基於 karatsuba 分解法實現低複雜度的多位元串列 GF(2^m)乘法器

李秋瑩
龍華科技大學
資訊網路工程系
pp010@gm.lhu.edu.tw

范家禎
交通大學
資訊科學與工程研究所
Wandy260178@yahoo.com.tw

葉為勳
龍華科技大學
資訊網路工程系
pp010@gm.lhu.edu.tw

摘要

近年來，由於無線通訊的普遍，資訊安全成為重要的研究議題，密碼系統也是其中重要的一環。本文提出一個新的低複雜度多位元串列 GF(2^m)乘法器，利用多位元串列的概念結合 Karatsuba 乘法器的原理，來降低電路使用的面積複雜度，並且可適用於橢圓曲線密碼學技術。我們所熟知的密碼系統核心運算電路是乘法器，然而密碼系統中乘法器的場非常大，所以設計一個降低空間與時間複雜度的乘法器是非常必要的。本文利用 Karatsuba 乘法器的原理，配合多位元串列的概念，利用 FPGA 來實現低複雜度乘法器，本方法僅需要 $\frac{3dm}{2}$ 個 AND 閘，$4m+n+\frac{3dm}{2}+\frac{m}{2}+d-5$ 個 XOR 閘以及 $3m-3$ 個暫存器，本文利用 Altera FPGA Quartus II 軟體環境，模擬四種不同位元數的乘法器，分別為 36×36，84×84，126×126 以及 204×204 個位元的乘法器，並實現於 Cyclone II EP2C70F896C8 之實驗平台上。由實驗結果可知，所提出的乘法器之時間複雜度皆小於文獻[13], [15]以及[19]，並且當乘法器相乘的位元數越多時，本架構可降低的時間×空間複雜度越大。

關鍵詞：Karatsuba、有限場、多位元串列。

壹、前言

在現今網路以及手機蓬勃的時代，資訊傳輸對我們的生活，變成不可或缺的一環，也正是因為如此，所以關於資訊安全必要性也變得非常的重要，在 1985 年時，由 Miler[1]和 Koblitz[2]首先引進密碼學並提出一個新的公開金鑰密碼系統，稱為橢圓曲線密碼系統(elliptic curve cryptosystem, ECC)，橢圓曲線密碼系統廣泛應用於有限場。一般 ECC 的運算是根據質數場 GF(p)或二位元場 GF(2^m)，其中 m 為有限場的大小，而乘法演算法是橢圓曲線密碼系統的核心運算，所以 m 也為乘法元素的位元數。橢圓曲線密碼系統之所以被重視的原因主要在於相同的安全強度下，橢圓曲線密碼系統的金鑰長度遠短於其它公開金鑰密碼系統的金鑰長度，他具有低功率，對儲存容量需求較小等優點，這特性非常適合用於手機；smart card 等等資源較小的地方，其中橢圓曲線密碼系統的核心，就是用乘法器所構成，乘法器設計的好壞，直接影響到橢圓曲線密碼實現的性能以及安全性。近年來許多的學者，致力於有限場乘法器的研究，文獻[3,4,5,6]中提到許多不同的GF(2^m)乘法器，包括位元串列、位元平行及多位元串列結構，位元平行乘法器通常採用最低有

效位元(LSB)或最高有效位元(LSB)的方式。

為了降低空間以及時間複雜度許多學者提出一些特別多項式的有限場乘法器，文獻[7, 8, 9]分別於全一多項式，五項多項式以及三項多項式，利用矩陣的方式發展位元平行乘法器。文獻[10]是利用最低有效位元(LSB)實現新的三項多項式乘法器，其時間複雜度為$2\sqrt{m}$個脈波週期，其中 m 為乘法元素的位元數，然而文獻[11]將[10]的乘法器做延伸發展出新的全一多項式乘法器。然而，另一些研究者提出低空間複雜度位元串列的乘法器[12]，他僅只需要 O(m)的空間複雜度，但是相對的時間複雜度也隨之拉長。為了將空間複雜度與時間複雜度，達到平衡，於是漸漸地發展出不同的多位元串列乘法器[13, 14]。傳統的多位元乘法器的延遲時間為 O($\frac{m}{d}$)個脈波週期，其中 d 為所選的多位元的位元數大小。文獻[15]提出具有位元進位元結構之多位元串列乘法器。

Karatsuba-Ofman's 演算法(KOA) [16] 發表於 1962 年，是第一個打破位置編排的整數乘法運算，由於計算方式簡單，所以其多項式版本廣泛應用在GF(2^m)中的 VLSI 乘法器密碼系統。有限場以及 KOM 的相關論文相繼被提出，C. Grabbe [17]，設計一個 240 位元的乘法器實現在GF(2^{233})上。文獻 [18]將 Karatsuba 演算法的概念，用於有限場$GF(2^m)$上，提出一個低複雜度的乘法器。

本篇論文的其他章節說明如下：第貳章為數學基礎，說明本篇論文使用的數學概念。第參章為所提出的低複雜度多位元串列乘法器，包含本文的演算法及架構。第肆章為實驗結果，並與其他論文比較其效能與時間空間複雜度。最後，第伍章為本論文的結論。

貳、相關文獻

2.1 有限場表示法

在這個部分簡短的介紹有限場的表示方法，有限場$GF(2^m)$中有2^m個元素，每個元素可表示成m維度的向量，而每一個向量皆由GF(2)所定義出來，即$GF(2^m) = \{A = a_0 + a_1x + a_2x^2 + \cdots + a_{m-1}x^{m-1}\}$，其中$a_i = GF(2)$。而$m$必定為正整數，在有限場中有許多基底表示法，其中最普遍的三大基底：多項式基底（polynomial basis, PB），正規基底(normal basis, NB)以及雙重基底(dual basis, DB)。

有限場的每一個元素也可以表示成$A = a_0 + a_1x + a_2x^2 + \cdots + a_{m-1}x^{m-1}$其中，$N = \{a_0, a_1, a_2, \ldots, a_{m-1}\}$這個稱為有限場的基底表示法。例如：$a_i = x^i$，$A = \{1, x^1, x^2, \ldots, x^{m-1}\}$是稱為多項式基底（polynomial basis, PB），其中，x是為$F(x) = f_0 + f_1x + f_2x^2 + \cdots + f_{m-1}x^{m-1} + x^m = 1 + \sum_{i=1}^{m-1} x^i + x^m$的根，也就是說$F(x) = 0$,

$$x^m = f_0 + f_1x + f_2x^2 + \cdots + f_{m-1}x^{m-1} \quad (1)$$

我們可利用方程式(1)，做乘法運算的降階之用。

為了方便說明乘法原理，我們採用多項式基底來說明乘法器的架構，假設有限場GF(2^m)是由不可分解多項式$F(x) = f_0 + f_1x + f_2x^2 + \cdots + f_{m-1}x^{m-1} + x^m = 1 +$

