一個AES內建CBC模式加密與解密之VLSI架構
A VLSI Architecture for AES with CBC mode

林銘波(Ming-Bo Lin)

國立台灣科技大學電子工程系

Email: mblin@mail.ntust.edu.tw

曾英彰(Ing-Jang Tzeng)

謝名峻(Ming-Chun Hsieh)

國立台灣科技大學電子工程系

Email:m9702111@mail.ntust.edu.tw

中文摘要

本論文中提出了一個具有管線(pipeline)架構的AES plus CBC加解密VLSI架構。依據AES演算法實現出來的加解密晶片輸入的每一筆資料為128位元，金鑰為可選擇的128位元、192位元以及256位元。在使用者介面的選擇性方面，所設計的晶片相容於微處理機控制介面，可操作於資料寬度為8位元、16位元以及32位元三種模式，並且內建了ECB、CBC、CFB以及OFB四種操作模式，供使用者彈性的搭配微處理器使用，依情況選擇適合的操作模式；在降低功率消耗方面，子金鑰是在加解密前由金鑰擴展程序產生，然後儲存於RAM裡面。對於每筆輸入的資料而言，由於不需反覆計算相同的子金鑰，因此可以降低功率的消耗。在位元組替代轉換部分，利用歐幾里得演算法建立一個新的乘法反元素替換表，取代原本的S-box與InvS-box，面積約減少36.45%。

AES plus CBC加解密VLSI架構已經分別在Xilinx的Vertex 400型FPGA以及TSMC 0.25 μm元件庫(Cell Library)上實現與驗證。在FPGA設計部分，工作頻率為30 MHz，資料處理量最高為230.4 Mbps，消耗了Vertex 400型FPGA實驗板上73%的LUT與80%的Block RAM；在元件庫設計的部分，工作頻率為156.25 MHz，資料處理量最高為1200 Mbps，核心(core)面積為1671.73 μm × 1671.73 μm，等效閘數量(gate count)約為76610個，消耗功率為128.4 mW。

關鍵字：先進加密標準、Rijndael演算法、歐幾里得演算法、電子密碼書模式、密文區段串接模式、密文反饋模式、輸出反饋模式。

Abstract

In the thesis, a pipelined architecture of AES plus CBC Encryption/Decryption IP (Intellectual Property) is proposed. This architecture implements the AES algorithm, the input data is a sequence of 128 bits and the cipher key is a sequence of 128, 192, or 256 bits. To provide the flexibility of interfacing with common microprocessors, the data bus width can be set to 8, 16, or 32 bits. To apply AES in a variety of applications, four “modes of operation” (ECB, CBC, CFB, and OFB) have been defined in the IP. To reduce the power consumption, the same set of Round Keys will not be generated many times for every data blocks because all of the Round Keys have been generated and stored in RAM by Key

Schedule. Obviously, it consumes less power to compute the same set of Round Keys. Compared with the widely used lookup-table architecture, the Euclid's multiplicative inverse lookup-table architecture used in the thesis may reduce the hardware overhead of the S-box and InvS-box by an amount of 36.45%.

The AES plus CBC Encryption/Decryption IP has been implemented and verified with both Xilnx Vertex 400 FPGA and TSMC 0.25 μm cell library. In the FPGA part, it operates at 30 MHz and can achieve a high throughput of 230.4 Mbps. It takes up LUTs of 73% and Block RAM of 80% in FPGA board. In the cell-based part, it operates at 156.25 MHz and can achieve a high throughput of 1200 Mbps. The core occupies the area of 1671.73 μm×1671.73 μm, which is approximately equivalent to 76610 gates, and consumes about 128.4 mW in the typical operating condition.

Keywords: AES, Rijndael algorithm, Euclid's algorithm, EBC, CBC, CFB, OFB.

一、簡介

目前最廣為使用的資料加密標準(Data Encryption Standard，簡稱DES) [7]是1977年由國家標準局(National Bureau of Standard)所採用，用來保護敏感性的資料內容。國家標準局現在已改名為國家標準與技術協會(National Institute of Standards and Technology，簡稱NIST)。但是隨著加解密破解技術的進步以及電腦運算速度的提升，在1997年DES已經被DESCHALL [3] team成功破解，使得原本的資料保護有被攻擊的危險。在2001年，NIST公佈了AES [1]為下一代加密標準，以便能夠繼續讓網路上傳輸的資料保持高度安全性。AES是採用對稱性區段加密法的設計原理[2]，不管在傳輸所占的頻寬以及加解密資料的速度與安全性上等方面，都明顯的優於其它加解密演算法。因此AES勢必成為未來密碼系統的主流。

本論文中將對AES所使用的Rijndael演算法作效能分析，並作為論文中軟智產實現之核心演算法。由於AES只是達成資料安全的一個建構基石。為了能將AES結合在各種應用程式與硬體架構中，除了原本的基本加解密模式，在晶片內部特別架構了四種特殊的操作模式[7]，包含電子密碼書模式(ECB)、密文區段串接模式(CBC)、密文反饋模式(CFB)、輸出反饋模式(OFB)的AES plus CBC加解密軟智產(Soft Intellectual Property，簡稱Soft IP)。此加解密軟智產應用於微處理器的副處理器(Coprocessor)上，所以AES plus CBC加解密軟智產相容於微處理器使用介面。另外在金鑰位元長度的考量上，也會因為現今電腦技術的進步，而加快破解密碼系統的時間。配合Rijndael演算法，我們將實現加密金鑰為可選擇的128位元、192位元及256位元，依不同的金鑰長度分別稱為"AES-128"、"AES-192"及"AES-256"。在設計AES plus CBC加解密軟智產時，將透過模組化的設計理念，期望達到安全度的增強，與各模組的獨立性與重複使用性。

