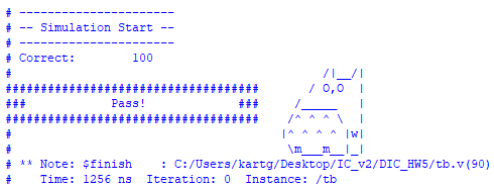
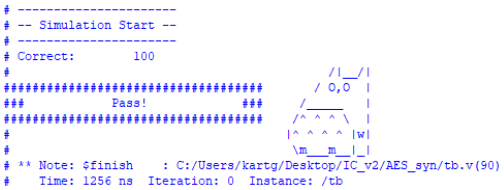
2024 Digital IC Design Homework 5

NAME	黃偉峰
Student ID	E34106010
Score = area*timing (ps)	24571128000
Cycle time (ns)	10 ns

Simulation Result

Functional simulation	Completed	Gate-level simulation	Completed





Description of your design

實做部分較複雜我直接寫在下方

Flow Status	Successful - Fri Jun 14 22:04:06 2024
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	AES
Top-level Entity Name	AES
Family	Cyclone IV E
Device	EP4CE75F29C8
Timing Models	Final
Total logic elements	19,563 / 75,408 (26 %)
Total registers	5254
Total pins	387 / 427 (91 %)
Total virtual pins	0
Total memory bits	0 / 2,810,880 (0 %)
Embedded Multiplier 9-bit elements	0 / 400 (0 %)
Total PLLs	0 / 4 (0 %)

The scoring standard: (The smaller, the better)

Scoring =

*Area cost * Timing cost*

Area cost =

*Total logic elements + total memory bits + 9*embedded multiplier 9-bit elements*

Timing cost =

Simulation time

實作介紹在下一頁

1. AES Inner and Outer Pipeline Implementation

在實現高效的 AES 加密過程中，此次 Project 我使用 Inner/Outer pipeline 作為設計。Inner Pipeline 分割了每一輪中的 SubByte 和其他操作，並在其之間加入 Pipeline register，而 Outer Pipeline 則在各個輪次之間添加 Pipeline register，以進一步提高處理效率。Outer Pipeline 的部分參考自[1]，並在下面圖片描述了這兩部分的實現方法。

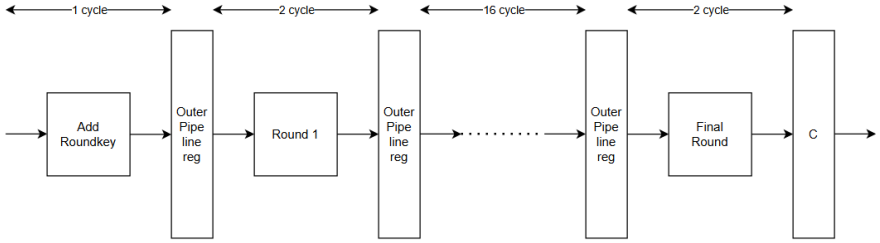


Figure1. Outer Pipeline implementation

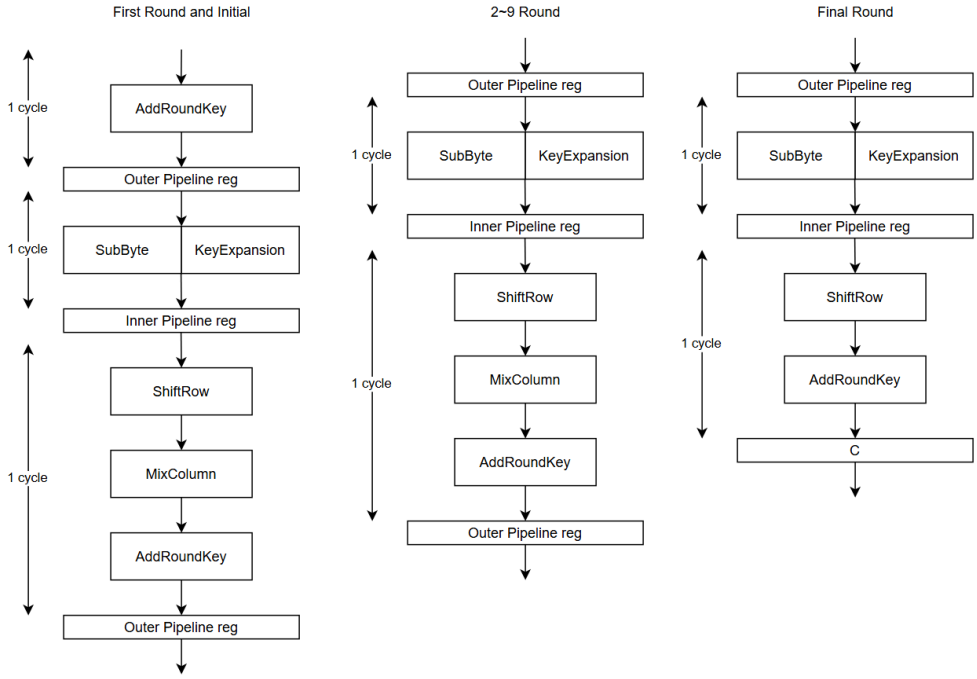


Figure1. Inner Pipeline implementation

在 AES 當中 **SubByte 通常具有最長的 Delay 時間**，因此在每輪的內部設計時，透過將 Register 加在 SubByte 之後能夠有效減短 Critical Path，並在每輪結束時都加入 Register 達成 Outer Pipeline 能夠更進一步加快電路速度並且設計上也更模組化更簡單。

2. AES Inner Module Modify

在 AES 加密算法中，AddRoundKey、SubByte、ShiftRow、MixColumn 和 KeyExpansion 是其主要的模組。其中，SubByte 和 KeyExpansion 都使用到了 S-Box 部分。一般實現中，S-Box 通常使用 Lookup Table 的方式，但這在硬體電路中會耗費大量的 MUX。因此，在此次專案中，我選擇了基於 S-Box 底層數學計算的方法來實現。而 MixColumn 部分涉及 GF(2^8)的乘法，因為乘的皆為固定數字，所以我找出常數矩陣將 GF(2^8)的乘法轉換成單純的 XOR 邏輯運算。

i) MixColumn

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

在 MixColumn 中我們會在 GF(2^8)下做矩陣乘法，並且可以看到乘的數值固定為 1、2、3，故我們只要找出這三個常數所代表的矩陣就可以將乘法變成 XOR 運算，以下是一個找出常數矩陣 3 的例子:

Constant matrix:

$$original = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

$$3 = x + 1$$

$$a{\cdot}3 = (a_7x^7+a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0){\cdot}(x+1) \mod (x^8+x^4+x^3+x+1)$$

$$\begin{aligned} result = {} & (a7 + a6)x^7 + (a6 + a5)x^6 + (a5 + a4)x^5 + (a7 + a4 + a3)x^4 \\ & +(a7 + a3 + a2)x^3 + (a2 + a1)x^2 + (a7 + a1 + a0)x + (a7 + a0) \end{aligned}$$

我們只要將 x 的各個次方的各項係數表示成如下圖的矩陣即可，並且依此類推可繼續找到 1 和 2 的常數矩陣。

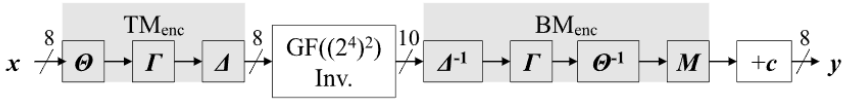
$$
\begin{array}{ll}
10000001 & a_0 \\
11000001 & a_1 \\
01100000 & a_2 \\
00110001 & a_3 \\
00011001 & a_4 \\
00001100 & a_5 \\
00000110 & a_6 \\
00000011 & a_7
\end{array}
$$

