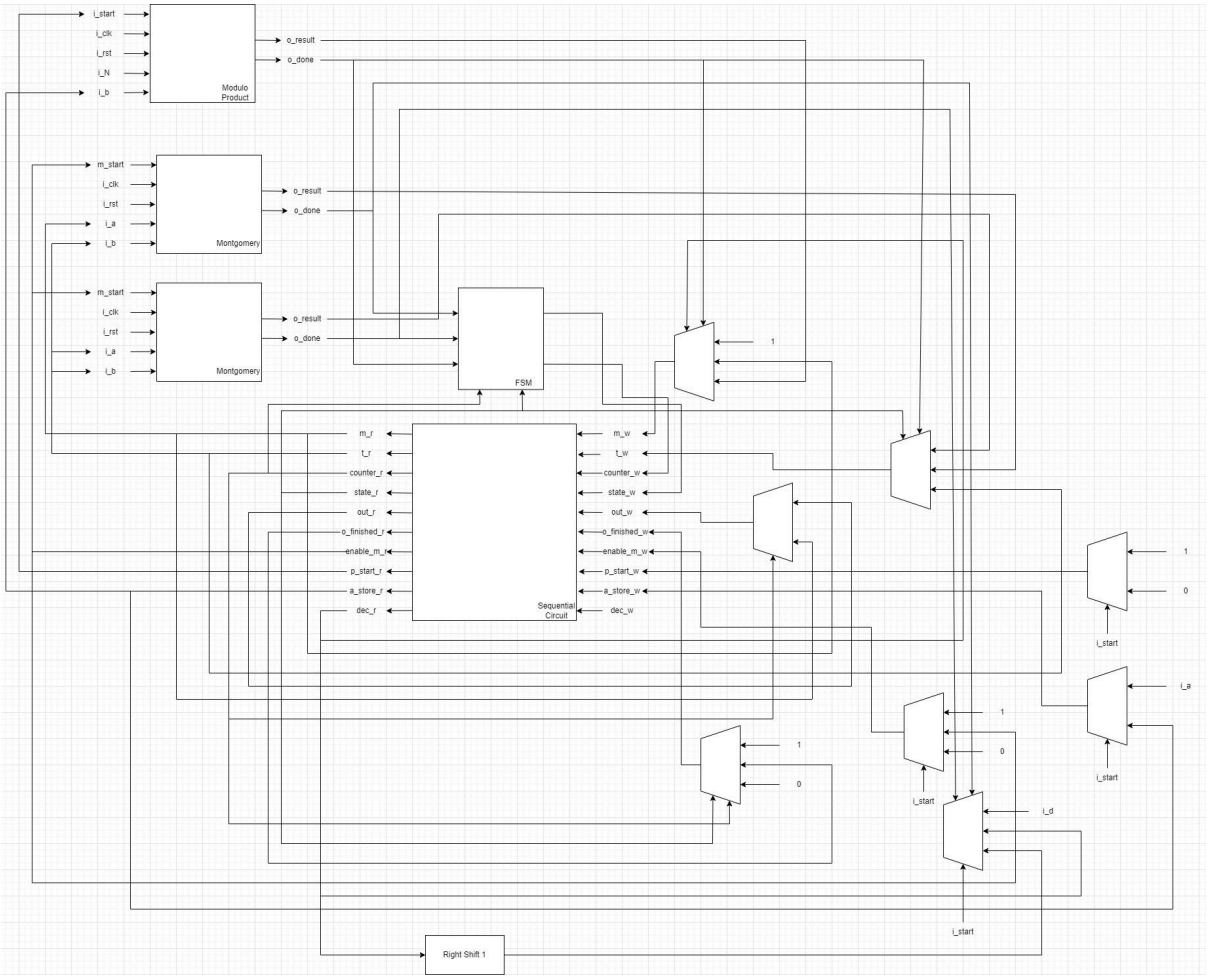
File Structure:

