CS5120 Final Project – RSA
Student ID: 107062612 Name: 熊祖玲

1. Design Concept

A. Algorithm

由於2048 bits的乘法和modular計算相當耗時，因此使用以下演算法加快運算。

I. RL Algorithm

輸入：M, E, N (plaintext, (E, N): public key pair)

輸出：$M^E \ mod \ N$

如圖一所示，根據E的第i個bit決定是否要乘上plaintext的2i次方，每做一次乘法就將得到得值做modular運算，而此運算時間相當耗時且無法合成，因此使用Montgomery algorithm做計算。

i	E_i	Step 4 (C)	Step 5 (P)
0	1	$1 \cdot M = M$	$(M)^2 = M^2$
1	1	$M \cdot M^2 = M^3$	$(M^2)^2 = M^4$
2	1	$M^3 \cdot M^4 = M^7$	$(M^4)^2 = M^8$
3	0	M^7	$(M^8)^2 = M^{16}$
4	1	$M^7 \cdot M^6 = M^{23}$	$(M^{16})^2 = M^{32}$
Step 7 $E_5 = 1$, thus $C = M^{23} \cdot M^{32} = M^{55}$			

```
1. C = 1; P = M
2. for i = 0 to h − 2
3.     if E_i = 1
4.         C = CP (mod N)
5.     P = PP (mod N)
6. if E_{h−1} = 1
7.     C = CP (mod N)
8. return C
```

圖一、RL Algorithm示意圖（左圖：演算法、右圖：執行範例）

II. Montgomery Algorithm

輸入：X, Y, N

輸出：$XYR^{-1} \ mod \ N \ (R = 2^{2048})$

如圖二所示，先將A設為0，若$A + x_k Y$為偶數則直接將$A + x_k Y$向右shift 1 bit並設為下一個iteration的A，反之則將$A + x_k Y + N$再向右shift 1 bit並設為下一個iteration的A，最後檢查A是否大於N，是則減去N。

```
1. A = 0
2. for k = 0 to m − 1
3.     if A + x_k Y is even
4.         A = (A + x_k Y)/2
5.     else
6.         A = (A + x_k Y + N)/2
7. if A > N
8.     A = A − N
9. return A
```

圖二、Montgomery Algorithm示意圖

經過Montgomery algorithm所得到的解為$XYR^{-1} \ mod \ N$，而非$XY \ mod \ N$，因此需要透過以下四個步驟才可以得到我們想要的解。

$XY \ mod \ N$

1. $\text{Mont}(X, R^2) = XR \ \ mod \ N$
2. $\text{Mont}(Y, R^2) = YR \ \ mod \ N$
3. $\text{Mont}(YR \ \ mod \ N, XR \ \ mod \ N) = XYR \ mod \ N$
4. $\text{Mont}(XYR \ mod \ N \ , 1) = XY \ mod \ N$

圖三、Montgomery Algorithm計算$XY \ \ mod \ \ N$之流程

計算R^2與$R^2 \ mod \ N$的仍然相當費時，使用圖四之演算法可以大幅降低計算時間。

輸入：b, N

輸出：$R^2 \ mod \ N \ (2^{2b} \ mod \ N)$

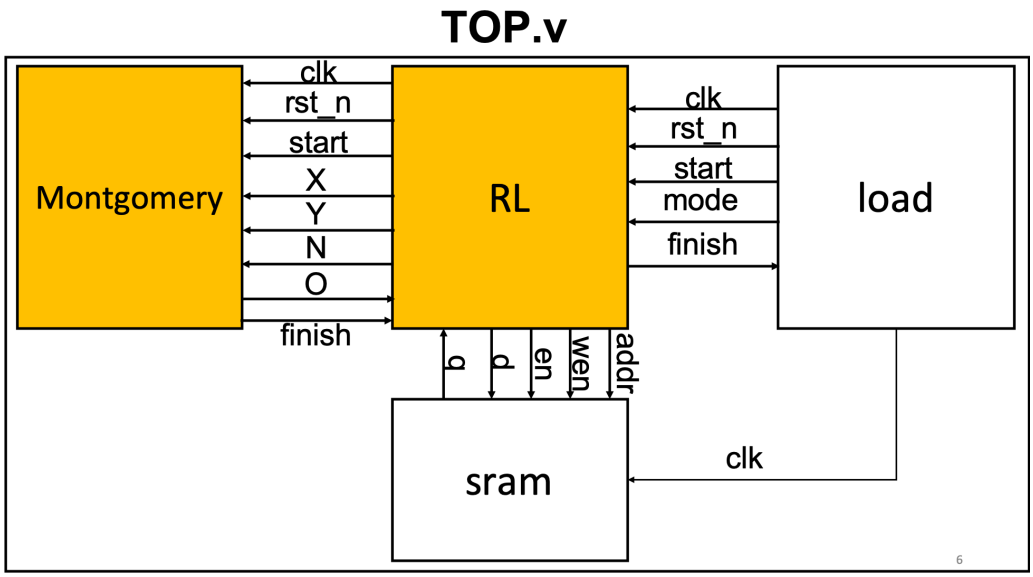
因為R^2為2^{2b}，所以可以透過$2b$次的乘2並且在每個iteration中都做一次modular運算 (將當前值不斷減去N，直到小於N為止) 得到$R^2 \ mod \ N$的結果。

