設計內容

[1] 設計者姓名及連絡電話

學生姓名：張家翔、陳宥辰、陳永縉、莊詠翔
連絡電話：0965397080、0921141598、0917668235、0975720571
Email：b08901062@ntu.edu.tw, b08901048@ntu.edu.tw, b08901061@ntu.edu.tw, b08901093@ntu.edu.tw

[2] 專題名稱

中文專題名稱：使用蒙地卡羅法進行美式選擇權定價
英文專題名稱：American Option Pricing Using Monte-Carlo Method

[3] 全新設計或改版說明

此案件為設計者全新設計。

[4] 原理及架構說明

1. 簡介

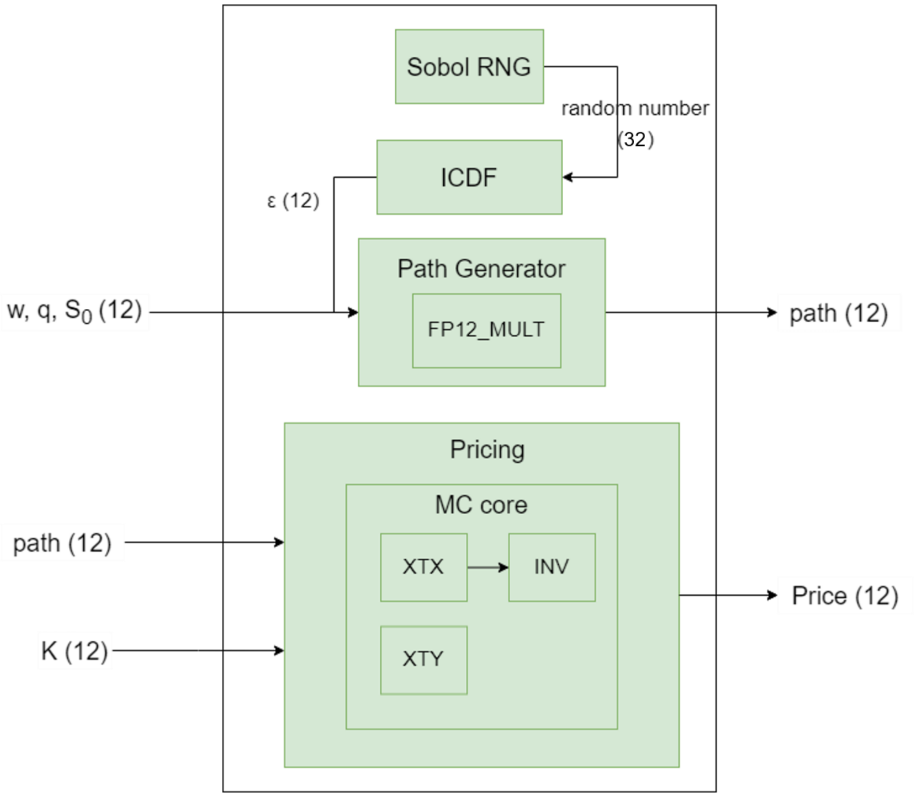
選擇權是一種期權，是一種未來可以購買或販賣某樣特定商品的憑證，而依據履約規定又可以分成美式選擇權及歐式選擇權，美式選擇權的所有權人可以在到期日或到期日前任一日要求履約，而歐式選擇權的所有權人僅可以在到期日當天要求履約。由於美式選擇權可在到期日前的任何時間履約，因此定價時必須考慮從定價當下到到期日之間的所有股價變動可能，計算方式較歐式選擇權複雜。

本設計是用蒙地卡羅演算法解決美式買權的定價問題。依照股價漲跌的隨機過程公式，我們產生多組隨機的選擇權價格路徑（Path），並利用回歸的方式計算每組路徑下的期望現金流，也就是所有權人的期望報酬，最後將所有路徑計算出的期望現金流做平均，代表該時刻所有權人持有買權的期望收益，同時作為買權之定價。

我們先利用 Sobol 隨機變數產生器生成一個 16 bits 的隨機變數，將此隨機變數導入 ICDF 逆累積分佈函數（Inverse Cumulative Distribution Function）後生成一個高斯分佈的訊號ε，將 w, q, S_0和剛剛生成的ε一起導入每日股價產生器（Path Generator）就可以生成具有隨機性的每日股價（Path），其中 w 和 q 為與股價及股價波動性有關之常數，S_0為目前股價。接著把每日股價和此美式選擇權的執行價格（K）一同導入第二部分的蒙地卡羅運算核心（Monte-Carlo Core）就可以得到一個現金流矩陣（Cash Flow Matrix），將此現金流矩陣的每一列做算數平均，就可以得到此美式選擇權的定價。

本晶片設計之整體架構如圖一所示，可以分為兩大部分，第一部分是產生每日隨機股

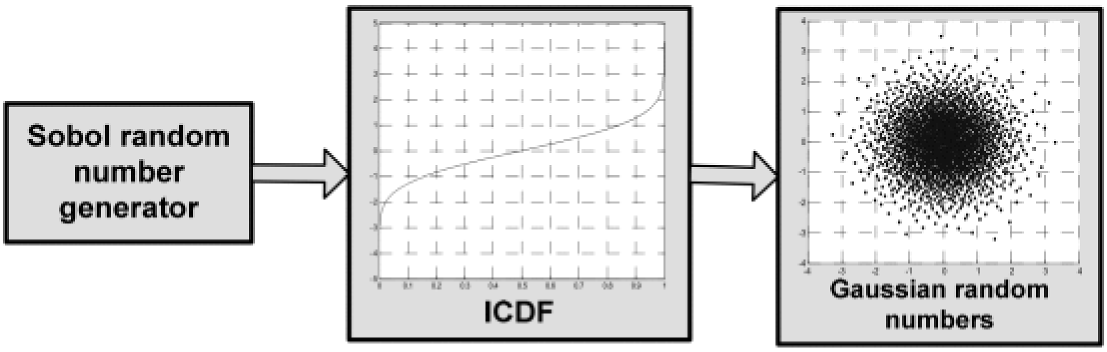
價（Sobol RNG、ICDF、Path Generator），第二部分是美式選擇權定價（Pricing），接下來將依序說明。



圖一、晶片整體架構

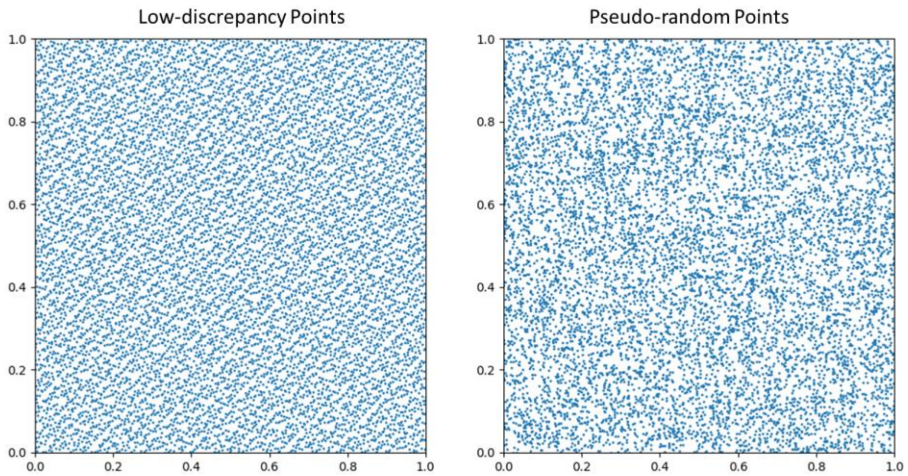
2. 產生隨機股價之原理與架構

在每條價格路徑的產生時，需要使用一個隨機變數 ε 來模擬價格變化的隨機性。圖二為 ε 的產生方式，本次實驗採用 Sobol 低差異性數列（Sobol low-discrepancy sequence）作為產生隨機變數的依據，並透過高斯分布的累積分布函數的反函數（Inverse Cumulative Distribute Function, ICDF）來將其轉換成高斯隨機變數，以作為價格路徑生成所需。

