LAB 06
Exercise

2023.04.19

Presenter: Tzu-Yun, Yen

Outline

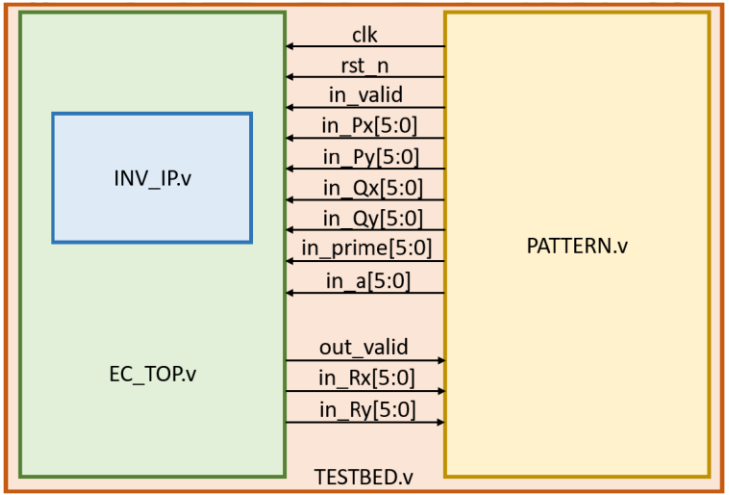
- Topic Review
- Soft IP
  - Extended Euclidean Algorithm
  - Design
  - Optimization
- Top Module
  - Data Flow Graph
  - Optimization

Topic Review

- Given the parameter a of an elliptic curve and two points P and Q on the curve, the task is to find the third point R s.t. P+Q+R=O, where O means an abstract point at infinity (無窮遠點).
- All computations should be performed in a specific prime field (Fp).
- Since we need to do the division over Fp, we need to find the inverse of the field elements, inverses can be efficiently computed by using the extended Euclidean algorithm (輾轉相除法).

Topic Review

- The design can be derived into two part:
  1. Soft IP – compute the inverse of the field element.
  2. Top Module – compute the coordinate of R.



Soft IP

Soft IP

- The inverse of the field element b with the prime number a is defined as:

$$b * b^{-1} \equiv 1 \mod a$$

The congruence relation may be rewritten as:

$$b * y = 1 + a * x$$
$$\Rightarrow a * x' + b * y = 1$$

- We can use Extended Euclidean Algorithm to find the inverse of b.
(i.e. the value of y)

Extended Euclidean Algorithm

- Extended Euclidean Algorithm Example
  - a = 113, b = 30
  - When r = 1, we get x = -13, y = 49 ⇒ inverse = 49

index	a	b	q	r	x	y
					1	0
					0	1
0	113	30	113 / 30 = 3	113 % 30 = 23	1 - 3 * 0 = 1	0 - 3 * 1 = -3
1	30	23	30 / 23 = 1	30 % 23 = 7	0 - 1 * 1 = -1	1 - 1 * -3 = 4
2	23	7	23 / 7 = 3	23 % 7 = 2	1 - 3 * -1 = 4	-3 - 3 * 4 = -15
3	7	2	7 / 2 = 3	7 % 2 = 1	-1 - 3 * 4 = -13	4 - 3 * -15 = 49

a*x + b*y = 113 * -13 + 30 * 49 = 1

Verilog Code

compute a, b

```
always @(*) begin
    a[i] = b[i-1];
    b[i] = r[i-1];
end
```

compute q, r

```
always @(*) begin
    q[j] = a[j] / b[j];
    r[j] = a[j] % b[j];
end
```

compute y

```
always @(*)  y[k] = y[k-2] + q[k] * y[k-1];
```

select output

```
always @(*) begin
    if(b[0] == 1)       temp_out = 1;
    else if(r[0] == 1)  temp_out = y[0];
    else if(r[1] == 1)  temp_out = y[1];
    else if(r[2] == 1)  temp_out = y[2];
    else if(r[3] == 1)  temp_out = y[3];
    else                temp_out = y[4];
end
```

Problems in Soft IP Design

- How many iterations do we need to compute inverter?
  - IP_WIDTH = 5 ⇒ 5 iters
  - IP_WIDTH = 6 ⇒ 6 iters
  - IP_WIDTH = 7 ⇒ 7 iters
- How many bits are required for each variable?
  - a, b, q, r ⇒ IP_WIDTH bits (unsigned)
  - y ⇒ IP_WIDTH + 1 bits (signed)