1. $t = 1$
2. for $i = 0 \ to \ 2 \cdot b$
3. $\quad\quad t = t + t \ (mod \ N)$
4. return t

圖四、Montgomery Algorithm計算$R^2 \ \ mod \ \ N$之演算法

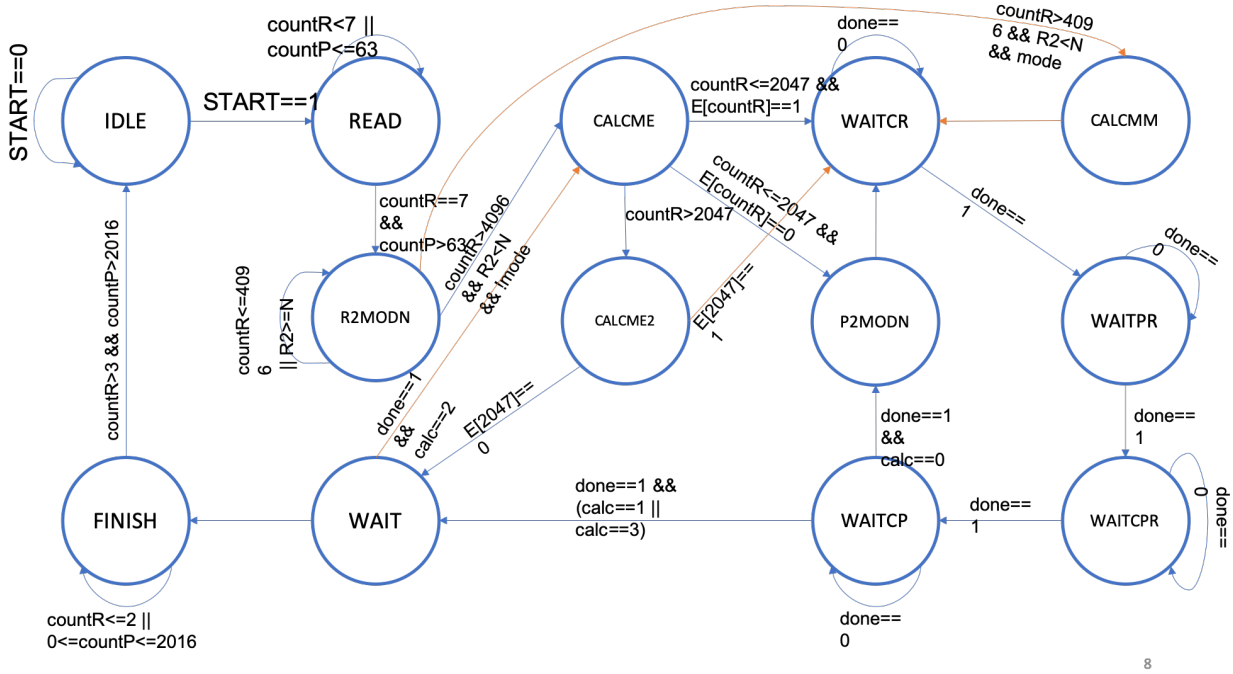
B. Block diagram

圖四為此RSA Engine的block diagram及與其它modules的連接情形。



圖五、Block Diagram

C. Finite state machine – RL



圖六、Finite State Machine圖 – RL algorithm

當RST_N訊號由1變0時，將所有訊號做初始化，並且進入IDLE狀態。

I. IDLE狀態
   i. START為0：停留在IDLE階段，並將所有訊號設為初始值。
   ii. START為1：進入READ狀態。

II. READ狀態
   從sram中取得資料並排列成2048 bit的plaintext和public key pair。
   i. countR為7且countP大於63：進入R2MODN狀態。
   ii. 其他：留在READ狀態。

