# 第一章 緒論

## 1.1 研究動機

1960 年代美國國防部，成立了一個 ARPA 單位。ARPA 建設了 ARPAnet 實驗性網路，提供資料能夠在各種機型及廠牌的電腦互相通訊，為最早網際網路發展起源。1970 年代末期，加州柏克萊大學將 TCP/IP 架構植入 unix 機器。網際網路由於 TCP/IP 架構的出現日趨成熟, 這種以資訊技術為中心的技術革命，對人類的經濟、社會與文化均產生革命性的影響。隨著網際網路的普及深入社會，我們面臨病毒、駭客攻擊及木馬程式之入侵，因此架構網路安全環境亦成為不可輕忽的問題。

在過去，密碼學使用在重要的軍事行動來保護資料隱秘性，如在間諜和反間諜間或外交官和總部聯繫之間等。在近幾十年，密碼學可使用的場合越來越廣；它甚至成為網路安全的基礎工具，用來提昇資料的安全性，保護重要的資料。

DES 是一種秘密金鑰加密法(secret-key block cipher)，它的明文長度及密文長度皆為 64 位元(bits)，秘密金鑰長度為 56 位元。為美國政府於 1977 年公佈的加密演算法，但由於最近密碼技術及電腦硬體的快速發展，使得 DES 的安全性受到嚴重的挑戰。美國政府於 1997 年 1 月開始對外昭告徵求 AES 標準[1]後，1999 年 3 月經由 15 種加密標準中初步選出 5 種，在 2000 年 10 月 2 日 NIST 宣佈 AES 獲選演算法為 Rijndael[2]。

## 1.2 研究方向

在 AES 先進加密演算法研究中，一般有分軟體研究及硬體研究兩大部分。在硬體實現方面，特殊應用 IC(ASIC; application-speific integrated circuit)是指依特定用途而設計的特殊規格邏輯 IC。若以數位實作的方式來區分分為 Full-custom ASIC 及 semi-custom ASIC. Full-custom ASIC 一般是依客戶所需要特

定規格，製作不同的元件。所以開發的時間較長，但也因全訂製流程未能反應市場快速多變需求，所以很少使用在電子產品上。semi-custom ASIC 在市場最多使用在標準元件電路(standard cells), 可程式化邏輯電路(FPGA)及可規劃邏輯元件(PLD)。可規劃邏輯元件(PLD)應用在小型電子產品上，並不需要加入時序電路; 可程式化邏輯電路(FPGA)最主要應用在彈性及需要縮短電子產品製作時間; 標準元件電路(standard cells)比可程式化邏輯電路(FPGA)開發時間較長，但性能目標較好。

一般來說，硬體晶片設計有三種目的，第一種目的速度快，硬體電路速度越快，但通常拌隨而來的是電力功率(power)升高。第二種目的電力功率低，愈節省電力功率。第三種目的減少晶片上的閘道數目(Gate counts)。

## 1.3 章節概要

本論文主要在研究如何將先進加密演算法(AES)實現成晶片，論文共分為 6 章，簡介如下:

第一章主要說明論文研究動機，研究方向及各章節介紹。第二章主要介紹 AES 標準規範。第三章詳細描述本論文新提出的電路架構-平行架構整合式查表法，使用唯讀記憶體、互斥或閘(xor)及多工器取代乘法器，實現 AES-128 加密及解密電路。第四章設計模擬驗證，使用 Matlab 建立 AES 演算法平台作功能模擬; RTL 驗證方面，首先使用 Modelsim 作時序驗證;再利用 Synopsys 的 design analyzer 軟體來完成電路合成及驗證。第五章為實驗結果及跟其它論文比較結果。第六章為本文總結以及未來研究方向。

# 第二章 NIST AES 標準規範

## 2.1 AES 演算法參數[3]

AES 演算法共有 3 個重要的參數:

1. 加解密區塊數目(Nb) :欲加解密的資訊，以 32bits(=1word)為一區塊單位。

2. 金鑰區塊數目(Nk) ：金鑰長度, 以 32bits (=1word)為一區塊單位。

3. 運算回合次數(Nr)：加密及解密編解碼的運算回合次數。

依 AES 標準規範，若依金鑰輸入長度分類，可分:AES-128,AES-192 及 AES-256。

**表 2.1 運算回合數$N_r$與$N_b$和$N_k$之關係**

| | 金鑰區塊數目 (Nk) | 加解密區塊數目 (Nb) | 運算回合次數 (Nr) |
|---|---|---|---|
| **AES-128** | **4** | **4** | **10** |
| **AES-192** | **6** | **4** | **12** |
| **AES-256** | **8** | **4** | **14** |

AES標準規範欲加解密的資訊長度都等於 128 位元,但金鑰長度則可以有 128 位元, 192 位元或 256 位元, 可以供選擇。從表 2.1，運算回合數$N_r$與$N_b$和$N_k$之關係，我們可以得知運算回合次數，是由金鑰長度所決定，金鑰長度愈長，運算回合次數愈多。並可以得到如下關係式:

$$Nr=6+Nk \qquad (2\text{-}1)$$

**AES-128**：金鑰區塊數目等於 4, 加解密區塊數目等於 4，運算回合數等於 10。

**AES-192**：金鑰區塊數目等於 6, 加解密區塊數目等於 4，運算回合數等於 12。

**AES-256**：金鑰區塊數目等於 8, 加解密區塊數目等於 4，運算回合數等於 14。

## 2.2 AES 加密演算法的運算函數[3]

AES 演算法加密演算步驟如下: 在執行第一個回合之前，先把明文與初始金鑰經過 AddRoundKey 運算。再經過(Nr-1)回合運算，每回合會運用到的四個函數：SubBytes、ShiftRows、MixColumns 及 AddRoundKey 。最後一個回合運算和其他回合是不一樣的，省略 MixColumns 運算，只經過 SubBytes、ShiftRows 及 AddRoundKey 運算。

### 2.2.1 SubBytes 函數

SubBytse 函數轉換是將狀態矩陣(state)，經過下列兩個步驟:

1. 首先對狀態矩陣每一 byte 求出其在有限場乘法反元素。
2. 將第一步運算結果，經仿射轉換(affine transformation)，如下列數學公式 2-2。

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (2\text{-}2)$$

上述兩步驟，可以簡化成S-box表格(表 2.2)。S-box 是一個包含了 256 個byte數值的表格；查詢時我們將每個byte的最高 4 個位元拿來當作列的索引，每個byte的最低 4 個位元當作行的索引，查出所對應的數值。例如:若$s_{3,0}=(57)_{16}$，最高 4 個位元為 5, 所以查表第 5 列; 最低 4 個位元為 7, 所以查表第 7 行，利用S-box(表 2.2)查第 5 列第 7 行對應到$(5b)_{16}$，因此我們知道經S-box轉換可以得到$s'_{3,0}=(5b)_{16}$。

表 2.2 S-box 位元轉換對照表

| | Y | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | | 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 |

### 2.2.2 ShiftRows 左旋轉位移函數

ShiftRows 函數為一向左旋轉位移函數。Shiftrows 就是將狀態矩陣的每一列分別做不同程度的旋轉位移。第一列不做任何動作外，第二列向左旋轉位移一個位元組(byte)，第三列向左旋轉位移兩個位元組(byte)，第四列向左旋轉位移三個位元組(byte)。

### 2.2.3 MixColumns 函數

MixColumns 是將狀態矩陣的每一行是被視為在GF($2^8$)中的多項式，乘上一固定多項式$\alpha$(x)={03}$x^3$+{01}$x^2$+{01}x+{02}之後，如果發生溢位則同餘($x^4$+1)。

我們可將其簡化為矩陣乘法，令$s'(x)=\alpha(x)\otimes s(x)$

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad (2\text{-}3)$$

展開得到如下式子:

$$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$

$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$$

$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$$

### 2.2.4 AddRoundKey 函數

AddRoundKey 主要運算是將狀態矩陣(state)與每回合運算出來的子金鑰執行互斥或閘的運算。每回合子金鑰產生是經由初始密鑰經過金鑰排程(key schedule)所產生。

## 2.3 AES 解密演算法的運算函數[3]

AES 演算法解密使用的回合子金鑰，與加密使用的回合子金鑰相同，只是順序相反。AES 演算法解密演算步驟如下: 在執行第一個回合之前，先將密文與回合子金鑰執行 AddRoundKey 運算。再經過(Nr-1)回合運算(round)，每回合會運用到的四個函數：InvShiftRows、InvSubBytes、AddRoundKey 及 InvMixColumns。最後一個回合運算和其他回合是不一樣的，省略 InvMixColumns 運算，只經過 InvSubBytes、InvShiftRows 及 AddRoundKey 運算。

### 2.3.1 InvSubBytes 函數

InvSubBytes 函數轉換可經由查 Inverse S-box 表格(如表 2.4)得到。

表 2.3 Inverse S-box 位元轉換對照表

| | Y | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | | 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 | af | 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 |

Inverse S-box 表格產生如下過程:

InvSubByte 函數轉換是將狀態矩陣(state)每一位元組(byte),經過下列兩個步驟完成:

1. 首先將狀態矩陣每一位元組(byte)乘以一個反轉換陣列，如下列數學公式 2-4。

2. 求出其乘法反元素。

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \left( \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \qquad (2\text{-}4)$$

## 2.3.2 InvShiftRows 右旋轉函數

InvShiftRows 右旋轉函數是將狀態矩陣往右旋轉位移。第一列不做任何動作外，第二列向右旋轉位移一個位元組(byte)，第三列向右旋轉位移兩個位元組(byte)，第四列向右旋轉位移三個位元組(byte)。

## 2.3.3 InvMixColumn 反混行運算

InvMixcolumn 反混行運算是將狀態矩陣的每一行是被視為在GF($2^8$)中的多項式，乘上一固定多項式$\alpha^{-1}(x)=\{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\}$之後如果發生溢位則同餘($x^4+1$)。其中$\alpha^{-1}(x)$必需符合下列關係$\alpha^{-1}(x) \bullet \alpha(x)=1$，其中$\alpha^{-1}(x)$與2.2.3節$\alpha(x)$多項式互為乘法反元素。

可將其簡化為矩陣乘法，令$\mathbf{s}'(x) = \alpha^{-1}(x) \otimes s(x)$

$$\begin{bmatrix} S_{0,c}' \\ S_{1,c}' \\ S_{2,c}' \\ S_{3,c}' \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 0e & 09 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \qquad (2\text{-}5)$$

$S_{0,c}' = (\{0e\} \bullet S_{0,c}) \oplus (\{0b\} \bullet S_{1,c}) \oplus (\{0d\} \bullet S_{2,c}) \oplus (\{09\} \bullet S_{3,c})$

$S_{1,c}' = (\{09\} \bullet S_{0,c}) \oplus (\{0e\} \bullet S_{1,c}) \oplus (\{0b\} \bullet S_{2,c}) \oplus (\{0d\} \bullet S_{3,c})$

$S'_{2,c} = (\{0d\} \bullet S_{0,c}) \oplus (\{09\} \bullet S_{1,c}) \oplus (\{0e\} \bullet S_{2,c}) \oplus (\{0b\} \bullet S_{3,c})$

$S'_{3,c} = (\{0b\} \bullet S_{0,c}) \oplus (\{0d\} \bullet S_{1,c}) \oplus (\{09\} \bullet S_{2,c}) \oplus (\{0e\} \bullet S_{3,c})$

## 2.4 相關硬體研究工作

在近來 AES 論文研究，硬體電路研究方面大致分成二類:一種利用 FPGA[15][17][18] [19], 另外[7-12][16]是晶片設計。

在SubBytes運算函數實現有一種是非使用查表法，[14]提出將有限場GF($2^8$)基底轉換成GF($2^4$) 基底; [18]提出將有限場GF($2^8$) 基底轉換成GF($2^4$) 基底並且局部匯流排管線化(subpipeline)。

另外一種是使用查 Sbox 表格，查表法速度會比較快，但可能較占面積。在論文[17]經實驗模擬證明查表法，確實速度會比較快，但占面積。

為了達成速度快為目的，本文提出一平行架構整合查表法。但在本論文我們採用的查表，在加密部份是將每回合運算所需用到的 3 個運算函數(整合 SubBytes、ShiftRows、MixColumns 運算函數)轉換成所需表格，每個位元組運算都需要 4 個表格，最後回合運算(整合 SubBytes、ShiftRows 電路)，運算時需 sbox 表格; 解密部份是將每回合運算所需用到的 3 個運算函數(整合 InvSubBytes、InvShiftRows、InvMixColumns 運算函數)轉換成所需表格，每個位元組運算都需要 4 個表格，最後回合運算函數(整合 InvSubBytes、InvShiftRows 運算函數)，運算時需 Inverse-Sbox 表格。

由本論文第 5 章，證實平行架構整合查表法，運算電路簡易不複雜，速度快，頻寬輸出(throughput)又高。

# 第 3 章 AES 硬體電路架構實現

本論文是以設計實現 AES-128 高速硬體電路為主要目的。因此採用平行架構及同步電路提供單一時脈。

在本章，主要介紹平行架構整合電路查表法如何實現。第 3.2 節詳細描述加密電路是如何設計。第 3.3 節說明解密電路如何設計。第 3.4 節及 3.5 節分別介紹加密及解密金鑰擴展電路。

## 3.1 平行架構硬體實現方式

一般來說，硬體架構電路實現有兩種方式。一種是平行架構；另外一種為串列架構。

平行架構有如下特性:

- 硬體處理速度快。
- 硬體面積大，消耗功率大。
- 硬體成本高。

串列架構有如下特性:

- 硬體運算速度慢。
- 與平行架構比較電路面積小。
- 硬體成本低。

本論文主要目的是實現速度快的硬體電路，因此採取平行架構。

## 3.2 新架構加密電路實現

### 3.2.1 加密電路流程

新架構 AES-128 加密電路流程如圖 3.1。每回合整合運算電路整合

SubBytes、ShiftRows、MixColumns 電路，最後回合整合運算電路整合 SubBytes、ShiftRows 電路。每回合運算時經過圖左邊每回合整合運算電路，最後回合經過右邊最後回合整合運算電路。

在電路流程圖中，圖中加入正反器，最主要原因是本文採用同步電路，所以當 clock 改變時，組合邏輯電路會根據目前的狀態及輸入的訊號去計算下一個狀態。

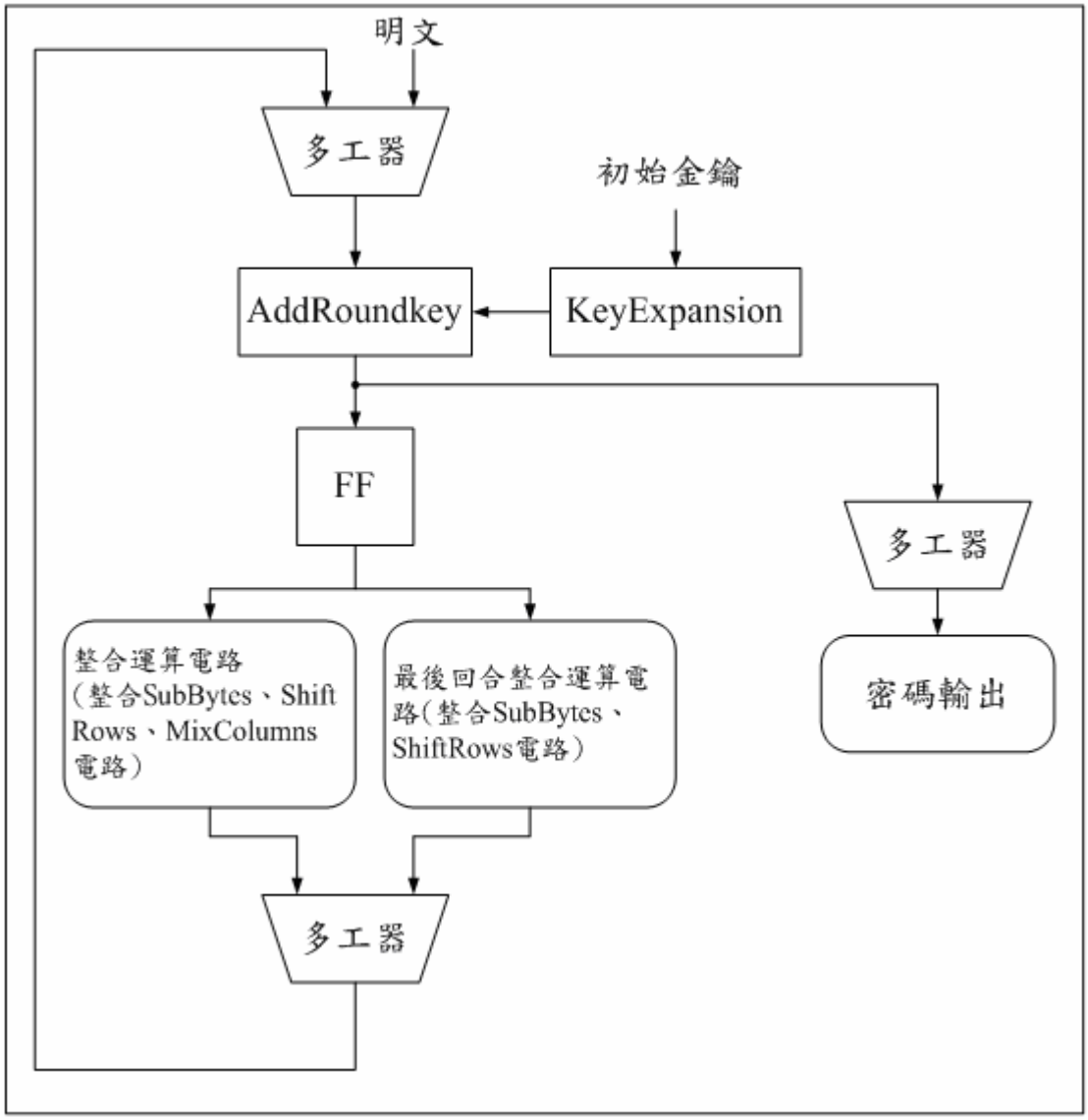


圖 3.1 新架構 AES-128 加密電路流程圖

### 3.2.2 平行架構整合查表法(加密電路)

本節將介紹本論文提出新的 AES 加密電路-平行架構整合查表法,是如何產生。如下步驟:

步驟 1 如圖 3.2,將輸入的明文或每回合運算電路輸出結果與每回合子金鑰,經 AddRoundKey 函數處理。

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s'_{0,0}$ | $s'_{0,1}$ | $s'_{0,2}$ | $s'_{0,3}$ | ⊕ | $w_{0,0}$ | $w_{0,1}$ | $w_{0,2}$ | $w_{0,3}$ | = | $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ |
| $s'_{1,0}$ | $s'_{1,1}$ | $s'_{1,2}$ | $s'_{1,3}$ | | $w_{1,0}$ | $w_{1,1}$ | $w_{1,2}$ | $w_{1,3}$ | | $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ |
| $s'_{2,0}$ | $s'_{1,1}$ | $s'_{2,2}$ | $s'_{2,3}$ | | $w_{2,0}$ | $w_{2,1}$ | $w_{2,2}$ | $w_{2,3}$ | | $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ |
| $s'_{3,0}$ | $s'_{1,1}$ | $s'_{3,2}$ | $s'_{3,3}$ | | $w_{3,0}$ | $w_{3,1}$ | $w_{3,2}$ | $w_{3,3}$ | | $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ |

圖3.2 經AddRoundKey函數處理所得結果

步驟2 如圖3.3, 經過SubBytes函數處理

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ | → | **SubBytes** 函數 | → | $P(a_{0,0})$ | $P(a_{0,1})$ | $P(a_{0,2})$ | $P(a_{0,3})$ |
| $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ | | | | $P(a_{1,0})$ | $P(a_{1,1})$ | $P(a_{1,2})$ | $P(a_{1,3})$ |
| $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ | | | | $P(a_{2,0})$ | $P(a_{2,1})$ | $P(a_{2,2})$ | $P(a_{2,3})$ |
| $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ | | | | $P(a_{3,0})$ | $P(a_{3,1})$ | $P(a_{3,2})$ | $P(a_{3,3})$ |

圖3.3 經SubBytes函數處理所得結果

步驟 3　如圖 3.4 經 Shiftrows 函數處理

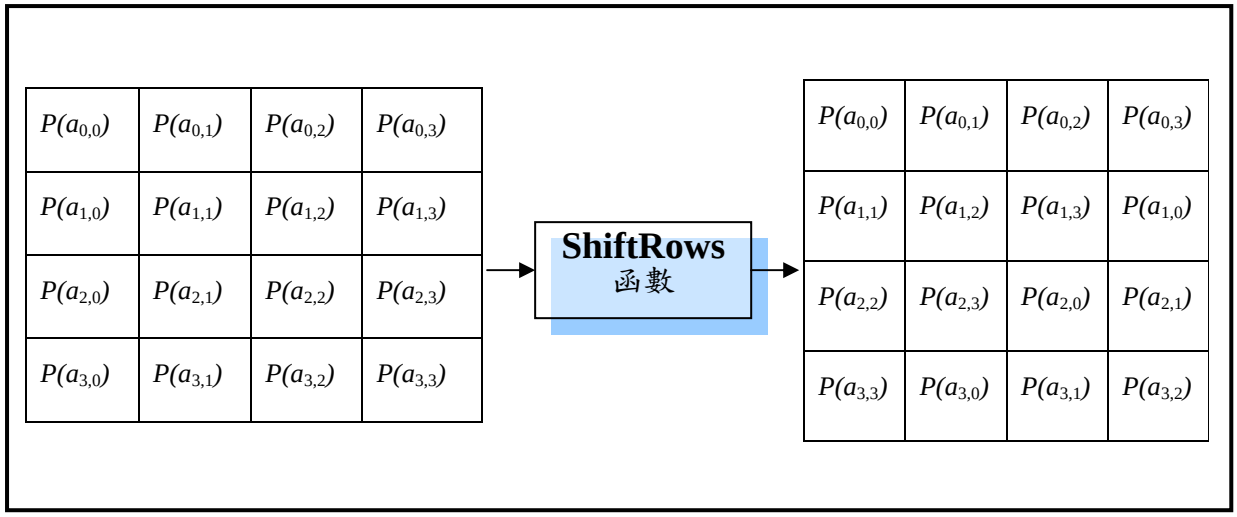


圖3.4 經ShiftRows函數處理所得結果

步驟4　經過Mixcolums函數處理(如本論文2.2.3節)，可以得到如下式子

$s'_{0,0} = 2P(a_{0,0}) \oplus 3P(a_{1,1}) \oplus P(a_{2,2}) \oplus P(a_{3,3})$;

$s'_{0,1} = 2P(a_{0,1}) \oplus 3P(a_{1,2}) \oplus P(a_{2,3}) \oplus P(a_{3,0})$;

$s'_{0,2} = 2P(a_{0,2}) \oplus 3P(a_{1,3}) \oplus P(a_{2,0}) \oplus P(a_{3,1})$;

$s'_{0,3} = 2P(a_{0,3}) \oplus 3P(a_{1,0}) \oplus P(a_{2,1}) \oplus P(a_{3,2})$;

$s'_{1,0} = P(a_{0,0}) \oplus 2P(a_{1,1}) \oplus 3P(a_{2,2}) \oplus P(a_{3,3})$;

$s'_{1,1} = P(a_{0,1}) \oplus 2P(a_{1,2}) \oplus 3P(a_{2,3}) \oplus P(a_{3,0})$;

$s'_{1,2} = P(a_{0,2}) \oplus 2P(a_{1,3}) \oplus 3P(a_{2,0}) \oplus P(a_{3,1})$;

$s'_{1,3} = P(a_{0,3}) \oplus 2P(a_{1,0}) \oplus 3P(a_{2,1}) \oplus P(a_{3,2})$;

$s'_{2,0} = P(a_{0,0}) \oplus P(a_{1,1}) \oplus 2P(a_{2,2}) \oplus 3P(a_{3,3})$;

$s'_{2,1} = P(a_{0,1}) \oplus P(a_{1,2}) \oplus 2P(a_{2,3}) \oplus 3P(a_{3,0})$;

$s'_{2,2} = P(a_{0,2}) \oplus P(a_{1,3}) \oplus 2P(a_{2,0}) \oplus 3P(a_{3,1})$;

$s'_{2,3} = P(a_{0,3}) \oplus P(a_{1,0}) \oplus 2P(a_{2,1}) \oplus 3P(a_{3,2})$;

$s'_{3,0} = 3P(a_{0,0}) \oplus P(a_{1,1}) \oplus P(a_{2,2}) \oplus 2P(a_{3,3})$;

$s'_{3,1} = 3P(a_{0,1}) \oplus P(a_{1,2}) \oplus P(a_{2,3}) \oplus 2P(a_{3,0})$;

$s'_{3,2} = 3P(a_{0,2}) \oplus P(a_{1,3}) \oplus P(a_{2,0}) \oplus 2P(a_{3,1})$;

$s'_{3,3}$ =3P($a_{0,3}$) ⊕P($a_{1,0}$) ⊕P($a_{2,1}$)⊕2P($a_{3,2}$);

由上列式子，可以歸納得到: 每回合每個位元組(byte)運算，可以用如上一個運算數學式表示。因此可以整合SubBytes、ShiftRows、MixColumns運算為一個整合查表電路。在硬體實現上每個位元組(byte)運算需要 4 個表格及 3 個互斥或閘運算。4 個表格分別為*P*表格、2*P* 表格及 3*P*表格。其中*P*表格即為s-box表格；2*P* 表格是將S-box表內所有數值乘以{02}, 若有溢位(overflow), 則同餘 283;3*P*表格是將S-box表內所有數值乘以{03}, 若有溢位(overflow), 則同餘 283。2*P* 表格如表 3.1;3*P*表格如表 3.2。

其中 2*P*($a_{i,j}$)為查$a_{i,j}$在 2*P* 表格的數值, 3*P*($a_{i,j}$)為查$a_{i,j}$在 3 *P*表格的數值。若2*P*($a_{3,1}$)=$(3c)_{16}$， 最高 4 個位元為 3, 所以查 2*P*表第 3 列; 最低 4 個位元為c, 所以查 2*P*表第c行，對應到$(cd)_{16}$。

圖 3.5 及圖 3.6 為加密每回合平行架構電路圖, Rin表示輸入, Rout表示輸出。由電路圖可知每個位元組都需經*P*、2*P* 及 3*P* 整合電路表格。

表 3.1 2P 轉換對照表表格

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | c6 | f8 | ee | f6 | ff | d6 | de | 91 | 60 | 02 | ce | 56 | e7 | b5 | 4d | ec |
| 1 | 8f | 1f | 89 | fa | ef | b2 | 8e | fb | 41 | b3 | 5f | 45 | 23 | 53 | e4 | 9b |
| 2 | 75 | el | 3d | 4c | 6c | 7e | f5 | 83 | 68 | 51 | d1 | f9 | e2 | ab | 62 | 2a |
| 3 | 08 | 95 | 46 | 9d | 30 | 37 | 0a | 2f | 0e | 24 | 1b | df | cd | 4e | 7f | ea |
| 4 | 12 | 1d | 58 | 34 | 36 | dc | b4 | 5b | a4 | 76 | b7 | 7d | 52 | dd | 5e | 13 |
| 5 | a6 | b9 | 00 | c1 | 40 | e3 | 79 | b6 | d4 | 8d | 67 | 72 | 94 | 98 | b0 | 85 |
| 6 | bb | c5 | 4f | ed | 86 | 9a | 66 | 11 | 8a | e9 | 04 | fe | a0 | 78 | 25 | 4b |
| 7 | a2 | 5d | 80 | 05 | 3f | 21 | 70 | f1 | 63 | 77 | af | 42 | 20 | e5 | fd | bf |
| 8 | 81 | 18 | 26 | c3 | be | 35 | 88 | 2e | 93 | 55 | fc | 7a | c8 | ba | 32 | e6 |
| 9 | c0 | 19 | 9e | a3 | 44 | 54 | 3b | 0b | 8c | c7 | 6b | 28 | a7 | bc | 16 | ad |
| a | db | 64 | 74 | 14 | 92 | 0c | 48 | b8 | 9f | bd | 43 | c4 | 39 | 31 | d3 | f2 |
| b | d5 | 8b | 6e | da | 01 | b1 | 9c | 49 | d8 | ac | f3 | cf | ca | f4 | 47 | 10 |
| c | 6f | f0 | 4a | 5c | 38 | 57 | 73 | 97 | cb | a1 | e8 | 3e | 96 | 61 | 0d | 0f |
| d | e0 | 7c | 71 | cc | 90 | 06 | f7 | 1c | c2 | 6a | ae | 69 | 17 | 99 | 3a | 27 |
| e | d9 | eb | 2b | 22 | d2 | a9 | 07 | 33 | 2d | 3c | 15 | c9 | 87 | aa | 50 | a5 |
| f | 03 | 59 | 09 | 1a | 65 | d7 | 84 | d0 | 82 | 29 | 5a | 1e | 7b | a8 | 6d | 2c |

表 3.2 3P 轉換對照表表格

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | a5 | 84 | 99 | 8d | 0d | bd | b1 | 54 | 50 | 03 | a9 | 7d | 19 | 62 | e6 | 9a |
| 1 | 45 | 9d | 40 | 87 | 15 | eb | c9 | 0b | ec | 67 | fd | ea | bf | f7 | 96 | 5b |
| 2 | c2 | 1c | ae | 6a | 5a | 41 | 02 | 4f | 5c | f4 | 34 | 08 | 93 | 73 | 53 | 3f |
| 3 | 0c | 52 | 65 | 5e | 28 | a1 | 0f | b5 | 09 | 36 | 9b | 3d | 26 | 69 | cd | 9f |
| 4 | 1b | 9e | 74 | 2e | 2d | b2 | ee | fb | f6 | 4d | 61 | ce | 7b | 3e | 71 | 97 |
| 5 | f5 | 68 | 00 | 2c | 60 | 1f | c8 | ed | be | 46 | d9 | 4b | de | d4 | e8 | 4a |
| 6 | 6b | 2a | e5 | 16 | c5 | d7 | 55 | 94 | cf | 10 | 06 | 81 | f0 | 44 | ba | e3 |
| 7 | f3 | fe | c0 | 8a | ad | bc | 48 | 04 | df | c1 | 75 | 63 | 30 | 1a | 0e | 6d |
| 8 | 4c | 14 | 35 | 2f | el | a2 | cc | 39 | 57 | f2 | 82 | 47 | ac | e7 | 2b | 95 |
| 9 | a0 | 98 | d1 | 7f | 66 | 7e | ab | 83 | ca | 29 | d3 | 3c | 79 | e2 | 1d | 76 |
| a | 3b | 56 | 4e | 1e | db | 0a | 6c | e4 | 5d | 6e | ef | a6 | a8 | a4 | 37 | 8b |
| b | 32 | 43 | 59 | b7 | 8c | 64 | d2 | e0 | b4 | fa | 07 | 25 | af | 8e | e9 | 18 |
| c | d5 | 88 | 6f | 72 | 24 | f1 | c7 | 51 | 23 | 7c | 9c | 21 | dd | dc | 86 | 85 |
| d | 90 | 42 | c4 | aa | d8 | 05 | 01 | 12 | a3 | 5f | f9 | d0 | 91 | 58 | 27 | b9 |
| e | 38 | 13 | b3 | 33 | bb | 70 | 89 | a7 | b6 | 22 | 92 | 20 | 49 | ff | 78 | 7a |
| f | 8f | f8 | 80 | 17 | da | 31 | c6 | b8 | c3 | b0 | 77 | 11 | cb | fc | d6 | 3a |

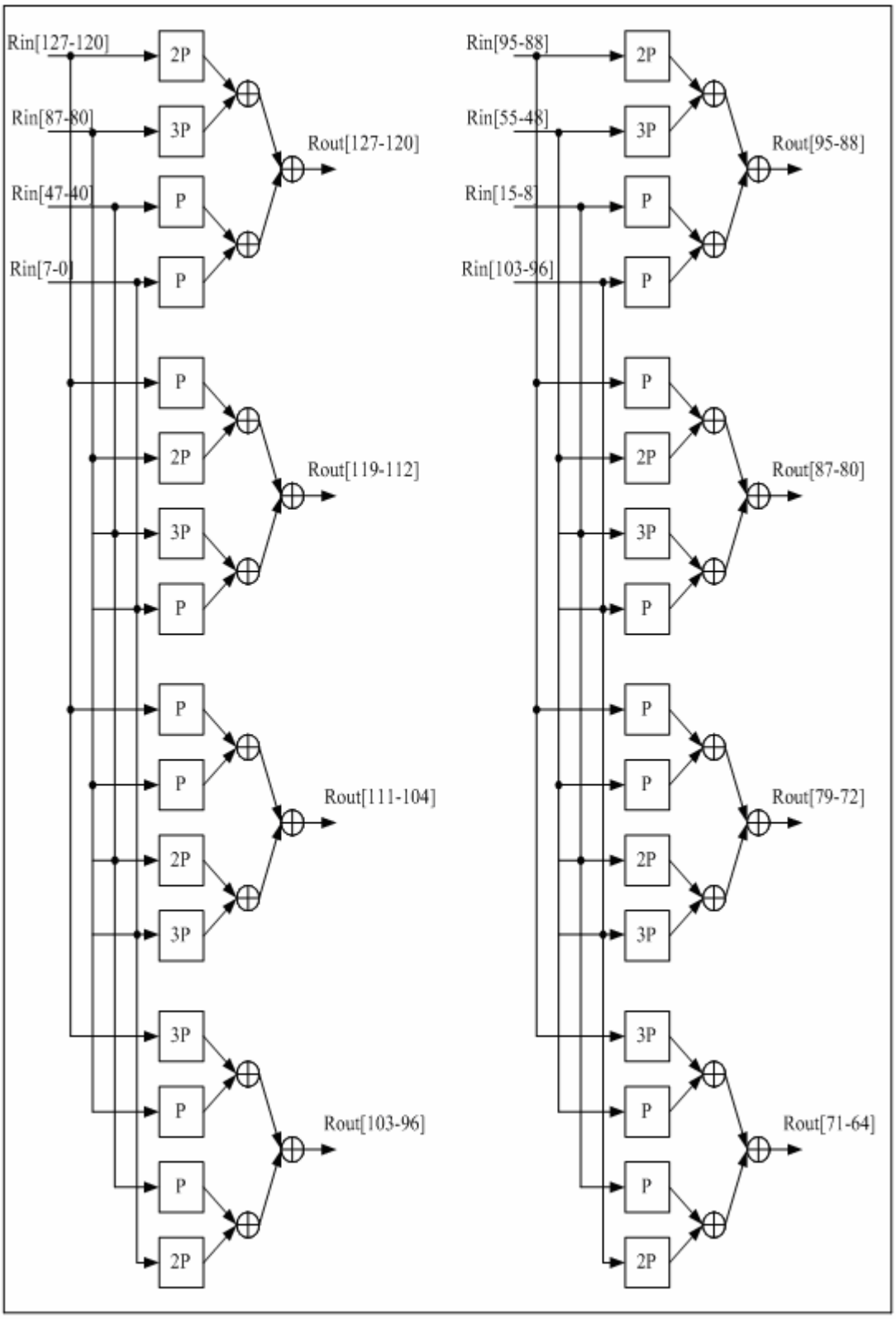


圖 3.5 加密每回合平行架構電路圖(一)

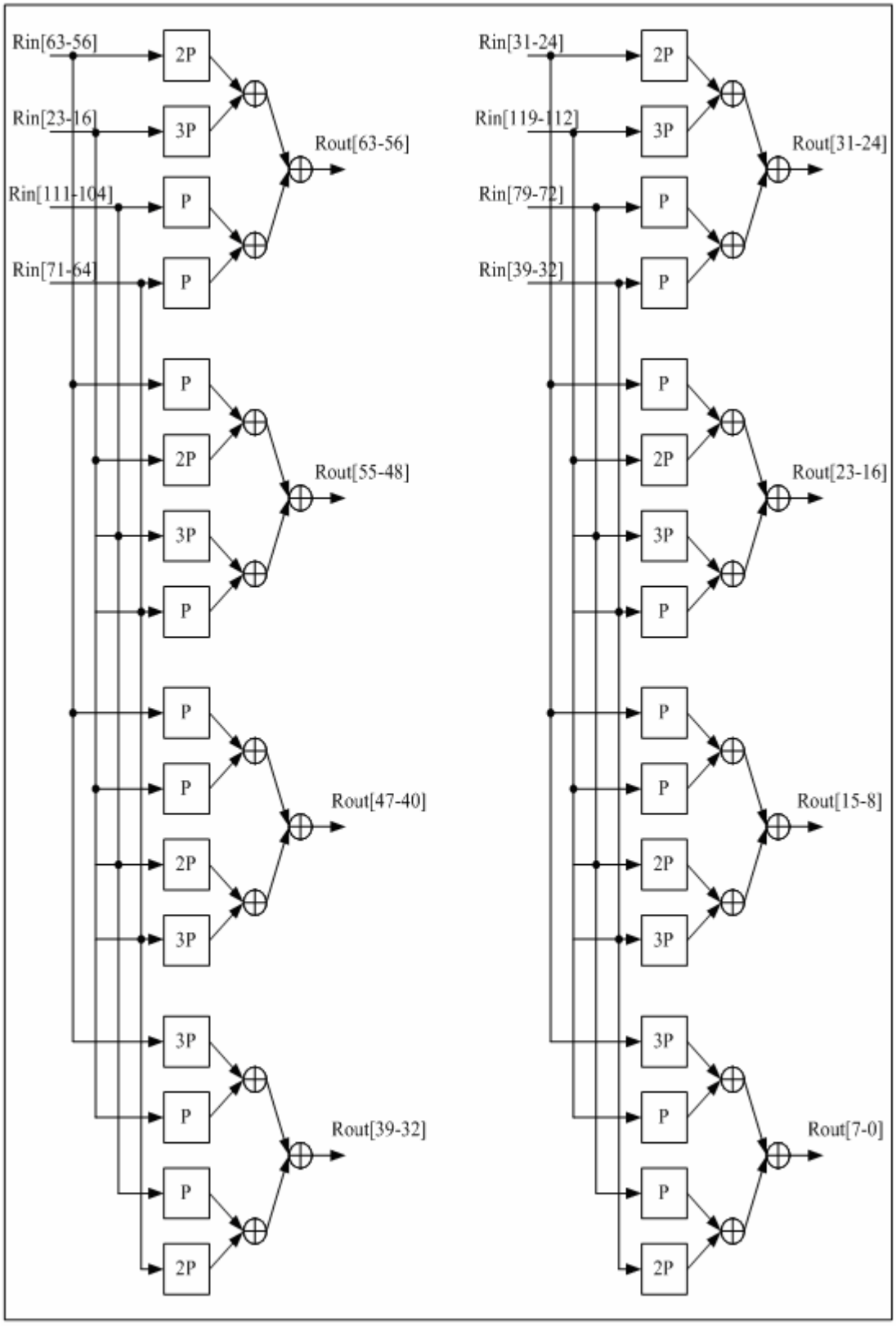


圖 3.6 加密每回合平行架構電路圖(二)

步驟 5 加上回合子金鑰, 可以得到下列式子

$s'_{0,0}$ =2P($a_{0,0}$) ⊕3P($a_{1,1}$) ⊕P($a_{2,2}$)⊕P($a_{3,3}$)⊕$W_{0,0}$

$s'_{0,1}$ =2P($a_{0,1}$) ⊕3P($a_{1,2}$) ⊕P($a_{2,3}$)⊕P($a_{3,0}$)⊕$W_{0,1}$

$s'_{0,2}$ =2P($a_{0,2}$) ⊕3P($a_{1,3}$) ⊕P($a_{2,0}$)⊕P( $a_{3,1}$)⊕$W_{0,2}$

$s'_{0,3}$ =2P($a_{0,3}$) ⊕3P($a_{1,0}$) ⊕P($a_{2,1}$) ⊕P( $a_{3,2}$)⊕$W_{0,3}$

$s'_{1,0}$ = P($a_{0,0}$)⊕P($a_{1,1}$)⊕2P($a_{2,2}$)⊕3P($a_{3,3}$)⊕$W_{1,0}$

$s'_{1,1}$ = P($a_{0,1}$)⊕2P($a_{1,2}$)⊕3P($a_{2,3}$)⊕P($a_{3,0}$)⊕$W_{1,1}$

$s'_{1,2}$ = P($a_{0,2}$)⊕2P($a_{1,3}$)⊕3P($a_{2,0}$)⊕P($a_{3,1}$)⊕$W_{1,2}$

$s'_{1,3}$ = P($a_{0,3}$)⊕2P($a_{1,0}$)⊕3P($a_{2,1}$)⊕P($a_{3,2}$)⊕$W_{1,3}$

$s'_{2,0}$ =P($a_{0,0}$)⊕P($a_{1,1}$)⊕2P($a_{2,2}$)⊕3P($a_{3,3}$)⊕$W_{2,0}$

$s'_{2,1}$ = P($a_{0,1}$)⊕P($a_{1,2}$)⊕2P($a_{2,3}$)⊕3P($a_{3,0}$)⊕$W_{2,1}$

$s'_{2,2}$ = P($a_{0,2}$)⊕P($a_{1,3}$)⊕2P($a_{2,0}$)⊕3P($a_{3,1}$)⊕$W_{2,2}$

$s'_{2,3}$ = P($a_{0,3}$)⊕P($a_{1,0}$)⊕2P($a_{2,1}$)⊕3P($a_{3,2}$)⊕$W_{2,3}$

$s'_{3,0}$ =3P($a_{0,0}$) ⊕P($a_{1,1}$) ⊕P($a_{2,2}$)⊕2P($a_{3,3}$)⊕$W_{3,0}$

$s'_{3,1}$ =3P($a_{0,1}$) ⊕P($a_{1,2}$) ⊕P($a_{2,3}$)⊕2P($a_{3,0}$)⊕$W_{3,1}$

$s'_{3,2}$ =3P($a_{0,2}$) ⊕P($a_{1,3}$) ⊕P($a_{2,0}$)⊕2P($a_{3,1}$)⊕$W_{3,2}$

$s'_{3,3}$ =3P($a_{0,3}$) ⊕P($a_{1,0}$) ⊕P($a_{2,1}$)⊕2P($a_{3,2}$)⊕$W_{3,3}$

步驟 6 重覆步驟 2 至步驟 5 總共 9 次, 將所得結果傳送至最後回合運算電路；最後回合運算電路, 為整合 SubBytes、ShiftRows 兩個函數為一個數學運算式。如下列式子

$s'_{0,0}$= P($a_{0,0}$); $s'_{0,1}$ = P($a_{0,1}$); $s'_{0,2}$ = P($a_{0,2}$); $s'_{0,3}$ = P($a_{0,3}$) ;

$s'_{1,0}$ = P($a_{1,1}$); $s'_{1,1}$= P($a_{1,2}$); $s'_{1,2}$= P($a_{1,3}$); $s'_{1,3}$ = P($a_{1,0}$);

$s'_{2,0}$ = P($a_{2,2}$); $s'_{2,1}$ = P($a_{2,3}$); $s'_{2,2}$ = P($a_{2,0}$); $s'_{2,3}$ = P($a_{2,1}$);

$s'_{3,0}$ = P($a_{3,3}$) ; $s'_{3,1}$ = P($a_{3,0}$); $s'_{3,2}$ = P($a_{3,1}$); $s'_{3,3}$ = P($a_{3,2}$);

圖 3.7 為加密最後回合整合運算電路架構圖, Fin 表示輸入, Fout 表示輸出。

步驟 7 與最後回合子金鑰相加所得結果

$s'_{0,0} = P(a_{0,0}) \oplus W_{0,0}$;

$s'_{0,1} = P(a_{0,1}) \oplus W_{0,1}$;

$s'_{0,2} = P(a_{0,2}) \oplus W_{0,2}$;

$s'_{0,3} = P(a_{0,3}) \oplus W_{0,3}$;

$s'_{1,0} = P(a_{1,1}) \oplus W_{1,0}$;

$s'_{1,1} = P(a_{1,2}) \oplus W_{1,1}$;

$s'_{1,2} = P(a_{1,3}) \oplus W_{1,2}$;

$s'_{1,3} = P(a_{1,0}) \oplus W_{1,3}$

$s'_{2,0} = P(a_{2,2}) \oplus W_{2,0}$;

$s'_{2,1} = P(a_{2,3}) \oplus W_{2,1}$;

$s'_{2,2} = P(a_{2,0}) \oplus W_{2,2}$;

$s'_{2,3} = P(a_{2,1}) \oplus W_{2,3}$;

$s'_{3,0} = P(a_{3,3}) \oplus W_{3,0}$;

$s'_{3,1} = P(a_{3,0}) \oplus W_{3,1}$;

$s'_{3,2} = P(a_{3,1}) \oplus W_{3,2}$;

$s'_{3,3} = P(a_{3,2}) \oplus W_{3,3}$;

在硬體實現上每個位元組運算需要 1 個表格及 1 個互斥或閘運算。1 個表格為*P*表格。其中*P*表格即為s-box表格, $P(a_{i,j})$為查$a_{i,j}$在s-box表格的數值。

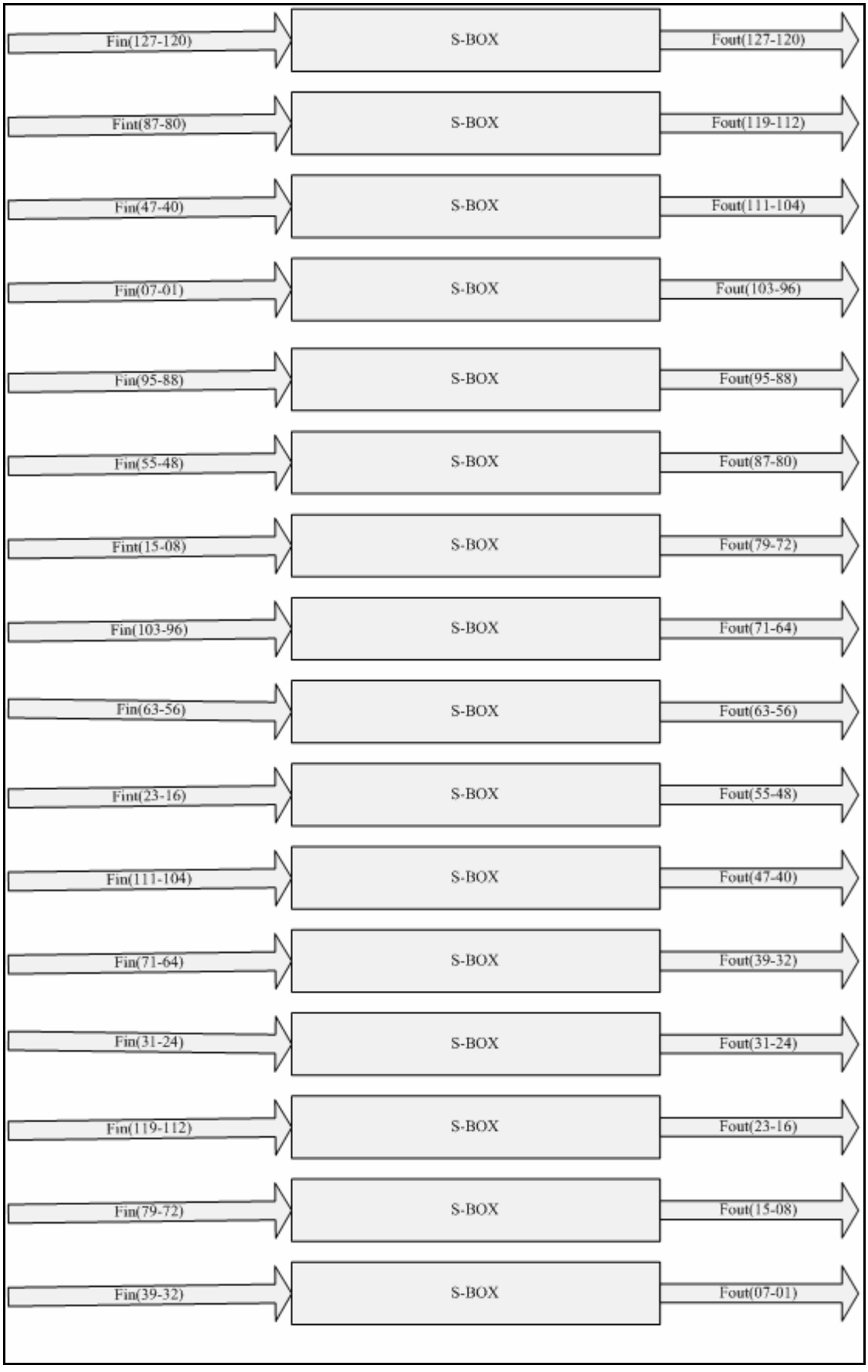


圖 3.7 加密最後回合平行架構電路圖

## 3.3 新架構解密電路實現

### 3.3.1 解密電路流程

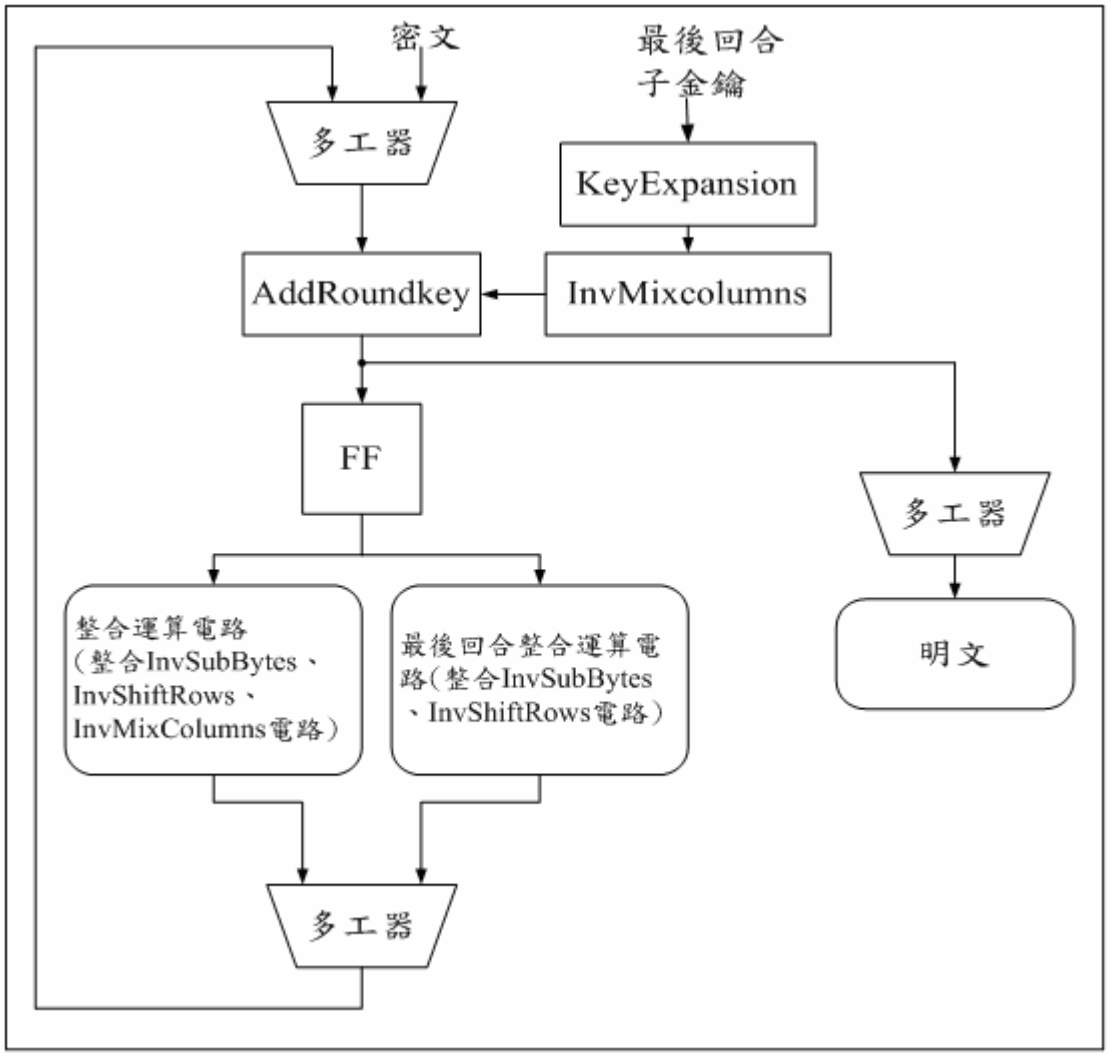


圖 3.8 新架構 AES-128 解密電路流程圖

新架構 AES-128 解密電路流程如圖 3.8。每回合整合運算電路整合 InvSubBytes、InvShiftRows、InvMixColumns 電路，最後回合整合運算電路整合 InvSubBytes、InvShiftRows 電路。每回合電路運算時經過圖左邊每回合整合運算電路，最後回合經過右邊最後回合整合運算電路。

本文因採用 Equivalent inverse decrypt[3]，每回合子金鑰需增加 InvMixcolums 函數處理，解密電路會比加密電路面積大。

### 3.3.2 平行架構整合查表法(解密電路)

本節將介紹本論文提出新的 AES 解密電路-平行架構整合查表法,是如何產生。如下步驟:

步驟 1 如圖 3.9，將輸入的密文或每回合運算電路輸出結果與每回合子金鑰經 AddRoundKey 函數處理。

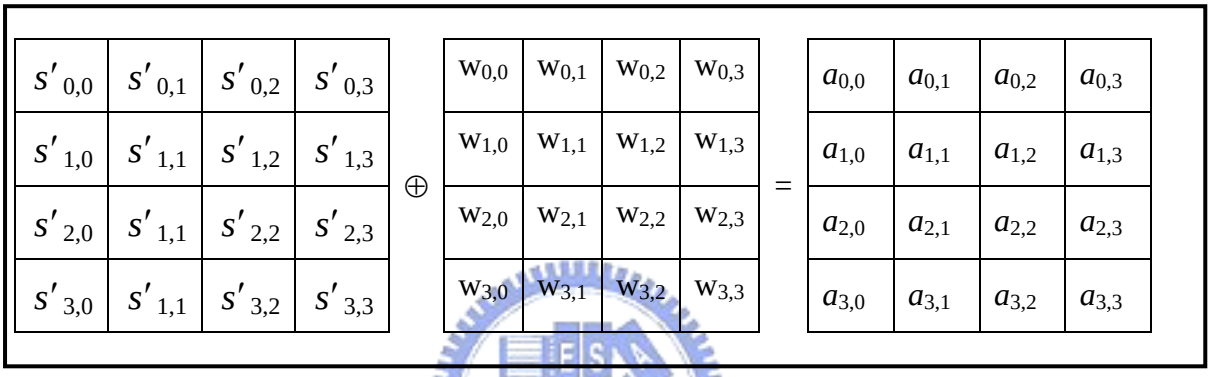


圖3.9 經AddRoundKey函數處理所得結果

步驟 2 如圖 3.10，經過 InvSubBytes 函數處理。

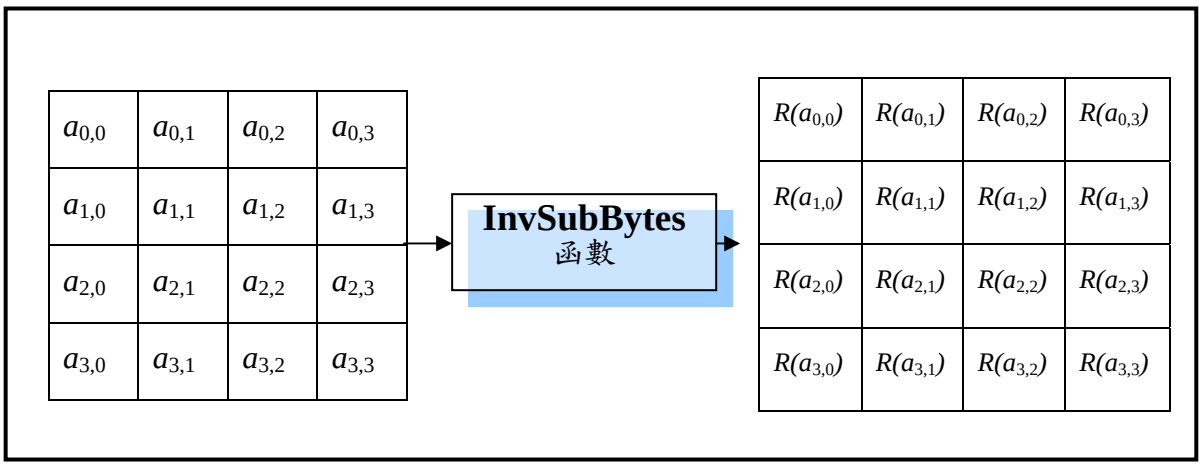


圖3.10 經過InvSubBytes函數處理所得結果

步驟 3 如圖 3.11，經過 InvShiftrows 函數處理

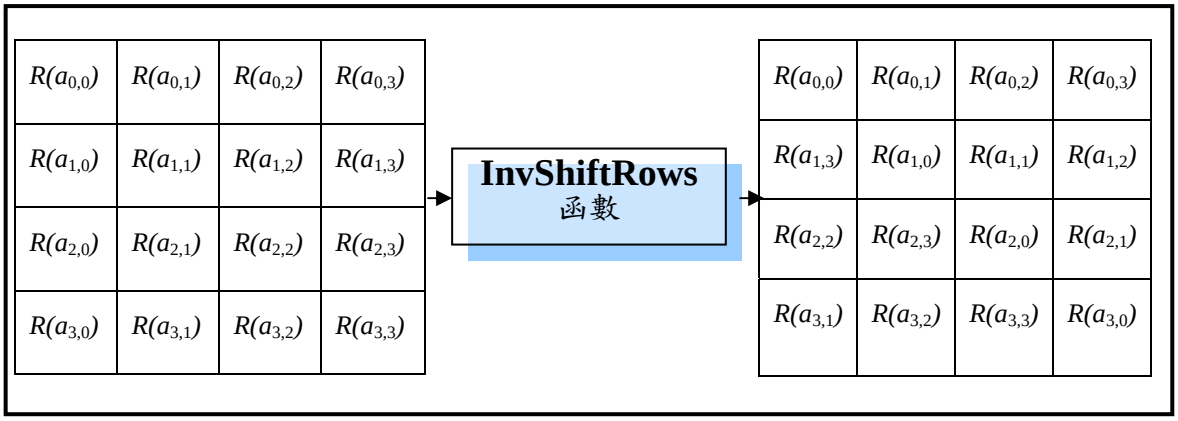


圖 3.11 經過InvShiftRows處理所得結果

步驟4 經過InvMixcolums函數處理，可以得到如下式子

$s'_{0,0}$ =eR($a_{0,0}$) ⊕bR($a_{1,3}$) ⊕dR($a_{2,2}$) ⊕9R($a_{3,1}$);

$s'_{0,1}$ =eR($a_{0,1}$) ⊕bR($Ra_{1,0}$) ⊕dR($a_{2,3}$) ⊕9R($a_{3,2}$);

$s'_{0,2}$ =eR($a_{0,2}$) ⊕bR($a_{1,1}$) ⊕dR($a_{2,0}$) ⊕9R($a_{3,3}$);

$s'_{0,3}$ =eR($a_{0,3}$) ⊕bR($a_{1,2}$) ⊕dR($a_{2,1}$) ⊕9R($a_{3,0}$);

$s'_{1,0}$ =9R($a_{0,0}$) ⊕eR($a_{1,3}$) ⊕dR($a_{2,2}$) ⊕bR($Ra_{3,1}$);

$s'_{1,1}$ =9R($a_{0,1}$) ⊕eR($a_{1,0}$) ⊕dR($a_{2,3}$) ⊕bR($a_{3,2}$);

$s'_{1,2}$ =9R($a_{0,2}$) ⊕eR($a_{1,1}$) ⊕dR($a_{2,0}$) ⊕bR($a_{3,3}$);

$s'_{1,3}$ =9R($a_{0,3}$) ⊕eR($a_{1,2}$) ⊕dR($a_{2,1}$) ⊕bR($a_{3,0}$);

$s'_{2,0}$ =dR($a_{0,0}$) ⊕9R($a_{1,3}$) ⊕eR($a_{2,2}$) ⊕bR($a_{3,1}$);

$s'_{2,1}$ =dR($a_{0,1}$) ⊕9R($a_{1,0}$) ⊕eR($a_{2,3}$) ⊕bR($a_{3,2}$);

$s'_{2,2}$ =dR($a_{0,2}$) ⊕9R($a_{1,1}$) ⊕eR($a_{2,0}$) ⊕bR($a_{3,3}$);

$s'_{2,3}$ =dR($a_{0,3}$) ⊕9R($a_{1,2}$) ⊕eR($a_{2,1}$) ⊕bR($a_{3,0}$);

$s'_{3,0}$ =bR($a_{0,0}$) ⊕dR($a_{1,3}$) ⊕9R($a_{2,2}$) ⊕eR($a_{3,1}$);

$s'_{3,1}$ =bR($a_{0,1}$) ⊕dR($a_{1,0}$) ⊕9R($a_{2,3}$) ⊕eR($a_{3,2}$);

$s'_{3,2}$ =bR($a_{0,2}$) ⊕dR($a_{1,1}$) ⊕9R($a_{2,0}$) ⊕eR($a_{3,3}$);

$s'_{3,3}$ =bR($a_{0,3}$) ⊕dR($a_{1,2}$) ⊕9R($a_{2,1}$) ⊕eR($a_{3,0}$);

由上列式子，可以歸納得到: 每回合每個位元組(byte)運算，可以用如上一個運算數學式表示。因此可以整合InvSubBytes、InvShiftRows、InvMixColumns為一個新的整合電路。在硬體實現上每個位元組運算需要 4 個表格及 3 個及斥或閘運算。4 個表格分別為 9*R*表格、b*R*表格、d*R*表格及e*R*。其中 9*R*表格是將Inverse S-box表內所有數值乘以{09}, 若有溢位(overflow), 則同餘 283;b*R*表格是將Inverse S-box表內所有數值乘以{0b}, 若有溢位(overflow), 則同餘 283;d*R*表格是將Inverse S-box表內所有數值乘以{0d}, 若有溢位(overflow), 則同餘 283; e*R*表格是將Inverse S-box表內所有數值乘以{0e}, 若有溢位(overflow), 則同餘 283。其中9R($a_{i,j}$)為查$a_{i,j}$在 9R表格的數值, bR($a_{i,j}$)為查$a_{i,j}$在bR表格的數值, dR($a_{i,j}$)為查$a_{i,j}$在dR 表格的數值, eR($a_{i,j}$)為查$a_{i,j}$在eR表格的數值。9R表格如表 3.3;bR表格如表 3.4;dR格如表 3.5;eR 表格如表 3.6。

圖 3.12 及圖 3.13 為加密每回合整合運算電路架構圖, IRin 表示輸入, IRout 表示輸出。由電路圖可知每個位元組都需經 9R、bR、dR 及 eR 整合電路表格。

表 3.3　9R 轉換對照表表格

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | f4 | 41 | 17 | 27 | ab | 9d | fa | e3 | 30 | 76 | cc | 02 | e5 | 2a | 35 | 62 |
| 1 | b1 | ba | ea | fe | 2f | 4c | 46 | d3 | 8f | 92 | 6d | 52 | be | 74 | e0 | c9 |
| 2 | c2 | 8e | 58 | b9 | e1 | 88 | 20 | ce | df | 1a | 51 | 53 | 64 | 6b | 81 | 08 |
| 3 | 48 | 45 | de | 7b | 73 | 4b | 1f | 55 | eb | b5 | c5 | 37 | 28 | bf | 03 | 16 |
| 4 | cf | 79 | 07 | 69 | da | 05 | 34 | a6 | 2e | f3 | 8a | f6 | 83 | 60 | 71 | 6e |
| 5 | 21 | dd | 3e | e6 | 54 | c4 | 06 | 50 | 98 | bd | 40 | d9 | e8 | 89 | 19 | c8 |
| 6 | 7c | 42 | 84 | 00 | 80 | 2b | 11 | 5a | 0e | 85 | ae | 2d | 0f | 5c | 5b | 36 |
| 7 | 0a | 57 | ee | 9b | c0 | dc | 77 | 12 | 93 | a0 | 22 | 1b | 09 | 8b | b6 | 1e |
| 8 | f1 | 75 | 99 | 7f | 01 | 72 | 66 | fb | 43 | 23 | ed | e4 | 31 | 63 | 97 | c6 |
| 9 | 4a | bb | f9 | 29 | 9e | b2 | 86 | c1 | b3 | 70 | 94 | e9 | fc | f0 | 7d | 33 |
| a | 49 | 38 | ca | d4 | f5 | 7a | b7 | ad | 3a | 78 | 5f | 7e | 8d | d8 | 39 | c3 |
| b | 5d | d0 | d5 | 25 | ac | 18 | 9c | 3b | 26 | 59 | 9a | 4f | 95 | ff | bc | 15 |
| c | e7 | 6f | 9f | b0 | a4 | 3f | a5 | a2 | 4e | 82 | 90 | a7 | 04 | ec | Cd | 91 |
| d | 4d | ef | aa | 96 | d1 | 6a | 2c | 65 | 5e | 8c | 87 | 0b | 67 | db | 10 | d6 |
| e | d7 | a1 | f8 | 13 | a9 | 61 | 1c | 47 | d2 | f2 | 14 | c7 | fb | fd | 3d | 44 |
| f | af | 68 | 24 | a3 | 1d | e2 | 3c | 0d | a8 | 0c | b4 | 56 | cb | 32 | 6c | b8 |

表 3.4　bR 轉換對照表表格

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 50 | 53 | c3 | 96 | cb | f1 | ab | 93 | 55 | f6 | 91 | 25 | fc | d7 | 80 | 8f |
| 1 | 49 | 67 | 98 | el | 02 | 12 | a3 | c6 | e7 | 95 | eb | da | 2d | d3 | 29 | 44 |
| 2 | 6a | 78 | 6b | dd | b6 | 17 | 66 | b4 | 18 | 82 | 60 | 45 | e0 | 84 | 1c | 94 |
| 3 | 58 | 19 | 87 | b7 | 23 | e2 | 57 | 2a | 07 | 03 | 9a | a5 | f2 | b2 | ba | 5c |
| 4 | 2b | 92 | f0 | a1 | cd | d5 | 1f | 8a | 9d | a0 | 32 | 75 | 39 | aa | 06 | 51 |
| 5 | f9 | 3d | ae | 46 | b5 | 05 | 6f | ff | 24 | 97 | cc | 77 | bd | 88 | 38 | db |
| 6 | 47 | e9 | c9 | 00 | 83 | 48 | ac | 4e | fb | 56 | 1e | 27 | 64 | 21 | d1 | 3a |
| 7 | b1 | 0f | d2 | 9e | 4f | a2 | 69 | 16 | 0a | e5 | 43 | 1d | 0b | ad | b9 | c8 |
| 8 | 85 | 4c | bb | fd | 9f | bc | c5 | 34 | 76 | dc | 68 | 63 | ca | 10 | 40 | 20 |
| 9 | 7d | f8 | 11 | 6d | 4b | f3 | ec | d0 | 6c | 99 | fa | 22 | c4 | 1a | d8 | ef |
| a | c7 | c1 | fe | 36 | cf | 28 | 26 | a4 | e4 | 0d | 9b | 62 | c2 | e8 | 5e | f5 |
| b | be | 7c | a9 | b3 | 3b | a7 | 6e | 7b | 09 | f4 | 01 | a8 | 65 | 7e | 08 | e6 |
| c | d9 | ce | d4 | d6 | af | 31 | 30 | c0 | 37 | a6 | b0 | 15 | 4a | f7 | 0e | 2f |
| d | 8d | 4d | 54 | df | e3 | 1b | b8 | 7f | 04 | 5d | 73 | 2e | 5a | 52 | 33 | 13 |
| e | 8c | 7a | 8e | 89 | ee | 35 | ed | 3c | 59 | 3f | 79 | bf | ea | 5b | 14 | 86 |
| f | 81 | 3e | 2c | 5f | 72 | 0c | 8b | 41 | 71 | de | 9c | 90 | 61 | 70 | 74 | 42 |

表 3.5　dR 轉換對照表表格

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | a7 | 65 | a4 | 5e | 6b | 45 | 58 | 03 | fa | 6d | 76 | 4c | d7 | cb | 44 | a3 |
| 1 | 5a | 1b | 0e | c0 | 75 | f0 | 97 | f9 | 5f | 9c | 7a | 59 | 83 | 21 | 69 | c8 |
| 2 | 89 | 79 | 3e | 71 | 4f | ad | ac | 3a | 4a | 31 | 33 | 7f | 77 | ae | a0 | 2b |
| 3 | 68 | fd | 6c | f8 | d3 | 02 | 8f | ab | 28 | c2 | 7b | 08 | 87 | a5 | 6a | 82 |
| 4 | 1c | b4 | f2 | e2 | f4 | be | 62 | fe | 53 | 55 | el | eb | ec | ef | 9f | 10 |
| 5 | 8a | 06 | 05 | bd | 8d | 5d | d4 | 15 | fb | e9 | 43 | 9e | 42 | 8b | 5b | ee |
| 6 | 0a | 0f | 1e | 00 | 86 | ed | 70 | 72 | ff | 38 | d5 | 39 | d9 | a6 | 54 | 2e |
| 7 | 67 | e7 | 96 | 91 | c5 | 20 | 4b | 1a | ba | 2a | e0 | 17 | 0d | c7 | a8 | a9 |
| 8 | 19 | 07 | dd | 60 | 26 | f5 | 3b | 7e | 29 | c6 | fc | f1 | dc | 85 | 22 | 11 |
| 9 | 24 | 3d | 32 | a1 | 2f | 30 | 52 | e3 | 16 | b9 | 48 | 64 | 8c | 3f | 2c | 90 |
| a | 4e | d1 | a2 | 0b | 81 | de | 8e | bf | 9d | 92 | cc | 46 | 13 | b8 | f7 | af |
| b | 80 | 93 | 2d | 12 | 99 | 7d | 63 | bb | 78 | 18 | b7 | 9a | 6e | e6 | cf | e8 |
| c | 9b | 36 | 09 | 7c | b2 | 23 | 94 | 66 | bc | ca | d0 | d8 | 98 | da | 50 | f6 |
| d | d6 | b0 | 4d | 04 | b5 | 88 | 1f | 51 | ea | 35 | 74 | 41 | 1d | d2 | 56 | 47 |
| e | 61 | 0c | 14 | 3c | 27 | c9 | e5 | b1 | df | 73 | ce | 37 | cd | aa | 6f | db |
| f | f3 | c4 | 34 | 40 | c3 | 25 | 49 | 95 | 01 | b3 | e4 | c1 | 84 | b6 | 5c | 57 |

表 3.6　eR 轉換對照表表格

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 51 | 7e | 1a | 3a | 3b | 1f | ac | 4b | 20 | ad | 88 | f5 | 4f | c5 | 26 | b5 |
| 1 | de | 25 | 45 | 5d | c3 | 81 | 8d | 6b | 03 | 15 | bf | 95 | d4 | 58 | 49 | 8e |
| 2 | 75 | f4 | 99 | 27 | be | f0 | c9 | 7d | 63 | e5 | 97 | 62 | b1 | bb | fe | f9 |
| 3 | 70 | 8f | 94 | 52 | ab | 72 | e3 | 66 | b2 | 2f | 86 | d3 | 30 | 23 | 02 | ed |
| 4 | 8a | a7 | f3 | 4e | 65 | 06 | d1 | c4 | 34 | a2 | 05 | a4 | 0b | 40 | 5e | bd |
| 5 | 3e | 96 | dd | 4d | 91 | 71 | 04 | 60 | 19 | d6 | 89 | 67 | b0 | 07 | e7 | 79 |
| 6 | a1 | 7c | f8 | 00 | 09 | 32 | 1e | 6c | fd | 0f | 3d | 36 | 0a | 68 | 9b | 24 |
| 7 | 0c | 93 | b4 | 1b | 80 | 61 | 5a | 1c | e2 | c0 | 3c | 12 | 0e | f2 | 2d | 14 |
| 8 | 57 | af | ee | a3 | f7 | 5c | 44 | 5b | 8b | cb | b6 | b8 | d7 | 42 | 31 | 84 |
| 9 | 85 | d2 | ae | c7 | 1d | dc | 0d | 77 | 2b | a9 | 11 | 47 | a8 | a0 | 56 | 22 |
| a | 87 | d9 | 8c | 98 | a6 | a5 | da | 3f | 2c | 50 | 6a | 54 | f6 | 90 | 2e | 82 |
| b | 9f | 69 | 6f | cf | c8 | 10 | e8 | db | cd | 6e | ec | 83 | e6 | aa | 21 | ef |
| c | ba | 4a | ea | 29 | 31 | 2a | c6 | 35 | 74 | fc | e0 | 33 | f1 | 41 | 7f | 17 |
| d | 76 | 43 | cc | e4 | 9e | 4c | c1 | 46 | 9d | 01 | fa | fb | b3 | 92 | e9 | 6d |
| e | 9a | 37 | 59 | eb | ce | b7 | e1 | 7a | 9c | 55 | 18 | 73 | 53 | 5f | df | 78 |
| f | ca | b9 | 38 | c2 | 16 | bc | 28 | ff | 39 | 08 | d8 | 64 | 7b | d5 | 48 | d0 |

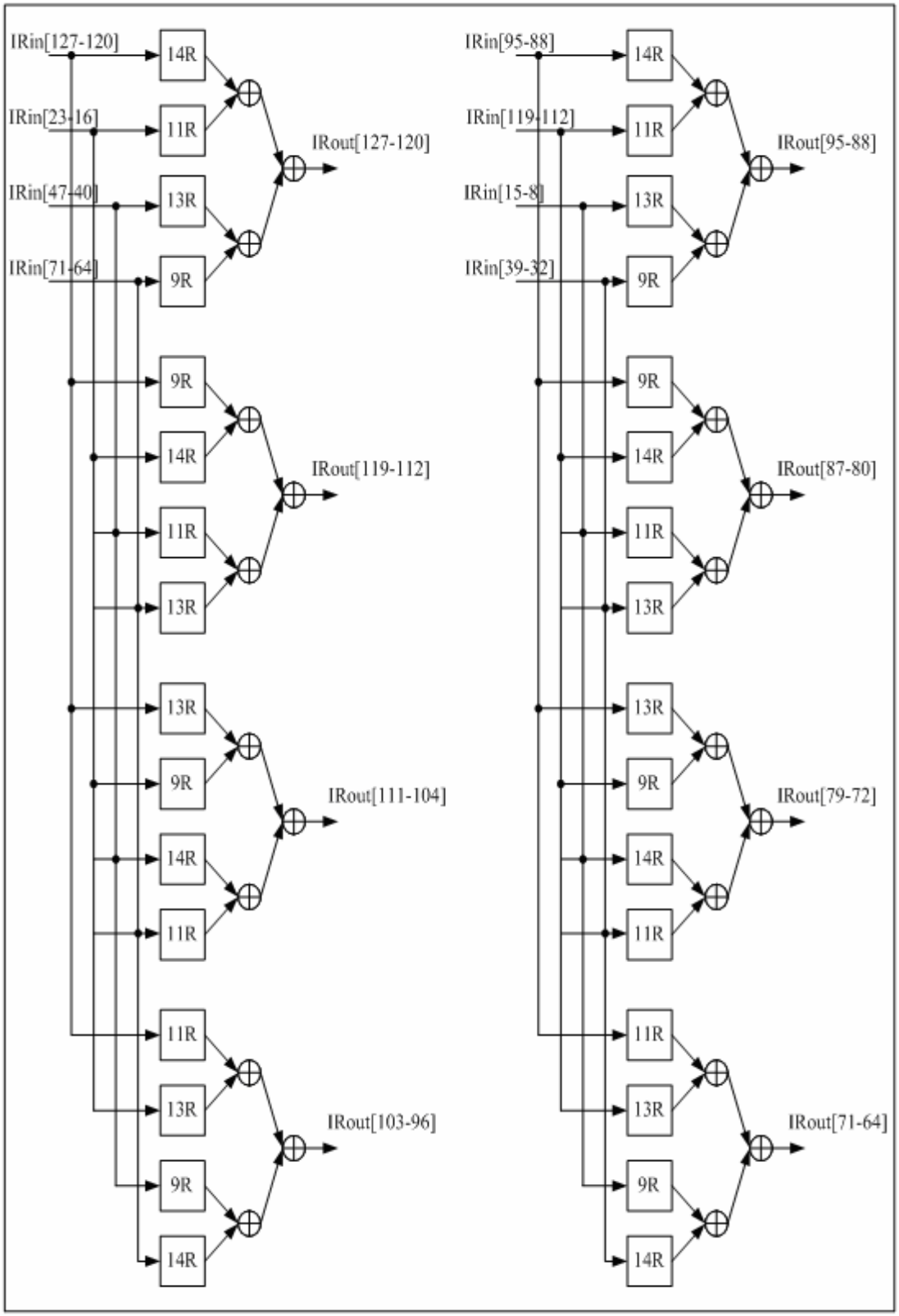


圖 3.12 解密每回合整合運算電路架構圖(一)

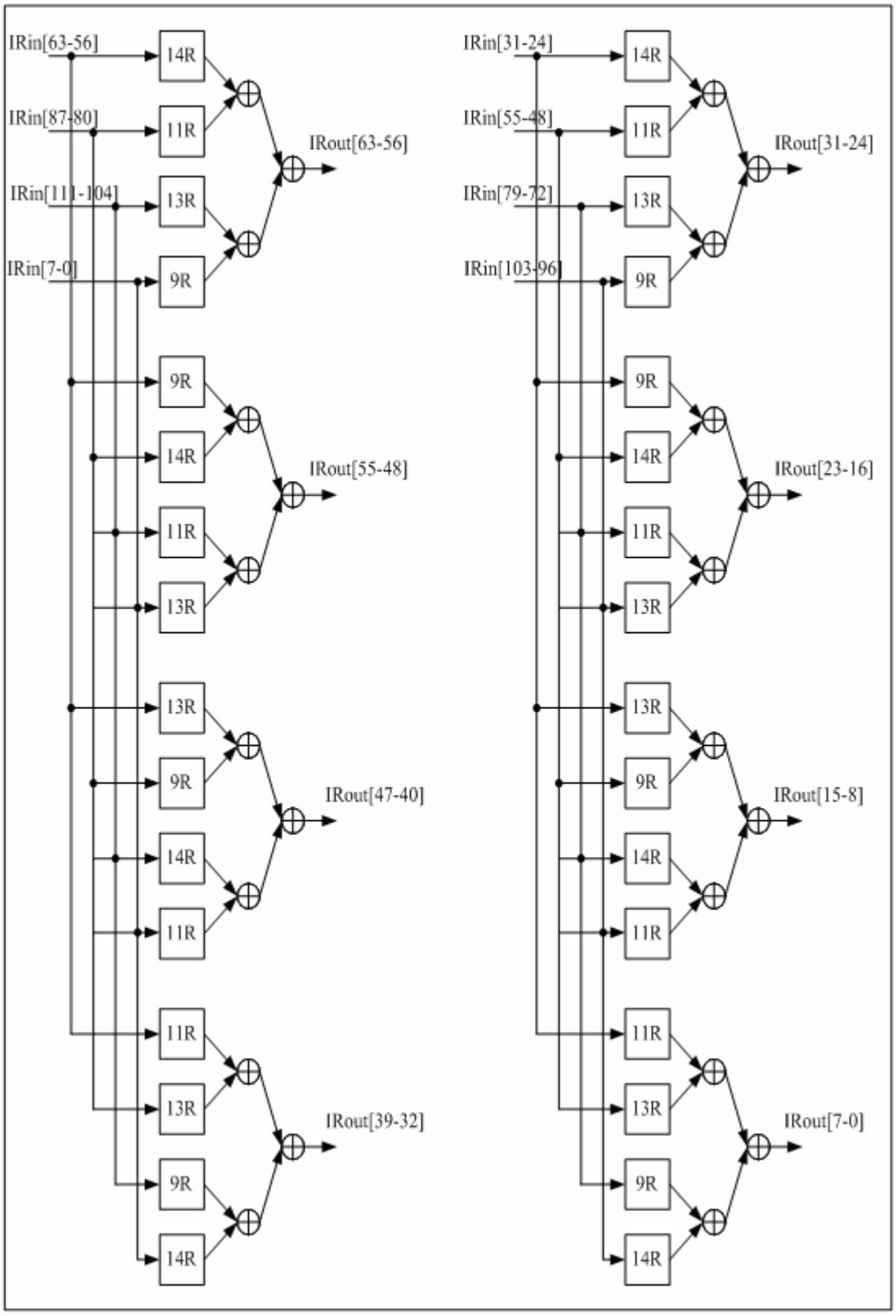


圖 3.13 解密每回合整合運算電路架構圖(二)

步驟 5 加上回合子金鑰, 可以得到下列式子

$s'_{0,0}$ =eR($a_{0,0}$) ⊕bR($Ra_{1,3}$)⊕dR($a_{2,2}$) ⊕9R($a_{3,1}$)⊕$W_{0,0}$;

$s'_{0,1}$ =eR($a_{0,1}$) ⊕bR($a_{1,0}$) ⊕dR($a_{2,3}$) ⊕9R($a_{3,2}$)⊕$W_{0,1}$;

$s'_{0,2}$ =eR($a_{0,2}$) ⊕bR($a_{1,1}$) ⊕dR($a_{2,0}$) ⊕9R($a_{3,3}$)⊕$W_{0,2}$;

$s'_{0,3}$ =eR($a_{0,3}$) ⊕bR($a_{1,2}$) ⊕dR($a_{2,1}$) ⊕9R($a_{3,0}$)⊕$W_{0,3}$;

$s'_{1,0}$ =9R($a_{0,0}$) ⊕eR($a_{1,3}$) ⊕bR($a_{2,2}$) ⊕dR($a_{3,1}$)⊕$W_{1,0}$;

$s'_{1,1}$ =9R($a_{0,1}$) ⊕eR($a_{1,0}$) ⊕bR($a_{2,3}$) ⊕dR($a_{3,2}$)⊕$W_{1,1}$;

$s'_{1,2}$ =9R($a_{0,2}$) ⊕eR($a_{1,1}$) ⊕bR($a_{2,0}$) ⊕dR($a_{3,3}$)⊕$W_{1,2}$;

$s'_{1,3}$ =9R($a_{0,3}$) ⊕eR($a_{1,2}$) ⊕bR($a_{2,1}$) ⊕dR($a_{3,0}$)⊕$W_{1,3}$;

$s'_{2,0}$ =dR($a_{0,0}$) ⊕9R($a_{1,3}$) ⊕eR($a_{2,2}$) ⊕bR($a_{3,1}$)⊕$W_{2,0}$;

$s'_{2,1}$ =dR($a_{0,1}$) ⊕9R($a_{1,0}$) ⊕eR($a_{2,3}$) ⊕bR($a_{3,2}$)⊕$W_{2,1}$;

$s'_{2,2}$ =dR($a_{0,2}$) ⊕9R($a_{1,1}$) ⊕eR($a_{2,0}$) ⊕bR($a_{3,3}$)⊕$W_{2,2}$;

$s'_{2,3}$ =dR($a_{0,3}$) ⊕9R($a_{1,2}$) ⊕eR($a_{2,1}$) ⊕bR($a_{3,0}$)⊕$W_{2,3}$;

$s'_{3,0}$ =bR($a_{0,0}$) ⊕dR($a_{1,3}$) ⊕9R($a_{2,2}$) ⊕eR($a_{3,1}$)⊕$W_{3,0}$;

$s'_{3,1}$ =bR($a_{0,1}$) ⊕dR($a_{1,0}$) ⊕9R($a_{2,3}$) ⊕eR($a_{3,2}$)⊕$W_{3,1}$;

$s'_{3,2}$ =bR($a_{0,2}$) ⊕dR($a_{1,1}$) ⊕9R($a_{2,0}$) ⊕eR($a_{3,3}$)⊕$W_{3,2}$;

$s'_{3,3}$ =bR($a_{0,3}$) ⊕dR($a_{1,2}$) ⊕9R($a_{2,1}$) ⊕eR($a_{3,0}$)⊕$W_{3,3}$;

步驟 6 重覆步驟 2 至步驟 5 總共 9 次, 將所得結果傳送至最後回合運算電路；最後回合運算電路, 為整合 InvSubBytes、InvShiftRows 二個函數為一個數學運算式。如下列式子

$s'_{0,0}$= R($a_{0,0}$); $s'_{0,1}$ = R($a_{0,1}$) ; $s'_{0,2}$ = R($a_{0,2}$); $s'_{0,3}$ = R($a_{0,3}$) ;

$s'_{1,0}$ = R($a_{1,3}$); $s'_{1,1}$= R($a_{1,0}$); $s'_{1,2}$= R($a_{1,1}$) ; $s'_{1,3}$ = R($a_{1,2}$)

$s'_{2,0}$ = R($a_{2,2}$); $s'_{2,1}$ = R($a_{2,3}$); $s'_{2,2}$ = R($a_{2,0}$); $s'_{2,3}$ = R($a_{2,1}$);

$s'_{3,0}$ = R($a_{3,1}$); $s'_{3,1}$ = R($a_{3,2}$); $s'_{3,2}$ = R($a_{3,3}$); $s'_{3,3}$ = R($a_{3,0}$);

圖 3.14 為解密最後回合整合運算電路架構圖, RFin 表示輸入, RFout 表示輸出。

步驟 7 與回合子金鑰相加所得結果

$s'_{0,0} = R(a_{0,0}) \oplus W_{0,0}$;

$s'_{0,1} = R(a_{0,1}) \oplus W_{0,1}$;

$s'_{0,2} = R(a_{0,2}) \oplus W_{0,2}$;

$s'_{0,3} = R(a_{0,3}) \oplus W_{0,3}$ ;

$s'_{1,0} = R(a_{1,3}) \oplus W_{1,0}$;

$s'_{1,1} = R(a_{1,0}) \oplus W_{1,1}$;

$s'_{1,2} = R(a_{1,1}) \oplus W_{1,2}$ ;

$s'_{1,3} = R(a_{1,2}) \oplus W_{1,3}$

$s'_{2,0} = R(a_{2,2}) \oplus W_{2,0}$;

$s'_{2,1} = R(a_{2,3}) \oplus W_{2,1}$;

$s'_{2,2} = R(a_{2,0}) \oplus W_{2,2}$;

$s'_{2,3} = R(a_{2,1}) \oplus W_{2,3}$;

$s'_{3,0} = R(a_{3,1}) \oplus W_{3,0}$;

$s'_{3,1} = R(a_{3,2}) \oplus W_{3,1}$;

$s'_{3,2} = R(a_{3,3}) \oplus W_{3,2}$;

$s'_{3,3} = R(a_{3,0}) \oplus W_{3,3}$;

在硬體實現上每個位元組運算需要 1 個表格及 1 個及斥或閘運算。其中$R(a_{i,j})$為查$a_{i,j}$在Inverse S-box位元轉換對照表(如表 2.4)的數值。

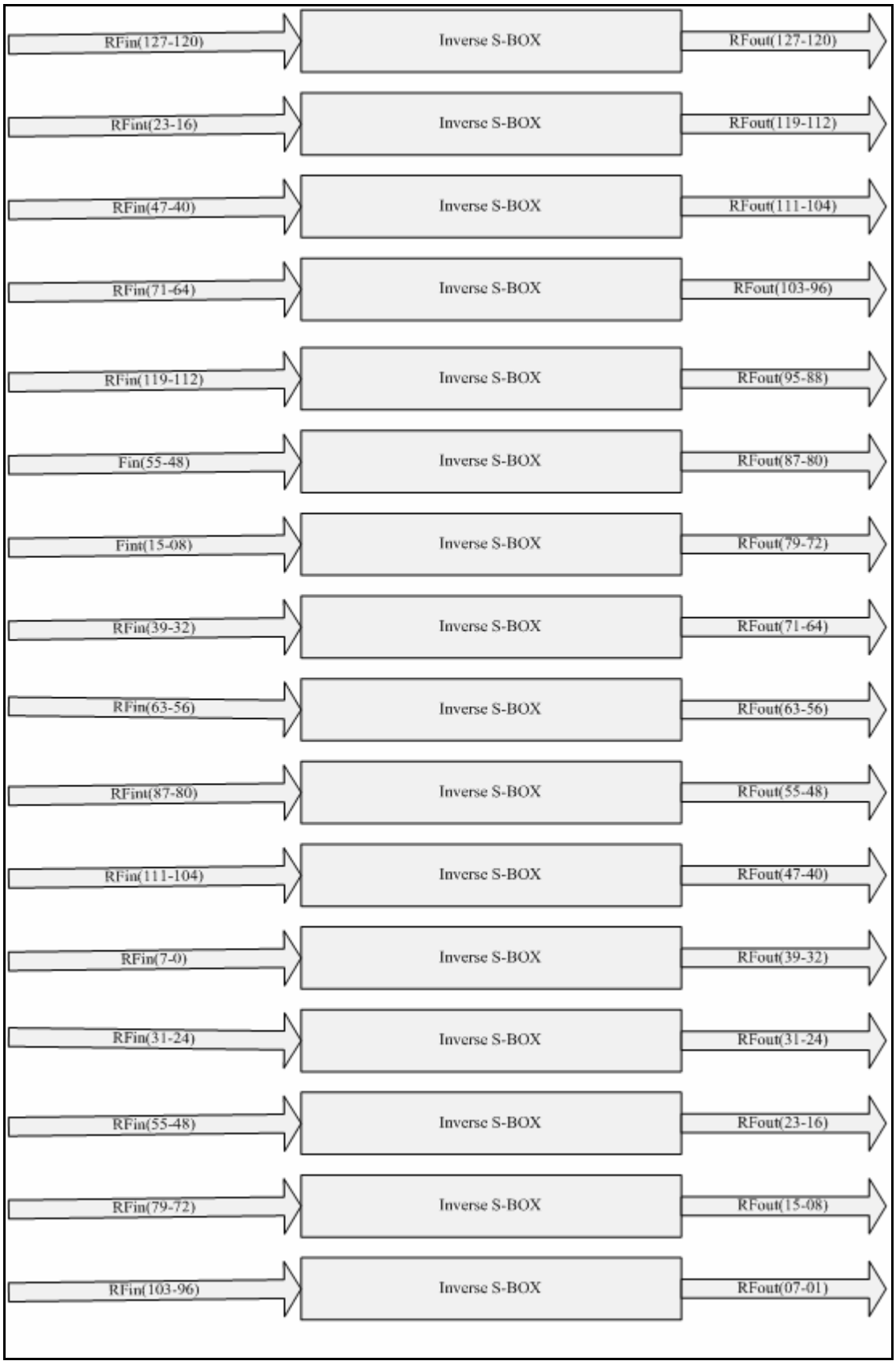


圖 3.14 解密最後回合整合運算電路架構圖

## 3.4 加密金鑰擴展（**Key Expansion**）

金鑰擴展運算包含三個運算函數: Rcon、RotByte 及 Subword。

一般金鑰擴展電路運算結果，第一種是將運算所得到的結果存放在記憶體，當執行加密或解密過程時需要擴展金鑰，則依照順序從記憶體(memory)讀取出來。這種方法是應用在不會經常更改輸入的初始密鑰，當加密或解密電路再次需要各回合子金鑰時,各回合子金鑰不需要重新計算,只需要直接重記憶體中取出。

