# 專案(二)

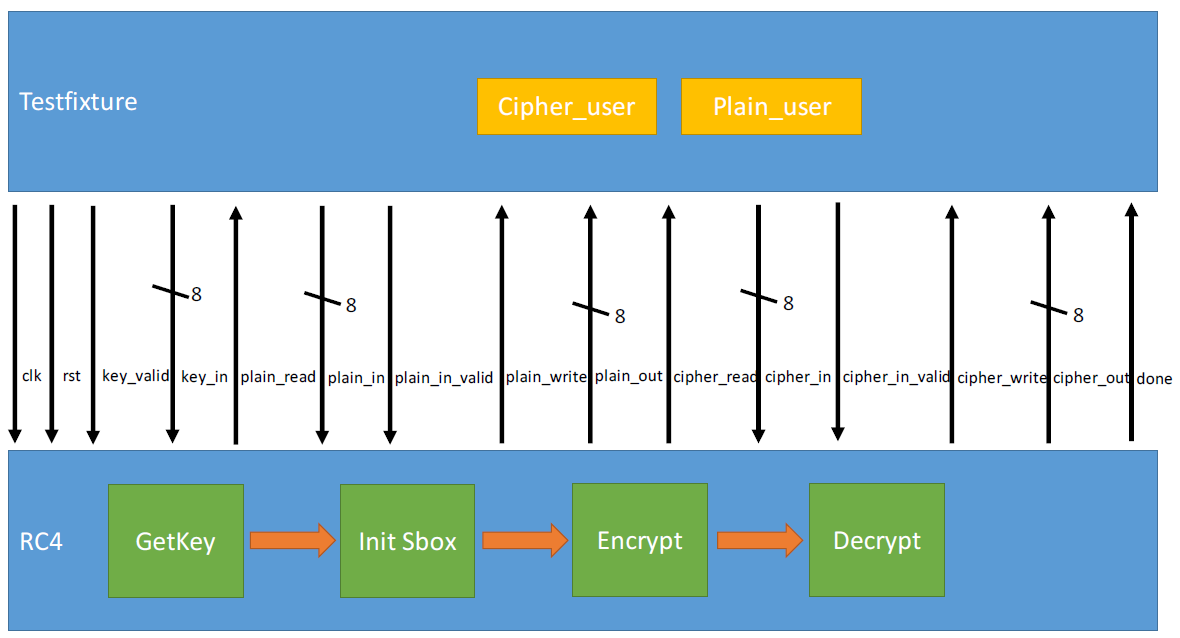
## 專案名稱 - Rivest Cipher 4加解密電路實現

## 簡介

在密碼學中，RC4（來自 Rivest Cipher 4的縮寫）是一種流加密算法，密鑰長度可變。它加解密使用相同的密鑰，因此也屬於對稱加密算法。RC4是有線等效加密（WEP）中採用的加密算法，也曾經是TLS可採用的算法之一。由於 RC4算法存在弱點，2015年 2月所發布的RFC7465規定禁止在TLS中使用 RC4加密算法。RC4由偽隨機數生成器和異或運算組成。~~RC4的密鑰長度可變，範圍是[1,255]~~。RC4一個字節一個字節地加解密。給定一個密鑰，偽隨機數生成器接受密鑰並產生一個Sbox。 Sbox用來加密數據，而且在加密過程中Sbox會變化。由於異或運算的對合性，RC4加密解密使用同一套算法。此次作業請實作一個RC4加解密的電路，其中密鑰的長度固定為32bytes，明文的長度為不固定，最長長度不超過2048bytes，Sbox大小為64bytes利用輸入金鑰對明文進行加密，然後將加密完的字元輸出，再將所輸出加密的字元輸入，進行解密，還原出原本的明文。