本論文在章節的安排上，第二節將對AES做基本的介紹，包括AES的演進、數學背景、Rijndael演算法說明與四種加解密模式。第三節將以硬體實現之面積與效能兩個觀點作考量，分析AES各種不同的硬體架構之優缺點，並且設計出合乎我們需求的硬體電路。 第四節介紹AES plus CBC架構在FPGA的實現結果以及測試程式執行的結果。第五節為結論。

二、AES簡介

AES(Advanced Encryption Standard)是屬於128位元的對稱式區塊密碼系統。在加解密的過程中，允許的資料長度為128位元，而對應的加

解密金鑰區塊長度，可分為128位元、192位元以及256位元。並依據金鑰的長度重複10、12與14次回合數。總觀Rijndael演算法的優點如下：有足夠的安全性，對抗目前已知的所有攻擊方法。對於軟體的設計及硬體的實現都具有很高執行效率。執行速度快，並且可跨越不同的作業平台。演算法以區塊的觀念做加解密的動作，所用的金鑰長度可以依不同的需求做三種不同的選擇。

2.1、Rijndael加密演算法

Rijndael加密演算法在加密時，會將明文複製成狀態陣列，與初始的主金鑰相加，然後經過($Nr-1$)個回合運算，再加上最後一個不完全的回合運算所完成。每一回合基本上可以分為四個部分：SubBytes、ShiftRows、MixColumns和AddRoundkey。整個加密運算過程的虛擬碼如下：

```
EnCipher (byte in[4*Nb], byte out[4*Nb],
          word w[Nb*(Nr+1)] )
{
  byte state [4, Nb];
  state = in;
  AddRoundKey (state, w [0, Nb-1]);
  for (round = 1; round < Nr; round ++){
       SubBytes (state);
       ShiftRows (state);
       MixColumns (state);
       AddRoundKey (state,w[round*Nb,
                           (round+1)*Nb-1]);
  }    //end for
  SubBytes (state);
  ShiftRows (state);
   AddRoundKey (state, w[Nr*Nb, (Nr+1)*Nb-1]);
  out = state;
}
```

2.2、Rijndael金鑰擴展程序

金鑰擴展程序(Key expansion schedule)是將使用者所輸入的主金鑰使用線性位移轉換、非線性運算代換以及XOR的邏輯運算來加以擴充。擴展的金鑰需配合Nb個行字元，經過Nr回合的運算過程，產生總數達$Nb(Nr+1)$個行字元的回合金鑰，金鑰擴展程序的虛擬碼如下：

```
KeyExpansion (byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
{
  for (i = 0; i < Nk; i ++)
     {w[i]=(key[4*i],key[4*i+1],key[4*i+2], key[4*i+3]); }
  for(i = Nk; i < Nb*(Nr+1); i ++)
     {temp = W[i-1];
      if (i % Nk == 0)
         {temp=SubWord(RotWord(temp))^Rcon[i/Nk]; }
      else if ((i % Nk == 4) && (Nk == 8))
         {temp=SubByte(temp)}
      w[i] = w[i-Nk] ^ temp; }    //end for
}
```

2.3、Rijndael解密演算法

由於Rijndael是屬於對稱性質的演算法，所以基本上將加密的過程逆向操作就是解密演算法(Inverse Cipher Algorithm)。原本加密的四個步驟中的運算，在解密的過程中則需要做一些改變，成為InvShiftRows、InvSubBytes和InvMixColumns，而AddRoundKey運算所使用的解密金鑰順序則是與加密時的順序相反。所以整個解密運算過程的虛擬碼如下：

```
InvCipher (byte in[4*Nb], byte out[4*Nb],
           word w[Nb*(Nr+1)] )
{
  byte state [4, Nb];
  state = in;
  AddRoundKey (state, w[Nr*Nb, (Nr+1)*Nb-1]);
  for (round = Nr-1; round <1 ; round --)
  {
       InvShiftRows (state);
       InvSubBytes (state);
       AddRoundKey (state, w[round*Nb,
                           (round+1)*Nb-1]);
       InvMixColumns (state);
  }    //end for
  InvShiftRows (state);
  InvSubBytes (state);
  AddRoundKey (state, w [0, Nb-1]);
```

```
    out = state;
}
```

2.4、密文區段串接模式

密文區段串接模式(Cipher Block Chaining Mode，簡稱CBC模式)是為了克服ECB模式安全上的缺失，所需要的技術是：如果相同的明文區段重複出現的話，結果會產生不同的密文區段。CBC模式就是符合這個需求的一個簡單的方法，如圖1所示。在這個架構中，加密演算法的輸入是由目前的明文區段與前一回合的密文區段XOR之後所形成的；並且所有的區段都使用相同的金鑰。事實上的效應是，將一連串處理明文區段的程序串接起來。對每個明文區段而言，它被傳入加密函數時的形式與它本身的成分並無固定的關係。因此，重複出現的128位元的明文區段就不會被推導出來。

就解密而言，整個密文區段都會傳給解密演算法。而其結果與前一回的密文區段XOR之後就會得到明文區段。為了要證明這個方法是對的，先定義：

$$C_n = E_k[C_{n-1} \oplus P_n]$$

然後

$$D_k[C_n] = D_k[E_k(C_{n-1} \oplus P_n)]$$

$$D_k[C_n] = (C_{n-1} \oplus P_n)$$

$$D_{n-1} \oplus D_k[C_n] = C_{n-1} \oplus C_{n-1} \oplus P_n = P_n$$

為了要產生第一個密文區段，所以需要將一個起始向量(initialization vector，IV)與第一個密文區段XOR起來。在解密時，會將IV與解密演算法的第一個輸出XOR起來，這樣就可以得到第一個明文區段。

傳送與接收雙方都必須知道IV。為了達到最高的安全性，IV與金鑰都必須妥善保管。要達到這一點的話，可以用ECB加密法來傳送IV。

總括來說，因為CBC的串接機制，使得這個模式很適合用來對長度超過128位元的訊息加密。另外除了用CBC來達成保密性之外，CBC模式也可以用來達成確認性。

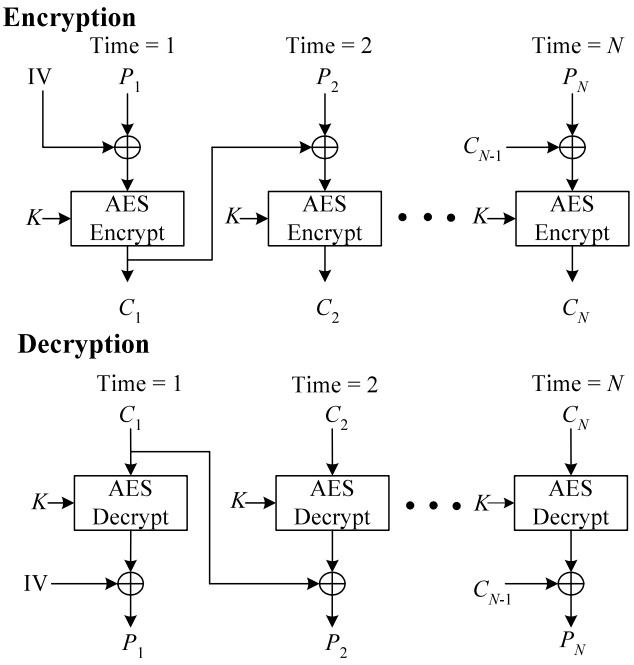

圖1. 密文區段串接模式

三、AES密碼器之架構分析與改進

在AES密碼器的硬體架構建立方面，配合Rijndael演算法，加密金鑰長度可以為128位元、192位元及256位元，依不同的金鑰長度分別稱為“AES-128”、“AES-192”以及“AES-256”。而運算回合的次數也分為10次、12次以及14次，至於明文與密文長度則是固定的128位元。若是使用CFB模式或OFB模式時，長度不足的部分就會被0取代。下列將考慮實現AES各種不同的架構，以面積和效能兩個觀點作考量，從中選擇一個適合的架構。

3.1、回合運算架構分析

Rijndael演算法主要包含兩個核心部分，其中之一就是回合運算，因此慎選回合運算的架構關係著將來實現硬體的面積與效能，下面將舉出四種常用的回合運算架構[8]，並分析優缺點。

3.1.1、基本回合運算架構

如圖2所示，將一個回合運算當成一個組合

邏輯電路，配合一組暫存器與多工器，首先資料從輸入端輸入，經過一個回合後，第二個回合透過多工器選擇從輸出端回饋的資料，依照 Nr 值決定回饋次數，直到完成加解密為止，這個架構基本上是最省面積的。

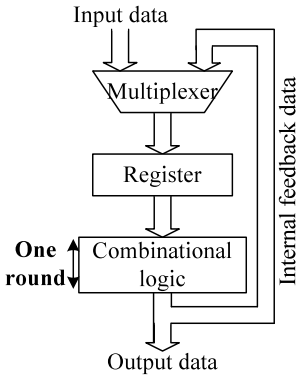

圖2. 基本回合運算架構

3.1.2、循環展開之回合運算架構

如圖3所示，循環展開(loop unrolling)是將回合運算的電路再增加，最多為Nr值。與基本回合架構相較下，其優點為為可以減少多工器與暫存器的延遲時間，增加的速度如下式所列：

$$speed_{lu}/speed_{ba} = (1+\tau)/(1+\tau/k) \qquad (1)$$

其中，τ是多工器與暫存器的延遲時間佔一回合延遲時間的比例，一般約為0.15～0.2之間[8]；k是循環展開所用的硬體回合數目。當k為最大值時，(1)式中 (τ/k) 的部分幾乎為0，所以速度提升最佳的情形為 $(1+\tau)$。因此硬體資源資加了k倍，可是得到的速度提升效應卻是很少，僅增加約15%～20%，因此一般不建議使用此架構。

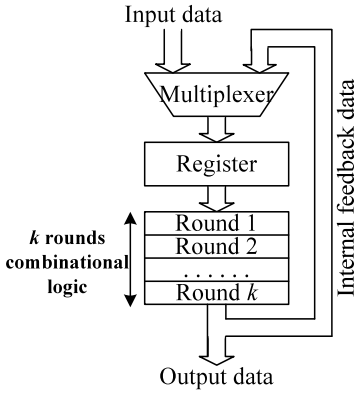

圖3. 循環展開回合運算架構

3.1.3、內部回合管線架構

使用管線(pipeline)能大量的增加資料處理量(throughput)。內部回合管線(Inner-round pipeline)是將一回合分成k個管線階段(pipeline stage)，因此一次可以運算k筆資料，如圖4所示。與基本回合架構比較，其資料處理量約可以增加k倍。使用內部回合管線架構的優點為只需增加 $(k-1)$ 組暫存器即能得到不錯的資料處理量。因此，本論文將採用此種架構。

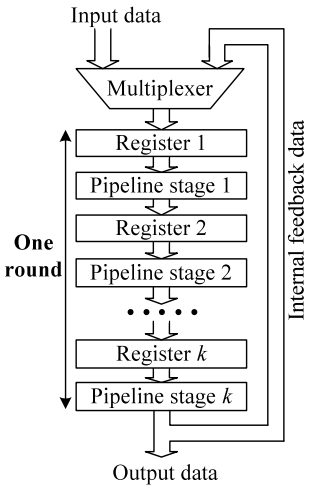

圖4 內部回合管線架構

3.1.4、外部回合管線架構

外部回合管線架構(Outer-round pipelining)是循環展開回合架構的再延伸，在回合與回合中間加上暫存器。如圖5所示，將一個回合當成一個管線階段，最多可以分成Nr個管線階段。在ECB模式下，這種設計的資料處理量幾乎是以倍數增加。分成k個管線階段，資料處理量就能提升k倍，但是面積也會增加k倍。如果是在回授模式下(非ECB的模式)，資料處理量就無法隨著管線階段的增加而提升。因此在非ECB的模式或是於有面積限制的情形下，不建議使用外部回合管線架構。

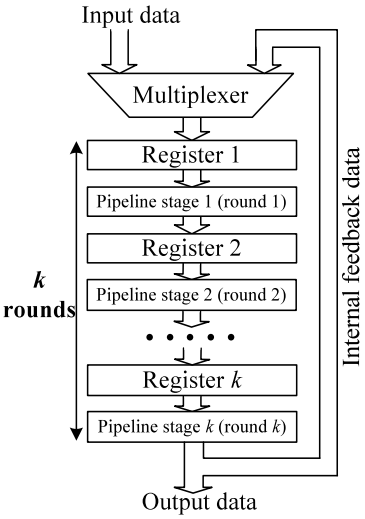

圖5　外部回合管線架構

3.2、回合運算單元

回合運算單元是AES加解密時的核心部分，每回合運算中有四個不同的運算單元，分別是位元組替代轉換、列移位轉換、混合行轉換以及加入回合金鑰轉換。仔細分析演算法的四個運算過程，發現Rijndael演算法是適合使用管線架構，增加資料處理量。而區分管線階段的方式是將回合運算分成數個部分，依運算性質區分，其中列位移轉換只是位元組的位移，可以利用接線完成，所以我們將列位移轉換與混合行轉換這兩個部分的運算分成一個管線階段。位元組替代轉換與加入回合金鑰轉換各分成一個管線階段。因此整個Rijndael加解密的回合架構共可分成三個內部回合的管線階段，如圖6所示。

至於解密的部分，我們採用等價解密演算法，即在解密過程中，回合轉換運算的順序與加密的順序一樣。同樣的，我們也將解密的部分分成三個管線階段。反列位移轉換與反混合行轉換為同一個管線階段，反位元組替代轉換與加入回合金鑰各為一個管線階段。由於在同一層的管線階段中，加密與解密的運算類似，所以可以將加密與解密的部分整合在一起，並且使用多工器來控制加密與解密的資料流向。如圖3.6所示，為三階段的回合管線架構，使用三組128位元的暫存器區隔管線階段，接著將詳細討論每個運算單元的設計。

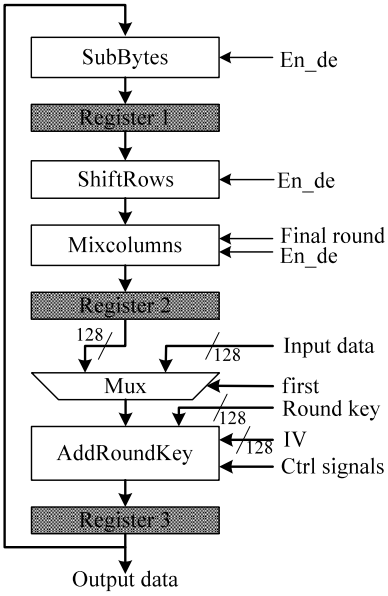

圖6　三階段的回合管線架構

3.2.1、位元組替代轉換單元

位元組替代轉換單元包含了加密時的位元組替代轉換與解密時的反位元組替代轉換，最後使用多工器選擇所需要的資料。

a. Subbytes

Subbytes區塊是代表加密時的位元組替代轉換，這部分是使用GF法則設計，每個位元組有其一對一且可逆的函數值，其運算結果可以儲存於ROM中。由於Rijndael演算法的資料運算寬度為128位元，因此需要16個ROM以同時替換16位元組。圖7顯示Subbytes區塊的運作方式。

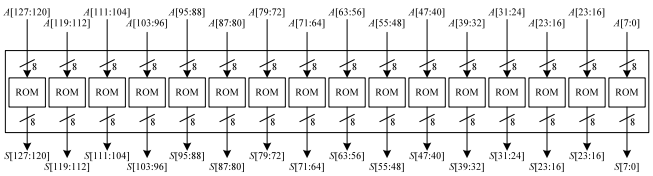

圖7　Subbytes區塊

b. InvSubbytes

InvSubbytes區塊使用於解密演算法，與Subbytes一樣，使用ROM儲存Inverse S-box的值，其結構與圖7相同，但是ROM儲存不同的值。因此，總共需要32個ROM，每個ROM的大小是256×8位元。

3.2.2、修正的位元組替代轉換單元

由於使用ROM完成加密與解密的位元組替代轉換，總共需要8 k位元組的ROM。雖然其架構簡單，但是需要相當大的面積，所以在有面積限制的情形下並不實際。若仔細觀察位元組替代轉換中的運算過程，可以很容易地發現，只要能針對乘法反元素的部分設計出一個有效率的硬體架構，就可以不必使用大量的ROM，也可以完成具有相同效果的位元組替代轉換。

a. 歐幾里得演算法

計算乘法反元素的方法有很多種如Fermat定理[4]、歐幾里得演算法[5] [6]等，其中歐幾里得演算法(Euclid's algorithm)除了可以快速的算出乘法反元素，對於硬體架構的實現也相當的容易。歐幾里得演算法的虛擬碼如下所示：

```
Euclid Algorithm for Inversion
{
  S = B;   T = M;   {deg M(x) > deg B(x)}
  U = 1;   V = 0;
  while(S≠1){
    Q = T div S
    Temp = T + Q · S;      T = S;      S = temp;
    Temp = V + Q · U;      V = U;      U = temp;
  }
  C = U;
}
```

其中，$B(x)$為欲轉換的多項式、$M(x)$為一個不可約的多項式、$C(x)$為所求出$B(x)$乘法反元素的結果。

由上述虛擬碼可知，歐幾里得演算法包含兩個部分，一個是有限場的除法器，另一個是有限場的乘法器

透過歐幾里得轉換器所得的結果即為乘法反元素，只需經過仿射運算後，就可得到位元組替代轉換的結果。歐幾里得轉換器所需的面積約比使用ROM的位元組替代轉換減少18.4%。由於此架構由於具有很高的規則性，可以很容易的擴充至有限場$GF(2^m)$，但是需要2(m-1)個時脈週期完成運算。在Rijndael演算法中，因為使用有限場$GF(2^8)$，所以需要14個時脈週期。總之歐幾里得轉換器雖然降低了所需之面積，但是顯著地增加了運算時間，大大的降低資料處理量，且不利於管線架構。對於要求高效能的運算，仍然不切實際。

b. 等效的LUT位元組替代轉換單元

由於加密與解密的運算都需要求出每個位元組乘法反元素的結果。利用此特性可以採用查表法(Look up table，簡稱LUT)，將歐幾里得轉換器的運算結果建立成一個新的乘法反元素對應表。

透過表1建立一個新的LUT，可以很容易的得到每個位元組的乘法反元素，只要加入位元組替代轉換中的仿射運算即可得到S-box、InvS-box值。面積比完全利用ROM完成位元組替代轉換約減少36.45%，且完成運算只需要3個時脈週期，因此不會影響管線架構。圖8為新的位元組替代轉換單元的硬體架構圖。

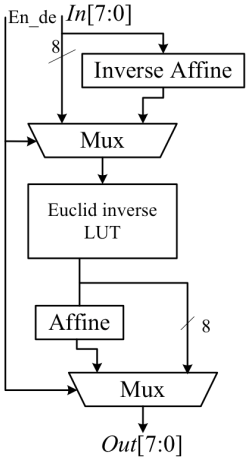

圖8　等效LUT之位元組替代轉換架構

表2為處理128位元的資料時，所使用的三種不同的位元組替代轉換方式，所需要的邏輯閘數量與所需要的時脈週期。

表2　128位元所需邏輯閘數與所需要的時脈週期比較

	ROM 32×256×8	Euclid's architerture	Eqv. LUT and Affine
Gate count	25,126	20,504	15,976
Area reduction	0%	18.4%	36.45%
Clock cycle	1	15	3

3.2.2、列位移轉換單元

a. Shiftrows

加密時的列位移轉換為將128位元的資料利用區塊演算法的特性將整個區塊的第2、3、4列做位元組的位移，如圖9所示。

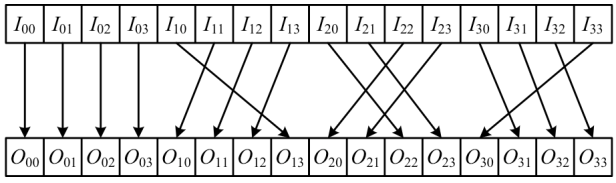

圖9　ShiftRows線路連結

b. InvShiftrows

解密的反位移轉換與ShiftRows的運作方式相反，因此可以回復原來的資料。在架構設計方面，也是直接以線路的連接方式來設計，只是與ShiftRows相反，如圖10所示。

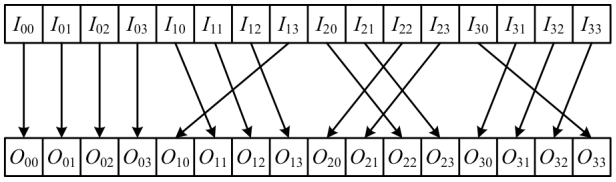

圖10　InvShiftRows線路連結

3.2.3、混合行轉換單元

a. MixColumns

MixColumns是屬於多項式矩陣乘法的運算，即狀態的每一行要乘上$c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$，我們使用xtime()的運算代替乘法，xtime()是向左位移的一個位元，如果發生溢位的情況，則需要與{1b}做XOR運算，如圖11所示。