$\sum_{i=1}^{m-1} x^i + x^m$ 所產生的，且 $F(x)$ 的根為 α。令元素 $A = (a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{m-1}\alpha^{m-1}) = \sum_{i=0}^{m-1} a_i\alpha^i$ ， $B = (b_0 + b_1\alpha + b_2\alpha^2 + \cdots + b_{m-1}\alpha^{m-1}) = \sum_{i=0}^{m-1} b_i\alpha^i$ ，$C = (c_0 + c_1\alpha + c_2\alpha^2 + \cdots + c_{m-1}\alpha^{m-1}) = \sum_{i=0}^{m-1} c_i\alpha^i$ 為有限場GF(2^m)中的任意函數，其中C為元素A及元素B的乘積，也就是說，元素C可表示成以下：

$$C = AB = (a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{m-1}\alpha^{m-1})B$$
$$= a_0B + a_1B\alpha + a_2B\alpha^2 + \cdots + a_{m-1}B\alpha^{m-1} \quad (2)$$
$$= (\ldots((a_{m-1}B)\alpha + a_{m-1}B)\alpha + \cdots a_1B)\alpha + a_0B \quad (3)$$

根據方程式(2)，這種的乘法結構我們稱為LSB (Least Significant Bit)乘法演算法；方程式(3)稱為MSB(Most Significant Bit)乘法演算法。以下我們根據這兩種的乘法演算法來說明電路架構。

2.2 *GF*(2^m)的傳統 digit-serial 乘法器

在這個部分簡短的介紹多位元串列演算法。令 GF(2^m)是長度為 m 的 $F(x)$不可分解多項式所組成的，其中元素 A 及 B 為 GF(2^m)當中的兩個元素：

$$A = a_0 + a_1x + \cdots + a_{m-1}x^{m-1}$$
$$B = b_0 + b_1x + \cdots + b_{m-1}x^{m-1}$$

其中 a_i 和 $b_i \in \{0,1\}$。然後有限場的元素 A 和 B 需要 mod $F(x)$

$$C=AB \bmod F(x) \quad (4)$$

為了實現方程式(4)，有許多不同的模組，可以在有限制的環境下，實現硬體需求。以下的圖 1，我們使用 Least Significant Digit (LSD)乘法器描述多位元串列乘法器結構。

Digit-serial 乘法器將 m 個位元的乘法元素 A，B 分別切成許多相同長度的位元做運算，進而得到運算結果。假設 m 為乘法器的運算位元數，d 為所選的切割大小，我們將獲得 k 筆 d 個位元的資料，其中 $k=\left\lceil \frac{m}{d} \right\rceil$，如 m 不是 dk 的倍數，就在元素的最高位元補 0。

$$A = (a_0, a_1 \ldots, a_{m-1}, 0 \ldots, 0)$$

其中 0 為 kd-m 個 bit，元素 A 可表示成：

$$A = \sum_{i=0}^{k-1} A_i x^{id}$$

其中$A_i = a_{id} + a_{id+1} + \cdots + a_{id+d}x^{d-1}$

使用 LSD 乘法模組可得到 C 如下：

$$C=AB \bmod F(x)$$
$$= \left(A_0 + A_1x^d + \cdots + A_{k-1}x^{(k-1)d}\right)B \bmod F(x)$$
$$= C_0 + C_1 + \cdots + C_{k-1} \bmod F(x) \quad (5)$$

其中

$$C_i = A_iB^{(i)} \quad (6)$$
$$B^{(i)} = Bx^{di} \bmod F(x) = x^dB^{(i-1)} \bmod F(x) \quad (7)$$

其中 $0 \le i \le k$-1

LSD 多位元串列乘法演算法
輸入：A 和 B 為 GF(2^m)中的元素
輸出：$C=AB \bmod F(x)$
1. $\overline{C}=0$;
2. $A=A_0+A_1x^d+\ldots+A_{k-1}x^{(k-1)d}$
3. *for* $i=0$ *to* k-1
4. $\overline{C}=\overline{C}+A_iB$;
5. $B=Bx^i \bmod F(x)$;
6. *end for*
7. $C=\overline{C} \bmod F(x)$;

圖1 LSD 多位元串列乘法演算法

根據圖 1 的架構可獲得如下圖 2 之多位元串列乘法器，此架構包含乘法核心電路，兩個暫存器，一個降低多項式($\overline{C} \bmod F(x)$)，及一個($m+d$)個 bit 的加法器。而圖 1 的乘法核心，是步驟 4 的 A_iB 運算，初始步驟是將元素 B 的暫存器設為元素 B 的初始值，而 $\overline{C}$ 的暫存器設為 0。根據圖 1 的步驟 3 到步驟 6，k 個脈波週期後，暫存器 $\overline{C}$ 可獲得 $\overline{C}_0+\overline{C}_1+\cdots+\overline{C}_{k-1}$。而且下一個脈波週期可實現完整的降階多項式，其運算為 $C=\overline{C} \bmod F(x)$，並且獲得最後的乘法運算。根據圖 2 之傳統多位元串列乘法器需要$\left\lceil\frac{m}{d}\right\rceil$+1 個 clock。

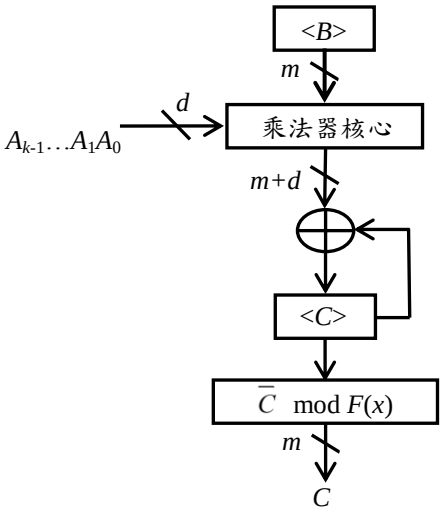

圖2 傳統的多位元串列乘法器

2.3 Karatsuba 乘法的基本原理

在這個部分，我們介紹將 Karatsuba 乘法器，此乘法器可將乘法元素，切割成許多筆相同位元的資料，假設乘法元素 A，B 為 m 個位元，利用 Karatsub 的方式切割成 n 段，利用分割法運算會產生 $n(n+1)/2$ 個子乘法器，如果所要運算的乘法器元素 A 及元素 B 為 m 個位元，利用 Karatsuba 乘法器將元素 A 及元素 B 切割成 n 段，那麼每一段就會有$\frac{m}{n}$個位元，將切割好的元素 A 及元素 B 相乘，那麼每一個子乘法器皆為$\frac{m}{n}\times\frac{m}{n}$個位元。