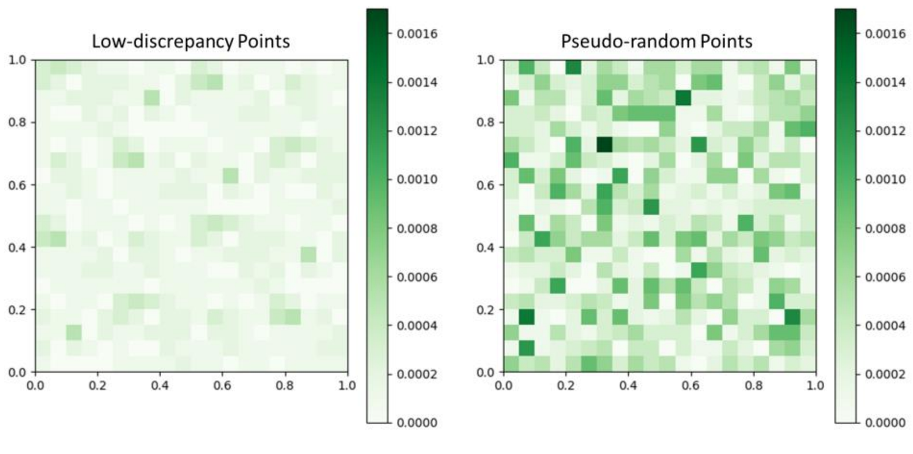


圖二、隨機變數 ε 產生流程

Sobol 低差異性數列是一種準隨機亂數（quasi-random number），此種方法給定了一個規則來產生所需數列，雖然嚴格上來說並不是完全隨機的數字，但由於其產生速度比真隨機數高，並且已被證明在蒙地卡羅的模擬上有接近真隨機數的效果，故被廣泛用於金融財務相關的分析。使用準隨機亂數而非一般偽隨機數（pseudo-random number）的目的是為了改善偽隨機數在多維度上的分布不均的問題，如下圖三所示，偽隨機數以梅森旋轉算法（Mersenne Twister random number generator）產生，圖四為將分布範圍分割成 20*20 的小方格，計算每格中點的實際出現比例與平均分布的出現率的差值，可以發現右圖會有較明顯的叢聚（Cluster）現象。

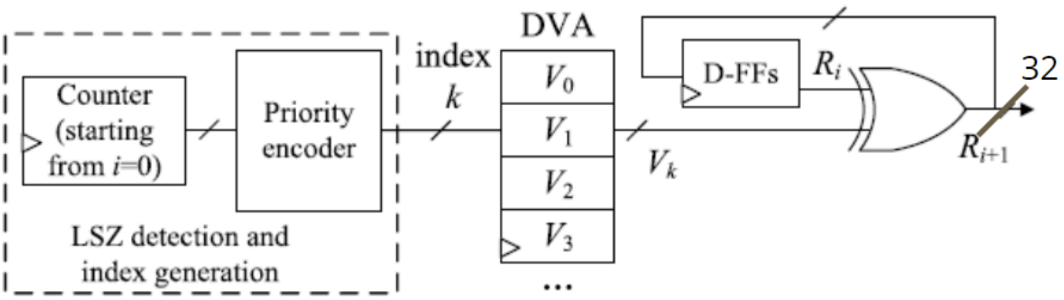


圖三、10000 個低差異性數列及偽隨機數點分布



圖四、分割方格中點的出現率與理論差值（平分面積成 400 個方格）

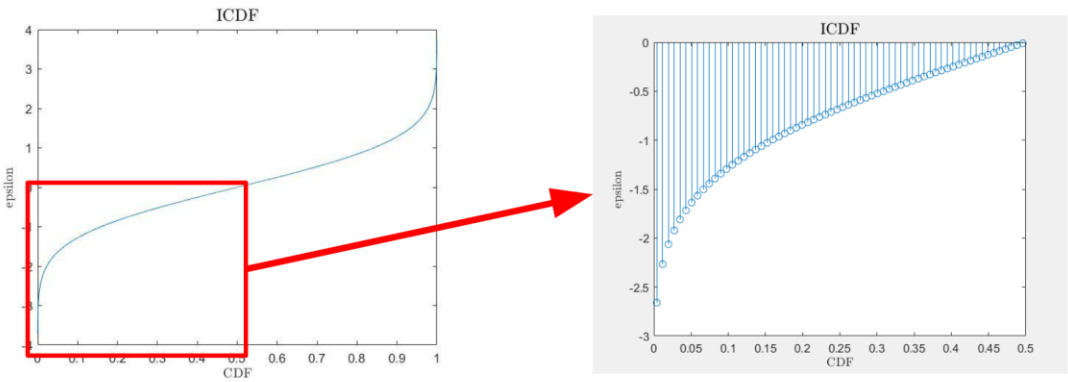
下圖五是 Sobol 亂數產生器的硬體架構細節。生成演算法主要採用 Antonov-Saleev(1979)提出的 Gray code 遞迴方式。首先使用一計數器（Counter）從 0 開始產生 index，接著將 32bits 的 index 送入 Priority encoder 以偵測最低有效位 0（Least Significant Zero，LSZ），即二進位表示下的最低位 0。接著根據 LSZ 的位置，將當前的亂數 R_i 與對應的 Direction Vector 做互斥或運算，即可產生下一個亂數 R_{i+1}。 Direction Vector 是透過本質多項式的係數經由遞迴運算產生，我們使用 Python 程式預先將其全部建立好後，儲存在晶片上的 Direction Vector Array（DVA）中，可以簡化計算的流程並增加計算速度。



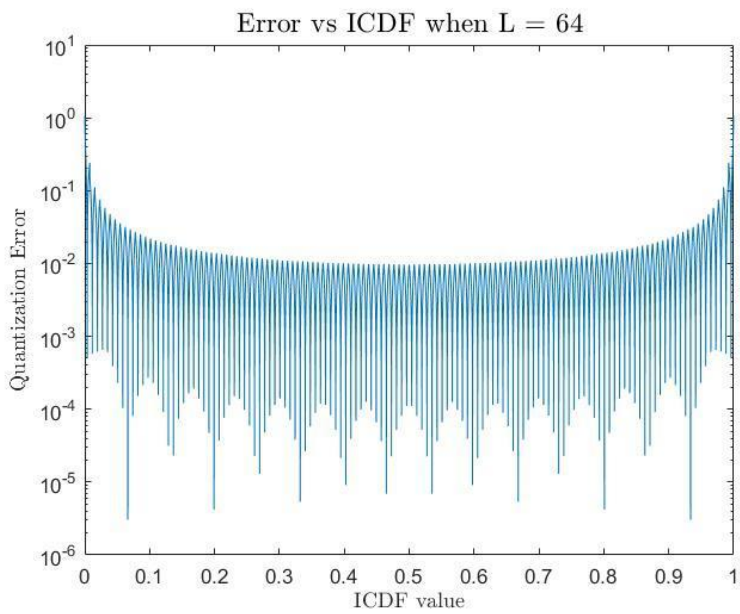
圖五、Sobol 亂數產生器架構圖

為了將 Sobol 亂數產生器得到的亂數轉為高斯分布，需要將其透過高斯分布的累積分布函數的反函數（Inverse Cumulative Distribution Function，ICDF）進行轉換。受限於晶片腳位數量的限制，在晶片外做轉換是不實際且費時的。但高斯分布的 ICDF 本身不是可解析的（analytic），故我們使用一階線性分段逼近來模擬原函式圖。作法是在原函式上解出 64 個點，並將這些值儲存在晶片上，以做量化的查表對應所需。由於 ICDF 對 x=0.5 是對稱的，可以將 x=0~0.5 段圖形作為標準來做參照，如圖六。這種方法可以達成在晶片上的高斯分布轉換，誤差率平均為 1.27%，如圖七所示。

