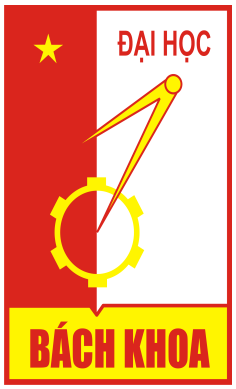
ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN - ĐIỆN TỬ



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

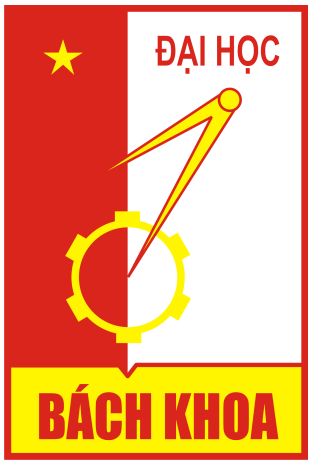
Đề tài:

THIẾT KẾ VÀ KIỂM THỬ KHỐI XỬ LÝ MẬT MÃ AES128

Sinh viên thực hiện: PHÍ MẠNH TOÀN
Lớp Điện tử 08 - K64
Giảng viên hướng dẫn: TS. NGUYỄN HOÀNG DŨNG

Hà Nội, 4 tháng 8, 2023

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN - ĐIỆN TỬ



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

Đề tài:

THIẾT KẾ VÀ KIỂM THỬ KHỐI XỬ LÝ MẬT MÃ AES128

Sinh viên thực hiện: PHÍ MẠNH TOÀN
Lớp Điện tử 08 - K64
Giảng viên hướng dẫn: TS. NGUYỄN HOÀNG DŨNG
Cán bộ phản biện:

Hà Nội, tháng , 2024

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho giảng viên hướng dẫn)

Tên giảng viên đánh giá: .....

Họ và tên sinh viên: ..... MSSV: .....

Tên đồ án: .....

.....

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:

Rất kém (1); Kém(2); Đạt(3); Giỏi(4); Xuất sắc(5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phẩy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) *(chọn 1 trong 3 trường hợp)*						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế	5				
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest)	2				
10c	Không có thành tích về nghiên cứu khoa học	0				
Điểm tổng		/50				
Điểm tổng quy đổi về thang 10						

Nhận xét khác *(về thái độ và tinh thần làm việc của sinh viên)*

.....

.....

.....

.....

.....

.....

Ngày: ... / ... / 20...

Người nhận xét

(Ký và ghi rõ họ tên)

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho cán bộ phản biện)

Giảng viên đánh giá: . . . . . .

Họ và tên sinh viên: . . . . . MSSV: . . . . . .

Tên đồ án: . . . . . .

. . . . . .

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:

Rất kém (1); Kém(2); Đạt(3); Giỏi(4); Xuất sắc(5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phẩy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) *(chọn 1 trong 3 trường hợp)*						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế	5				
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest)	2				
10c	Không có thành tích về nghiên cứu khoa học	0				
Điểm tổng		/50				
Điểm tổng quy đổi về thang 10						

Nhận xét khác của cán bộ phản biện

.....

.....

.....

.....

.....

.....

Ngày: ... / ... / 20...

Người nhận xét

(Ký và ghi rõ họ tên)

LỜI NÓI ĐẦU

Trước khi trình bày nội dung phần báo cáo đồ án của mình, em xin gửi lời cảm ơn đến thầy **TS Nguyễn Hoàng Dũng**, thầy đã tận tình hướng dẫn và giúp đỡ em để có thể hoàn thành đồ án tốt nghiệp này. Em cũng xin gửi lời cảm ơn tới thầy **Nguyễn Tiến Hòa** từ phòng Lab xử lý tín hiệu băng gốc hệ thống 5G đã cung cấp cho chúng em một bản template Latex rất chuyên nghiệp để trình bày báo cáo. Cuối cùng em xin gửi lời cảm ơn tới các anh chị tại công ty **Dolphin Technology Center Vietnam**, đã tạo điều kiện cho em được học tập và làm việc tại công ty, giúp em tiếp thu nhiều kiến thức để có những cái nhìn rõ ràng hơn về lĩnh vực vi mạch cũng như kinh nghiệm để làm việc trong tương lai. Do hạn chế về mặt thời gian và kiến thức, nội dung đồ án của em không tránh khỏi các sai sót. Em mong được sự góp ý của các thầy cô và các bạn để đề tài của em có thể hoàn thiện hơn. Một lần nữa em xin gửi lời cảm ơn và lời chúc sức khỏe, hạnh phúc tới thầy cô và gia đình.

LỜI CAM ĐOAN

Tôi tên là PHÍ MẠNH TOÀN, mã số sinh viên 20193142, sinh viên lớp Điện tử 08, khóa K64. Người hướng dẫn là TS. Nguyễn Hoàng Dũng. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án "Thiết kế và kiểm thử khối xử lý mật mã AES" là kết quả tìm hiểu nghiên cứu của tôi. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ, các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

Hà Nội, tháng , 2024

Người cam đoan

Phí Mạnh Toàn

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

	Tên tiếng Anh đầy đủ	**Tên tiếng Việt**
ĐATN	Graduation Thesis	Đồ án tốt nghiệp
AES	Advanced Encryption Standard	Tiêu chuẩn mã hoá nâng cao
RTL	Register Transfer Level	Mức độ thanh ghi
Soft IP	Intellectual Property	Quyền sở hữu trí tuệ
IC	Integrated Circuit	Vi mạch tích hợp
ECB	Electronic Codebook	
CBC	Cipher Block Chaining	
CFB	Cipher Feedback	
CFB	Output Feedback	
CTR	Counter	
UVM	Universal Verification Methodology	
DUT	Design Under Test	
LUTs	Look Up Table	

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN

THESIS SUMMARY

MỞ ĐẦU

Đặt vấn đề

Vi mạch tích hợp (IC - Integrated Circuit) là một trong những thành tựu quan trọng nhất của ngành điện tử và công nghệ thông tin trong thế kỷ 20. Thiết kế vi mạch luôn là lĩnh vực cốt lõi và lâu đời, luôn đi song song và là bệ phóng cho sự phát triển vượt bậc của các ngành kỹ thuật khác. Vi mạch tích hợp là một mạch điện tử được chế tạo trên một chip bán dẫn, có thể chứa hàng triệu hoặc thậm chí hàng tỷ thành phần điện tử như transistor, điện trở, tụ điện, và nhiều thành phần khác trên một chip nhỏ. Vi mạch tích hợp đã mang lại những cải tiến đột phá về hiệu năng, kích thước, chi phí, và độ tin cậy của các thiết bị điện tử, từ máy tính, điện thoại di động, đến tivi, máy giặt, và nhiều ứng dụng khác.

Trong lĩnh vực truyền thông, việc bảo mật thông tin là một vấn đề rất quan trọng và cần thiết. Các hệ thống truyền thông, như internet, điện thoại, vệ tinh, hay mạng không dây, đều có thể bị xâm nhập, đánh cắp, hay thay đổi thông tin bởi các kẻ có ý đồ xấu. Do đó, việc sử dụng các thuật toán mã hóa để bảo vệ thông tin là một giải pháp hiệu quả và phổ biến. . Một trong những thuật toán mã hóa được sử dụng rộng rãi trong các hệ thống truyền thông là AES (Advanced Encryption Standard). AES là một thuật toán mã hóa hiệu quả, an toàn và dễ dàng thực hiện bằng phần mềm hoặc phần cứng cũng như được áp dụng trong nhiều chuẩn giao tiếp phổ biến.

Kết hợp 2 vấn đề trên, em quyết định sẽ thực hiện đề tài **"Thiết kế và kiểm thử khối xử lý mật mã AES128"** đi từ thiết kế kiến trúc logic tới lập trình và mô phỏng trên các công cụ thiết kế vi mạch.

Mục tiêu và đóng góp của đồ án

Mục tiêu trong đồ án này là thiết kế một khối xử lý mật mã xử lý thuật toán AES. Bắt đầu từ việc tìm hiểu nền tảng lý thuyết về cơ sở toán học, các chế độ mã hoá, giải mã, các bước biến đổi. Tiếp đó là tìm hiểu về quy trình sản xuất vi mạch, các công cụ, phần mềm hỗ trợ từ đó kết hợp để lập trình, mô phỏng và kiểm thử thiết kế.

Bố cục của đồ án

Nội dung của đồ án gồm có 3 chương như sau:

Chương 1: Cơ sở lý thuyết

Trong chương này em sẽ trình bày tổng quan các kiến thức nền tảng bao gồm cấu trúc, các phép biến đổi trong hệ mật mã AES. Ngoài ra,

Chương 2: : Thiết kế khối kiến trúc mật mã

Trong chương này, em sẽ trình bày phương pháp tiếp cận của mình cho vấn đề đã đặt ra ở trên, trình bày các bước thiết kế kiến trúc chi tiết khối xử lý mật mã.

Chương 3: Xây dựng môi trường kiểm thử và kết quả mô phỏng

Trong chương này, em sẽ bắt đầu xây dựng môi trường và thuật toán kiểm thử, cuối cùng sẽ trình bày các kết quả mô phỏng đạt được để kiểm tra tính chính xác của thiết kế đã đề ra ở Chương 2.

Chương 4: Kết luận

Chương này em sẽ tổng kết về quá trình thực hiện đồ án, đưa ra các điểm đã đạt được và hạn chế của thiết kế cũng như hướng phát triển và mở rộng đồ án trong tương lai.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

Trong chương này em sẽ tập trung vào giới thiệu về cơ sở lý thuyết, bao gồm lý thuyết về hệ mật AES, cơ sở toán học, các bước biến đổi trong quá trình mã hoá, giải mã.

1.1 Giới thiệu chung

Năm 1997, NIST (National Institute of Standards and Technology) của Hoa Kỳ đã tổ chức một cuộc thi để chọn ra một thuật toán mật mã mới thay thế DES. Cuộc thi này thu hút sự quan tâm của nhiều chuyên gia an ninh và nhà nghiên cứu trên toàn thế giới. Năm 2001, Rijndael, một thuật toán được thiết kế bởi hai nhà nghiên cứu người Bỉ là Vincent Rijmen và Joan Daemen, được chọn và công bố làm chuẩn mật mã AES bởi NIST. Lý do chọn Rijndael bao gồm khả năng hiệu suất tốt, độ an toàn cao và khả năng triển khai dễ dàng trên nhiều nền tảng. Tiêu chuẩn mã hoá nâng cao- Advanced Encrytion Standard (AES) là một hệ mật mã khối đối xứng được phát triển để thay thế chuẩn DES và được ứng dụng trong nhiều lĩnh vực. So với các chuẩn mật mã hoá công khai khác, cấu trúc của AES cũng như các hệ mật đối xứng khá phức tạp và không thể tấn công dễ dàng như các hệ mật khác.

1.2 Cơ sở toán học

Trong hệ mật AES, tất cả các phép toán, phép biến đổi đều được thực hiện trên đơn vị bytes (8-bits). Đặc biệt các phép toán học như phép cộng, nhân, chia được thực hiện trên trường hữu hạn $GF(2^8)$.

Về bản chất, một trường số học là một tập hợp mà ta có thể thực hiện các phép toán cộng trừ nhân chia mà không cần rời khỏi tập hợp đó. Phép chia được xác định theo quy tắc sau: $(a/b = a(b^{-1}))$. Ví dụ về trường số học hữu hạn: Z_p bao gồm các số nguyên từ 0,1,..., p_1 trong đó p là một số nguyên tố và trong đó thực hiện phép tính số học modulo p.

Hầu như tất cả các thuật toán mã hóa, đối với cả khóa thông thường lẫn khóa công khai, đều liên quan đến các phép toán số học trên số nguyên. Nếu một trong các phép toán được sử dụng là phép chia, thì chúng ta cần làm việc với phép toán số học được xác định trong một trường, điều này là do phép chia yêu cầu mỗi phần tử khác không phải có một phép nhân nghịch đảo

Có một cách xác định trường hữu hạn 2^n phần tử, một trường như vậy được gọi là GF(2^n). Xét tập S gồm tất cả các đa thức bậc n-1 hoặc nhỏ hơn với các hệ số nhị phân. Như vậy mỗi đa thức có dạng:

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + ... + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

Với a_i nhận giá trị 0 hoặc 1. Trong tập hợp S có tổng cộng 2^n đa thức khác nhau. Với n=3 thì 2^3 = 8 đa thức trong tập hợp là: 0, x, x^2, x^2+x, 1, x+1, x^2+1, x^2+x+1 .

1.3 Cấu trúc của hệ mật AES-128

1.3.1 Cấu trúc tổng thể

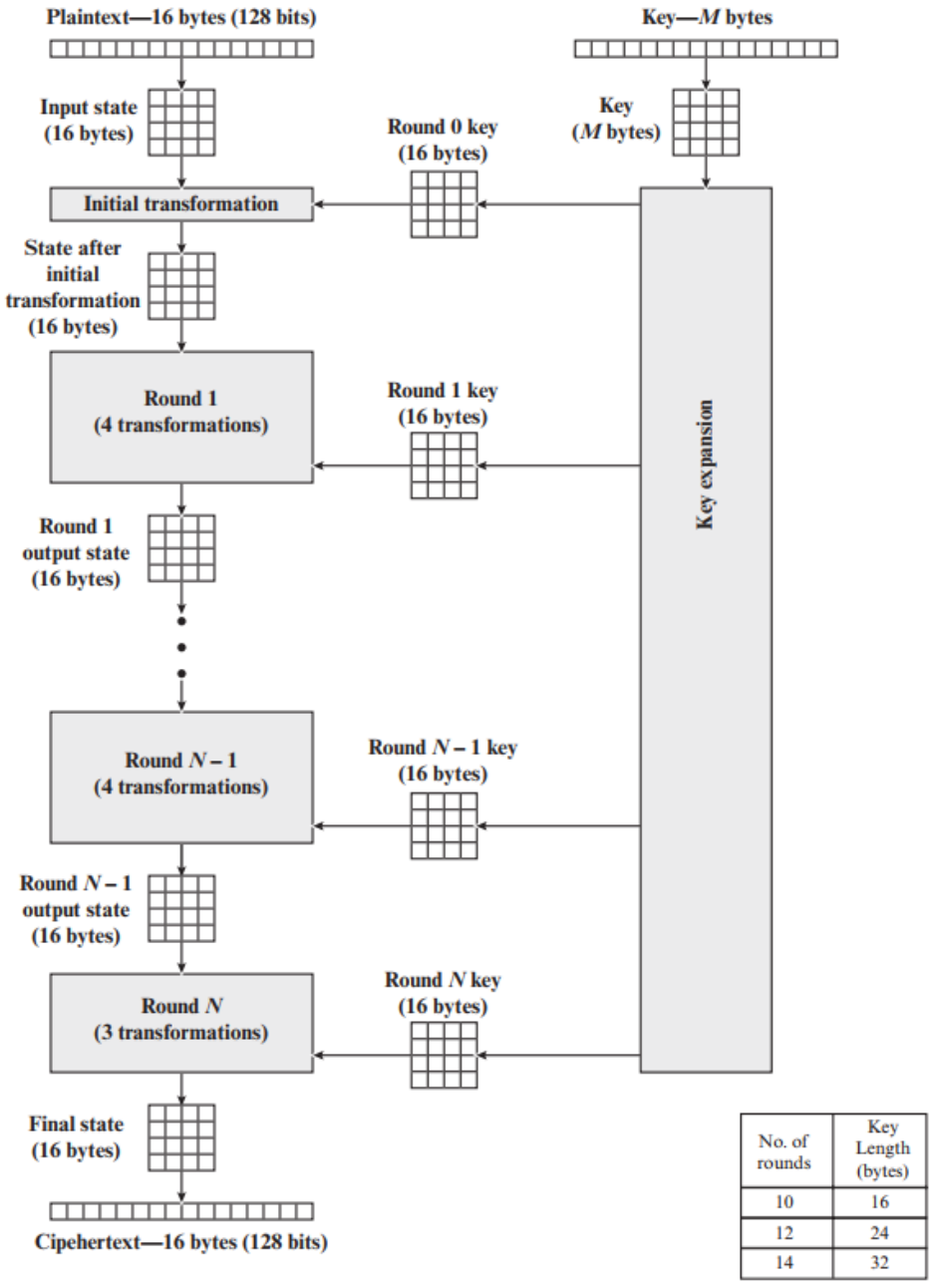
Hình cho thấy cấu trúc tổng thể của quy trình mã hóa AES. Mật mã khối AES có kích thước bản rõ (Plaintext) là 128 bits, hay 16 bytes. Độ dài của khóa (key) có thể là 16, 24, hoặc 32 bytes (128 bits, 192 bits, 256 bits). Thuật toán được gọi là AES-128, AES-192 hoặc AES-256 tùy thuộc vào độ dài của khóa.