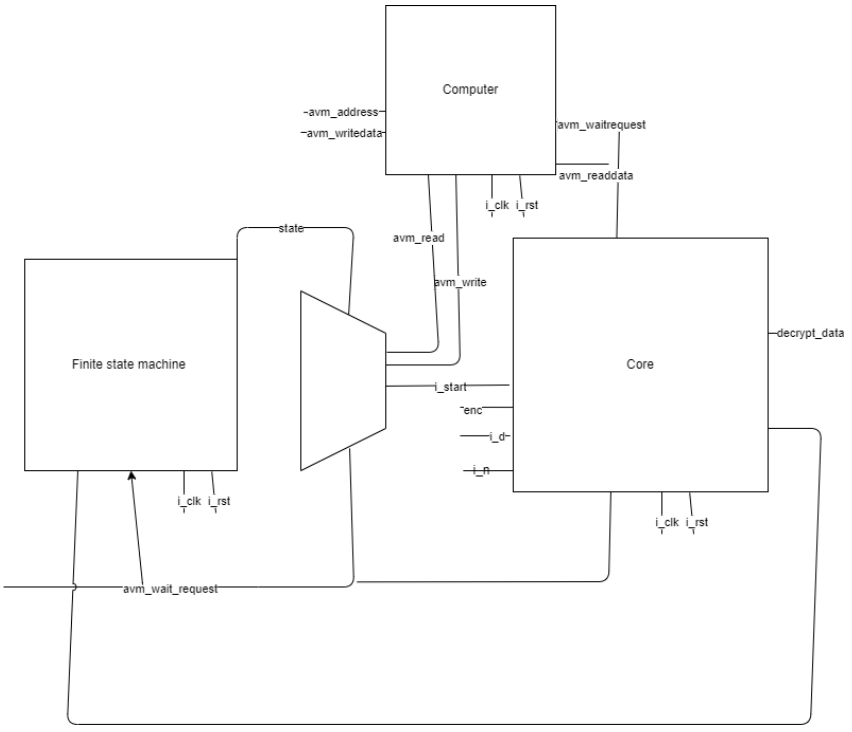
```
team01_lab2
│  team01_lab2_report.pdf
│
└─src
   │  MyRsa256Core.sv
   │  MyRsa256Wrapper.sv
   │
   ├─DE2_115
   │     DE2_115.qsf
   │     DE2_115.sdc
   │     DE2_115.sv
   │
   ├─pc_python
   │  │  enc.bin
   │  │  key.bin
   │  │  python
   │  │  rs232.cpp
   │  │  rs232.py
   │  │
   │  └─golden
   │        dec1.txt
   │        dec2.txt
   │        dec3.txt
   │        enc1.bin
   │        enc2.bin
   │        enc3.bin
   │        key.bin
   │        key_ascii.txt
   │        rsa.py
   │
   └─tb_verilog
         PipelineCtrl.v
         PipelineTb.v
         tb.sv
         test_wrapper.sv
         wrapper_input.txt
         wrapper_output.txt
```

System Architecture:

Data Path of RSA_Core:



Data Path of Wrapper:



Hardware Scheduling:

Algorithm of Modulo Product:

```
function ModuloProduct(N, a, b, k)                    ▷ k is number of bits of a
    t ← b
    m ← 0
    for i ← 0 to k do
        if i-th bit of a is 1 then
            if m + t ≥ N then
                m ← m + t − N                         ▷ perform modulo operation in each iteration
            else
                m ← m + t
            end if
        end if
        if t + t ≥ N then
            t ← t + t − N                             ▷ perform modulo operation in each iteration
        else
            t ← t + t
        end if
    end for
    return m
end function
```

Need additional hardware for comparison

Calculates $a \times b \ (mod\ N)$

Algorithm of Montgomery:

```
function MontgomeryAlgorithm(N, a, b)
    m ← 0
    for i ← 0 to 255 do
        if i-th bit of a is 1 then
            m ← m + b
        end if
        if m is odd then                              ▷ 4~6: replace multiplication with
            m ← m + N                                   successive addition
        end if
        m ← m/2                                       ▷ 7~10: calculate the modulo of a·2^−1
    end for                                             → Montgomery reduction
    if m ≥ N then
        m ← m − N
    end if
    return m
end function
```

