

1. 設計者姓名及聯絡電話

學生姓名1：賴譯文0915090510

學生姓名2：徐瑞擇 0958004230

學生姓名3：王珽 0918822507

2. 專題名稱

中文專題名稱：在質數體下搭載查找表的32-bit ECC處理器

英文專題名稱：32-bit ECC processor over prime field with lookup table

3. 全新設計或改版說明

在IC設計的研究領域裡，有相當多的研究在進行密碼學演算法相關的加速器，其中，橢圓曲線加密由於需要的key size較小，耗能也比其他非對稱式密碼學來得小，而被用在IOT（Internet of Thing）上面。由於橢圓曲線能在質數體以及二元體兩種field下進行運算，相對應的硬體設計也有所不同，因而造成有許多相關的研究蓬勃發展。

此案件為採用他人設計[1]，進行改善，並且做出一個32 bit的ECC處理器。在先前的設計中，設計者使用了特殊的演算法，來執行prime field下的乘法以及除法運算，圖一和圖二分別列出了在這次設計中，prime field下的乘法以及除法的演算法，在運算中，原本的運算數字皆會先乘以 2^{32} ，來將所有數字從一般的整數轉到montgomery field，會有這樣的前處理，是因為在ECC的運算中，所有的數字都必須要在prime field底下計算，因此會有大量和mod（同餘）相關的運算，而在硬體上，為了加速mod的計算效率，我們會先讓整數轉到montgomery field中，如此才能將運算速度慢、成本高的乘法、除法，轉換成有效率的shift和加減的形式，來讓整個橢圓曲線運算的critical path不會因為乘法、除法而變得過長，影響到整體的運算時間。

在傳統的ECC加密中，會利用點P（P為橢圓曲線下的Base Point）和私鑰k進行加密，計算出kP作為公鑰，而kP的座標值，會利用double-and-add的演算法，搭配上P的addition和double的公式（圖三）來進行計算，這樣的演算法，會利用key shifter，讓k一次shift一個bit，並且對shift出來的那個bit進行判斷：若該bit等於1的話，會對點座標進行add和double；若該bit等於0，則會對點座標進行double，如此一來，一個32 bit的k，會讓整個加密的過程中，進行32次的point double，以及平均16次的point addition。而在這次的設計中，我們使用了lookup table來進行運算速度的優化，藉由事先存取P~3P的座標，來讓key shifter能一次shift兩個bit，並且一次讀取兩個bit，讀出來的結果如果是00，則會對點座標直接進行double；如果是01、10、11的話，會讓點座標分別和P、2P、3P的座標做point addition後，再進行point double，進而讓point

addition的數量能減少，達成降低減少運算時間的目標。這樣的設計，為的就是在整體面積小幅增加的情況下，讓整體的運算時間下降，達成合理而有效的面積及時間的trade off。

Input: $A \equiv a \cdot r \pmod{p}$, $B \equiv b \cdot r \pmod{p}$, p , n .
Output: $C = \text{MonMul}(A, B) \equiv A \cdot B \cdot r^{-1} \pmod{p} \equiv a \cdot b \cdot r \pmod{p}$.

1. Let $A = (A_{n-1}, A_{n-2}, \dots, A_1, A_0)_2$, $C = 0$.
2. For i from 0 to $n - 1$ do $\{ T = (C + A_i \cdot B), C = \frac{T+T_0 \cdot p}{2} \}$
3. If $C \geq p$, then $C = C - p$.
4. Return C .

圖一、ECC scalar multiplication under prime field

Input: $A \equiv a \cdot r \pmod{p}$, $B \equiv b \cdot r \pmod{p}$, p , n .
Output: R , where $\begin{cases} R = \text{MonDiv}(A, B) \equiv a \cdot b^{-1} \cdot r \pmod{p}. \\ R = \text{ModDiv}(A, B) \equiv a \cdot b^{-1} \pmod{p}. \end{cases}$

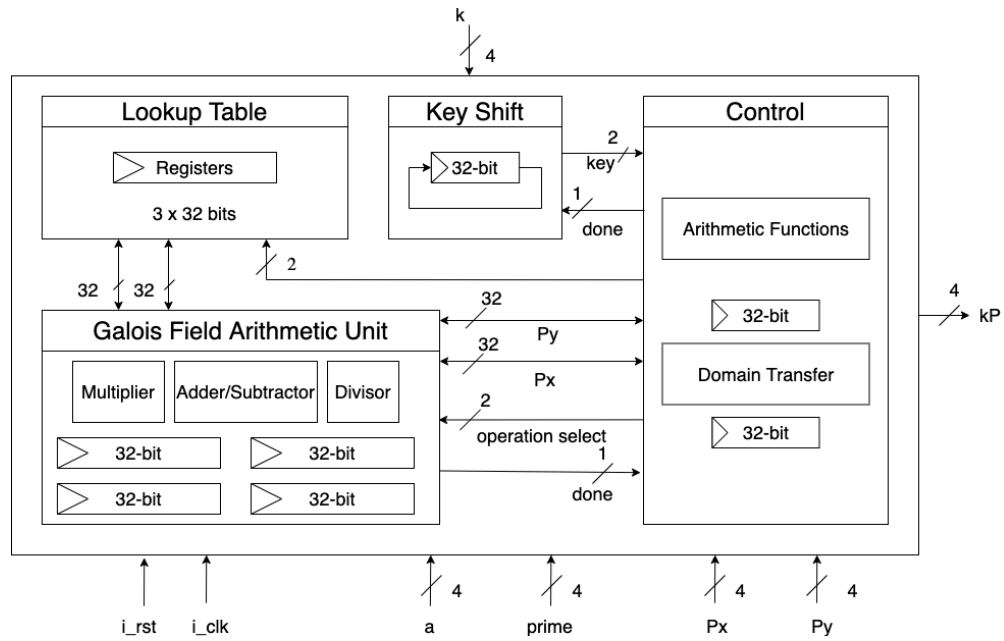
1. Let $U = p$, $V = B$, $R = 0$, $S = A$, $i = 0$.
2. While ($V > 0$) do{
 - If U is even, then $\{ U = \frac{U}{2}, S = 2S \}$
 - else if V is even, then $\{ V = \frac{V}{2}, R = 2R \}$
 - else if $U > V$, then $\{ U = \frac{U-V}{2}, R = R + S, S = 2S \}$
 - else $\{ V = \frac{V-U}{2}, R = 2R, S = R + S \}$
 - $i = i + 1$.
 - If $R \geq p$, then $R = R - p$.
 - If $S \geq p$, then $S = S - p$.
3. For j from 1 to $\begin{cases} i - n \\ i \end{cases}$ do $R = \frac{R+R_0 \cdot p}{2}$.
4. $R = p - R$.
5. Return R .

圖二、ECC scalar division under prime field

	Point Addition ($P_3 = P_1 + P_2$)	Point Doubling ($P_3 = 2P_1$)
$GF(p)$	$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ $x_3 = \lambda^2 - x_1 - x_2$ $y_3 = \lambda(x_1 - x_3) - y_1$	$\lambda = \frac{3x_1^2 + a_p}{2y_1}$ $x_3 = \lambda^2 - 2x_1$ $y_3 = \lambda(x_1 - x_3) - y_1$
$GF(2^n)$	$\lambda = \frac{y_2 + y_1}{x_2 + x_1}$ $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a_b$ $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$	$\lambda = x_1 + \frac{y_1}{x_1}$ $x_3 = \lambda^2 + \lambda + a_b$ $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$

圖三、Formulas for point addition and point double under prime field

4. 原理及架構說明

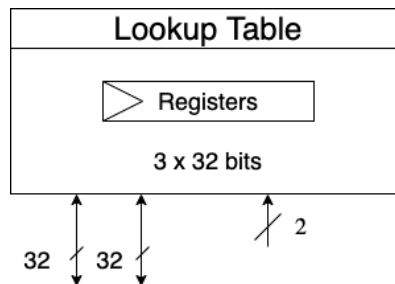


圖四、Design Architecture

圖四為我們本次設計的架構，主要分成五個module來設計，分別是Lookup Table、Key Shift、Control、Galois Field Arithmetic Unit (GFAU)、以及最外層的Top module，外圍的input為重置訊號i_rst、Clock訊號i_clk、常數a、質數prime、基點x座標Px、基點y座標Py、密鑰k，output是最後加密的結果kP。在執行加密的過程中，密鑰會先存在Key Shift Module，而Control module會先利用GFAU module來建構出Lookup Table，再來，會利用Lookup Table儲存的點座標，搭配Key Shift提供的2 bit key來決定使用Lookup Table裡的哪一點，並且利用GFAU module來完成在prime field下加減乘除的運算。

以下將會針對Lookup Table、GFAU、Control三個module來做更詳細的介紹。

A. Lookup Table

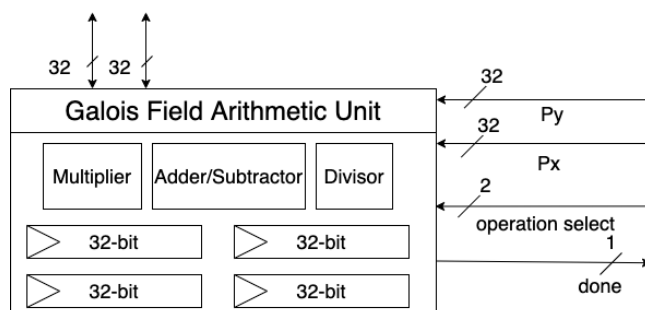


圖五、Lookup Table

Lookup Table的module架構圖如圖五，裡面會放三個32 bit的register來存放P~3P的座標，來提供之後運算時可以直接取得P~3P的座標，並且讓point addition的次數能夠減少。在這個module中，有兩個32 bit的input是拿來接收計算好的P~3P的座

標值、一個2 bit的input拿來決定要寫入、讀出P~3P的哪一個對應的register，還有兩個32 bit的output，來提供Lookup Table的資料給GFAU。

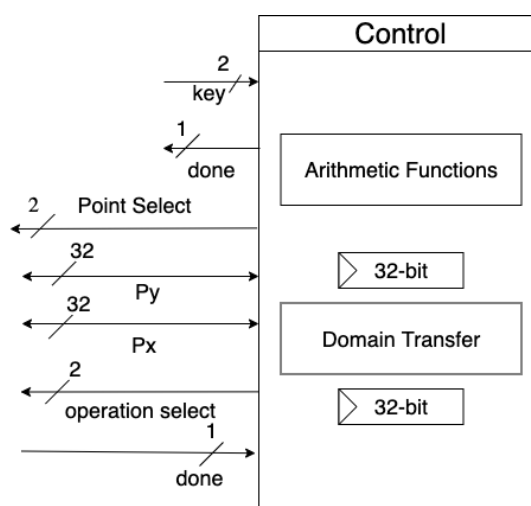
B. GFAU



圖六、GFAU

圖六為GFAU module的架構圖，在GFAU裡，會有實現前述乘法、除法演算法的multiplier和divisor，除了進行運算的電路外，GFAU裡面還有四個額外的register，用來存進行point addition、point double時，未完成的點座標暫存值。這個module有連接一個2 bit的input，用來決定這次是要做的是哪一種運算（加、減、乘、除）。在運算進行完成後，GFAU也會有一個1 bit的output，來讓control知道運算已經完成，並且從另外兩個32 bit的output輸出運算的x座標、y座標結果。

C. Control

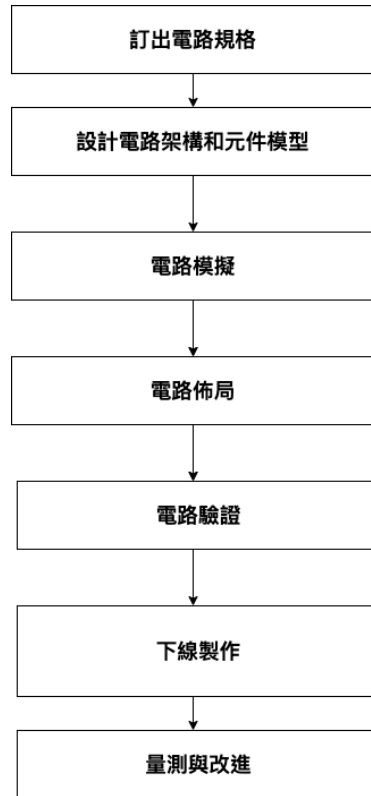


圖七、Control

圖七為Control module的架構圖，在這個module中，有一個Domain Transfer的sub-module，來將一般整數轉換到montgomery field上，並且在加密後的結果算出來後，將montgomery field上的數字轉換回一般的整數。此外，Control module也決定了該取用Lookup Table的哪一點、該用GFAU的哪一種運算，並且利用兩個32 bit的registers來

儲存算出來的點座標。此外，Control module會在準備好其他輸出訊號後，才會將output的done變為1，藉此讓其他module開始運作；此外，當其他module傳回done的訊號時，Control module才會跳到下一個工作，並重新發送指令給其他module。

5. 設計流程



6. 模擬結果

a. Pre-layout:

```
ncsim> run
FSDB Dumper for IUS, Release Verdi_N-2017.12, Linux, 11/12/2017
(C) 1996 - 2017 by Synopsys, Inc.
*Verdi* FSDB WARNING: The FSDB file already exists. Overwriting the
*Verdi* : Create FSDB file 'Top.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
-----
All data have been generated successfully!
-----PASS-----
-----
Simulation complete via $finish(1) at time 9865150 NS + 0
./Top_tb.v:315      $finish;
```

b. Post-layout:

```
(C) 1996 - 2017 by Synopsys, Inc.
*Verdi* FSDB WARNING: The FSDB file already exists. Overwriting
*Verdi* : Create FSDB file 'Top.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
-----
All data have been generated successfully!
-----PASS-----
Simulation complete via $finish(1) at time 9865150 NS + 0
./Top_tb.v:315      $finish;
```

7. 預計量測流程

- (1) 亂數產生多筆P的座標以及私鑰k。
- (2) 利用電腦模擬程式模擬加密結果，並且在RTL設計完成後也跑相同的測試，並對照兩者結果是否相同。
- (3) 儀器設置：電源供應器調整為1.8V直流電，並接上晶片的電源腳位。
- (4) 儀器設置：訊號產生器接至晶片的輸入腳位，並產生100MHz的方波，輸入i_clk腳位。
- (5) 儀器設置：將邏輯分析儀接至晶片的輸出腳位。
- (6) 使用訊號產生器向晶片輸入所需的測試資料。
- (7) 將晶片輸出與電腦模擬的結果相比對，看結果是否相同。