Optimization 1

- Observation: The sign of y is reversed at each iteration.
- Change the calculation of y to

$$y[k] = y[k-2] + q[k] * y[k-1]$$

⇒ the bit width of y can be reduced by one. (signed → unsigned)

y
0
1
0 - 3 * 1 = -3
1 - 1 * -3 = 4
-3 - 3 * 4 = -15
4 - 3 * -15 = 49

y	neg
0	
1	
0 + 3 * 1 = 3	1
1 + 1 * 3 = 4	0
3 + 3 * 4 = 15	1
4 + 3 * 15 = 49	0

Optimization 2

- Observation
  - The inverse of b is equal to the negative of the inverse of prime - b.
  - One more iteration is required if b > prime/2.

```
gcd(113, 30) = 1 | iter = 4, y = 49
gcd(113, 83) = 1 | iter = 5, y = -49
```

- Limit the field element b between 1 and prime/2
  ⇒ the iteration can be reduced by one.

Optimization 3

- For IP_WIDTH = 6, set each variable’s bit width separately.

index	a	b	q	r
0	6	5	5	5
1	5	5	4	4
2	5	4	4	3
3	4	3	3	2
4	3	2	1	1

- Since the quotient of the 5th iteration can only be 1, actually, we don't need to do the 5th division

  ⇒ only need 4 dividers(/), 4 remainders(%), 3 multipliers(*) and 5 adders(+)

After Optimization

- How many iterations do we need to compute?
  - IP_WIDTH = 5 ⇒ 4 iters
  - IP_WIDTH = 6 ⇒ 4 iters (+ 1adder)
  - IP_WIDTH = 7 ⇒ 6 iters
- How many bits are required for each variable (a, b, q, r, y)?
  - for IP_WIDTH = 5 or IP_WIDTH = 7 ⇒ IP_WIDTH bits (unsigned)
  - for IP_WIDTH = 6 ⇒ set separately

Top Module

Top Module

- The elliptic curve over Fp is defined as

$$y^2 \equiv x^3 + a * x + b \mod p$$

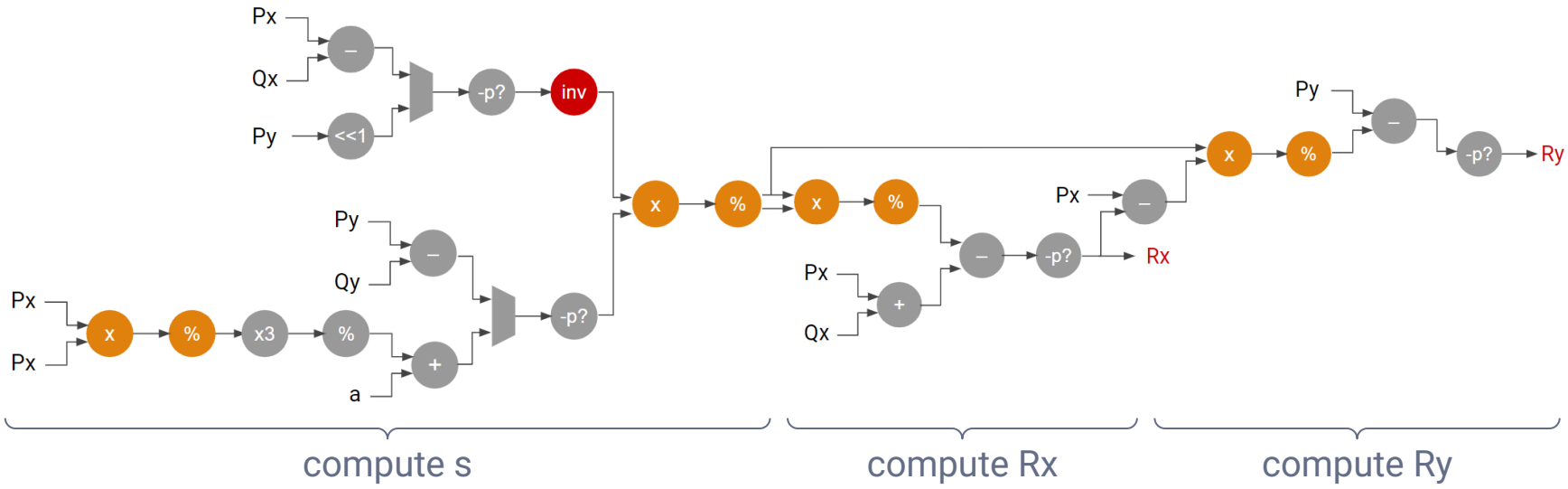
- The expressions for the group operation over prime field