下面我們將介紹切割成兩段的 Karatsuba 乘法器，應用到多位元串列結構的多項式乘法器上[16]，詳細說明如下：

元素 $A=a_0+a_1x+a_2x^2+\cdots+a_{m-1}x^{m-2}+a_{m-1}x^{m-1}$ 及元素 $B=b_0+b_1x+b_2x^2+\cdots+b_{m-1}x^{m-2}+b_{m-1}x^{m-1}$ 為 GF(2^m)內的元素，它是由不可分解多項式所組合而成的，且元素 A 及 B 為兩個長度為 m 的多項式，而元素 C 為元素 AB 的乘積，我們可將元素 A 及元素 B 分成兩個部份如下：

$$A=\sum_{i=0}^{m-1}a_ix^i=\sum_{i=\frac{m}{2}}^{m-1}a_ix^i+\sum_{i=0}^{\frac{m}{2}-1}a_ix^i$$
$$=x^{\frac{m}{2}}\sum_{i=0}^{\frac{m}{2}-1}a_{i+\frac{m}{2}}x^i+\sum_{i=0}^{\frac{m}{2}-1}a_ix^i=x^{\frac{m}{2}}A_1+A_0 \quad (8)$$

$$B=\sum_{i=0}^{m-1}b_ix^i=\sum_{i=0}^{m-1}b_ix^i+\sum_{i=0}^{\frac{m}{2}-1}b_ix^i$$
$$=x^{\frac{m}{2}}\sum_{i=0}^{\frac{m}{2}-1}b_{i+\frac{m}{2}}x^i+\sum_{i=0}^{\frac{m}{2}-1}b_ix^i=x^{\frac{m}{2}}B_1+B_0 \quad (9)$$

其中元素 A_0、A_1、B_0 及 B_1 長度皆為 $\frac{m}{2}$ 位元的多項式，元素 A 和 B 的乘積 C 表示如下：

$$C=AB=A_0B_0+(A_1B_0+A_0B_1)\,x^{\frac{m}{2}}+A_1B_1x^m \quad (10)$$

為了改善乘積 C 的計算方法，我們可以將方程式(10)，也可表示成：

$$C=AB=A_0B_0+[(A_0B_0+A_1B_1)+(A_0+A_1)(B_0+B_1)]\,x^{\frac{m}{2}}+A_1B_1x^m \quad (11)$$

我們發現可由三種乘法分別為 A_0B_0，$(A_0+A_1)(B_0+B_1)$以及 A_1B_1 獲得方程式(11)乘法運算結果。為了實現方程式(11)，可利用以下三個步驟實現乘法乘積：

步驟一: 評估運算點(KO-EP)：根據以上的三種乘法可評估出元素 A、B 的三個運算點為：

$$\text{KO-EP}(A)=(A_0,\ A_0+A_1,\ A_1) \quad (12)$$
$$\text{KO-EP}(B)=(B_0,\ B_0+B_1,\ B_1) \quad (13)$$

步驟二: 點對點的運算(KO-PWM)：利用以上評估點將他們做相乘，得到以下方程式：

$$\text{KO-PWM}(C)=\text{KO-EP}(A)\times\text{KO-EP}(B)$$

$$=(A_0B_0,(A_0+A_1)(B_0+B_1),A_1B_1)=(t_0,t_1,t_2) \quad (14)$$

步驟三: 重建模組(KO-R)：最後，依據乘法點的運算結果，將 t_0,t_1 及 t_2 做還原排列，即可獲得原始乘法的最後結果如下：

$$\text{KO-R}(C)=(\text{KO-PWM}(C)=(C_0,C_1,C_2)$$
$$=(A_0B_0, A_0B_0+(A_0+A_1)(B_0+B_1)+A_1B_1, A_1B_1) \quad (15)$$

根據以上三個步驟，可以畫出以下架構圖 3，步驟一評估運算點，可將乘法元素 A 及元素 B 分成相同的兩段，分別為 A_0，A_1 及 B_0，B_1，並且利用 XOR 閘分別將 A_0、A_1 及 B_0、B_1 加起來組成(A_0+A_1)及(B_0+B_1)，接著步驟二將獲得的三個運算點，送入三個乘法器中進行點對點的運算，獲得乘法結果t_0,t_1,t_2，最後，步驟三將乘法結果t_0,t_1,t_2，利用 XOR 閘進行 Karatsuba 乘法的重建模組，並且送出傳統乘法運算結果。

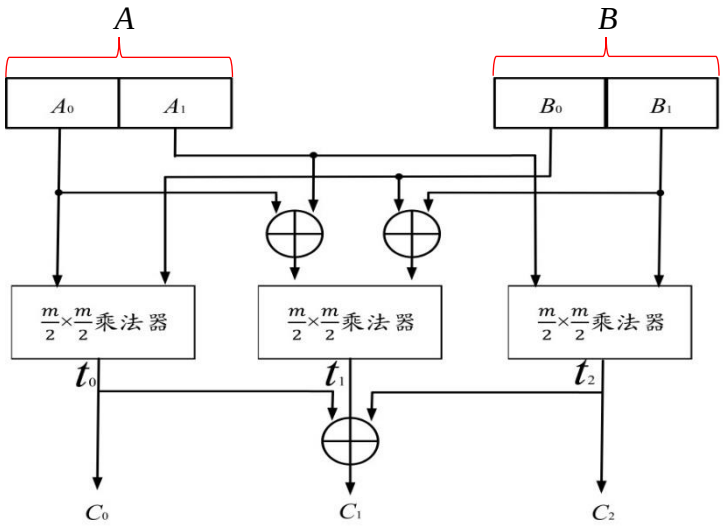

圖3 傳統的 Karatsuba 乘法架構

依據以上方法可實現遞迴演算法，將所有的元素分段成極小的位元數，若以兩段分法，來延伸乘法結構，Karatsuba 三大步驟的每一個步驟複雜度可分別如下，若 m 為元素的長度，且 $n=2^i$ 其中 i 為遞迴次數，步驟一需要 m 個 XOR 閘，步驟二需要 $(\frac{m}{n})^2\times 3^i$ 個 AND 閘以及 $(\frac{m}{n}\text{-}1)^2\times 3^i$ 個 XOR 閘，步驟三需要 $(\frac{2m}{n}\text{-}1)\times(3^i\text{-}1)$ 個 XOR 閘，則每個模組可獲得表 1 之複雜度分析結果：

表1 Karatsubau 每個模組電路複雜度

元件 / 步驟	AND 閘	XOR 閘
KO-EP	0	m
KO-PWM	$(\frac{m}{n})^2\times 3^i$	$(\frac{m}{n}-1)^2\times 3^i$
KO-R	0	$\left(\frac{2m}{n}-1\right)\times(3^i-1)$

參、低複雜度多位元串列 GF(2^m)乘法器

本文所提出的乘法器結合 Karatsuba 乘法器與 digit-serial 乘法器的概念，利用 Karatsuba 乘法器切割方法將 m 個位元的元素 A,B，切割成 n 段，每一筆元素 A,B 將有$\frac{m}{n}$個位元，接著將切割好的元素 A，利用 digit-serial 乘法器概念再次做切割，然而每一段的位元數為 d 個位元，因此每一個子乘法器皆為$d \times \frac{m}{n}$個位元，本文與 Karatsuba 乘法器切割的方式不同的地方在於，單純利用 Karatsuba 乘法器切割，切割越多段會產生越多的子乘法器，然而利用本文所提出的方法不會產生多的子乘法器。所提出的乘法器，針對切割成兩段的 Karatsuba 演算法，推導新的多位元串列乘法。假設有限場 GF(2^m)是由不可分解多項式 $F(x)$所建構的，多項式 A 與 B 為有限場 GF(2^m)之兩元素，且分別表示如下：

$$A=A_L+A_H x^{\frac{m}{2}}$$

$$B=B_L+B_H x^{\frac{m}{2}}$$

其中

$$A_L = a_0 + a_1 x + \cdots + a_{\frac{m}{2}-1} x^{\frac{m}{2}-1}$$

$$A_H = a_{\frac{m}{2}} + a_{\frac{m}{2}+1} x + \cdots + a_{m-1} x^{\frac{m}{2}-1}$$

$$B_L = b_0 + b_1 x + \cdots + b_{\frac{m}{2}-1} x^{\frac{m}{2}-1}$$

$$B_H = b_{\frac{m}{2}} + b_{\frac{m}{2}+1} x + \cdots + b_{m-1} x^{\frac{m}{2}-1}$$