路徑。



圖六、ICDF 量化示意圖



圖七、ICDF 量化點與原函數值誤差

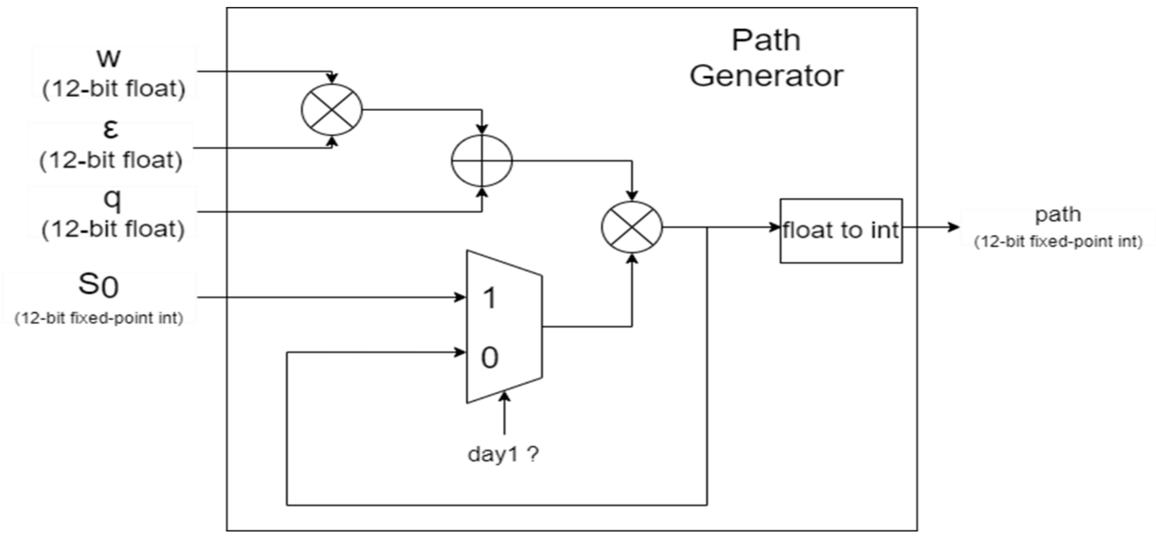
價格路徑的產生是透過下圖八的架構完成。由價格路徑方程式：

$$S_{\Delta t} = S_0(1 + \left(\mu - \frac{\sigma^2}{2}\right)\delta t + \sigma\varepsilon\sqrt{\delta t})$$

令上式中 $w = \sigma\sqrt{\delta t}$，$q = 1 + (\mu - \frac{\sigma^2}{2})\delta t$ 。我們可以改寫成：

$$S_{\Delta t} = S_0(q + w\varepsilon)$$

其中ε為圖二中產生隨機變數，S_0 為選擇權價格初始值，**w** 和 **q** 作為參數由晶片外輸入。計算第 **n+1** 天的價格時，再使用第 **n** 天的價格作為 S_0 遞迴求取，即可產生出所需的價格路徑。



圖八、價格路徑產生器架構

3. 美式選擇權定價之原理與架構

在軟體的模擬中，我們發現若考慮的 path 數量太少，則預測的結果可能會和市場上的實際定價相差超過 5%，因此決定考慮 1024 條 path，此數量在軟體模擬中都可以得到與實際定價相差小於 5%的結果。在前一階段的隨機股價產生過程中，共會產生 1024 條 path，每條 path 中含有 8 個隨機股價，即我們要使用 1024 個股價可能路徑來預測 8 天後到期的美式買權。

在每個時間點，選擇權所有人會考量「執行的收益」和「不執行的預期收益」何者較大，決定是否當下執行選擇權，對於美式買權，前者為股價和執行價格的差值，後者為接下來每個時間點的預期股價和執行價格的差值之平均。

以下舉例 3 天後到期的美式買權，假設 T=0 時股價為 1 元，執行價格為 0.9 元，考慮 8 條 path 如下表：

Path	T=0	T=1	T=2	T=3
1	1	0.91	0.92	0.66
2	1	0.84	0.74	0.46
3	1	0.78	0.93	0.97
4	1	1.07	1.03	1.09
5	1	0.89	0.44	0.48
6	1	1.24	1.23	1.1
7	1	1.08	1.16	0.99

8	1	1.12	0.78	0.66

在 T=3 時，因為選擇權已到期，所以只要股價大於執行價格就會執行，如下表：

Path	T=3	所有權人選擇	Cashflow matrix
1	0.66	不執行	0
2	0.46	不執行	0
3	0.97	執行	0.07
4	1.09	執行	0.19
5	0.48	不執行	0
6	1.1	執行	0.2
7	0.99	執行	0.09
8	0.66	不執行	0

接下來考慮 T=2，首先計算若此時不執行選擇權，將會有多少預期收益。以 T=2 時的股價為 X，此時的現金流矩陣為 Y 進行線性回歸，可得 Y=0.2558X－0.1624，如下表：

Path	T=2 X	T=2 Y	0.2558X-0.1624
1	0.92	0	0.07
2	0.74	0	0.03
3	0.93	0.07	0.08
4	1.03	0.19	0.10
5	0.44	0	0
6	1.23	0.2	0.15
7	1.16	0.09	0.13
8	0.78	0	0.04

比較此值和執行收益的大小，決定每個情況下是否要執行選擇權，若要執行則更新現金流矩陣：

Path	T=2 執行收益	T=2 不執行 預期收益	所有權人選擇	Original cashflow matrix	New cashflow matrix
1	0.02	0.07	不執行	0	0

2	0	0.03	不執行	0	0
3	0.03	0.08	不執行	0.07	0.07
4	0.13	0.10	執行	0.19	0.13
5	0	0	不執行	0	0
6	0.33	0.15	執行	0.2	0.33
7	0.26	0.13	執行	0.09	0.26
8	0	0.04	不執行	0	0

相同的，考慮 T=1 不執行選擇權將會有多少預期收益。進行線性回歸後可得 Y=0.5688X－0.4651，如下表：

Path	T=1 X	T=1 Y	0.5688X- 0.4651
1	0.91	0	0.05
2	0.84	0	0.01
3	0.78	0.07	0
4	1.07	0.13	0.14
5	0.89	0	0.04
6	1.24	0.33	0.24
7	1.08	0.26	0.15
8	1.12	0	0.17

比較此值和執行收益的大小，決定每個情況下是否要執行選擇權，若要執行則更新現金流矩陣：

Path	T=1 執行收益	T=1 不執行 預期收益	所有權人選擇	Original cashflow matrix	New cashflow matrix
1	0.01	0.05	不執行	0	0
2	0	0.01	不執行	0	0
3	0	0	不執行	0.07	0.07
4	0.17	0.14	執行	0.13	0.17
5	0	0.04	不執行	0	0

