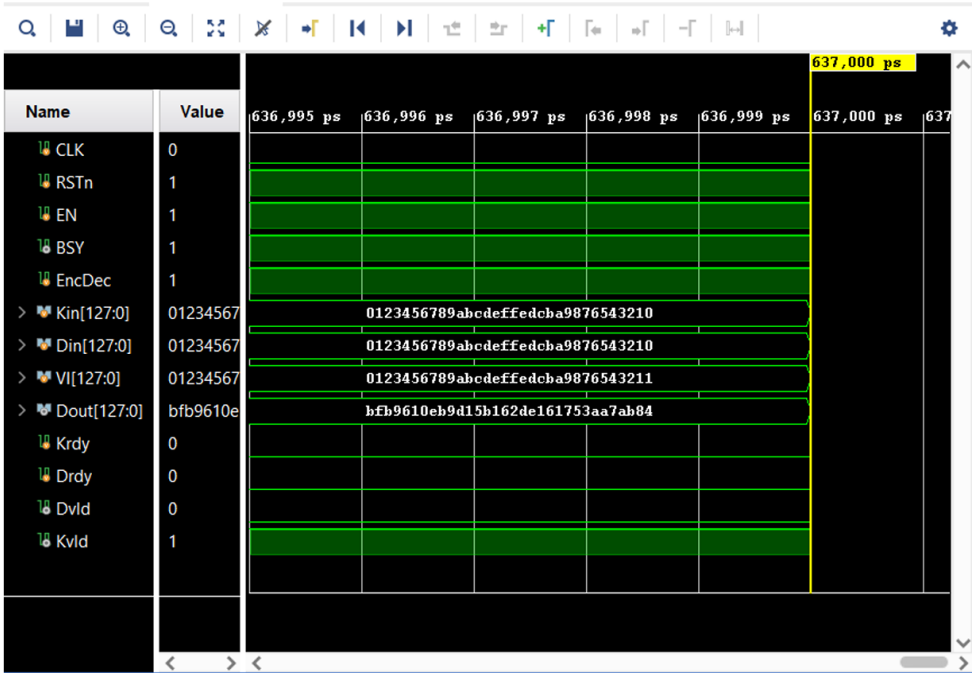
6. Kết thúc

- Sau khi khối dữ liệu được xử lý:
  - **ctr_reg tăng lên 1** để sẵn sàng cho khối tiếp theo.
  - Hệ thống tiếp tục xử lý khối tiếp theo cho đến khi hết dữ liệu.

