NCTU-EE IC LAB - Spring 2022

Lab06 Exercise

Design: RSA

Data Preparation

1. Extract test data from TA's directory:

   % tar xvf ~iclabta01/Lab06.tar

2. The extracted LAB directory contains:

   a. Practice/

   b. Exercise_SoftIP/

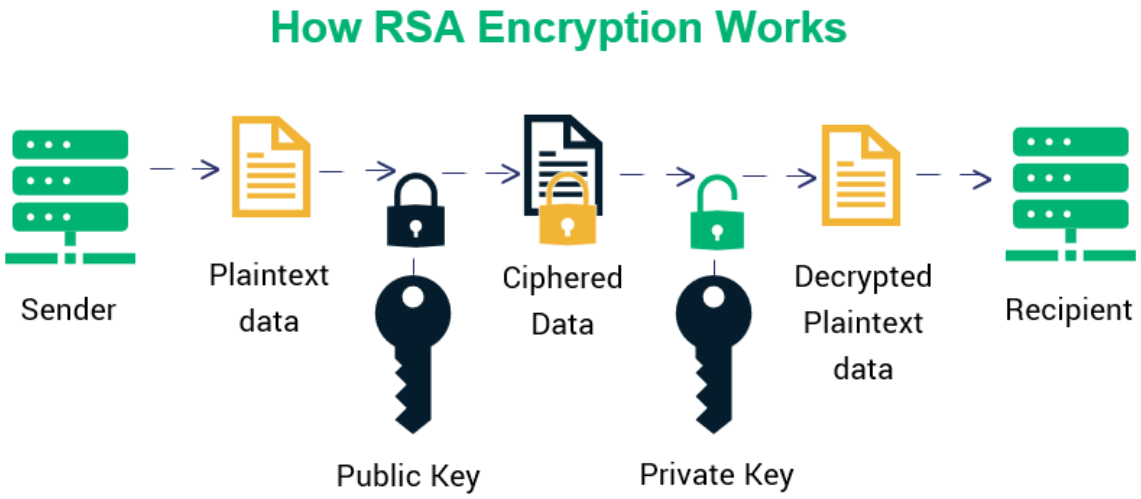
   c. Exercise/

RSA Cryptosystem

■ ***RSA***

RSA is a **public-key cryptosystem** that is widely used for secure data transmission, and is also one of the oldest. The acronym "RSA" comes from the surnames of **Ron Rivest, Adi Shamir and Leonard Adleman**, who publicly described the algorithm in 1977.

RSA involves a **public key** and a **private key**. The public key can be known by everyone and is used for encrypting messages. The intention is that **messages encrypted with public key can only be decrypted in a reasonable amount of time by using the private key**. Since the process of encryption and decryption use different keys, the RSA cryptosystem is also known as **asymmetric cryptosystem**.



■ ***Operation***

Suppose that **Bob** wants to send information to **Alice**, and they decides to use RSA algorithm. It includes four steps: **key generation, key distribution, encryption, and decryption**.

■ ***Key generation***

Alice can generate the keys for the RSA algorithm in the following way:

1. Choose two distinct **prime numbers** **p** and **q**.
2. Compute $N = pq$.
3. Compute **Euler totient function** $\varphi(N) = (p-1)(q-1)$, that is, the amount of integers that smaller and coprime with N.
4. Choose an integer **e** such that $1 < e < \varphi(N)$ and $gcd(e, \varphi(N)) = 1$.
5. Determine **d** such that $ed \equiv 1(mod\ \varphi(N))$.
6. Acquire **public key** $KU = \{e, N\}$ and **private key** $KR = \{d, N\}$.

Example

1. Choose two distinct prime numbers, $p = 61,\ q = 53$.
2. Compute $N = 61 \times 53 = 3233$.
3. Compute $\varphi(N) = (61-1)(53-1) = 3120$.
4. Choose $e = 17$. ($1 < e < \varphi(N),\ gcd(e, \varphi(N)) = 1$)
5. Compute $d = 2753$. (one of the possible answers that satisfy $ed \equiv 1(mod\ \varphi(N))$)
6. Acquire public key $KU = \{17, 3233\}$ and private key $KR = \{2753, 3233\}$.

2753×17 % 3120 = 1

■ ***Key distribution***

To enable Bob to send his encrypted messages, **Alice transmits her public key to Bob**. Alice's private key is never distributed.