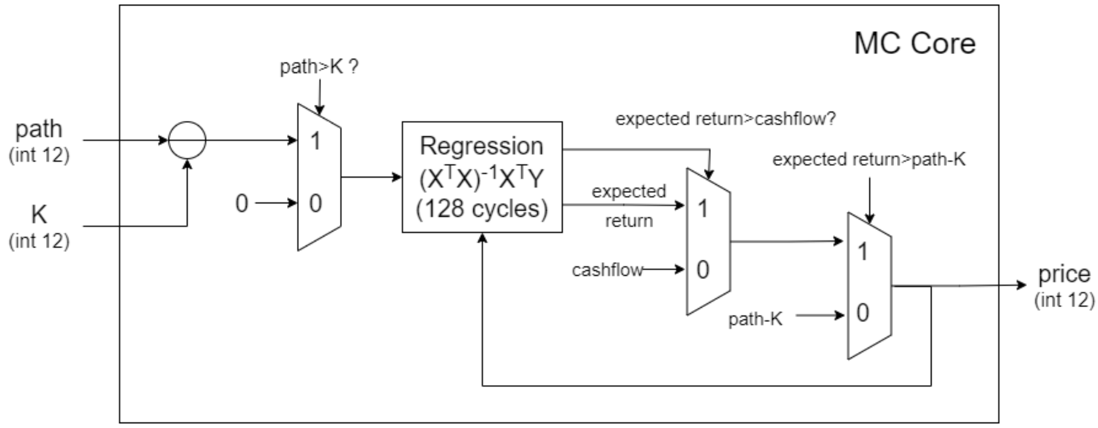
6	0.34	0.24	執行	0.33	0.34
7	0.18	0.15	執行	0.26	0.18
8	0.22	0.17	執行	0	0.22

此時已考慮完 T=1, 2, 3 三個時間點的情況，並算出現金流矩陣(上表最右欄)，將現金留矩陣的所有元素平均得到 0.1225，因此此選擇權的定價為 0.1225 元。

接下來介紹定價的硬體架構。進行定價時需隨時記錄現金流矩陣，而現金流矩陣的每一項皆為 12 bits 的數字，在考量面積的因素後，我們將這 1024 條 path 分成 8 組送進 Monte-Carlo Core 分別進行定價，再將這 8 個定價結果平均以得到更精確的結果。

Monte-Carlo Core 中的架構如圖九所示。首先比較隨機股價和執行價格的大小，若股價比較大，則進入 Regression 模組中進行回歸。回歸的結果為不執行的預計報酬，若此預期報酬比執行後的報酬小，所有權人會選擇執行選擇權，更新現金流值為執行得到的報酬；若此預期報酬比執行價格大，所有權人會選擇不執行選擇權，若預期報酬大於現金流值則更新現金流矩陣。完成 8 次回歸之後輸出現金流矩陣的元素平均值。

每送入 128 個隨機股價會進行一次回歸，算出一組回歸係數，再利用回歸係數算出 128 個預期報酬，更新 128*1 的現金流矩陣。此過程會重複 8 次。



圖九、Monte-Carlo Core 硬體架構

Regression 模組中分成 X^TX、X^TY 和 INVERSE 三個子模組，X^TX 和 X^TY 為矩陣乘法模組，前者為 2*128 和 128*2 的矩陣相乘，後者為 2*128 和 128*1 的矩陣相乘。INVERSE 為 2*2 反矩陣計算模組，必須先算出行列式值的倒數，再乘以各元素：

$$\left(X^TX\right)^{-1} = \frac{1}{nb-a^2}\begin{bmatrix} b & -a \\ -a & n \end{bmatrix}$$

其中

$$a = \sum x_i$$
$$b = \sum x_i^2$$
$$X^T X = \begin{pmatrix} 1 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}$$

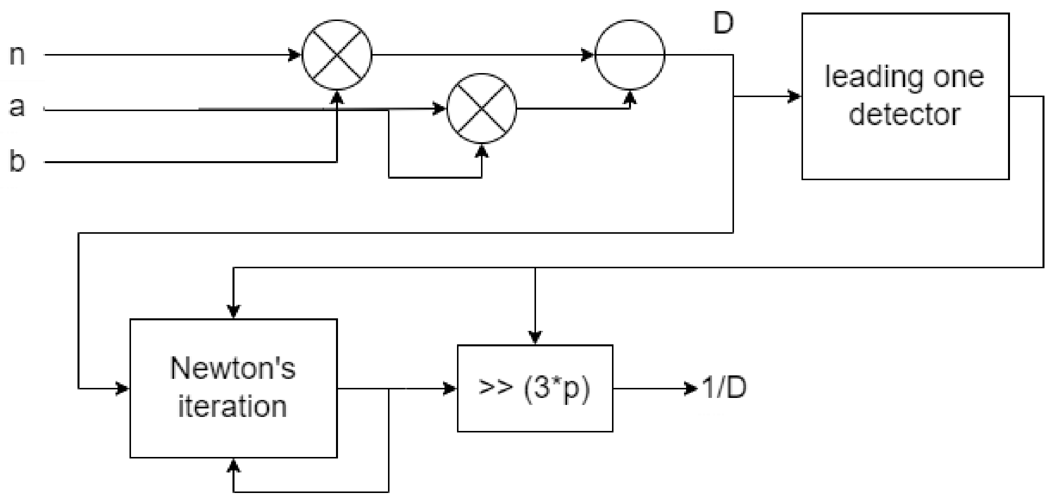
我們使用 Newton-Raphson iteration 進行倒數計算：

$$x_{i+1} = x_i(2 - D \cdot x_i)$$

若數列 x 沒有發散，則在數次迭代後 x 將會收斂至 1/D，而 x_0 的選法是數列是否發散的關鍵。當 x_0 介於區間(0, 2/D)時數列會收斂，因此若先將 D 乘以一個常數 k，使得 k*D 介於區間[0.5, 1)，則選取 $x_0=1$ 必能使數列收斂，且收斂過程為 quadratically converge，實驗過後發現重複 2 次迭代(即算出 x2)即可得到足夠接近的 1/(k*D)，最後再將結果乘以 k 即可得到 1/D。

我們設定 1/k 為比 D 大的最小 2 的次方數，則可以滿足上述要求且硬體實現上只需要進行平移。

行列式值倒數的計算架構如圖十。因為 X^TX 的計算結果為 2*2 對稱矩陣，定義左上角元素為 n，右上角及左下角元素為 a，右下角元素為 b。首先計算矩陣行列式值 D，以 leading one detector 找到比 D 大的最小 2 的次方數，其二進位位數為 p。接下來進行兩次 Newton's iteration 後，將結果向右平移 3*p 位後即可得到 1/D。之所以要平移 3*p 位，是因為 n, a, b 皆為 fixed point variable，在 Newton's iteration 裡計算時會不斷把小數點往左移，因此最後要統一將小數點往右移才可以得到正確的 1/D 值。

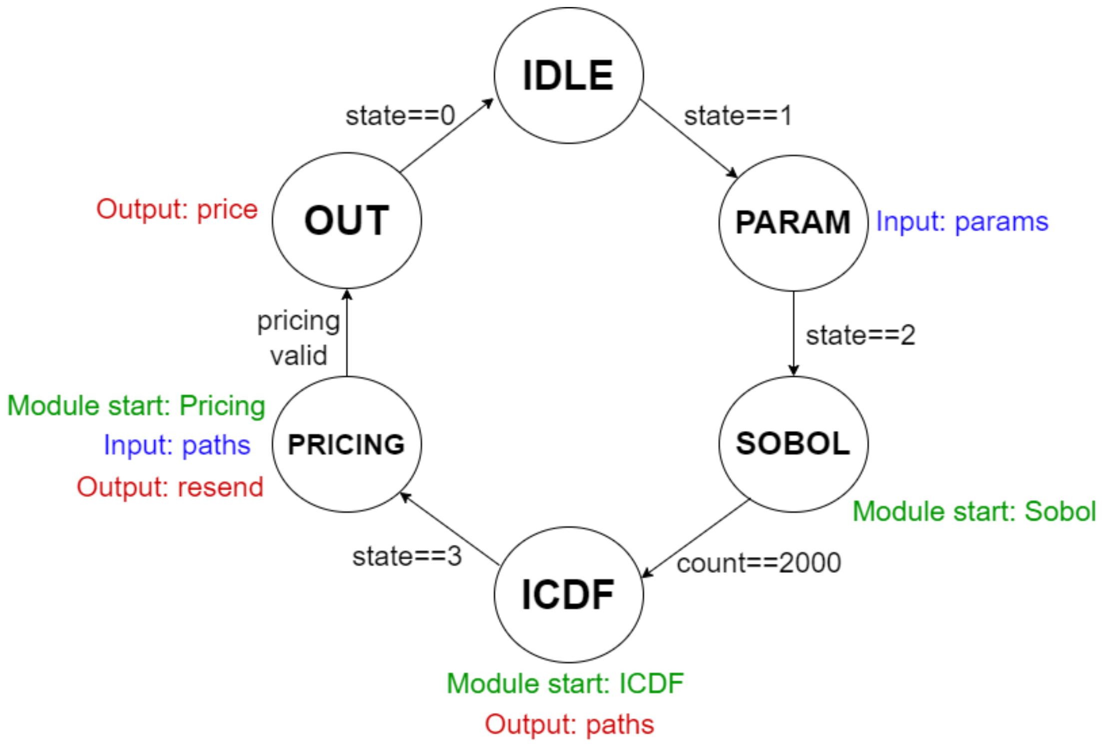


圖十、INVERSE 模組中計算行列式值倒數之架構

將 n, a, b 分別乘以上述之結果並交換順序後，即可得到$(X^TX)^{-1}$。最後將$(X^TX)^{-1}$和 X^TY 相乘，算出之 2*1 矩陣為 b_0 和 b_1 值，也就是回歸係數。

4. Finite State Machine

晶片控制的 FSM 如圖十一所示。開始時輸入 state 為 00，狀態為 IDLE。當輸入 state 為 01 時，開始從晶片外輸入 w, q, S_0, K 等常數，狀態為 PARAM。當輸入 state 為 10 時，啟動 Sobol RNG，狀態為 SOBOL，而 Sobol RNG 前 2000 筆輸出為無效輸出，因此經過 2000 個 cycles 後進入 ICDF 狀態，此時 Sobol RNG 繼續產生亂數，並將這些亂數查表，代入公式後得到 path 並輸出。當輸入 state 為 11 時，啟動定價模組，狀態為 PRICING，此時從晶片外輸入 path，計算回歸係數，並在計算結束後重新輸入 path 以更新現金流矩陣；重複此過程 8 次完成一次定價，可得到一個定價結果，重複 8 次定價並將 8 個定價結果平均後會得到最終定價，此時進入狀態 OUT，將最終定價輸出。



圖十一、晶片整體之 Finite State Machine

5. 硬體實現運算時間

以 python 實現此演算法進行定價時，一次定價需花費 1.84 秒。而在硬體的實現中，我們使用的 cycle time 為 10ns，在一次定價過程，各過程所花費的 cycle 如下：

Sobol Generator：進行初始化花費 2000 cycles

ICDF 及 Path Generator：產生 paths 花費 1024 * 8 = 8192 cycles

Pricing：每次回歸需花費約 128 cycles，更新現金流矩陣花費 128cycles，以上過程重複 8 次完成一次定價，完成 8 次定價共需(128 + 128) * 8 *8 = 16384 cycles

總和為 26576 個 cycles，即 265760 ns =0.00026576 s，與軟體相比大幅減低需花費的時間。

6. 輸入/輸出腳位配置

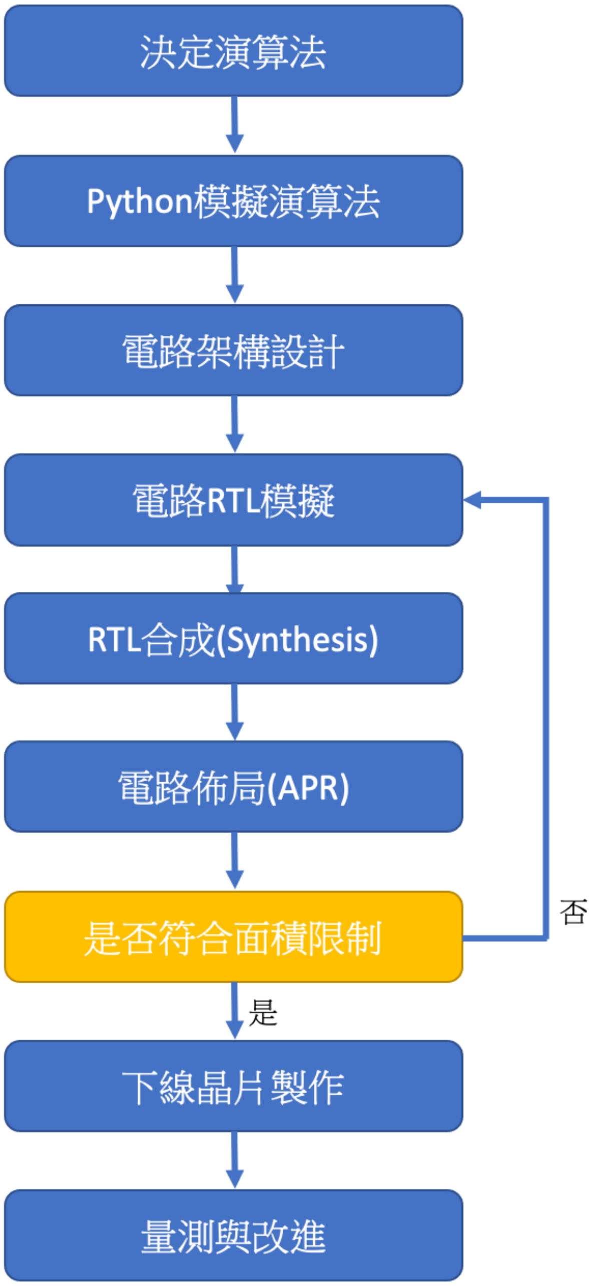
表一為本晶片的輸入及輸出腳位配置。rst_n 為 asynchronous active low；state 控制晶片內部進行哪種運算，當 state 為 00 時為 IDLE，01 時為接收股價相關常數，10 時為進行 path generate 並將 path 傳出晶片，11 時為進行定價並將股價傳出晶片；in 為資料的輸入，股價相關常數及 path 的輸入都由這個腳位輸入；valid 代表晶

片運算結束，當偵測到 valid 為 positive edge 時即可進入下個運算階段；out 為晶片輸出，path 和最後的定價都由這個腳位輸出；而 resend 是代表可重新傳入 path，在定價過程中，計算回歸係數和更新現金流矩陣都需要用到 path，因此 path 必須重複傳入晶片，而 resend 為 positive edge 時則代表要將 path 重新傳入。

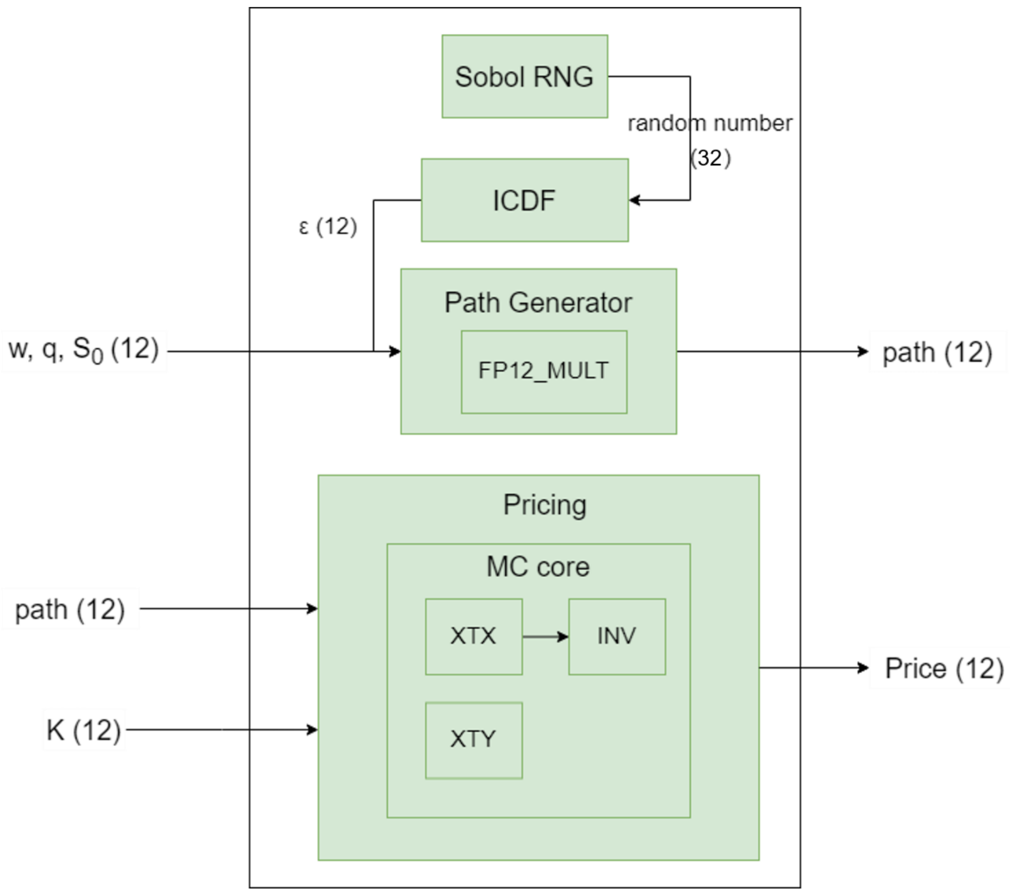
Signal Name	**I/O**	**Width**
clk	I	1
rst_n	I	1
state	I	2
in	I	12
valid	O	1
out	O	12
resend	O	1

表一、輸入及輸出腳位

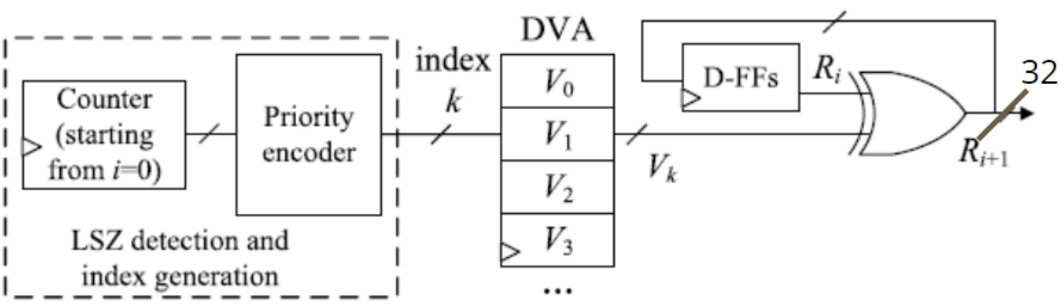
[5] 設計流程


決定演算法
Python模擬演算法
電路架構設計
電路RTL模擬
RTL合成(Synthesis)
電路佈局(APR)
是否符合面積限制
否
是
下線晶片製作
量測與改進