A 與 B 之乘積則可表示成

$$C=(A_L+A_H x^{\frac{m}{2}})\,(B_L+B_H x^{\frac{m}{2}}) \bmod F(x)$$

$$= A_L B_L(1+x^{\frac{m}{2}})+A_H B_H(x^{\frac{m}{2}}+x^m)+(A_L+A_H)(B_L+B_H)x^{\frac{m}{2}} \bmod F(x) \qquad (16)$$

根據方程式(16)，乘法包含三個子乘法即 $A_L B_L$、B_L+B_H、$(A_L+A_H)(B_L+B_H)$。令子多項式表示為

$$A_i = \begin{cases} A_L & for \quad i=0 \\ A_H & for \quad i=1 \\ A_L+A_H & for \quad i=0+1 \end{cases}$$

$$B_i = \begin{cases} B_L & for \quad i = 0 \\ B_H & for \quad i = 1 \\ B_L + B_H & for \quad i = 0 + 1 \end{cases}$$

那麼，乘積 C 可表示成

$$C=C_0(1+x^{\frac{m}{2}})+C_1(x^{\frac{m}{2}}+x^m)+C_2 x^{\frac{m}{2}} \bmod F(x) \tag{17}$$

其中

$$C_i=A_iB_i$$

假設 d 為選取的區段長度，每個子多項式 A_i 可表示成

$$A_i=a_{i,0}+a_{i,1}x+\cdots+a_{i,\frac{m}{2}-1}x^{\frac{m}{2}-1}=\sum_{j=0}^{k-1}A_{i,j}x^{jd} \tag{18}$$

其中

$$A_{i,j}=\sum_{l=0}^{d-1}a_{i,j}x^l, k=\left\lceil\frac{m}{2d}\right\rceil$$

利用方程式(18)，子乘積 $C_i=A_iB_i$ 可表示成

$$C_i=A_iB_i=\sum_{j=0}^{k-1}A_{i,j}B_ix^{jd}=(((A_{i,k-1}B_i)x^d+A_{i,k-2}B_i)x^d+\cdots)x^d+A_{i,0}B_i \tag{19}$$

根據方程式(17)-(19)，提出的乘法演算法如下圖 4 所示:

Inputs：$A=A_L+A_H x^{\frac{m}{2}}$ and $B=B_L+B_H x^{\frac{m}{2}}$ are two elements in GF(2^m).

Output：$C=AB$ mod $F(x)$.

1. $C_0=0$, $C_1=0$, $C_2=0$.
2. $A_0= A_L$, $A_1= A_H$, $B_0= B_L$, $B_1= B_H$.
3. For $j=k$-1 to 0

*/initialization step

4. $A_{0+1,j}= A_{0,j}$, + $A_{1,j}$, where $A_{i,j}=(a_{i,dj}, a_{i,dj+1},\cdots, a_{i,dj+d-1})$.
5. $B_{0+1}= B_0+ B_1$.

*/subword product computation step

6. $AB_0= A_{0,j}\ B_0$.
7. $AB_1= A_{1,j}\ B_1$.
8. $AB_{0+1}= A_{0+1,j}\ B_{0+1}$.
9. $C_0= C_0x^{kd}+ AB_0$.
10. $C_1= C_1x^{kd}+ AB_1$.
11. $C_2= C_2x^{kd}+ AB_{0+1}$.
12. endfor

*/ finial polynomial reduction step

13. C=$C_0\left(1+x^{\frac{m}{2}}\right)+C_1\left(x^{\frac{m}{2}}+x^m\right)+C_2x^{\frac{m}{2}}$ mod F(x)

圖4 提出的多位元 Karatsuba 乘法演算法

根據圖 4 的結構，可畫出所提出的乘法架構如圖 5 所示，這個架構包含三個乘法核心電路，三個暫存器，一個降階多項式，三個(m+d)個位元的加法器，三個移位電路以及一個由 XOR 閘組成的還原模組。Karatsuba 是將具有 m 個位元的乘法元素 A 及 B 分別切割成$\frac{m}{2}$個位元，分別為A_0，A_1，B_0以及B_1，接著令元素A_0+A_1為A_{0+1}，B_0+B_1為B_{0+1}，我們將元素A_0與B_0、A_1與B_1以及A_{0+1}與B_{0+1}相乘，然後再利用 digit-serial 的概念，將元素A_0，A_1，A_{0+1}切成多筆資料，其中每一筆資料為 d 個位元，因此元素A_0，A_1及A_{0+1}將被切割為 k 筆資料，其中 k=$\left\lceil\frac{m}{2d}\right\rceil$，如果當 k 值越大時，所消耗需要的乘法空間複雜度將會越小，但所需的時間複雜度將會增加。根據圖 4 的步驟 1 先將三個暫存器歸零，接著利用步驟 2 將元素分成 A_0，A_1 及 B_0，B_1，然而圖 5 的元件 Convert 1 及 Convert 2 所做的運算即為

圖 4 的步驟 4 以及步驟 5，步驟 4 將元素 $A_{0,j}$ 及 $A_{1,j}$ 相加獲得 $A_{0+1,j}$，而步驟 5 將元素 B_0 及 B_1 相加獲得 B_{0+1}，接著將元素 $A_{0,j}$ 與 B_0 相乘送入步驟 6 運算，而乘法結果為圖 5 的乘法器 AB_0，元素 $A_{1,j}$ 與 B_1 相乘送入步驟 7 運算，而乘法結果為圖 5 的乘法器 AB_1，接著步驟 8 將元素 $A_{0+1,j}$ 與 B_{0+1} 相乘，而乘法結果為圖 5 的乘法器 AB_{0+1}，接著圖 4 的步驟 9-步驟 11 就是將圖 5 的三個暫存器<C_0>，<C_1>及< C_2>做移位動作並與乘法器 AB_0，AB_1 及 AB_{0+1} 所獲得的資料相加，然後存回暫存器<C_0>，<C_1>及<C_2>，根據圖 4 所示，在 j 個脈波週期後，暫存器<C_0>，<C_1>及<C_2>可獲得其運算結果，其中$0 \le j \le k-1$。接著依據圖 4 的步驟 13 就是將運算好的 C_0、C_1 及 C_2 分配至對應的位置即為圖 5 的還原模組。最後，於下一個脈波週期可實現完整的降階多項式，其運算為 $C=C \bmod F(x)$，並且獲得最後的乘法運算。根據圖 5 所示，所提出的乘法器需要三個子乘法器分別為 AB_0，AB_1 及 AB_{0+1}，每個子乘法器皆有兩筆輸入， AB_0 乘法器的輸入為元素 $A_{0,j}$ 及元素 B_0，乘法器 AB_1 的輸入為元素 $A_{1,j}$ 及元素 B_1 以及乘法器 AB_{0+1} 的輸入為元素 A_{0+1} 及 B_{0+1}，然而元素 $A_{0,j}$，$A_{1,j}$ 以及 $A_{0+1,j}$ 為 d 個位元，B_0，B_1 及 B_{0+1} 為 $\frac{m}{2}$ 個位元，故所獲得的乘積 AB_0，AB_1 及 AB_{0+1} 為$(\frac{m}{2} \times d)$-1 個位元。根據圖 4 的步驟 13 可畫出圖 6 的還原模組處理單元。

