FPGA\_Design final project AES-GCM

E24041810 E24046755 E24046307

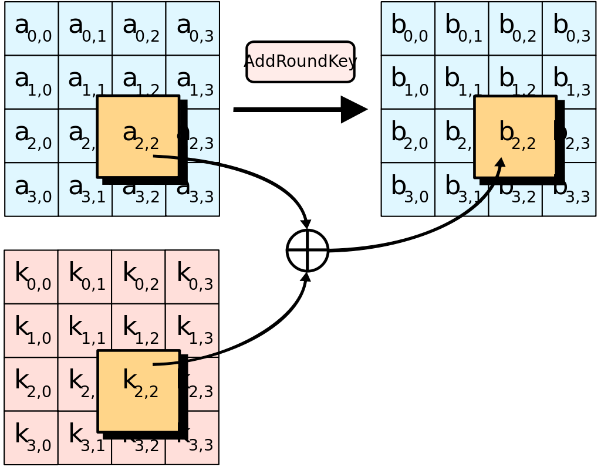
1. AES-GCM 介紹
   1. AES-GCM的名詞解釋：

AES-GCM分為兩個部分：AES與GCM

* + 1. **AES**：即進階加密標準(**A**dvanced **E**ncryption **S**tandard)，相比於上一個世代的DES，AES具有較長的金鑰與運算函式所以具有較高的保密性，可大致分為4個步驟：AddRoundKey、SubBytes、ShiftRows與MixColumns，在稍後會有較為詳細的說明
    2. **GCM**：Galois/Counter Mode，為AES的其中一種加密方式，分為GMAC(Galois message authentication code mode)與CTR Mode(Counter Mode)兩個部分，之後會分別解說
  1. AES的四個步驟

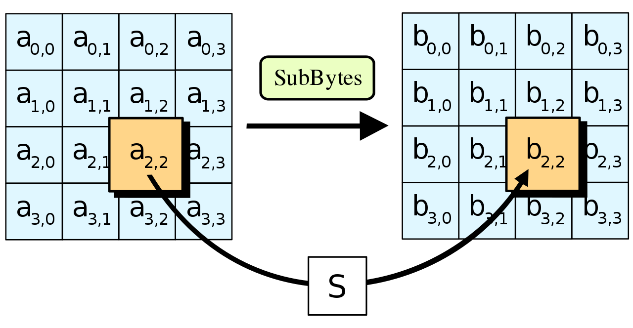
在進行AES的運算之前，我們必須先把讀進去的資料每8bits分為一個單位，而每次讀取16單位也就是128bits的資料進行運算

* + 1. AddRoundKey：



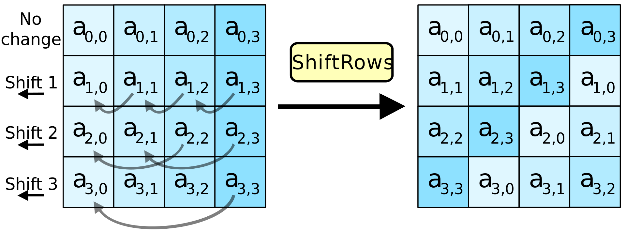
我們把資料視為4\*4的矩陣，在做運算時每筆資料與該次的key做XOR運算產生一個新的4\*4矩陣

* + 1. SubBytes：



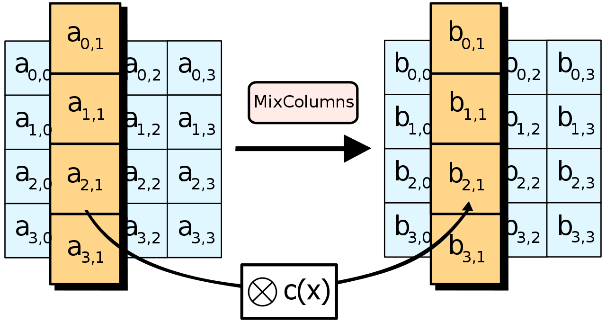
在這個步驟，我們將上一個步驟所產生的矩陣資料一筆筆的讀出來，並透過S-box運算(一種查表的方式，有興趣可以上維基百科搜尋”S-box”)，產生新的矩陣。