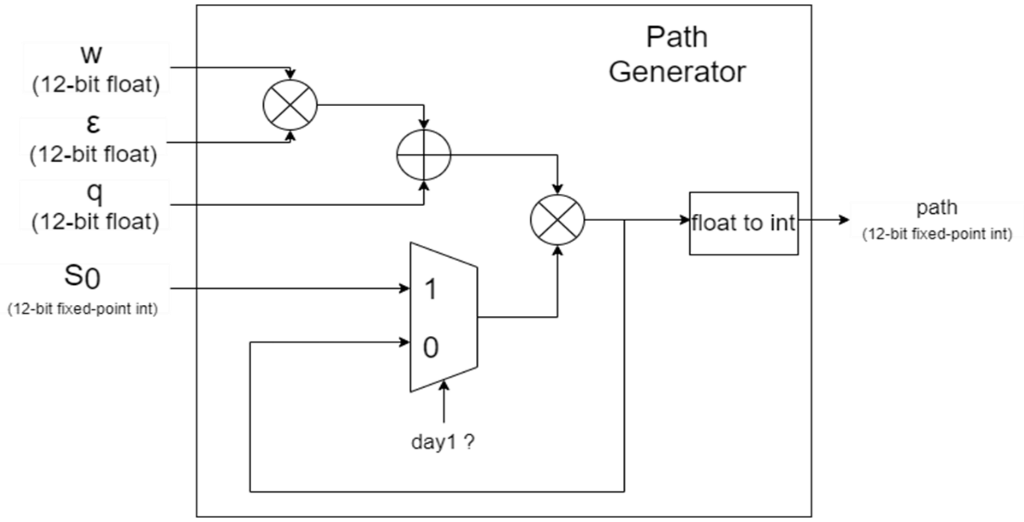
[6] 電路詳圖



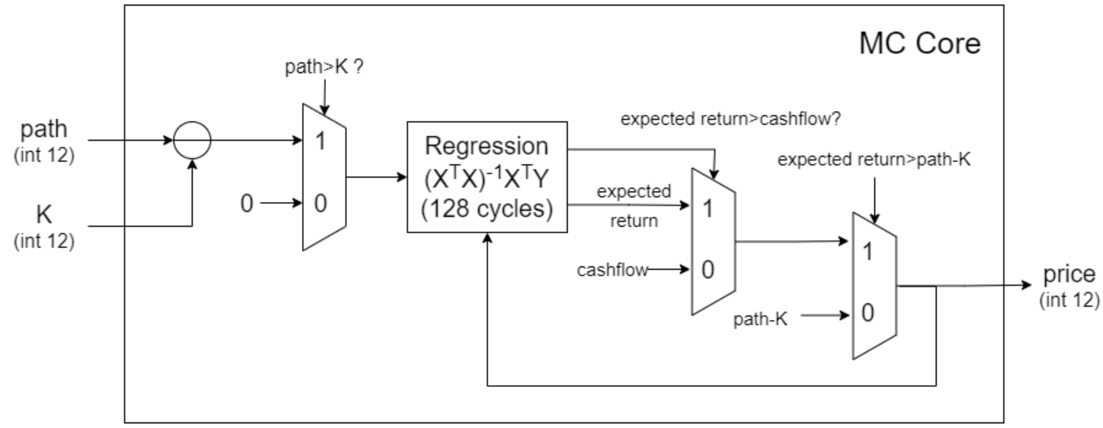
晶片整體架構圖



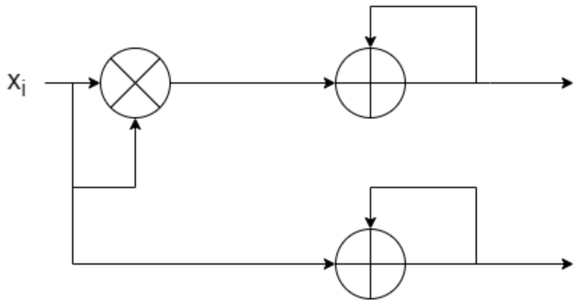
Sobol 亂數產生器架構圖



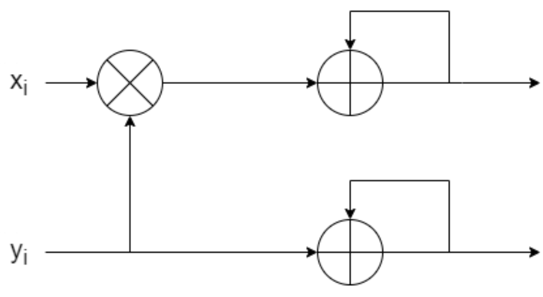
價格路徑產生器架構圖



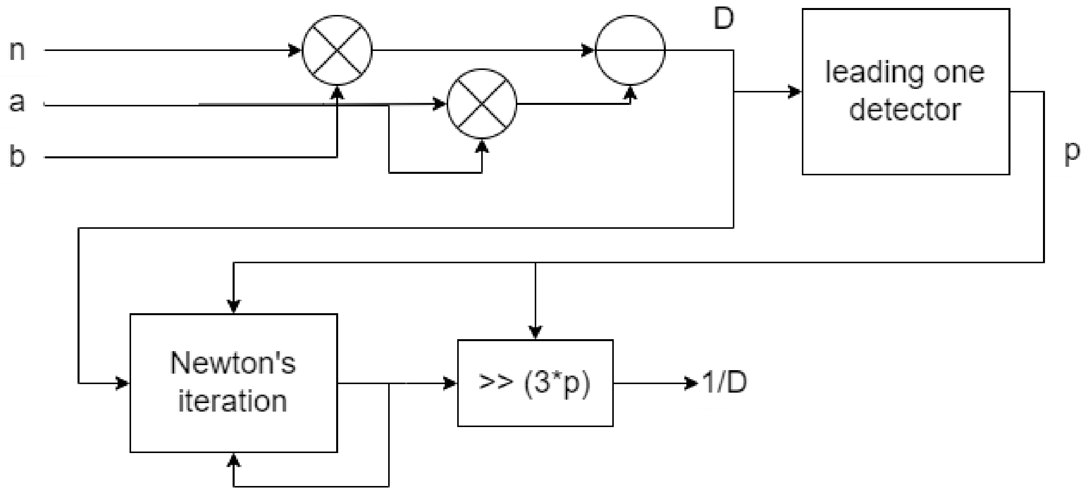
蒙地卡羅法選擇權定價架構圖



XTX 矩陣相乘架構圖(上方輸出為右下角元素，下方輸出為右上及左下角元素)



XTY 矩陣相乘架構圖(上方輸出為第二個元素，下方輸出為第一個元素)

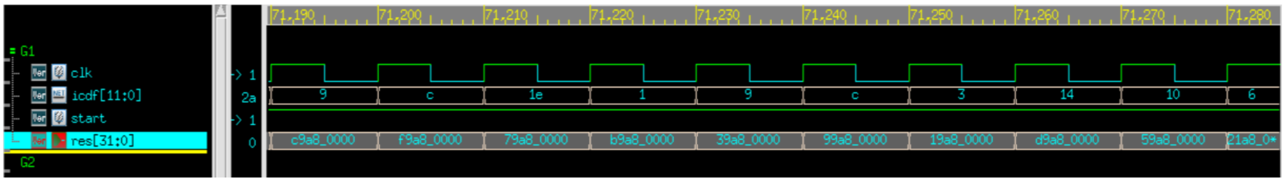


INVERSE 模組中計算行列式值倒數之架構

[7] 模擬結果

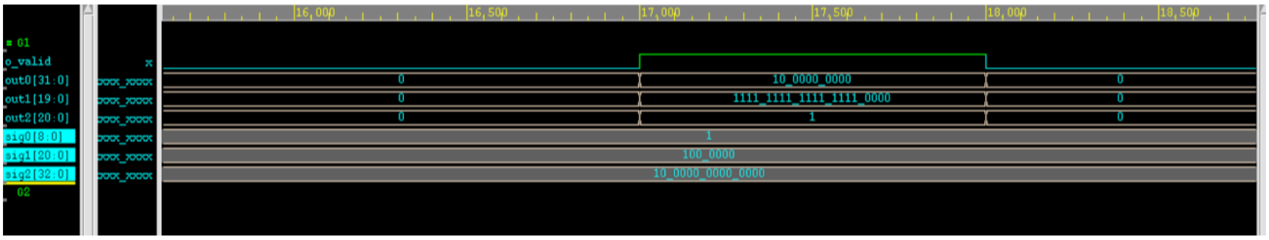
波形圖：(時間單位 ns，週期 10ns)

1. Sobol RNG 和 ICDF



一個 cycle 會輸出 12bit 的 icdf 對照結果，作為 Path Generator 的ε。

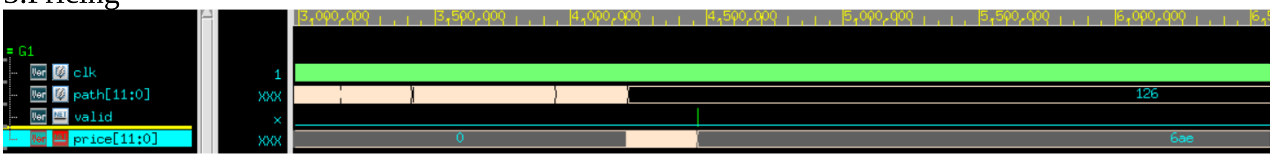
2. INVERSE 模組



此模組計算 2*2 反矩陣，以(整數，小數)fixed point 表達數字，三個輸入分別為(9，0)、(17，4)、(25，8)，輸出分別為(24，8)、(14，6)、(17，4)。

圖中測資輸入之 2*2 矩陣為$[1\ 4\ 4\ 32]$，輸出之 2*2 反矩陣為$\left[2\ \frac{-1}{4}\ \frac{-1}{4}\ \frac{1}{16}\right]$。

3.Pricing



此模組輸入 path，輸出選擇權之定價。

[8] 量測考量

預計量測流程

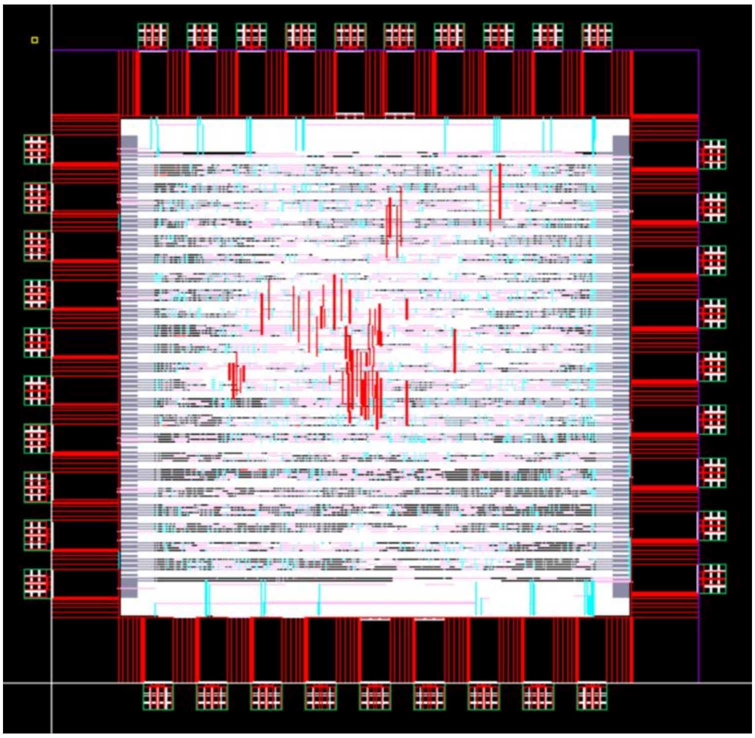
1. 將現有的測試資料進行調整，使其能作為測試晶片之輸入
2. 儀器設置：電源供應器調整為 3.3V 直流電，並接上晶片的電源腳位
3. 儀器設置：將訊號產生器接至晶片的輸入腳位，並產生 100MHz 的方波，輸入 clk 腳位
4. 儀器設置：將邏輯分析儀接至晶片的輸出腳位
5. 使用訊號產生器向晶片輸入計算美式選擇權定價的參數(第一階段所需的測試資料)
6. 將晶片輸出之選擇權價格路徑與軟體生成之資料比對，確認第一階段結果是否正確
7. 使用訊號產生器向晶片輸入擇權價格路徑(第一階段所生成的輸出資料)
8. 將晶片輸出之選擇權定價與原有軟體生成之資料比對，確認晶片結果是否正確

[9] 佈局驗證結果錯誤說明

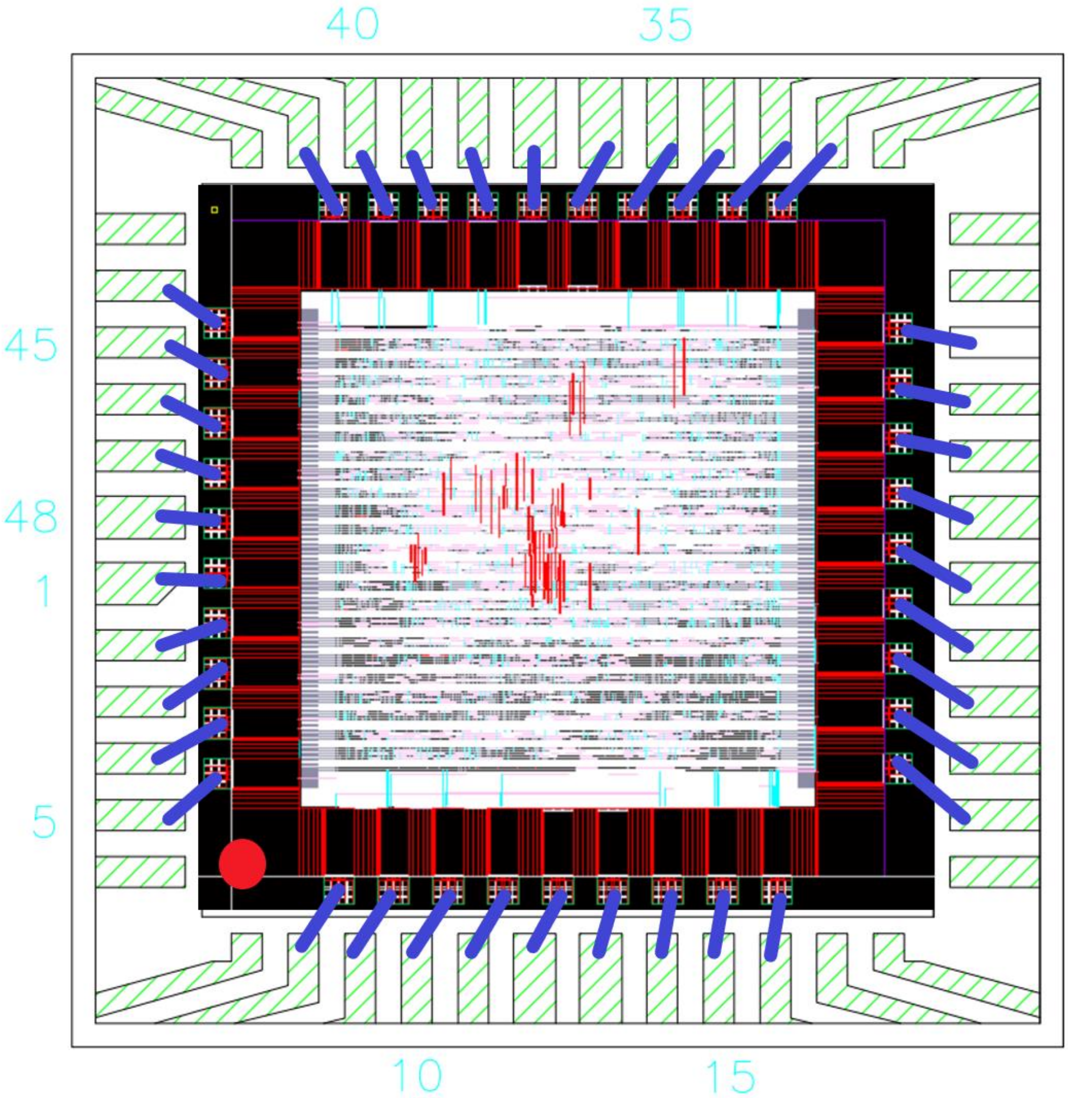
不對外公開而在此省略

[10] 佈局平面圖

Chip Size: 1463.58x1464.00μm^2
Power Dissipation: 18.49mW
Max. Frequency: 100MHz



[11] 打線圖



[12] 預計規格列表

Specification	Spec.	Pre-sim(tt)	Post-sim(tt)
Supply Voltage	1.8V	1.8V	
Frequency	100MHz	100MHz	
Chip size (μm^2)	< 1500x1500	1500x1500	
Power	-	23.9mW	
PADs	40	40	

[13] 參考文獻

[1] X. Tian and K. Benkrid, "American option pricing on reconfigurable hardware using Least-Squares Monte Carlo method," 2009 International Conference on Field-Programmable Technology, 2009, pp. 263-270, doi: 10.1109/FPT.2009.5377662.

[2] I. L. Dalal, D. Stefan and J. Harwayne-Gidansky, "Low discrepancy sequences for Monte Carlo simulations on reconfigurable platforms," 2008 International Conference on Application-Specific Systems, Architectures and Processors, 2008, pp. 108-113, doi: 10.1109/ASAP.2008.4580163.

[3] S. Liu and J. Han, "Toward Energy-Efficient Stochastic Circuits Using Parallel Sobol Sequences," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 7, pp. 1326-1339, July 2018, doi: 10.1109/TVLSI.2018.2812214.