3. FPGA project

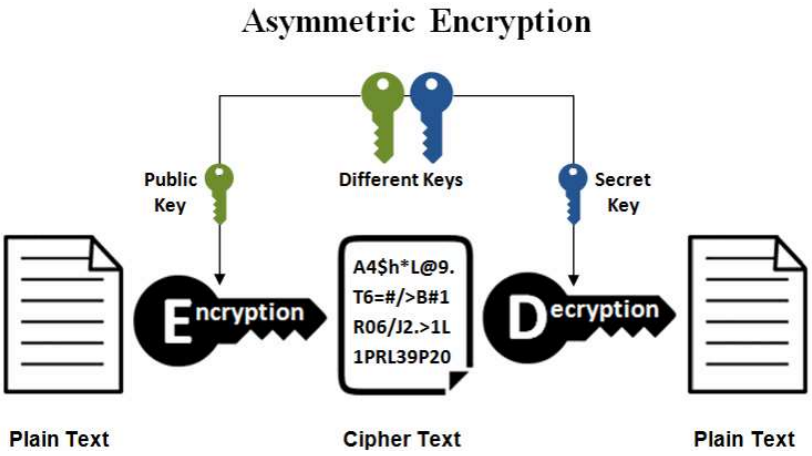
GitHub URL: https://github.com/Chia-Yu-Kuo/FPGA-Project

(1) Elliptic-curve cryptography

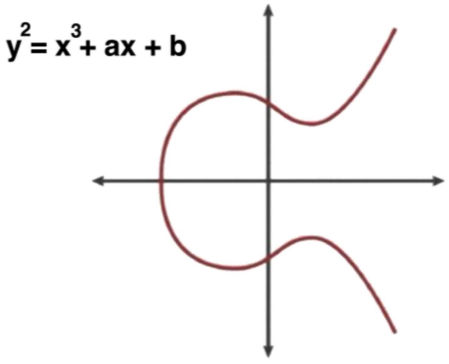
甲、摘要

ECC 是一種非對稱密碼學。對非對稱加密算法利用公鑰(Public Key)，對明文(Plain text)進行加密產生密文(Cipher text)，再利用私鑰(Private Key)對密文進行解密，產生明文。而 ECC 是一種基於橢圓取線上取利散點進行運算的密碼學，其安全性能更高且處理速度更快，在我們的加解密過程中，會需要用的橢圓取線上的點加法與標量乘法運算。

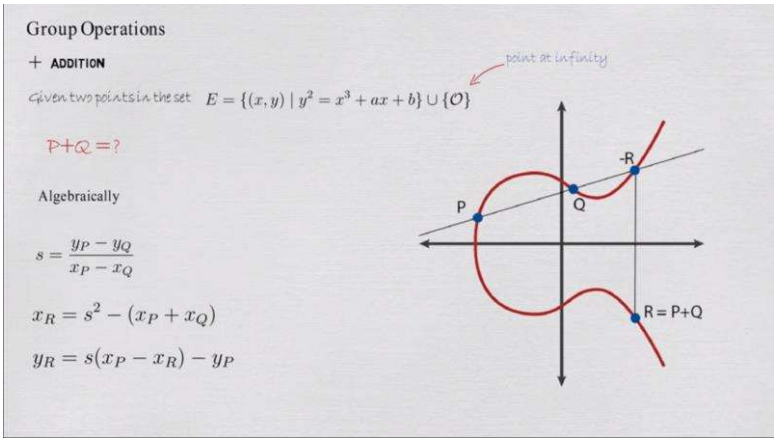
ECC 演算法主要可由 point addition 及 point double，這兩個計算過程中會遇到要求 mod 的反函數，而由於加密都是大數運算，所以會一直用到 modulus 運算。