Đầu vào của các thuật toán mã hóa và giải mã về cơ bản là một khối 128-bit duy nhất. Trong FIPS PUB 197 [2], khối này được mô tả dưới dạng ma trận vuông 4x4. Khối này được sao chép vào mảng **Trạng thái** (State array), được sửa đổi ở mỗi giai đoạn mã hóa và giải mã. Sau giai đoạn cuối cùng, **Trạng thái** được sao chép vào ma trận đầu ra. Các thao tác này được mô tả trong hình 1.1. Tương tự, khoá mỗi vòng cũng được mô tả với kích thước 128-bit và được chuyển thành ma trận 4x4 tương ứng. Hình 1.1 biểu diễn quá trình mở rộng khoá cho tiêu chuẩn AES-128 tương ứng với 44 word (4-byte). Thứ tự sắp xếp của các byte tương ứng trong ma trận là theo cột. Ví dụ, bốn byte đầu tiên của đầu vào văn bản gốc 128-bit cho mật mã mã hóa chiếm cột đầu tiên của ma trận Trạng thái, bốn byte thứ hai chiếm cột thức hai,....Tương tự, bốn byte đầu tiên của khóa mở rộng tạo thành một từ, chiếm cột đầu tiên của ma trận khoá mở rộng.

Quá trình xử lý mã hoá, giải mã bao gồm **N** vòng, tuỳ thuộc vào độ dài khoá tương ứng: 10 vòng cho khóa 16-byte, 12 vòng cho khóa 24-byte và 14 vòng cho khóa 32-byte được thể hiện trong bảng 1.1. N - 1 vòng đầu tiên bao gồm bốn hàm chuyển đổi riêng biệt:**SubBytes**,**ShiftRows**, **MixColumns** và **AddRoundKey** sẽ được mô tả cụ thể trong mục 1.3.2. Mỗi phép biến đổi tương ứng có một hoặc nhiều ma trận 4x4 làm đầu vào và tạo một ma trận 4x4 làm đầu ra. Chức năng mở rộng khoá tạo ra N+1 khoá vòng, mỗi khoá là một ma trận 4x4 riêng biệt. Mỗi khoá vòng tương ứng đóng vai trò là đầu vào trong bước biến đổi AddRoundKey trong mỗi vòng.

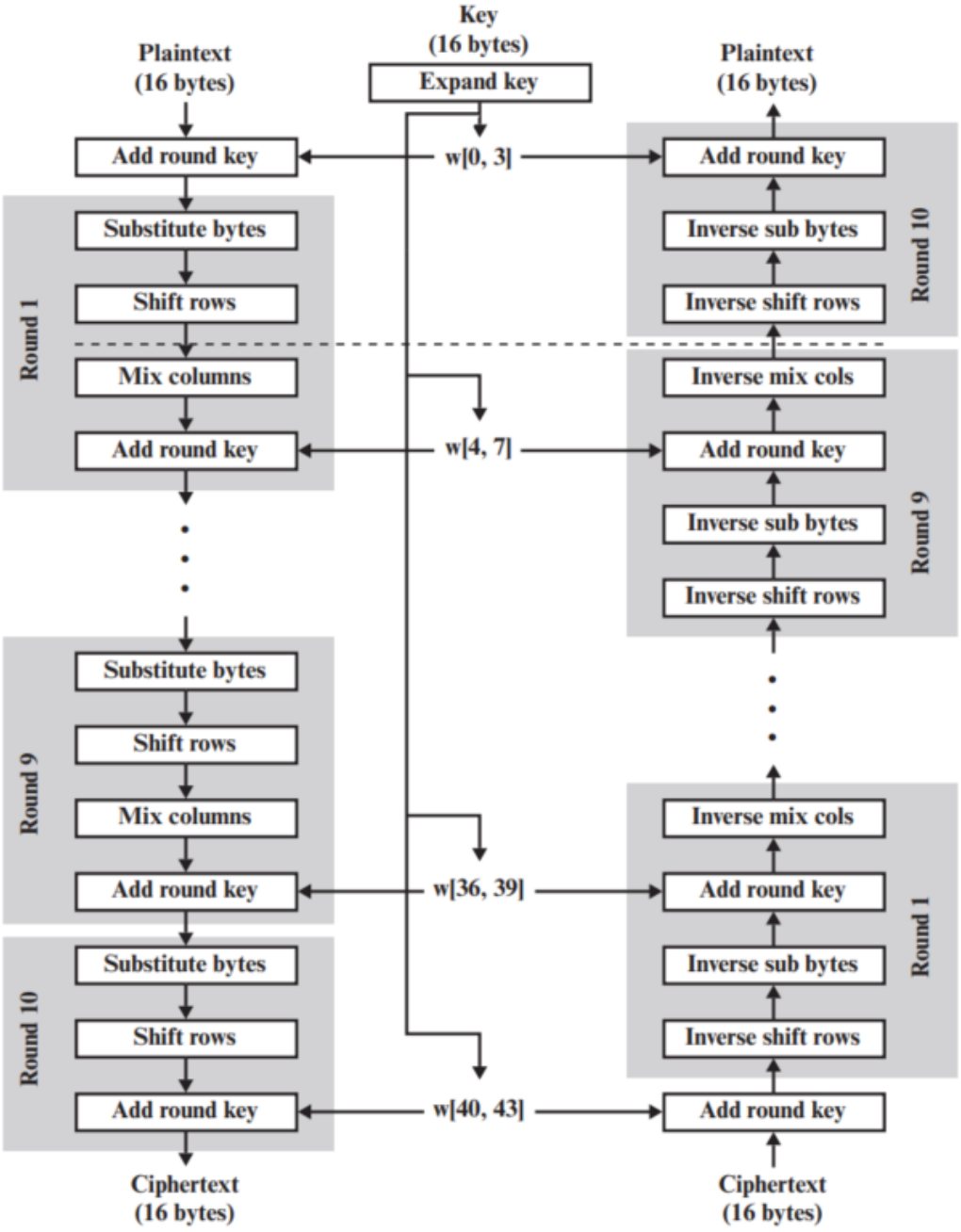
Kích thước khoá (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Kích thước bản rõ (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Số vòng xử lý (words/bytes/bits)	10	12	14
Kích thước khoá vòng (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Kích thước khoá mở rộng (words/bytes)	44/176	52/208	60/24

Bảng 1.1 Các thông số trong hệ mật mã AES



Hình 1.1 Cấu trúc tổng thể của hệ mật mã AES [1]

1.3.2 Cấu trúc chi tiết



Hình 1.2 Cấu trúc chi tiết của hệ mật mã AES [1]

1.4 Các hàm chuyển đổi

Trong phần này, em sẽ trình bày cụ thể các bước của các hàm chuyển đổi trong một vòng mã hoá/ giải mã bao gồm chức năng, tính chất của từng hàm chuyển đổi.

1.4.1 Substitue Bytes Tranformation

Biến đổi Byte thay thế thuận, gọi tắt là SubBytes, là một phép tra cứu bảng đơn giản. Chuẩn mã hoá AES định nghĩa một ma trận 16x16 của các giá trị byte, được gọi là bảng S-box, được mô tả như trong hình 1.3, chứa một hoán vị của tất cả 256 giá trị 8bit có thể. Mỗi byte trạng thái riêng lẻ được ánh xạ riêng biệt vào một byte trong bảng

với quy tắc sau: 4 bit ngoài cùng bên trái của byte được sử dụng làm giá trị hàng và 4 bit ngoài cùng bên phải được sử dụng làm giá trị cột. Ví dụ giá trị thập lục phân 8'h95 được tham chiếu đến hàng 9, cột 5 của bảng S-box, chứa giá trị 8'h2A. Vậy giá trị 8'h95 được ánh xạ và thay thế bằng giá trị 8'h2A.

Tương tự với biến đổi Byte thay thế nghịch, gọi tắt là Invert SubBytes, tương ứng có bảng Inv S-box như Hình 1.4

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Hình 1.3 Bảng Sbox [1]

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Hình 1.4 Bảng Inv Sbox [1]

Xét về mặt toán học, bảng S-box hay Inv S-box được xây dựng dựa trên biến đổi Affine có quy định rõ trong chuẩn mã hoá AES [2]. SubBytes() thực hiện tính toán theo thứ tự sau:

- Tính nghịch đảo nhân trong trường hữu hạn GF(2^8). Giá trị 0 có nghịch đảo là 0. Các giá trị khác phải biến đổi để tìm nghịch đảo.

- Thực hiện biến đổi Affine trên GF(2).

Trong hai bước tính, bước biến đổi Affine đã được quy định rõ trong chuẩn AES và thể hiện bằng phép nhân và cộng ma trận như sau:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Hình 1.5 Ma trận biến đổi Affine [1]

Trong đó, mỗi bit sẽ được tính theo công thức (1.1) với giá trị c_i tương ứng là giá trị thứ i của 8'h05 hay $(c_7c_6c_5c_4c_3c_2c_1c_0)$ = (00000101). Trong đó, y_i là bit thứ i của byte cần chuyển đổi, x_i là bit thứ i của byte kết quả sau chuyển đổi:

$$y_i = x_i \oplus x_{(i+4)mod8} \oplus x_{(i+4)mod8} \oplus x_{(i+4)mod8} \oplus x_{(i+4)mod8} \oplus ci \qquad (1.1)$$

Triển khai từ hình 1.5 và công thức (1.1) ta có biểu thức tính giá trị từng bit trong biến đổi Affine như sau:

$$\begin{aligned} y_0 &= x_0 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus 1 \\ y_1 &= x_0 \oplus x_1 \oplus x_5 \oplus x_6 \oplus x_7 \oplus 1 \\ y_2 &= x_0 \oplus x_1 \oplus x_2 \oplus x_6 \oplus x_7 \\ y_3 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_7 \\ y_4 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \\ y_5 &= x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_5 \oplus 1 \\ y_6 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus 1 \\ y_7 &= x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \end{aligned}$$

Ngược lại phép biển đổi InvSubBytes() thực hiện tính toán theo thứ tự sau:

- Thực hiện biến đổi Affine nghịch trên GF(2).

- Tính nghịch đảo nhân trong trường hữu hạn GF(2^8). Giá trị 0 có nghịch đảo là 0. Các giá trị khác phải biến đổi để tìm nghịch đảo.

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Hình 1.6 Ma trận biến đổi Affine nghịch [1]

Trong đó, mỗi bit sẽ được tính theo công thức (1.2) với giá trị c_i tương ứng là giá trị thứ i của 8'h63 hay $(d_7d_6d_5d_4d_3d_2d_1d_0)$ = (01100011) Trong đó, x_i là bit thứ i của byte cần chuyển đổi, y_i là bit thứ i của byte kết quả sau chuyển đổi:

$$y_i = x_i \oplus x_{(i+2)mod8} \oplus x_{(i+5)mod8} \oplus x_{(i+7)mod8} \oplus di \tag{1.2}$$

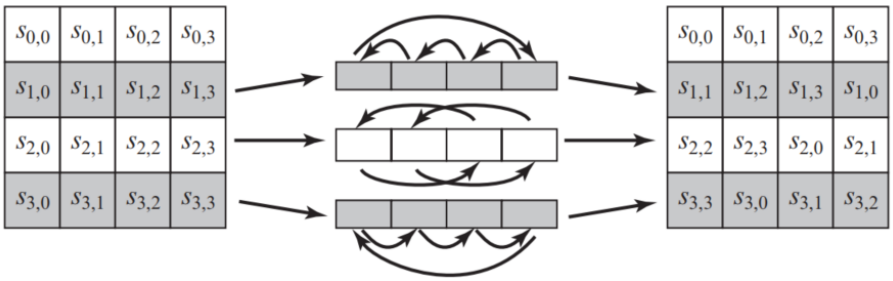
Triển khai từ ma trận 1.6 và công thức (1.2) ta có:

$$\begin{aligned} y_0 &= x_2 \oplus x_5 \oplus x_7 \oplus 1 \\ y_1 &= x_0 \oplus x_3 \oplus x_6 \oplus x_6 \\ y_2 &= x_1 \oplus x_4 \oplus x_7 \oplus 1 \\ y_3 &= x_0 \oplus x_2 \oplus x_5 \\ y_4 &= x_1 \oplus x_3 \oplus x_6 \\ y_5 &= x_2 \oplus x_4 \oplus x_7 \\ y_6 &= x_0 \oplus x_3 \oplus x_5 \\ y_7 &= x_1 \oplus x_4 \oplus x_6 \end{aligned}$$

1.4.2 ShiftRows Tranformation

Sự biến đổi dịch chuyển hàng, được gọi là **ShiftRows**. Đối với mỗi ma trận trạng thái 4x4, hàng đầu tiên không bị thay đổi. Hàng thứ hai dịch vòng trái 1 byte, hàng thứ 3 dịch vòng trái 2 bytes, hàng cuối cùng dịch vòng trái 3 bytes. Quy trình về phép ShiftRows được mô tả trong hình 1.7.

Tương ứng với thuật toán giải mã, phép biến đổi **InvShiftRows** cũng thay đổi với quy tắc tương tự với ShiftRows nhưng dịch phải các hàng tương ứng.

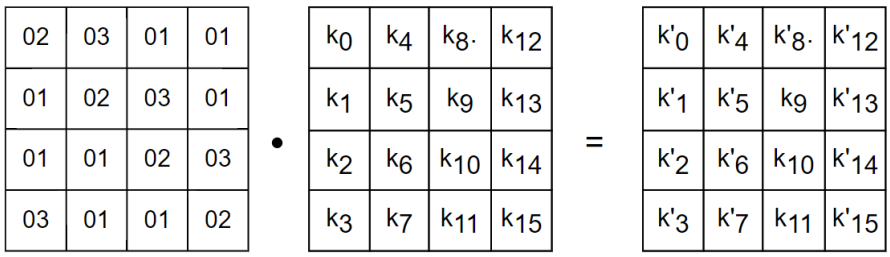


Hình 1.7 Các bước trong phép biến đổi ShiftRows

1.4.3 MixColumns Tranformation

Phép biến đổi trộn cột, được gọi là MixColumns, hoạt động trên từng cột riêng lẻ. Mỗi byte của cột được ánh xạ thành một giá trị mới là hàm của cả bốn byte trong cột đó. Phép nhân từng cột của ma trận trạng thái với một ma trận tiêu chuẩn được quy định bởi chuẩn AES như sau:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$



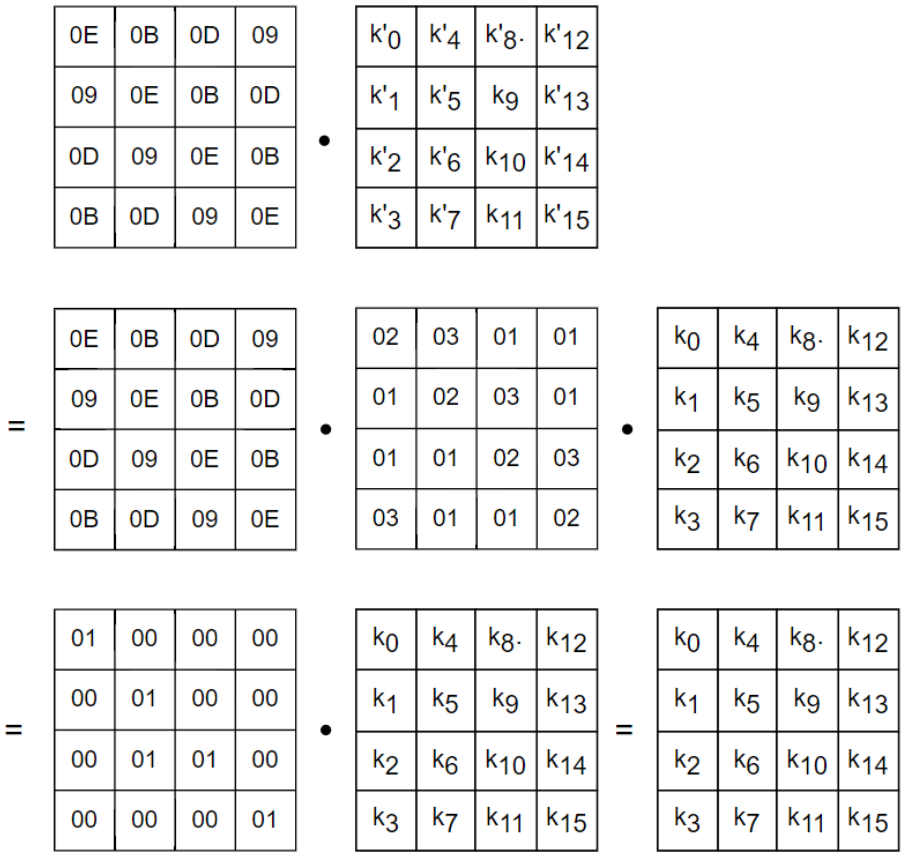
Hình 1.8 Phép biến đổi MixColumns

Tương tự như MixColumns, phép biến đổi ngược, gọi là InvMixColumns bằng phép nhân ngược lại kết quả ở hình 1.8 với ma trận sau

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

Phép chứng minh được thể hiện trong hình 1.9, thể hiện việc biến đổi ngược từ ma

trận kết quả ở hình 1.8 trở về ma trận trạng thái ban đầu. Kết quả là phép nhân ma trận đơn vị với ma trận trạng thái ban đầu nên kết quả được dễ dàng chứng minh.