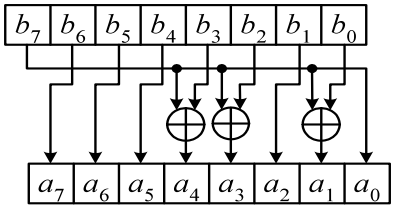

圖11　xtime硬體架構

再利用變數代換的原理，提出相同因數並讓4個式子能共用最多的因數，如此將MixColumn運算式簡化後，原式可轉換如下：

$$R_0' = \{02\}(R_0 \oplus R_1) \oplus \{01\}(R_1 \oplus R_2 \oplus R_3)$$

$$R_1' = \{02\}(R_1 \oplus R_2) \oplus \{01\}(R_0 \oplus R_2 \oplus R_3)$$

$$R_2' = \{02\}(R_2 \oplus R_3) \oplus \{01\}(R_0 \oplus R_1 \oplus R_3)$$

$$R_3' = \{02\}(R_0 \oplus R_3) \oplus \{01\}(R_0 \oplus R_1 \oplus R_2)$$

其硬體結構如圖12所示。

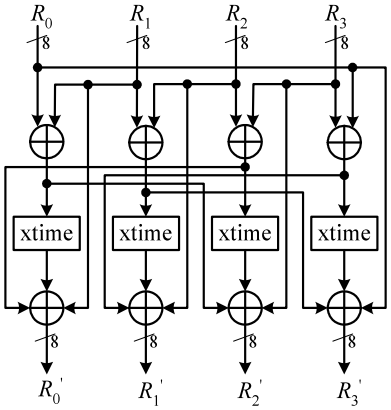

圖12　MixColumns硬體架構

b. InvMixColumns

InvMixColumns是Mixcolumns的反運算，即是乘上$c(x)$的反矩陣：$c^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$，即可以求出原來的值。其硬體架構與mixColumns類似，不再贅述。

3.2.4、加入回合金鑰轉換單元

在Rijndael演算法中，加入回合金鑰轉換是將目前128位元的狀態與128位元的回合金鑰做XOR運算。雖然加密與解密的金鑰使用順序不同，在本論文中金鑰是先在金鑰擴展程序單元中運算後，儲存於RAM裡，所以當運算至第n個回合時，只需要由記憶體中讀出金鑰即可。在加密時，由RAM讀出的回合金鑰直接傳送至加入回合金鑰轉換單元中；在解密時，從RAM中所讀出的回合金鑰需要再經過反混合行轉換，才傳送至加入回合金鑰轉換單元中。因此，加入回合金鑰轉換單元，只需要128個XOR邏輯閘即可完成，並且加密與解密是共用的。其硬體架構如圖13所示。

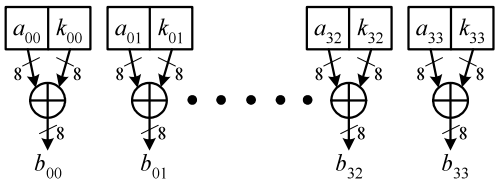

圖13　addroundkey硬體架構

3.2.5、擴展的加入回合金鑰轉換單元

由於本論文中所設計的AES架構亦包括CBC模式，所以需要將加入回合金鑰轉換做修正。如圖14所示，其中Data_in代表由外部所輸入的資料、Mixclm_in為混合行轉換單元輸出的結果、Bef_in_reg為前一筆資料加密後的結果、Bef_out_reg為前一筆資料解密的結果、Round_key為回合金鑰，配合由一些控制訊號、多工器與XOR閘，所構成具有CBC功能的架構，稱為擴展的加入回合金鑰轉換。

此架構的優點，在ECB模式下，不會影響與其它管線階段的運作；在CBC模式下，加密前與解密後不需要各耗費一個時脈週期，判斷是否需要與前一筆的加密或解密結果做XOR運算，就可以直接輸出。缺點是所需要的面積比原本的加入回合金鑰轉換的面積大。

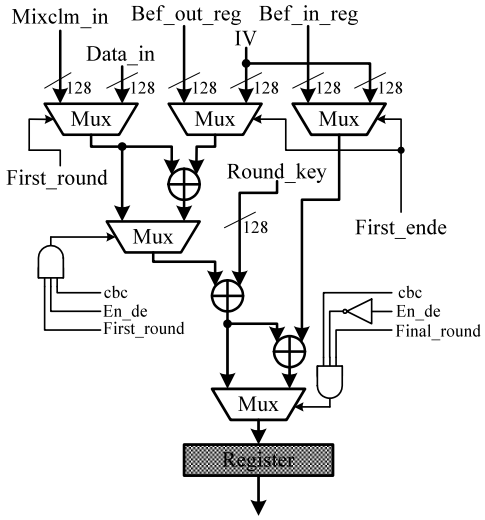

圖14　擴展的加入回合金鑰硬體架構

3.3、金鑰擴展程序單元

金鑰擴展程序單元是Rijndael演算法中另一個獨立的運算單元，其功能是產生加密或解密所需要回合金鑰，一般設計方法可分為即時運算設計與先行運算設計，這兩種設計方法最明顯的分別就是有無RAM的使用。下面將分析兩種設計方法的優缺點。

3.3.1、設計方法分析與選擇

即時運算設計

金鑰擴展程序所有的回合金鑰都可以利用即時運算(On-the-fly)的方式產生，當運算至某個回合時，才產生該回合所需要的回合金鑰。因此在晶片工作時，永遠只會有一組回合金鑰在金鑰擴展程序單元裡。所以在設計時，只需要使用一個長度為Nk個行字元的暫存器儲存當下的回合金鑰即可。其優點為減少設計面積，缺點則為較高的功率消耗，較多的I/O接腳或較低的資料處理量，及架構較複雜。