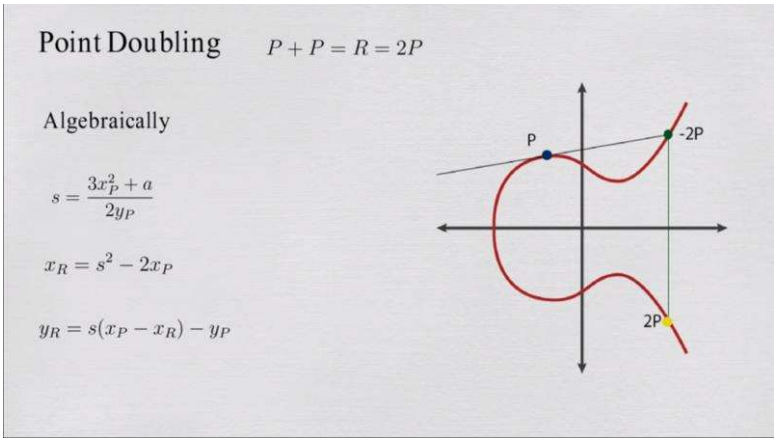
圖一、非對稱加密流程



圖二、橢圓曲線方程式



圖三、point add 運算

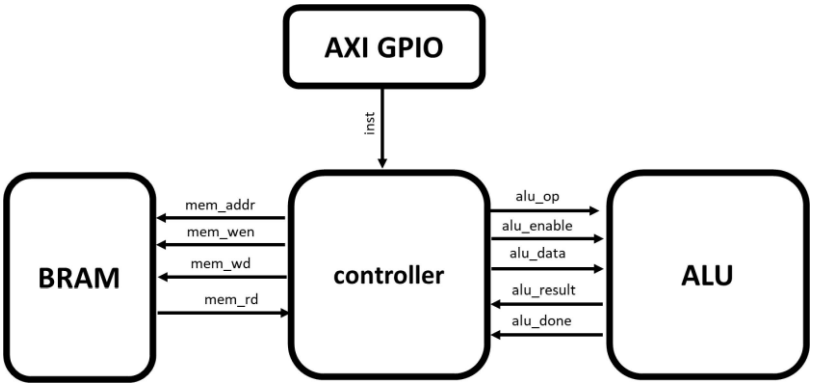


圖四、point doubling 運算

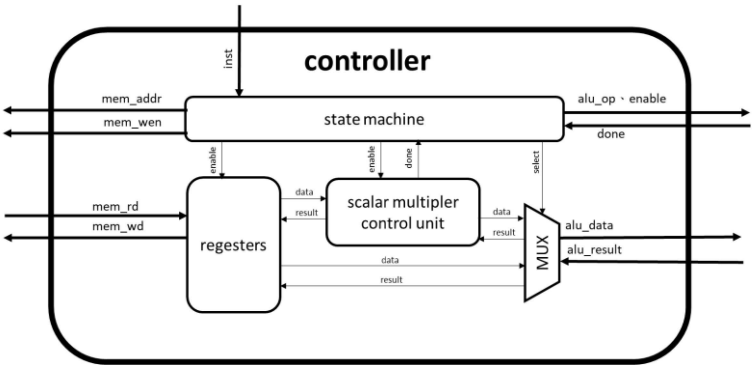
乙、想法

i. Architecture:

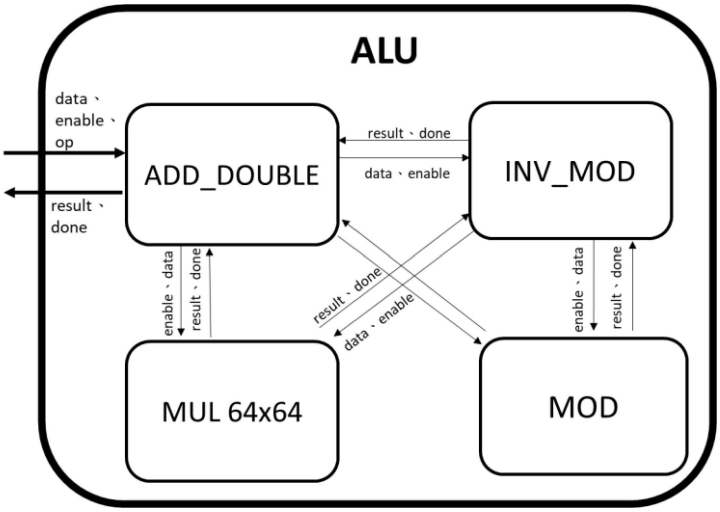
PL 端電路將其劃分為 Controller, ALU 及 BRAM 三塊，其中 Controller 會接收 GPIO 從 PS 端接收指令，來進行產生公鑰及加解密。而 ALU 由 ADD_DOUBLE、INV_MOD、MUL64 及 MOD 所組成。Bram 則是 PS 端透過 Bram Controller 輸入金鑰資訊。



圖五、系統方塊圖



圖六、Controller 架構

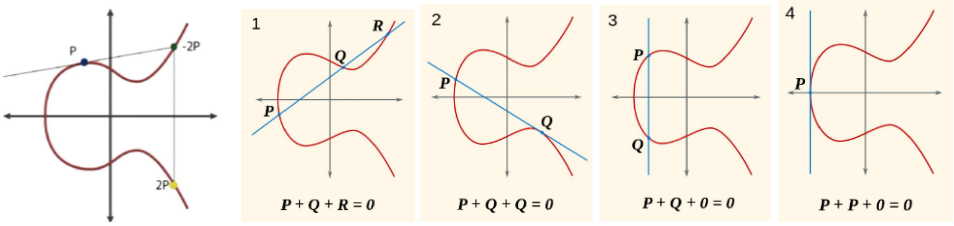


圖七、ALU 架構

ii. Algorithm

由於 ALU 內部是對大數的點做投射，但礙於硬體有限制，因此必須設計一套適合運算大數的電路。此外由於電路會非常龐大，因此會一直用 Resource sharing 搭配 Handshake 機制來實現 ECC 電路。

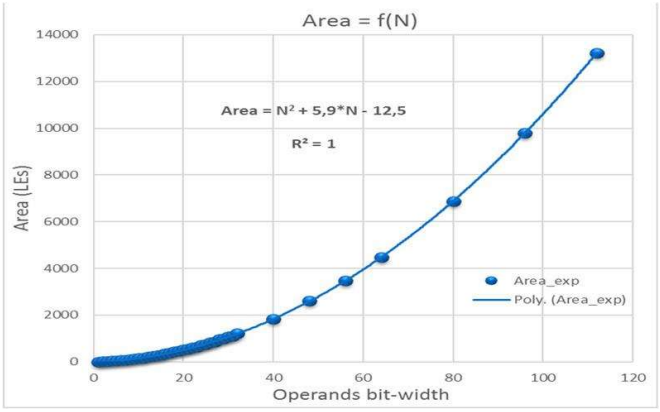
ADD_DOUBLE: 將 ADD、SUB、DOUBLE 三者演算法做比較可發現其硬體部分可以共用，為了節省資源，將可共用的電路提出，並利用 Controller 控制來區分 mode。



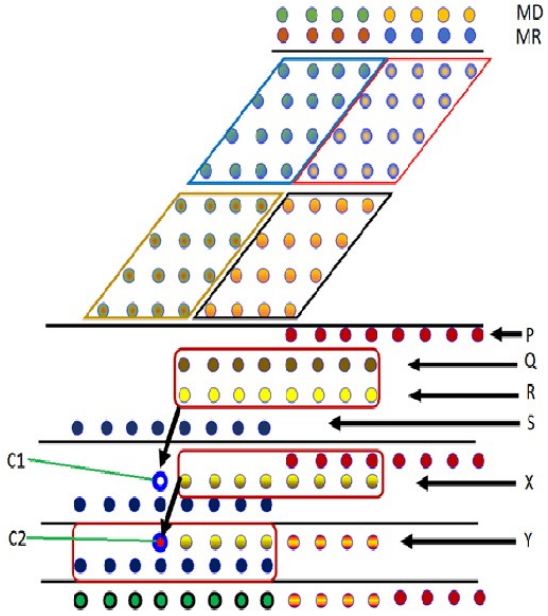
圖八、ADD、SUB、DOUBLE 運作

MUL64: 由於大數的乘法器是非常沒效率的，而且面積相對也會隨 bits 數增加而指數成長，因此我們將 64*64 的動作拆成四組 32*32，但是所使用的 FPGA 版提供 DSP 的

最大 bit 數為 18bits，因此又將每組 32*32 拆成四組 16*16。



圖九、乘法器 Area 隨 bits 數增加而指數成長



圖十、拆解乘法器原理

MOD: 原先是利用多級除法器取餘數實作 mod，後來利用 pre-division + shift 可快速做出 mod 運算。

$$a \,\%\, q = a - \left\lfloor \frac{a}{q} \right\rfloor \cdot q = a - \left\lfloor \frac{a \times \frac{2^k}{q}}{2^k} \right\rfloor \cdot q$$