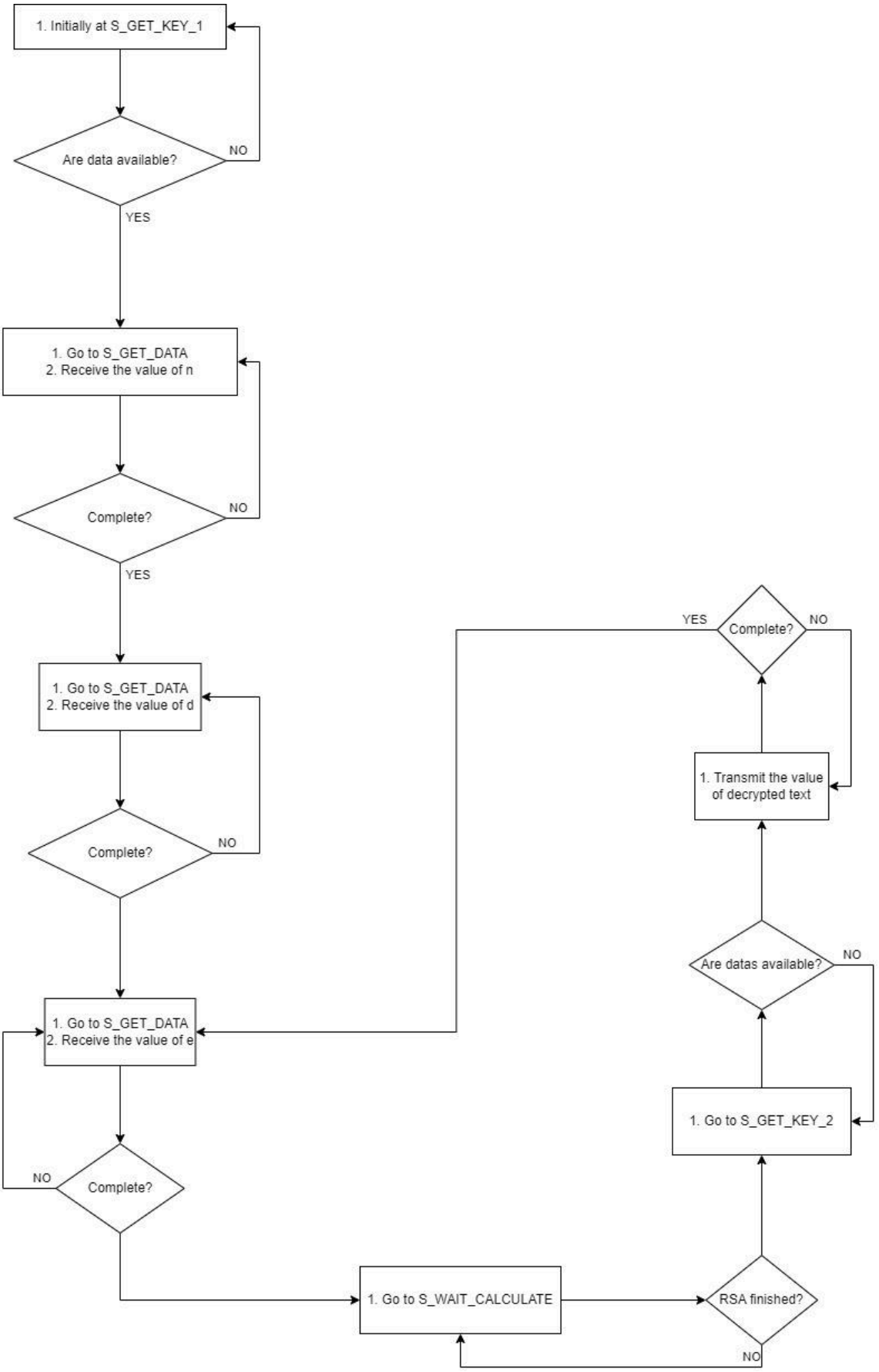
Calculates $a \times b \times 2^{-256} (mod\ N)$

Overall Algorithm:

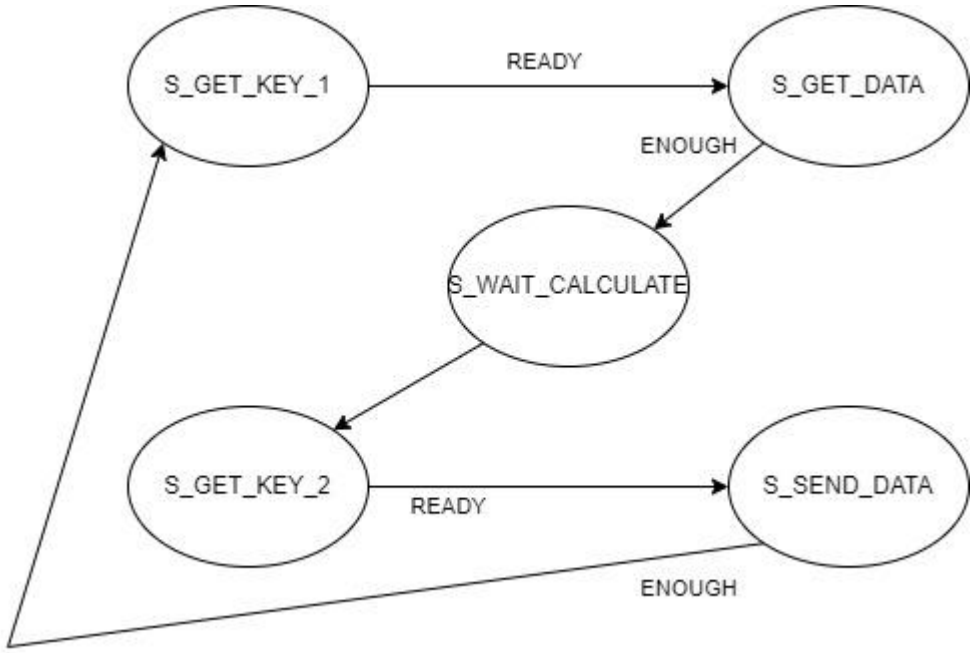
```
function RSA256Mont(N, y, d)
    t ← ModuloProduct(N, 2^256, y, 256)   ⟶  t = y * 2^256 (modN)
    m ← 1
    for i ← 0 to 255 do
        if i-th bit of d is 1 then
            m ← MontgomeryAlgorithm(N, m, t)   ⟶  m * t * 2^−256 (modN)
        end if
        t ← MontgomeryAlgorithm(N, t, t)       ⟶  t * t * 2^−256 (modN)
    end for
    return m
end function
```

Calculates $y^d \ (mod\ N)$

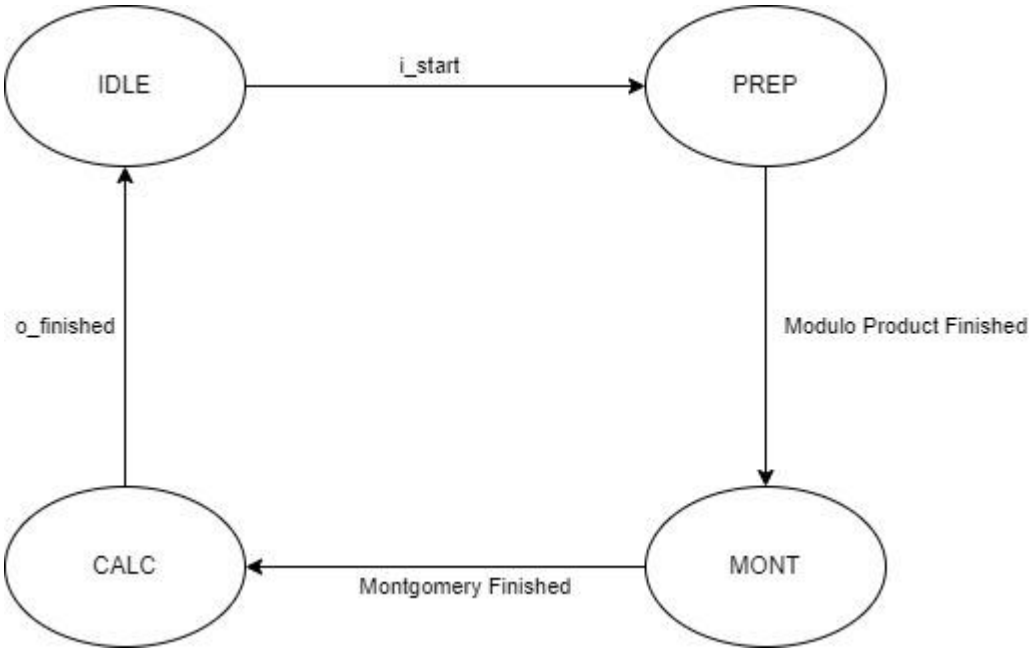
Algorithm of Wrapper:



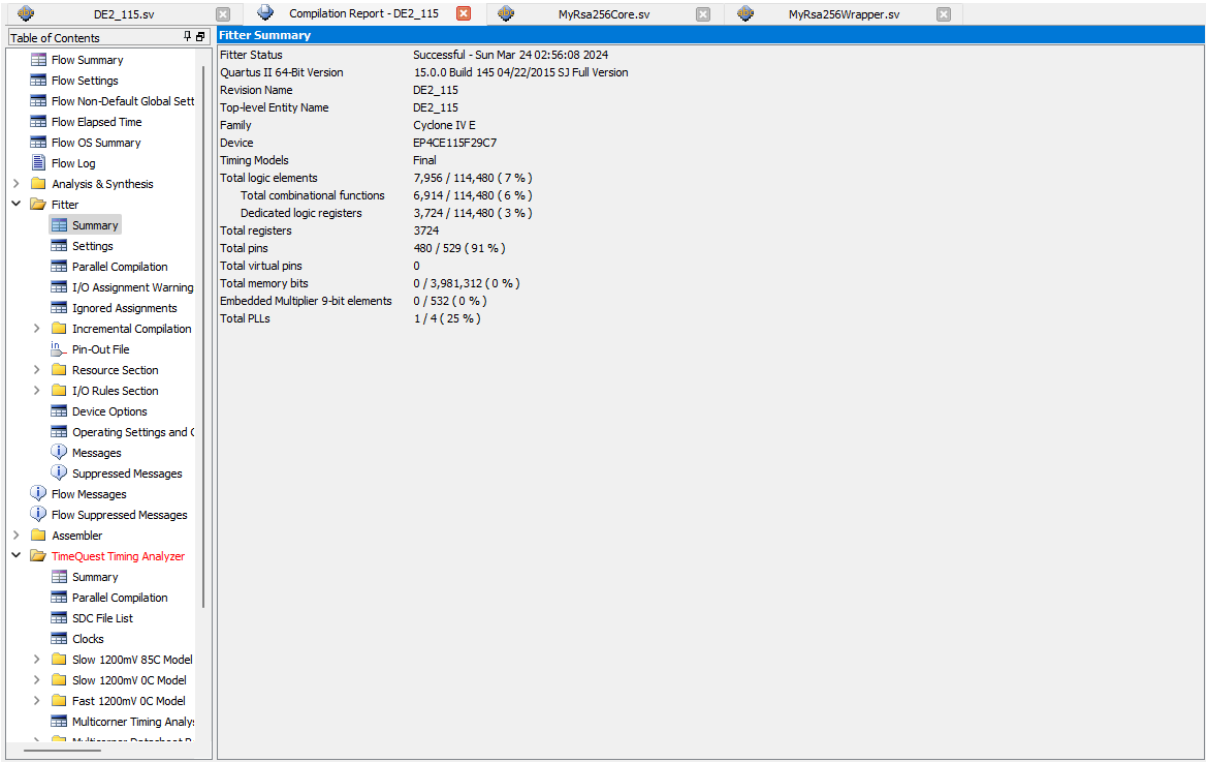
FSM of Wrapper:



FSM of RSA_Core:



Fitter Summary:



Timing Analyzer: