NYCU-EE IC LAB – Spring 2023

Lab06 Exercise

Design: Elliptic Curve Group Operation

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/

Design Introduction

■ ***Computation in Prime Field***

In mathematics, a finite field or Galois field is a field that contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules. The most common examples of finite fields *Fp* are given by the integers mod p when p is a prime number. Elements of the field *Fp* can be represented by integers 0,1, . . . , p−1. The two operations of the field are modular integer addition and integer multiplication modulo p.

The additive inverse of any element a is given by $\mathbf{a + (-a) = 0}\ \boldsymbol{mod\ p}$ and the multiplicative inverse of any nonzero element a is defined as $\boldsymbol{a * a^{-1} = 1\ mod\ p}$. Let's have a look at an example of a prime field.

Example

We consider the finite field $F_5 = \{0,1,2,3,4\}$. The tables below describe how to add and multiply any two elements, as well as the additive and multiplicative inverse of the field elements.

addition

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	$\textcircled{2}$	3

$\rightarrow (4+3)\%5=2$

additive inverse

$-0 = 0$
$-1 = 4$
$-2 = 3$
$-3 = 2$
$-4 = 1$

multiplication

×	0	1	2	3	4	a^{-1}
0	0	0	0	0	0	
1	0	1	2	3	4	$\rightarrow 1$
2	0	2	4	1	3	$\rightarrow 3$
3	0	3	1	4	2	$\rightarrow 2$
4	0	4	3	2	1	$\rightarrow 4$

multiplicative inverse

0^{-1} does not exist
$1^{-1} = 1$
$2^{-1} = 3$
$3^{-1} = 2$
$4^{-1} = 4$

Because the operation is calculated over *Fp*, all the operation result will be positive. If the result is smaller than 0, plus prime p to turn negative to positive.

The operation to find multiplicative inverse is also called *inversion operation*. Inverses can be efficiently computed by the extended Euclidean algorithm for integers. Therefore, the division in *Fp* will be seen as the inversion operation. Notice that the inversion of 0 does not exist and the inversion of 1 will be 1.

■ ***Elliptic Curve***

The elliptic curve over *Fp*, p > 3, is the set of all pairs (x,y) ∈ *Fp* which fulfill:

$$\boldsymbol{y^2 \equiv x^3 + a * x + b \bmod p}$$

together with an imaginary point of infinity O, and the condition:

$$\boldsymbol{a, b \in Fp \text{ and } 4 * a^3 + 27 * b^2 \neq 0 \bmod p}$$

Denote the group operation with the addition symbol "+". "Addition" means that given two points and their coordinates, say P = (x_P, y_P) and Q = (x_Q, y_Q), we have to compute the coordinates of a third point R such that:

- **Point addition P+Q:** This is the case where we compute R=P+Q and P≠Q. The construction works as follows: Draw a line through P and Q and obtain a third point of intersection between the elliptic curve and the line. Mirror this third intersection point along the x-axis. This mirrored point is, by definition, the point R. Figure 1. shows the point addition on an elliptic curve over the real numbers.
- **Point doubling P+P:** This is the case where we compute P+Q but P=Q. Hence, we can write R = P+P = 2P. We need a slightly different construction here. We draw the tangent line through P and obtain a second point of intersection between this line and the elliptic curve. We mirror the point of the second intersection along the x-axis. This mirrored point is the result R of the doubling. Figure 2. shows the doubling of a point on an elliptic curve over the real numbers.

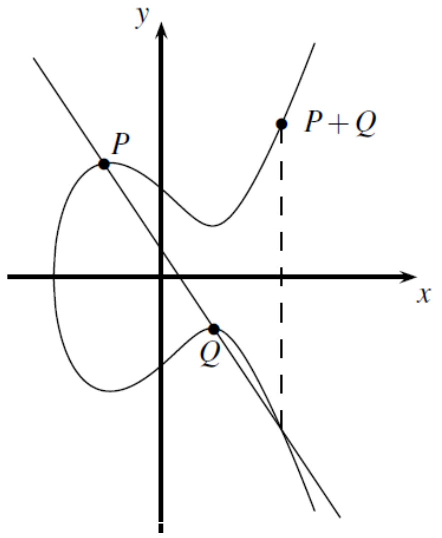


Fig 1.

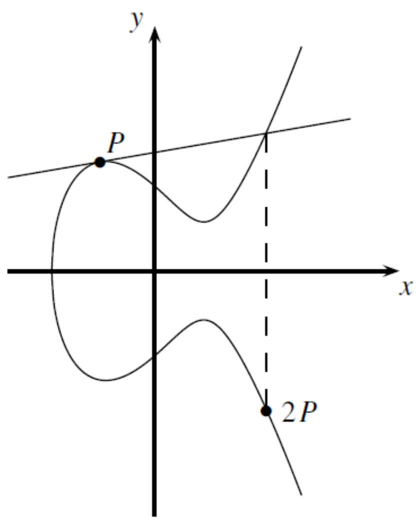


Fig 2.

In particular, we can take the curve equation from above, but we now consider it over prime fields *Fp* rather than over the real numbers. This yields the following analytical expressions for the group operation:

$$x_R = s^2 - x_P - x_Q \ mod\ p \quad and \quad y_R = s(x_P - x_R) - y_P \ mod\ p$$

where

$$s = \begin{cases} \dfrac{y_Q - y_P}{x_Q - x_P} \ mod\ p \ \ if\ P \neq Q\ (point\ addition) \\ \dfrac{3 * x_P^2 + a}{2 * y_P} mod\ p \ \ if\ P = Q\ (point\ doubling) \end{cases}$$

According the group definition, we also define the additive inverse −P of any group element P as:

$$\mathrm{P} + (-\mathrm{P}) = \mathcal{O} \ \text{ and } \ -\mathrm{P} = (x_P, \mathrm{p} - y_p)$$

which O means an abstract point at infinity. To avoid this situation, we will design the pattern which will not occur that $x_P = x_Q$ and $y_Q = p - y_p$ during operation in this lab. E.g (0,6) and (0,11) when p=17.

Example

Given point P=(5,1) and Q=(5,1) and the curve:

$$\boldsymbol{E: y^2 \equiv x^3 + 2 * x + 2 \ mod\ 17}$$

We want to find the coordinate of R such that R=P+Q.

$$\mathrm{R} = \mathrm{P} + \mathrm{Q} = 2\mathrm{P} = \mathrm{P} + \mathrm{P} = (5,1) + (5,1) = (x_R, y_R)$$

$$s = \frac{3 * x_P^2 + a}{2 * y_P} = (2 * 1)^{-1}(3 * 5^2 + 2) = 2^{-1} * 9 \equiv 9 * 9 \equiv 13 \ mod\ 17$$

$$x_R = s^2 - x_P - x_Q \ mod\ p = 13^2 - 5 - 5 \ mod\ 17 \equiv 159 \ mod\ 17 \equiv 6 \ mod\ 17$$

$$y_R = s(x_P - x_R) - y_P \ mod\ p = 13(5 - 6) - 1 \ mod\ 17 \equiv -14 \ mod\ 17 \equiv 3 \ mod\ 17$$

$$R = (x_R, y_R) = 2\mathrm{P} = \mathrm{P} + \mathrm{P} = (5,1) + (5,1) = (6,3)$$

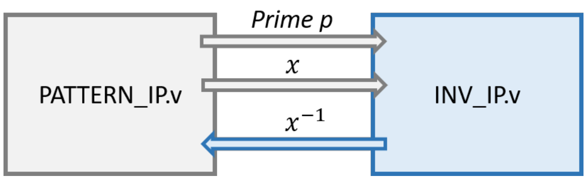
Design Description

Design	Definition
Soft IP	Given two input integers. One of the inputs will be the prime number (p>3) and another one will be an integer x over the *Fp* (except 0). The soft IP need to calculate x^{-1} (the inversion of x) in prime field p.
Top Design	Given two points P and Q, the top design needs to calculate the operation result with the inversion result from soft IP.

■ ***Soft IP***

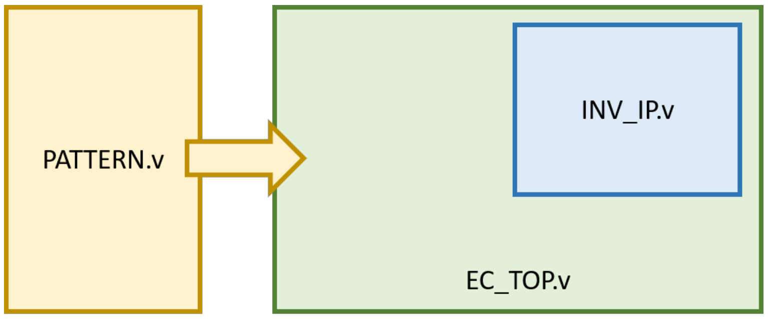
輾転相除法

Note that x^{-1} can be computed effectively by using the **extended Euclidean algorithm**.

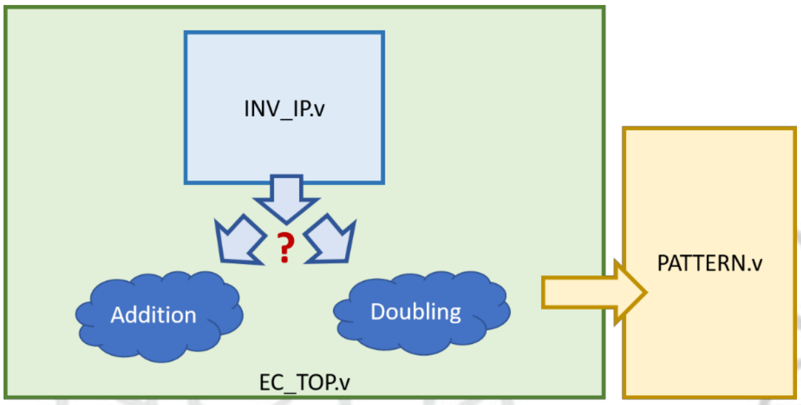


■ ***Top Design***

Step-1: Given two points P and Q, prime p and parameter a, and use your own designed soft IP to calculate the inversion result.



Step-2: Use the inversion result to calculate the output coordinate.



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_Px	6	The x-coordinate of the first point over prime field.
in_Py	6	The y-coordinate of the first point over prime field.
in_Qx	6	The x-coordinate of the second point over prime field.
in_Qy	6	The y-coordinate of the second point over prime field.
in_prime	6	Elliptic curve over prime filed p.
in_a	6	The parameter of the curve over prime field.

- Because parameter b will not impact on the calculating result, will not be given in this lab.
- The input value of in_Px, in_Py, in_Qx, in_Qy and in_a will be an integer over the *Fp* (except 0).

■ The following are the definitions of output signals:

Output signal	Bit width	Definition
out_valid	1	High when output signals are valid.
out_Rx	6	Output the x-coordinate over prime field.
out_Ry	6	Output the y-coordinate over prime field.

Inputs and Outputs (Soft IP)

■ The following are the definitions of input signals:

Input signal	Bit width	Definition
IN_1	IP_WIDTH	The first input to do inversion in Prime Fields
IN_2	IP_WIDTH	The second input to do inversion in Prime Fields

- One of the inputs will be the prime number (p>3) and another one will be an integer over the *Fp* (except 0).

■ The following are the definitions of two parameters:

Parameter	Bit width	Definition
IP_WIDTH	-	The length of input binary bits.

- If the result is smaller than 0, plus prime p to turn negative to positive.

Example: #(4)

1. 4 means input bits which the largest number is 15.
2. Of course, the truly output bits should be 4 bits. (you can set your output bits by using the parameter.)
3. **There won't be illegal case #(0)**.

■ The following are the definitions of output signals:

Output signal	Bit width	Definition
OUT_INV	IP_WIDTH	Output of the inversion result.

Specifications (Top Design)

Top module

1. Top module name: **EC_TOP** (design file name: **EC_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, 40.5 is allowed, but 40.55 is not allowed.
3. The execution latency is limited to 10,00 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 (INV_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 1 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_Rx** and **out_Ry** should be correct when **out_valid** is high.
16. The next input pattern will come in 2~4 **negative edge of clock** after your **out_valid** falls.

Synthesis

17. In this lab, you should write your own **syn.tcl file**.
18. Use **top** wire load mode and **compile_ultra**.
19. Use analyze + elaborate to read your design.
20. The input delay is set to 0.5*(clock period).
21. The output delay is set to 0.5*(clock period), and the output loading is set to 0.05.
22. The input delay of clk and rst_n should be zero.
23. The synthesis time should be less than 2 hours.
24. The synthesis result (syn.log) of data type **cannot** include any latches and error.
25. After synthesis, you can check **EC_TOP.area** and **EC_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

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

Supplement

27. In this lab, you are NOT allowed to use Designware IP.
28. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *pass*, *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: **INV_IP** (design file name: **INV_IP.v**)
   Input signals: **IN_1,   IN_2**
   Output signals: **OUT_INV**
   Two parameters : **IP_WIDTH**
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)

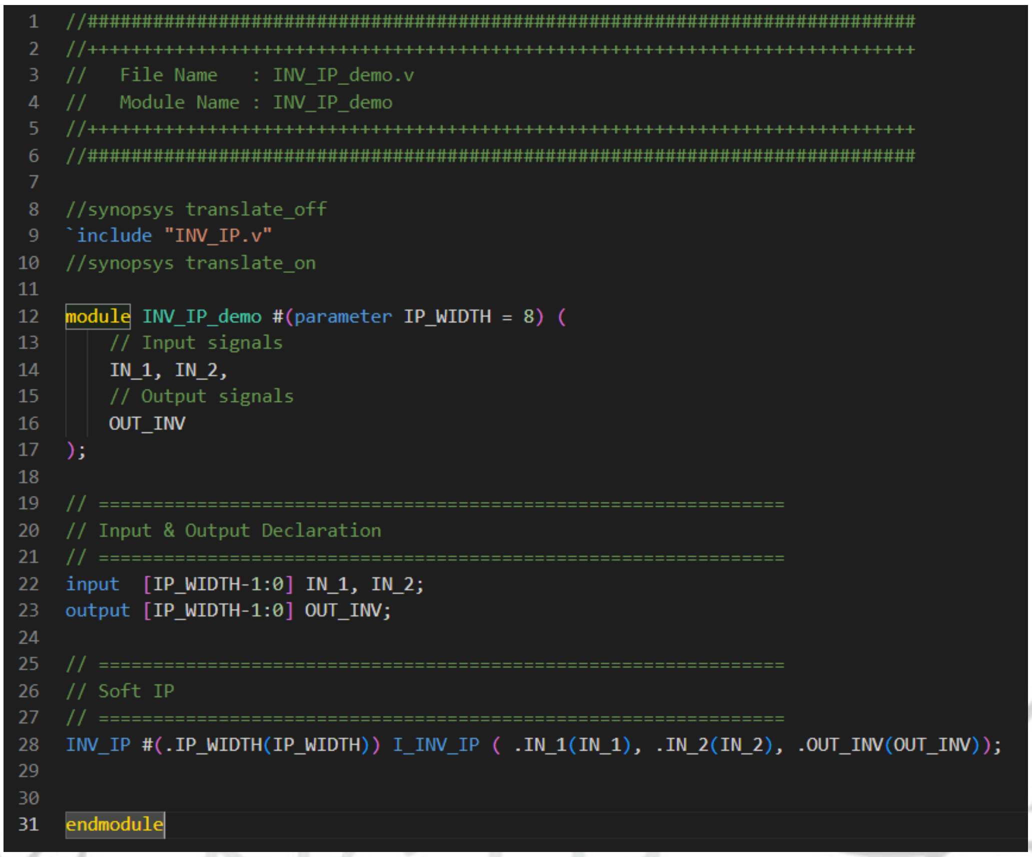
Synthesis

5. Output loading is set to **0.05**.
6. Using **top** wire load mode and **compile_ultra**.
7. Use analyze + elaborate to read your design.

Supplement

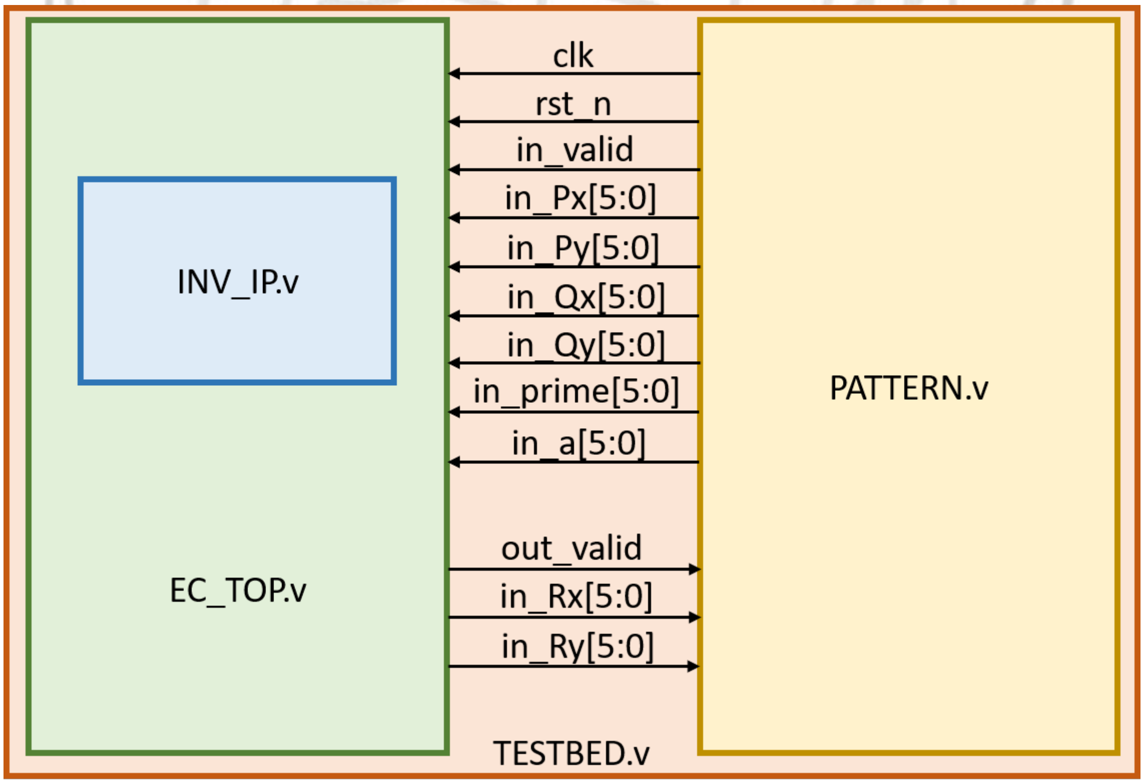
8. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *pass*, *latch* or *fail* otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
9. Don't write chinese comments or other language comments in the file you turned in.
10. 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



```verilog
//##############################################################################
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//   File Name   : INV_IP_demo.v
//   Module Name : INV_IP_demo
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//##############################################################################

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

module INV_IP_demo #(parameter IP_WIDTH = 8) (
    // Input signals
    IN_1, IN_2,
    // Output signals
    OUT_INV
);

// ===============================================================
// Input & Output Declaration
// ===============================================================
input  [IP_WIDTH-1:0] IN_1, IN_2;
output [IP_WIDTH-1:0] OUT_INV;

// ===============================================================
// Soft IP
// ===============================================================
INV_IP #(.IP_WIDTH(IP_WIDTH)) I_INV_IP ( .IN_1(IN_1), .IN_2(IN_2), .OUT_INV(OUT_INV));


endmodule
```

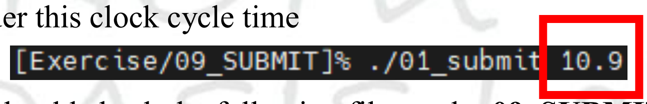
Block diagram



Note

1. **Grading policy:**
   - Function Validity: 50% (RTL and Gate-level simulation correctness)
     - ✓ Top design
   - Performance: 30% $Area^2 * (Total\ Latency\ Time + 1)$
     - ✓ Top design
   - Soft IP function correctness: 20% (No second demo)
     - ✓ IP_WIDTH = 7-bits: 10%
     - ✓ IP_WIDTH = 6-bits: 5%
     - ✓ IP_WIDTH = 5-bits: 5%
   - The performance is determined by area and latency of your design. The less cost your design has, the higher grade you get.
   - The grade of 2nd demo would be 30% off.

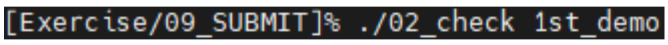
2. **Please submit your files under 09_SUBMIT. (09_SUBMIT** is under **Exercise/.)**
   - **1st demo : before 12:00 p.m. on Apr. 3(Mon.)**
     2nd demo : before 12:00 p.m. on Apr. 5(Wed.)
   - In this lab, you can adjust your clock cycle time. Consequently, make sure to key in your clock cycle time after the command like the figure below. It's means that the TA will demo your design under this clock cycle time



```
[Exercise/09_SUBMIT]% ./01_submit 10.9
```

   - After that, you should check the following files under **09_SUBMIT/Lab06_iclabXXX/**
     - Top : EC_TOP_iclabxxx.v
       Soft IP : INV_IP_iclabxxx.v
       Cycle Time : CYCLE_iclabxxx.txt
     - xxx is your account number, i.e. EC_TOP_iclab999.v、INV_IP_iclab999.v、30.6_iclab999.txt
     - If you miss any files on the list, you will fail this lab.
     - If the uploaded file violating the naming rule, you will get 5 deduct points.

   - Then use the command like the figure below to check the files are uploaded or not.

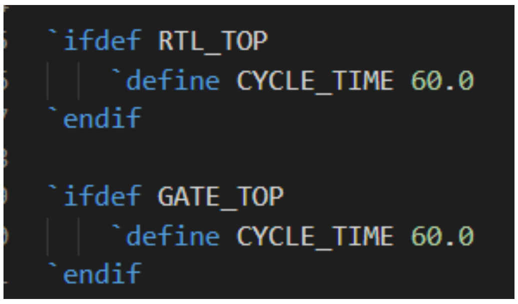


```
[Exercise/09_SUBMIT]% ./02_check 1st_demo
```

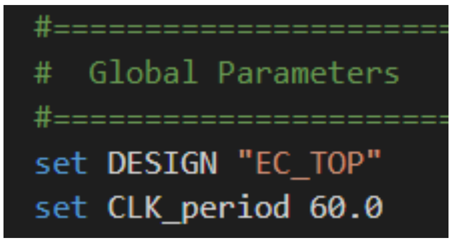
3. **Template folders and reference commands:**

   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/EC_TOP.timing)
   (Check the design's area in /Report/EC_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.
- You should make sure the **three clock period values identical** in 00_TESTBED/PATTERN.v and 02_SYN/syn.tcl



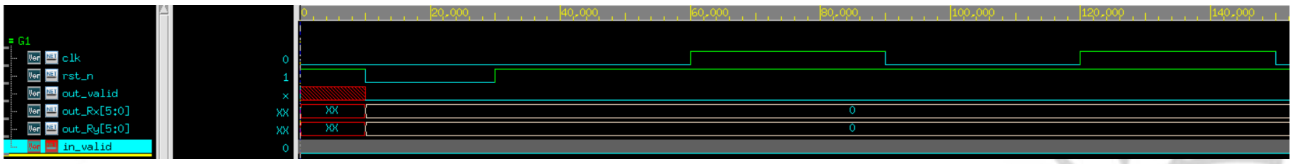
```
`ifdef RTL_TOP
    `define CYCLE_TIME 60.0
`endif

`ifdef GATE_TOP
    `define CYCLE_TIME 60.0
`endif
```



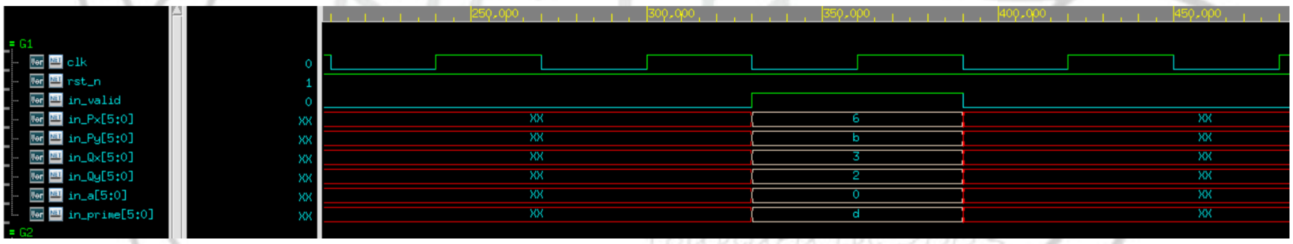
```
#=====================
#  Global Parameters
#=====================
set DESIGN "EC_TOP"
set CLK_period 60.0
```

Sample Waveform

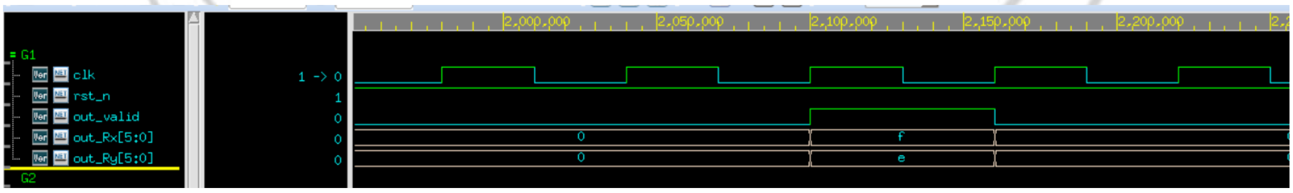
1. Asynchronous reset and active-low and reset all output.



2. 1 cycle for input the information of each round/pattern.



3. 1 cycle for output the result of each round/pattern.



- The **out_Rx** and **out_Ry** should be correct when **out_valid** is high, that is, **both signals will be checked at every clock negative edge**.

Appendix: Modular Arithmetic

Modular arithmetic is a system of arithmetic for integers, which considers the remainder. For example, since both 15 and -9 leave the same remainder 3 when divided by 12, we say that:

$$15 \equiv -9 \bmod 12$$

Properties of addition in modular arithmetic:

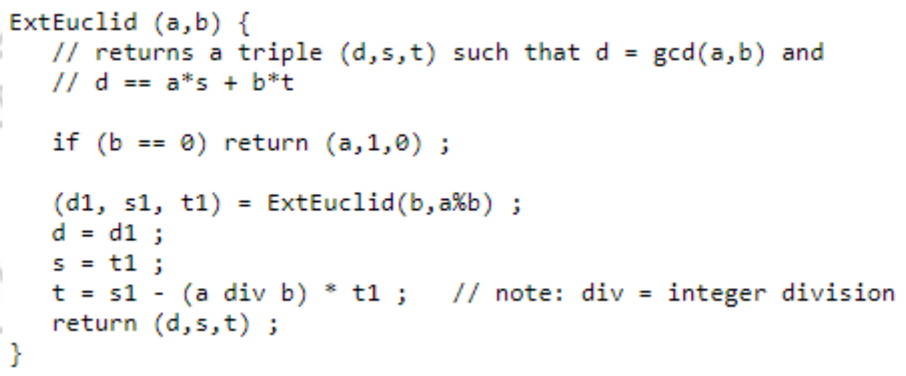
1. If $a + b = c$, then $a \ (\mathrm{mod}\ N) + b \ (\mathrm{mod}\ N) \equiv c \ (\mathrm{mod}\ N)$.
2. If $a \equiv b \ (\mathrm{mod}\ N)$, then $a + k \equiv b + k \ (\mathrm{mod}\ N)$ for any integer k.
3. If $a \equiv b \ (\mathrm{mod}\ N)$ and $c \equiv d \ (\mathrm{mod}\ N)$, then $a + c \equiv b + d \ (\mathrm{mod}\ N)$.
4. If $a \equiv b \ (\mathrm{mod}\ N)$, then $-a \equiv -b \ (\mathrm{mod}\ N)$

Appendix: Extended Euclidean algorithm

In arithmetic and computer programming, the extended Euclidean algorithm is an extension to the Euclidean algorithm, and computes, in addition to the greatest common divisor (gcd) of integers a and b, which are integers x and y such that:

$$ax + by \equiv \gcd(a, b)$$

Here's the pseudo-code for the Extended Euclid algorithm:



```
ExtEuclid (a,b) {
   // returns a triple (d,s,t) such that d = gcd(a,b) and
   // d == a*s + b*t

   if (b == 0) return (a,1,0) ;

   (d1, s1, t1) = ExtEuclid(b,a%b) ;
   d = d1 ;
   s = t1 ;
   t = s1 - (a div b) * t1 ;   // note: div = integer division
   return (d,s,t) ;
}
```

Ref: https://redirect.cs.umbc.edu/~chang/cs203.s09/exteuclid.shtml

Reference

- ***Elliptic Curve***
  https://en.wikipedia.org/wiki/Elliptic_curve
- ***Modular Arithmetic***
  https://en.wikipedia.org/wiki/Modular_arithmetic
- ***Extended Euclidean Algorithm***
  https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm