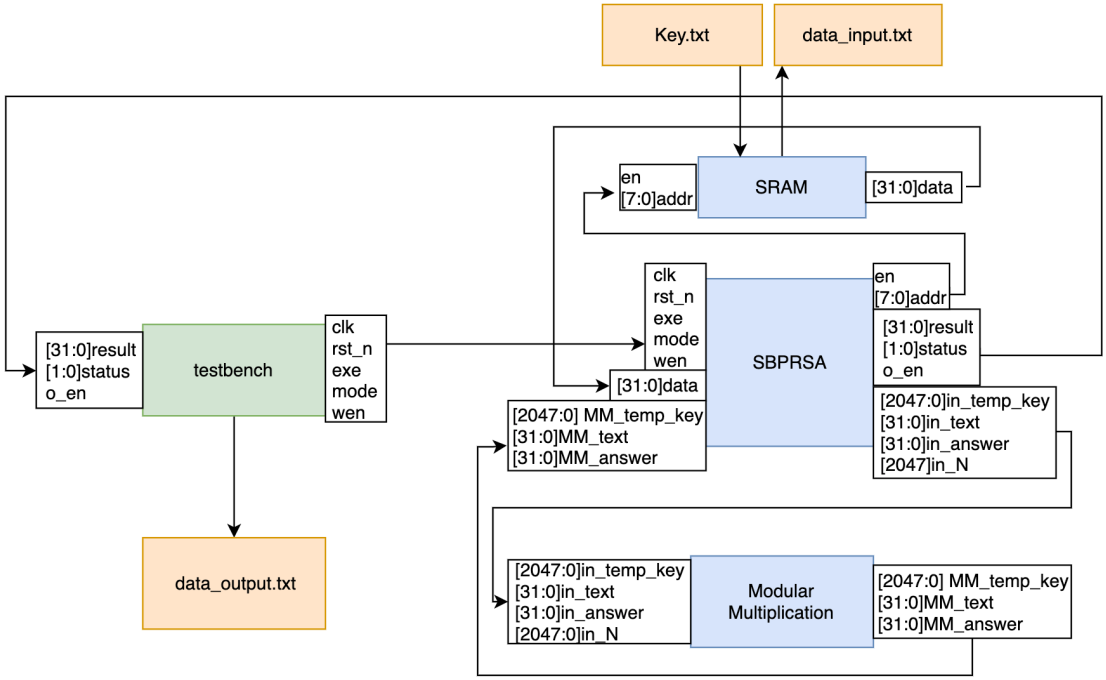
CS5120 Final Project - RSA

Student ID: 104062135 Name: 魏聖修

應用主題、設計概念、架構細節、方塊圖



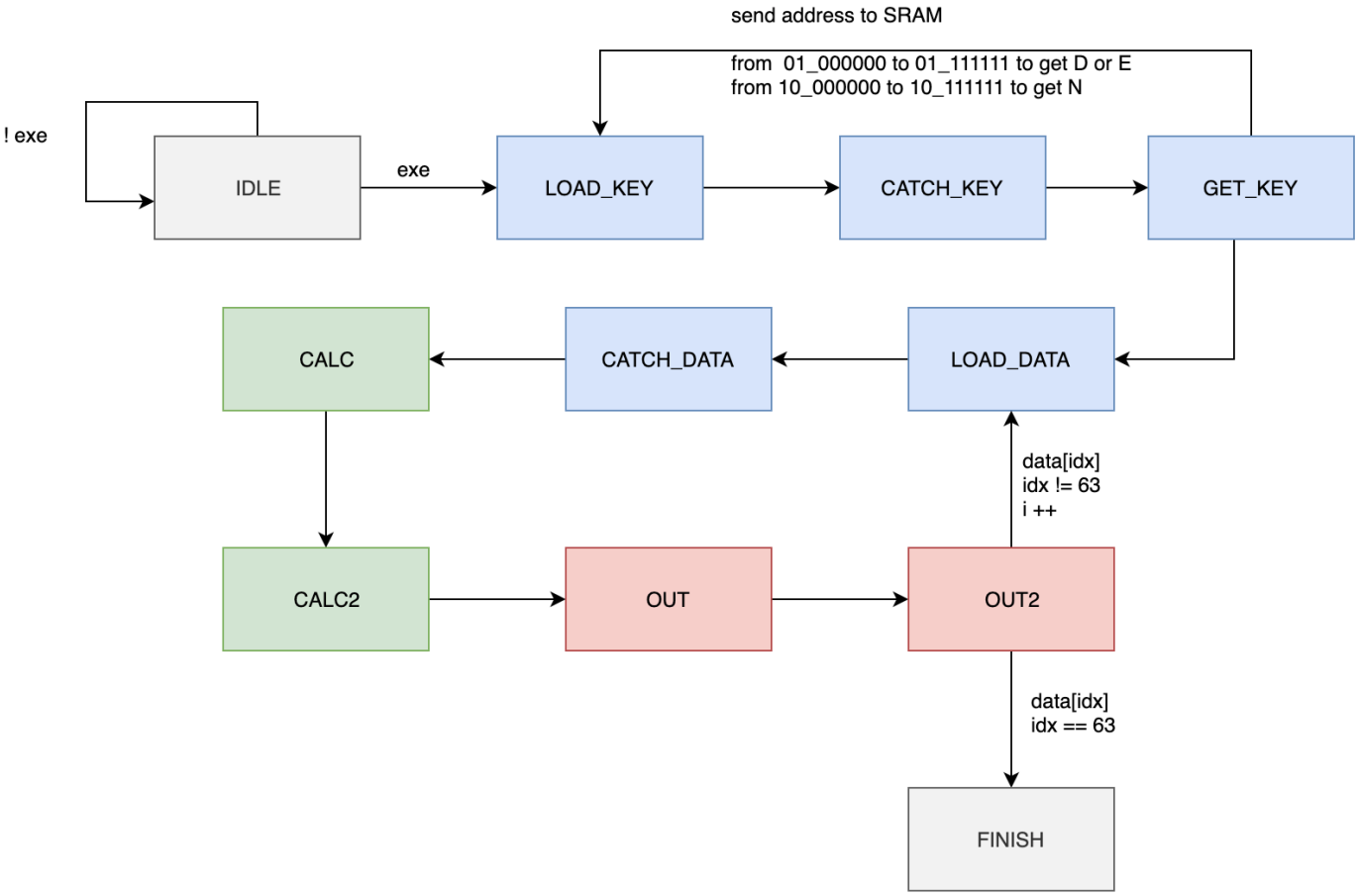
這次做 final project 我選擇的題目是 RSA。
而 module 的形式上大致是參考 spec 上的提示，再做一些改變，讓自己好做一點。
主要要設計的部分是 SRAM, SBPRSA, Modular Multiplication。

首先我先決定了 key 的 D 和 N 要怎麼存入 SRAM 內，我使用的方法是既然 D 和 N 都是 2048bit，我就自己寫檔案 key.txt，裡面包含了 D 和 N 的資訊，這樣要使用特定的 key 的話，修改也比較方便，對照例子也比較好驗證。再來 data 我是用 random 的，共有 2048bit 的 data 並且每 32bit 視為一個 word 進行一次加解密的運算，也為了方便驗證，random 完我有把數字記錄下來輸出到 data_input.txt 中。

再來就是主要的 SBPRSA 的部分。除了依照要求輸入 exe, mode, wen 等等的信號後，一開始就是把 SRAM 內的資訊讀進來，一開始是先讀 key，由於每次只能傳輸 key 的其中 32bits，所以我在 get_key state 時會把目前讀進來的一部分 key 做 bit shift，再加到已有的 key 中，D 和 N 都一樣。
再來就是進行 data 的加解密的主要運算了，由於加解密是很大的數字的高次方模運算，不能只寫成 data ** D mod N。所以改用 spec 內提示的 Montgomery Algorithm。
我主要參考這一篇的內容 https://blog.csdn.net/linraise/article/details/17490769
不過我本來是把 Modular Multiplication 的運算寫在 SBPRSA 內，可以 simulation，也能得到正確的結果，但是要合成時，可能是因為加法器用太多且 2048bit 空間的 key 造成合成時間在 compile 這段非

常的久，睡了一覺起來還沒好。所以我索性把 Modular Multiplication 寫成獨立的 module 再去接就能順利的合成了。

最後只要把加解密完的資訊再寫成 data_output.txt 去做驗證看看是否正確
至於驗證的方法就是用 python 去進行 data ** D mod N，比對答案是否相同，但是由於 example 4 的 E 3,037,987,481 太大了，甚至 python 都跑很慢，所以我只驗證了 21,401 的。