* + 1. ShiftRows：



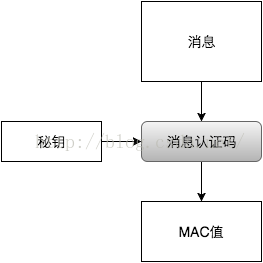
如圖所示，將不同列的資料依序向左平移，第一列不移動，第二列平移一格，依此類推

* + 1. MixColumns：

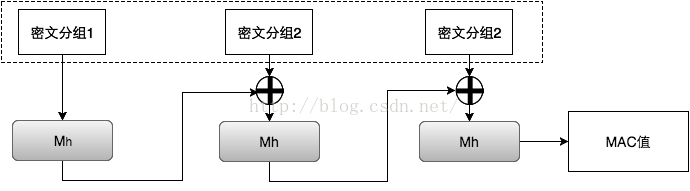


在MixColumns步驟，每一列的四個位元組透過線性變換互相結合。每一列的四個元素分別當作1 , x , x^2 , x^3的係數，接著將此多項式和一個固定的多項式 c(x)=3x^3+x^2+x+2相乘。ShiftRows和MixColumns兩步驟為這個密碼系統提供了擴散性。(擴散性：當輸入值改變時，最終結果有相當大的變化。在資料遭到竄改時，容易發現異狀)

* 1. MAC與GMAC
     1. MAC(消息驗證碼)：要校驗消息的完整性，必須引入另一個概念：消息驗證碼。消息驗證碼是一種與秘鑰相關的單項散列函數。密文的收發雙發需要提前共享一個秘鑰。密文發送者將密文的MAC值隨密文一起發送，密文接收者通過共享秘鑰計算收到密文的MAC值，這樣就可以對收到的密文做完整性校驗。當篡改者篡改密文後，沒有共享秘鑰，就無法計算出篡改後的密文的MAC值。

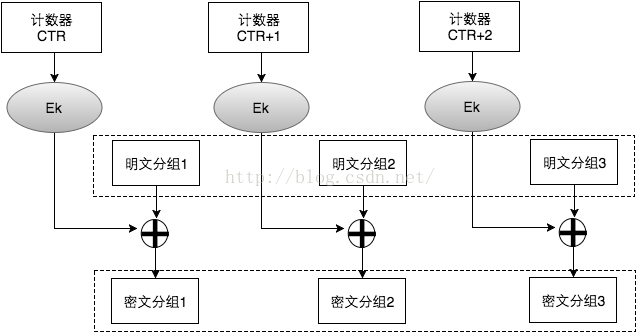


* + 1. **GMAC**：就是利用伽羅華域(Galois Field，GF，有限域)乘法運算來計算消息的MAC值。假設秘鑰長度為128bits, 當密文大於128bits時，需要將密文按128bits進行分組



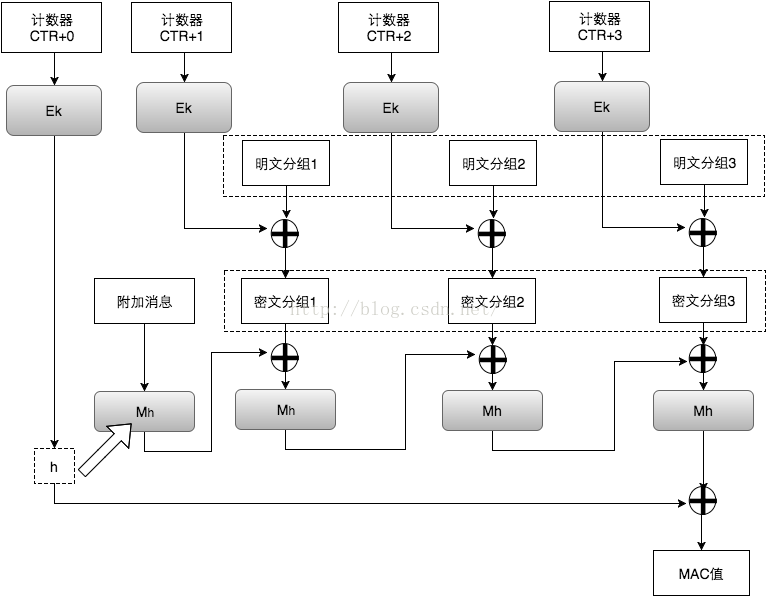
* 1. CTR Mode：

CTR模式（Counter mode，CM）也被稱為ICM模式（Integer Counter Mode，整數計數模式）和SIC模式（Segmented Integer Counter）。它通過遞增一個加密計數器以產生連續的金鑰流，其中，計數器可以是任意保證長時間不產生重複輸出的函式，但使用一個普通的計數器是最簡單和最常見的做法。使用簡單的、定義好的輸入函式是有爭議的：批評者認為它「有意的將密碼系統暴露在已知的、系統的輸入會造成不必要的風險」



* 1. GCM Mode

GCM中的G就是指GMAC，C就是指CTR。GCM可以提供對消息的加密和完整性校驗，另外，它還可以提供附加消息的完整性校驗。在實際應用場景中，有些信息是我們不需要保密，但信息的接收者需要確認它的真實性的，例如源IP，源端口，目的IP，IV，等等。因此，我們可以將這一部分作為附加消息加入到MAC值的計算當中。下圖的EK表示用對稱秘鑰ķ對輸入做AES運算。最後，密文接收者會收到密文、IV（計數器CTR的初始值）、MAC值，並可利用下圖中由下而上的方式倒推得到明文。