Hình 1.9 Chứng minh ma trận InvMixColumns

1.5 Thuật toán mở rộng khoá

Thuật toán mở rộng khóa AES nhận một khóa bốn từ (word) tương đương (16 byte) làm đầu vào và tạo ra một mảng tuyến tính của 44 words (176 byte). Điều này đủ để cung cấp một khóa vòng bốn từ cho giai đoạn AddRoundKey ban đầu và mỗi trong 10 vòng của mã hóa. Thuật toán được mô tả dưới dạng Pseudocode như sau:

```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++) w[i] = (key[4*i], key[4*i+1],
                                    key[4*i+2],
                                    key[4*i+3]);
    for (i = 4; i < 44; i++)
    {
    temp = w[i - 1];
    if (i mod 4 = 0) temp = SubWord (RotWord (temp))
                              ⊕ Rcon[i/4];
    w[i] = w[i-4] ⊕ temp
    }
}
```

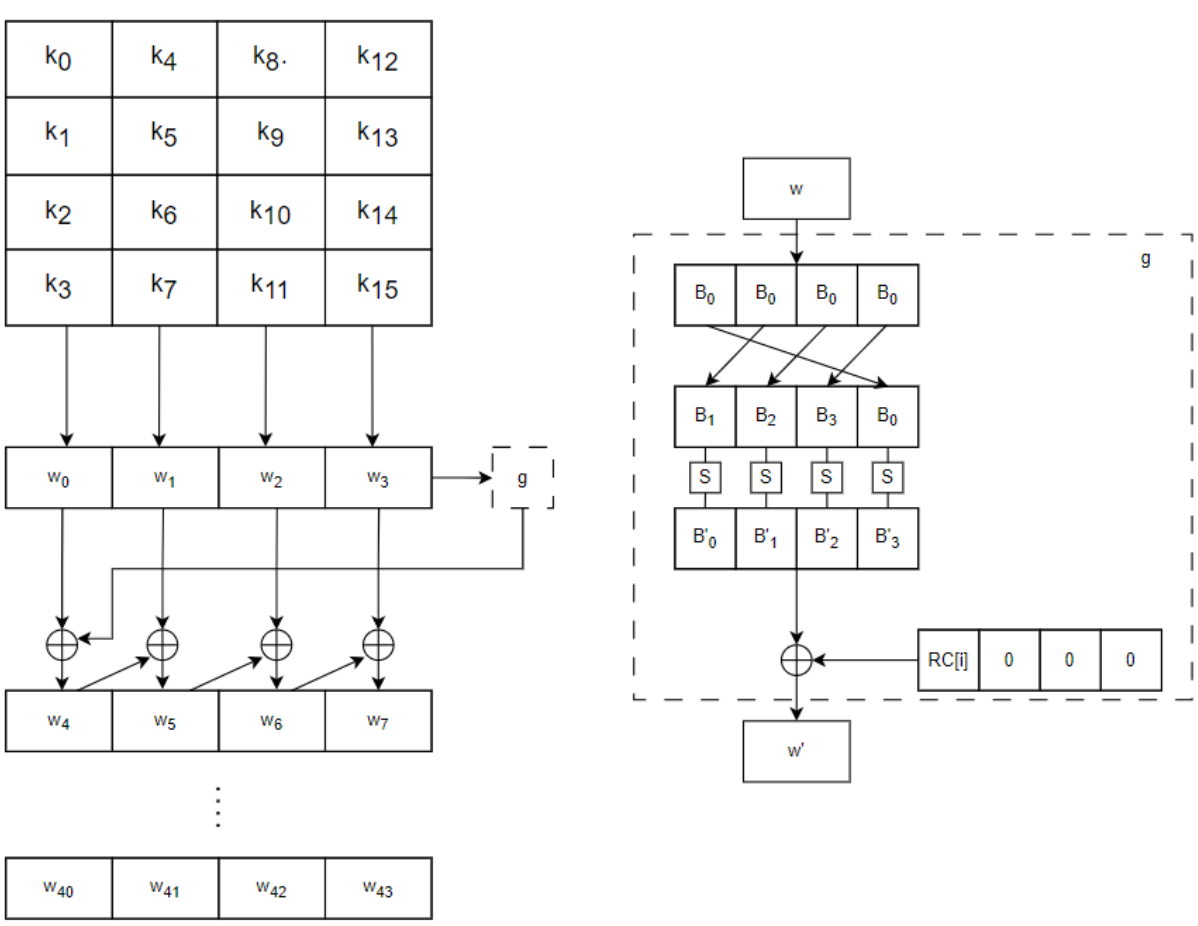
Hình 1.10 Pseudocode chức năng mở rộng khoá

Khóa được sao chép vào bốn từ đầu tiên của khóa mở rộng. Phần còn lại của khóa mở rộng được điền vào bốn từ một lần. Mỗi từ được thêm vào w[i] phụ thuộc vào từ trước đó, w[i - 1], và từ bốn vị trí trở lại, w[i - 4]. Trong ba trong bốn trường hợp, sử dụng một XOR đơn giản. Cho một từ có vị trí trong mảng w là bội số của 4, sử dụng một hàm phức tạp hơn. Hình 1.11 minh họa việc tạo ra khóa mở rộng, sử dụng ký hiệu g để biểu thị hàm phức tạp đó. Hàm g bao gồm các hàm con sau :

- **RotWord**: thực hiện dịch vòng trái 1 đơn vị trên một word.
- **SubWord**: Thực hiện thay thế byte trên mỗi byte trong word tương ứng bằng bảng S-box.
- **Rcon**: Kết quả các bước trên được thực hiện phép XOR với một hằng số trong chuỗi hằng số Rcon[i] = $(RC[i], 0, 0, 0)$. Chuỗi hằng số này được quy định trong chuẩn AES với ba byte bên phải luôn là 0, giá trị cụ thể được trình bày trong bảng 2.2.

Vòng mã hoá	1	2	3	4	5	6	7	8	9	10
Giá trị RC[i]	01	02	04	08	10	20	40	80	1B	36

Bảng 1.2 Bảng giá trị hằng số RC



Hình 1.11 Sơ đồ mô tả thuật toán mở rộng khoá

1.6 Các chế độ mã hoá và giải mã

1.6.1 Tổng quan

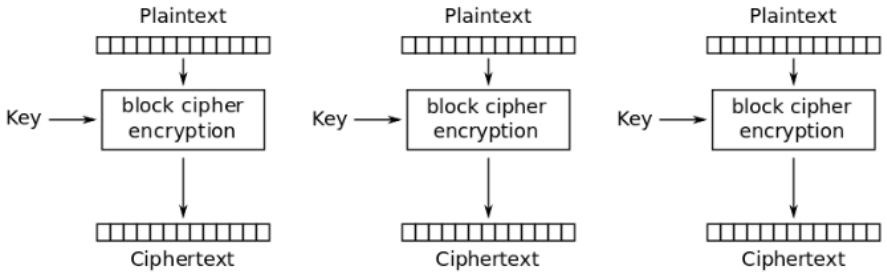
Các thuật toán mã hóa như mã hóa khối (block cipher algorithm) cung cấp cơ chế chuyển đổi thuận nghịch giữa giá trị thực được dùng bởi hệ thống, gọi là bản rõ (plaintext), và giá trị mã hóa, gọi là bản mã (ciphertext). Mã hóa khối (block cipher) là mã hóa thực thi trên từng khối bit có kích thước cố định. Ví dụ, DES mã hóa trên 64 bit dữ liệu đầu vào, AES mã hóa trên 128, 192 hoặc 256 bit dữ liệu đầu vào. Mã hóa khối khác với mã hóa dòng (stream cihper), hay còn gọi là mã hóa luồng. Mã hóa dòng xử lý trên từng bit dữ liệu đầu vào. Mã hóa khóa mã đối xứng (symmetric key) là thuật toán dùng khóa mã hóa và khóa giải mã có mối liên hệ với nhau, có thể dùng khóa này biến đổi thành khóa kia và ngược lại, hoặc hai khóa mã hoàn toàn giống nhau. Trong khi vai trò chính của một thuật toán mã hóa là sử dụng khóa mã bảo mật (secure/secret key) kết hợp với bản rõ để tạo ra bản mã, hoặc ngược lại, kết hợp với bản mã để khôi phục lại bản rõ thì vai trò của "chế độ mã hóa" là quy định cách thức sử dụng của một thuật toán mã hóa. Cùng một thuật toán mã hóa, cách thức sử dụng khác nhau sẽ tạo ra các kết quả mã hóa dữ liệu khác nhau với mức độ bảo mật khác nhau.

Tóm lại, quá trình mã hóa dữ liệu gồm 2 phần quan trọng là:

- Thuật toán mã hóa (cipher algorithm).
- Chế độ mã hóa (cipher mode).

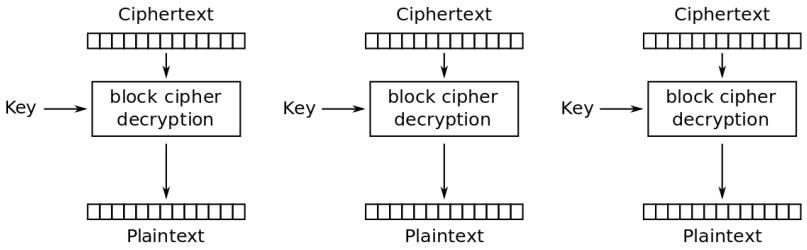
1.6.2 ECB (Electronic Codebook)

ECB là chế độ mã hóa từng khối bit độc lập. Với cùng một khóa mã K, mỗi khối plaintext ứng với một giá trị ciphertext cố định và ngược lại.



Electronic Codebook (ECB) mode encryption [3]

Hình 1.12 Chế độ mã hoá ECB



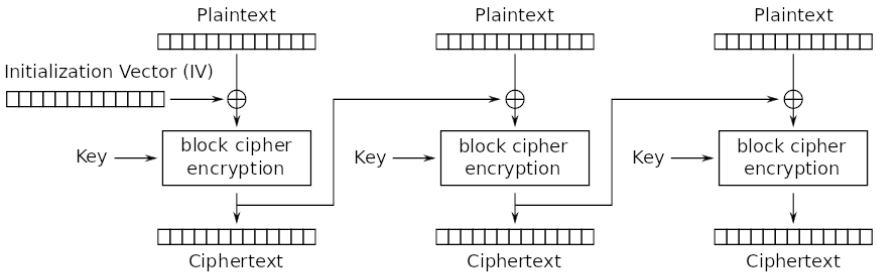
Electronic Codebook (ECB) mode decryption [3]

Hình 1.13 Chế độ giải mã ECB

Với chế độ mã hoá/ giải mã này, thiết kế phần cứng đơn giản. Vấn đề cần quan tâm chính là thiết kế logic cho thuật toán mã hóa. Lỗi bit không bị lan truyền. Nếu lỗi bit xuất hiện trên một ciphertext của một khối dữ liệu thì nó chỉ ảnh hưởng đến việc giải mã khối dữ liệu đó chứ không ảnh hưởng đến việc giải mã khác khối dữ liệu khác. Có thể thực hiện mã hóa/giải mã song song nhiều khối dữ liệu cùng lúc. Điều này giúp tăng tốc độ xử lý trong các hệ thống đòi hỏi mã hóa/giải mã tốc độ cao. Tuy vậy, do thiết kế đơn giản nên khả năng bảo mật kém, do giá trị plaintext và ciphertext được ánh xạ độc lập một-một nên thông tin mã hóa dễ bị sửa đổi bằng cách như xóa bớt khối dữ liệu, chèn thêm khối dữ liệu, hoán đổi vị trí khối dữ liệu để làm sai lệch thông tin tại nơi nhận.

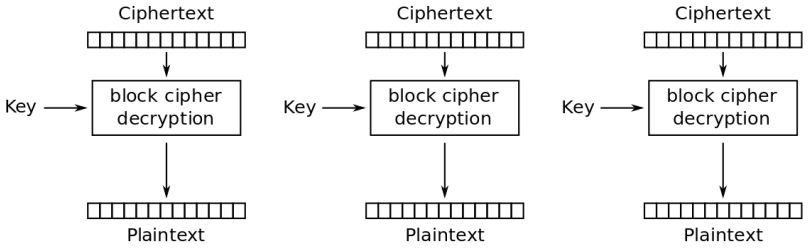
1.6.3 CBC (Cipher Block Chaining)

CBC là chế độ mã hóa chuỗi, kết quả mã hóa của khối dữ liệu trước (ciphertext) sẽ được tổ hợp với khối dữ liệu kế tiếp (plaintext) trước khi thực thi mã hóa.



Cipher Block Chaining (CBC) mode encryption [3]

Hình 1.14 Chế độ mã hoá CBC



Electronic Codebook (ECB) mode decryption [3]

Hình 1.15 Chế độ giải mã ECB

Ưu điểm của chế độ này là khả năng bảo mật cao hơn ECB. Ciphertext của một khối dữ liệu plaintext có thể khác nhau cho mỗi lần mã hóa vì nó phụ thuộc vào IV hoặc giá trị mã hóa (ciphertext) của khối dữ liệu liền trước. Quá trình giải mã (mã hóa nghịch) vẫn có thể thực hiện song song nhiều khối dữ liệu.

Do có sự tác động của Vector Init vào quá trình xử lý nên thiết kế phần cứng phức tạp hơn ECB, ngoài logic thực thi thuật toán mã hóa, người thiết kế cần thiết kế thêm:

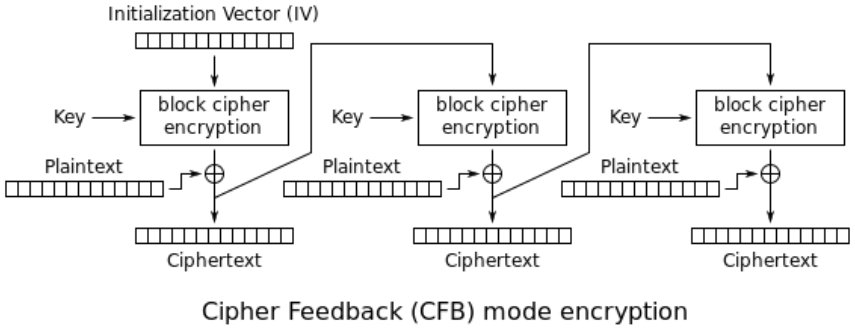
- Logic quản lý độ dài chuỗi dữ liệu sẽ được mã hóa, cụ thể là số lượng khối dữ liệu trong chuỗi dữ liệu.
- Bộ tạo giá trị ngẫu nhiên cho IV (Vector khởi tạo).

Hạn chế của chế độ mã hoá này là lỗi bit bị lan truyền. Nếu một lỗi bit xuất hiện trên ciphertext của một khối dữ liệu thì nó sẽ làm sai kết quả giải mã của khối dữ liệu đó và khối dữ liệu tiếp theo. Không thể thực thi quá trình mã hóa song song vì xử lý của khối dữ liệu sau phụ thuộc vào ciphertext của khối dữ liệu trước, trừ lần mã hóa đầu tiên.

1.6.4 CFB (Cipher Feedback)

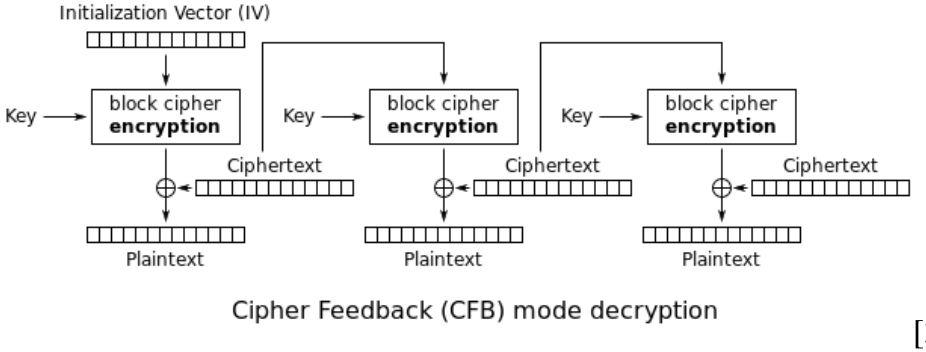
CFB là chế độ mã hóa mà ciphertext của lần mã hóa hiện tại sẽ được phản hồi (feedback) đến đầu vào của lần mã hóa tiếp theo. Nghĩa là, ciphertext của lần mã hóa

hiện tại sẽ được sử dụng để tính toán ciphertext của lần mã hóa kế tiếp. Mô tả có vẻ giống CBC nhưng quá trình trực hiện lại khác.