8. 佈局驗證結果錯誤說明

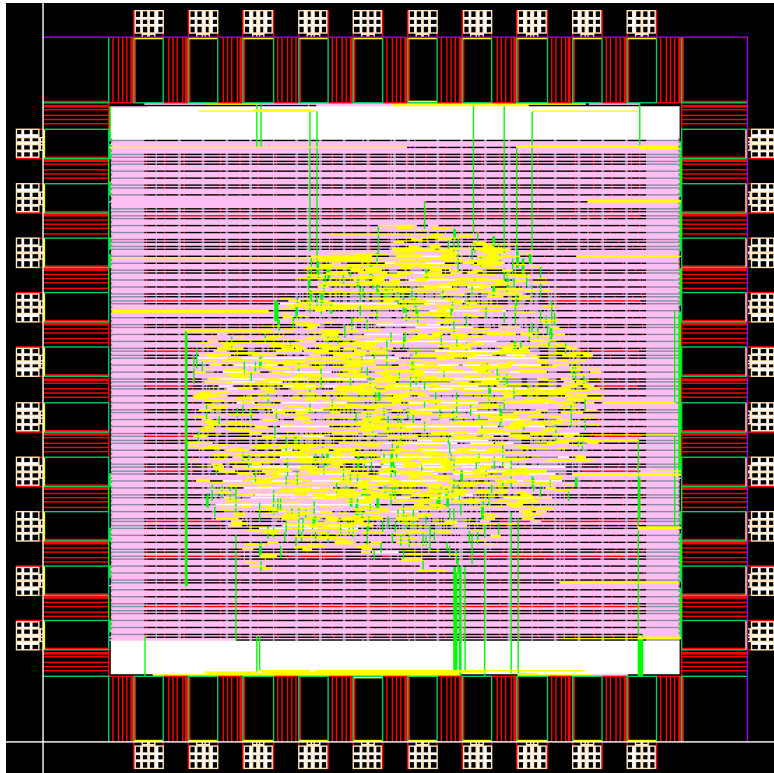
DRC:

```
~/coding/RISC-V/RISC-V-Single-Cycle-Processor --- b4502040@cad29:BaseRule --- ssh -X b4502040@140.112.20.72
4.22G_M2.density 4.31E_M6.density me
4.24F_M3.density 4.31F_M6.density
4.24G_M3.density G-DF-Mixed_Mode_RFCMOS18-1.8v_3.3v-1P6M-MMC-Calibre-DRC-2.11_P2
[b4502040@cad29 BaseRule]$ vim QA_pass.sum
[b4502040@cad29 BaseRule]$ vim QA_pass.sum
[b4502040@cad29 BaseRule]$ grep ^RULECHECK QA_pass.sum | grep -v 0$
RULECHECK 4.1M ..... TOTAL Result Count = 1
RULECHECK 4.29NOTICE ..... TOTAL Result Count = 1
RULECHECK 4.31A.8KA ..... NOT EXECUTED
RULECHECK 4.31B.a.8KA ..... NOT EXECUTED
RULECHECK 4.31B.b.8KA ..... NOT EXECUTED
RULECHECK 4.31C.a_d.8KA ..... NOT EXECUTED
RULECHECK 4.31C.b_c_e.8KA ..... NOT EXECUTED
RULECHECK 4.31D.8KA ..... NOT EXECUTED
RULECHECK 4.31A.12KA ..... NOT EXECUTED
RULECHECK 4.31B.a.12KA ..... NOT EXECUTED
RULECHECK 4.31B.b.12KA ..... NOT EXECUTED
RULECHECK 4.31C.12KA ..... NOT EXECUTED
RULECHECK 4.31D.12KA ..... NOT EXECUTED
RULECHECK 4.20F ..... TOTAL Result Count = 1
RULECHECK 4.20G ..... TOTAL Result Count = 1
RULECHECK 4.22F ..... TOTAL Result Count = 1
RULECHECK 4.22G ..... TOTAL Result Count = 1
RULECHECK 4.24F ..... TOTAL Result Count = 1
RULECHECK 4.24G ..... TOTAL Result Count = 1
RULECHECK 4.26F ..... TOTAL Result Count = 1
RULECHECK 4.26G ..... TOTAL Result Count = 1
RULECHECK 4.28F ..... TOTAL Result Count = 1
RULECHECK 4.28G ..... TOTAL Result Count = 1
RULECHECK 4.31E ..... TOTAL Result Count = 1
RULECHECK 4.31F ..... TOTAL Result Count = 1
RULECHECK 4.20C ..... TOTAL Result Count = 3
RULECHECK 4.22C ..... TOTAL Result Count = 3
RULECHECK 4.24C ..... TOTAL Result Count = 4
RULECHECK 6Aa.ME1 ..... NOT EXECUTED
RULECHECK 6Ab.ME1 ..... NOT EXECUTED
RULECHECK 6Ba.ME1 ..... NOT EXECUTED
RULECHECK 6Bb.ME1 ..... NOT EXECUTED
RULECHECK 6Aa.ME2 ..... NOT EXECUTED
RULECHECK 6Ab.ME2 ..... NOT EXECUTED
RULECHECK 6Ba.ME2 ..... NOT EXECUTED
RULECHECK 6Bb.ME2 ..... NOT EXECUTED
RULECHECK 6Aa.ME3 ..... NOT EXECUTED
```

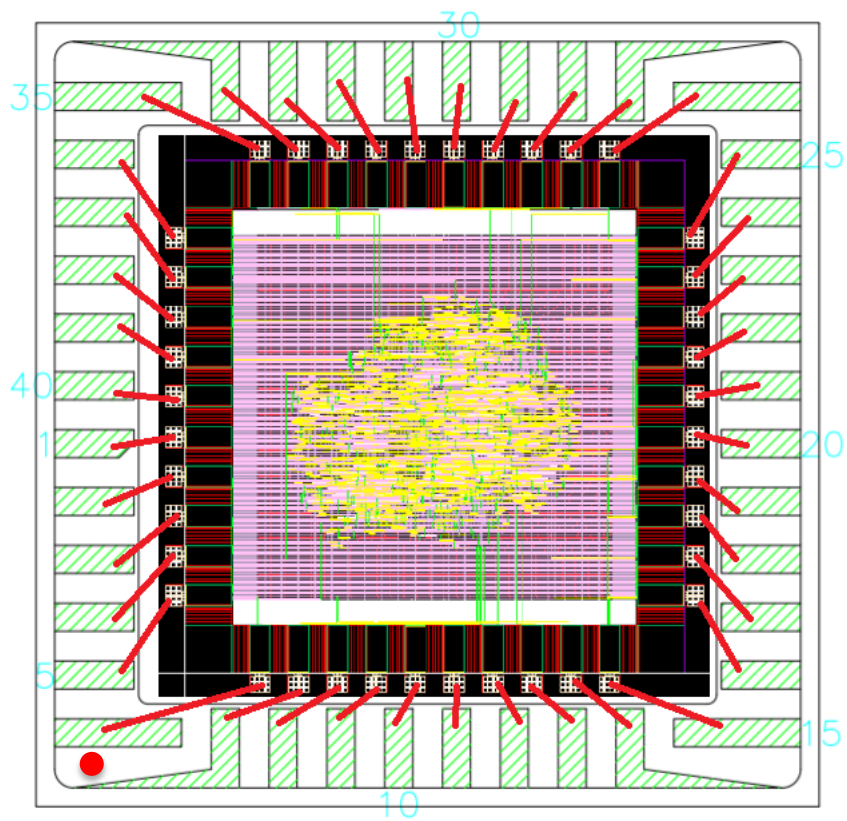
```
~/coding/RISC-V/RISC-V-Single-Cycle-Processor --- b4502040@cad29:BaseRule --- ssh -X b4502040@140.112.20.72
RULECHECK 6Ba.ME2 ..... NOT EXECUTED
RULECHECK 6Bb.ME2 ..... NOT EXECUTED
RULECHECK 6Aa.ME3 ..... NOT EXECUTED
RULECHECK 6Ab.ME3 ..... NOT EXECUTED
RULECHECK 6Ba.ME3 ..... NOT EXECUTED
RULECHECK 6Bb.ME3 ..... NOT EXECUTED
RULECHECK 6Aa.ME4 ..... NOT EXECUTED
RULECHECK 6Ab.ME4 ..... NOT EXECUTED
RULECHECK 6Ba.ME4 ..... NOT EXECUTED
RULECHECK 6Bb.ME4 ..... NOT EXECUTED
RULECHECK 6Aa.ME5 ..... NOT EXECUTED
RULECHECK 6Ab.ME5 ..... NOT EXECUTED
RULECHECK 6Ba.ME5 ..... NOT EXECUTED
RULECHECK 6Bb.ME5 ..... NOT EXECUTED
RULECHECK 6Aa.ME6 ..... NOT EXECUTED
RULECHECK 6Ab.ME6 ..... NOT EXECUTED
RULECHECK 6Ba.ME6 ..... NOT EXECUTED
RULECHECK 6Bb.ME6 ..... NOT EXECUTED
RULECHECK 5.1B ..... NOT EXECUTED
RULECHECK 5.1C ..... NOT EXECUTED
RULECHECK 5.1D ..... NOT EXECUTED
RULECHECK 5.1E ..... NOT EXECUTED
RULECHECK 5.1F ..... NOT EXECUTED
RULECHECK 5.1G ..... NOT EXECUTED
RULECHECK 5.1H ..... NOT EXECUTED
RULECHECK 5.1I ..... NOT EXECUTED
RULECHECK 5.1J ..... NOT EXECUTED
RULECHECK 5.1L ..... NOT EXECUTED
RULECHECK 5.1M ..... NOT EXECUTED
RULECHECK 5.1N ..... NOT EXECUTED
RULECHECK 5.1O ..... NOT EXECUTED
RULECHECK 5.1P ..... NOT EXECUTED
RULECHECK 5.1R ..... NOT EXECUTED
RULECHECK 5.1Q ..... NOT EXECUTED
RULECHECK 5.1S ..... NOT EXECUTED
RULECHECK 5.1T ..... NOT EXECUTED
RULECHECK 5.1U ..... NOT EXECUTED
RULECHECK 5.1V ..... NOT EXECUTED
RULECHECK 5.1W ..... NOT EXECUTED
RULECHECK 5.RECOMMEND ..... NOT EXECUTED
RULECHECK 5.NOTICE1 ..... NOT EXECUTED
RULECHECK 5.NOTICE2 ..... NOT EXECUTED
[b4502040@cad29 BaseRule]$
```

LVS: LVS-OK

9. 佈局平面圖



10. 打線圖



11. 效能展示

最高工作頻率	100 MHz
功率消耗	7.7216 mW
晶片面積	1500 um * 1500 um

12. 參考文獻

[1] Jen-Wei Lee, Yao-Lin Chen, Chih-Yeh Tseng, Hsie-Chia Chang, and Chen-Yi Lee, “A 52 1-bit Dual-Field Elliptic Curve Cryptographic Processor with Power” , Sept. 2010