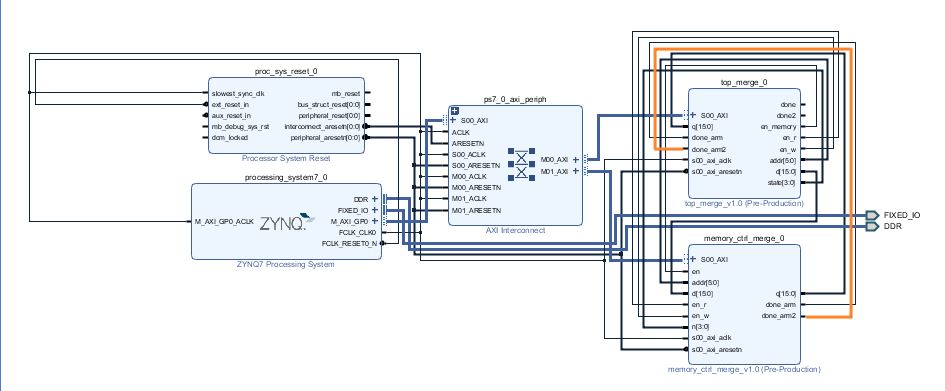
* 1. 參考資料
     1. 什么是AES-GCM加密算法- T0mato\_的博客- CSDN博客

<https://blog.csdn.net/T0mato_/article/details/53160772>

* + 1. Wikipedia 高級加密標準

https://zh.wikipedia.org/wiki/%E9%AB%98%E7%BA%A7%E5%8A%A0%E5%AF%86%E6%A0%87%E5%87%86

1. Block diagram:



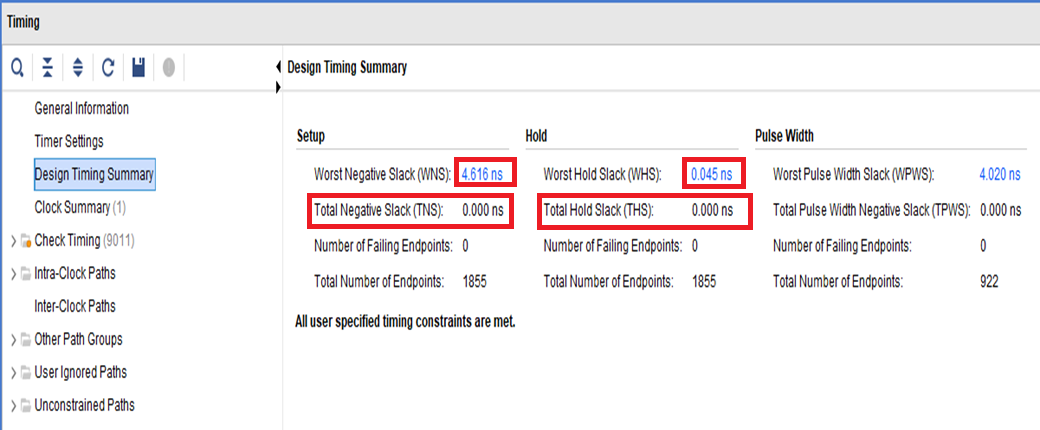
1. 設計說明:
2. 金鑰長度為16 bytes。
3. plaintext大小為64 bytes，由SDK寫進memory。
4. top這個IP中，包含了6個module，分別為controller、AES、XOR1、GF、XOR2、top。以加密來說:
5. **controller**用來控制address、enable訊號線等等。
6. **AES**用來對counter做AES加密。
7. **XOR1**用來對plaintext以及AES加密後的counter來做XOR，即產生密文。
8. **GF**用來做GF乘法運算。
9. **XOR2**用來對密文以及GF乘法結果來做XOR。
10. **top**用來將上述五個module接線接在一起。
11. memory\_ctrl這個IP中，包含了2個module，分別為memory、memory\_ctrl。
12. **memory**一開始儲存plaintext，加密後的ciphertext亦會存回此memory，而最後解密回去後，也會將結果存回此memory。
13. **memory\_ctrl**用來聯絡memory以及SDK，使我們可以從SDK讀寫memory。
14. 此設計為做完加密後，done這個訊號線會拉成1，以便從SDK讀出加密後的結果；接著便做解密，若解密完成，done2這個訊號線會拉成1，以便從SDK讀出解密後的結果。
15. 共測了2筆data，皆與軟體模擬(C++)的結果一致。
16. 實作結果:
17. 加密後的結果以及MAC值(測試兩筆不同的data):

|  |  |
| --- | --- |
|  |  |
| Test 1 | Test 2 |

1. 解密後的結果以及MAC值(測試兩筆不同的data):