## 設計規格

### Block overview

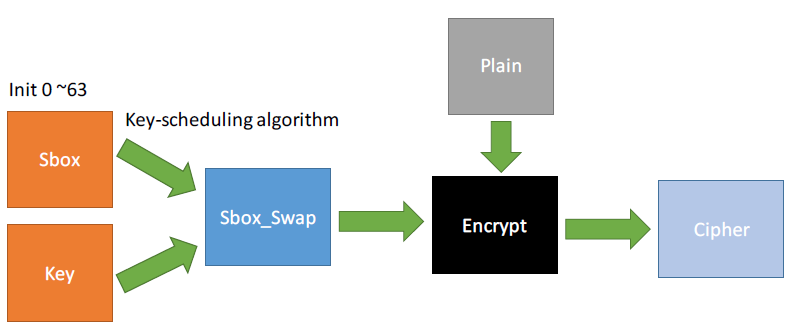


圖一、系統方塊圖

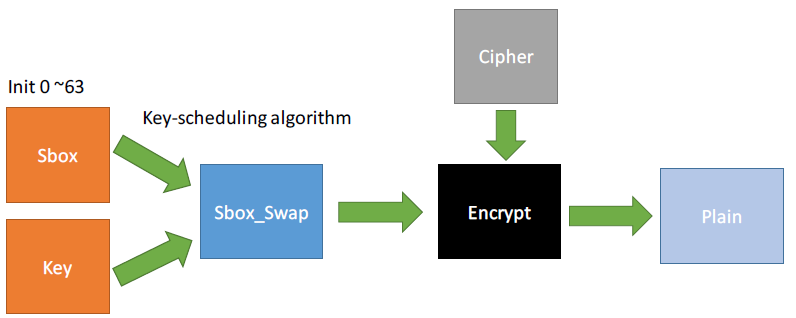
### I/O Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Name | I/O | Width | Description |
| clk | I | 1 | 系統時脈訊號。本系統為同步於時脈正緣之同步設計 |
| rst | I | 1 | 高位準"非"同步(active high asynchronous)之系統重置信號 |
| key\_valid | I | 1 | 當key準備好時，會先將key\_valid設成high，然後在下一個負緣輸出key值。 |
| key\_in | I | 8 | key data輸入訊號線，輸入的data size為8 bits，在key\_valid設成high的下一個cycle的負緣輸出。 |
| plain\_read | O | 1 | 當要索取明文時請將plain\_read設成high且plain\_write設成low，再下一個cycle後會將其值輸入。 |
| plain\_in | I | 8 | 輸入明文資料訊號，由8bits 整數組成，為無號數 |
| plain\_in\_valid | I | 1 | 因為明文長度不固定，所以當輸入的明文為有效時，plain\_in\_valid為high，若無效時plain\_in\_valid為low。 |
| plain\_write | O | 1 | 若要將解密後的明文輸出至testfixture記憶體，將plain\_write設為high，其他不須寫入時務必設成low。 |
| plain\_out | O | 8 | 解密後運算結果記憶體寫出訊號，由8bits 整數(MSB)組成，為無號數。 |
| cipher\_read | O | 1 | 當要索取密文時請將cipher\_read設成high且cipher\_write設成low，再下一個cycle後會將其值輸入。 |
| cipher\_in | I | 8 | 輸入密文資料訊號，由8bits 整數組成，為無號數 |
| cipher\_in\_valid | I | 1 | 因為密文長度不固定，所以當輸入的密文為有效時，cipher\_in\_valid為high，若無效時cipher\_in\_valid為low。 |
| cipher\_write | O | 1 | 若要將加密後的密文輸出至tb記憶體，將cipher\_write設為high，其他不須寫入時務必設成low。 |
| cipher\_out | O | 8 | 加密後運算結果記憶體寫出訊號，由8bits 整數(MSB)組成，為無號數。 |
| done | O | 1 | 如果系統的運算結束，將done訊號輸出 |

### Function Description



圖一、加密流程圖



圖二、解密流程圖

for i from 0 to 63:

sbox[i]=i;

k=0;

for j from 0 to 63:

k=(k+sbox[j]+key\_men[j % 32]) % 64

swap values of sbox[j] and sbox[k]