先行運算設計

本論文中所採用的方法就是先行運算(Pre-compution)的設計方式。先將所有的回合金鑰產生出來，連同主金鑰一起儲存在RAM裡，

等到要加密或解密時再適時的讀取RAM裡面所儲存的回合金鑰。先行運算設計與即時運算設計比較，除了面積較大外，其他考量都是優點多於缺點。優點為設計具彈性與較高的執行效率。缺點則為完整性較低

圖15顯示金鑰擴展程序的硬體架構。其中金鑰控制器(Key controller)中，用一個256位元的暫存器儲存主金鑰或金鑰擴展核心(Key Expansion Kernel)運算所產生的回合金鑰。

3.3.2、金鑰擴展核心運算

金鑰擴展核心運算負責產生回合金鑰。由於希望在14個時脈週期內，將所有的回合金鑰產生並存入RAM中。根據金鑰展開演算法，我們設計如圖16的硬體架構，其中*S*區塊是表示S-box替代轉換;*Rcon*是回合常數,儲存了1, 2, 4, 8, 16, 32, 64, 128, 27, 54等十個十進位數值，當回合金鑰的行字元為*Nk*的倍數時，控制器將依照倍數指定*Rcon*值；Bef_K_final是指上一組回合金鑰中為*Nk*倍數的行字元。

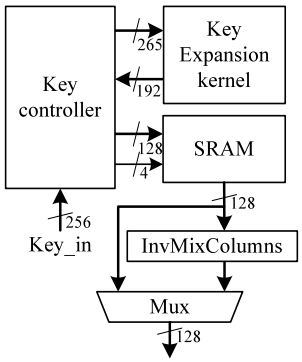

圖15　金鑰擴展程序硬體架構圖

當主金鑰長度為128位元時，依序輸出至*K*[0] ~ *K*[2]的行字元中，並且只接收*K*'[0] ~ *K*'[2]為正確的展開金鑰；當主金鑰長度為192位元時，則輸出至*K*[0]~*K*[5]，並且接收*K*'[0]~*K*'[5]為正確的展開金鑰；當主金鑰長度為256位元時，將金鑰分成兩個128位元的金鑰，依序輸出至*K*[0]~*K*[2]，只接收*K*'[0]~*K*'[2]，並利用多工器選擇Bef_K_final是否須做RotWord運算。

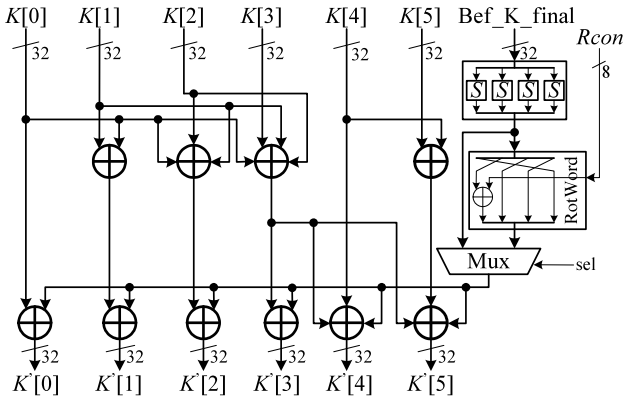

圖16　金鑰擴展核心運算架構圖

四、 晶片佈局與布局後模擬結果

圖17為完成後的晶片照片。晶片面積為2550.24 μm × 2550.24 μμm，核心面積(不含I/O pad與power ring)為1671.73 μm × 1671.73 μm，屬於Core-Limited設計。晶片包裝共計68腳，使用68-PIN CLCC包裝。其中，設置內部電源(VDD)4組供晶片本身(Core)使用、外部電源(extVDD)8組供I/O Pad使用。此外，晶片內部外加2組垂直方向的Power Stripe,使晶片內部供電效果穩定。

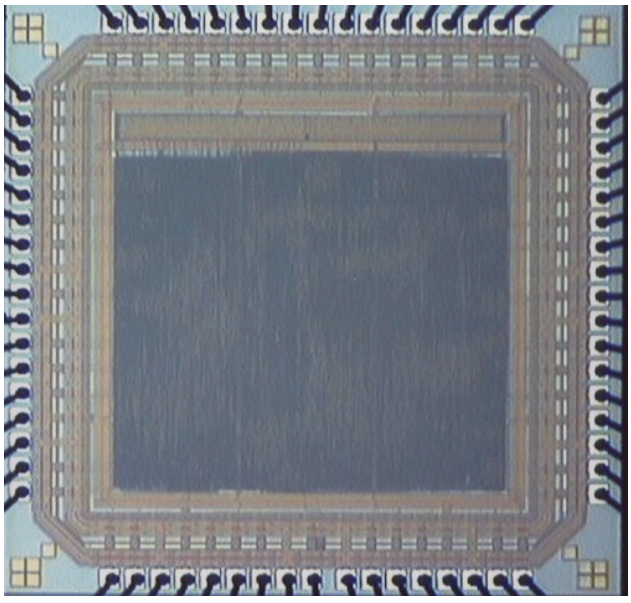

圖17　晶片照片

表3為最終的晶片規格。

表3　晶片規格

製程	TSMC 0.25 μm
包裝種類	68-PIN CLCC
晶片面積	2550.24 μm × 2550.24 μm
核心面積 (不含I/O pad和power ring)	1671.73 μm × 1671.73 μm
閘數量	76610
工作頻率	156.25 MHz
工作溫度範圍	0 □ ~ 125 □
功率消耗 (Typical Condition, load cap. = 10 pF)	128.4 mW