■ ***Encryption***

After Bob obtains Alice's public key {e, N}, he can send a message M to Alice.

1. Turn M into an plaintext **m** such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a **padding scheme**.
2. Compute the ciphertext **c** by using Alice's public key, corresponding to:

$$c \equiv m^e (mod\ N)$$

Note that the **modular exponentiation** can be done quickly, even for very large numbers.

3. Transmit **c** to Alice.

■ ***Decryption***

Alice can recover **m** from **c** by using her private key $\{d, N\}$.

1. Compute $c^d \equiv (m^e)^d \equiv m(mod\ N)$.
2. Recover the original message M by reversing the padding scheme.

■ ***Cryptanalysis***

The security of RSA algorithm relies on the practical difficulty of factoring the product of two large prime numbers. In other words, the more difficult it is to factorize a very large integer, the more reliable the RSA algorithm is. If someone find out a fast factoring algorithm, the reliability of messages encrypted with RSA would be greatly reduced, but the possibility of finding such an algorithm is very low.

Today, only short encrypted messages of RSA can be brute-forced attacked. As long as the length of the keys is long enough, the information encrypted with RSA is practically unbreakable. **So far, there is no reliable way to crack a 1024-bit key**. But the 1024-bit key is threatened since an 829-bit key has been broken, it is recommended to use at least a 2048-bit key.

Besides, today's increasingly mature distributed computing technology and quantum computer theory, the security of RSA encryption has been challenged.

Design Description

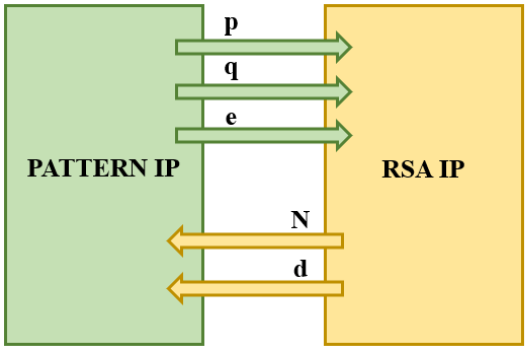
According to the above introduction, you need to **design your own soft IP to generate the private key**, and use it to **decrypt the encrypted messages given in each round in the top design**.

The summary of both design in this Exercise are as following:

Design	Definition
Soft IP	Given two distinct prime numbers **p**, **q**, and integer **e**, the soft IP design need to calculate the **private key**, that is, $KR = \{d, N\}$.
Top Design	Given **p**, **q**, **e**, and **8 encrypted messages**, the top design need to use **private key generated by above soft IP design** to **decrypt them all**.

■ ***Soft IP***

- Step-1: Given integers p, q, and e.
- Step-2: Calculate the private key, KR={d, N}.
- Note that **d** can be computed effectively by using the **extended Euclidean algorithm**. If the result is smaller than 0, plus $\varphi(n)$ to turn negative to positive.



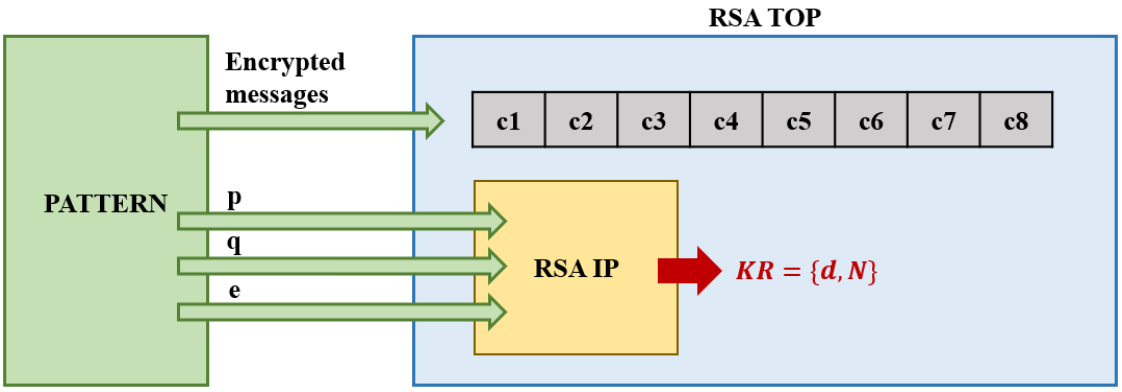
$$ed \equiv 1 (\mathrm{mod}\ \varphi(N))$$