圖十一、MOD by pre-division + shift

INV_MOD: 利用費馬小定理將原先必須用方程式才能解的問題改成函式，再利用模冪求指數的 mod 運算。

費馬小定理（英語：Fermat's little theorem）是數論中的一個定理。假如a是一個整數，p是一個質數，那麼$a^p - a$是p的倍數，可以表示為

$$a^p \equiv a \pmod{p}$$

如果a不是p的倍數，這個定理也可以寫成更加常用的一種形式

$$a^{p-1} \equiv 1 \pmod{p}$$[1][註 1]

圖十二、費馬小定理

模冪（英語：modular exponentiation）是一種對模進行的冪運算，在計算機科學，尤其是公開密鑰加密方面有一定用途。

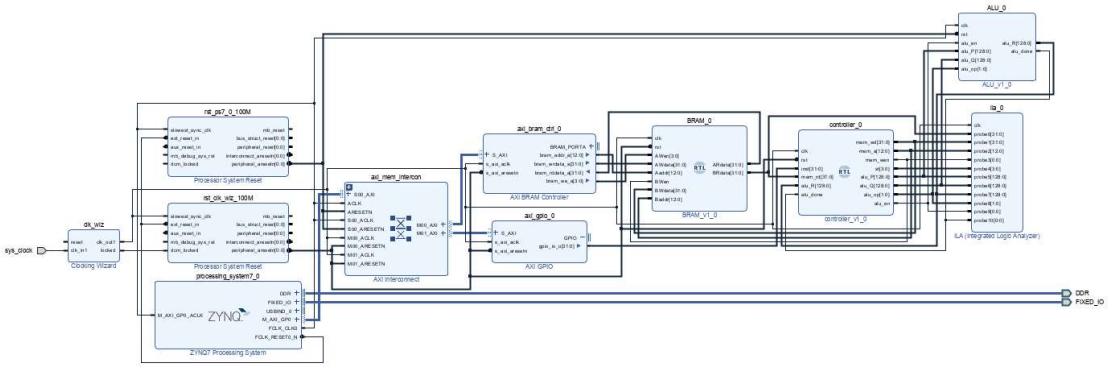
模冪運算是指求整數b的e次方b^e被正整數m所除得到的餘數c的過程，可用數學符號表示為$c = b^e \bmod m$。由c的定義可得$0 \leq c < m$。

例如，給定$b = 5$，$e = 3$和$m = 13$，$5^3 = 125$被13除得的餘數$c = 8$。

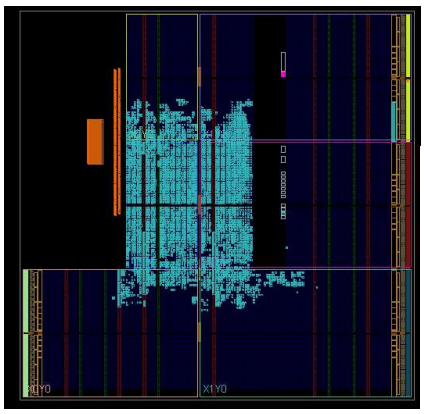
圖十三、模冪

最後利用 Block design 搭建好電路後利用 embedded C 在 Vitis IDE 上控制 PS 端來驗證電路。

iii. 結果



圖十四、Block design



Resource	Utilization	Available	Utilization %
LUT	7684	53200	14.44
LUTRAM	658	17400	3.78
FF	8465	106400	7.96
BRAM	14.50	140	10.36
DSP	4	220	1.82
IO	1	125	0.80
MMCM	1	4	25.00

