DC Lab2 Report

Team 04

B10901163 張顥譽 B10901176 蔡弘祥 B10901179 鄭承瑞

File Structure

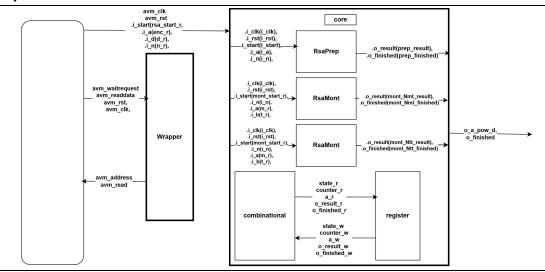
team04_lab2

|- team04_lab2_report

|- src

- |- Rsa256Wrapper.sv
- |- Rsa256core.sv
- |- DE2_115.qsf
- |- DE2_115.sv
- |- DE2_115.sdc

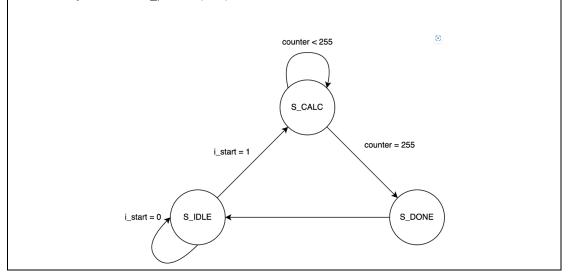
System Architecture



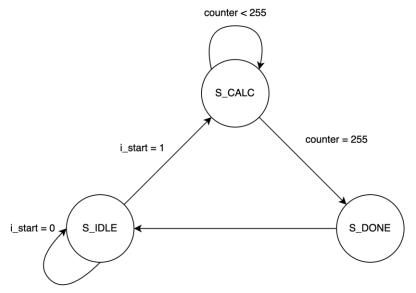
Hardware Scheduling

Modules / Submodules

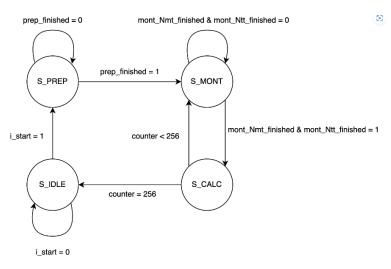
1. RsaPrep: Do modulo_product(N, a)



- S IDLE: wait for master module call
- S_CALC: do the for-loop in the pseudocode, need 256 cycles
- S_DONE: output the result and set the finish signal to 1
- 2. RsaMont: Do montgomery_algorithm(N, a, b)



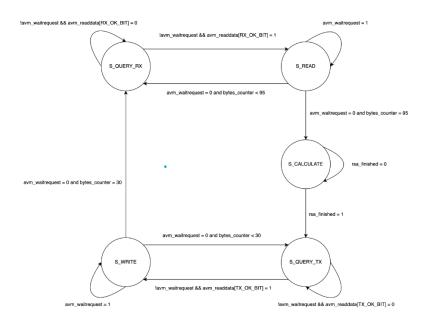
- S_IDLE: wait for master module call
- S_CALC: do the for-loop in the pseudocode, need 256 cycles
- S_DONE: output the result and set the finish signal to 1
- **3.** Rsa256Core: Do rsa256_mont(N, y, d)



- S_IDLE: wait for master module call
- S_PREP: do modulo_product(N, a). if done, go to S_MONT
- S_MONT: do montgomery_algorithm(N, m, t) and montgomery_algorithm(N, t, t) parallelly. if both done, go to S_MONT

• S_CALC: do the for-loop in the pseudocode, need 256 cycles. output the result and set the finish signal to 1 in the last cycle