$$ed + k\varphi(N) = 1$$

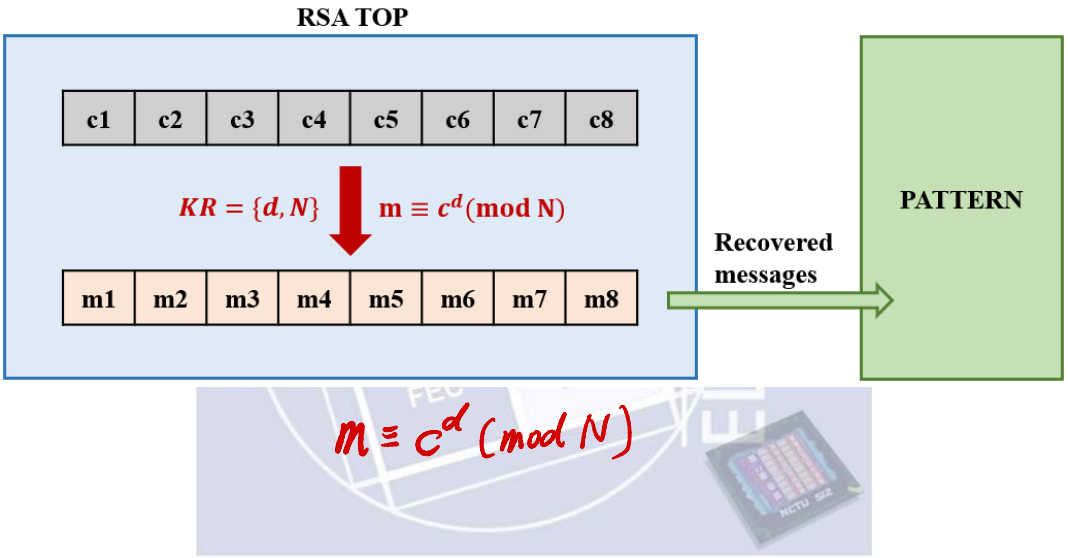
Use Extended Euclidean Algorithm to solve it

■ *Top Design*

- Step-1: Given integers p, q, e, and 8 encrypted messages, and use your own designed soft IP to calculate the private key.



- Step-2: Decrypt the 8 ciphertexts all with **modular exponentiation**.



Inputs and Outputs (Top Design)

■ The following are the definitions of input signals:

Input signal	Bit width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when input signals are valid.
in_p	4	The first prime number.
in_q	4	The second prime number.
in_e	8	The selected integer for public key.
in_c	8	The ciphertext.

■ The following are the definitions of output signals:

Output signal	Bit width	Definition
out_valid	1	High when input signals are valid.
out_m	8	The plaintext.

Inputs and Outputs (Soft IP)

■ The following are the definitions of input signals:

Input signal	Bit width	Definition
IN_P	IP_WIDTH	The first prime number.
IN_Q	IP_WIDTH	The second prime number.
IN_E	IP_WIDTH*2	The selected integer for public key.

■ The following are the definitions of output signals:

Output signal	Bit width	Definition
OUT_N	IP_WIDTH*2	The product of two prime numbers.
OUT_D	IP_WIDTH*2	Output the result for private key.

■ **Every private key that satisfy the specific relationship, $ed \equiv 1 (mod\ \varphi(n))$, is acceptable**.

Specifications (Top Design)

Top module

1. Top module name: **RSA_TOP** (design file name: **RSA_TOP.v**).
2. You can adjust your clock period by yourself, but the maximum period is 60ns. The precision of clock period is 0.1, for example, 4.5 is allowed, but 4.55 is not allowed.
3. The execution latency is limited to 10,000 cycles. The latency is the clock cycles between the falling edge of the last cycle of **in_valid** and the rising edge of the **out_valid**.
4. The total cell area should not be larger than 2,000,000 um^2.
5. The look-up table method is forbidden, and you need to use your own-designed soft IP (RSA_IP) in the top module. (TA will check your design)

Reset

6. It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) in your design, you may fail to reset signals.
7. The reset signal(**rst_n**) would be given only once at the beginning of simulation. All output signals should be reset after the reset signal is asserted.

in valid

8. **in_valid** will come after reset.

