ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

ĐỒ ÁN MÔN HỌC

GIẢI THUẬT MÃ HÓA AES

GIẢNG VIÊN HƯỚNG DẪN

TS.Lâm Đức Khải

SINH VIÊN THỰC HIỆN

Lê Minh Giang	**19521446**
Nguyễn Tấn Thiên	**19522266**
Thượng Hiếu Tâm	**19522165**

TP. HỒ CHÍ MINH, 2022

MỤC LỤC

Chương 1. Advanced Encryption Standard 1
1.1. Giới Thiệu 1
1.2. Ứng dụng 1
1.3. Các Thuật Toán 1
Chương 2. Thuật toán Rijndael 2
2.1. Qui ước. 2
2.1.1. Input and Output 2
2.1.2. Byte 2
2.1.3. Arrays of Bytes 2
2.1.4. The state 3
2.1.5. The State as an Array of Columns 3
2.2. Mô tả thuật toán 4
2.2.1. Mã hóa 4
2.2.2. Key Expansion 8
2.2.3. Giải mã 10
Chương 3. Phần mềm 12
Chương 4. Triển khai phần cứng 13
4.1. Datapath 13
4.2. Key Expansion 14
4.3. Controller 15
4.4. Tài nguyên phần cứng 16
4.5. Tần số tổng hợp trên quartus 17

4.6. Kết quả post_synthesis 17

Tài liệu tham khảo 18

DANH MỤC HÌNH ẢNH

Hình 2-1: Chỉ số cho byte và bit 2
Hình 2-2: State array input and output. 3
Hình 2-3: Mô tả thuật toán 4
Hình 2-4: Áp dụng S-box cho mỗi byte State 5
Hình 2-5: S-box 5
Hình 2-6: ShiftRows dịch chuyển (quay trái) các byte của State. 6
Hình 2-7: Ma trận chuyển đổi sử dụng trong MixColumns 6
Hình 2-8: Minh họa hoạt động MixColumns 7
Hình 2-9: AddroundKey xor mỗi cột của State với từng word của khóa 8
Hình 2-10: Hoạt động của Key Expansion 9
Hình 2-11: Giá trị Rcon 10
Hình 2-12: InvShiftRows dịch chuyển (quay phải) các byte của State 11
Hình 2-13: Inverse S – box 11
Hình 2-14: Ma trận chuyển đổi sử dụng trong InvMixColumns 12
Hình 3-1: Kết quả phần mềm 13
Hình 4-1: Datapath 13
Hình 4-2: Key Expansion 14
Hình 4-3: Controller (1) 15
Hình 4-4: Controller (2) 15
Hình 4-5: Hình ảnh thiết kế 16
Hình 4-6: Tài nguyên phần cứng 16
Hình 4-7: Tần số tổng hợp trên quartus 17
Hình 4-8: Kết quả Post-synthesis 17

Chương 1. Advanced Encryption Standard

1.1. Giới Thiệu

Thuật toán AES (còn được gọi là thuật toán Rijndael) là một thuật toán mật mã khối đối xứng với kích thước khối là 128 bits và chuyển chúng thành bản mã bằng cách sử dụng các khóa 128, 192 và 256 bits

1.2. Ứng dụng

- Bảo mật không dây chống lại tin tặc: WiFi, VPN, mạng riêng ảo,…
- Các phiên trình duyệt được mã hóa để bảo vệ tốt hơn khỏi tin tặc: Whats App, Messenger, Telegram,…
- Mã hóa tập tin chung: WinZip, 7 Zip, Win RAR,…
- Bảo mật bộ xử lý để ngăn chặn tin tặc: Smart card,…

1.3. Các Thuật Toán

- Thuật toán mã hóa MARS
- Thuật toán mã hóa Rivest Cipher 6 (RC 6)
- Thuật toán mã hóa Rijndael
- Thuật toán mã hóa Serpent
- Thuật toán mã hóa Twofish

- Thuật toán mã hóa Rijndael được hai tổ chức NIST và NSA xác định là một thuật toán an toàn nhất để sử dụng trong AES
- Sau khi kiểm tra nghiêm ngặt các thuật toán trên thì thuật toán Rijndael được chọn để sử dụng trong AES
- Việc sử dụng khóa 256 bits mang lại khả năng bảo mật mạnh mẽ, đồng thời duy trì khả năng tương tác của nó với phần cứng và phần mềm hiện có
- Mặc dù tồn tại mật mã mạnh hơn nhưng chúng không có khả năng triển khai vào hệ thống một cách dễ dàng như mật mã Rijndael

Chương 2. Thuật toán Rijndael

2.1. Qui ước.

2.1.1. Input and Output

Mỗi **Input** và **Output** cho thuật toán AES bao gồm **các chuỗi 128 bit** (các chữ số có giá trị 0 hoặc 1).

Khóa mật mã của AES-128 là một chuỗi 128 bit. Các bit trong chuỗi như vậy sẽ được đánh số bắt đầu từ 0 và kết thúc ở một nhỏ hơn độ dài chuỗi (độ dài khóa). Số i được gắn vào một bit được gọi là chỉ số của nó và sẽ nằm trong một trong các phạm vi $0 \leq i < 128$.

2.1.2. Byte

Đơn vị cơ bản để xử lý trong thuật toán AES là một **byte**. Một chuỗi 8 bit được coi như là một thực thể duy nhất.

Chuỗi bit đầu vào và đầu ra và khóa mật mã được xử lý dưới dạng mảng byte được hình thành bằng cách chia chuỗi bit này (input, output, khóa mật mã) thành các nhóm 8 bit(1 byte) thành các mảng byte.

2.1.3. Arrays of Bytes

Mảng byte sẽ được biểu diễn dưới dạng sau:

$$a_0a_1a_2\ldots a_{15}$$

Các byte và thứ tự bit trong byte được lấy từ chuỗi đầu vào 128 bit

$$input_0\ input_1\ input_2\ \ldots\ input_{126}\ input_{127}$$

như sau:

$$a_0 = \{input_0, input_1, \ldots, input_7\};$$
$$a_1 = \{input_8, input_9, \ldots, input_{15}\};$$
$$\vdots$$
$$a_{15} = \{input_{120}, input_{121}, \ldots, input_{127}\}.$$

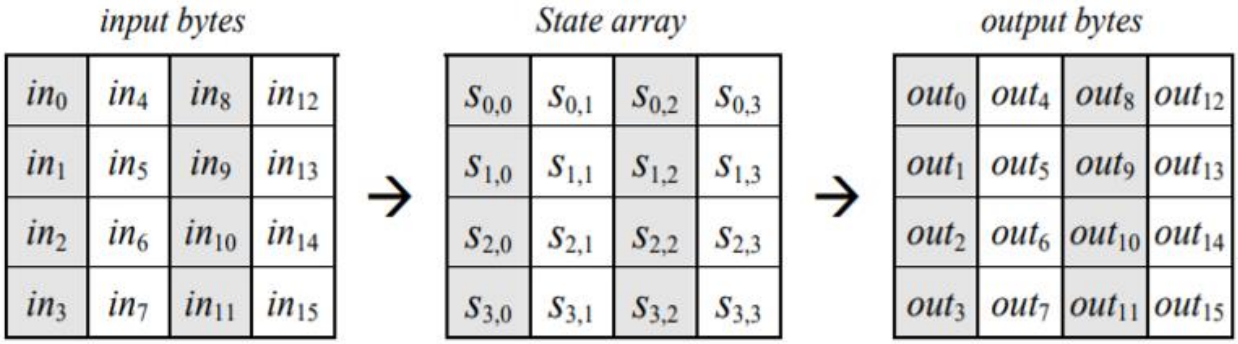
Input bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	…
Byte number	0								1								2								…
Bit numbers in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…

Hình 2-1: Chỉ số cho byte và bit

2.1.4. The state

Các hoạt động của thuật toán AES được thực hiện trên một mảng byte hai chiều được gọi là **State**. State bao gồm bốn hàng byte, mỗi hàng chứa ***Nb*** byte, trong đó ***Nb*** là độ dài khối chia cho 32. Trong mảng State được ký hiệu bằng ký hiệu s, mỗi byte riêng lẻ có hai chỉ số, với số hàng r trong phạm vi $0 \leq r < 4$ và số cột của nó c trong khoảng $0 \leq c < Nb$. Điều này cho phép một byte riêng lẻ của State được gọi là sr, c hoặc s [r, c]. Đối với tiêu chuẩn AES - 128, ***Nb*** = 4, tức là, $0 \leq c < 4$.

Khi bắt đầu Mã hóa và Giải mã, đầu vào - mảng byte in_0, in_1,… in_{15} - được sao chép vào mảng State. Sau đó, các hoạt động Mã hóa hoặc Giải mã được thực hiện trên mảng State này, sau đó giá trị cuối cùng của nó được sao chép đến đầu ra - mảng các byte out_0, out_1,… out_{15}.



Hình 2-2: State array input and output.

Do đó, ở phần đầu của Mật mã hoặc Mật mã nghịch đảo, mảng đầu vào, trong, được sao chép sang mảng Trạng thái theo lược đồ:

$$s[r, c] = in[r + 4c] \text{ for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb$$

và ở cuối Mật mã và Mật mã nghịch đảo, Trạng thái được sao chép vào mảng đầu ra như sau:

$$out[r + 4c] = s[r, c] \text{ for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb.$$

2.1.5. The State as an Array of Columns

Bốn byte trong mỗi cột của mảng State tạo thành các **words** 32 bit, trong đó số hàng r cung cấp chỉ mục cho bốn byte trong mỗi từ. Do đó, State có thể được hiểu là một mảng một chiều gồm các từ (cột) 32 bit, w_0 ... w_3, trong đó số cột c cung cấp một chỉ mục vào mảng này.Do đó, State có thể được coi là một mảng gồm bốn word, như sau:

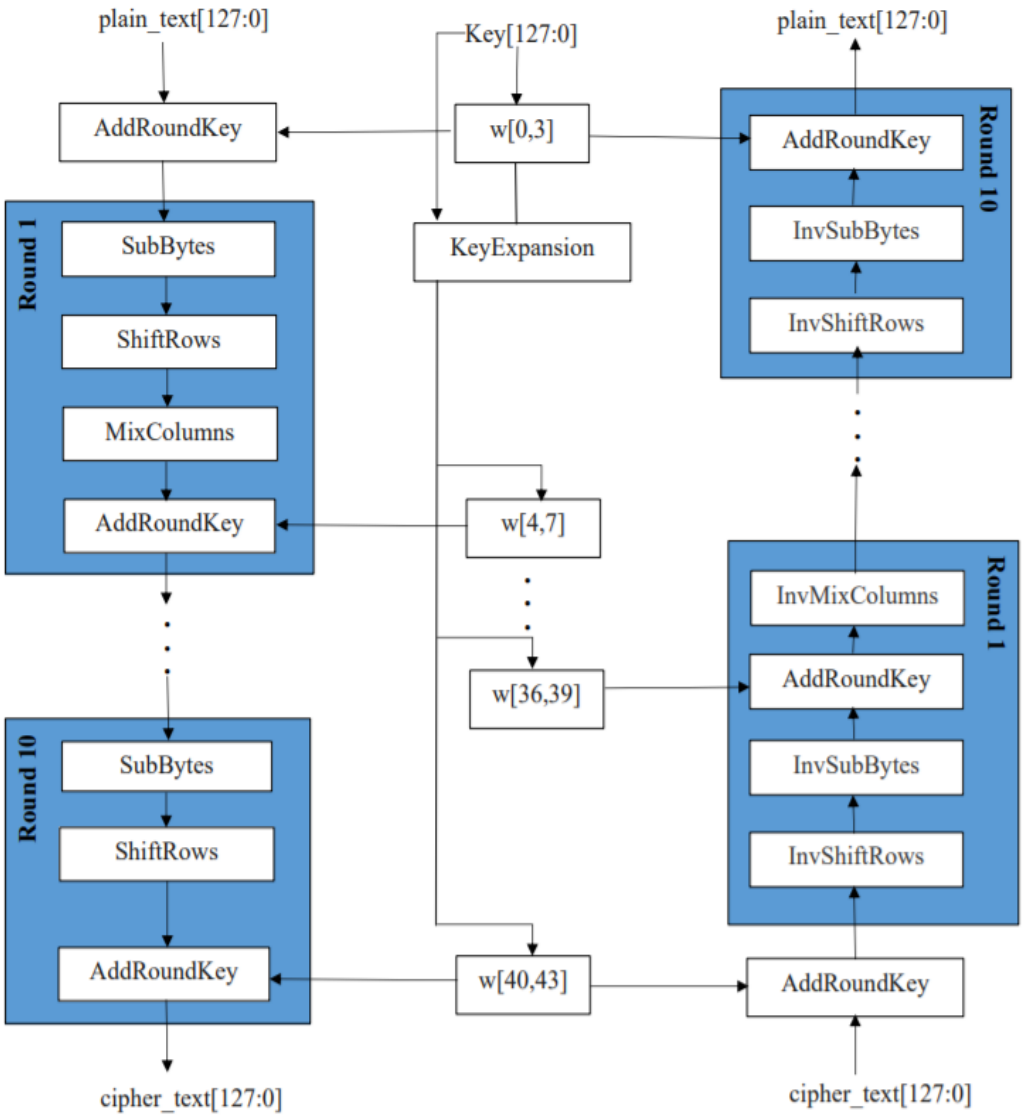
$w_0 = s_{0,0}\ s_{1,0}\ s_{2,0}\ s_{3,0}$

$w_2 = s_{0,2}\ s_{1,2}\ s_{2,2}\ s_{3,2}$

$w_1 = s_{0,1}\ s_{1,1}\ s_{2,1}\ s_{3,1}$

$w_3 = s_{0,3}\ s_{1,3}\ s_{2,3}\ s_{3,3}$.

2.2. Mô tả thuật toán

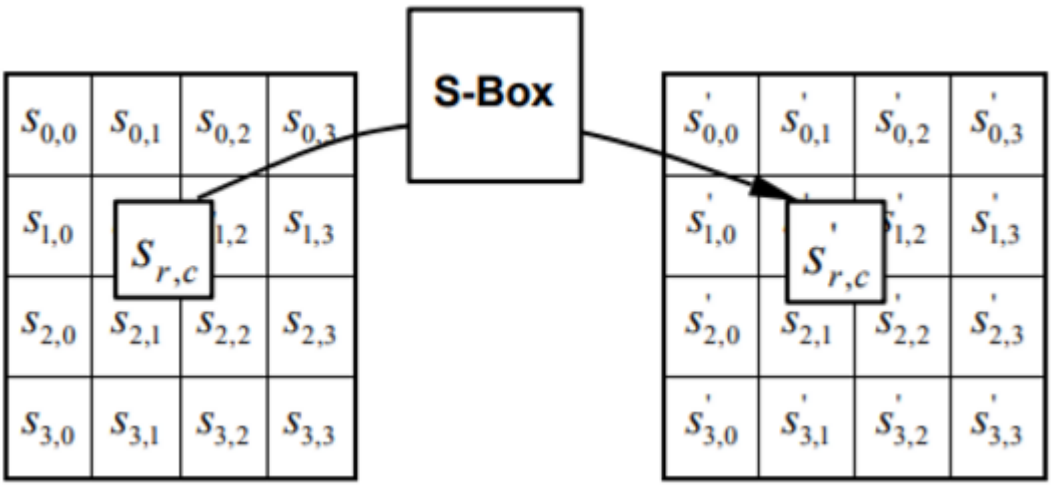


Hình 2-3: Mô tả thuật toán

2.2.1. Mã hóa

2.2.1.1. SubByte

SubBytes là phép thay thế byte phi tuyến tính hoạt động độc lập trên mỗi byte của State bằng cách sử dụng bảng thay thế (S-box).



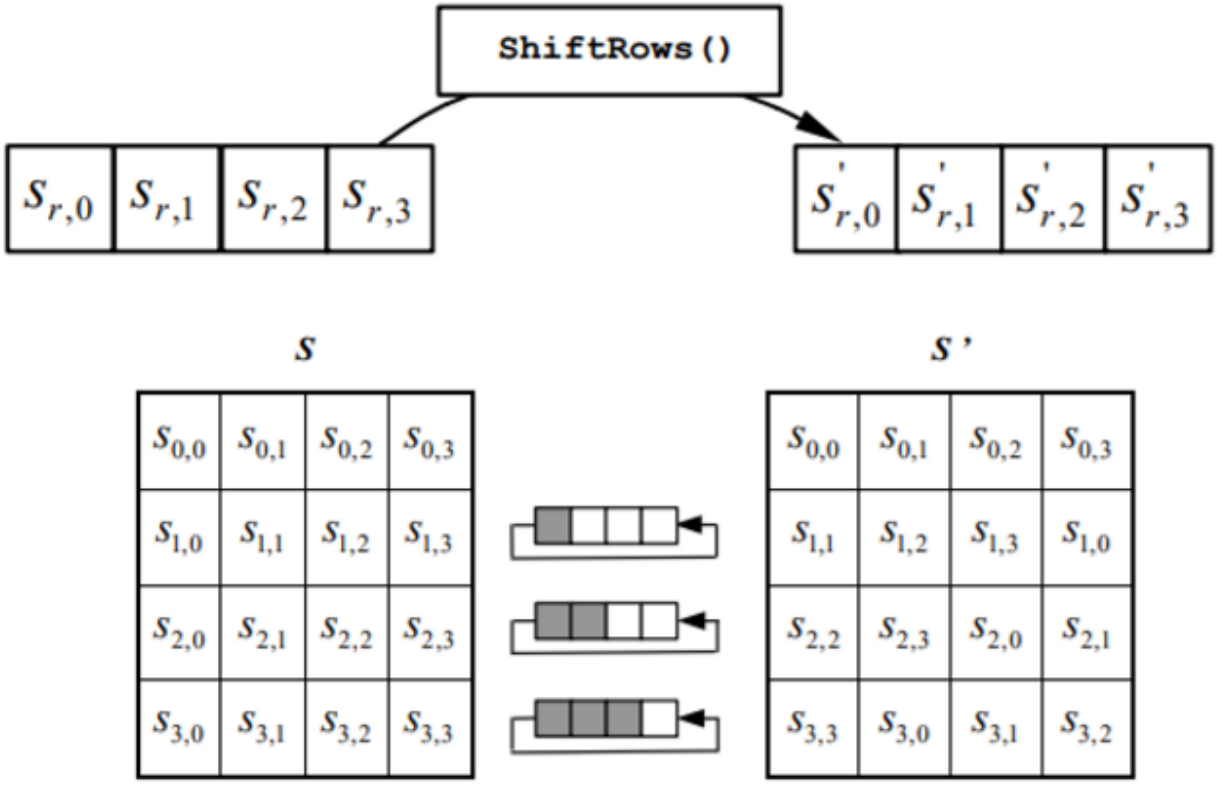
Hình 2-4: Áp dụng S-box cho mỗi byte State

		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 2-5: S-box

2.2.1.2. ShiftRows

ShiftRows dịch chuyển các byte trong ba hàng cuối cùng của Statetheo chu kỳ qua các số byte (hiệu số) khác nhau. Hàng đầu tiên, r = 0, không bị dịch chuyển.



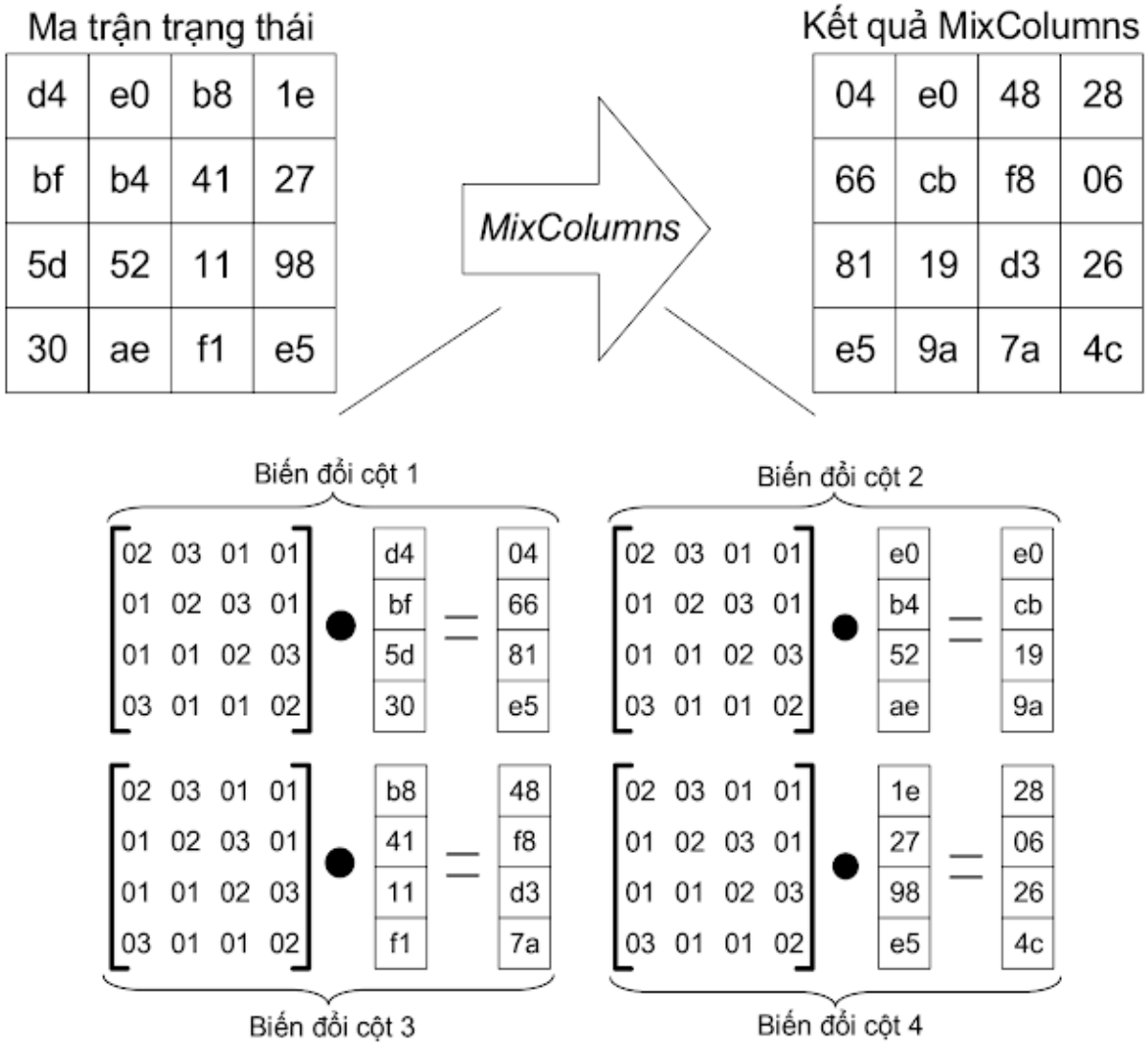
Hình 2-6: ShiftRows dịch chuyển (quay trái) các byte của State.

2.2.1.3. Mixcolumns

MixColumns thực hiện nhân từng cột của ma trận trạng thái (ngõ ra của ShiftRows) với một ma trận chuyển đổi quy định bởi chuẩn AES.

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

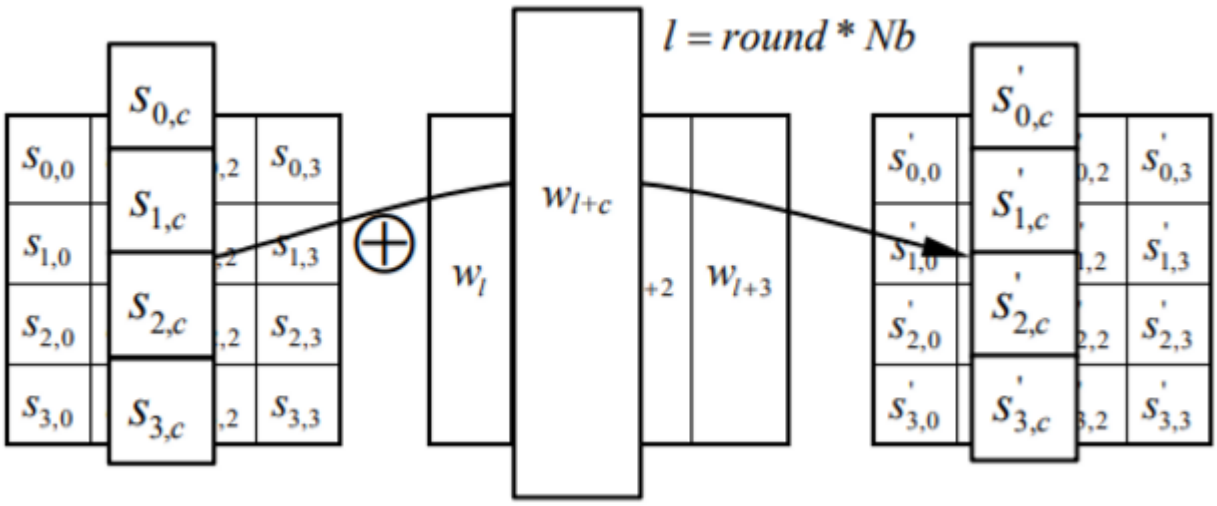
Hình 2-7: Ma trận chuyển đổi sử dụng trong MixColumns



Hình 2-8: Minh họa hoạt động MixColumns

2.2.1.4. AddRoundKey

AddRoundKey là một khóa vòng được thêm vào state bằng phép toán XOR.



Hình 2-9: AddroundKey xor mỗi cột của State với từng word của khóa.

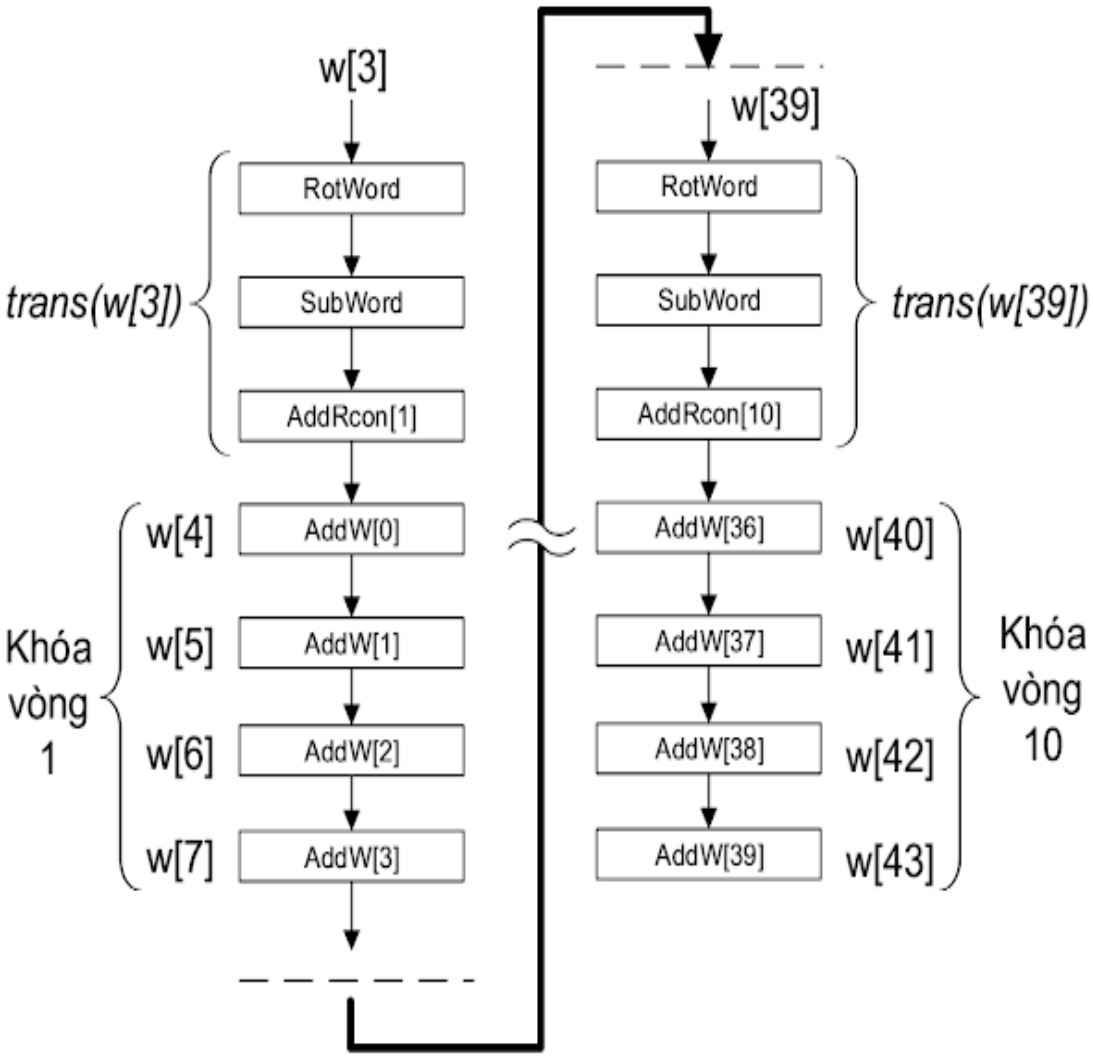
2.2.2. Key Expansion

KeyExpansion thực hiện tính toán khóa vòng cho bước lặp mã hóa và bước tạo ngõ ra. Kết quả của một lần thực thi KeyExpansion là một khóa vòng sử dụng cho chức năng AddRoundKey.

KeyExpansion được thực hiện thông qua 4 chức năng là RotWord, SubWord, AddRcon và AddW.

Mỗi khóa vòng có 128 bit được chia làm 4 word, mỗi word là 4 byte và ký hiệu là w[j] với j là số nguyên.

Mã hóa AES-128 có 1 khóa mã và 10 khóa vòng nên tổng số word là 44 và được đánh số từ 0 đến 43.



Hình 2-10: Hoạt động của Key Expansion

- **RotWord** thực hiện quay trái từ w[j] một byte
- **SubWord** thực hiện thay thế từng byte của kết quả RotWord theo bảng S-box.
- **AddRcon** thực hiện XOR kết quả SubWord và giá trị Rcon[j/4] với j là bội số của 4.
- Số lượng giá trị Rcon[j/4] là 10 tương ứng với 10 lần tính khóa vòng. Chức năng AddRcon sẽ tạo ra kết quả cuối cùng của biến đổi trans(w[j-1]).

Rcon[j/4]	Giá trị HEX	Vị trí sử dụng
Rcon[1]	01000000	sử dụng cho trans(w[3]) khi tính w[4]
Rcon[2]	02000000	sử dụng cho trans(w[7]) khi tính w[8]
Rcon[3]	04000000	sử dụng cho trans(w[11]) khi tính w[12]
Rcon[4]	08000000	sử dụng cho trans(w[15]) khi tính w[16]
Rcon[5]	10000000	sử dụng cho trans(w[19]) khi tính w[20]
Rcon[6]	20000000	sử dụng cho trans(w[23]) khi tính w[24]
Rcon[7]	40000000	sử dụng cho trans(w[27]) khi tính w[28]
Rcon[8]	80000000	sử dụng cho trans(w[31]) khi tính w[32]
Rcon[9]	1b000000	sử dụng cho trans(w[35]) khi tính w[36]
Rcon[10]	36000000	sử dụng cho trans(w[39]) khi tính w[40]

Hình 2-11: Giá trị Rcon

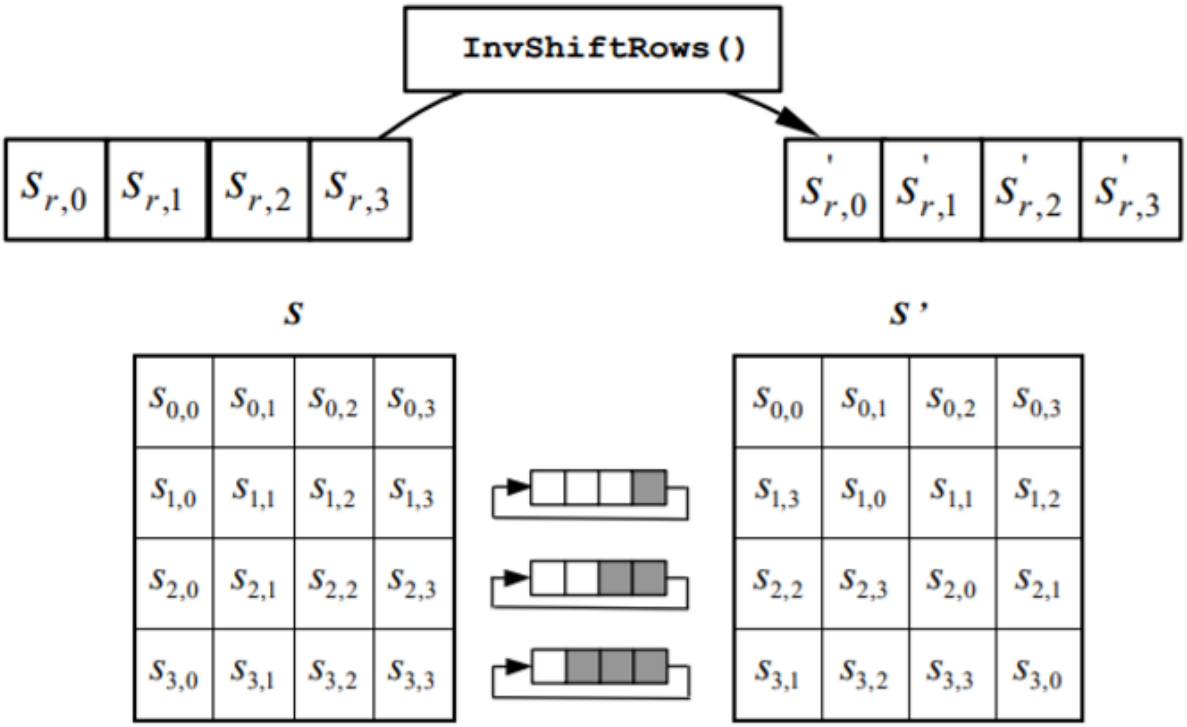
2.2.3. Giải mã

AddRoundKey của Giải mã giống với **AddRoundKey** của Mã hóa.

2.2.3.1. InvShiftRows

InvShiftRows là nghịch đảo của **ShiftRows.**

InvShiftRows dịch chuyển các byte trong ba hàng cuối cùng của Statetheo chu kỳ qua các số byte (hiệu số) khác nhau. Hàng đầu tiên, r = 0, không bị dịch chuyển.



Hình 2-12: InvShiftRows dịch chuyển (quay phải) các byte của State

2.2.3.2. InvSubBytes

InvSubBytes là nghịch đảo của đổi thay thế byte, trong đó Inverse S - box được áp dụng cho mỗi byte của State.

		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 2-13: Inverse S – box

2.2.3.3. InvMixColumns

InvMixColumns là nghịch đảo của phép biến đổi **MixColumns.**

$$\begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$

Hình 2-14: Ma trận chuyển đổi sử dụng trong InvMixColumns

2.2.3.4. InvKeyExpansion

Trong AES-128, chúng ta sẽ có một khóa mã và 10 khóa vòng.

Khi mã hóa, các khóa được sử dụng theo thứ tự như sau:

Khóa mã (key) à khóa vòng 1 à khóa vòng 2 à ... à khóa vòng 10

Khi giải mã theo phương pháp nghịch đảo (Inverse cipher), các khóa được sử dụng theo thứ tự như sau:

Khóa vòng 10 à khóa vòng 9 à ... à khóa vòng 1 à khóa mã (key)

Chương 3. Phần mềm

Sử dụng ngôn ngữ lập trình Python xây dựng các hàm phù hợp với Datapath, thực thi giải thuật mã hóa và giải mã, dùng để so sánh kết quả khi thực thi với phần cứng.

Kế hoạch kiểm tra kết quả mô phỏng:

- Xây dựng 2 tệp input, bao gồm Plain_text và Key mỗi tệp gồm n (do mình mong muốn) hàng, mỗi hàng gồm 32 bits hexadecimal.
- Cứ mỗi hàng sẽ tạo thành một input cho thiết kế, từ đó mã hóa ra Cipher_text theo chuẩn AES-128 bits.
- Tiếp tục dùng Cipher_text và khóa vòng thứ 10 để giải mã, sẽ thu được kết quả là Plain_text ban đầu
- Như vậy, nếu đúng hết toàn bộ tệp input thì tính chính xác của thiết kế theo thuật toán được đảm bảo.

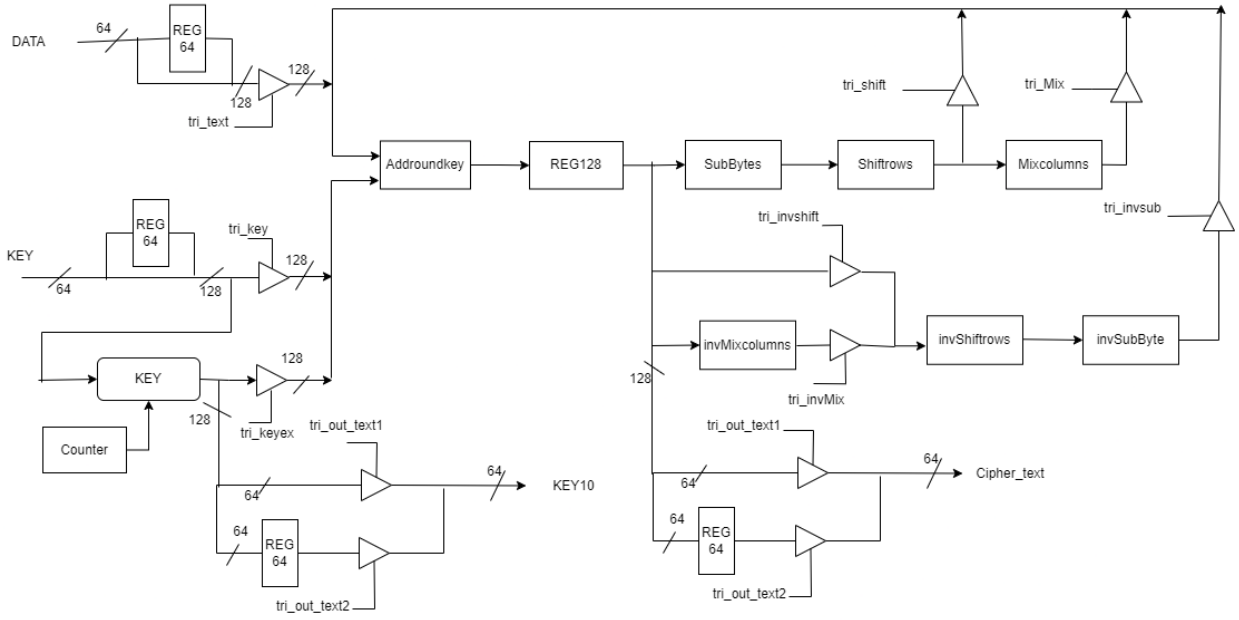
```
In [1]: runfile('D:/HK5/CE213/LyThuyet/AES_Software/AES.py', wdir='D:/HK5/CE213/
LyThuyet/AES_Software')
The two texts are identical!
```

Hình 3-1: Kết quả phần mềm

Kết luận: Sau khi so sánh file text kết quả từ phần mềm và file text kết quả từ ModelSim chạy mô phỏng Post-synthesis thì hoàn toàn giống nhau.

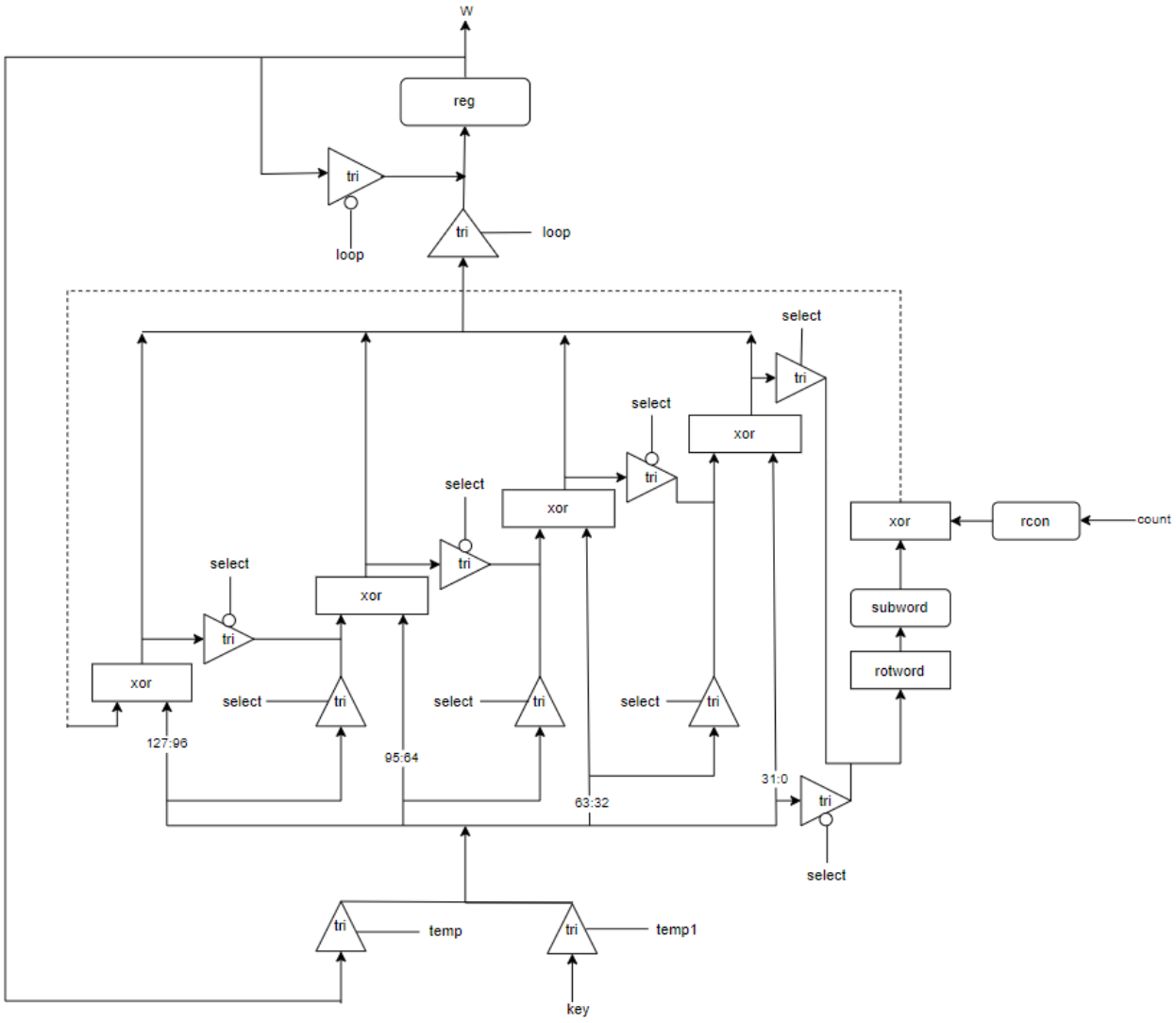
Chương 4. Triển khai phần cứng

4.1. Datapath



Hình 4-1: Datapath

4.2. Key Expansion



Hình 4-2: Key Expansion

4.3. Controller

	start/select				0 = keyex 1 = invkey							
	'00	'01	'10	'11	e	load	rst	tri_text	tri_key	tri_keyex	tri_shift	tri_Mix
S0	S0	S0	S1	S4	0	0	0	0	0	0	0	0
S1	S2	S2	S2	S2	0	1	1	1	1	0	0	0
S2	if (count < 10) S2 else if (count = 10) S3				0	0	1	0	0	1	0	1
S3	S8	S8	S8	S8	0	0	1	0	0	1	1	0
S4	S5	S5	S5	S5	1	1	1	1	1	0	0	0
S5	S6	S6	S6	S6	1	0	1	0	0	1	0	0
S6	if (count >= 2 && count <=9) S6 else S7				1	0	1	0	0	1	0	0
S7	S8	S8	S8	S8	1	0	1	0	0	1	0	0
S8	S9	S9	S9	S9	1	0	1	0	0	0	0	0
S9	S0	S0	S0	S0	1	0	1	0	0	0	0	0

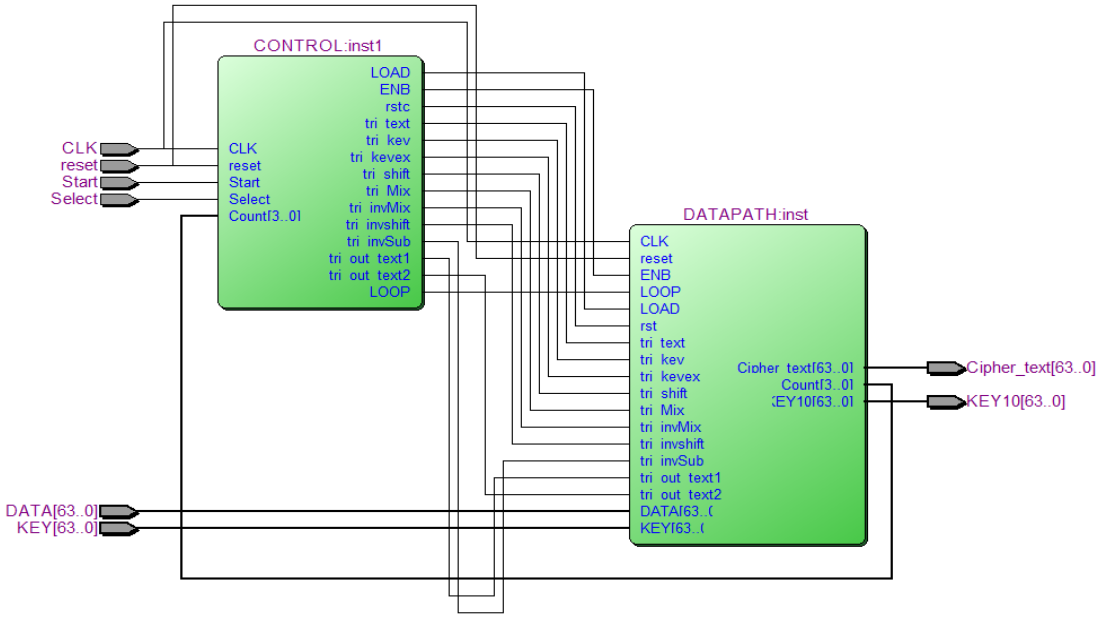
Hình 4-3: Controller (1)

	start/select									
	'00	'01	'10	'11	tri_invMix	tri_invshift	tri_invSub	loop	tri_out_text	tri_out_text1
S0	S0	S0	S1	S4	0	0	0	1	0	0
S1	S2	S2	S2	S2	0	0	0	1	0	0
S2	if (count < 10) S2 else if (count = 10) S3				0	0	0	1	0	0
S3	S8	S8	S8	S8	0	0	0	0	0	0
S4	S5	S5	S5	S5	0	0	0	1	0	0
S5	S6	S6	S6	S6	0	1	1	1	0	0
S6	if (count >= 2 && count <=9) S6 else S7				1	0	1	1	0	0
S7	S8	S8	S8	S8	1	0	1	0	0	0
S8	S9	S9	S9	S9	0	0	0	0	1	0
S9	S0	S0	S0	S0	0	0	0	0	0	1

Hình 4-4: Controller (2)

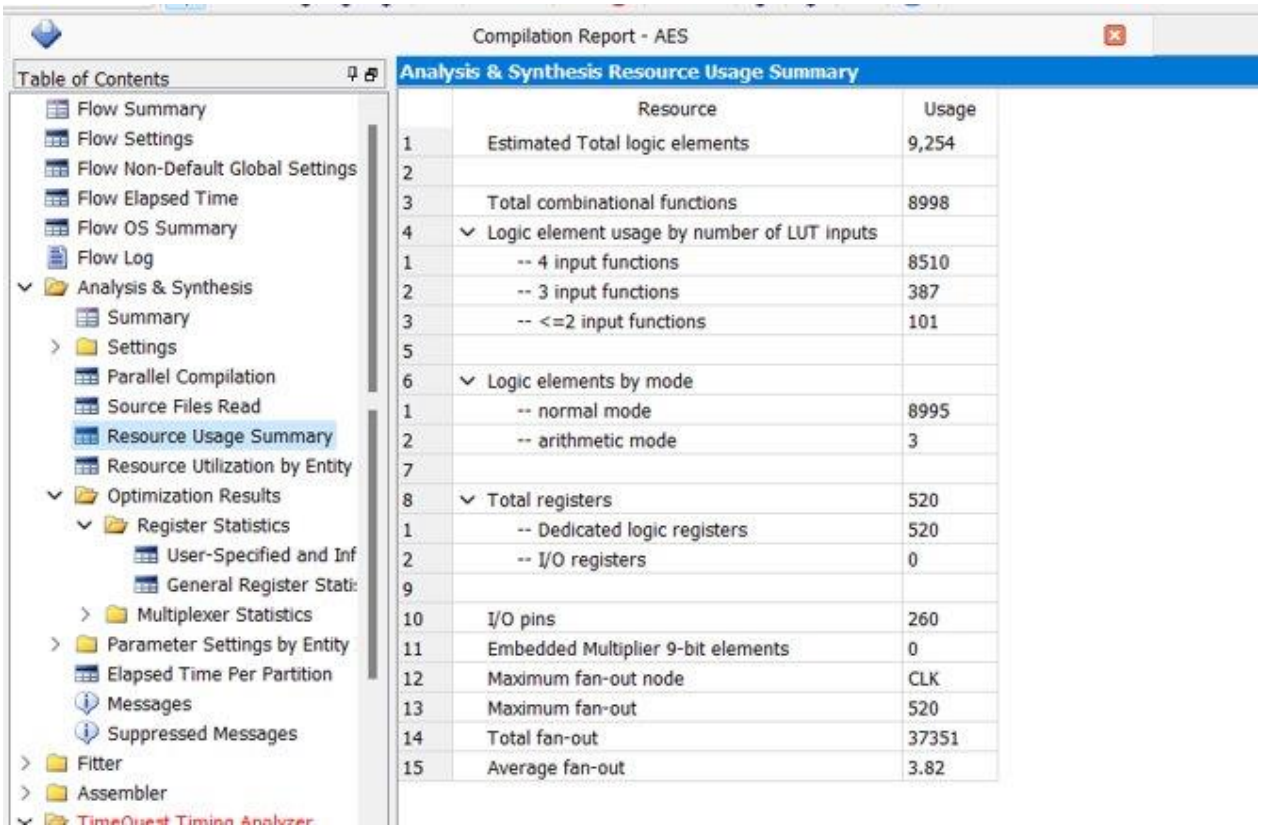
Chức năng của từng trạng thái:

- S0: Đưa 64 bits cao của Plain_text và Key vào Register chờ tín hiệu Start
- S1: Đưa 64 bits thấp của Plain_text và Key, tính bước khởi tạo của mã hóa
- S2: Thực hiện vòng lặp của mã hóa 9 lần
- S3: Thực hiện bước tạo output cho mã hóa
- S4: Đưa 64 bits thấp của Plain_text và Key vào và tính bước khởi tạo của giải mã
- S5: Thực hiện 1 lần bước lặp giải mã
- S6: Thực hiện vòng lặp giải mã 8 lần
- S7: Thực hiện bước tạo output cho giải mã
- S8: Xuất 64 bits cao của Cipher_text và khóa vòng thứ 10.
- S9: Xuất 64 bits thấp của Cipher_text và khóa vòng thứ 10.



Hình 4-5: Hình ảnh thiết kế

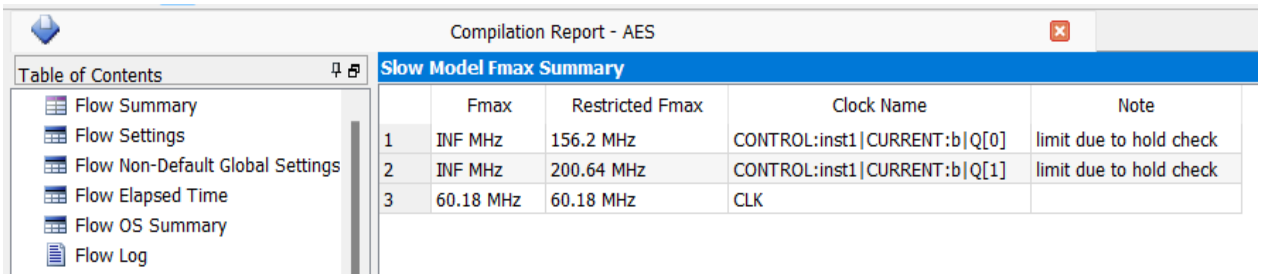
4.4. Tài nguyên phần cứng



	Resource	Usage
1	Estimated Total logic elements	9,254
2		
3	Total combinational functions	8998
4	Logic element usage by number of LUT inputs	
1	-- 4 input functions	8510
2	-- 3 input functions	387
3	-- <=2 input functions	101
5		
6	Logic elements by mode	
1	-- normal mode	8995
2	-- arithmetic mode	3
7		
8	Total registers	520
1	-- Dedicated logic registers	520
2	-- I/O registers	0
9		
10	I/O pins	260
11	Embedded Multiplier 9-bit elements	0
12	Maximum fan-out node	CLK
13	Maximum fan-out	520
14	Total fan-out	37351
15	Average fan-out	3.82

Hình 4-6: Tài nguyên phần cứng

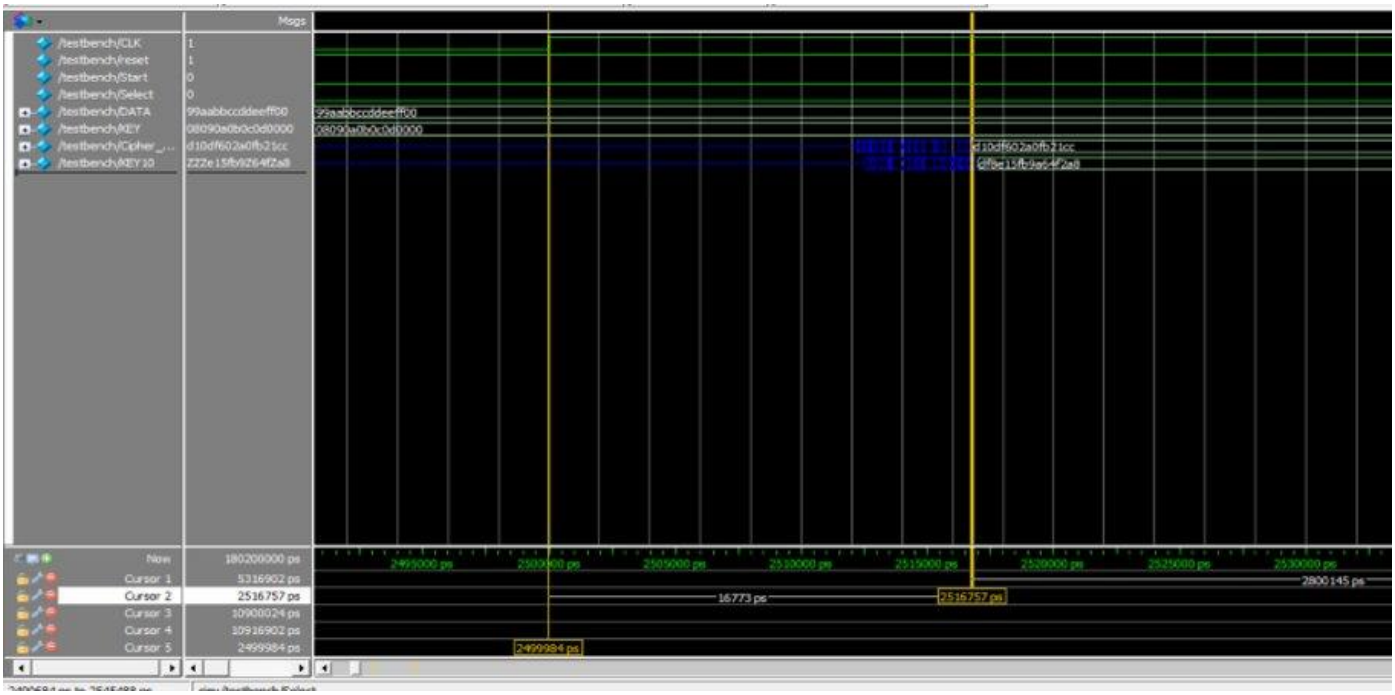
4.5. Tần số tổng hợp trên quartus



	Fmax	Restricted Fmax	Clock Name	Note
1	INF MHz	156.2 MHz	CONTROL:inst1|CURRENT:b|Q[0]	limit due to hold check
2	INF MHz	200.64 MHz	CONTROL:inst1|CURRENT:b|Q[1]	limit due to hold check
3	60.18 MHz	60.18 MHz	CLK	

Hình 4-7: Tần số tổng hợp trên quartus

4.6. Kết quả post_synthesis



Hình 4-8: Kết quả Post-synthesis

Tần số : 1/16,773*10^3 = 59,6 MHz

Quá trình mã hóa và giải mã mỗi quá trình tốn 11 chu kỳ để thực thi xong

Tài liệu tham khảo

- Reference: [1] Federal Information, Processing Standards Publication 197; *ADVANCED ENCRYPTION STANDARD (AES)*; November 26, 2001. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
- [2] St Denis, T., & Johnson, S. (2007). *Advanced Encryption Standard. Cryptography for Developers, 139–202.* Advanced Encryption Standard.pdf
- [3] Wright, M. A. (1999). *The evolution of the Advanced Encryption Standard. Network Security, 1999(11), 11–14.* Advanced Encryption Standard