[3]

Hình 1.16 Chế độ mã hoá CFB



[3]

Hình 1.17 Chế độ giải mã ECB

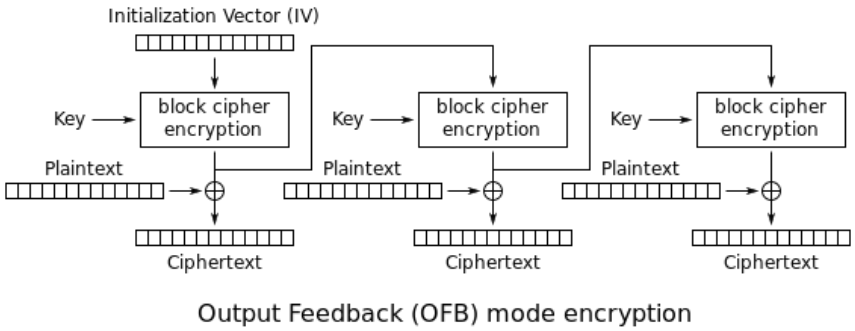
So sánh với chế độ CBC, chế độ CFB có những điểm khác biệt thuật toán mã hóa không áp dụng trực tiếp trên plaintext mà dùng để biển đổi một khối dữ liệu sinh ra từ IV và ciphertext. Do thiết kế phức tạp hơn các chế độ đã nêu ở trên, CFB có khả năng bảo mật cao hơn ECB. Ciphertext của một khối dữ liệu plaintext có thể khác nhau cho mỗi lần mã hóa vì nó phụ thuộc vào Vector Init hoặc giá trị mã hóa (ciphertext) của khối dữ liệu liền trước. Quá trình giải mã (mã hóa nghịch) vẫn có thể thực hiện song song nhiều khối dữ liệu. Ngoài ra còn một số hạn chế như thiết kế phần cứng phức tạp, lỗi bit lan truyền, không thể thực thi quá trình mã hoá song song.

1.6.5 OFB (Output Feedback)

OFB là chế độ mã hóa mà giá trị ngõ ra của khối thực thi thuật toán mã hóa, không phải ciphertext, của lần mã hóa hiện tại sẽ được phản hồi (feedback) đến ngõ vào của lần mã hóa kế tiếp. Thuật toán mã hóa không áp dụng trực tiếp trên plaintext mà dùng để biển đổi một khối dữ liệu sinh ra từ IV và khối ngõ ra của lần mã hóa trước đó. Điểm này tương tự với CFB.

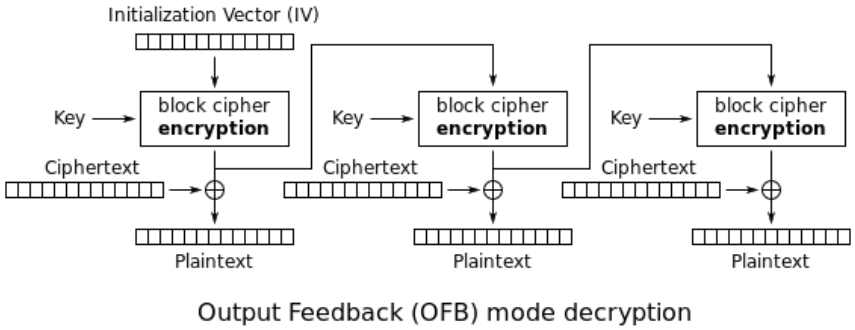
Khả năng bảo mật cao hơn ECB. Ciphertext của một khối dữ liệu plaintext có thể

khác nhau cho mỗi lần mã hóa vì nó phụ thuộc vào IV hoặc khối ngõ ra của lần mã hóa trước đó. Lỗi bit không bị lan truyền. Khi một lỗi bit xuất hiện trên một ciphertext, nó chỉ ảnh hưởng đến kết quả giải mã của khối dữ liệu hiện tại Thiết kế phần cứng đơn giản hơn CFB. Tuy vậy, chế độ mã hoá này cũng không thể mã hoá/ giải mã song song nhiều khối dữ liệu.



[3]

Hình 1.18 Chế độ mã hoá OFB



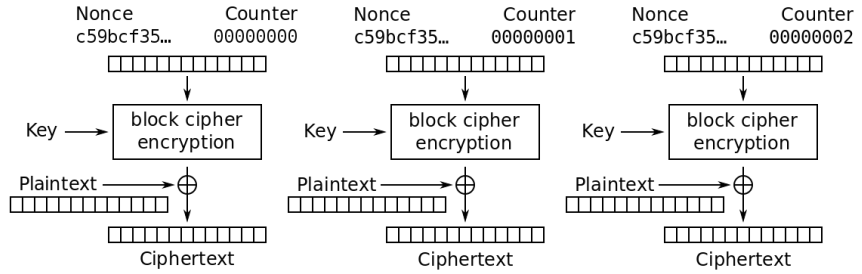
[3]

Hình 1.19 Chế độ giải mã ECB

1.6.6 CTR (Counter)

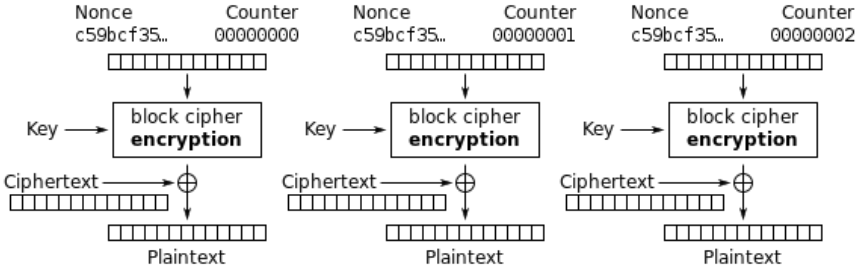
CTR là chế độ mã hóa sử dụng một tập các khối ngõ vào, gọi là các counter, để sinh ra một tập các giá trị ngõ ra thông qua một thuật toán mã hóa. Sau đó, giá trị ngõ ra sẽ được XOR với plaintext để tạo ra ciphertext trong quá trình mã hóa, hoặc XOR với ciphertext để tạo ra plaintext trong quá trình giải mã. Khả năng bảo mật cao hơn ECB. Tuy quá trình mã hóa/giải mã của mỗi khối dữ liệu là độc lập nhưng mỗi plaintext có thể ảnh xạ đến nhiều ciphertext tùy vào giá trị bộ đếm của các lần mã hóa. Có thể mã hóa/giải mã song song nhiều khối dữ liệu.

Nhược điểm của chế độ này là phần cứng cần thiết kế thêm các bộ đếm counter hoặc giải thuật tạo các giá trị counter không lặp lại.



[3]

Hình 1.20 Chế độ mã hoá CTR



[3]

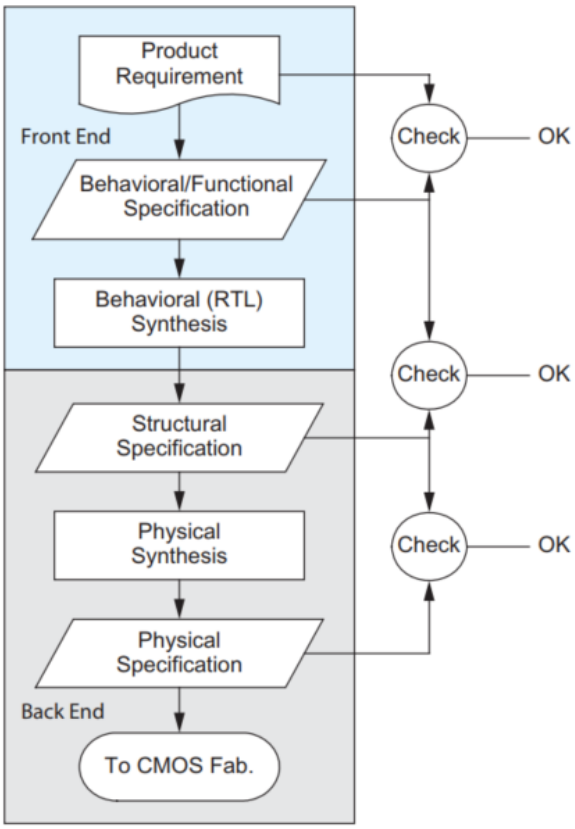
Hình 1.21 Chế độ giải mã ECB

1.7 Quy trình thiết kế vi mạch số và Ngôn ngữ mô tả phần cứng

1.7.1 Quy trình thiết kế vi mạch số

Quy trình thiết kế vi mạch bao gồm tập hợp các bước cho phép người thiết kế phát triển sản phẩm, bắt đầu từ thông số, yêu cầu kỹ thuật (Specifications) cho đến bước cuối cùng là sản xuất ra chip bán dẫn. Các bước cơ bản được mô tả như hình 1.22. Bắt đầu từ yêu cầu thiết kế sản phẩm (Product Requirement), kĩ sư vi mạch sẽ thiết kế ở mức độ hành vi (Behavioral Specification) hay các bản mô tả chức năng, sau đó sẽ mô tả chi tiết về mặt cấu trúc (logic gate/ register). Bước tiếp theo trong quy trình là sử dụng các ngôn ngữ mô tả phần cứng HDL và kết hợp với các design tools, design compiler để chuyển toàn bộ thiết kế về mức độ gate netlist. Quá trình này bao gồm các bước như lập trình (RTL code), mô phỏng (Simulation), kiểm thử (Verification), tổng hợp (Synthesis) song song với quá trình kiểm tra nghiêm ngặt để xác định rằng thiết kế đã đáp ứng đầy đủ về mặt chức năng, thời gian, hiệu suất.

Bước tiếp theo trong quy trình sản xuất vi mạch số là Physical Design hay thiết kế về mặt vật lý. Sau khi đã thu được mô hình logic từ các bước trên (Front-end Design), Physical Design thực hiện chuyển đổi các mô hình logic thành một bản thiết kế vật lý có thể sản xuất được trên các loại vật liệu bán dẫn. Về cơ bản, các bước chính trong quy trình Physical Design bao gồm: Placement, Floorplanning, Routing, DRC, LVS, STA...



[4]

Hình 1.22 Quy trình thiết kế vi mạch

1.7.2 Ngôn ngữ mô tả phần cứng

Ngôn ngữ mô tả phần cứng được sử dụng trong quy trình thiết kế vi mạch để mô tả hành vi của phần cứng. Trong lịch sử phát triển của ngành vi mạch, đã có rất nhiều loại ngôn ngữ mô tả phần cứng được sử dụng phổ biến như VHDL, Verilog, SystemVerilog.

SystemVerilog được bắt nguồn từ Verilog, kế thừa và bổ sung nhiều chức năng để đáp ứng nhu cầu ngày càng gia tăng về độ phức tạp cũng như yêu cầu về sự chính xác trong quá trình sản xuất vi mạch. SystemVerilog có rất nhiều đặc điểm quan trọng, vượt trội hơn Verilog như:

- Cung cấp nhiều kiểu dữ liệu mới, liên quan đến các kiểu dữ liệu quen thuộc từ các ngôn ngữ lập trình như: queue, mảng động (dynamic array), mảng kết hợp (associative array), ... [5]
- Hỗ trợ cấu trúc lập trình hướng đối tượng OOP: SystemVerilog cung cấp các kiểu cấu trúc như class, object và các tính chất của lập trình hướng đối tượng như inheritance (tính kế thừa), polymorphism (tính đa hình), encapsulation (tính đóng gói), abstract (tính trừu tượng). Các cấu trúc lập trình này được sử dụng để tăng tính trừu tượng, tối ưu cho môi trường kiểm thử. [5]

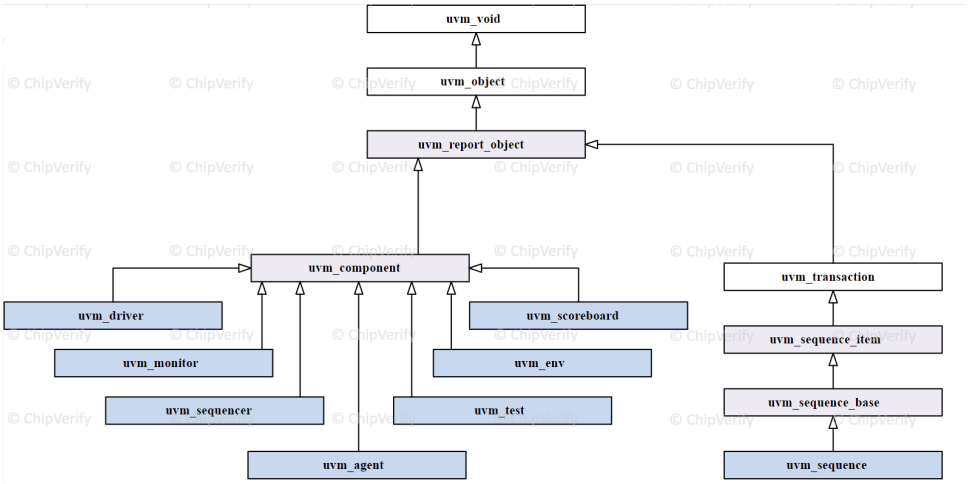
- Cung cấp các cấu trúc mới để tối ưu trong quá trình kiểm tra lỗi, tính đúng đắn của thiết kế về mặt chức năng như: assertion, coverage, ... [5]

Với những ưu điểm trên, em quyết định sẽ chọn SystemVerilog là ngôn ngữ mô tả phần cứng chính, dùng để lập trình và phát triển khối IP.

1.8 Phương pháp kiểm thử thiết kế số - Universal Verification Methodlogy

1.8.1 Giới thiệu về UVM

Universal Verification Methodlogy- UVM, được phát triển bởi tổ chức Accellera Systems Initiative, một tổ chức cung cấp các giao diện tiêu chuẩn trong lĩnh vực tự động hóa thiết kế điện tử và thiết kế và sản xuất mạch tích hợp. UVM là phương pháp kiểm thử được sử dụng như một quy chuẩn trong quy trình thiết kế và sản xuất vi mạch số. UVM cung cấp một môi trường kiểm thử đồng nhất và có tính chất tái sử dụng. UVM sẽ giúp tối ưu quá trình kiểm thử thiết kế số về mặt thời gian và có thể sử dụng lại trong các dự án khác nhau do sử dụng cấu trúc lập trình hướng đối tượng như đã nói ở mục 1.4.1. Cấu trúc phân cấp các lớp của thư viện UVM như hình 1.23:



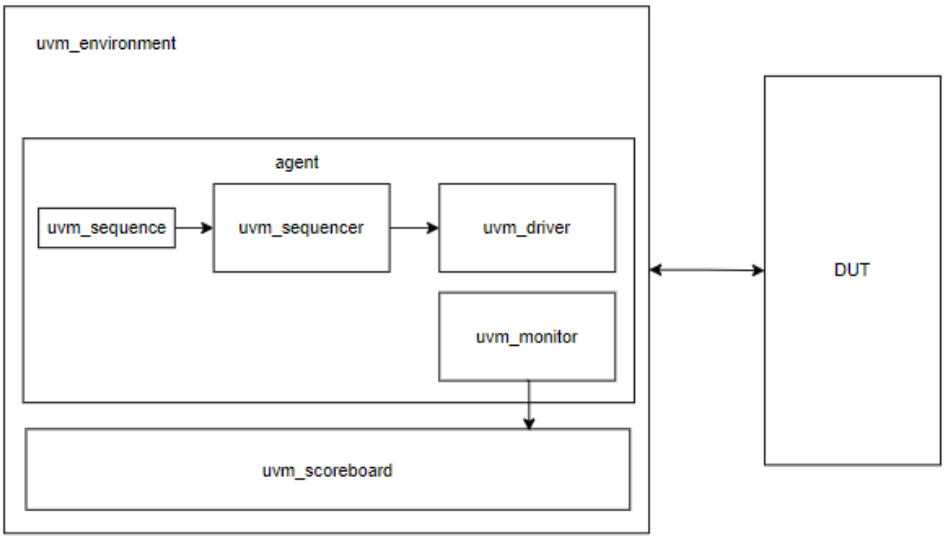
[6]

Hình 1.23 Cấu trúc phân cấp các lớp trong UVM

UVM được xây dựng dựa trên ngôn ngữ SystemVerilog, cung cấp một môi trường mang tính trừu tượng hoá gồm các thành phần (component) với các chức năng khác nhau. Trong một môi trường kiểm thử UVM tiêu chuẩn, bao gồm có các thành phần chính như hình 1.3 mô tả:

- **uvm_environment** : Môi trường kiểm thử chính, có vai trò tương tác trực tiếp với DUT như gửi, nhận và phân tích và đánh giá dữ liệu trả về từ DUT.
- **uvm_driver**: Thành phần con chứa trong uvm_environment, có nhiệm vụ lấy dữ liệu (kích thích) từ môi trường kiểm thử và đưa tới DUT.

- **uvm_sequence**: Tạo ra các kích thích một cách tự động dưới dạng các sequence-item để đưa tới uvm_driver.
- **uvm_sequencer**: Khối trung gian, điều khiển quá trình giao tiếp giữa sequence và driver.
- **uvm_monitor**: Nhận dữ liệu trả về từ DUT, phân tích và đưa lên khối uvm_scoreboard để đánh giá.
- **uvm_scoreboard**: Nhận dữ liệu từ monitor, đánh giá và phân tích tính chính xác của dữ liệu trả về từ thiết kế và hiển thị kết quả cho kĩ sư kiểm thử.