9. All input signals are synchronized at **negative edge** of the clock.
10. When **in_valid** is low, input is tied to unknown state.

out valid

11. **out_valid** should not be raised when **in_valid** is high.
12. The **out_valid** is limited to be high for only 8 cycles when the output value is valid.
13. All output signals should be synchronized at clock positive edge.
14. The TA's pattern will capture your output for checking at **clock negative edge**.
15. The **out_n** should be correct when **out_valid** is high.
16. The **out_n** should be reset after your **out_valid** is pulled down.
17. The next input pattern will come in 2~4 **negative edge of clock** after your **out_valid** falls.

Synthesis

18. In this lab, you should write your own **syn.tcl file**.
19. Use **top** wire load mode and **compile ultra**.
20. Use analyze + elaborate to read your design.
21. The input delay is set to 0.5*(clock period).
22. The output delay is set to 0.5*(clock period), and the output loading is set to 0.05.
23. The input delay of clk and rst_n should be zero.
24. The synthesis time should be less than 2 hours.
25. The synthesis result (syn.log) of data type **cannot** include any latches and error.
26. After synthesis, you can check **RSA_TOP.area** and **RSA_TOP.timing**. The area report is valid only when the slack in the end of timing report should be **non-negative** and the result should be **MET**.

Gate level simulation

27. The gate level simulation cannot include any timing violations without the notimingcheck command.

Supplement

28. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *latch* or *fail* otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
29. Don't write chinese comments or other language comments in the file you turned in.
30. Verilog commands //synopsys dc_script_begin, //synopsys dc_script_end
    //synopsys translate_off, //synopsys translate_on are only allowed during the usage of including and setting designware IPs, other design compiler optimizations are forbidden.
    Using the above commands are allowed, however any error messages during synthesize and simulation, regardless of the result will lead to failure in this lab.
    Any form of display or printing information in verilog design is forbidden. You may use this methodology during debugging, but the file you turn in should not contain any coding that is not synthesizable.

Specifications (Soft IP)

Top module

1. Top module name: **RSA_IP** (design file name: **RSA_IP.v**)
   Input signals : **IN_P, IN_Q, IN_E**
   Output signals : **OUT_N, OUT_D**
2. The clock period is 60ns. Finish calculating within **one** clock cycle.
3. You need to use **generate** to design this soft IP.
4. The look-up table method is forbidden. (TA will check your design)

Synthesis

5. Output loading is set to **0.05**.
6. Using **top** wire load mode and **compile ultra**.

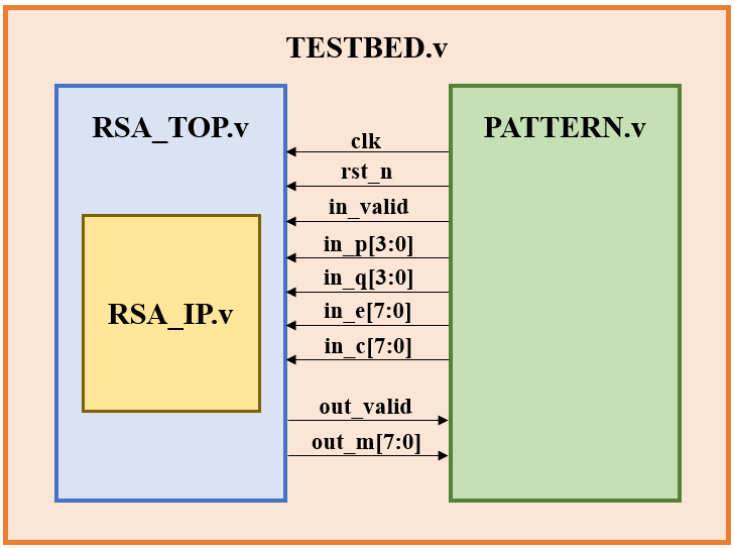
Supplement

7. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *latch* or *fail* otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
8. Don't write chinese comments or other language comments in the file you turned in.
9. Verilog commands //synopsys dc_script_begin, //synopsys dc_script_end
   //synopsys translate_off, //synopsys translate_on are only allowed during the usage of including and setting designware IPs, other design compiler optimizations are forbidden.
   Using the above commands are allowed, however any error messages during synthesize and simulation, regardless of the result will lead to failure in this lab.
   Any form of display or printing information in verilog design is forbidden. You may use this methodology during debugging, but the file you turn in should not contain any coding that is not synthesizable.