圖十五十六、FPGA 硬體使用率

```
 Loading work.INV_MOD
 Refreshing C:/Users/oppo1/Desktop/FPGA/final_project/final/src/work.ADD_DOUBLE
 Loading work.ADD_DOUBLE
SIM 82> run -all
              109960 00000000000000001000000000000005 * 00000000000000001000000000000005  outcome:                3,                6, op=2
              219280 00000000000000001000000000000005 * 00000000000000003000000000000006  outcome:                6,               10, op=0
              328600 00000000000000001000000000000005 * 0000000000000000600000000000000a  outcome:                1,                3, op=0
              437920 00000000000000001000000000000005 * 00000000000000001000000000000003  outcome:               16,                9, op=0
              547240 00000000000000001000000000000005 * 00000000000000011000000000000009  outcome:               13,               16, op=0
              656560 00000000000000001000000000000005 * 0000000000000000d000000000000010  outcome:                6,                0, op=0
              765880 00000000000000001000000000000005 * 00000000000000006000000000000000  outcome:                7,               13, op=0
              875200 00000000000000001000000000000005 * 0000000000000000700000000000000d  outcome:                6,                7, op=0
              984520 00000000000000001000000000000005 * 00000000000000006000000000000007  outcome:               11,                7, op=0
             1093840 00000000000000001000000000000005 * 0000000000000000b000000000000007  outcome:               10,               13, op=0
             1203160 00000000000000001000000000000005 * 0000000000000000a000000000000000  outcome:               11,                0, op=0
             1312480 00000000000000001000000000000005 * 0000000000000000b000000000000000  outcome:                4,               16, op=0
             1421800 00000000000000001000000000000005 * 00000000000000004000000000000010  outcome:                1,                9, op=0
             1531120 00000000000000001000000000000005 * 00000000000000001000000000000009  outcome:               16,                3, op=0
             1640440 00000000000000001000000000000005 * 00000000000000001000000000000003  outcome:               11,               10, op=0
             1749760 00000000000000001000000000000005 * 0000000000000000b000000000000000a  outcome:               14,                6, op=0
             1859080 00000000000000001000000000000005 * 0000000000000000e000000000000006  outcome:               16,                5, op=0
             1968400 00000000000000001000000000000005 * 00000000000000001000000000000005  outcome:               16,                7, op=0
** Note: $finish    : C:/Users/oppo1/Desktop/FPGA/final_project/final/src/alu_tb.v(161)
   Time: 1968400 ns  Iteration: 3  Instance: /alu_tb
```

$$E:\ y^2 \equiv x^3 + 2x + 2 \pmod{17}$$

$G = (5, 1)$
$2G = (6, 3)$
$3G = (10, 6)$
$4G = (3, 1)$
$5G = (9, 16)$
$6G = (16, 13)$
$7G = (0, 6)$
$8G = (13, 7)$
$9G = (7, 6)$
$10G = (7, 11)$
$11G = (13, 10)$
$12G = (0, 11)$
$13G = (16, 4)$
$14G = (9, 1)$
$15G = (3, 16)$
$16G = (10, 11)$
$17G = (6, 14)$
$18G = (5, 16)$
$19G = \mathcal{O}$

圖十七十八、橢圓曲線運算結果與 Golden 相同

```
Please choose operation

(1: generate key, 2: encrypt, 3: decrypt)
mode is 1

Please input private key k : (in hex)
 k is : 66

Public Key :
Kx = 5f4c2e935e437212 (in hex)
Ky = 64774be9e77b27bc (in hex)
```

圖十九、金鑰產生過程

```
Please choose operation

(1: generate key, 2: encrypt, 3: decrypt)
mode is 2

Please input x position of public key Kx : (in hex)
Kx is 5f4c2e935e437212

Please input y position of public key Ky : (in hex)
Ky is 64774be9e77b27bc

Please input x position of message Mx : (in hex)
Mx is 1a0a

Please input y position of message My : (in hex)
My is 8f5c356536c9c638

Please input a random number r for encryption : (in hex)
 r is 5
Encrypted data :
C1x = 586ce3a67803a04d (in hex)
C1y = 5bf5cf683216ee70 (in hex)
C2x = 58269077f58dab89 (in hex)
C2y = 4ed1cc5e6f278f73 (in hex)
```

圖二十、加密過程


Please choose operation
(1: generate key, 2: encrypt, 3: decrypt)
mode is 3
Please input private key k : (in hex)
k is 66
Please input encrypted data C1x : (in hex)
C1x is 586ce3a67803a04d
Please input encrypted data C1y : (in hex)
C1y is 5bf5cf683216ee70
Please input encrypted data C2x : (in hex)
C2x is 58269077f58dab89
Please input encrypted data C2y : (in hex)
C2y is 4ed1cc5e6f278f73
Decrypted message :
Mx = bcf2249014e0131 (in hex)
My = 6767caa0dba7aad2 (in hex)

圖二一、解密過程