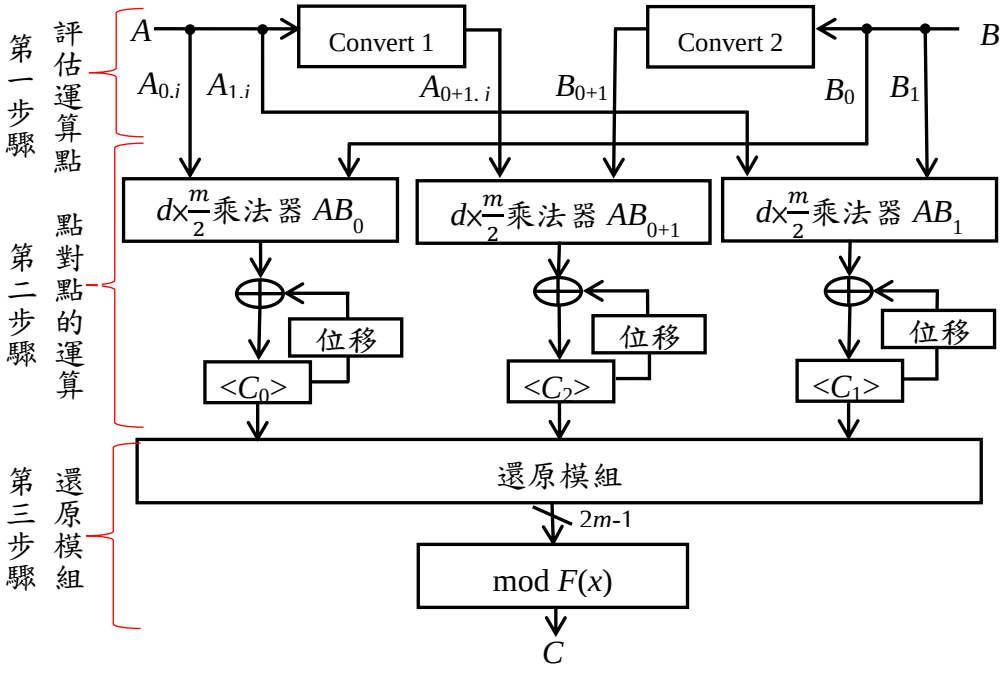

圖5 所提出的乘法器架構

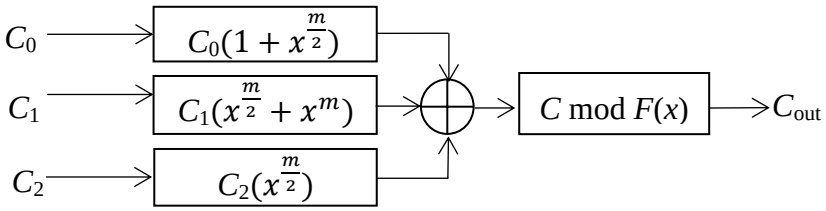

圖6 還原模組處理單元

為了更清楚的了解，我們舉一個由 $F(x)=x^{12}+x^3+1$ 產生的有限場來說明所提出的乘法器。令 $A=\sum_{i=0}^{11}a_i x^i$ 及 $B=\sum_{i=0}^{11}b_i x^i$ 為有限場 GF(2^{12})上的兩個元素，然而 GF(2^{12})為 12×12 個位元的乘法器，C 為元素 A、B 的乘積，其中 d=2，利用圖 4 的步驟 2，可獲得如下方程式：

$$A=A_0+A_1x^6$$

$$B=B_0+B_1x^6$$

利用圖 4 的步驟 2 及步驟 4，可獲得 A_0，A_1 及 A_{0+1} 如下：

$$A_0=A_{0,0}+A_{0,1}x^2+A_{0,2}x^4=(a_0+a_1x)+(a_2+a_3x)x^2+(a_4+a_5x)x^4$$

$$A_1=A_{1,0}+A_{1,1}x^2+A_{1,2}x^4=(a_6+a_7x)+(a_8+a_9x)x^2+(a_{10}+a_{11}x)x^4$$

$$A_{0+1}=A_{0+1,0}+A_{0+1,1}x^2+A_{0+1,2}x^4=(A_{0,0}+A_{1,0})+(A_{0,1}+A_{1,1})x^2+(A_{0,2}+A_{1,2})x^4$$

利用圖 4 的步驟 6-步驟 11，可獲得 C_0，C_1 及 C_2 如下：

$$C_0：(A_{0,0}B_0)+(A_{0,1}B_0)x^2+(A_{0,2}B_0)x^4$$

$$C_1：(A_{1,0}B_1x^6)+(A_{1,1}x^6B_1x^6)x^2+(A_{1,2}x^6B_1x^6)x^4$$

$$C_2：[A_{0+1,0}(B_0+B_1x^6)]+[A_{0+1,1}(B_0+B_1x^6)]x^2+[A_{0+1,2}(B_0+B_1x^6)]x^4$$

由於這個例子，由 $F(x)=x^{12}+x^3+1$ 所生成，所以利用圖 4 的步驟 13 將高於 x^{12} 全部做 mod $F(x)$的動作，因此 $x^{12}=x^3+1$，$x^{13}=x^4+x$ 以此類推，進而獲得以下結果：

$c_0=c_0+c_{12}+c_{21}$	$c_1=c_1+c_{13}+c_{22}$	$c_2=c_2+c_{14}$	$c_3=c_3+c_{12}+c_{15}+c_{21}$
$c_4=c_4+c_{13}+c_{16}+c_{22}$	$c_5=c_5+c_{14}+c_{17}$	$c_6=c_6+c_{15}+c_{18}$	$c_7=c_7+c_{16}+c_{19}$
$c_8=c_8+c_{17}+c_{20}$	$c_9=c_9+c_{18}$	$c_{10}=c_{10}+c_{19}$	$c_{11}=c_{11}+c_{20}$