第二種作法將一般金鑰擴展電路運算所得到每回合子金鑰時，即時提供給加密或解密電路。優點是不需要記憶體。這種被稱作 on-the-fly key 計算方法。

本論文採取第二種作法，解密電路所需要每回合子金鑰時，則是利用加密電路過程中所計算出來最後合子金鑰，輸入解密金鑰擴展電路，計算出每回合子金鑰，並即時提供給解密電路，解密金鑰擴展電路將於下一節詳加說明。

AES-128 加密金鑰擴展電路流程如圖 3.15，從圖可以清楚知道，第 0 回合 W[0], W[1], W[2]及 W[3]即是初始金鑰；若 i 為回合數，第一回合之後， W[3]經過 RotWord 、Subword 函數，再與 Rcon 作互斥或閘，與原先 W[0]數值作互斥或閘後，得到新的 W[4] 數值;與原先 W[1]數值作互斥或閘，得到新的 W[5]；與原先 W[2]數值作互斥或閘，得到新的 W[6]; 與原先 W[3]數值作互斥或閘，得到新的 W[7]進入第二回合之後，再經 RotWord 、Subword 函數，再與 Rcon 作互斥或閘。同理，重覆將所有回合子金鑰完全計算出來。

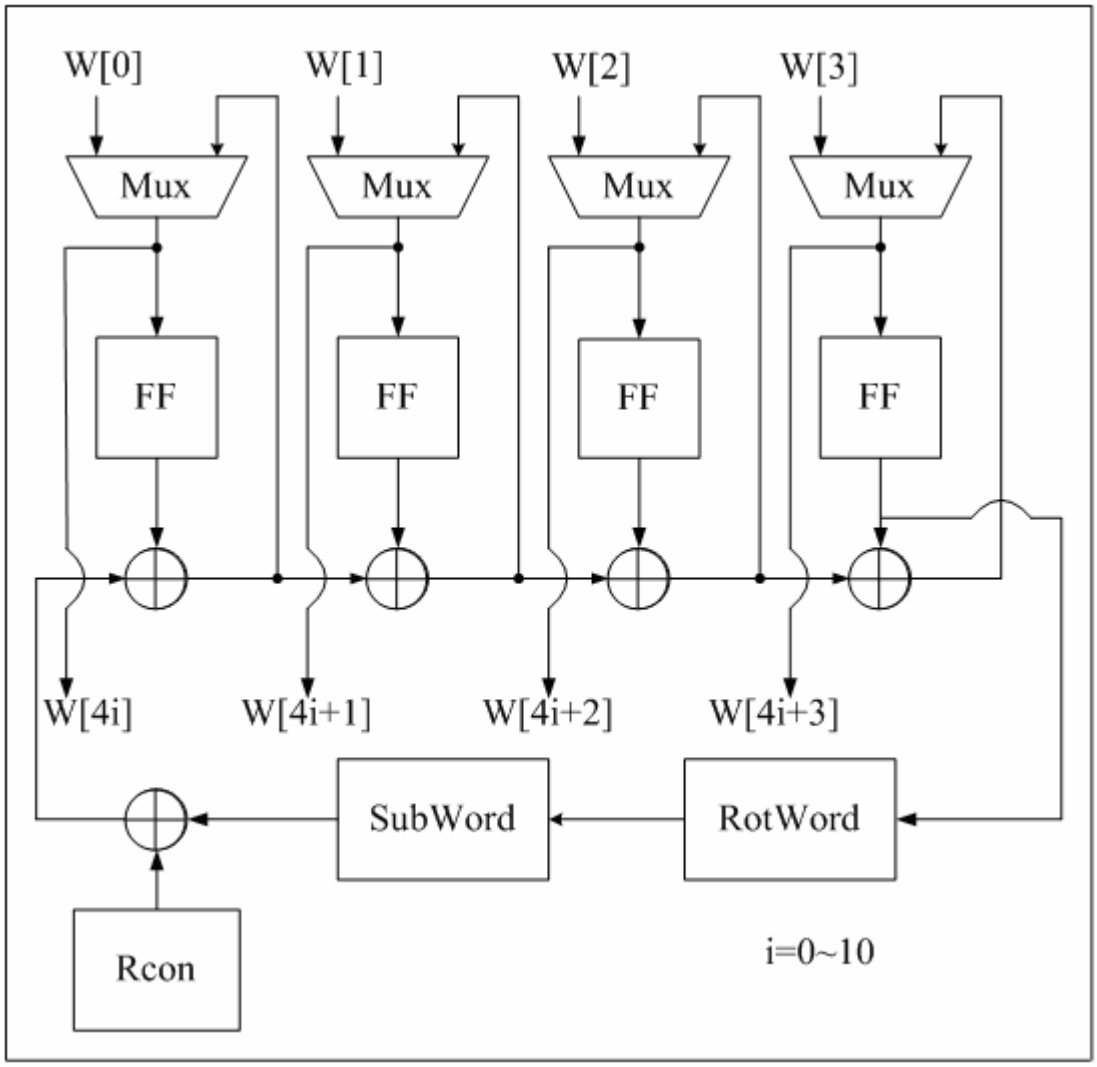


圖 3.15 AES-128 加密金鑰擴展電路流程圖

## 3.5 解密金鑰擴展

解密金鑰擴展電路與加密金鑰擴展電路相同部份，都包含二個運算函數:RotByte 及 Subword。

解密金鑰擴展電路使用 InvRcon 運算函數，只是頭尾順序位置相反。以 AES-128 為例，加密時，首先查 Rcon[1]，接著 Rcon[2]，Rcon[3]. . . . 直到 Rcon[10]。解密時，先查 Rcon[10], 接著 Rcon[9], Rcon[8]. . . . 直到 Rcon[1]。

加密金鑰擴展電路是利用初始金鑰產生回合子金鑰; 解密金鑰擴展電路是利用加密金鑰擴展電路所產生最後回合子金鑰去產生其它回合子金鑰。

AES-128 解密金鑰擴展電路流程，如圖 3.16。它跟加密金鑰擴展電路流程圖 3.15 有少許不同。第 0 回合 W[40], W[41], W[42]及 W[43]即是初始回合金鑰；第一回合開始，W[40]與與原先 W[41]數值作互斥或閘，得到新的 W[37] ; W[41]與與原先 W[42]數值作互斥或閘，得到新的 W[38]; W[42]與與原先 W[43]數值作互斥或閘，得到新的 W[39]; W[39] 再經 RotWord 、Subword 函數與 InvRcon 作互斥或閘，與原先 W[40]數值作互斥或閘，得到新的 W[36], 我們可以得到第一回合子金鑰 W[36]，W[37] ，W[38]及 W[39]。同理，重覆將所有回合子金鑰完全計算出來。明顯得, 加密金鑰擴展電路所產生的回合子金鑰與解密金鑰擴展電路所產生的回合子金鑰，都是相同的。只是兩者順序相反。

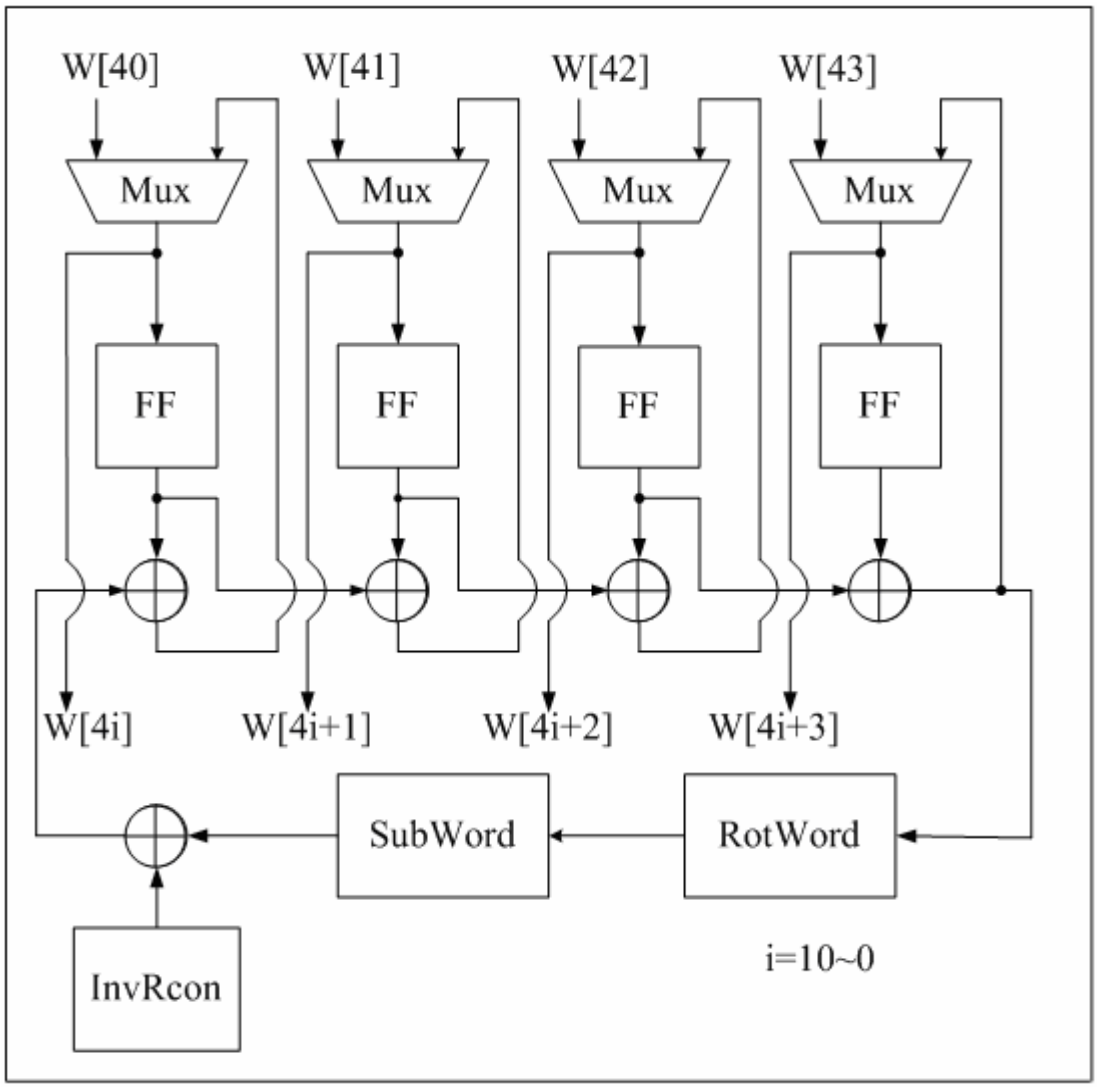


圖 3.16 AES-128 解密金鑰擴展電路流程圖

# 第 4 章 設計驗證

本章介紹本論文的設計驗證流程。在圖 4.1, 首先使用 Matlab 作功能模擬; 在 HDL 設計與驗證,加入時脈,利用 Modelsim 軟體作時序模擬驗證;在 Gate level 模擬，採用 Synopsys 公司的 design analyzer 軟體，加入限制條件，合成想要的電路。最後，使用 Debussy 軟體觀看 wave。

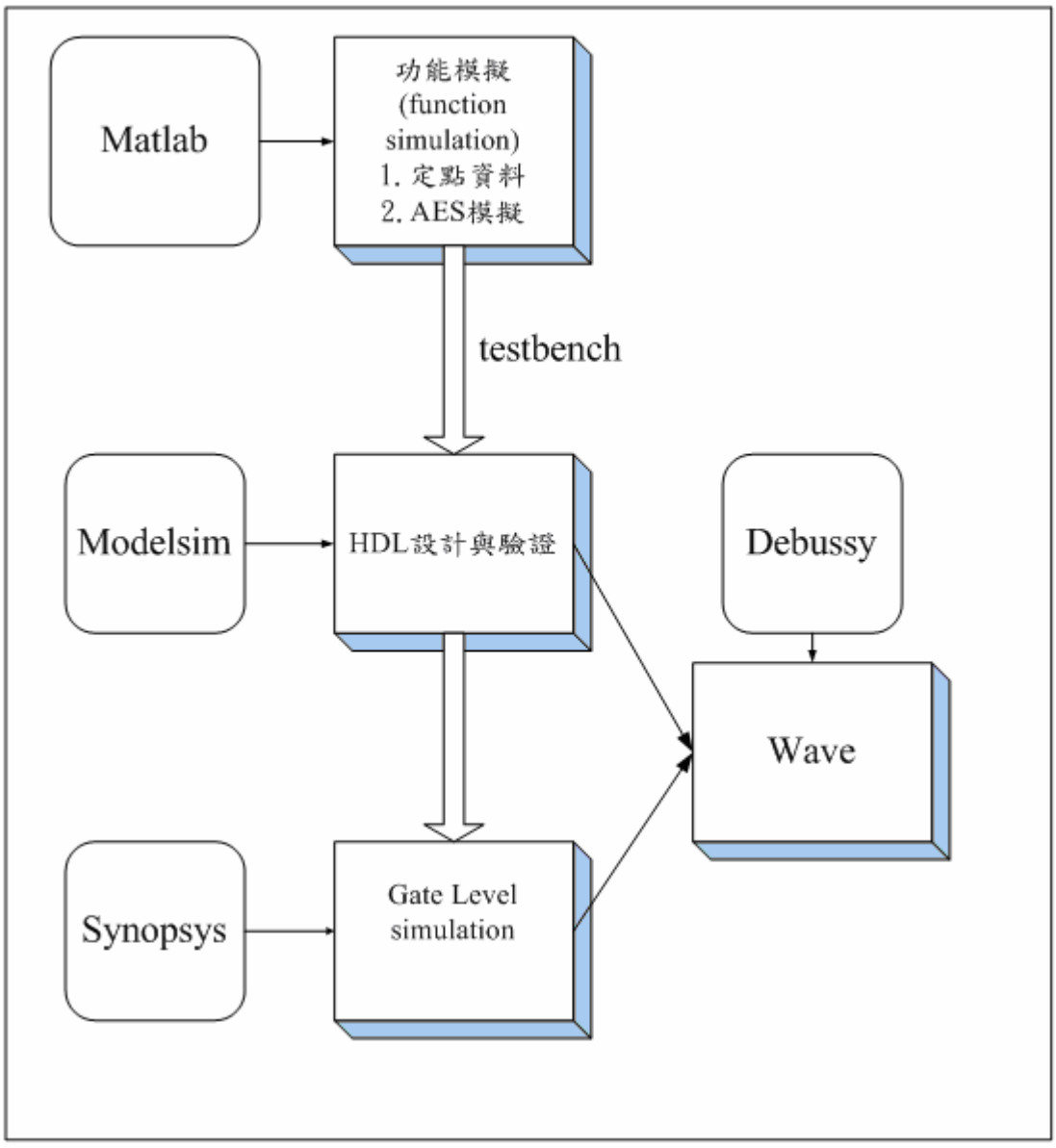


圖 4.1 設計驗證流程

## 4.1 功能模擬(function simulation)

Matlab 軟體具有用法簡易、可靈活運用及程式結構強等特性，因此在本論文研究上，採用 Matlab 軟體產生定點資料及 AES 功能模擬。

### 4.1.1 定點資料

在本論文第三章，表格 3.1~表格 3.6 產生是透過 Matlab 軟體，編寫程式，產生定點資料並加以整理得到這些數學表格。

### 4.1.2 AES 功能模擬

為了驗證設計出來的 AES 晶片是否功能正常，本文依照 AES 的規範，使用 Matlab 程式語言[5]建立 AES-128;AES-192;AES-256 模組。Matlab 運算所得到的結果將跟 verilog 運算出來的結果比較。在功能模擬的階段上，並不需要將 clock 加入電路，只需驗證整個 AES 加密及解密的功能是否正常。圖 4.2 及 4.3 為使用 Matlab 程式模擬 AES-128 加密及解密，