III. R2MODN狀態
   用圖四的演算法計算$R^2 \bmod N\ (2^{2b} \bmod N)$。
   i. countR大於4096且mode為0：進入CALCME狀態。
   ii. countR大於4096且mode為1：進入CALCMM狀態。
   iii. 其它：留在R2MODN狀態。

IV. CALCME狀態
   用RL algorithm計算$M^E \bmod N$。
   i. countR大於2047：完成演算法中的迴圈部分，進入CALCME2狀態。
   ii. countR不大於2047且E_k為1：設定計算$CR \bmod N$時，Montgomery module中所需之參數，進入WAITCR狀態。
   iii. countR不大於2047且E_k為0：進入P2MODN狀態。

V. CALCMM狀態
   利用圖三的流程及Montgomery algorithm計算$XY \bmod N$。設定計算$CR \bmod N$時，Montgomery module中所需之參數，進入WAITCR狀態。

VI. WAITCR狀態

等待Montgomery module完成$CR \bmod N$的計算。

i. done為0：尚未完成則留在WAITCR狀態。

ii. done為1：完成則設定計算$PR \bmod N$時Montgomery module中所需之參數，進入WAITPR狀態。

VII. WAITPR狀態

等待Montgomery module完成$PR \bmod N$的計算。

i. done為0：尚未完成則留在WAITCR狀態。

ii. done為1：完成則設定計算$CPR \bmod N$時Montgomery module中所需之參數，進入WAITCPR狀態。

VIII. WAITCPR狀態

等待Montgomery module完成$CPR \bmod N$的計算。

i. done為0：尚未完成則留在WAITCPR狀態。

ii. done為1：完成則設定計算$CP \bmod N$時Montgomery module中所需之參數，進入WAITCP狀態。

IX. WAITCP狀態

等待Montgomery module完成$CP \bmod N$的計算。

i. done為0：尚未完成則留在WAITCP狀態。

ii. done為1且calc為0：完成則進入P2MODN狀態。

iii. done為1且calc為1或3：完成則進入WAIT狀態。

X. P2MODN狀態

利用圖三的流程及Montgomery algorithm計算$PP \bmod N$。設定計算$PR \bmod N$時，Montgomery module中所需之參數，進入WAITCR狀態。

XI. CALCME2狀態

完成RL algorithm計算$M^E \bmod N$中的迴圈部分後，根據E的most significant bit判斷是否再做一次$CP \bmod N$。

i. E[2047]為1：需再做一次$CP \bmod N$，則設定計算$CR \bmod N$時，Montgomery module中所需之參數，進入WAITCR狀態。