4. Wrapper



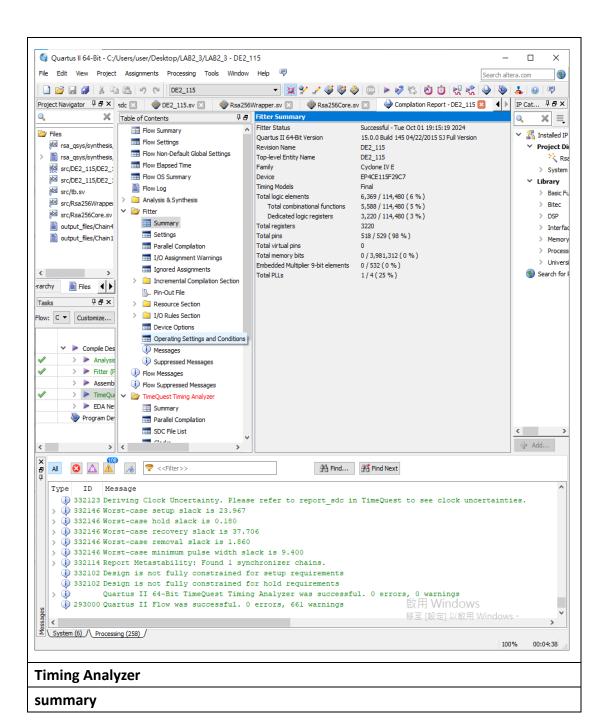
Algorithm

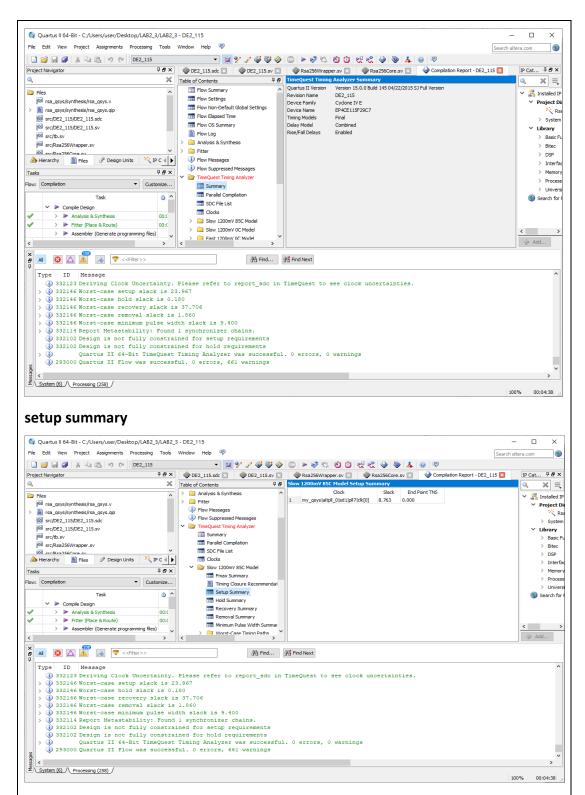
below algorithm is pseudocode,

```
def modulo_product(N, a):
    Function to perform modular multiplication: (2^{(256)} * a) % N
   Args:
       N : modulus
        a : operand
    Returns:
       result of the modular product
   t = a
   m = 0
   for i in range(256):
       if (2**256 >> i) & 1:
           if m + t >= N:
                m = m + t - N
            else:
                m = m + t
       if t + t >= N:
           t = t + t - N
        else:
           t = t + t
    return m
```

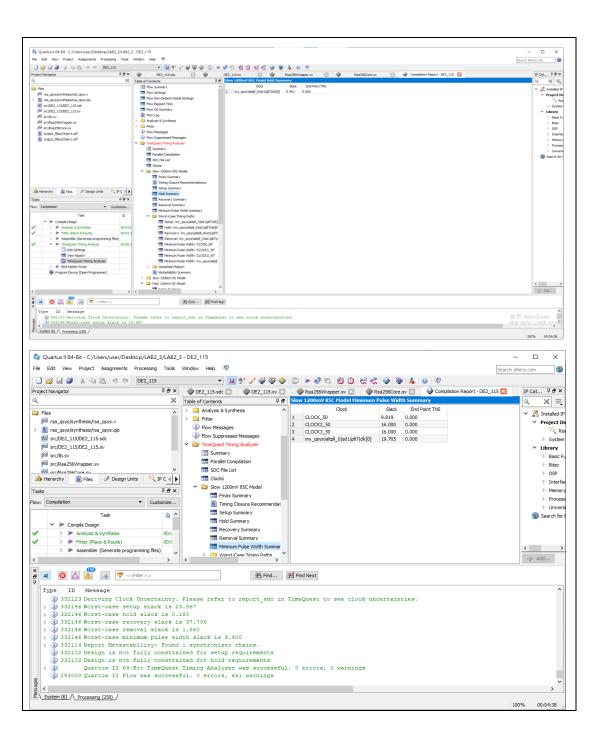
```
def montgomery_algorithm(N, a, b):
    Montgomery Algorithm to compute (a * b * 2^-256) % N
    Args:
       N : modulus
        a : operand 1
        b : operand 2
    Returns:
        result of Montgomery multiplication
    m = 0
    for i in range(256):
        if (a >> i) & 1:
            m += b
        if m % 2 == 1:
            m += N
        m //= 2
    if m >= N:
        m -= N
    return m
def rsa256_mont(N, y, d):
    m = 1
    t = modulo_product(N, y)
    # Iterate over the bits of the exponent d
    for i in range(256):
        if (d >> i) & 1:
            m = montgomery_algorithm(N, m, t)
        t = montgomery_algorithm(N, t, t)
    return m
```