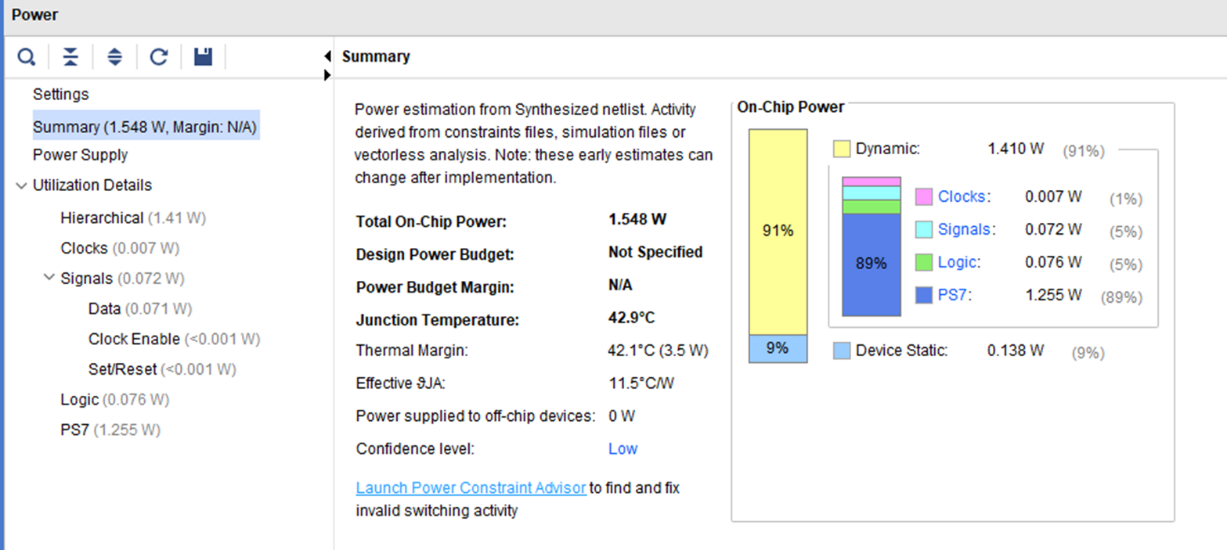
|  |  |
| --- | --- |
|  |  |
| Test 1 | Test 2 |

1. Report
2. Timing Report



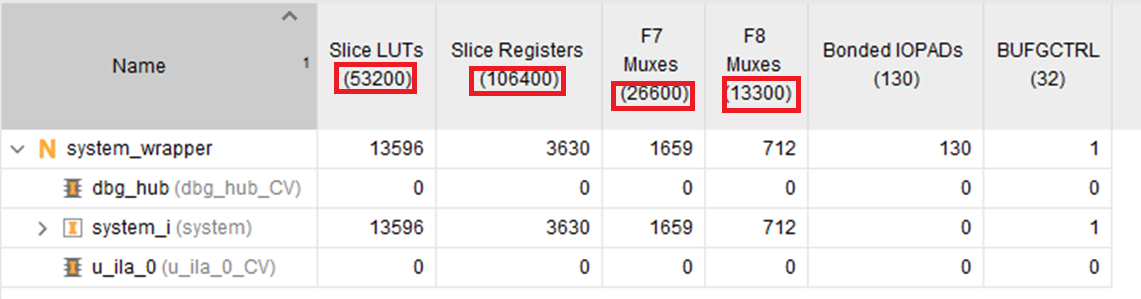
Setup 和 Hold Slack都是正的，且total negative slack是0，所以符合這個操作頻率下的timing需求，沒有timing violation的問題。

1. Power Report

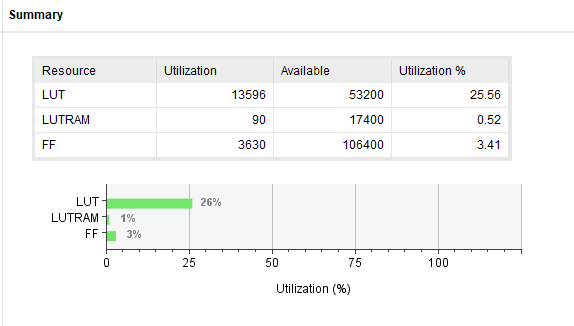


這是power消耗的情形。

1. Utilization Report



Pynq-z2有13300個slices，一個slice有包含4個6 input的LUT，以及8個Register，所以紅色框框的部分是板子擁有的資源數量(F7是7 input mux，F8是8 input mux)。



上圖是LUT和Flip Flops使用數量以及佔的百分比，其中LUT RAM是指LUT作為Distributed RAM。整體來看AES使用FPGA資源情形的話，LUT用了26%，這26%可以分為25%是LUT作為logic用，而1%是作Memory使用，Slice Register則是用了3%。