State diagram 的部分，基本上就是從 SRAM 裡拿資料 -> 運算 -> output
拿取資料的部分：
LOAD_KEY, LOAD_DATA 都是傳出 addr 給 SRAM。
CATCH_KEY, CATCH_DATA 都是下個 cycle 拿剛剛傳的 addr 後，SRAM 回傳的資料。
GET_KEY 是對 key 再另外做必要的運算。
運算資料的部分：
CALC 是把目前的 data, key, N 都傳出去給 Modular Multiplication。
CALC2 是接 Modular Multiplication 傳回來的資料。
輸出資料的部分：
OUT 是把 o_en 設 1’b1 並把結果丟給 testbench 寫檔，以達成 FIFO。
OUT2 是把 o_en 設回 1’b0。
當全部都輸出完後就 Finish。

實作、測試完成度

demo 完後我多試了一個驗證方式，就是先加密再解密，看會不會變成原來的

目前我只寫了 key 4d 的驗證

Plain

```
B[         0] :   303379748
B[         1] :  3230228097
B[         2] :  2223298057
B[         3] :  2985317987
B[         4] :   112818957
B[         5] :  1189058957
B[         6] :  2999092325
B[         7] :  2302104082
B[         8] :    15983361
B[         9] :   114806029
B[        10] :   992211318
B[        11] :   512609597
B[        12] :  1993627629
B[        13] :  1177417612
B[        14] :  2097015289
B[        15] :  3812841926
B[        16] :  3807872197
B[        17] :  3574846122
B[        18] :  1924134885
B[        19] :  3151131255
B[        20] :  2301810194
B[        21] :  1206705039
B[        22] :  2033215986
B[        23] :  3883308750
B[        24] :  4093672168
B[        25] :  3804909253
B[        26] :   777537884
B[        27] :  3733858493
B[        28] :  2527811629
B[        29] :  2997298789
B[        30] :  2985255523
B[        31] :    91457290
B[        32] :  3225100928
B[        33] :   274997536
B[        34] :  1433945514
B[        35] :  3469528221
B[        36] :  3407888022
B[        37] :  2307110931
B[        38] :  2260482061
B[        39] :  2846348883
B[        40] :   899669355
B[        41] :  3936758485
B[        42] :  2165787138
B[        43] :  3612753582
B[        44] :   251652381
B[        45] :  3888476879
B[        46] :   293882147
B[        47] :    84501770
B[        48] :  3849521866
B[        49] :  2654030908
B[        50] :  2036907506
B[        51] :  1160667530
B[        52] :   549761857
B[        53] :  3964351704
B[        54] :  1008792440
B[        55] :  3297383049
B[        56] :  1975848427
B[        57] :  1526883766
B[        58] :  1665923526
B[        59] :  1460999086
B[        60] :  3732210364
B[        61] :   353361194
```

cypher

```
A[         0] :  3084430199
A[         1] :  2779622216
A[         2] :  1749042942
A[         3] :   186571227
A[         4] :  1696370600
A[         5] :  3778618399
A[         6] :  2811351461
A[         7] :   269561446
A[         8] :   871515217
A[         9] :  1137602211
A[        10] :  1568305078
A[        11] :  1274886529
A[        12] :   386621747
A[        13] :  3561525872
A[        14] :  1991342889
A[        15] :  2607399285
A[        16] :   231151433
A[        17] :   111387006
A[        18] :  2010239971
A[        19] :   513045910
A[        20] :  1289989572
A[        21] :  2790060766
A[        22] :  2362479240
A[        23] :  3050016880
A[        24] :  3149783633
A[        25] :  2696074007
A[        26] :  2675972709
A[        27] :  1220482394
A[        28] :  3274807277
A[        29] :   559360627
A[        30] :  1092120173
A[        31] :  1581896421
A[        32] :  2989166035
A[        33] :   344847497
A[        34] :   632503714
A[        35] :  2162447698
A[        36] :  1744523811
A[        37] :   547910502
A[        38] :  3003339284
A[        39] :  3535297378
A[        40] :   427323184
A[        41] :    71169144
A[        42] :  3863628495
A[        43] :  1337326289
A[        44] :   932008593
A[        45] :  3031572400
A[        46] :  3201443848
A[        47] :   609945104
A[        48] :   916540758
A[        49] :  3790357990
A[        50] :  1733171429
A[        51] :   531170888
A[        52] :  3772331000
A[        53] :  1417507730
A[        54] :   579313691
A[        55] :  3584614670
A[        56] :    52759852
A[        57] :  3409795118
A[        58] :   277977186
A[        59] :  2300765206
A[        60] :  3651186607
A[        61] :  3778329553
```