Hình 1.24 Cấu trúc cơ bản của môi trường kiểm thử UVM

1.9 Kết luận chương 1

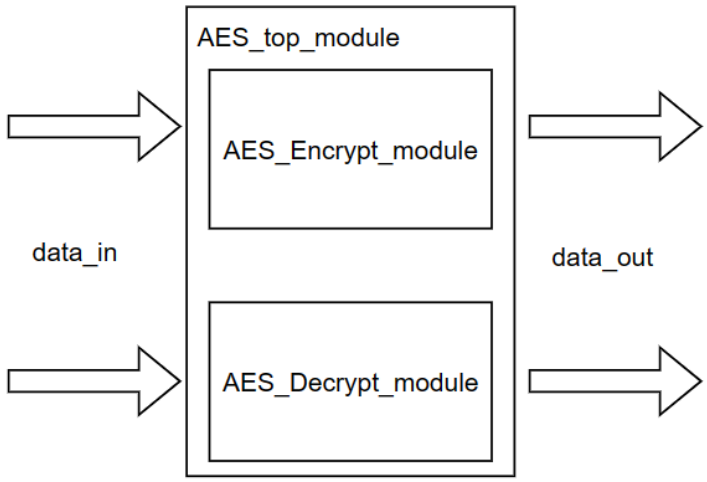
Trong phạm vi chương này, em đã giới thiệu các kiến thức tổng quát như cấu trúc, thuật toán mã hoá, giải mã của hệ mật AES, quy trình tổng quát thiết kế vi mạch số, ngôn ngữ mô tả phần cứng, phương pháp kiểm thử thiết kế UVM. Từ các kiến thức nền tảng này sẽ làm tiền đề cho quá trình thiết kế cụ thể ở các chương sau.

CHƯƠNG 2: THIẾT KẾ KIẾN TRÚC KHỐI XỬ LÝ MẬT MÃ

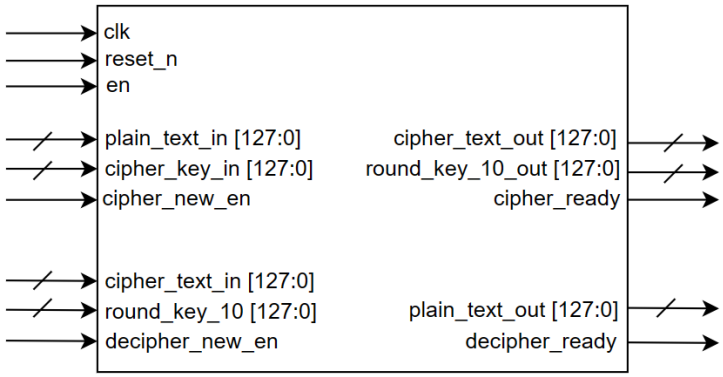
2.1 Thiết kế yêu cầu kĩ thuật

2.1.1 Sơ đồ khối tổng quát

Dựa trên kiến thức đã tìm hiểu ở chương 1, trong chương 2, em sẽ bắt đầu thiết kế yêu cầu kĩ thuật, chi tiết kiến trúc logic bên trong khối xử lý. Module được xây dựng bao gồm có 2 chức năng tương ứng với các sub-module như trong 1.6. Chức năng chính bao gồm mã hoá và giải mã với các chân tín hiệu vào, ra riêng biệt được trình bày cụ thể trong hình 1.5 và bảng 2.1 thông số chân tín hiệu.



Hình 2.1 Dataflow của thiết kế



Hình 2.2 Chân tín hiệu khối top module

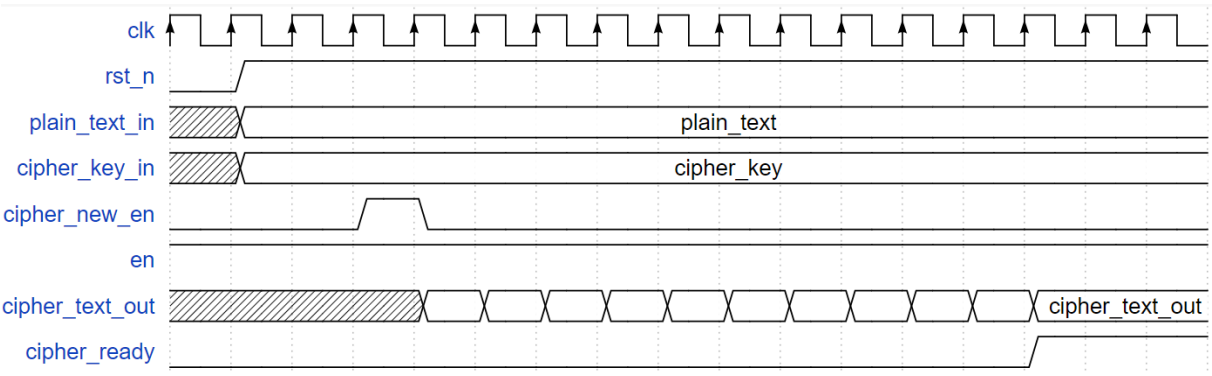
Tên tín hiệu	Số bit	Hướng tín hiệu	Mô tả
clk	1	Input	Xung nhịp điều khiển
reset_n	1	Input	Tín hiệu reset tích cực mức thấp
plain_text_in	128	Input	Bản rõ đầu vào
cipher_key_in	128	Input	Khoá đầu vào
cipher_new_en	1	Input	Tín hiệu bắt đầu quá trình mã hoá
cipher_text_in	128	Input	Bản mã đầu vào
round_key_10	128	Input	Khoá vòng thứ 10 đầu vào
decipher_new_en	1	Input	Tín hiệu bắt đầu quá trình giải mã
en	1	Input	Tín hiệu điều khiển quá trình mã hoá/ giải mã
cipher_text_out	128	Output	Bản mã đầu ra
round_key_10_out	128	Output	Khoá vòng thứ 10 đầu ra
cipher_ready	1	Output	Cờ báo hiệu quá trình mã hoá hoàn thành
plain_text_out	128	Output	Bản rõ đầu ra
decipher_ready	1	Output	Cờ báo hiệu quá trình giải mã hoàn thành

Bảng 2.1 Bảng mô tả các chân tín hiệu khối AES_top

2.1.2 Mô tả chức năng

Khi có tín hiệu reset_n mức thấp, tất cả các tín hiệu được đưa về 0. Hoạt động của khối xử lý được chia thành 2 chức năng riêng biệt là mã hoá và giải mã.

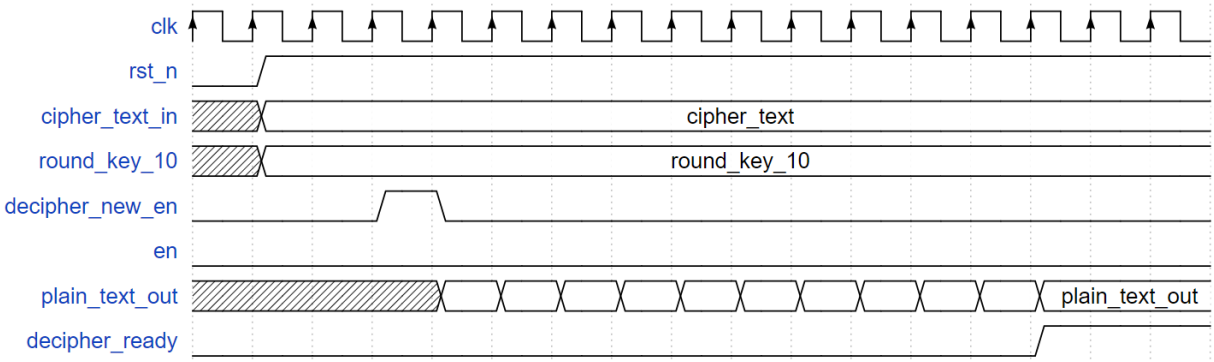
Đối với quá trình mã hoá, khi tín hiệu cipher_new_en tích cực, tín hiệu en lên mức 1 để thực hiện chức năng mã hoá. Với đầu vào là dữ liệu tại các chân plain_text_in và cipher_key_in. Sau khi quá trình mã hoá kết thúc, tín hiệu cipher_ready tích cực mức cao. Đầu ra thu được tại các chân tín hiệu cipher_text_out và round_key_10. Sơ đồ timing được mô tả như hình 2.3.



Hình 2.3 Sơ đồ timing chức năng mã hoá

Ngược lại, đối với quá trình giải mã, khi tín hiệu decipher_new_en tích cực, tín hiệu en lên mức 1 để thực hiện chức năng mã hoá. Với đầu vào là dữ liệu tại các

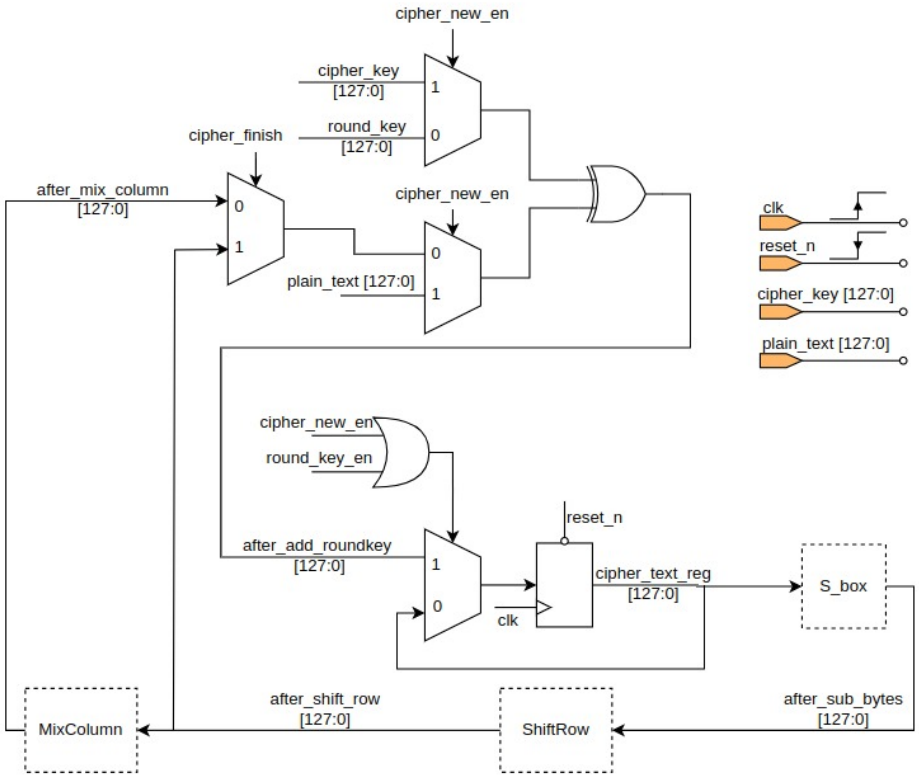
chân cipher_text_in và round_key_10_in. Sau khi quá trình mã hoá kết thúc, tín hiệu decipher_ready tích cực mức cao. Đầu ra thu được tại các chân tín hiệu plain_text_out. Sơ đồ timing được mô tả như hình 2.4.



Hình 2.4 Sơ đồ timing chức năng giải mã

2.2 Kiến trúc chi tiết

2.2.1 Khối xử lý mã hoá

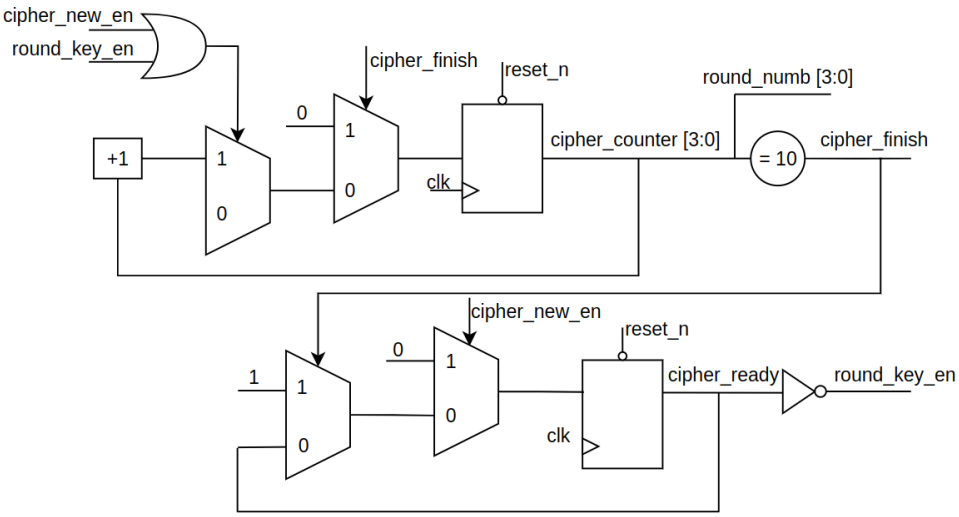


Hình 2.5 Kiến trúc khối mã hoá

Hình 2.5 mô tả logic xử lý chính của khối xử lý mã hoá. Khi tín hiệu cipher_new_en tích cực, mạch bắt đầu quá trình mã hoá. Mạch logic bộ mã hóa AES-128 gồm 1 thanh ghi cipherText_reg[127:0] lưu lại giá trị sau mỗi vòng lặp mã hóa, thanh ghi này đặt sau chức năng AddRoundKey. Bắt đầu từ tín hiệu after_addRoundkey[127:0], tín hiệu này

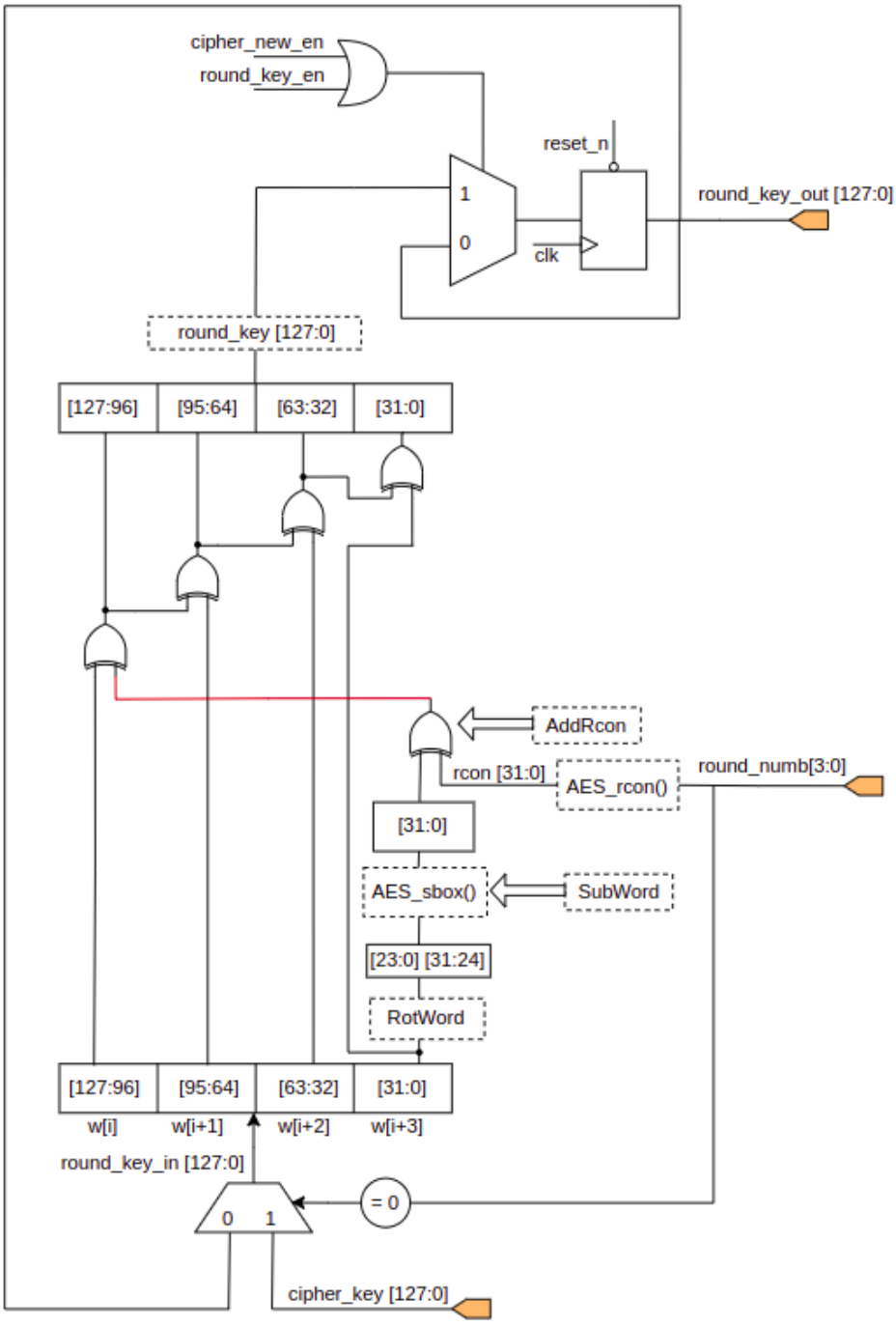
được tạo ra từ 1 cổng XOR, đây là chức năng AddRoundKey, với các ngõ vào được lấy từ:

- Khóa mã, cipher_key[127:0], và bản rõ, plain_text[127:0] cho lần tính đầu tiên khi cipher_new_en =1.
- Khóa vòng, round_key[127:0], và bản mã trung gian cho các lần tính tiếp theo, sau lần đầu tiên. Trong đó, bản mã trung gian lấy từ:
  - Ngõ ra chức năng MixColumns nếu đây là lần lặp từ 1 đến 9.
  - Ngõ vào chức năng MixColumns, ứng với ngõ ra chức năng ShiftRows, nếu đây là lần lặp cuối cùng lần thứ 10..



Hình 2.6 Bộ đếm số vòng mã hoá

Khối này sử dụng một bộ đếm để điều khiển quá trình mã hóa. Bộ đếm sẽ bắt đầu hoạt động khi cipher_new_en=1. Sau đó, nó sẽ đếm lên trong suốt quá trình mã hóa, ứng với round_key_en =1, khi đó cipher_ready=0. Bộ đếm bị xóa tại vòng lặp mã hóa thứ 10, bước tính cuối cùng của mã hóa AES-128. Bên cạnh đó, bit cipher_ready, tín hiệu báo trạng thái mã hóa sẽ bằng 0 trong suốt quá trình mã hóa và tích cực 1 khi quá trình mã hóa kết thúc.



Hình 2.7 Kiến trúc khối mở rộng khoá cho chức năng mã hoá

Khối này thực hiện chức năng KeyExpansion đã trình bày ở mục 1.5. Khối này sử dụng một thanh ghi round_key_reg[127:0] để lưu lại giá trị khóa vòng sau mỗi lần tính toán. Tín hiệu round_num[3:0] sẽ cho biết khóa vòng bao nhiêu đang được tính, nếu bằng 0 thì khóa mã gốc cipher_key[127:0] sẽ nối đến key_in[127:0] và đưa vào mạch logic để tính khóa vòng đầu tiên. Từ lần tính tiếp theo trở đi, round_num[3:0] bằng 1 đến 10, giá trị được dùng để tính toán sẽ lấy từ thanh ghi round_key_reg[127:0]. Ở đây 32 bit cuối, ứng với w[3] (w[0] là 32 bit đầu [127:96]), sẽ được biến đổi qua các bước:

- RotWord: đảo byte MSB xuống LSB.
- SubWord: ứng với function S_box() được thết kế ở mục 2.2.3
- AddRcon: lấy ngõ ra SubWord XOR với giá trị Rcon chọn từ function AES_Rcon(). Function này chọn Rcon theo giá trị của round_num[3:0].
- Kết quả sau khi AddRcon sẽ được dùng để tạo ra các word của round_key[127:0] và lưu vào thanh ghi round_key_reg[127:0].

2.2.2 Khối xử lý giải mã/ Thuật toán mã hoá nghịch tương đương

Thuật toán mã hoá nghịch tương đương

Dựa theo các bước giải mã đã trình bày ở mục 1.3, em rút ra được một số nhận xét như sau:

Thuật toán giải mã mật mã AES không giống với mã hoá. Điều này có nghĩa là chuỗi các biến đổi cho giải mã khác với biến đổi cho mã hoá, mặc dù cấu trúc lịch trình khóa cho mã hoá và giải mã giống nhau. Tuy nhiên, có một phiên bản tương đương của thuật toán giải mã có cùng cấu trúc như thuật toán mã hoá. Phiên bản tương đương này có cùng chuỗi các biến đổi như thuật toán mã hoá (với các biến đổi được thay bằng các biến đổi ngược). Để đạt được tính tương đương này, cần một sự thay đổi trong lịch trình khóa. Tuy nhiên, có một phiên bản tương đương của thuật toán mã hóa có cấu trúc tương tự như thuật toán giải mã.

Phiên bản tương đương này có cùng chuỗi biến đổi như thuật toán mã hóa (với các biến đổi được thay thế bằng các biến đổi đối xứng). Để đạt được tương đương này, cần sự thay đổi lịch trình trong khóa. Để đồng bộ hoá cấu trúc giải mã với cấu trúc mã hóa, cần hai thay đổi riêng biệt. Như được minh hoạ trong các hình trên, một vòng mã hóa có cấu trúc SubBytes, ShiftRows, MixColumns, AddRoundKey. Vòng giải mã tiêu chuẩn có cấu trúc InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. Do đó, hai giai đoạn đầu tiên của vòng giải mã cần được hoán vị, và hai giai đoạn cuối của vòng giải mã cũng cần được hoán vị.

- **Hoán đổi InvShiftRows và InvSubBytes** InvShiftRows ảnh hưởng đến thứ tự các bytes trong state nhưng không thay đổi giá trị của byte và không phụ thuộc vào giá trị của byte để thực hiện biến đổi đó. InvSubBytes ảnh hưởng đến giá trị của byte thay thế nhưng không thay đổi thứ tự của byte trong State. Do đó 2 bước này hoàn toàn có thể hoán vị cho nhau.

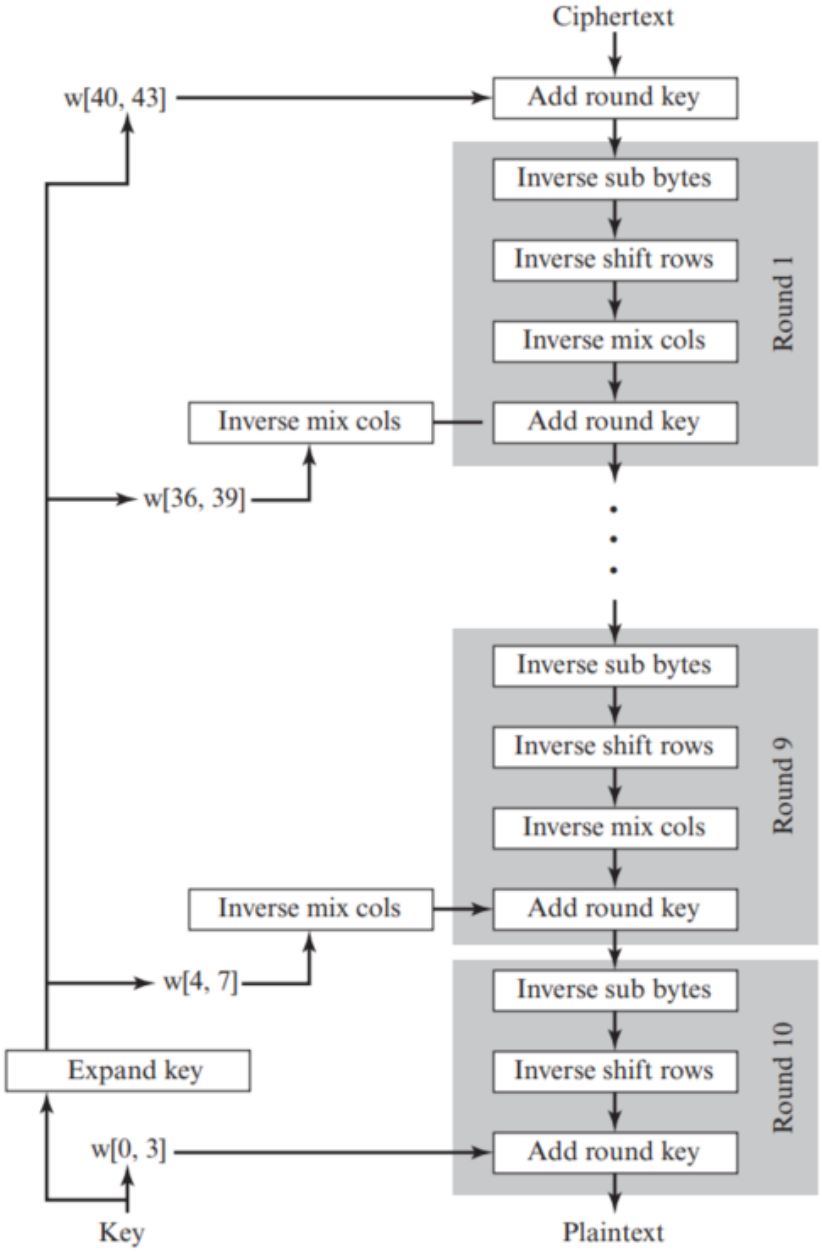
  InvShiftRow [**InvSubbytes**(S_i)] = **InvSubbytes** [**InvShiftRows**(S_i)]

- **Hoán đổi AddRoundKey và InvMixColumns** Các biến đổi AddRoundKey và InvMixColumns không thay đổi thứ tự các byte trong State. Nếu chúng ta xem khóa

là một chuỗi từ, thì cả AddRoundKey và InvMixColumns hoạt động trên State một cột một lần. Hai hoạt động này là tuyến tính với đầu vào cột. Đó là, cho một State S_i cho trước và một khóa vòng w_j cho trước.

InvMixColumns $(S_i \oplus w_j)$ = **InvMixColumns** $(S_i) \oplus$ **InvMixColumns** (w_j)

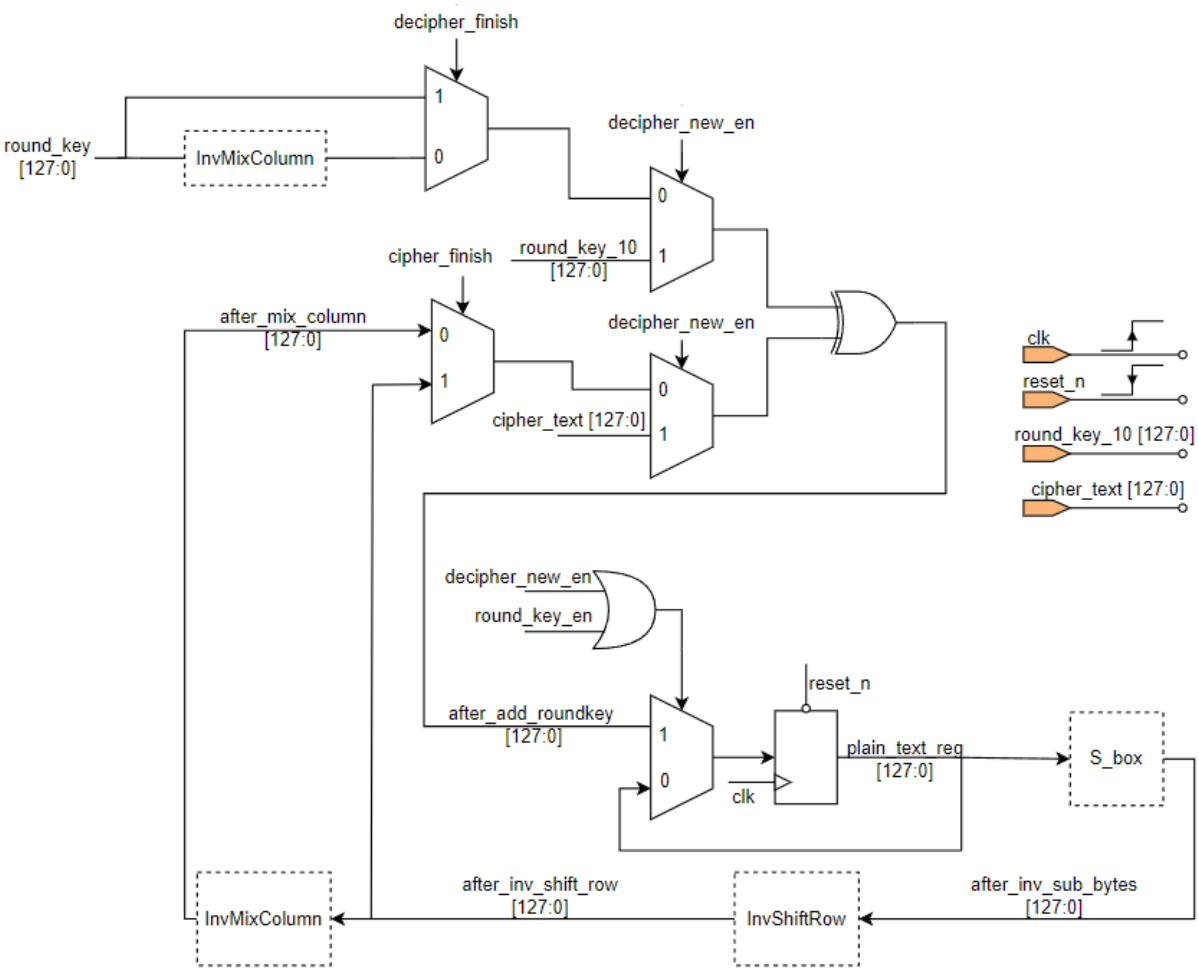
Như vậy, thuật toán giải mã hay có thể gọi là mã hoá nghịch tương đương được mô tả như hình 2.8 bên dưới:



Hình 2.8 Thuật toán giải mã

[1]

Từ thuật toán giải mã đã được triển khai như trên, khối xử lý giải mã sẽ có cấu trúc khá tương đồng với khối mã hoá.



Hình 2.9 Kiến trúc khối mã hoá

Mạch logic bộ giải mã AES-128 gồm 1 thanh ghi plain_text_reg[127:0] lưu lại giá trị sau mỗi vòng lặp mã hóa, thanh ghi này đặt sau chức năng AddRoundKey. Bắt đầu từ tín hiệu after_add_roundkey[127:0], tín hiệu này được tạo ra từ 1 cổng XOR, đây là chức năng AddRoundKey, với các ngõ vào được lấy từ: Khóa vòng số 10, round_key_10[127:0], và bản mã, cipher_text[127:0] cho lần tính đầu tiên khi decipher_en=1. Khóa vòng, round_key[127:0], và giá trị trung gian cho các lần tính tiếp theo, sau lần đầu tiên. Trong đó, giá trị trung gian lấy từ: Ngõ vào chức năng InvMixColumns, ứng với ngõ ra chức năng InvSubByte. after_addRoundkey[127:0] sẽ được lưu vào thanh ghi plaintext_reg[127:0] sau mỗi lần tính toán. Giá trị thanh ghi này sẽ lần lượt đi qua các chức năng: InvSubBytes: tính bằng function S_box(). InvShiftRows: chuyển đổi vị trí các byte trong 1 hàng bằng function InvShiftRow(). InvMixColumns: tính bằng function InvMixColumn().

