ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỆN TỬ

----------o0o----------

ĐỒ ÁN MÔN HỌC

THIẾT KẾ PHẦN CỨNG XỬ LÝ MÃ HÓA AES

GVHD: Th.S Trịnh Vũ Đăng Nguyên

SVTH: Phạm Xuân Thi

MSSV: 1814120

TP. HỒ CHÍ MINH, THÁNG 05 NĂM 2022

LỜI CẢM ƠN

Trong suốt quá trình học tập, thực hiện và hoàn thành đồ án này, em đã nhận được sự hướng dẫn, giúp đỡ quý báu của các thầy cô, các anh chị. Với lòng kính trọng và biết ơn sâu sắc em xin được bày tỏ lời cảm ơn chân thành tới: ThS. Trịnh Vũ Đăng Nguyên, người thầy kính mến đã hết lòng giúp đỡ, dạy bảo, động viên và tạo mọi điều kiện thuận lợi cho em trong suốt quá trình thực hiện đồ án. Cảm ơn các quý thầy cô trong khoa Điện – Điện tử, trường Đại Học Bách Khoa thành phố Hồ Chí Minh đã tận tình chỉ dạy và truyền đạt kiến thức giúp em có thể đạt được kết quả như ngày hôm nay.

Bên cạnh đó, em xin chân thành cảm ơn bố mẹ và gia đình đã luôn hỗ trợ, động viên về mặt vật chất và tinh thần, giúp em hoàn thành tốt được đồ án này.

Tp. Hồ Chí Minh, ngày 30 tháng 05 năm 2022

Sinh viên

Phạm Xuân Thi

TÓM TẮT ĐỒ ÁN

Đồ án này trình bày về thiết kế phần cứng xử lý mã hóa AES. Đánh giá tài nguyên phần cứng khi thực thi trên FPGA Cyclone V 5CGXFC9E7F35C8.

MỤC LỤC

DANH SÁCH HÌNH MINH HỌA

DANH SÁCH BẢNG SỐ LIỆU

1. GIỚI THIỆU

1.1 Tổng quan

AES (viết tắt của từ tiếng anh: Advanced Encryption Standard, hay Tiêu chuẩn mã hóa nâng cao) là một thuật toán mã hóa khối được chính phủ Hoa Kỳ áp dụng làm tiêu chuẩn mã hóa. Thuật toán được xây dựng dựa trên Rijndael Cipher phát triển bởi 2 nhà mật mã học người Bỉ: Joan Daemen và Vincent Rijmen.

AES làm việc với các khối dữ liệu 128-bit và độ dài khóa 128-bit, 192-bit hoặc 256-bit. Các khóa mở rộng sử dụng trong chu trình được tạo ra bởi thủ tục sinh khóa Rijndael. Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn của các byte. Mỗi khối dữ liệu đầu vào 128-bit được chia thành 16-byte, có thể xếp thành 4 cột, mỗi cột 4 phần tử hay một ma trận 4x4 của các byte, nó gọi là ma trận state.

Tùy thuộc vào độ dài của khóa khi sử dụng 128-bit, 192-bit hay 256-bit mà thuật toán được thực hiện với số lần lặp khác nhau.

1.2 Nhiệm vụ đề tài

Nội dung 1: Tìm hiểu lý thuyết về mã hóa AES theo phương pháp mã hóa Rijndael.

Nội dung 2: Phân tích thiết kế các khối và tổng hợp.

Nội dung 3: Kết quả mô phỏng và đánh giá phần cứng xử lý mã hóa AES.

2. LÝ THUYẾT

2.1 Mã hóa Rijndael

❖ Trường hữu hạn GF (2^n)

AES sử dụng số học trong trường hữu hạn GF (2n) với bất khả quy đa thức $m(x) = x^8 + x^4 + x^3 + x + 1$ khi nhân 2 đa thức, còn khi cộng 2 đa thức ta thực hiện phép XOR của từng bit.

Ví dụ: xét 2 phần tử $A = (a_7 a_6 \ldots a_1 a_0)$ và $B = (b_7 b_6 \ldots b_1 b_0)$. Phép cộng $C = A + B = (c_7 c_6 \ldots c_1 c_0)$, khi $c_i = a_i \oplus b_i$. Phép nhân $\{02\} \bullet A = (a_6 \ldots a_1 a_0 0)$ khi $a_7 = 0$ và $\{02\} \bullet A = (a_6 \ldots a_1 a_0 0) \oplus (00011011)$ khi $a_7 = 1$.

❖ Đặc tả thuật toán

Xử lý các khối dữ liệu 128 bit (plaintext và ciphertext).

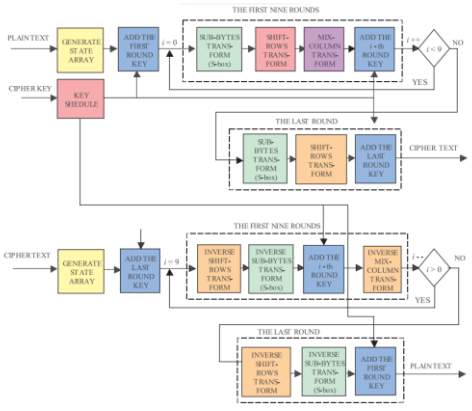
Tùy thuộc vào độ dài của key khi sử dụng 128-bit, 192-bit và 256-bit mà mã hóa AES có số vòng lặp khác nhau. Cụ thể như sau:

	Độ dài khóa (N_k)	Kích thước khối (N_b)	Số vòng lặp (N_r)	Kích thước khóa vòng	Kích thước khóa được mở rộng
AES-128	4	4	10	4	44
AES-192	6	4	12	4	52
AES-256	8	4	14	4	60

Bảng 1: Thông số AES

2.2 Cấu trúc AES

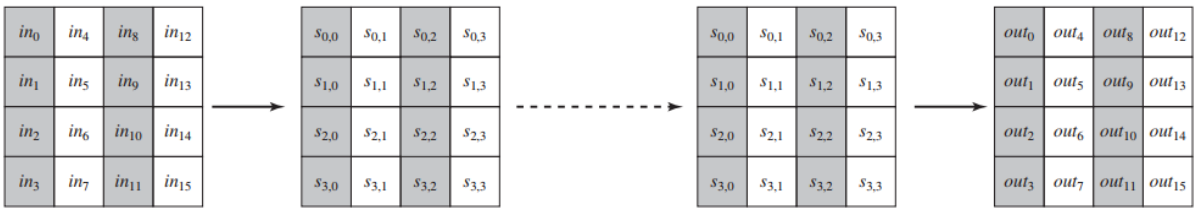
❖ Cấu trúc chung



Hình 1: Cấu trúc AES

Thuật toán gồm 3 hàm chính: Cipher, Inverse Cipher và Key Expansion. Hàm Cipher chuyển đổi dữ liệu sang dữ liệu khác là ciphertext trong khi hàm Inverse Cipher chuyển đổi dữ liệu ciphertext trở về ban đầu. Hàm Key Expansion tạo ra một Key Schedule từ dữ liệu cipher key được sử dụng trong quy trình Cipher và Inverse Cipher. Thông số của Cipher và Inverse Cipher được liệt kê trong Bảng 1.

Đầu vào và đầu ra cho thuật toán AES bao gồm các chuỗi 128-bit. Dữ liệu cipher key của thuật toán AES là một chuỗi 128, 192 hoặc 256-bit. Đơn vị cơ bản để xử lý trong thuật toán AES là 1 byte (8-bit), vì vậy chuỗi bit đầu vào đầu tiên được chuyển thành byte. Trong bước tiếp theo, một mảng hai chiều của byte (được gọi là State) được xây dựng. Mảng trạng thái (State array) bao gồm 4 hàng byte, mỗi hàng chứa N_b byte, trong đó N_b là kích thước khối (block size) chia cho 32-word. Tất cả các hoạt động của Cipher và Inverse Cipher của thuật toán AES được thực hiện trên State array, sau đó giá trị cuối cùng của nó được sao chép vào đầu ra (State array được biến đổi trở lại thành chuỗi bit).



(a) Đầu vào, mảng state và đầu ra

(b) Mảng khóa và khóa mở rộng

Hình 2: Cấu trúc dữ liệu của AES

❖ Cấu trúc chi tiết

Cấu trúc AES này không phải là cấu trúc Feistel. AES xử lý toàn bộ khối dữ liệu dưới dạng một ma trận duy nhất trong mỗi vòng bằng cách sử dụng thay thế và hoán

vị. Còn cấu trúc Feistel là một nửa khối dữ liệu được sử dụng để sửa đổi và sẽ được hoán đổi giữa hai khối dữ liệu.

Khóa được cung cấp dưới dạng đầu vào được mở rộng thành một mảng 44-word 32-bit, $w[i]\ (i = 0 \rightarrow 43)$. Bốn word riêng biệt (128 bit) đóng vai trò như một khóa vòng cho mỗi vòng.

Bốn giai đoạn khác nhau được sử dụng, một giai đoạn hoán vị và ba giai đoạn thay thế:

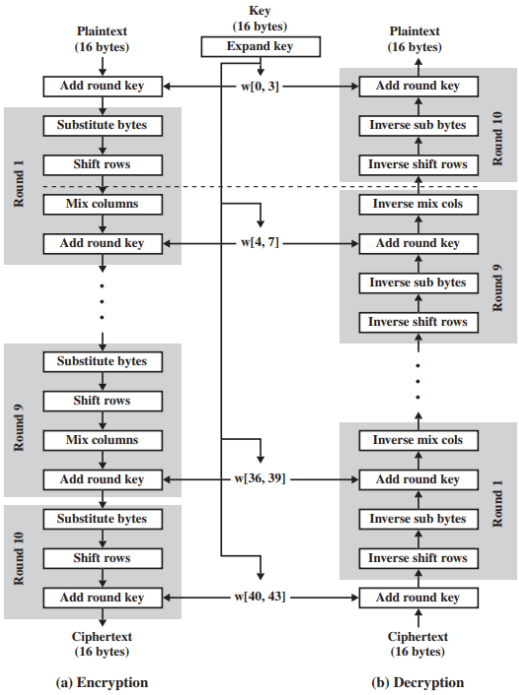
- **Substitute bytes:** Sử dụng S-box để thực hiện thay thế từng byte của khối.
- **ShiftRows:** Hoán vị theo từng dòng của mảng.
- **MixColumns:** Thay thế sử dụng số học.
- **AddRoundKey:** Phép XOR bit của khối hiện tại với một phần của khóa mở rộng.

Đối với cả mã hóa và giải mã, mật mã bắt đầu với một giai đoạn AddRoundKey, tiếp theo là chín vòng mà mỗi vòng bao gồm tất cả bốn giai đoạn, tiếp theo là vòng thứ mười gồm ba giai đoạn. Hình 4 mô tả cấu trúc của một vòng mã hóa đầy đủ.

Chỉ giai đoạn AddRoundKey sử dụng khóa. Vì lý do này, mật mã bắt đầu và kết thúc bằng một giai đoạn AddRoundKey. Bất kỳ giai đoạn nào khác, được áp dụng ở phần đầu hoặc phần cuối, đều có thể hoàn nguyên mà không cần biết về khóa và do đó sẽ không tăng thêm tính bảo mật.

Mỗi giai đoạn có thể dễ dàng đảo ngược. Đối với các giai đoạn Substitute Byte, ShiftRows và MixColumns, một hàm nghịch đảo được sử dụng trong thuật toán giải mã. Đối với giai đoạn AddRoundKey, nghịch đảo đạt được bằng cách XOR cùng một phím tròn vào khối, sử dụng kết quả $A \oplus B \oplus B = A$.

Như với hầu hết các mật mã khối, thuật toán giải mã sử dụng khóa mở rộng theo thứ tự ngược lại. Tuy nhiên, thuật toán giải mã không giống với thuật toán mã hóa. Đây là hệ quả của cấu trúc đặc biệt của AES.



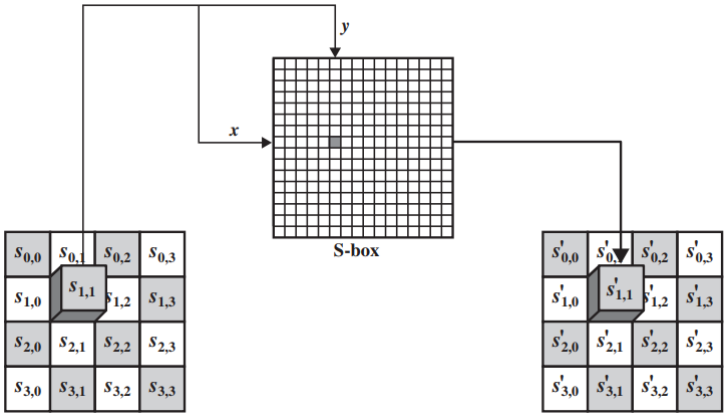
Hình 3: Mã hóa và giải mã AES

Hình 4: Vòng mã hóa AES

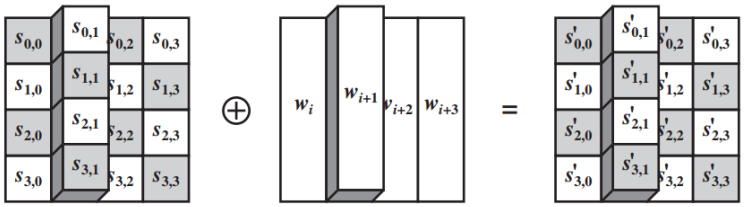
2.3 Các hàm chức năng chuyển đổi AES

❖ Hàm Substitute Bytes

Phép biến đổi byte thay thế, được gọi là SubBytes, là một phép tra cứu bảng. AES định nghĩa một ma trận của giá trị byte, được gọi là S-box, chứa một hoán vị của tất cả 256 giá trị 8-bit có thể có. Mỗi byte state riêng lẻ được ánh xạ thành một byte theo: 4-bit ngoài cùng bên trái của byte được sử dụng làm giá trị hàng và 4-bit ngoài cùng bên phải của byte được sử dụng làm giá trị cột. Các giá trị hàng và cột này đóng vai trò là xem giá trị đầu ra 8-bit khi nhìn vào S-box.



(a) Chuyển đổi byte thay thế



(b) Cộng với biến đổi khóa vòng

Hình 5: Hoạt động theo từng byte

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

(a) S-box

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

(b) S-box nghịch đảo

Bảng 2: AES S-box

Ví dụ: Thực hiện hàm SubBytes

EA	04	65	85	→	87	F2	4D	97
83	45	5D	96		EC	6E	4C	90
5C	33	98	B0		4A	C3	46	E7
F0	2D	AD	C5		8C	D8	95	A6

Tương tự với phép SubBytes, phép biến đổi byte thay thế nghịch đảo, được gọi là InvSubBytes, sử dụng bảng S-box nghịch đảo.

❖ Hàm ShiftRows

Phép chuyển đổi hàng để dịch chuyển về phía trước, được gọi là ShiftRows. Hàng đầu tiên của state không bị thay đổi. Đối với hàng thứ hai, ba và tư thì dịch sang trái lần lượt là 1, 2 và 3-byte.

Ví dụ:

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Phép biến đổi hàng dịch ngược, được gọi là InvShiftRows, thực hiện dịch byte theo hướng ngược lại cho mỗi hàng trong số ba hàng cuối cùng.

❖ Hàm MixColumns

Phép biến đổi cột hỗn hợp chuyển tiếp, được gọi là MixColumns, hoạt động trên từng cột riêng lẻ. Mỗi byte của một cột được ánh xạ thành một giá trị mới là một hàm của tất cả 4-byte trong cột đó. Các phép biến đổi có thể được xác định bằng phép nhân ma trận trên state. Mỗi phép cộng và phép nhân được thực hiện trong GF (2^8).

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

$$\Rightarrow \begin{aligned} s'_{0,j} &= (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j}) \\ s'_{3,j} &= (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j}) \end{aligned}$$

Ví dụ:

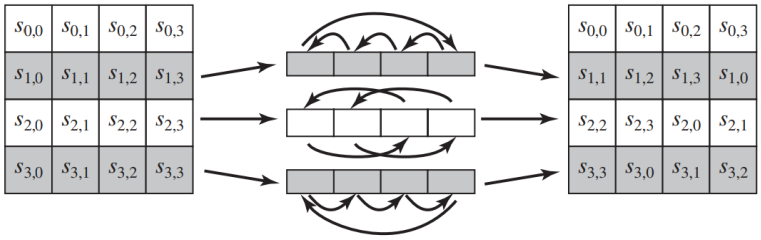
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

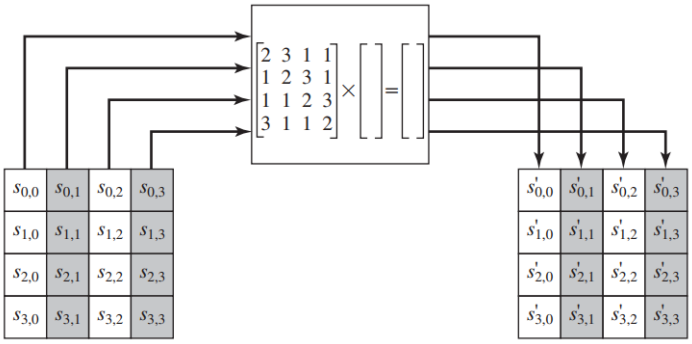
47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Phép biến đổi cột hỗn hợp nghịch đảo, được gọi là InvMixColumns, được xác định bằng phép nhân ma trận sau:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$



(a) Biến đổi ShiftRows



(b) Biến đổi MixColumns

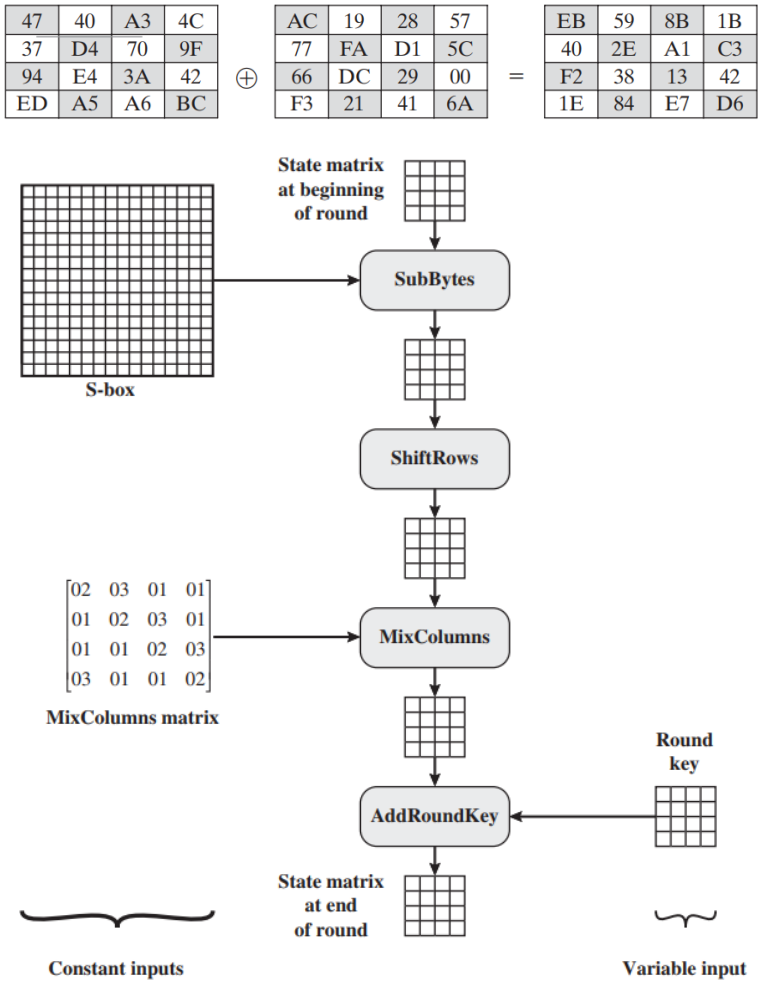
Hình 6: Cách hoạt động hàng và cột trong AES

❖ Hàm AddRoundKey

Trong phép biến đổi khóa vòng, được gọi là AddRoundKey, 128-bit của state thực hiện phép XOR từng bit với 128-bit của khóa vòng. Được mô tả ở hình 5b.

Trong giải mã AES, phép biến đổi khóa vòng nghịch đảo giống với phép biến đổi khóa vòng, do phép XOR là phép nghịch đảo của chính nó.

Ví dụ:



Hình 7: Một vòng trong mã hóa AES

2.4 Mở rộng khóa

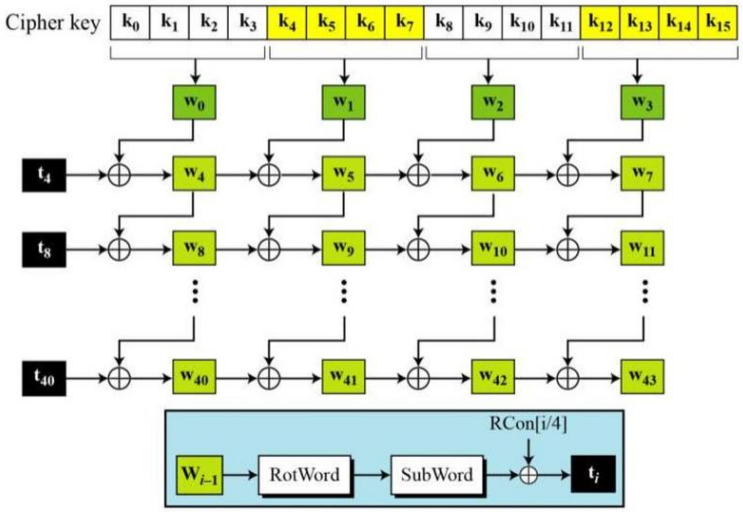
Giả sử trong trường hợp AES 128-bit, bộ mở rộng khóa sẽ sử dụng 128-bit đầu vào, tạo thành các khóa cho 1 pre-round và 10-round. Vậy sẽ tạo ra 44-word khóa từ 4-word (128-bit) khóa đầu vào.

Round	Word			
Pre-round	w_0	w_1	w_2	w_3
1	w_4	w_5	w_6	w_7

2	W 8	W 9	W 10	W 11
…	…	…	…	…
10	W 40	W 41	W 42	W 43

Bảng 3: Phân phối khóa cho các vòng

Quá trình mở rộng khóa được mô ta như hình dưới đây:



Hình 8: Sơ đồ mở rộng khóa

Trong đó, các word có chỉ số chia hết cho 4 (4, 8, 12, …, 40) được tạo ra bởi:

$$t = SubWord(\text{RotWord}(\text{w}[i])) \oplus \text{RCon}[i/4] \oplus w[i-4]$$

Với bảng giá trị RCon:

Round	RCon	Round	RCon
1	$(01\ 00\ 00\ 00)_{16}$	6	$(20\ 00\ 00\ 00)_{16}$
2	$(02\ 00\ 00\ 00)_{16}$	7	$(40\ 00\ 00\ 00)_{16}$
3	$(04\ 00\ 00\ 00)_{16}$	8	$(80\ 00\ 00\ 00)_{16}$
4	$(08\ 00\ 00\ 00)_{16}$	9	$(1B\ 00\ 00\ 00)_{16}$
5	$(10\ 00\ 00\ 00)_{16}$	10	$(36\ 00\ 00\ 00)_{16}$

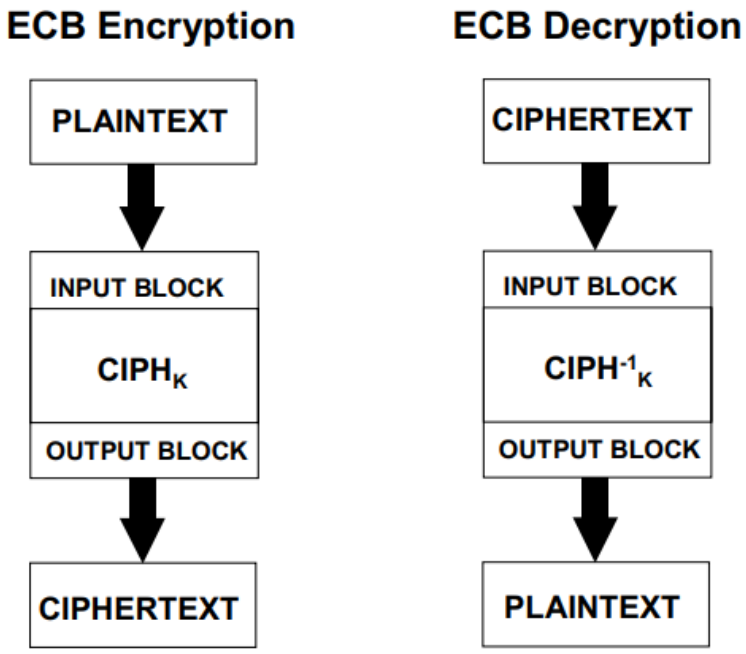
Bảng 4: Giá trị của Rcon

Các phép biến đổi SubWord và RotWord giống như thực hiện SubBytes và ShiftRows, với cùng bảng thay thế cũng như phép dịch, chỉ khác là thực hiện trên word thay vì thực hiện trên State.

2.5 Chế độ ECB (Electronic Codebook Mode):

Chế độ ECB là chế độ bảo mật có tính năng, đối với một khóa nhất định, việc gán khối ciphertext cố định cho mỗi khối plaintext. Chế độ ECB được định nghĩa như sau:

- Mã hóa ECB: $C_J = CIPH_K(P_J)$ với $j = 1 \ldots n$
- Giải mã ECB: $P_J = CIPH_K^{-1}(C_J)$ với $j = 1 \ldots n$



Hình 9: Chế độ ECB

Trong mã hóa ECB, hàm mã hóa được áp dụng trực tiếp và độc lập cho mỗi khối của khối plaintext. Chuỗi kết quả của các khối đầu ra là khối ciphertext.

Trong giải mã ECB, hàm giải mã được áp dụng trực tiếp và độc lập cho mỗi khối của ciphertext. Chuỗi kết quả của các khối đầu ra là plaintext.

Trong mã hóa ECB và giải mã ECB, nhiều hàm mã hóa và hàm giải mã có thể được tính toán song song.

Trong chế độ ECB, dưới một khóa nhất định, bất kỳ khối văn bản rõ ràng nào luôn được mã hóa thành cùng một khối bản mã. Nếu thuộc tính này là không mong muốn trong một ứng dụng cụ thể, thì không nên sử dụng chế độ ECB.

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

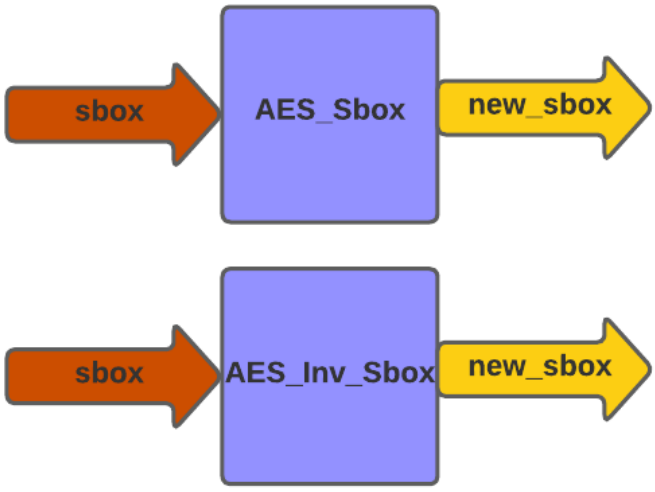
3.1 Các khối trong hệ thống

3.1.1. Khối AES_Sbox và AES_Inv_Sbox

3.1.1.1. Nguyên tắc hoạt động

Khối AES_Sbox và AES_Inv_Sbox như là ROM 256-byte. Nó xử lý song song 32-bit của State. Khối AES_Sbox được sử dụng cho biến đổi SubBytes của bộ mã hóa và biến đổi SubWord của bộ mở rộng khóa. Khối AES_Inv_Sbox được sử dụng cho biến đổi InvSubBytes của bộ giải mã.

3.1.1.2. Sơ đồ khối



Hình 10: Sơ đồ khối AES_Sbox và AES_Inv_Sbox

3.1.1.3. Mô tả tín hiệu vào ra

STT	Tên chân	Chiều	Độ rộng bit	Chức năng
1	sbox	Vào	32	Dữ liệu đưa vào để tra bảng Sbox hoặc Inv_Sbox
2	new_sbox	Ra	32	Dữ liệu ngõ ra

3.1.1.4. Giải thuật

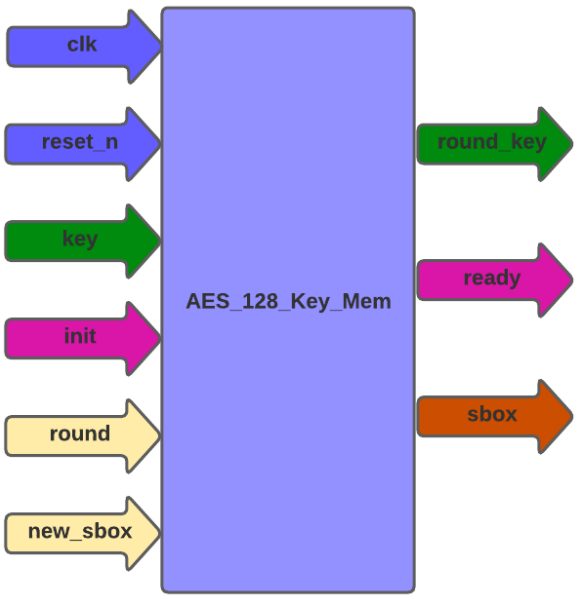
Tạo một mảng S-box gồm 256-byte. Khi tín hiệu đầu vào 32-bit được đưa vào thì xem giá trị đầu vào đó tra trong mảng S-box để xuất cho ngõ ra 32-bit.

3.1.2. Khối AES_128_Key_Mem

3.1.2.1. Nguyên tắc hoạt động

Thực hiện mở rộng khóa. Gồm các biến đổi SubWord, RotWord và tính toán Rcon.

3.1.2.2. Sơ đồ khối



Hình 11: Sơ đồ khối AES_128_Key_Mem

3.1.2.3. Mô tả tín hiệu vào ra

STT	Tên chân	Chiều	Độ rộng bit	Chức năng
1	clk	Vào	1	Clock cung cấp cho khối
2	reset_n	Vào	1	Reset bất đồng bộ
3	key	Vào	128	Dữ liệu khóa đầu vào
4	init	Vào	1	Tín hiệu khởi động khi có dữ liệu khóa đầu vào mới
5	round	Vào	4	Dữ liệu biến đếm của vòng khóa, gồm 1 vòng chuẩn bị và 10 vòng khóa
6	new_sbox	Vào	32	Dữ liệu khóa khi qua biến đổi SubWord
7	round_key	Ra	128	Dữ liệu vòng khóa
8	ready	Ra	1	Tín hiệu khi chuẩn bị xuất dữ liệu round_key
9	sbox	Ra	32	Dữ liệu khóa

3.1.2.4. Giải thuật

Khối AES_128_Key_Mem thực hiện cung cấp lần lượt 4-word (128-bit) khóa cho khối AES_128_Encipher_Block và khối AES_128_Decipher_Block. Khối này sẽ tạo ra 44-word khóa từ 4-word (128-bit) của dữ liệu key đầu vào.

3.1.2.5. Các khối con

reg_update: cập nhật thanh ghi theo tín hiệu xung clock hoặc tín hiệu reset bất đồng bộ và tạo 11 thanh ghi đệm key_mem (128-bit) để xuất cho tín hiệu ngõ ra round_key.

key_mem_read: tạo thanh ghi tmp_key_mem tạm chờ cập nhật.

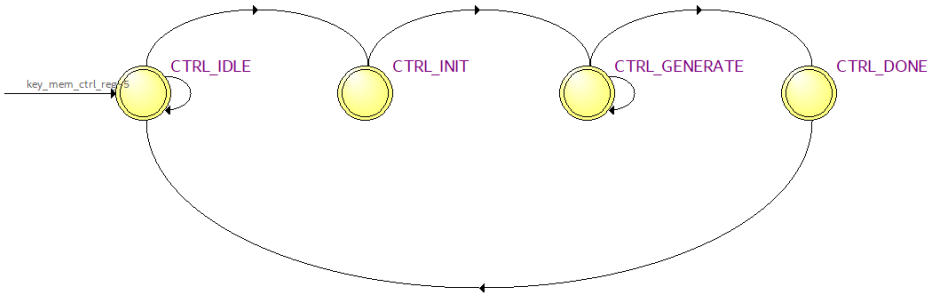
round_key_gen: tạo vòng khóa để thực hiện các biến đổi RotWord, SubWord, phép toán XOR và Rcon.

rcon_logic: tính toán giá trị Rcon cho mỗi lần mở rộng khóa (có 10 giá trị được tạo ra) theo trường hữu hạn GF (2^8).

round_ctr: bộ đếm vòng có chức năng reset và tăng biến đếm.

key_mem_ctrl: máy trạng thái (FSM) để điều khiển vòng khóa gồm các trạng thái:

- **CTRL_IDLE:** trạng thái khởi tạo.
- **CTRL_INIT:** trạng thái khởi động, reset biến đếm.
- **CTRL_GENERATE:** trạng thái cập nhật, tăng biến đếm.
- **CTRL_DONE:** trạng thái hoàn thành, xuất tín hiệu round_key ra và trở về trạng thái khởi tạo.



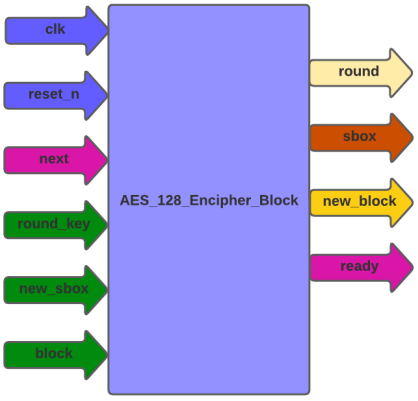
Hình 12: Máy trạng thái của khối AES_128_Key_Mem

3.1.3. Khối AES_128_Encipher_Block

3.1.3.1. Nguyên tắc hoạt động

Thực hiện mã hóa dữ liệu. Gồm các biến đổi SubBytes, ShiftRows, MixColumns và AddRoundKey.

3.1.3.2. Sơ đồ khối



Hình 13: Sơ đồ khối AES_128_Encipher_Block

3.1.3.3. Mô tả tín hiệu vào ra

STT	Tên chân	Chiều	Độ rộng bit	Chức năng
1	clk	Vào	1	Clock cung cấp cho khối
2	reset_n	Vào	1	Reset bất đồng bộ
3	next	Vào	1	Tín hiệu khi có dữ liệu plaintext mới
4	round_key	Vào	128	Dữ liệu khóa đầu vào
5	new_sbox	Vào	32	Dữ liệu khóa khi qua khối S-box
6	block	Vào	128	Dữ liệu plaintext vào
7	round	Ra	4	Dữ liệu biến đếm của vòng khóa, gồm 1 vòng chuẩn bị và 10 vòng khóa
8	sbox	Ra	32	Dữ liệu khóa thực hiện biến đổi SubBytes
9	new_block	Ra	128	Dữ liệu ciphertext ra
10	ready	Ra	1	Tín hiệu khi chuẩn bị có dữ liệu plaintext mới

3.1.3.4. Giải thuật

Khối AES_128_Encipher_Block thực hiện mã hóa thuật toán AES. Khối nhận dữ liệu đầu vào 128-bit block (plaintext) và 128-bit key (cipherkey) và xuất dữ liệu đầu ra 128-bit new-block (ciphertext).

3.1.3.5. Các hàm con

gm02, gm03: thực hiện phép tính nhân 1 số 8-bit với {02} và {03} trong trường hữu hạn GF (2^8).

mixw, mixcolumns: thực hiện phép biến đổi MixColumns theo từng cột 32-bit và tổng hợp tạo thành 1 State 128-bit.

shiftrows: thực hiện phép biến đổi ShiftRows tạo thành 1 State mới.

addroundkey: thực hiện phép biến đổi AddRoundKey bằng cách thực hiện phép XOR của State với dữ liệu round_key có được khi thực hiện khối AES_128_Key_Mem.

3.1.3.6. Các khối con

reg_update: cập nhật thanh ghi theo tín hiệu xung clock hoặc tín hiệu reset bất đồng bộ.

round_logic: cập nhật các thanh ghi tín hiệu.

sword_ctr: bộ đếm SubBytes với chức năng reset và tăng biến đếm.

round_ctr: bộ đếm vòng có chức năng reset và tăng biến đếm.

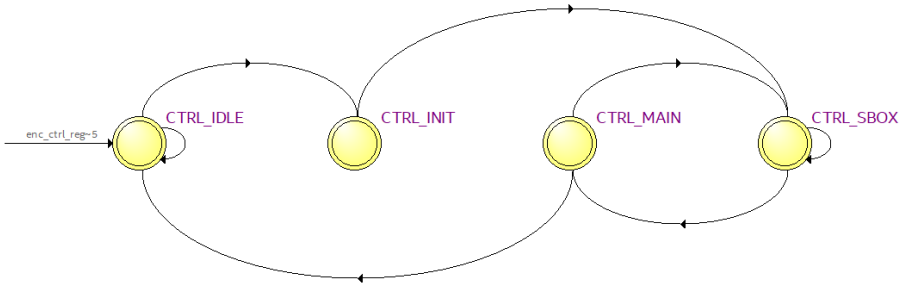
encipher_ctrl: máy trạng thái (FSM) để điều khiển vòng khóa gồm các trạng thái:

– **CTRL_IDLE:** trạng thái khởi tạo, reset biến đếm vòng khi có tín hiệu next tích cực.

– **CTRL_INIT:** trạng thái khởi động, reset biến đếm SubBytes và tăng biến đếm vòng.

– **CTRL_SBOX:** trạng thái khối S-box, tăng biến đếm SubBytes.

– **CTRL_MAIN:** trạng thái chính, reset biến đếm SubBytes và tăng biến đếm vòng.



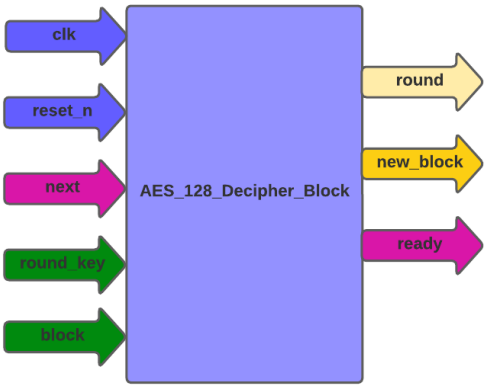
Hình 14: Máy trạng thái của khối AES_128_Encipher_Block

3.1.4. Khối AES_128_Decipher_Block

3.1.4.1. Nguyên tắc hoạt động

Thực hiện giải mã dữ liệu. Gồm các biến đổi SubBytes, ShiftRows, MixColumns và AddRoundKey.

3.1.4.2. Sơ đồ khối



Hình 15: Sơ đồ khối AES_128_Decipher_Block

3.1.4.3. Mô tả tín hiệu vào ra

STT	Tên chân	Chiều	Độ rộng bit	Chức năng
1	clk	Vào	1	Clock cung cấp cho khối
2	reset_n	Vào	1	Reset bất đồng bộ
3	next	Vào	1	Tín hiệu khi có dữ liệu ciphertext mới
4	round_key	Vào	128	Dữ liệu khóa đầu vào
5	block	Vào	128	Dữ liệu ciphertext vào
6	round	Ra	4	Dữ liệu biến đếm của vòng khóa, gồm 1 vòng chuẩn bị và 10 vòng khóa
7	new_block	Ra	128	Dữ liệu plaintext ra
8	ready	Ra	1	Tín hiệu khi chuẩn bị có dữ liệu ciphertext mới

3.1.4.4. Giải thuật

Khối AES_128_Decipher_Block thực hiện giải mã thuật toán AES. Khối nhận dữ liệu đầu vào 128-bit block (ciphertext) và 128-bit key (cipherkey) và xuất dữ liệu đầu ra 128-bit new-block (plaintext).

3.1.4.5. Các hàm con

gm02, gm03, gm04, gm08, gm09, gm11, gm13, gm14: thực hiện phép tính nhân 1 số 8-bit với {02}, {03}, {04}, {08}, {09}, {11}, {13}, {14} trong trường hữu hạn GF (2^8).

inv_mixw, inv_mixcolumns: thực hiện phép biến đổi InvMixColumns theo từng cột 32-bit và tổng hợp tạo thành 1 State 128-bit.

inv_shiftrows: thực hiện phép biến đổi InvShiftRows tạo thành 1 State mới.

addroundkey: thực hiện phép biến đổi AddRoundKey bằng cách thực hiện phép XOR của State với dữ liệu round_key có được khi thực hiện khối AES_128_Key_Mem.

3.1.4.6. Các khối con

reg_update: cập nhật thanh ghi theo tín hiệu xung clock hoặc tín hiệu reset bất đồng bộ.

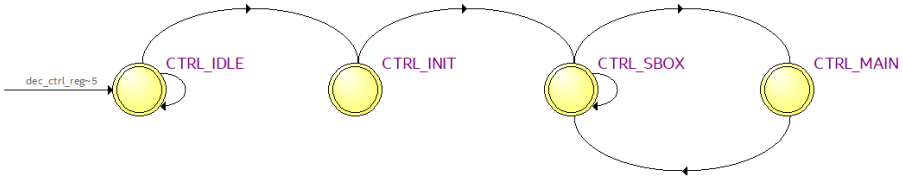
round_logic: cập nhật các thanh ghi tín hiệu.

sword_ctr: bộ đếm SubBytes với chức năng reset và tăng biến đếm.

round_ctr: bộ đếm vòng có chức năng reset và giảm biến đếm.

decipher_ctrl: máy trạng thái (FSM) để điều khiển vòng khóa gồm các trạng thái:

- **CTRL_IDLE:** trạng thái khởi tạo, cập nhật lại biến đếm vòng khi có tín hiệu next tích cực.
- **CTRL_INIT:** trạng thái khởi động, reset biến đếm SubBytes.
- **CTRL_SBOX:** trạng thái khối S-box, tăng biến đếm SubBytes và giảm biến đếm vòng khi thanh ghi biến đếm SubBytes được tăng 4 lần.
- **CTRL_MAIN:** trạng thái chính, reset biến đếm SubBytes và tăng biến đếm vòng.



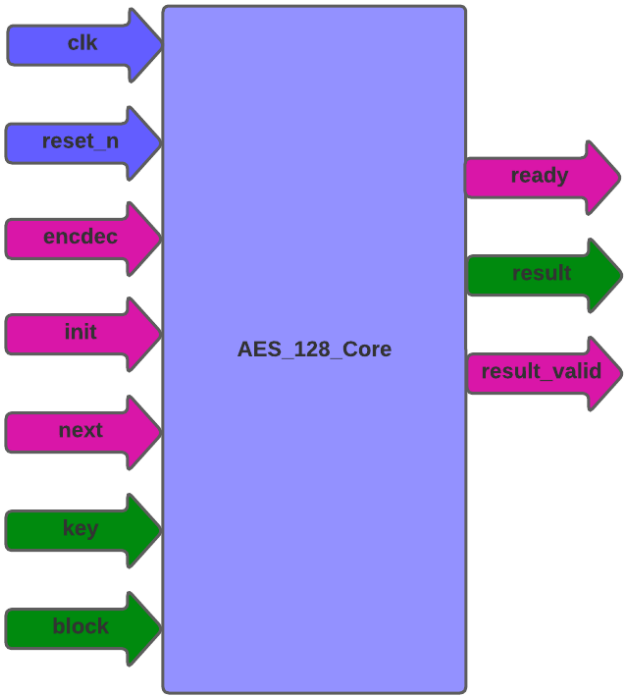
Hình 16: Máy trạng thái của khối AES_128_Decipher_Block

3.1.5. Khối AES_128_Core

3.1.5.1. Nguyên tắc hoạt động

Thực hiện tổng hợp các khối AES_128_Key_Mem, AES_128_Encipher_Block và AES_128_Decipher_Block.

3.1.5.2. Sơ đồ khối



Hình 17: Sơ đồ khối AES_128_Core

3.1.5.3. Mô tả tín hiệu vào ra

STT	Tên chân	Chiều	Độ rộng bit	Chức năng
1	clk	Vào	1	Clock cung cấp cho khối
2	reset_n	Vào	1	Reset bất đồng bộ
3	encdec	Vào	1	Tín hiệu phân biệt giữa bộ mã hóa và bộ giải mã
4	init	Vào	1	Tín hiệu khi có ngõ vào key mới
5	next	Vào	1	Tín hiệu khi có ngõ vào block mới

6	key	Vào	128	Dữ liệu khóa đầu vào
7	block	Vào	128	Dữ liệu plaintext hoặc ciphertext vào
8	ready	Ra	1	Tín hiệu khi chuẩn bị có dữ liệu block hoặc result xuất ra
9	result	Ra	128	Dữ liệu ciphertext hoặc plaintext ra
10	result_valid	Ra	1	Tín hiệu khi chuẩn bị có dữ liệu result xuất ra

3.1.5.4. Giải thuật

Khối AES_128_Core tổng hợp các dữ liệu plaintext hoặc ciphertext của khối mã hóa hoặc giải mã.

3.1.5.5. Các khối con

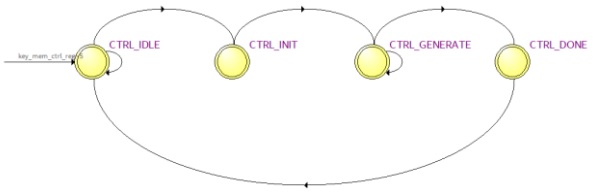
reg_update: cập nhật thanh ghi theo tín hiệu xung clock hoặc tín hiệu reset bất đồng bộ.

sbox_mux: bộ ghép kênh để chọn ngõ vào của khối AES_Sbox.

encdec_mux: bộ ghép kênh để chọn chế độ mã hóa hoặc giải mã.

aes_core_ctrl: máy trạng thái (FSM) để điều khiển vòng khóa gồm các trạng thái:

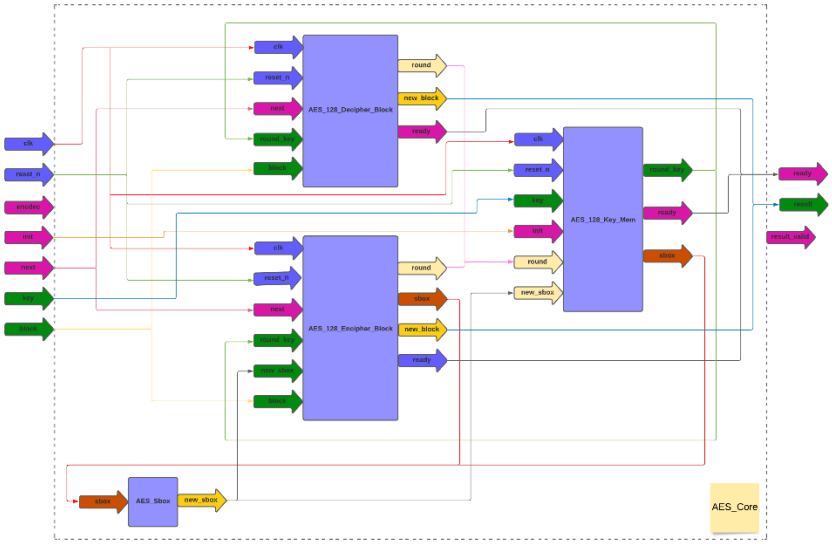
- **CTRL_IDLE:** trạng thái khởi tạo.
- **CTRL_INIT:** trạng thái khởi động.
- **CTRL_NEXT:** trạng thái chính.



Hình 18: Máy trạng thái của khối AES_128_Core

3.2 Kiến trúc hệ thống

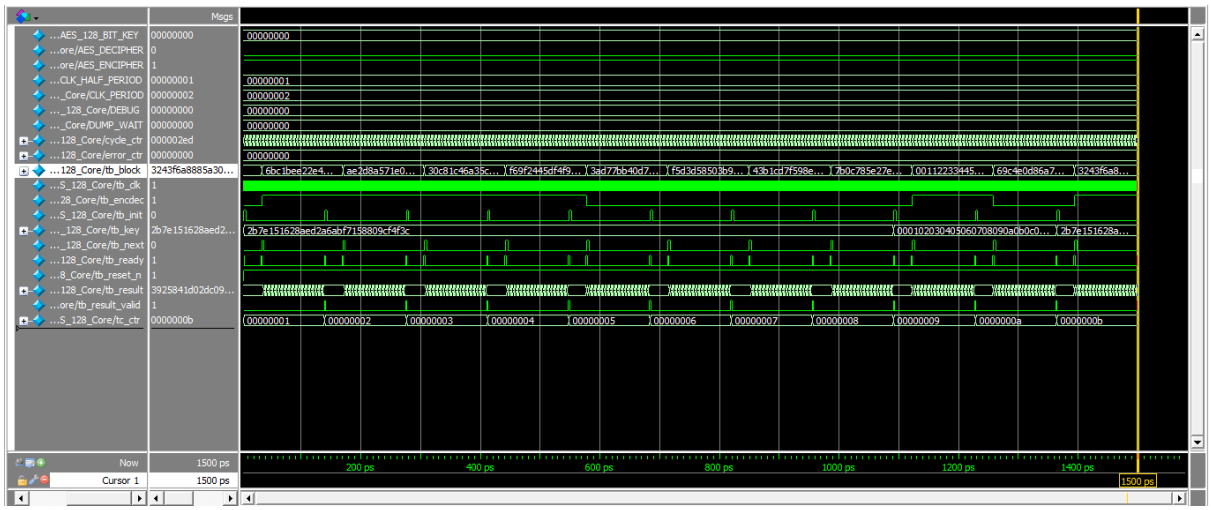
Thực hiện tổng hợp kết nối các khối AES_128_Key_Mem, AES_128_Encipher_Block, AES_128_Decipher_Block và AES_Sbox với nhau tạo thành khối AES_128_Core. Khối này có chức năng như lõi của bộ mã hóa và giải mã của thuật toán AES-128.



Hình 19: Sơ đồ khối chi tiết của AES_128_Core

4. KẾT QUẢ THỰC HIỆN

4.1 Kết quả mô phỏng bằng ModelSim



```
# *** TC 11 successful
#
#
# --- All 11 test cases completed successfully
#
# *** AES core simulation done. ***
# ** Note: $finish    : C:/Users/xuant/Documents/GitHub/AES_128_DoAn/tb/tb_AES_128_Core.v(369)
#    Time: 1500 ps  Iteration: 0  Instance: /tb_AES_128_Core
```

Hình 20: Kết quả mô phỏng ModelSim cho khối AES_128_Core

4.2 Kết quả mô phỏng bằng Quartus

Flow Summary

<<Filter>>

Flow Status	Successful - Fri May 27 11:58:22 2022
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	AES_128_Core
Top-level Entity Name	AES_128_Core
Family	Cyclone V
Device	5CGXFC9E7F35C8
Timing Models	Final
Logic utilization (in ALMs)	1,935 / 113,560 (2 %)
Total registers	1842
Total pins	391 / 616 (63 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

Hình 21: Tài nguyên phần cứng khi mô phỏng Quartus

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Kết quả của thiết kế phần cứng xử lý mã hóa AES đúng với mong muốn ban đầu đặt ra trong chương 1.

5.2 Hướng phát triển

Đề tài này có thể phát triển lên hướng tăng độ rộng bit của key thành 192-bit và 256-bit. Tăng tính bảo mật và phù hợp hơn với thế giới.

6. TÀI LIỆU THAM KHẢO

[1] K. Mateur, M.Alareqi, A. Mezouari, H. Dahou, L. Hlou and R. Elgouri, "Design hard ware implementation of AES algorithm on FPGA board", WITS, 2016.

[2] FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, November 2001 (http://csrc.nist.gov/ publications/fips/fips197/fips-197.pdf).

[3] M. Mali, F. Novak, A. Biasizzo, "Hardware Implementation of AES Algorithm", Journal of ELECTRICAL ENGINEERING, VOL. 56, NO. 9-10, 2005, 265–269.

[4] Shihai Zhu, "Hardware Implementation of AES Encryption and Decryption System Based on FPGA", The Open Cybernetics & Systemics Journal, 2015, 9, 1373-1377.

[5] K. Kumar, K.R. Ramkumar and A. Kaur, "A Design Implementation and Comparative Analysis of Advanced Encryption Standard (AES) Algorithm on FPGA", ICRITO, 2020.

7. PHỤ LỤC

Mã nguồn của đề tài: https://github.com/xuanthi280216/AES_128_DoAn.git