cypher

```
B[         0] :  3084430199
B[         1] :  2779622216
B[         2] :  1749042942
B[         3] :   186571227
B[         4] :  1696370600
B[         5] :  3778618399
B[         6] :  2811351461
B[         7] :   269561446
B[         8] :   871515217
B[         9] :  1137602211
B[        10] :  1568305078
B[        11] :  1274886529
B[        12] :   386621747
B[        13] :  3561525872
B[        14] :  1991342889
B[        15] :  2607399285
B[        16] :   231151433
B[        17] :   111387006
B[        18] :  2010239971
B[        19] :   513045910
B[        20] :  1289989572
B[        21] :  2790060766
B[        22] :  2362479240
B[        23] :  3050016880
B[        24] :  3149783633
B[        25] :  2696074007
B[        26] :  2675972709
B[        27] :  1220482394
B[        28] :  3274807277
B[        29] :   559360627
B[        30] :  1092120173
B[        31] :  1581896421
B[        32] :  2989166035
B[        33] :   344847497
B[        34] :   632503714
B[        35] :  2162447698
B[        36] :  1744523811
B[        37] :   547910502
B[        38] :  3003339284
B[        39] :  3535297378
B[        40] :   427323184
B[        41] :    71169144
B[        42] :  3863628495
B[        43] :  1337326289
B[        44] :   932008593
B[        45] :  3031572400
B[        46] :  3201443848
B[        47] :   609945104
B[        48] :   916540758
B[        49] :  3790357990
B[        50] :  1733171429
B[        51] :   531170888
B[        52] :  3772331000
B[        53] :  1417507730
B[        54] :   579313691
B[        55] :  3584614670
B[        56] :    52759852
B[        57] :  3409795118
B[        58] :   277977186
B[        59] :  2300765206
B[        60] :  3651186607
B[        61] :  3778329553
```

plain

```
A[         0] :   303379748
A[         1] :  3230228097
A[         2] :  2223298057
A[         3] :  2985317987
A[         4] :   112818957
A[         5] :  1189058957
A[         6] :  2999092325
A[         7] :  2302104082
A[         8] :    15983361
A[         9] :   114806029
A[        10] :   992211318
A[        11] :   512609597
A[        12] :  1993627629
A[        13] :  1177417612
A[        14] :  2097015289
A[        15] :  3812841926
A[        16] :  3807872197
A[        17] :  3574846122
A[        18] :  1924134885
A[        19] :  3151131255
A[        20] :  2301810194
A[        21] :  1206705039
A[        22] :  2033215986
A[        23] :  3883308750
A[        24] :    71283345
A[        25] :  3804909253
A[        26] :   777537884
A[        27] :  3733858493
A[        28] :  2527811629
A[        29] :  2997298789
A[        30] :  2985255523
A[        31] :    91457290
A[        32] :  3225100928
A[        33] :   274997536
A[        34] :  1433945514
A[        35] :  3469528221
A[        36] :  3407888022
A[        37] :  2307110931
A[        38] :  2260482061
A[        39] :  2846348883
A[        40] :   899669355
A[        41] :  3936758485
A[        42] :  2165787138
A[        43] :  3612753582
A[        44] :   251652381
A[        45] :  3888476879
A[        46] :   293882147
A[        47] :    84501770
A[        48] :  3849521866
A[        49] :  2654030908
A[        50] :  2036907506
A[        51] :  1160667530
A[        52] :   549761857
A[        53] :  3964351704
A[        54] :  1008792440
A[        55] :  3297383049
A[        56] :  1975848427
A[        57] :  1526883766
A[        58] :  1665923526
A[        59] :  1460999086
A[        60] :  3732210364
A[        61] :   353361194
```

這個之外也可以直接用 python 計算來驗證

Python 驗證

```
>>> 303379748**21599%430872577
220579764L
>>> 3230228097**21599%430872577
217842784L
>>> 2223298057**21599%430872577
118180589L
>>> 2985317987**21599%430872577
255613983L
>>> 112818957**21599%430872577
283279181L
>>> 1189058957**21599%430872577
253044L
>>>
```