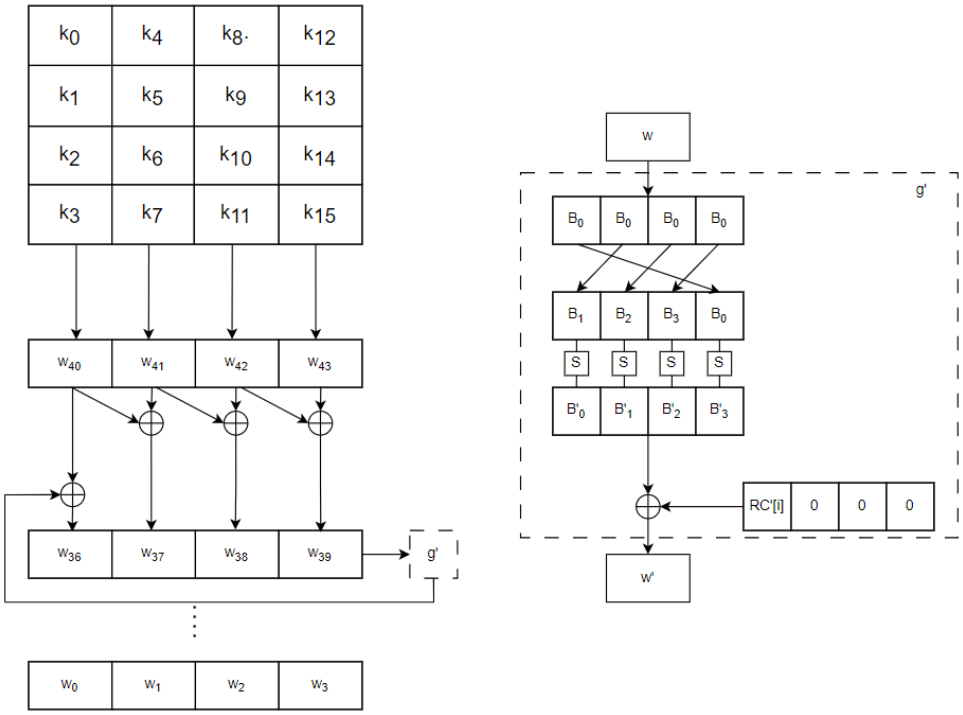
Việc biến đổi ngược round_key_10 trở về cipher_text ban đầu được dựa trên các phép biến đổi đã trình bày ở mục 1.5. Bằng công thức tổng quát như sau với : i = 0, 4, 8 ,...36, lần lượt các khoá vòng sẽ được biến đổi ngược về các khoá vòng trước đó.

$$w[i+3] = w[i+7] \oplus w[i+6]$$

$$w[i+2] = w[i+6] \oplus w[i+5]$$

$$w[i+1] = w[i+5] \oplus w[i+4]$$

$$w[i] = g'(w[i+3]) \oplus w[i+4]$$



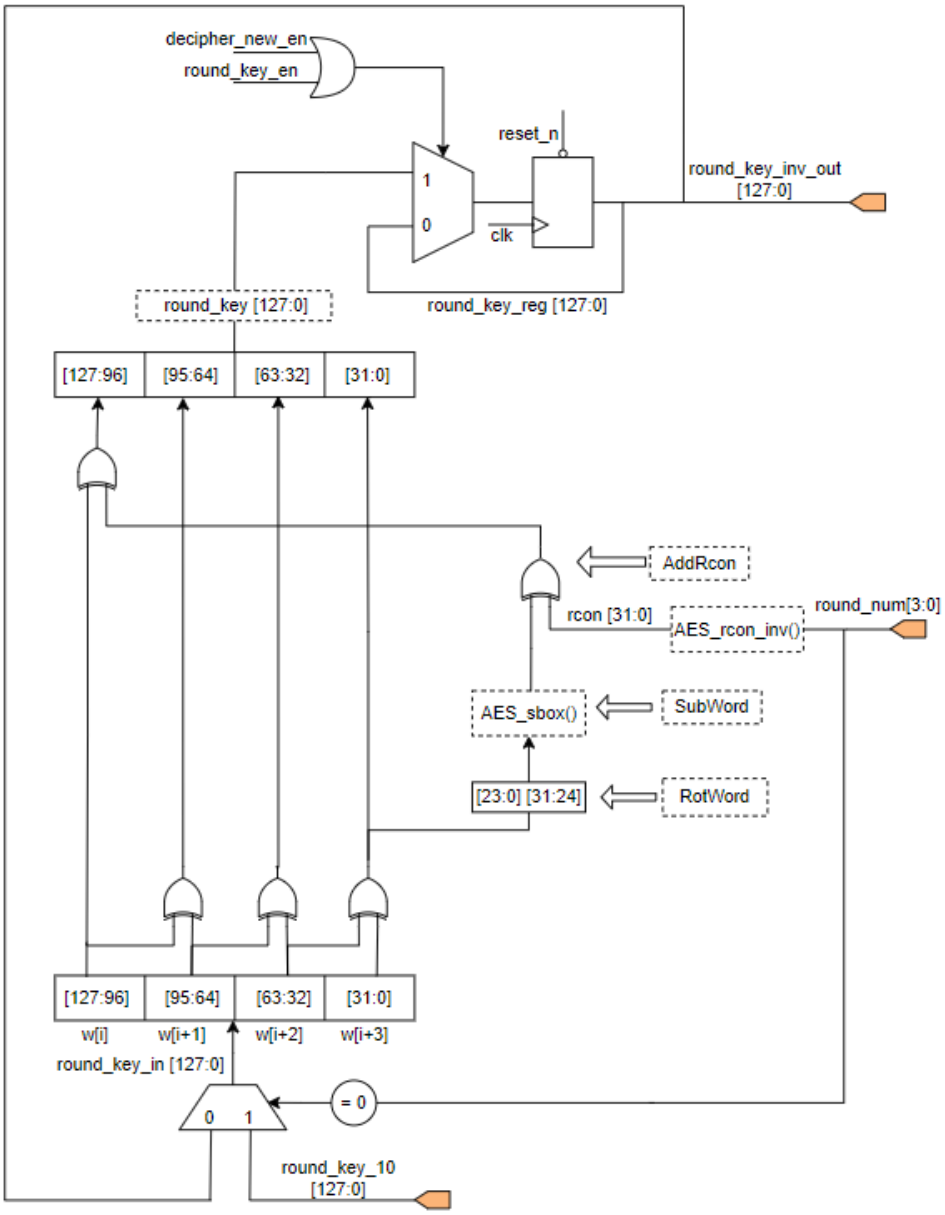
Hình 2.10 Thuật toán mở rộng khoá cho chức năng giải hoá

Với giá trị Rcon[i] được dùng bị đảo so với quá trình mã hoá như sau:

Vòng giải mã	1	2	3	4	5	6	7	8	9	10
Giá trị RC'[i]	36	1B	80	40	20	10	08	04	02	01

Bảng 2.2 Bảng giá trị hằng số RC' với quá trình giải mã

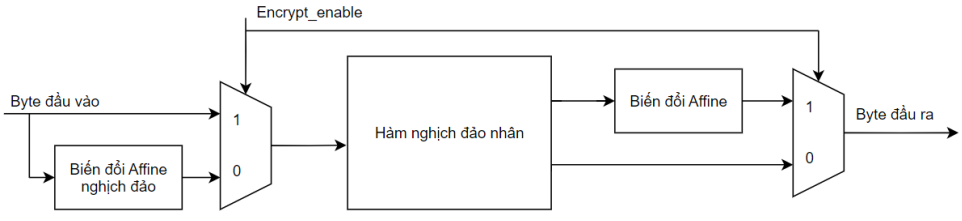
Như vậy kiến trúc khối mở rộng khoá cho chức năng giải mã sẽ được thiết kế như sau:



Hình 2.11 Kiến trúc khối mở rộng khoá cho chức năng giải mã

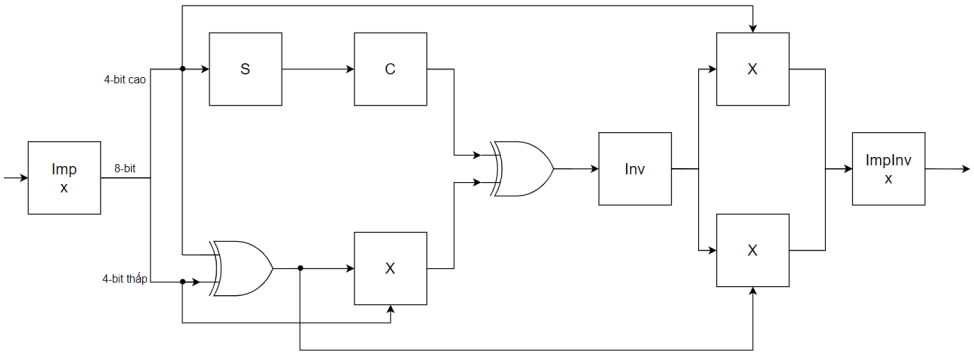
2.2.3 Khối S-box dựa trên biến đổi toán học

Ở mục này, em sẽ triển khai thiết kế khối S-box với chức năng SubBytes bằng biến đổi toán học như đã trình bày ở mục 1.4.1 với mô hình tham khảo được tác giả đưa ra trong tài liệu [7]. Về mặt logic, khối xử lý sẽ được triển khai như hình 2.12



Hình 2.12 Kiến trúc khối mở rộng khoá cho chức năng giải hoá

Mạch nguyên lý chung của bộ tính nghịch đảo trên trường như sau:



Hình 2.13 Kiến trúc khối biến đổi S-box

Chi tiết các khối chức năng trong sơ đồ trên như sau:

- Imp, gọi là Isomorphic Mapping, là khối ánh xạ phần tử của trường GF(2^8) vào trong trường hỗn hợp.
- S là khối tính bình phương trong trường GF(2^4).
- C là khối tính nhân với hằng số lambda λ trong trường GF(2^4). Lambda = 4'b1100
- Inv là khối tính phần tử nghịch đảo trong trường GF(2^4).
- X là khối nhân 2 số hạng trong trường GF(2^4).
- ImpInv, gọi là Inverse Isomorphic Mapping, là khối ánh xạ đảo của Imp, chuyển giá trị tình toán về trường GF(2^8)

Phép tính phần tử nghịch đảo trong trường hỗn hợp không thể được áp dụng trực tiếp vào phần tử trên trường GF(2^8). Nó phải được ánh xạ vào trường hỗn hợp thông qua biến đổi Isomorphic, Imp. Sau khi thực hiện xong phép tính phần tử nghịch đảo, kết quả sẽ được ánh xạ ngược trở lại từ trường hỗn hợp sang trường GF(2^8) tương ứng thông qua biến đổi ImpInv. Gọi q là phần tử cần chuyển đổi, trong đó q_7 là MSB còn q_0 là LSB.

Khối **Imp** thực hiện phép nhân ma trận sau đây:

$$\delta \times q = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix} = \begin{bmatrix} q_7 \oplus q_5 \\ q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_6 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_6 \oplus q_4 \oplus q_1 \\ q_6 \oplus q_1 \oplus q_0 \end{bmatrix}$$

Tương tự, khối **InvImp** thực hiện ma trận sau đây:

$$\delta \times q = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix} = \begin{bmatrix} q_7 \oplus q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_2 \\ q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_0 \end{bmatrix}$$

Các đa thức bất kỳ có thể biểu diễn dưới dạng $bx + c$ trong đó b là các phần trọng số cao còn c là phần trọng số thấp. Từ đó, một số nhị phân q có thể được biểu diễn là$qH.x + pL$. Ví dụ, q = 4'b1001 thì có thể biểu diễn là 2'b10.x + 2'b01, trong đó, q_H = 2'b10 và q_L=2'b01. q_H và q_L có thể tiếp tục được biểu diễn như sau:

$$q_H = B_1.x + B_0 \tag{2.3}$$
$$q_L = B_0.x + B_1 \tag{2.4}$$

Đây là ý tưởng được dùng để thực hiện các phép toán như cộng, nhân hay bình phương. Bên cạnh đó, quá trình hạ bậc đa thức để tính toán và biến đổi trong trường hỗn hợp sẽ sử dụng các đa thức tối giản sau đây:

$$GF(2^2) \rightarrow GF(2) : x^2 + x + 1$$
$$GF((2^2)^2) \rightarrow GF(2^2) : x^2 + x + \phi$$
$$GF(((2^2)^2)^2) \rightarrow GF((2^2)^2) : x^2 + x + \lambda$$

Trong đó: ϕ = 2'b10 và λ = 4'b1100

Việc tính bình phương trong trường GF(2^4) được thực hiện trên số k (4 bit) sẽ được biến đổi như các bước sau đây. Đầu tiên, giá trị k sẽ được biểu diễn dưới dạng:

$$k = q_H.x + q_L \tag{2.5}$$

Bình phương của k được tính như sau:

$$k^2 = (q_H.x + q_L)^2 = (q_H^2 x^2 + 2q_H.q_L.x + q_L^2) = q_H^2 + q_L^2 \tag{2.6}$$

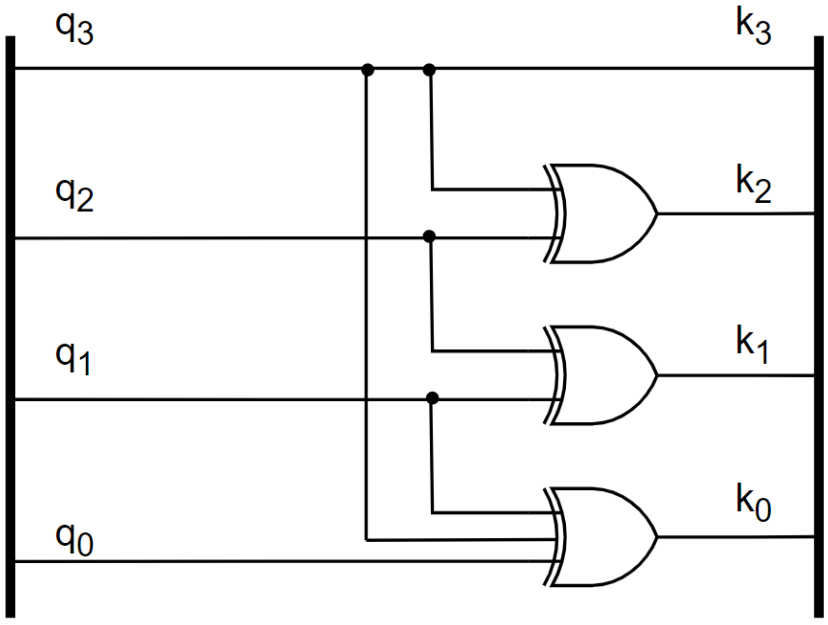
Thay tính k_H dựa trên $q_H = q_3 q_2$:

$$k_H = q_H^2 = (q_3 q_2)^2 = (q_3.x + q_2)^2 = q_3.x^2 + q2 \tag{2.7}$$

Như vậy, phép tính bình phương một giá trị 4 bit q=$q_3 q_2 q_1 q_0$ trong trường GF(2^4) sẽ cho kết quả là một giá trị 4 bit k= $k_3 k_2 k_1 k_0$ bằng logic sau đây:

$$k_3 = q_3, k_2 = q_3 \oplus q_2, k_1 = q_2 \oplus q_1, k_0 = q_3 \oplus q_1 \oplus q_0 \tag{2.8}$$

Logic thiết kế như sau:



Hình 2.14 Logic khối tính bình phương trong trường GF(2^4) (S)

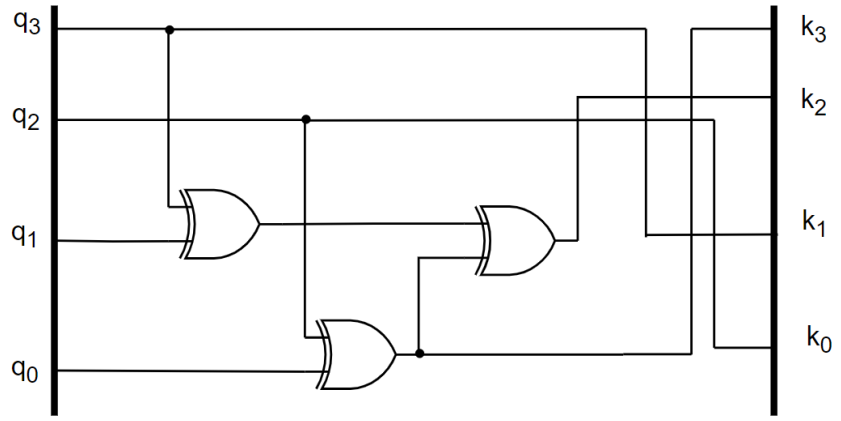
Đặt k = qλ, trong đó k= $k_3 k_2 k_1 k_0$ và q = $q_3 q_2 q_1 q_0$ và λ=4'b1100 là các phần tử 4 bit thuộc trường GF(2^4). Áp dụng phương pháp tính đưa về biểu thức dạng bx + c.

$$k = k_3 k_2 k_1 k_0 = k_H.x + k_L = (q_H x + q_L)(\lambda_H x + \lambda_L) = q_H \lambda_H x^2 + q_L \lambda_L x = q_H \lambda_H x^2 \tag{2.9}$$

Như vậy, phép nhân một giá trị 4 bit q = $q_3q_2q_1q_0$ trong trường GF(2^4) với hằng số lambda sẽ cho kết quả là một giá trị 4 bit k= $k_3k_2k_1k_0$ bằng logic sau đây:

$$k_3 = q_2 \oplus q_0$$
$$k_2 = q_3 \oplus q_2 \oplus q_1 \oplus q_0$$
$$k_1 = q_3$$
$$k_0 = q_2$$

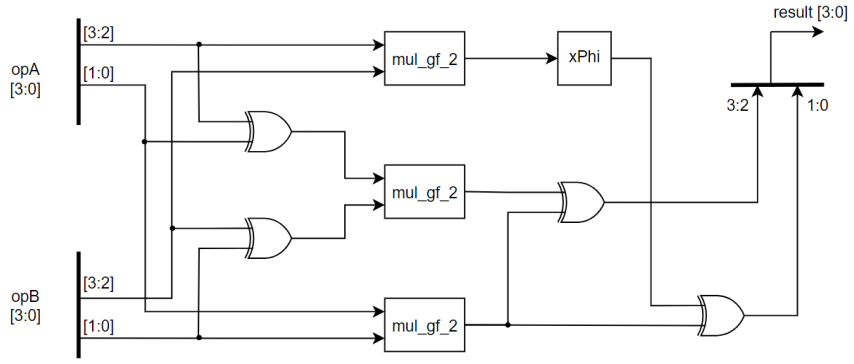
Logic thiết kế như sau:



Hình 2.15 Logic khối nhân với hằng số lambda GF(2^4) (S)

Tính nhân hai phần tử trong trường GF(2^4) (khối X) Đặt k=qw, trong đó k= $k_3k_2k_1k_0$, q = $q_3q_2q_1q_0$ và w = $w_3w_2w_1w_0$ là phần tử 4 bit của trường GF(2^4).

$$k = (\underbrace{k_3k_2}_{k_H} + \underbrace{k_1k_0}_{k_L}) = k_Hx + k_L$$
$$= (q_3q_2q_1q_0)(w_3w_2w_1w_0) = (q_Hx + q_L)(w_Hx + w_L) = (q_H + q_L)(w_H + w_L) + q_Lw_L$$