Soft IP Testing environment

```
//synopsys translate_off
`include "RSA_IP.v"
//synopsys translate_on

module RSA_IP_demo #(parameter WIDTH = 3) (
    // Input signals
    IN_P, IN_Q, IN_E,
    // Output signals
    OUT_N, OUT_D
);

// ================================================================
// Input & Output Declaration
// ================================================================
input  [WIDTH-1:0]   IN_P, IN_Q;
input  [WIDTH*2-1:0] IN_E;
output [WIDTH*2-1:0] OUT_N, OUT_D;

// ================================================================
// Soft IP
// ================================================================
RSA_IP #(.WIDTH(WIDTH)) I_RSA_IP ( .IN_P(IN_P), .IN_Q(IN_Q), .IN_E(IN_E), .OUT_N(OUT_N), .OUT_D(OUT_D) );


endmodule
```

Block diagram



Note

1. **Grading policy:**
   - Function Validity: 50% (RTL and Gate-level simulation correctness)
     - IP_WIDTH = 4
   - Performance: 30% (Area * Total latency time)
   - Soft IP function correctness: 20% (No second demo)
     - IP_WIDTH = 3

101944 × 60 = 6116640
112898 × 45 = 5080410
144269 × 35 = 5049415

   - The performance is determined by area and latency of your design. The less cost your design has, the higher grade you get.
   - There are many effective algorithms for both soft IP and top design. You can select your preferred one for better performance.
   - The grade of 2nd demo would be 30% off.

2. **Please upload the following files on new e3 platform before 23:59 p.m. on Apr. 7(Thu.):**
   - Top : RSA_TOP_iclabxxx.v
   - Soft IP : RSA_IP_iclabxxx.v
   - Cycle Time : CYCLE_iclabxxx.txt

   - xxx is your account number, i.e. RSA_TOP_iclab999.v、RSA_IP_iclab999.v、27.6_iclab999.txt
   - If the uploaded file violating the naming rule, you will get 5 deduct points.
   - **You need to fill the space in syn.tcl by yourself, but you don't need to upload to new e3 with your design.**

3. Template folders and reference commands:

01_RTL/ (RTL simulation) **./01_run**

02_SYN/ (Synthesis) **./01_run_dc**

(Check **latch** by searching the keyword "**Latch**" in **syn.log**)

(Check the design's timing in /Report/**RSA_TOP.timing**)

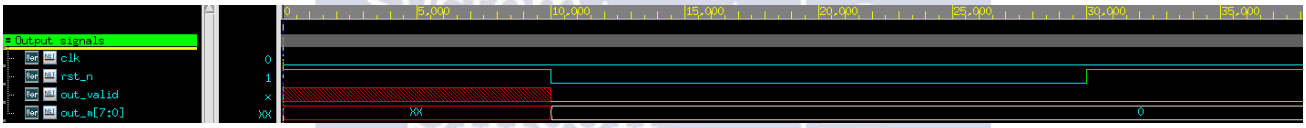
(Check the design's area in /Report/**RSA_TOP.area**)

03_GATE / (Gate-level simulation) **./01_run**

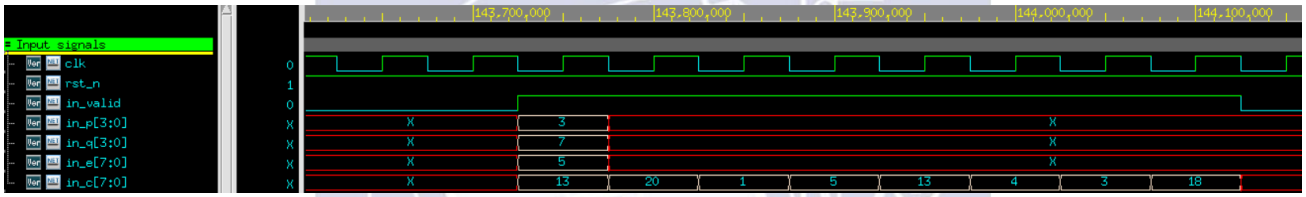
- You can key in **./09_clean_up** to clear all log files and dump files in each folder.