肆、效能分析及實驗結果

4.1 效能分析

在這一節，我們分析所提出的乘法器之硬體時間與空間複雜度。在第三章中，利用多位元串列的架構結合 Karatsuba 乘法概念，提出了低複雜度的多位元乘法器。根據圖 5 所提出的多位元串列乘法器架構，時間以及空間複雜度分析如下，主要分為三個部份：

(1) 評估運算點

第(1)步驟評估運算點的空間複雜度為 Convert 1 及 Convert 2 分別運算圖 4 的步驟 4 及步驟 5，Convert 1 的運算是將元素 A_0 與 A_1 相加，其中 $0 \le j \le d\text{-}1$，而 Convert 2 的運算是將元素 B_0 與 B_1 相加，然而元素 B_0 與 B_1 根據圖 4 的步驟 2 可知 $B_0=B_L$, $B_1=B_H$，從圖 4 的 input 可知 B_L 與 B_H 為兩個 $\frac{m}{2}$ 位元的元素，故 Convert 1 及 Convert 2 時間以及空間複雜度如下：

- 元件 Convert 1 需要 d 個 XOR 閘，以及一個 XOR 閘的延遲時間。
- 元件 Convert 2 需要 $\frac{m}{2}$ 個 XOR 閘，以及一個 XOR 閘的延遲時間。

(2) 點對點的運算

第(2)步驟點對點的運算包含三個乘法器 AB_0，AB_1 及 AB_{0+1}、數個 XOR 閘以及三個儲存乘法器結果的暫存器$<C_0>$，$<C_1>$及$<C_2>$，根據圖 4 所示，乘法器 AB_0，AB_1 及 AB_{0+1} 為圖 4 的步驟 6-步驟 8，此三個乘法器分別運算 $A_{0,j}B_0$，$A_{1,j}B_1$ 以及 $A_{0+1,j}B_{0+1}$，然而 B_0，B_1 及 B_{0+1} 為三筆 $\frac{m}{2}$ 個位元的資料，$A_{0,j}$，$A_{1,j}$ 以及 $A_{0+1,j}$ 為三筆 d 個位元的資料，故乘法器 AB_0，AB_1 及 AB_{0+1} 為三筆 $\frac{m}{2} \times d$ 個位元的乘法器。根據圖 5 乘法器 AB_0，AB_1 及 AB_{0+1} 運算好的乘法結果將與暫存器$<C_0>$，$<C_1>$以及$<C_2>$相加，並且將相加完畢的數值儲存回暫存器$<C_0>$，$<C_1>$以及$<C_2>$，其中每一個暫存的位元數為 m-1 個位元，所以三筆暫存器一共需要 3(m-1)個位元的暫存器。根據圖 4 的步驟 9-步驟 11 這三個乘法器的運算結果，需要 3($\frac{m}{2}$ +d-1)個 XOR 閘，因此第(2)步驟點對點的運算共需要 3($d\times\frac{m}{2}$)個 AND 閘，3($d\times\frac{m}{2}$)個 XOR 閘，及 3(m-1)個位元的暫存器。所以點對點的運算時間以及空間複雜度如下：

- 三個核心乘法器需要 3($d\times\frac{m}{2}$)個 AND 閘、3($\frac{m}{2}$ -1)(d-1)個 XOR 以及 d 個 AND 閘+(d-1)個 XOR 閘的延遲時間。
- 與核心乘法器相加的 XOR 閘需要 3($\frac{m}{2}$ +d-1)個 XOR 閘以及 1 個 XOR 閘的延遲時間。
- 暫存器$<C_i>$，其中 $0 \le i \le 2$，需要 3(m-1)個位元的暫存器以及 1 個暫存器的延遲時間。

(3) 還原模組

最後還原模組需要將暫存器<C_i >的值相加，其中 0≤ i ≤2，而且暫存器<C_i>為(m-1)個位元以及 mod 模組在 $F(x)=x^m+x^n+1$，要將大於 m 的位元做 mod 的動作所以還原模組的運算時間以及空間複雜度如下：

- 還原模組需要 2(m-1)個 XOR 閘以及兩個 XOR 閘的延遲時間。
- mod $F(x)$需要(2m+n-3)個 XOR 閘以及 3 個 XOR 閘的延遲時間。

根據以上所描述的我們可以統整出所提出的多位元串列乘法器各個元件的延遲時間以及空間複雜度如下表 2：

表2 GF(2^m)中提出的乘法器各個元件複雜度及延遲時間分析

元件	XOR	AND	暫存器	延遲時間
Convert 1、2	$d+\frac{m}{2}$	0	0	1 XOR
乘法器 AB_i	$3(\frac{m}{2}-1)(d-1)$	$3(d\times\frac{m}{2})$	0	d AND+(d-1) XOR
XOR	$3(\frac{m}{2}+d-1)$	0	0	1 XOR
<C_i>	0	0	3(m-1)	1 暫存器
還原模組	2(m-1)	0	0	2 XOR
mod $F(x)$	(2m+n-3)	0	0	3 XOR
合計	$4m+n+\frac{3dm}{2}+\frac{m}{2}+d-5$	$3(d\times\frac{m}{2})$	3(m-1)	d AND (d+6) XOR 1 暫存器

根據表 2 可得到所提出乘法器的空間複雜度，根據圖 4 所示主要的運算需要 k 次的乘法迴圈才可完成運算，其中 $k=\left\lceil\frac{m}{2d}\right\rceil$，在完成 k 次運算後，圖 4 的步驟 13 執行 mod 的動作需要消耗 1 個脈波週期，故所提出的時間複雜度需要($\left\lceil\frac{m}{2d}\right\rceil$+1)個脈波週期而文獻[13]需要$\left\lceil\frac{m}{d}\right\rceil$+2 個脈波週期，文獻[15]需要 $2\frac{m}{d}$ 個脈波週期，文獻[19]需要$\left\lceil\frac{m}{d}\right\rceil$個脈波週期，將所提出的多位元串列乘法器與文獻[13], [15]以及[19]相比，可得到如下表 3 之時間以及空間複雜度比較，根據表 3，可獲得最佳的分割位元 d，由表 3 可知當分割位元 d 越大時，所需的時間複雜度越少，但相對的空間複雜度將會越大;當分割位元 d 越小時，所需的空間複雜度越少，但相對的空間複雜度將會越大。

表3 GF(2^m)中提出的乘法器與現有乘法器時間以及空間複雜度比較

乘法器	Kumar [13]	Talapatra [15]	M.Morales[19]	Proposed
AND 閘	$(m+2)d+2(d-1)$	dm	dm	$\frac{3dm}{2}$
XOR 閘	$md+3d-2+(m-1)(n-1)$	$dm+2d$	$d(2m+1)$	$4m+n+\frac{3dm}{2}+\frac{m}{2}+d-5$
暫存器	$(n+2)m+2d-(n+1)$	$4m+3d+1$	$2m$	$3m-3$
多工器	0	$2m$	0	0
脈波周期	$\lceil\frac{m}{d}\rceil+2$	$2\frac{m}{d}$	$\lceil\frac{m}{d}\rceil$	$\lceil\frac{m}{2d}\rceil+1$