Hình 2.16 Logic nhân 2 phần tử trong trường GF(2^4) (Inv)

Phép nhân hai phần tử 2 bit cho kết quả như sau:

$$k_1 = q_1 w_1 \oplus q_0 w_1 \oplus q_1 w_0$$
$$k_0 = q_1 w_1 \oplus q_0 w_0$$

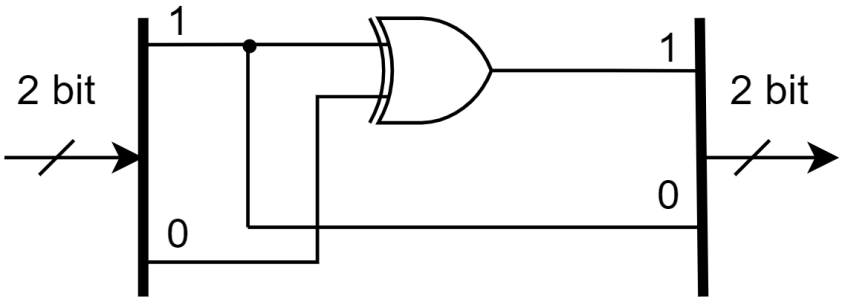
Với cách biến đổi tương tự, chúng ta có thể tính được phép nhân một phần tử 2 bit với với hằng số ϕ = 2'b10. Đặt k = $q\phi$, trong đó k=$k_1 k_0$ và q=$q_1 q_0$ là phần tử 2 bit của trường GF(2^2).

$$k = k_1 x + k_0 = (q_1 q_0)(10_2) = (q_1 x + q_0)x = q_1 x^2 + q_0 x$$
$$= q_1(x+1) + q_0 x = (q_1 + q_0)x + q_1$$

Vậy phép nhân một phần tử 2 bit với ϕ cho kết quả như sau:

$$k_1 = q_1 \oplus q_0$$

$$k_0 = q_1$$

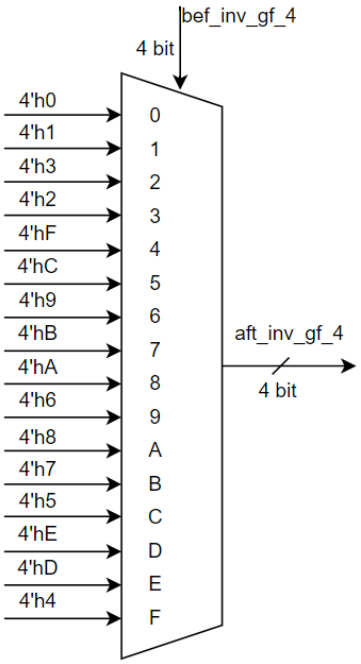


Hình 2.17 Logic nhân phần tử với hằng số ϕ trong trường GF(2^2) (Inv)

Tính nghịch đảo trên trường GF(2^4): Giá trị nghịch đảo của một số q = $q_3 q_2 q_1 q_0$ được tính như sau:

$$q_3^{-1} = q_3 \oplus q_3 q_2 q_1 \oplus q_3 q_0 \oplus q_2$$
$$q_2^{-1} = q_3 q_2 q_1 \oplus q_3 q_2 q_0 \oplus q_3 q_0 \oplus q_2 \oplus q_2 q_1$$
$$q_1^{-1} = q_3 \oplus q_3 q_2 q_1 \oplus q_3 q_1 q_0 \oplus q_2 \oplus q_2 q_0 \oplus q_1$$
$$q_0^{-1} = q_3 q_2 q_1 \oplus q_3 q_2 q_0 \oplus q_3 q_1 \oplus q_3 q_1 q_0 \oplus q_3 q_0 \oplus q_2 \oplus q_2 q_1 \oplus q_2 q_1 q_0 \oplus q_1 \oplus q_0$$

Thiết kế logic tương đương với một bộ MUX 16 giá trị như sau:

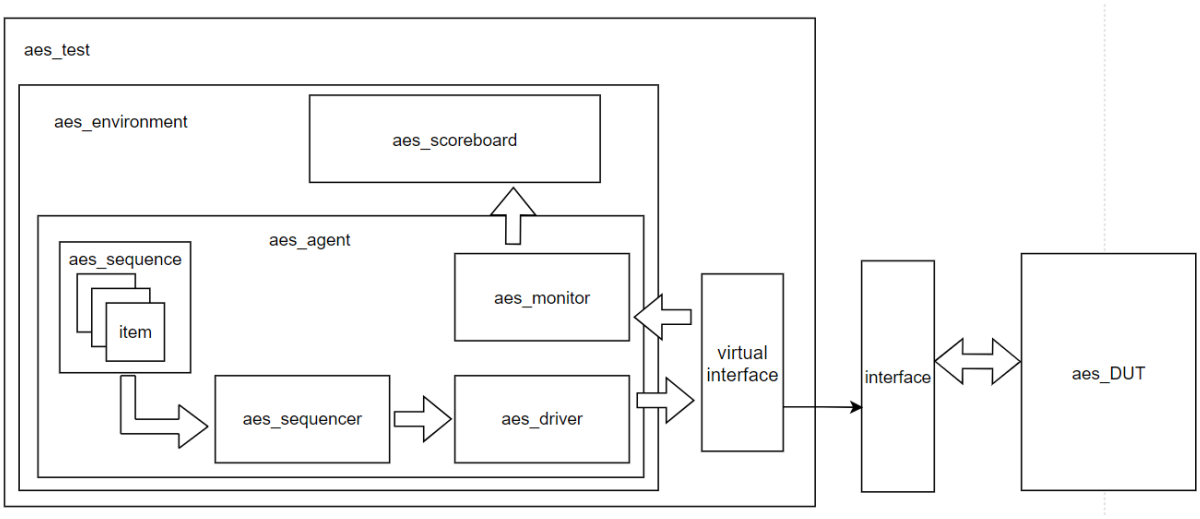


Hình 2.18 Logic tính nghịch đảo trong trường GF(2^4) (Inv)

Như vậy, dựa vào biến đổi toán học, có thể thiết kế được khối biến đổi S-box từ các phần tử logic thay vì cách lập trình dựa trên LUTs và tránh được độ trễ cố định.

CHƯƠNG 3: XÂY DỰNG MÔI TRƯỜNG KIỂM THỬ VÀ KẾT QUẢ MÔ PHỎNG

3.1 Xây dựng môi trường kiểm thử



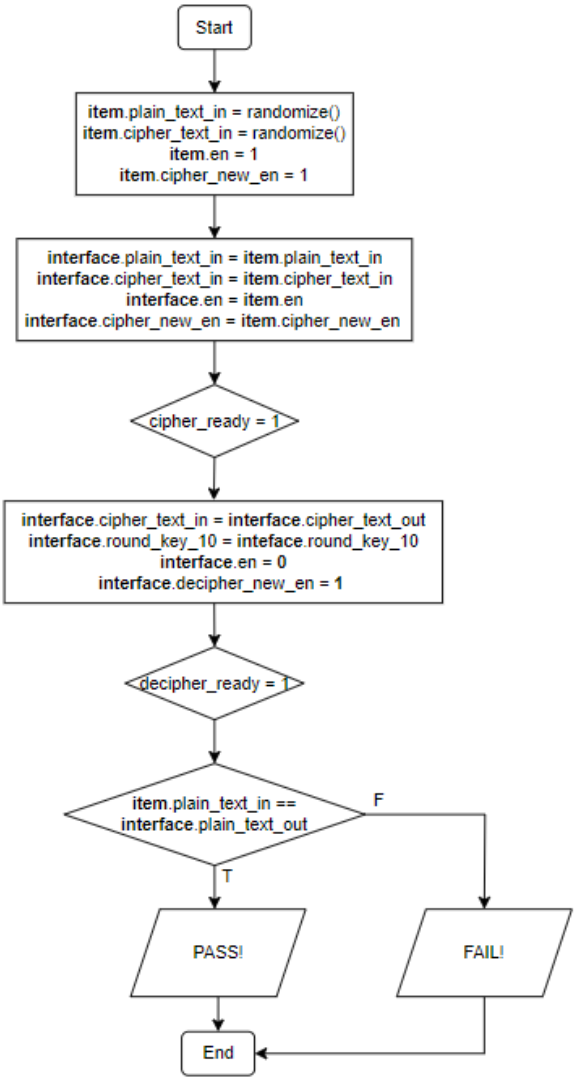
Hình 3.1 Môi trường kiểm thử đề xuất

Về mặt chức năng, các khối trong môi trường kiểm thử như đề xuất trên hình 3.1 sẽ thực hiện với thuật toán như sau:

Tên khối	Lớp kế thừa	Mô tả chức năng
aes_test	uvm_test	Class cấp cao nhất, chứa các thành phần còn lại của kiến trúc uvm
aes_environment	uvm_environment	Môi trường chính, chứa scoreboard và agent để giao tiếp với khối DUT
aes_agent	uvm_agent	Các cá nhân trao đổi ý tưởng với người khác
aes_driver	uvm_driver	Tiếp nhận các kích thích từ sequence, điều kiển dữ liệu từ dạng sequence_item sang interface data
aes_monitor	uvm_monitor	Lấy dữ liệu trả về từ DUT và gửi lên scoreboard để phân tích, đánh giá
aes_sequence	uvm_sequence	Tạo ra các kích thích ngẫu nhiên, trực tiếp gửi cho driver thông qua sequencer bằng cơ chế driver-sequence handshake
aes_sequencer	uvm_sequencer	Khối trung gian điều khiển quá trình truyền các sequence_item giữa sequence và driver
aes_sequence_item	uvm_sequence_item	Kích thích đầu vào gửi tới DUT dưới dạng các item.
aes_scoreboard	uvm_scoreboard	So sánh kích thích đầu vào và dữ liệu trả về từ DUT, nếu trùng khớp sẽ hiển thị thông báo lên màn hình.

Bảng 3.1 Bảng mô tả chức năng các khối trong môi trường kiểm thử

Thuật toán kiểm thử cụ thể được thể hiện trong hình 3.2:



Hình 3.2 Thuật toán kiểm thử

3.2 Kết quả lập trình, mô phỏng và kiểm thử

3.2.1 Các công cụ sử dụng

Sau quá trình xây dựng kiến trúc các khối xử lý, môi trường kiểm thử và xây dựng thuật toán, em sẽ đi đến bước lập trình và mô phỏng để kiểm tra tính chính xác của thiết kế.

Một số công cụ phần mềm em sử dụng trong phần này như sau:

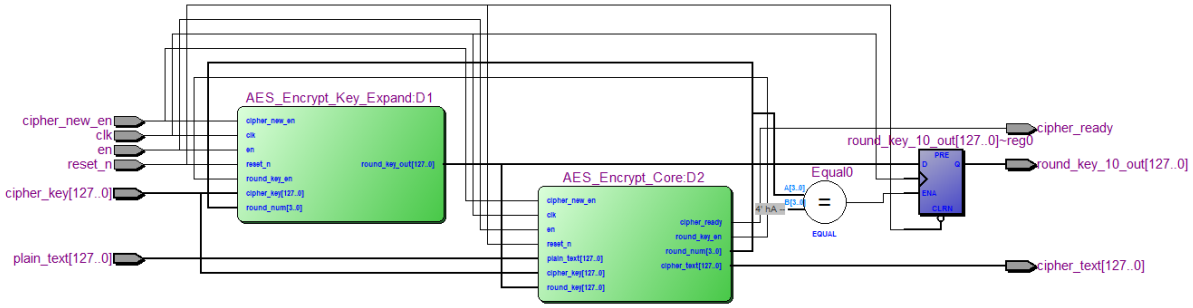
Lập trình RTL code: Sublime Text, Visual Code Studio.

Biên dịch, tổng hợp: Quartus II Web Edition 13.0.

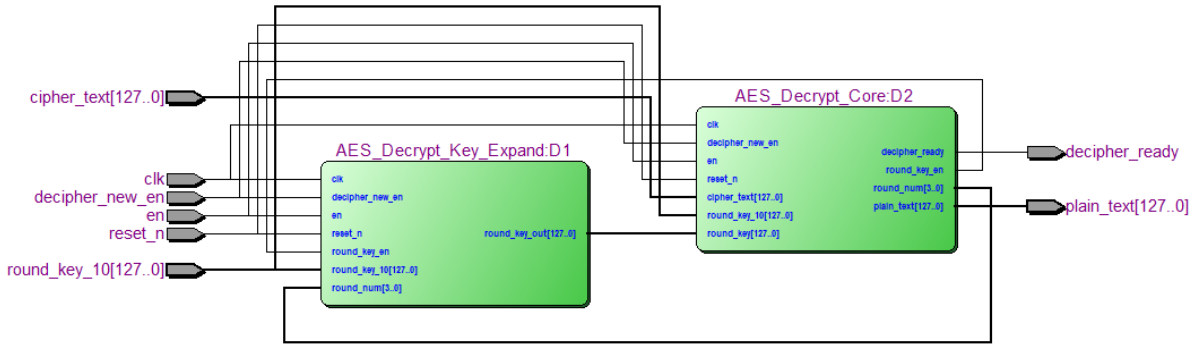
Mô phỏng: QuestaSim 10.2c.

3.2.2 Kết quả

Sau đây là một số kết quả tổng hợp trên công cụ Quartus II và kết quả mô phỏng trên QuestaSim:



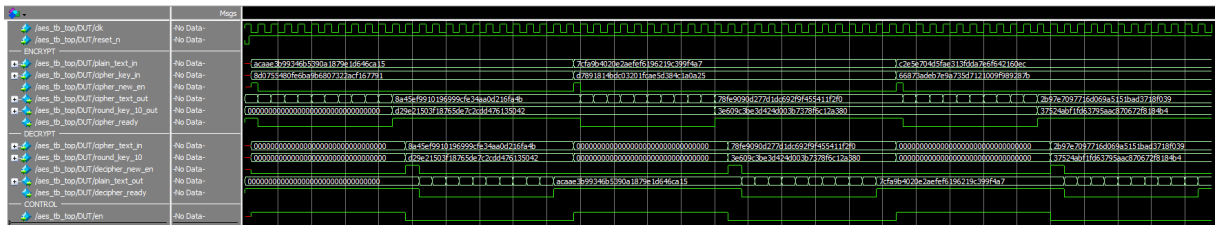
Hình 3.3 Tổng hợp khối Encrypt trên Quartus II



Hình 3.4 Tổng hợp khối Decrypt trên Quartus II

```
# UVM_INFO ../src/aes_agent/aes_encrypt_driver.sv(55) @ 20: uvm_test_top.env.encrypt_agent.driver [ENCRYPT_DRIVER] SEND DATA
# UVM_INFO ../src/aes_scoreboard/aes_scoreboard.sv(30) @ 480: uvm_test_top.env.scb [UVM_INFO] PASS!
# plain_text_in = acaae3b99346b5390a1879e1d646ca15 , plain_text_out = acaae3b99346b5390a1879e1d646ca15
# UVM_INFO ../src/aes_agent/aes_encrypt_driver.sv(55) @ 500: uvm_test_top.env.encrypt_agent.driver [ENCRYPT_DRIVER] SEND DATA
# UVM_INFO ../src/aes_scoreboard/aes_scoreboard.sv(30) @ 960: uvm_test_top.env.scb [UVM_INFO] PASS!
# plain_text_in = 7cfa9b4020e2aefef6196219c399f4a7 , plain_text_out = 7cfa9b4020e2aefef6196219c399f4a7
# UVM_INFO ../src/aes_agent/aes_encrypt_driver.sv(55) @ 980: uvm_test_top.env.encrypt_agent.driver [ENCRYPT_DRIVER] SEND DATA
# UVM_INFO ../src/aes_scoreboard/aes_scoreboard.sv(30) @ 1440: uvm_test_top.env.scb [UVM_INFO] PASS!
# plain_text_in = c2e5e704d5fae313fdda7e6f642160ec , plain_text_out = c2e5e704d5fae313fdda7e6f642160ec
```

Hình 3.5 Kết quả kiểm thử trên QuestaSim



Hình 3.6 Kết quả kiểm thử dạng tín hiệu trên QuestaSim

CHƯƠNG 4: KẾT LUẬN

TÀI LIỆU THAM KHẢO

[1] W. Stallings, *Cryptography and Network Security Principle and Practice Seventh Edition*. PEARSON INDIA, 2016.

[2] NIST, "Federal information processing standards publication 197," NIST, Tech. Rep., 2001.

[3] Wikipedia, "Block cipher mode of operation." [Online]. Available: en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[4] D. M. H. Neil H. E. Weste, *CMOS VLSI Design A Circuits and Systems Perspective-Fourth Edition*. PEARSON, 2016.

[5] G. T. Chris Spear, *SystemVerilog for Verification- A Guide to Learning the Testbench Language Features*. Springer, 2006.

[6] Accellera, "Uvm 1.2 class reference," Natural Docs, Tech. Rep., 2014.

[7] S. D. K. . D. M. Devi, "Design of s-box and inv s-box using composite field arithmetic for aes algorithm," 2018.