最後將矩陣根據乘上 2、3、1、1 的順序將常數矩陣加起來即可。

ii) S-BOX

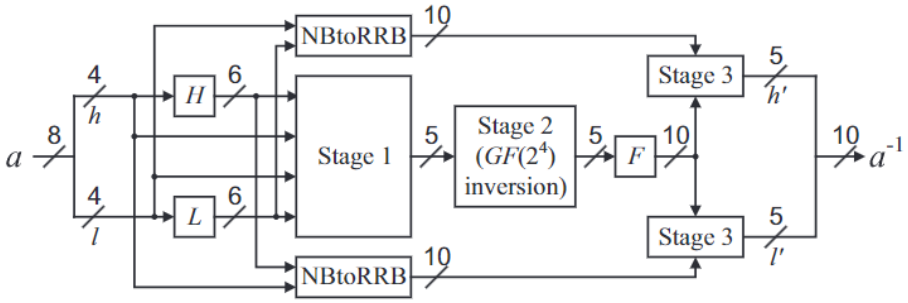
在 S-Box 的實現中，有兩種主要的方法。一種是簡單的方法，即使用 MUX 構建 LUT；另一種則是計算輸入值的 multiplicative inverse 並進行同構映射 isomorphic mapping。在此次 Project 中，我選擇了後者來實現 S-Box。計算乘法反元素是一個相對複雜的操作，為此我參考了[2]和[3]的方法。在此過程中，我對輸入值進行了特定的乘法和指數偏移映射（multiplicative 和 exponential offsets）。這種映射可以降低計算矩陣的 Hamming weight。較低的 Hamming weight 意味著需要更少的 XOR 操作，從而使得 S-Box 所需的邏輯閘數量更少，節省了硬體面積。此外，降低了 maximum Hamming weight 意味著縮短 critical path 的長度，這使得電路的運算速度更快。總結來說，這種映射技術不僅減少了所需的邏輯閘數量和佔用的面積，還提高了運算速度，使得 S-Box 在高效能和資源受限的應用場景中表現出色。

- **Linear Mapping:**此改善參考自[2]，在 S-BOX 當中涵蓋了 Multiplicative inverse、Affine transformation、Isomorphic mapping，而傳統上 Affine transformation 的矩陣有著較高的 Hamming weight，在矩陣中的每個 1 就代表電路中的一個 XOR，故該方法透過 multiplicative offsets/exponential offsets 將轉換所需要用到的矩陣能有較低的 Hamming weight，並且也有較小的 maximum hamming weight。



根據作者所提供的較優解，我選擇使用 Exponential offsets = 1，Multiplicative offsets = 79。

- **Multiplicative inverse**:該算法參考自[3]，當中有清楚寫出流程及相關算式可以直接照著刻電路即可。



上圖為 GF(2^8)inversion 的結構圖，分為三個主要階段，每個階段的具體實現方法如下：

(a) Stage 1:計算輸入的 16 次方和 17 次方。Normal Basis 來簡化這些計算。該階段的輸入為 GF(2^8)中的元素，並將其轉換到 GF((2^4)^2)域中進行操作。 圖中 H 和 L 分別高位和低位，通過 NBtoRRB 轉換成 Redundantly Represented Basis 表示。

(b) Stage2:進行 GF(2^4)的 inversion 計算。使用 Polynomial Ring Representation 表示進行 GF(2^4) inversion，利用該表示法可以提高效率。

(c) Stage3:執行 GF(2^4)的乘法操作。 使用 RRB 進行 GF(2^4)乘法，因為 RRB 在 GF(2^4)乘法中具有較高的計算效率。

3. Reference

[1]“Pipelined implementation of AES encryption based on FPGA,” IEEE Conference Publication | IEEE Xplore, Dec. 01, 2010. Available: https://ieeexplore.ieee.org/document/5688757/citations#citations

[2]“AES S-Box hardware with efficiency improvement based on linear mapping optimization,” Tohoku University. Available: https://tohoku.elsevierpure.com/en/publications/aes-s-box-hardware-with-efficiency-improvement-based-on-linear-ma/fingerprints/ R. Ueno, N. Homma, Y. Sugawara, Y. Nogami, and T. Aoki,

[3]“Highly efficient GF(2^8) inversion circuit based on redundant GF

arithmetic and its application to AES design," in Lecture notes in computer science, 2015, pp. 63–80. doi: 10.1007/978-3-662-48324-4_4. Available: https://www.researchgate.net/publication/285601676_Highly_Efficient_GF2^8_Inversion_Circuit_Based_on_Redundant_GF_Arithmetic_and_Its_Application_to_AES_Design