Sample Waveform

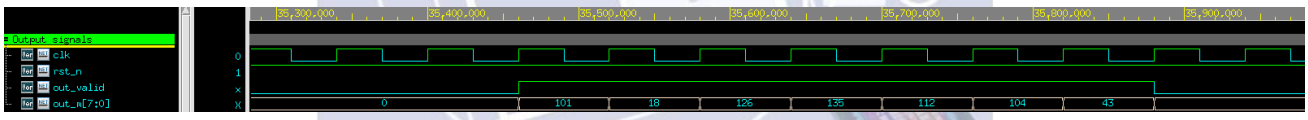
1. Asynchronous reset and active-low and reset all output



2. 8 cycles for input the information of each round(pattern)



3. 8 cycles for output the result of each round(pattern)



Reference

■ ***RSA Cryptosystem***

https://en.wikipedia.org/wiki/RSA_(cryptosystem)

■ ***Extended Euclidean Algorithm***

https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm

■ ***Modular Exponentiation***

https://en.wikipedia.org/wiki/Modular_exponentiation

Euclidean

$N = 55$

$p = 11 \quad q = 5 \quad \phi(N) = 40 \quad e$ choose 7

Step 1: $ed + k\phi(N) = 1$

$40x + 7y = 1$

先算 7 × 多少到 40 ⇒ 5

a = 40
b = 7

$40 \% 7 = 5$

$a = b$ ✗

$b = a \% b$

$40 = 5 \times 7 + 5$ $\qquad 5 = 40 - 5 \times 7$

$7 = 1 \times 5 + 2$ $\qquad 2 = 7 - 1 \times 5$

$5 = 2 \times 2 + 1$ $\qquad 1 = 5 - 2 \times 2$

(1) stop

Step 2: Back substitution

$1 = 5 - 2 \times 2$

$1 = 5 - 2(7 - 1 \times 5)$

$1 = 3(5) - 2 \times 7$

$1 = 3(40 - 5 \times 7) - 2(7)$

$1 = 3(40) - 17 \times 7$

$d = 40 - 17 = 23$

private key : (23, 55)

public key : (7, 55)

13 17 $N = 221$ $\phi(N) = 192$

$e = 17$ $\quad 192 = 11 \times 17 + 5$

$d = 113$ $\quad 17 = 3 \times 5 + 2$

$5 = 2 \times 2 + 1$

$1 = 5 - 2 \times 2$

$1 = 5 - 2(17 - 3 \times 5) = 7 \times 5 - 2 \times 17$

$1 = 7(192 - 11 \times 17) - 2 \times 17$

$1 = 7 \times 192 - 79 \times 17$

(reminder × quo)

1×2

↓ reminder = quo − quo

$1 + 2 \times 3 = 7$

$2 + 7 \times 11 = 79$

13 11 $\phi = 120$

$e = 13$

$120 = 9 \times 13 + 3$ $\quad 1 + 9 \times 4$

$13 = 4 \times 3 + 1$ $\quad 0 + 1 \times 4 = 4$

$120 = 1 \times 97 + 23$ $\quad 23 = 120 - 1 \times 97$

$97 = 4 \times 23 + 5$ $\quad 5 = 97 - 4 \times 23$

$23 = 4 \times 5 + 3$ $\quad 3 = 23 - 4 \times 5$

$5 = 1 \times 3 + 2$ $\quad 2 = 5 - 1 \times 3$

$3 = 1 \times 2 + 1$ $\quad 1 = 3 - 1 \times 2$

$1 \times b$

$(b, 0, 1)$

$$120 = 5 \times 23 + 5$$

$$23 = 4 \times 5 + 3$$

$$5 = 1 \times 3 + 2$$

$$3 = 1 \times 2 + 1$$

$$1 = 3 - 1 \times 2$$

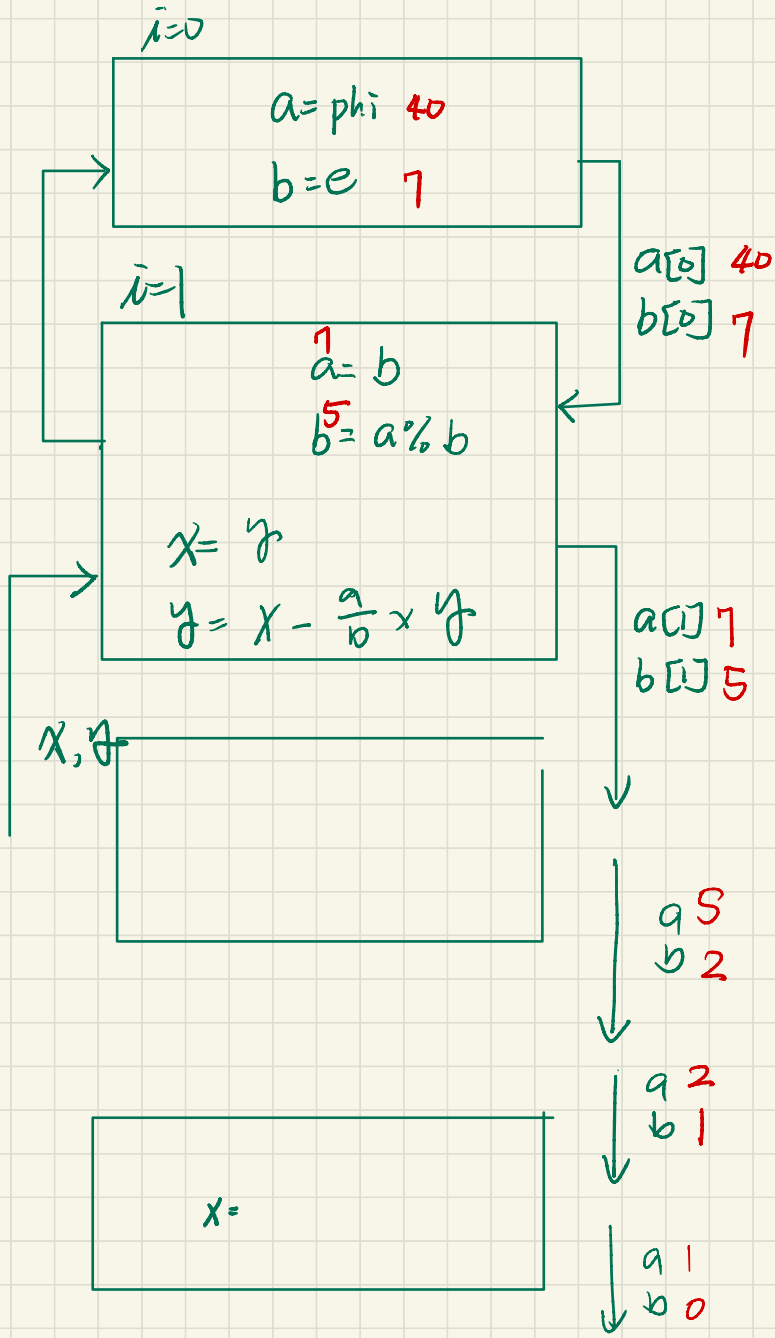
$$1 = 3 - 1 \times (5 - 1 \times 3) = 2 \times 3 - 1 \times 5$$

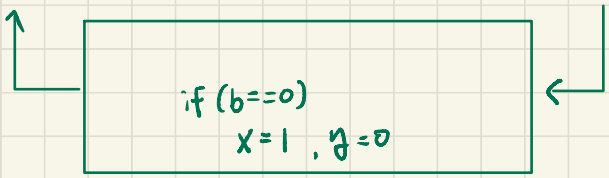
$$1 = 2 \times (23 - 4 \times 5) - 1 \times 5$$

$$= 2 \times 23 - 9 \times 5$$

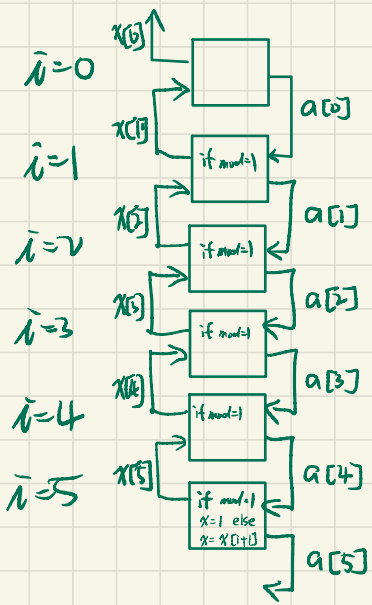
$$= 2 \times 23 - 9(120 - 5 \times 23)$$

$$= 47 \times 23 - 9 \times 120 = 1$$

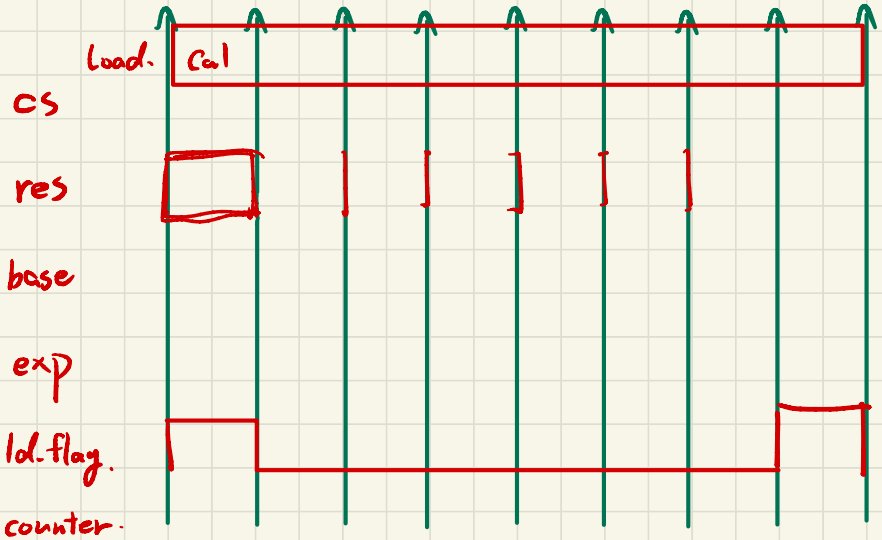

i=0
a= phi 40
b=e 7
a[0] 40
b[0] 7
i=1
7
a= b
5
b= a%b
x= y
y= x - a/b x y
a[1] 7
b[1] 5
x, y
a 5
b 2
a 2
b 1
x=
a 1
b 0



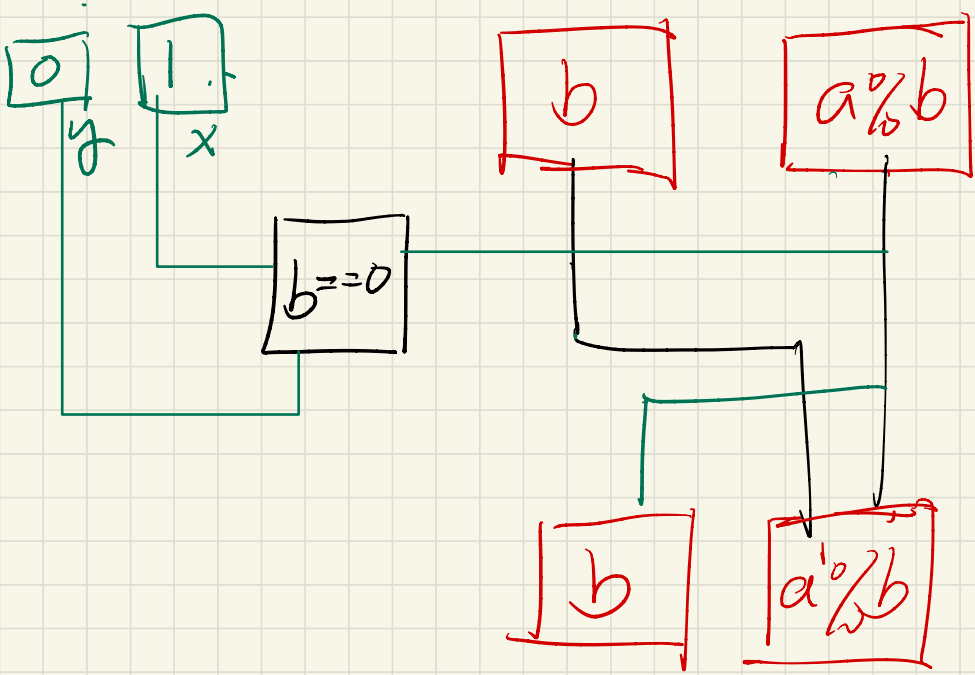
Assume WIDTH=3



if exp=[0] ,



if ld_flag


0
1
y
x
b==0
b
a%b
b
a%b