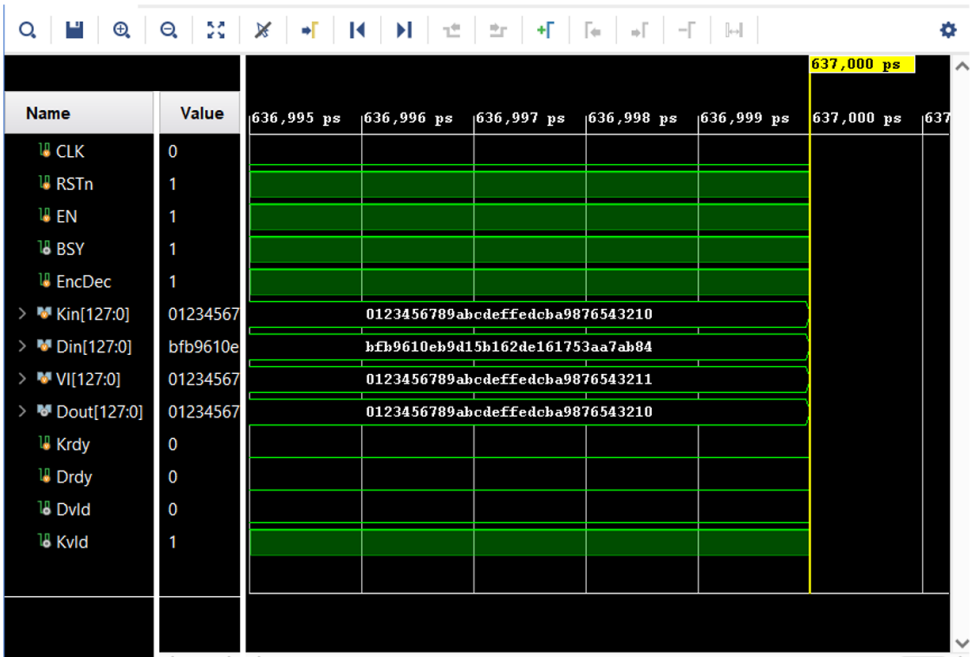
3. THỰC NGHIỆM, ĐÁNH GIÁ HỆ THỐNG

3.1 Mô phỏng hệ thống sử dụng Testbench

3.1.1 Kết quả mô phỏng mã hoá và giải mã



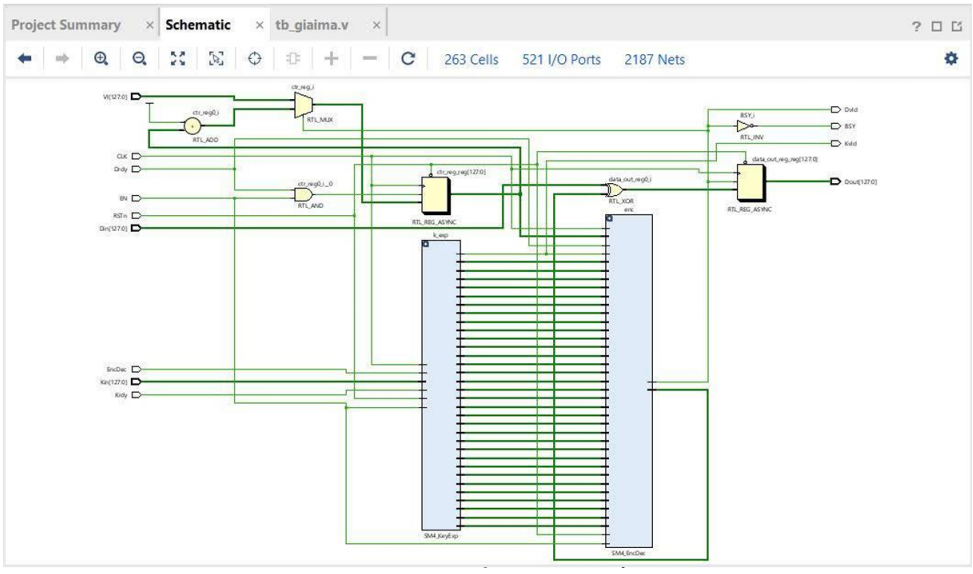
Hình 3.1: Kết quả mã hoá



Hình 3.2: Kết quả giải mã

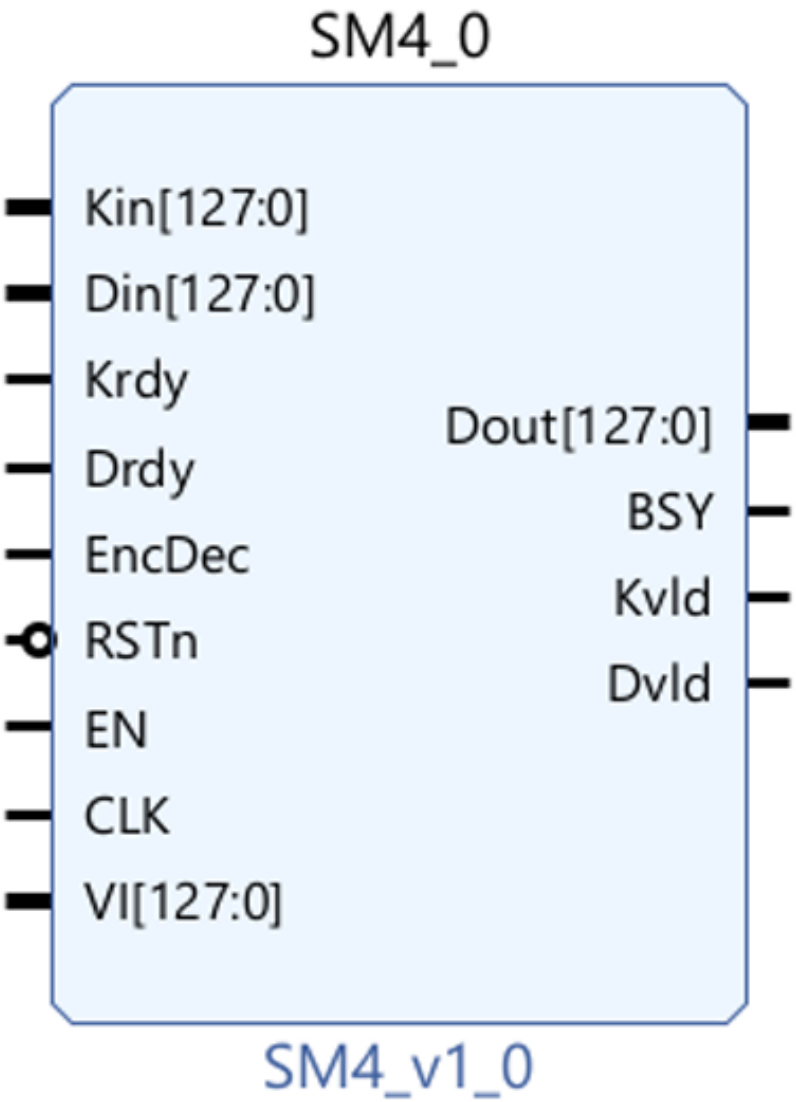
3.2 Tổng hợp hệ thống

3.2.1 Kết quả tổng hợp hệ thống



Hình 3.3: Tổng hợp hệ thống

3.2.2 Khối IP được tạo

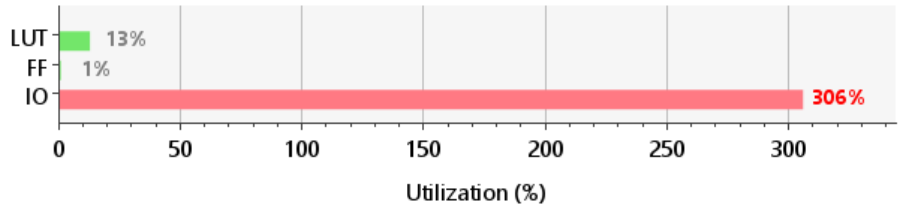


Hình 3.4: Khối IP Sm4-CTR

3.3 Tổng số tài nguyên FPGA cần sử dụng

3.3.1 Mức độ sử dụng tài nguyên

Resource	Utilization	Available	Utilization %
LUT	8359	63400	13.18
FF	1731	126800	1.37
IO	520	170	305.88



Hình 3.5: Mức độ sử dụng tài nguyên

Dữ liệu bảng

Tài nguyên	Sử dụng	Sẵn có	Tỷ lệ sử dụng
LUT (Look-Up Tables)	8,359	63,400	13.18%
FF (Flip-Flops)	1,731	126,800	1.37%
IO (Input/Output)	520	170	305.88%

Phân tích biểu đồ

- Biểu đồ cột hiển thị tỷ lệ sử dụng phần trăm của từng tài nguyên:
  - LUT: 13% (màu xanh lá)
  - FF: 1% (màu cam nhạt)
  - IO: 306% (màu đỏ)
- IO vượt quá 100%, cho thấy tài nguyên IO đã bị sử dụng quá mức so với khả năng sẵn có.

Nhận xét

- LUT và FF đang được sử dụng ở mức rất thấp (dưới 15%), cho thấy tài nguyên này còn dư dả.
- IO có tỷ lệ sử dụng 305.88%, nghĩa là số lượng IO sử dụng (520) vượt xa số lượng sẵn có (170).

- Điều này cho thấy thiết kế hiện tại không khả thi hoặc cần điều chỉnh để giảm số lượng IO hoặc tăng tài nguyên IO.

Kết luận

Hệ thống hiện tại có vấn đề nghiêm trọng với tài nguyên IO, cần được xem xét lại thiết kế hoặc cấu hình để đảm bảo sử dụng trong giới hạn khả năng.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Sau quá trình nghiên cứu và thực hiện, nhóm chúng em đã hoàn thành đề tài **"Thiết kế lõi IP thực hiện mã hóa và giải mã Chinese SM4 theo chế độ CTR"** theo đúng mục tiêu đã đặt ra.

Lõi IP SM4-CTR đã được:

- Đảm bảo tính chính xác và ổn định khi mã hóa và giải mã dữ liệu theo chuẩn SM4.
- Có thể tích hợp vào các hệ thống nhúng, FPGA, ASIC phục vụ cho các ứng dụng bảo mật, truyền thông an toàn.
- Tận dụng ưu điểm của chế độ CTR giúp mã hóa song song nhiều khối dữ liệu, không lan truyền lỗi giữa các khối, phù hợp với các hệ thống yêu cầu tốc độ xử lý cao.

2. Hướng phát triển

Trong tương lai, nhóm dự kiến phát triển đề tài theo các hướng:

Tối ưu hóa lõi IP

- Ứng dụng kỹ thuật *pipelining* sâu hơn, tối ưu xử lý song song để tăng *throughput*.
- Tối ưu kiến trúc FSM và quản lý bộ đếm để tiết kiệm hơn tài nguyên FPGA/ASIC.

Nâng cao bảo mật phần cứng

- Tích hợp các kỹ thuật chống tấn công kênh kề (*Side-Channel Attacks*) như DPA, SPA.

Bổ sung chức năng kiểm tra tính toàn vẹn

- Bổ sung cơ chế kiểm tra tính toàn vẹn dữ liệu (MAC) khi truyền nhận.

Triển khai thực tế

- Thử nghiệm lõi IP trên các nền tảng FPGA thực tế như Artix-7, Spartan-7, Cyclone.
- Kết hợp lõi IP SM4 CTR với các giao thức bảo mật khác để triển khai hệ thống bảo mật IoT, truyền thông tốc độ cao.

Mở rộng đa chế độ

- Phát triển lõi IP hỗ trợ các chế độ khác của SM4 như ECB, CBC, CFB, OFB để tăng tính linh hoạt và khả năng ứng dụng thực tế.

TÀI LIỆU THAM KHẢO

1. Ronald Henry Tse and Dr. Wai Kit Wong, *The SM4 Block Cipher Algorithm And Its Modes Of Operations*, https://datatracker.ietf.org/doc/draft-ribose-cfrg-sm4/02/

2. Nils Kopal, *The ShāngMì 4 (SM4) Block Cipher: A Deeper Look into China's Encryption Standard*, https://www.kopaldev.de/2024/02/05/the-shangmi-4-sm4-block-cipher-a-deeper-loc

3. CSDN Blog, *SM4 Implementation Overview*, https://blog.csdn.net/weixin_43261410/article/details/125153796