其中 m 為所選的乘法位元數，n 為 $F(x)$中間次方，d 為多位元串列選擇的大小

4.1 實驗結果

第 4.1 節談到的是理論的空間以及時間複雜度分析，在本節我們將所提出的乘法器實際模擬以及燒入至 FPGA 實驗平台，我們利用 Altera FPGA Quartus II 軟體環境來模擬所設計的乘法器。實現於 Cyclone II EP2C70F896C8 之實驗平台上，下面的部份我們針對 36×36、84×84、126×126 以及 204×204 個位元的乘法器去做分析以及實驗模擬，並且與文獻[13]、[15]以及[19]比較其時間複雜度，空間複雜度以及時間×空間複雜度[13]及[20]，獲得如下表 4，表 5 以及表 6，由表 4 可看出文獻[13]、[15]以及[19]當乘法器為 36×36 個位元時與乘法器為 204×204 個位元時所增加的邏輯閘數非常多，以文獻[13]為例，當乘法器為 36×36 個位元時需要 1811 個邏輯閘數，但乘法器為 204×204 個位元時需要到 51572 個邏輯閘數，而所提出的乘法器乘法器為 36×36 個位元時需要 701 個邏輯閘數，但乘法器為 204×204 個位元時只需要 9449 個邏輯閘數，由表 5 可看出當乘法器為 126×126 個位元以及 204×204 個位元時，所提出的乘法器所需要的時間複雜度比較高，但是與文獻[13]、[15]以及[19]相比卻降低了比較多的時間複雜度。由表 6 可看出當乘法器的乘法位元越大時，與其他文獻比較所降低的空間×時間複雜度百分比越大，當乘法器的乘法位元為 204 個位元時，所提出的乘法器與文獻[13]相比降低了 91.3%與文獻[15]相比降低了 79.4%，以及與文獻[19]相比降低了 70.4%。

表4 提出的乘法器與現有乘法器空間複雜度比較

乘法器 / 乘法器大小	Kumar [13]		Talapatra[15]		M.Morales [19]		Proposed
	Gate count	Reduced	Gate count	Reduced	Gate count	Reduced	Gate count
36 bits	1811	61.3%	910	23.0%	1044	32.9%	701
84 bits	9607	74.7%	4138	41.4%	5460	37.0%	2426
126 bits	17850	76.7%	6154	32.4%	8211	49.3%	4161
204 bits	54572	82.7%	15267	38.1%	21250	55.5%	9449
Average reduced	73.9%		33.7%		43.7%		-

表5 提出的乘法器與現有乘法器時間複雜度比較

乘法器 / 乘法器大小	Kumar [13]		Talapatra[15]		M.Morales [19]		Proposed
	Time	Reduced	Time	Reduced	Time	Reduced	Time
36 bits	240 ns	50%	360 ns	66.7%	160 ns	25.0%	120 ns
84 bits	240 ns	50%	320 ns	62.5%	160 ns	25.0%	120 ns
126 bits	320 ns	50%	480 ns	66.7%	240 ns	33.3%	160 ns
204 bits	320 ns	50%	480 ns	66.7%	240 ns	33.3%	160 ns
Average reduced	50%		57.4%		29.2%		-

表6 提出的乘法器與現有乘法器空間×時間複雜度比較

乘法器 / 乘法器大小	Kumar [13]		Talapatra[15]		M.Morales [19]		Proposed
	Gate count × Time	Reduced	Gate count × Time	Reduced	Gate count × Time	Reduced	Gate count × Time
36 bits	434640	80.6%	327600	74.3%	167040	49.6%	84120
84 bits	2305680	87.4%	1324160	78.0%	873600	66.7%	291120
126 bits	5712000	88.3%	2953920	77.5%	1970640	66.2%	665760
204 bits	17463040	91.3%	7328160	79.4%	5100000	70.4%	1511840
Average reduced	86.9%		77.3%		63.2%		-

將表 6 整理後，可獲得四種乘法器大小與文獻[13]、[15]及[19]之空間×時間複雜度比較如圖 7 以及所降低的空間×時間複雜度百分比如圖 8，由圖 7 可知當乘法器越大時，所增加的空間×時間複雜度就會越大，但是所提出的乘法器，當乘法器位元數增加時與其他論文比較上升的幅度相對地比較小。由圖 8 可得知所提出的乘法器與其他文獻相比，當乘法器的乘法位元越大時所降低的空間×時間複雜度百分比越大：