圖三、Key-scheduling algorithm (KSA)

a=0

b=0

while GeneratingOutput:

a = (a+1) % 64

b = (b+sbox[a])%64

swap values of sbox[a] and sbox[b]

c = inputByte^sbox[(sbox[a]+sbox[b]) % 64]

output c

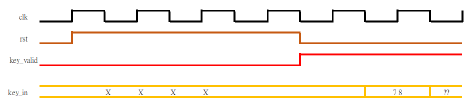
endwhile

圖三、加解密演算法

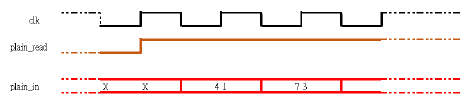
本系統的key長度為32bytes，而明文的長度為不固定，最長為2048bytes，key的資料儲存於testfixture中，在系統進行reset之後，下一個cycle 會先輸出key\_valid= high然後再過一個cycle後輸出key的值，當key\_valid為high(除了第一個cycle)時代表key 值有效，當key 值輸入完畢後，同學需先將key跟Sbox進行打亂，Sbox一開始為0~63，利用圖四的Pseudo code進行打亂後，再利用打亂後的Sbox進行加密，加密演算法如圖五的Pseudo code，當加密完成後的密文請利用cipher\_write和cipher\_out將其結果輸出至testfixture的記憶體中，當plain\_in\_valid等於high時代表明文為有效輸入，當plain\_in\_valid為low時代表明文輸入完畢，當明文輸入完畢後，方可藉由cipher\_read來控制密文的輸入（注意：若明文加密後有錯，輸入的密文也是錯誤的)，當cipher\_in\_valid 等於high時代表密文為有效輸入，當cipher\_in\_valid為low時代表為密文輸入完畢，解密的演算法流程與加密相同，若系統已經將加解密動作完成時，請將done 設為high，即可驗證加密。

圖六為key\_in輸入的時序圖，key\_valid為high後下一個clk cycle 便將key

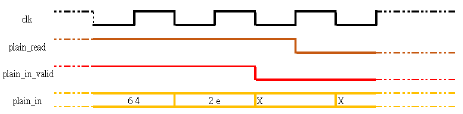
值輸入。圖七為cipher data及plain data的讀取時序圖，當read設為high的下一個clk將會把資料輸入。圖八為cipher data及plain data的資料結束時序圖，當plain\_in\_valid或cipher\_in\_valid由high轉low時代表資料輸入結束。



圖六、key\_in時序圖



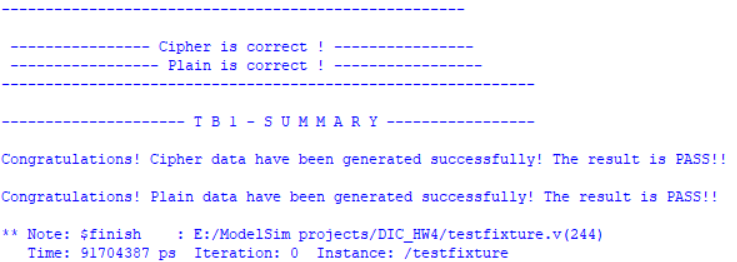
圖七、plain data及cipher data時序圖

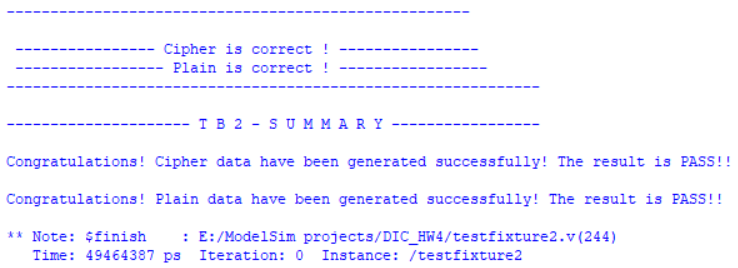


圖八、plain data及cipher data結束時序圖

## Result

#### Gate-Level simulation





#### Synthesis result