ii. E[2047]為0：完成所有計算，則進入WAIT狀態。

XII. WAIT狀態

等待訊號穩定，進入FINISH狀態。

XIII. FINISH狀態

將結果依序寫入sram中。

i. countR大於且countP大於2016：完成資料寫入，進入IDLE狀態。

ii. 其它：尚未完成資料寫入，留在FINISH狀態。

D. Finite state machine – Montgomery

當RST_N訊號由1變0時，將所有訊號做初始化，並且進入IDLE狀態。

I. IDLE狀態

i. START為0：停留在IDLE階段，並將所有訊號設為初始值。

ii. START為1：進入CALC狀態。

II. CALC狀態

計算$XYR^{-1} \bmod N$。

i. k皆小於2048：尚未完成計算，留在CALC狀態。

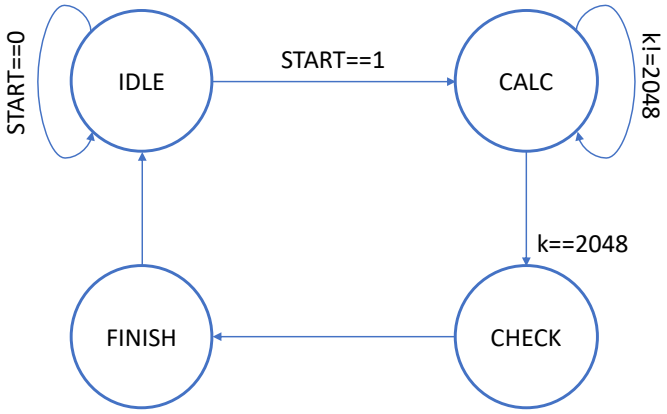
ii. 其它：完成計算，進入CHECK狀態。

III. CHECK狀態

檢查當前的結果是否大於N，是則減去，否則不變，最後進入FINISH狀態。

IV. FINISH狀態

將結果輸出並進入IDLE狀態。



圖七、Finite State Machine圖 – Montgomery algorithm

2. Simulation and Discussion

A. Simulation

圖八、九為模擬後的結果，波形檔案太大，因此以軟體輸出結果檢驗。

```
Mem[225]        0
Mem[226]        0
Mem[227]        0
Mem[228]        0
Mem[229]        0
Mem[230]        0
Mem[231]        0
Mem[232]        0
Mem[233]        0
Mem[234]        0
Mem[235]        0
Mem[236]        0
Mem[237]        0
Mem[238]        0
Mem[239]        0
Mem[240]        0
Mem[241]        0
Mem[242]        0
Mem[243]        0
Mem[244]        0
Mem[245]        0
Mem[246]        0
Mem[247]        0
Mem[248] 68187066
Mem[249] 4234121194
Mem[250] 1261655719
Mem[251] 3857262611
Mem[252] 3366201121
Mem[253] 3476572288
Mem[254] 3590481881
Mem[255] 1765092935
>> 67447646196807041044487161750000255686074259327873672287328618093365447670610 * 66464481640536293837185008583072588030979848284
59371225680772329584141411303 3 mod 74041861860884363236573073389577225936070545448027859973073691116391526935741
Answer = 18383196893274902041600229136097840461224214962458514124417310342438855102l5
Simulation complete via $finish(1) at time 146335 NS + 2
./load.v:109            $finish;
ncsim> exit
[kelly798186@ic22 final]$
```

圖八、modular multiplication模擬之螢幕截圖

```
Mem[226]        0
Mem[227]        0
Mem[228]        0
Mem[229]        0
Mem[230]        0
Mem[231]        0
Mem[232]        0
Mem[233]        0
Mem[234]        0
Mem[235]        0
Mem[236]        0
Mem[237]        0
Mem[238]        0
Mem[239]        0
Mem[240]        0
Mem[241]        0
Mem[242]        0
Mem[243]        0
Mem[244]        0
Mem[245]        0
Mem[246]        0
Mem[247]        0
Mem[248]        0
Mem[249]        0
Mem[250]        0
Mem[251]        0
Mem[252]        0
Mem[253]        0
Mem[254]    23001
Mem[255] 3143619459
>> 67447646196807041044487161750000255686074259327873672287328618093365447670610 ^ 6646448164053629383718500858307258803097984828459371225680772329584141411303
3 mod 74041861860884363236573073389577225936070545448027859973073691116391526935741
Answer =                        98791686394755
Simulation complete via $finish(1) at time 178145865 NS + 2
./load.v:110            $finish;
ncsim> exit
[kelly798186@ic22 final]$
```

圖九、modular exponentiation模擬之螢幕截圖

B. Simulation Execution Time

表一為modular multiplication和modular exponentiation在合成前、後進行模擬所花費的時間，可以得知合成後進行模擬所花費的時間皆比合成前多大約1000倍。

表一、 Simulation Execution Time統計表

	Execution Time (ps)
Pre Synthesis (modular multiplication)	146335
Post Synthesis (modular multiplication)	146335100
Pre Synthesis (modular exponentiation)	178145865
Post Synthesis (modular exponentiation)	178145865100

C. Speedup方法

I. 減少RL Algorithm的Iteration

可以判斷E尚未處理的bit(s)是否全部為零，是則可以提前完成所有計算，省去數次計算$PP \bmod N$的時間。

D. Area

如表二所示，進行RSA的計算所需之Combination area相對於SRAM area多很多，主要原因為SRAM中只儲存4個2048 bit的資料 (M, E, N, Output) ，而計算過程中用到許多DFF儲存當前計算的結果，因此花費的面積相當大。

表二、面積統計表

Combination area	266959.19564
Buf/Inv area	40112.800024
Noncombination area	204074.394598
Total area	undefined
Number of SRAM	1個256x32 sram
SRAM area	43400

3. Summary

完成這次project的過程中，演算法的實作並不困難，但我沒有太縝密的規劃，導致我的finite state machine相當複雜，而且閱讀性極低，不過也礙於需要準備其他科目的期末考與期末報告，所以沒有再做調整，也沒有做太多時間上的改善，甚至也沒時間操作APR tool完成整個layout。