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

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

Data Flow Graph

- The data flow graph is shown as below

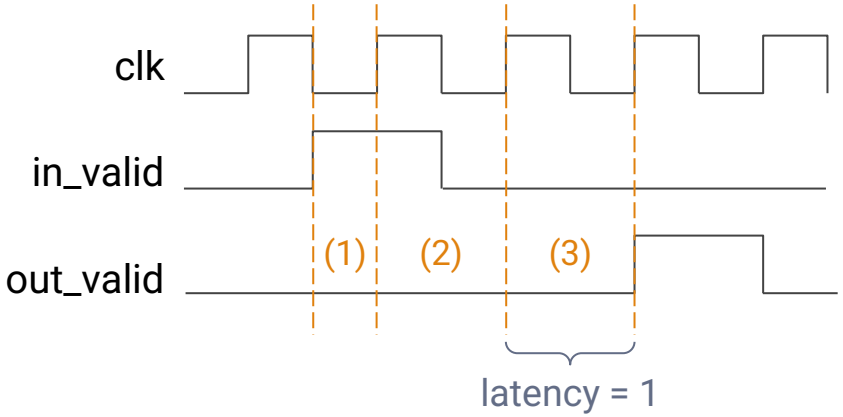


Pipeline

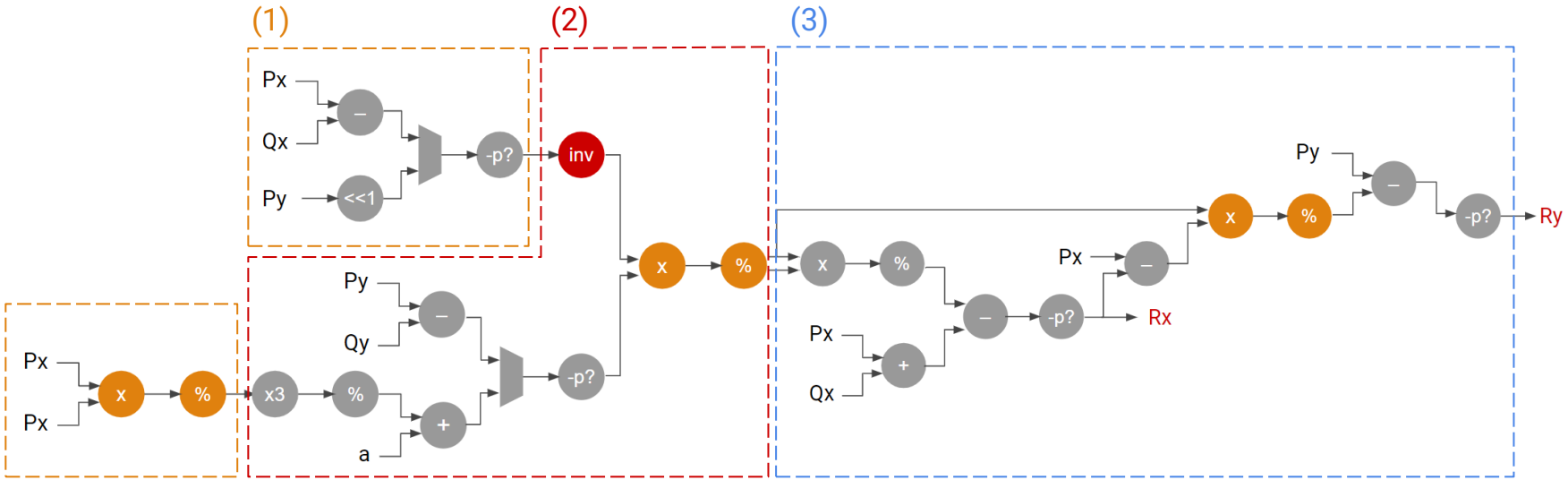
- Theoretically
  - pros: cycle time ⇓ & area ⇓ (∵ hardware reuse)
  - cons: latency ⇑
- Pratically
  - Cycle time is limited by the longest data path.
    ⇒ Too many stage of pipeline will cause pipeline imbalance.
  - The smaller cycle time, the larger components that Design Compiler will choose.
  - We need extra muxs to select the input of reused hardware.
    ⇒ The decrease in area is not proportional to the increase in latency.

Optimization

- Strategies
  - Divided into **2.5** pipeline stages
  - Only reuse one multiplier and one remainder in each pipeline.



After Optimization



The End

Thanks for listening :)