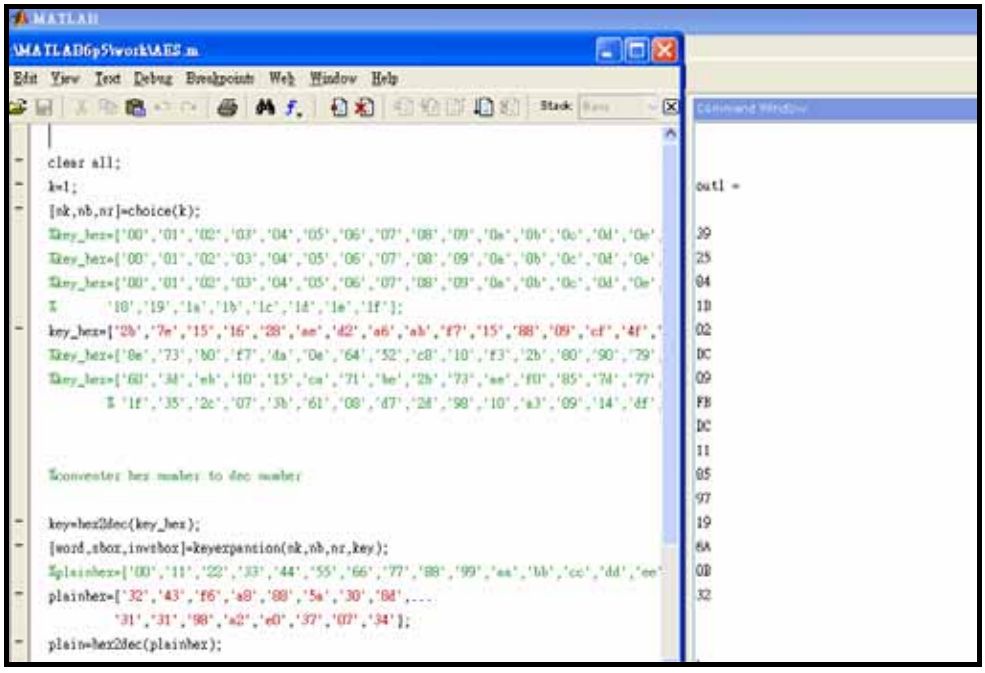


圖 4.2 Matlab 程式模擬 AES-128 加密

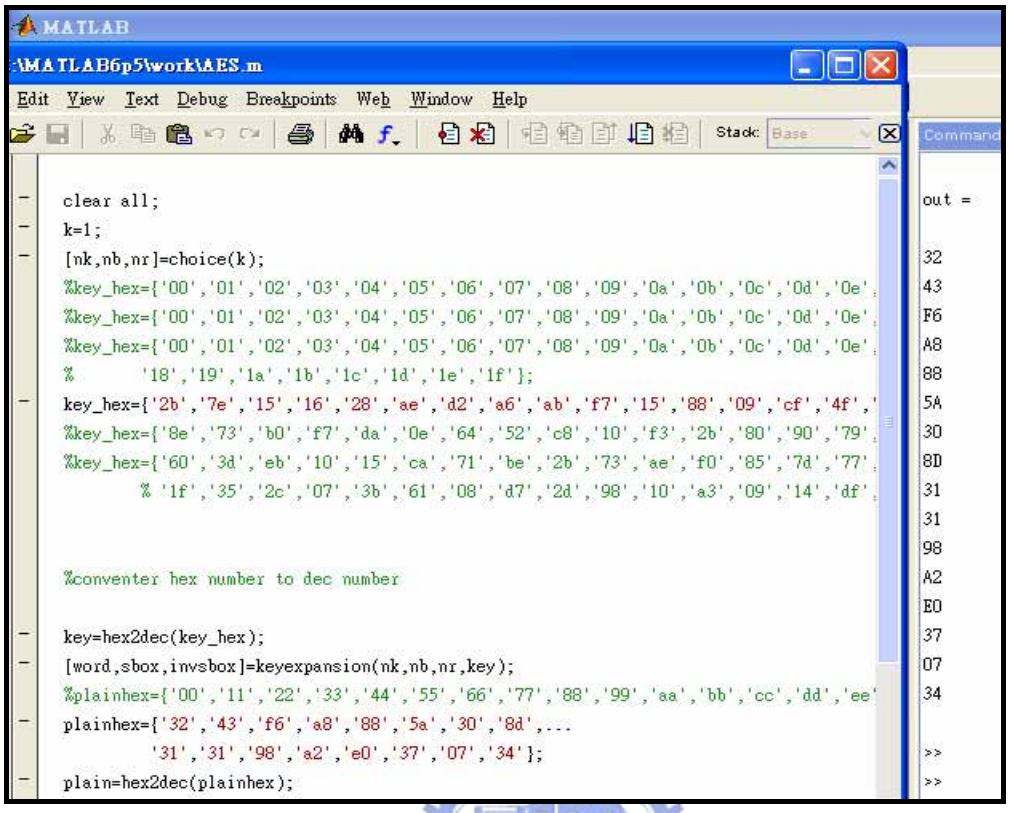


圖 4.3 Matlab 程式模擬 AES-128 解密

## 4.2 HDL 設計與驗證

HDL設計與驗證，本文採用ModelSim軟體[6]來模擬, 原因是Modelsim為工業界最受歡迎的軟體，可以支援window作業系統。在Modeisim邏輯功能模擬的階段上，將clock加入電路，驗證整個AES加密及解密的電路功能是否正常。

圖 4.4 為 Modelsim 模擬流程圖，本文利用 Modelsim 軟體建立 AES 架構，再將測試碼，輸入建立起來的架構。若輸出與 4.1 節使用 Matlab 模擬輸出結果相同，表示完成模擬。若有錯誤產生，則需修改 Modelsim 程式。

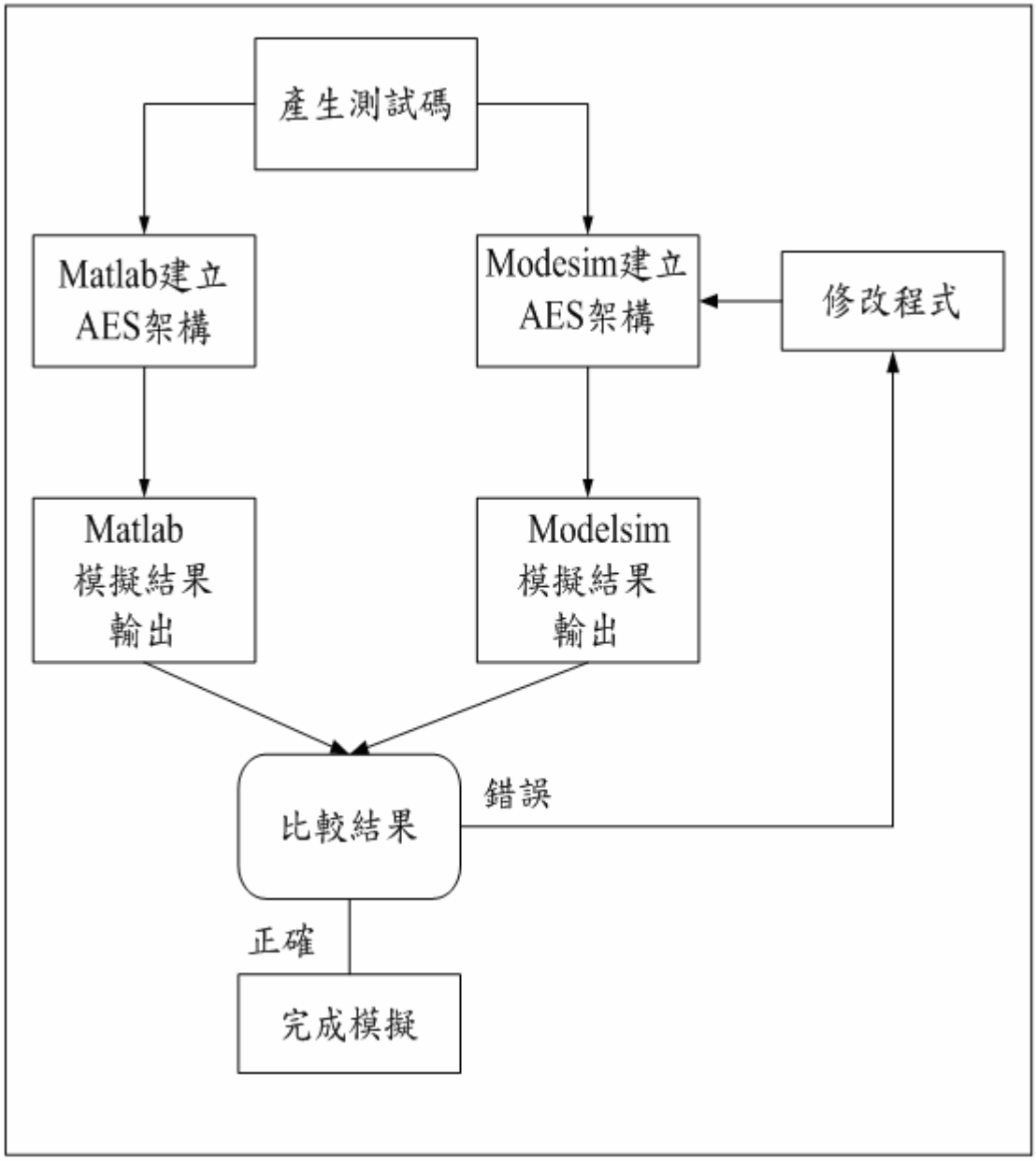


圖 4. 4 Modelsim 模擬驗證流程

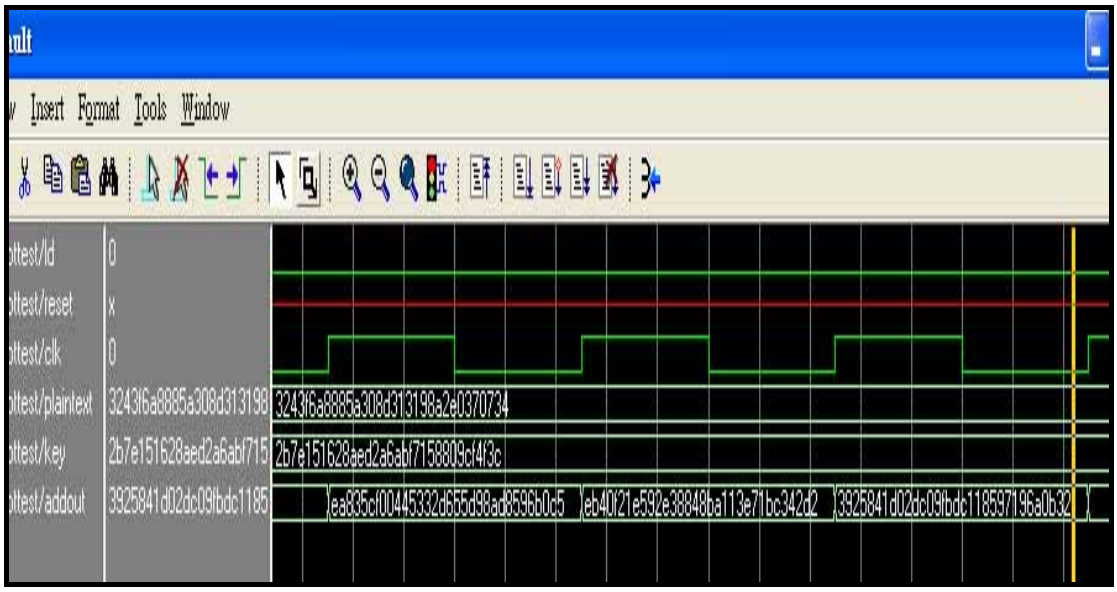


圖 4.5 AES-128 加密模擬

圖 4.5 為使用 Modelsim 軟體建立 AES-128 加密架構, 當輸入如下:

plaintext= $(3243f6a8885a308d313198a2e0370734)_{16}$;

key = $(2b7e151628aed2a6abf7158809cf4f3c)_{16}$;

可得到ciphertext=$(3925841d02dc09fbdc118597196a0b32)_{16}$;

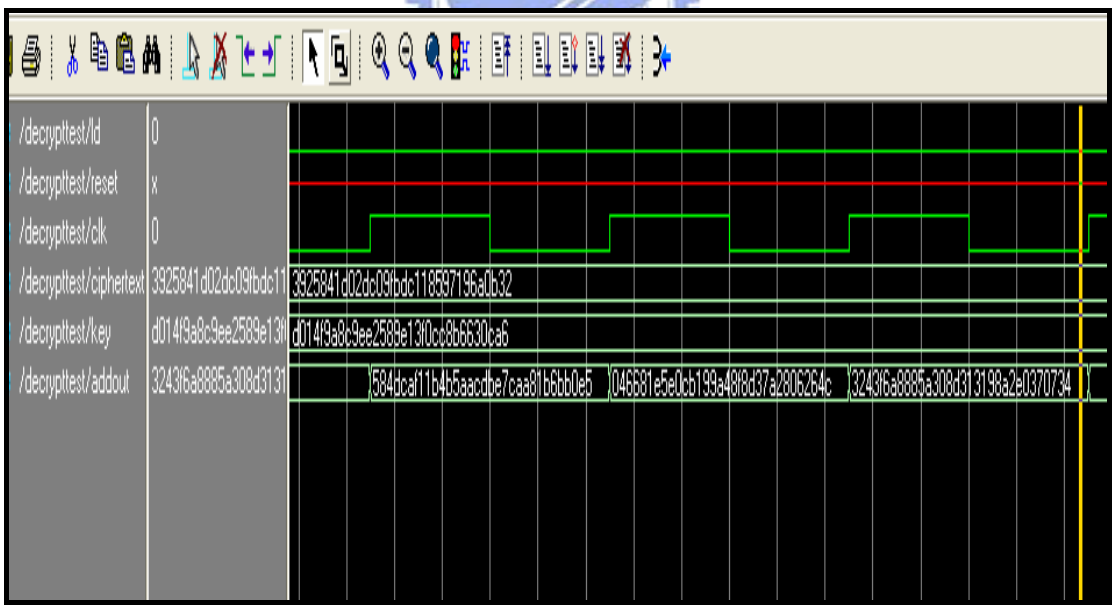


圖 4.6 AES-128 解密模擬

圖 4.6 為使用用 Modelsim 軟體建立 AES-128 加密架構, 當輸入如下:

ciphertext=$(3925841d02dc09fbdc118597196a0b32)_{16}$;

key = $(d014f9a8c9ee2589e13f0cc8b6630ca6)_{16}$;

可得到plaintext= $(3243f6a8885a308d313198a2e0370734)_{16}$;

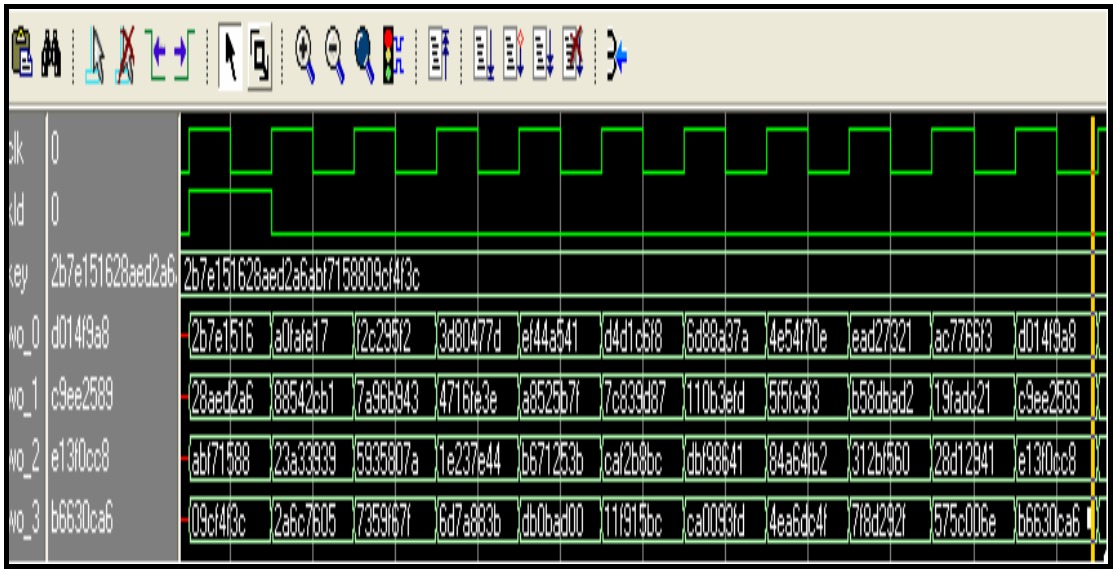


圖4.7 AES-128 加密擴展電路模擬

圖 4.7 為 AES-128 加密金鑰擴展電路模擬, 當輸入如下:

初始密鑰 = $(2b7e151628aed2a6abf7158809cf4f3c)_{16}$;

則連續產生 10 回合子金鑰。

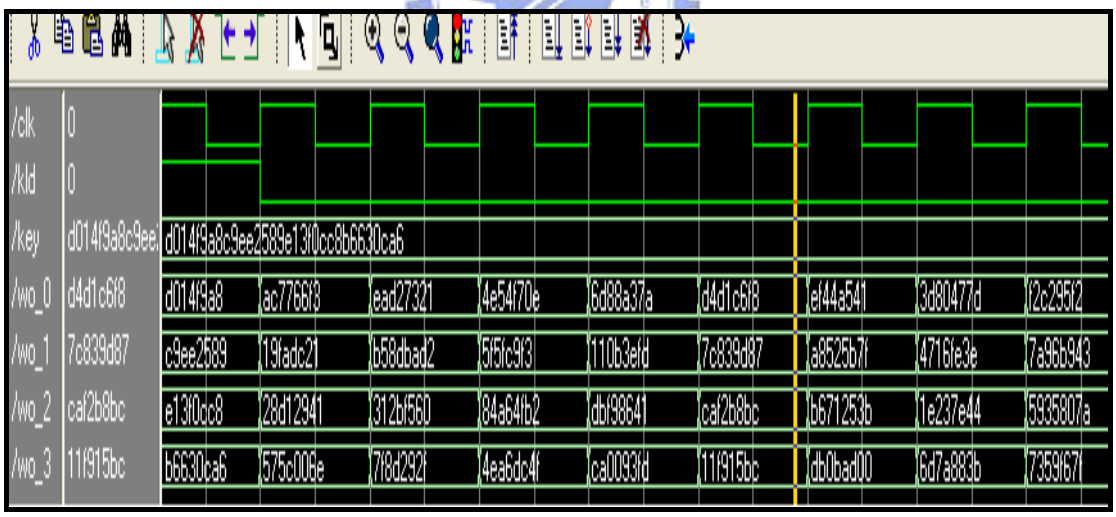


圖 4.8 AES-128 解密擴展電路模擬

圖 4.8 為 AES-128 解密金鑰擴展電路模擬, 當輸入如下:

最後回和子金鑰 = $(d014f9a8c9ee2589e13f0cc8b6630ca6)_{16}$;

則連續產生 10 回合子金鑰及初始密鑰。

## 4.3 邏輯閘 (Gate level)合成模擬驗證

在邏輯閘合成模擬驗證方面，使用 Synopsys 的 design analyzer 軟體，來做電路合成與模擬。Synopsys 的 design analyzer 軟體與前節 Modelsim 軟體最大差別，Modelsim 軟體無法做電路邏輯閘合成。

圖 4.9 為邏輯閘模擬驗證流程圖，由圖上，可以清楚看到，將 verilog 程式描述電路，經由 Synopsys HDL 編譯器讀入，檢查是否服符合 verilog 的語法格式。再經由電路編譯器(Synopsys design complier)，依設定的條件(constraints)及選擇的製程環境條件，去產生邏輯閘層次描述檔(gate level netlist)。

將產生邏輯閘層次描述檔及配合我們提供電路測試碼，執行邏輯閘層次模擬(gate level simulation) 。邏輯閘模擬輸出的結果與 4.2 節 Modelsim 模擬出來的結果比較，確認電路功能是否正常及訊號的波形是否正確。若正確，結束邏輯閘層次模擬。若不正確，檢查問題所在，重新修改 verilog 程式，重覆模擬，直到正確輸出。圖 4.10 及圖 4.11 為 AES 加密及解密 Synopsys 模擬報告輸出。

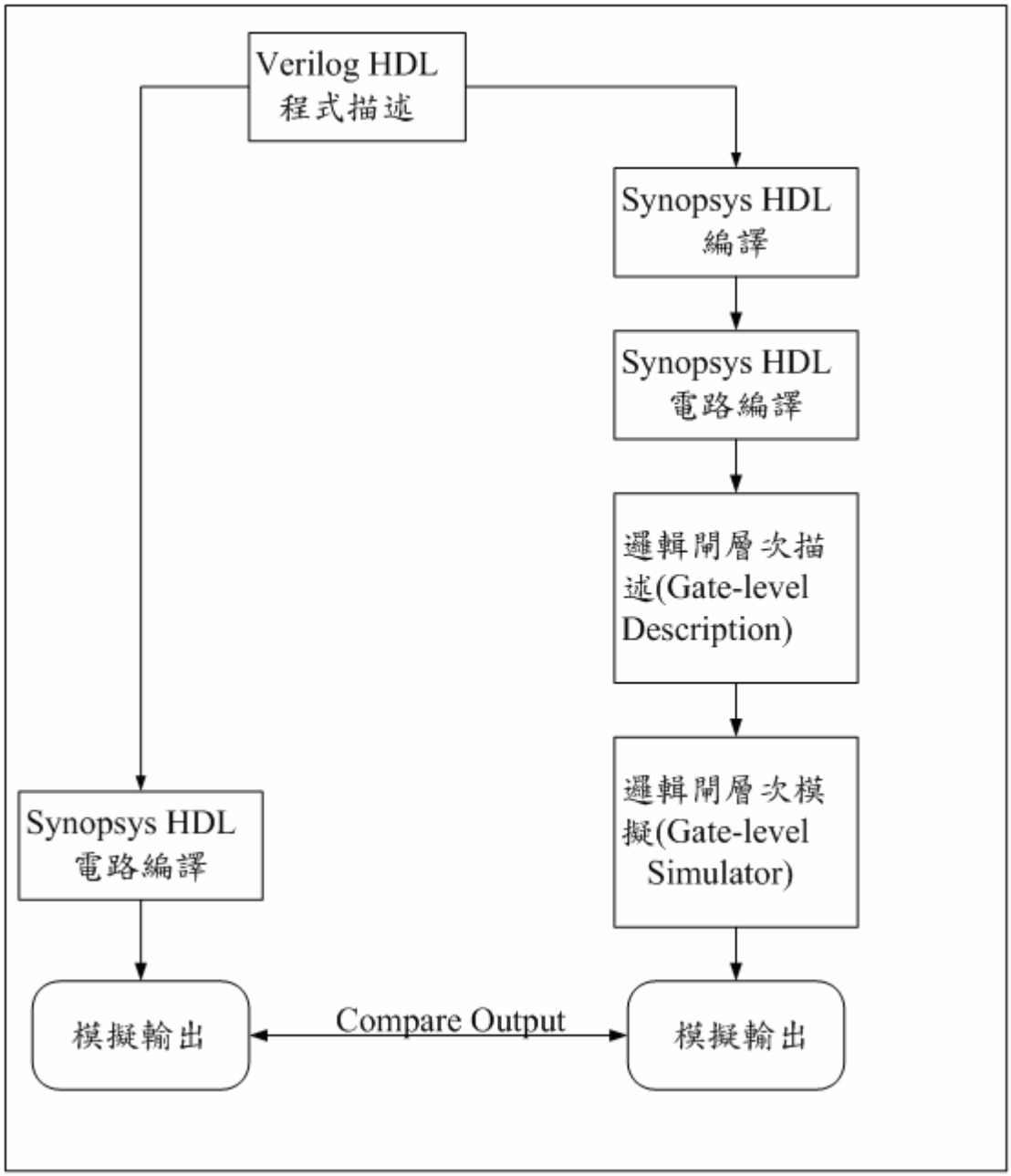


圖 4.9 邏輯閘合成驗證流程

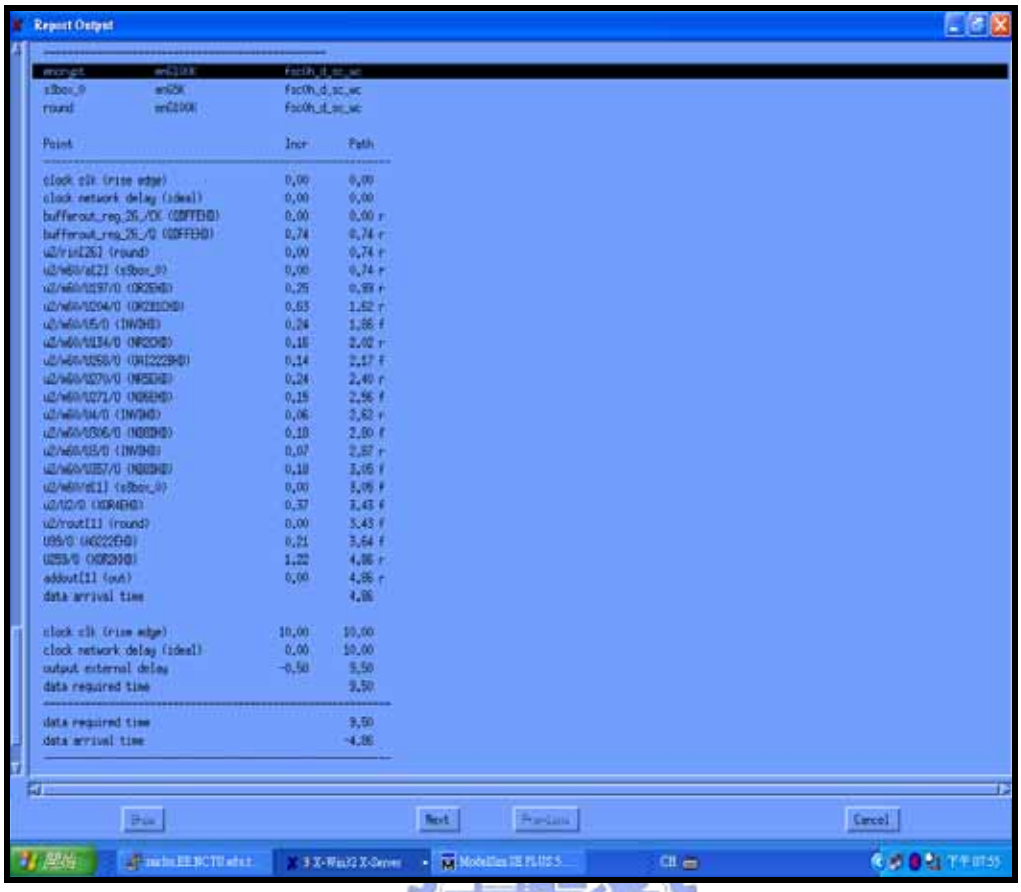


圖 4.10 AES 加密 Synopsys 模擬輸出

圖 4.11 AES 解密 Synopsys 模擬輸出

# 第5章 設計結果與比較

## 5.1 實驗設計結果

表5.1 AES-128實作結果(一般方法)

| 模組 | 製程技術 | 動態功率 | 總閘數 | clock速度 | 頻寬輸出 |
|---|---|---|---|---|---|
| AES-128加密<br>(含金鑰擴展電路) | 聯電<br>0.13um | 7.093mw | 115629 | 125MHZ | 1.6Gbps |
| AES-128解密<br>(含金鑰擴展電路) | 聯電<br>0.13um | 9.667mw | 266800 | 140MHZ | 1.79Gbps |

表5.1未使用改良方法，使用一般方法所得到的結果。

表5.2 AES-128實作結果(平行架構整合查表法)

| 模組 | 製程技術 | 動態功率 | 總閘數 | clock速度 | 頻寬輸出 |
|---|---|---|---|---|---|
| AES-128加密<br>(含金鑰擴展電路) | 聯電<br>0.13um | 11.628mw | 273823 | 204MHZ | 2.6Gbps |
| AES-128解密<br>(含金鑰擴展電路) | 聯電<br>0.13um | 16.148mw | 461027 | 200MHZ | 2.56Gbps |

依第三章平行架構整合查表法，在電路合成部份，使用 Synopsys 公司提供 designer analyzer 合成我們的電路。並配合聯電 0.13um CMOS 標準元件製程當作標準元件庫(library)。在加密部份，對於資料路徑 128 位元，工作頻率最高可達 204MHZ 左右，頻寬輸出 2.6Gbps，總閘數 273823，消耗功率約為 11.628mw; 在解密部份，對於資料路徑 128 位元，工作頻率最高可達 200MHZ 左右，頻寬輸出 2.56Gbps，總閘數 461027，消耗功率約為 16.148mw。整理可得表格 5.2。

## 5.2 實驗結果比較

表5.3 實驗結果比較

| 模組 | 製程技術 | 總閘數 | clock速度 | 頻寬輸出 | Critical path | 備註 |
|---|---|---|---|---|---|---|
| AES-128加密(含金鑰擴展電路) | 聯電 0.13um | 273863 | 204MHZ | 2.6Gbps | 4.86ns | 整合查表法 |
| AES-128解密(含金鑰擴展電路) | 聯電 0.13um | 461027 | 200MHZ | 2.56Gbps | 4.99ns | 整合查表法 |
| C.C Lu[7]* | 0.25um | 31900 | 100MHZ | 609Mbps | | 加密及解密整合法 |
| Namin [8] | 0.18um | 11300 | 52.7MHZ | 100Mbps | | Sbox based on LR |
| Verbauwhede [9] * | 0.18um | 17300 | 154MHZ | 2.29Gbps | | Sbox查表法 |
| Kim[10]* | 0.18um | 28696 | 465MHZ | 1.64Gbps | | 加密及解密整合 |
| Hodjat[11] (AES-128加密) | 0.18um | 473000 | 606MHZ | 7.76Gbps | 1.65ns | Pipeline & subpipeline |

(*表示有合成晶片)

整理實驗數據，並參考其他論文實驗數據，可以得到表格5.3。由表5.3 C.C Lu[7]及Kim[10]的設計，雖然電路速度快。但遠不及我們的頻寬輸出大。

Hodjat[11] 的設計比我們加密電路速度快，頻寬輸出大。在Hodjat所提出論文中，每個回合採用一個運算電路，10個運算回合採用10個運算電路。並採用匯流排管線化(pipeline), 每個運算回合中間及每個運算函數中間使用一個暫存器，所以他的面積比我們大。Hodjat對於資料路徑128位元，工作頻率可達606MHZ, 左右, 論文內頻輸出77.6Gbps, 外頻輸出應為7.76Gbps。

由實驗數據，我們的解密電路比加密電路面積約大1.5倍。原因是本文採用Equivalent inverse decrypt[3]，每回合子金鑰需增加InvMixcolums函數處理，未來可朝減少此部份電路面積努力。

本論文若採匯流排管線化[12][13](pipeline)的作法，在架構上沒有任何問題且運算速度可快2~3倍，頻寬的輸出更大。

# 第六章 結論與展望

## 6.1 結論

在一般AES論文，大部份都利用電路邏輯閘簡化電路，而本論文提供另一種想法-使用平行架構整合查表法;但不同於一般AES查表法論文，一般查表法，只有在加密SubBytes運算時使用Sbox表格，解密InvSubBytes運算時使用Inverse Sbox表格。本論文將AES每回合三個運算函數(加密為整合SubBytes、ShiftRows、MixColumns; 解密為整合InvSubBytes、InvShiftRows、InvMixColumns)，先行利用程式計算轉換成所需表格，省去實行計算電路之困難度。雖然面積增加，但電路運算速度快，頻寬輸出(throughput)高。

本文，因為加密及解密電路是在不同模組，可以同時執行加密及解碼編解碼功能;由於金鑰擴展電路採用on-the-fly key計算方式，可隨時更改初始金鑰，增加系統安全性，但又不影響電路速度。

## 6.2 未來的展望

綜合本文實驗結果，在加密及解密電路速度可達200Mhz以上，確實適合高速傳輸。但因為是查表法，從實際實驗結果，所佔面積較大，未來希望減少晶片面積。

本論文只完成邏輯模擬設計，未包括layout及tape out等合成晶片下線。另外，只實現AES-128編解碼部份，希望能整合AES-128，AES-192及AES-256於同一晶片上。其次，匯流排管線化是未來需加強的工作，本篇論文未採用匯流排管線化技巧去增加晶片速度。

# 參考文獻


[1] 邱榮輝“AES第二回合候選密碼演算法”，資訊安全通訊第六卷第四期，pp. 67-73，2000。

[2] J.Daemen and V.Rijmen, ” AES Proposal: Rijndael”, Document Version2, March9, 1999。

[3] National Institute of Standards and Technology, ” Specification for the Advanced encryption Standard(AES)”, FIPS PUB197, November26, 2001。

[4] National Institute of Standards and Technology, ” Specification for the Data encryption Standard(DES)”, FIPS PUB46-3, October, 1999。

[5] The mathworks, ”Matlab , The Language of Technical Computting”, http://www.matworks.com。

[6] 鄭信源，verilog 硬體描述語言數位電路，儒林圖書公司。

[7] C.C LU and S.Y Tseng, ”Integration of AES(Advanced Encryption Standard) encrypter and decrypter”, Proceeding,Aplication-Specific system Architecture and Processor, pp. 277-285, 2002。

[8] Namin Yu and Heys H.M , “Investigation of compact hardware implementation of the advanced encryption standard”; Electrical and Computer Engineering, 2005. Canadian Conference on1-4, pp. 1069 – 1072, May, 2005。

[9] I.Verbauwhede, P. Schaumont,and H.Kuo, ” Design and performance testing of a 2.29-GB/s Rijndael processor “, IEEE Journal of solid state circuits, 38, Volume  Issue 3, pp. 569 – 572, March 2003。

[10] N.S.kim,T.Mudge,and R.Brown, ”A 2.3Gb/sec fully integrated And synthesis Rijndael CORE”, IEEE Custom Integrated Circuit, Conf, pp. 193-196, Sep, 2003。

[11] A. Hodjat, I.Verbauwhede; ” Speed-area trade-off for 10 to 100 Gbits/s throughput AES processor”, IEEE CNF, signals, systems and computers, 2003.conference record of the Thirty_Seventh Asilomar conference on Volume 2, Vol.2, pp. 2147 – 2150, 9-12 Nov, 2003。

[12] A. Hodjat,;I.Verbauwhede; ”Minimum Area cost for a 30 to 70Gbit/s AES Processor”; IEEE CNF, VLSI, 2004, Proceedings.IEEE computer society Annual Symposium on 9-20 ,pp. 83 - 88 ,Feb, 2004。

[13] A. Hodjat, I.Verbauwhede; ” Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors**”**; IEEE JNL ,computers ,IEEE Transactions on Volume 55, Issue 4, pp. 366 – 372, April , 2006。

[14] Hsin-chung Wang, Chin-hsiu Lin, An-Yei wu; “Design and implementation of cost-efficient AES cryptographic engine”; 台大工程學刊88期, pp. 51-60, Jun, 2003。

[15] Tim Good and Mohammed Benaissa; ” AES on FPGA: from the fastest to the smallest”; Proceedings of CHES 2005, pp. 427-440, LNCS 3659, Spring, 2005。

[16] Alireza Hodjat, David Hwang, Bo-Cheng Lai, Kris Tiri and Ingrid Verbauwhede; ” A 3.84 Gbits/s AES crypto coprocessor with modes of operation in a 0.18-um CMOS Technology”; Proceedings of the 15th ACM Great Lakes Symposium on VLSI 2005, pp. 60-63, ACM, ACM Press, April, 2005。

[17] Akashi satoh, Sumio Morioka, Kohji Tkano, Seiji Muneto;"*A Compact Rijndael* Hardware Architecture with S-Box Optimization.", In Advances in Cryptology ” Asiacrypt 2001, volume 2248 of LNCS, pp. 239-254, Spring, 2001。

[18] Xinmiao Zhang and Keshab K. Parhiter; ” High-Speed VLSI Architectures for the AES Algorithm”; IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 12(9): pp. 957-967, September, 2004。

[19] Shuenn-Shyang Wang and Wan-Sheng Ni; “An Efficient FPGA Implementation of Advanced Encryption Standard Algorithm” ;ISCAS 2004, Proceedings, Volume~2, pp. 597-600, IEEE Computer Society, May, 2004。

# 簡　歷

姓名:　郭　昌　　華

出生年月:民國57年4月22日

籍貫:台灣省台中縣

e-mail:sonice_kuo@yahoo.com.tw

主修科目:數位積體電路

類比積體電路

影像處理

ASIC與大型邏輯設計

數位訊號處理

計算機架構

數位通訊