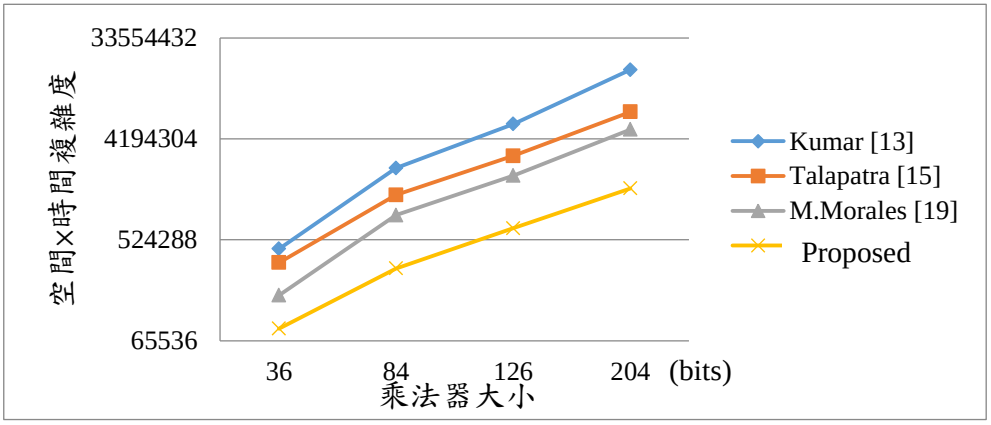

圖7 提出乘法器與現有論文空間×時間複雜度比較

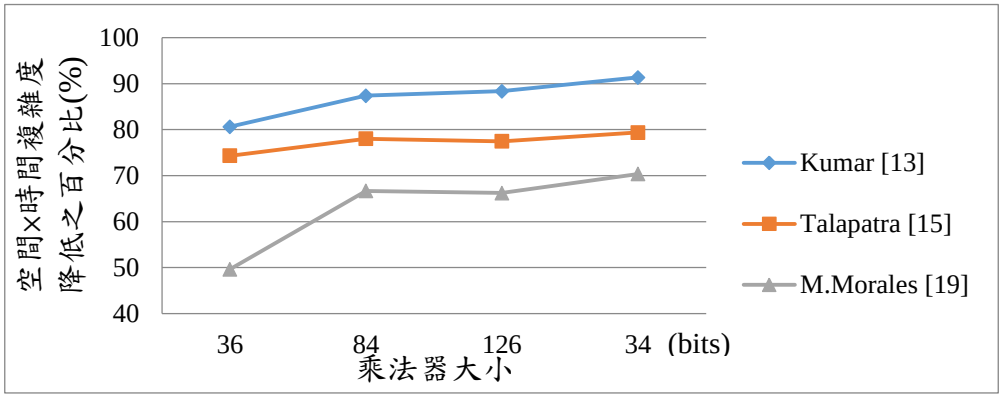

圖8　所提出乘法器與現有論文空間x時間複雜度降低之百分比

伍、結論

本文提出了低複雜度的乘法器在有限場 GF(2^m)內。這種方法結合了多位元串列以及 Karatsuba 兩種方式的概念並擷取其優點，如果分割位元 d 的大小選的好，可以降低空間複雜度，並且獲得時間與空間複雜度之間的平衡，當乘法器越大時，所節省的複雜度就會隨之增加，非常適用於環境資源很小，但所需的資料量很大的環境，例如手機以及 smart card 等等，本篇論文實現於 Altera FPGA Quartus II 軟體環境當中，根據實驗結果統計，分割位元的最佳值可使乘法器之時間×空間複雜度達到最低。本文提出四種不同乘法位元數的乘法器，分別為 36×36，84×84，126×126 以及 204×204 個位元的乘法器，本文最小的乘法器為 36×36 個位元，此乘法器與文獻[13]、[15]以及[19]相比分別降低了 80.6%，74.3%以及 49.7%，而本文最大的乘法器例子為 204×204，此乘法器與文獻[13]、[15]以及[19]相比分別降低了 91.3%，79.4%以及 70.4%，由實驗結果可證明，當乘法器相乘的位元數越多的時候，所提出的乘法器將會節省越多的空間x時間複雜度。

參考文獻

[1] V. S. Miller, "Use of elliptic curves in cryptography," *Lecture notes in computer*, pp. 417-429, 1986.

[2] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, No. 177, pp. 203-209, 1987.

[3] C. Y. Lee, "Low-complexity bit-parallel systolic multipliers over GF(2^m)," *IEEE International Conference on Systems*, vol. 2, pp. 1160-1165, Oct. 2006.

[4] C. Y. Lee, C. W. Chiou, J. M. Lin, and C. C. Hang, "Scalable and systolic Montgomery multiplier over GF(2^m) generated by trinomials," *IET Circuits, Devices & System*, vol.1, No. 6, pp. 477-484, Dec. 2007.

[5] S. T. J. Fenn, M. Benaissa, and O. Taylor, "Dual basis systolic multipliers for GF(2^m)," *IEEE Computers and Digital Techniques*, vol. 144, No. 1, pp. 43-46, Jan. 1997.

[6] P. K. Meher, "On efficient implementation of accumulation in finite field over GF(2^m) and its applications," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 17, No. 4, pp. 541-550, Apr. 2009.

[7] C. Y. Lee and C. W. Chiou, "Scalable gaussian normal basis multipliers over GF(2^m) using hankel matrix-Vector representation," *J. Signal Processing Systems*, vol. 69, No 2 pp. 197-211, 2012.

[8] H. Fan and M. A. Hasan, "A new approach to subquadratic space complexity parallel multipliers for extended binary fields," *IEEE Trans. on Computers*, vol. 56, No. 2, pp. 224-233, Feb. 2007.

[9] C. Y. Lee, E. H. Lu, and J. Y. Lee, "Bit-parallel systolic multipliers for GF(2^m) fields defined by all-one and equally spaced polynomials," *IEEE Trans. on Computers*, vol. 50, No. 5, pp. 385-393, May. 2001.

[10] P. K. Meher, "Systolic and non-systolic scalable modular designs of finite field multipliers for reed-solomon codec," *IEEE Trans. on Very Large Scale Integr. (VLSI) Syst*., vol. 17, No. 6, pp. 747-757, Jun. 2009.

[11] J. Xie, P. K. Meher, and J. He, "Low-complexity multiplier for GF(2^m) based on all-one polynomials," *IEEE Trans. on Very Large Scale Integr. (VLSI) Syst*, No. 99, pp. 1-5, Jan. 2012.

[12] G. N. Selimis, A. P. Fournaris, H. E. Michail, O. Koufopavlou, "Improved throughput bit-serial multiplier for GF(2^m) fields," *Integration*, in *Proc. of the VLSI journal*, vol. 42, pp. 217-226, 2009.

[13] S. Kumar, T. Wollinger, and C. Paar, "Optimum digit serial GF(2^m) multipliers for curve-based cryptography," *IEEE Trans. on Computers*, vol. 55, No. 10, pp. 1306-1311, Oct. 2006.

[14] A. Hariri and A. Reyhani-Masoleh, "Digit-serial structures for the shifted polynomial basis multiplication over binary extension fields," *Lecture Notes in Computer Science*, vol. 5130, pp. 103-116, 2008.

[15] S. Talapatra, H. Rahaman, and J. Mathew, "Low complexity digit serial systolic Montgomery multipliers for special class of GF(2^m)," *IEEE Trans. on Very Large Scale*

Integr. (VLSI) Syst., vol. 18, No. 5, pp. 487-852, 2010.

[16] Karatsuba A., Ofman Yu. "Multiplication of multi-digit numbers on automata", *Soviet Physics Doklady,* vol 7 pp. 595-596, Jan. 1963.

[17]C. Grabbe, M. Bednara, J. Teich, J. von zur Gathen, and J. Shokrollahi, "FPGA designs of parallel high performance GF(2^{233}) multipliers," *in Proc. Int. Symp. Circuits Syst. (ISCAS),* pp. 268-271, May. 2003.

[18]Zhengzheng Ge, Guochu Shou, Yihong Hu, and Zhigang Guo, "Design of Low Complexity GF(2^m) multiplier based on karatsuba algorithm," *IEEE 13th International Conference on Communication Technology* (*ICCT*), pp. 1018-1022, Sept. 2011.

[19]M. Morales-Sandoval, C. Feregrino-Uribe, and P. Kitsos, "Bit-serial and digit-serial GF(2^m) Montgomery multipliers using linear feedback shift registers," in *Proc. of Computers & Digital Techniques(IET)* , pp.86-94, March 2011.

[20] Pramod Kumar Meher, "Systolic and Non-Systolic Scalable Modular Designs of Finite Field Multipliers for Reed–Solomon Codec," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* , pp.747-757, June 2009.

李秋瑩於 1985 年獲得中原大學醫學工程學士學位，並且於 1994 年獲得中原大學電機工程研究所碩士學位，接著就讀長庚大學繼續攻讀博士學位，並且於 2001 獲得長庚大學電機工程研究所博士學位。他現在為龍華科技大學資訊網路工程系教授。他主要研究興趣包括有限場，錯誤更正碼訊號處理以及數位傳輸系統的計算。他是 IEEE 的高級會員，並且於 2001 年榮獲「中華民國斐陶斐榮譽學會」榮譽會員。

范家禎於 2001 年獲得龍華科技大學資訊網路工程系學士學位，並且於 2013 年獲得台北科技大學電腦與通訊研究所碩士學歷，目前正在就讀交通大學資訊科學與工程研究所博士學位。他的主要研究領域為有現場，錯誤控制碼以及訊號處理。

葉為勳目前正在就讀龍華科技大學資訊網路工程系學士學位。他的主要研究領域為有現場，錯誤控制碼以及訊號處理。