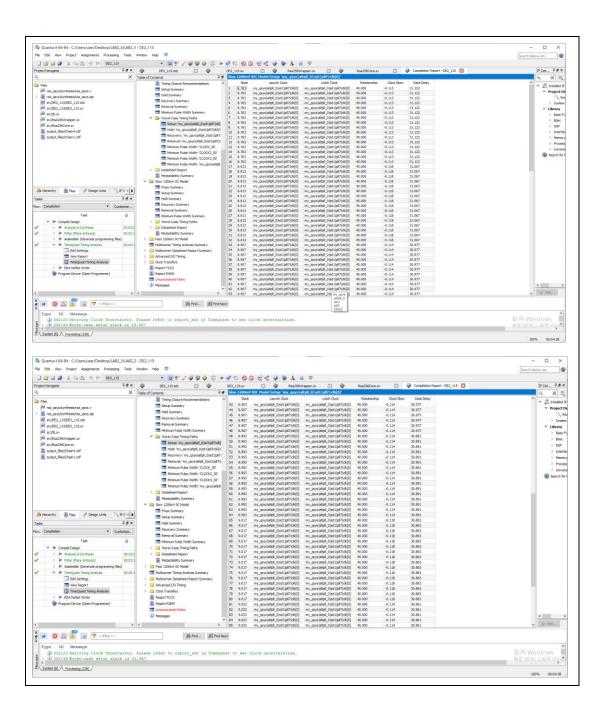
Fitter Summary

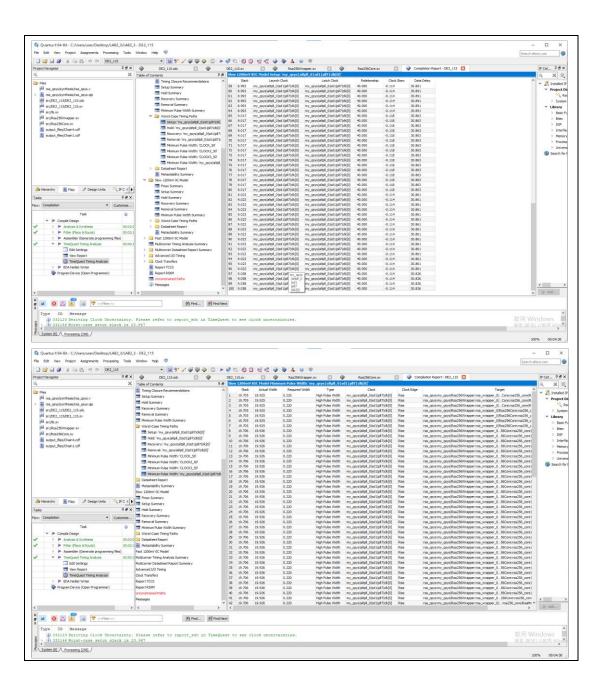


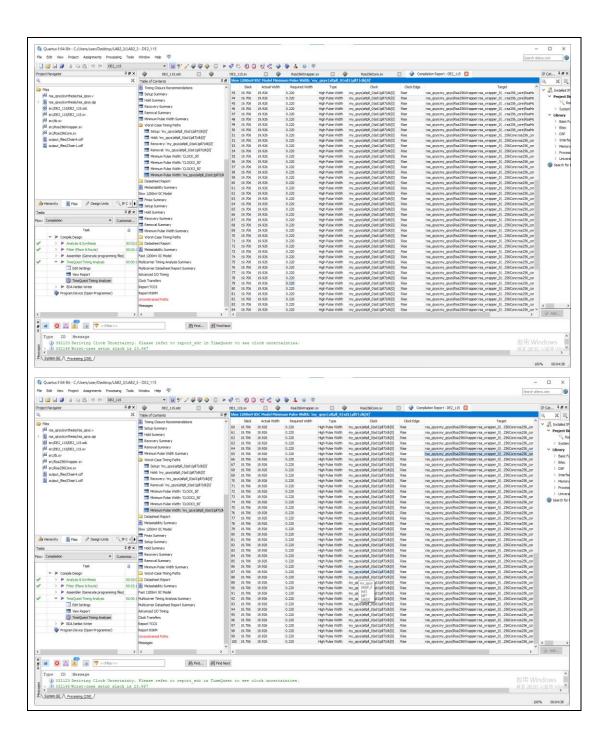


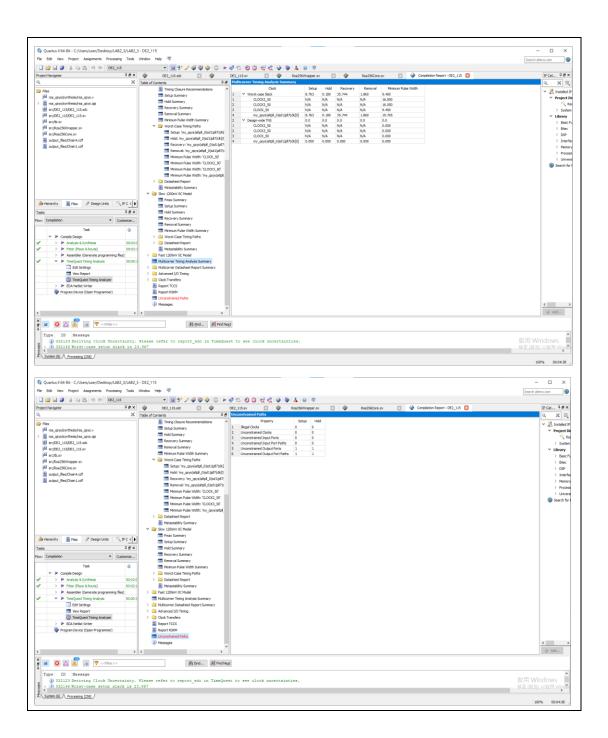
hold

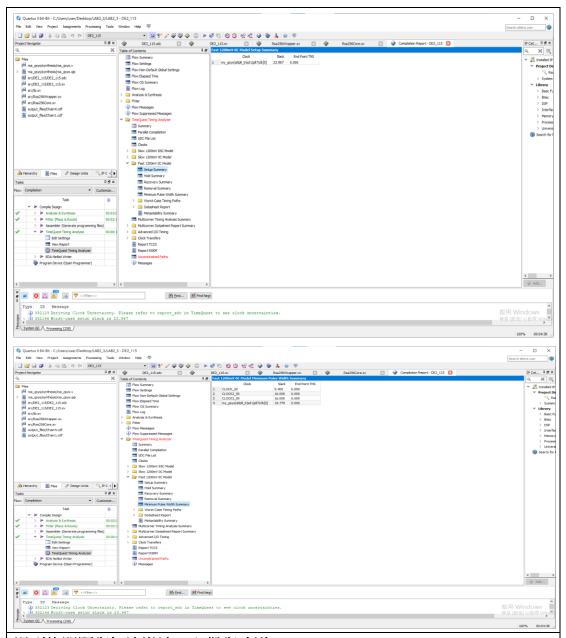












遇到的問題與解決辦法,心得與建議

遇到的問題:很多簡報的部分沒有寫清楚,例如輸入格式,還有一個是Rsa256Core 不能用 always comb(會報錯,助教可以補充在 slide 上),以及一個是 rst 要改成 reset,還有一個是 S_Write 結束時,bytecounter 要設成 64(因為他沒有要再讀 n,d),坑太多,族繁不及備載。

好險我們不會屈服於困難,和第二組並肩作戰,最後在實驗室搞了 5 個小時 終於完成。

心得:做出來的那瞬間就是 decipher 3 的輸出值。