我們所設計的硬體架構含有管線效能。在ECB模式下，最高可同時處理三筆128位元的資料；在CBC、CFB與OFB模式下，管線架構失效，每次只能處理一筆128位元資料。每個回合固定需要五個時脈週期做運算，而每次加解密所需的回合數由輸入的金鑰長度來決定。 各模式下資料處理量如表4所示。

表4　各模式下資料處理量列表

Key Size	ECB Mode	CBC Mode	CFB Mode	OFB Mode
128-bit	1200 Mbps	400 Mbps	400 Mbps	400 Mbps
192-bit	1000 Mbps	333 Mbps	333 Mbps	333 Mbps
256-bit	857 Mbps	286 Mbps	286 Mbps	286 Mbps

表5列出其他相關論文的比較結果。

表5　相關論文比較

Reference	Key size	Max Freq. (MHz)	Gate Count	Throughput (Mbps)	Throughput/ Gate count	Operation Mode
This work	128/192/256	156.25	76,610 (87,846)	1200 (2000)	15.66 (22.76)	ECB/CBC /CFB/OFB
M. H. Lee[9]	128/192/256	125	120,000	1000	8.33	ECB
董力中[13]	128	N/A	51,194	1160	22.66	ECB
H. Kuo[10]	128/192/256	154	173,000	1600	9.25	ECB
NSA[11]	128/192/256	N/A	1029,046	443.2	0.431	ECB
T. Ichikawa[12]	128/192/256	N/A	612,834	1950	3.18	ECB

五、結論

本論文提出了一個具有管線架構的AES plus CBC加解密VLSI架構。利用模組化的觀念將加解密機制做在一起，並且利用可重複利用性的模組，減少不必要設計面積的浪費。依據AES演算法實現出來的加解密晶片輸入的每一筆資料為128位元，金鑰為可選擇的128位元、192位元以及256位元。在使用者介面的選擇性方面，所設計的晶片相容於微處理機控制介面，可操作於資料寬度為8位元、16位元以及32位元三種模式，並且內建了ECB、CBC、CFB以及OFB四種操作模式，供使用者彈性的搭配微處理器使用，依情況選擇適合的操作模式；在降低功率消耗方面，子金鑰是在加解密前做金鑰擴展程序產生，然後儲存於RAM裡面，若是需要加解密大量的資料，由於不需反覆計算金鑰展開部分，因此可以降低功率的消耗；在位元組替代轉換部分，利用歐幾里得演算法建立一個新的乘法反元素替換表，取代原本的S-box與InvS-box，面積約減少36.45%。

AES plus CBC加解密軟智產已經分別在Xilinx的Vertex 400型FPGA以及TSMC 0.25 μm元件庫(Cell Library)上實現與驗證。在FPGA設計部分，工作頻率為30 MHz，資料處理量最高為230.4 Mbps，消耗了Vertex 400型FPGA實驗板上73%的LUT與80%的Block RAM；在元件庫設計的部分，工作頻率為156.25 MHz，資料處理量最高為1200 Mbps，晶片面積為2550.24 μm × 2550.24 μm，核心(core)面積為1671.73 μm × 1671.73 μm，其等效閘數量(gate count)約為76610個，消耗功率為128.4 mW。

參考文獻

[1] AES home page, URL: http://www.nist.gov.aes/.

[2] J. Daemen and V. Rijnmen, *AES Proposal: Rijndael*, AES Algorithm Submission, September 3, 1999, available at [1].

[3] RSA Security, “RSA’s 56-bit DES

Challenge," April 2001, available at URL: http://www.rsasecurity.com/.

[4] C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsh, J. K. Omura, and I. I. Reed, "VLSI architecutres for computing mutiplications and inverters in *GF*(2^m)," *IEEE Transactions on Computers*, Vol. C-34, pp. 709-717, Aug. 1985.

[5] H. Brunner, A. Curiger, and M. Hofstetter, "On computing multiplicative inverses in GF(2^m)," *IEEE Transactions on Computers*, Vol. 42, No.8, August 1993, pp. 1010-1015.

[6] Y. T. Horng and S. W. Wei, "Fast Inverters and Dividers for Finite Field GF(2^m)," *Asia-Pacific Conference on Circuits and Systems*, pp. 206-211, 1994.

[7] W. Stallings, *Cryptography and Network Security: Principles and Practices Third Edition*, Prentice Hall International, Inc, 2003.

[8] K. Gaj and P. Chodowiec, "Comparison of the Hardware Performance of the AES Candidates using Reconfigurable Hardware," *The Third Advanced Encryption Standard (AES3) Candidate Conference*, New York, USA, April 2000.

[9] M. H. Lee, *A Gbps AES Cipher,* Master Thesis, Dept. of Computer Science, National Tsing-Hua University, Hsinchu, Taiwan, June 2001.

[10] 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*, Volume:38, pp. 569-572, Issue: 3, March 2003.

[11] B. Weeks, M. Bean, T. Rozylowicz, and C. Ficke, "*Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms*," National Security Agency while paper, May 15, 2000, available at URL: http://csrc.nist.gov/CryptoToolkit/aes/round2/NSA-AESfinalreport.pdf.

[12] T. Ichikawa, T. Kasuya, and M. Matsui, "Hardware Evaluation of the AES Finalist," *in Proc. 3rd AES Candidate Conference*, 2000.

[13] 董力中, *非同步AES加解密晶片之設計與實作*, 碩士論文–國立東華大學資訊工程學系研究所, 2002年7月。