EXAMPLE 3_D

B[i]:原本的 data

```
B[          0] :   303379748
B[          1] :  3230228097
B[          2] :  2223298057
B[          3] :  2985317987
B[          4] :   112818957
B[          5] :  1189058957
B[          6] :  2999092325
B[          7] :  2302104082
B[          8] :    15983361
B[          9] :   114806029
B[         10] :   992211318
B[         11] :   512609597
B[         12] :  1993627629
B[         13] :  1177417612
B[         14] :  2097015289
B[         15] :  3812041926
B[         16] :  3807872197
B[         17] :  3574846122
B[         18] :  1924134885
B[         19] :  3151131255
B[         20] :  2301810194
B[         21] :  1206705039
B[         22] :  2033215986
B[         23] :  3883308750
B[         24] :  4093672168
B[         25] :  3804909253
B[         26] :   777537884
B[         27] :  3733858493
B[         28] :  2527811629
B[         29] :  2997298789
B[         30] :  2985255523
B[         31] :    91457290
B[         32] :  3225100928
B[         33] :   274997536
B[         34] :  1433945514
B[         35] :  3469528221
B[         36] :  3407888022
B[         37] :  2307110931
B[         38] :  2260482061
B[         39] :  2846348883
B[         40] :   899669355
B[         41] :  3936758485
B[         42] :  2165787138
B[         43] :  3612753582
B[         44] :   251652381
B[         45] :  3888476879
B[         46] :   293882147
B[         47] :    84501770
B[         48] :  3849521866
B[         49] :  2654030908
B[         50] :  2036907506
B[         51] :  1160667530
B[         52] :   549761857
B[         53] :  3964351704
B[         54] :  1008792440
B[         55] :  3297383049
B[         56] :  1975848427
B[         57] :  1526883766
B[         58] :  1665923526
B[         59] :  1460999086
B[         60] :  3732210364
B[         61] :   353361194
```

A[i]運算完的 data

```
A[          0] :   220579764
A[          1] :   217842784
A[          2] :   118180589
A[          3] :   255613983
A[          4] :   283279181
A[          5] :      253044
A[          6] :   400722183
A[          7] :   109780611
A[          8] :    26600300
A[          9] :    71038823
A[         10] :   182505066
A[         11] :    89685600
A[         12] :   156282766
A[         13] :   367684858
A[         14] :   280973385
A[         15] :   128461613
A[         16] :   341054594
A[         17] :   267295221
A[         18] :    19480097
A[         19] :   283667580
A[         20] :   393005290
A[         21] :   116370184
A[         22] :    84035849
A[         23] :   111795190
A[         24] :   137131132
A[         25] :   165869562
A[         26] :   362094024
A[         27] :   166378018
A[         28] :    18605011
A[         29] :      551627
A[         30] :   133853903
A[         31] :   341056967
A[         32] :    68827429
A[         33] :   430161932
A[         34] :   309651901
A[         35] :    80091270
A[         36] :   329539217
A[         37] :    65797921
A[         38] :   164513063
A[         39] :   140850208
A[         40] :   164069681
A[         41] :    79319495
A[         42] :    56653593
A[         43] :   340372981
A[         44] :    83300188
A[         45] :    74854727
A[         46] :   327454352
A[         47] :   381473689
A[         48] :   194240274
A[         49] :    86779662
A[         50] :   392833056
A[         51] :   394047591
A[         52] :   358725402
A[         53] :    86524316
A[         54] :   147902015
A[         55] :    98661993
A[         56] :    24780270
A[         57] :    17497058
A[         58] :   407664449
A[         59] :   312656317
A[         60] :   196191479
A[         61] :   109867123
```

```
****************************************
Report : area
Design : SBPRSA
Version: K-2015.06-SP1
Date   : Wed Jun 19 16:36:42 2019
****************************************

Library(s) Used:

    slow (File: /theda21_2/CBDK_IC_Contest/cur/SynopsysDC/db/slow.db)

Number of ports:                        18661
Number of nets:                         74518
Number of cells:                        64050
Number of combinational cells:          57759
Number of sequential cells:              6288
Number of macros/black boxes:               0
Number of buf/inv:                      16824
Number of references:                      39

Combinational area:            443393.119290
Buf/Inv area:                   68948.386937
Noncombinational area:         224014.371281
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:               667407.490571
Total area:                 undefined
1
```

```
  data arrival time                                                    8.32

  clock clk (rise edge)                                  10.00        10.00
  clock network delay (ideal)                             0.00        10.00
  N_reg[2047]/CK (EDFFX1)                                 0.00        10.00 r
  library setup time                                     -0.62         9.38
  data required time                                                   9.38
  --------------------------------------------------------------------------
  data required time                                                   9.38
  data arrival time                                                   -8.32
  --------------------------------------------------------------------------
  slack (MET)                                                          1.06
```

```
****************************************
Report : power
        -analysis_effort low
Design : SBPRSA
Version: K-2015.06-SP1
Date   : Wed Jun 19 16:37:43 2019
****************************************


Library(s) Used:

    slow (File: /theda21_2/CBDK_IC_Contest/cur/SynopsysDC/db/slow.db)


Operating Conditions: slow   Library: slow
Wire Load Model Mode: top


Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =  13.5610 mW   (97%)
  Net Switching Power  = 382.7162 uW    (3%)
                         ---------
Total Dynamic Power    =  13.9437 mW  (100%)

Cell Leakage Power     = 470.8052 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000              0.0000             0.0000  (   0.00%)
register          13.3726        4.1363e-03          1.5418e+08          13.5310  (  93.87%)
sequential         0.0000           0.0000              0.0000             0.0000  (   0.00%)
combinational      0.1887           0.3786          3.1662e+08             0.8838  (   6.13%)
--------------------------------------------------------------------------------------------------
Total             13.5613 mW        0.3827 mW       4.7081e+08 pW         14.4148 mW
1
```

這是這次設計的 area, timing, power。

Area 差不多有 2/3 是 combinational 主要是 state 控制和讀 data 處理 data 的部分。

另外 1/3 是 non-combinational 主要就是 Modular Multiplication 運算的部分。

再來和之前的作業比較一下 SRAM 的大小，是 32*256 的 SRAM 大概是 43400um^2

Timing 的部分 slack 是正的，應該沒有問題。

Power 的部分看得出來大部分都是 register 的部分高達 93.87%。

剩下就是 combinational 的部分大概 6.13%。

加速效果討論與分析

實作完成度基本上有 100%，
有做完基本功能的 2048bits 的 key 和 N 和 data，
RTL 和合成後都能正常運行，
不過並沒有做到當初說的額外功能，算是有點美中不足。

加速的部分
就大數字高次方模運算來討論的話，
如果使用 python 要跑 (32bits)數字的(32bits)次方 mod(32bits)數字，
基本上短時間內跑不出來，
但使用 Montgomery 來計算的話真的快很多，
整體加解密運算是很迅速的。
此外，我在把每 32bits 串接成 2048bits 使用的不是乘以 2 的次方，使用的是 bit shift，在速度上加速不明顯的，但最主要是合成時，少合成了很多個乘法器，合成速度加速不少，做了這個修改才讓我能夠順利合成的原因之一。

困難與解決方法

這次主要有點問題的部分是實作 Montgomery Algorithm，其實我本來看 spec 是看不太懂那個演算法是在做什麼，所以上網查了一下別人的解說方法，再加以應用，所以花了不少時間。
再來，因為我 data 是用 random 的，但是 seed 都會是一樣，除非重新 make 去改變 seed，否則每次出來的數字其實都一樣，我上網找了一下還是不知道怎麼解決，所以姑且就都用同一組數字。
然後，就是合成的問題了，之前都沒有遇過合成不完的情況，但這次遇到了，只能盡量想辦法減少加法器，簡化設計出來的東西，才勉強能合成完，我覺得應該有更好的設計方法和演算法，可以避免這種問題，不用直接吃 2048bits 的 key，不過由於這次時間不太夠，所以想不出更好的辦法。
而且這次甚至沒有做到 APR，感覺那部分又會有更多的問題。

心得討論

這次我本來也在 convolution 和 RSA 之間做選擇，不過想說 convolution 作業有做過類似的，只是複雜了好幾層，所以選擇了 RSA，想說可以試試看別種設計。而且之前上密碼學也有學過 RSA，能自己做一個是還蠻酷的。不過設計的時候問題真的有點多，所以花了很多的心力和時間，最後也才做到最基本的要求，也沒有其他的額外功能。不過這堂課真的學到蠻多的，雖然很多部分我們都還沒有辦法拿去應用。其中最重要的應該就是重新好好複習了 Verilog 了吧，經過兩年沒碰幾乎是重新學了一遍。最後再謝謝助教和教授，辛苦你們了，謝謝。