Arithmetic Operations over $\mathbb{F}_p$

Martin Novotný, Róbert Lórencz

Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
©Martin Novotný, Róbert Lórencz 2017

MI-BHW.16 Security and Hardware
Summer Semester

Outline

- Elliptic Curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, Necessary Operations
- Arithmetic Operations over $\mathbb{F}_p$
- Arithmetic Operations over $\mathbb{F}_p$ in Montgomery Domain

Outline

- Elliptic Curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, Necessary Operations
- Arithmetic Operations over $\mathbb{F}_p$
- Arithmetic Operations over $\mathbb{F}_p$ in Montgomery Domain

EC over $\mathbb{F}_p$

Weierstrass equation

$$y^2 \equiv x^3 + ax + b \mod p,$$

where $4a^3 + 27b^2 \not\equiv 0 \mod p$

EC over $\mathbb{F}_{2^m}$

Weierstrass equation

$$y^2 + xy \equiv x^3 + ax^2 + b \mod F(\alpha),$$

where $b \neq 0$

EC over $\mathbb{F}_p$ — point addition (basic operation)

Let $R = P + Q$, where $R = [x_R, y_R]$, $P = [x_P, y_P]$ a $Q = [x_Q, y_Q]$. Then

$$x_R = \lambda^2 - x_P - x_Q \mod p$$
$$y_R = (x_Q - x_R) \cdot \lambda - y_Q \mod p$$

where

$$\lambda = \begin{cases} \dfrac{y_P - y_Q}{x_P - x_Q} \mod p & \text{iff } P \neq Q \text{ (addition)} \\ \dfrac{3x_Q^2 + a}{2y_Q} \mod p & \text{iff } P = Q \text{ (doubling)} \end{cases}$$

Necessary operations over $\mathbb{F}_p$

- addition, subtraction: $x_Q - x_R = x_Q + (-x_R)$
- multiplication: $(x_Q - x_R) \cdot \lambda$
- inversion: $\frac{y_P - y_Q}{x_P - x_Q} = (y_P - y_Q) \cdot (x_P - x_Q)^{-1}$
- squaring – via multiplication: λ^2

EC over $\mathbb{F}_{2^m}$ — point addition (basic operation)

Let $R = P + Q$, where $R = [x_R, y_R]$, $P = [x_P, y_P]$ a $Q = [x_Q, y_Q]$. Then

$$x_R = a + \lambda^2 + \lambda + x_P + x_Q$$
$$y_R = (x_Q + x_R) \cdot \lambda + x_R + y_Q$$

where

$$\lambda = \begin{cases} \dfrac{y_P + y_Q}{x_P + x_Q} & \text{iff } P \neq Q \text{ (addition)} \\ x_Q + \dfrac{y_Q}{x_Q} & \text{iff } P = Q \text{ (doubling)} \end{cases}$$

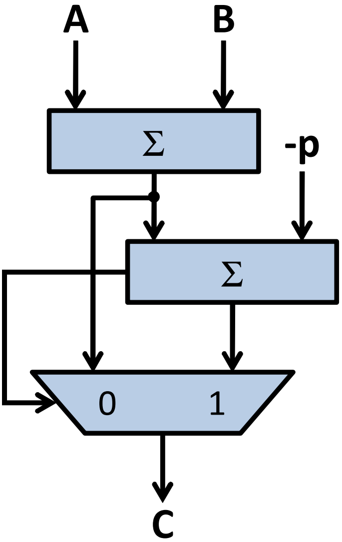
Necessary operations over $\mathbb{F}_{2^m}$

- addition
- multiplication
- inversion (division is multiplication by inverse element)
- squaring — it is worth of dedicated circuit

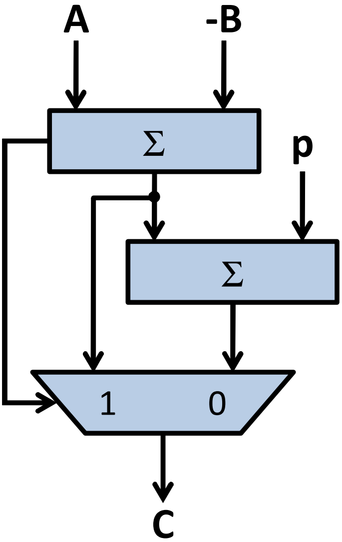
Outline

- Elliptic Curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, Necessary Operations
- Arithmetic Operations over $\mathbb{F}_p$
- Arithmetic Operations over $\mathbb{F}_p$ in Montgomery Domain

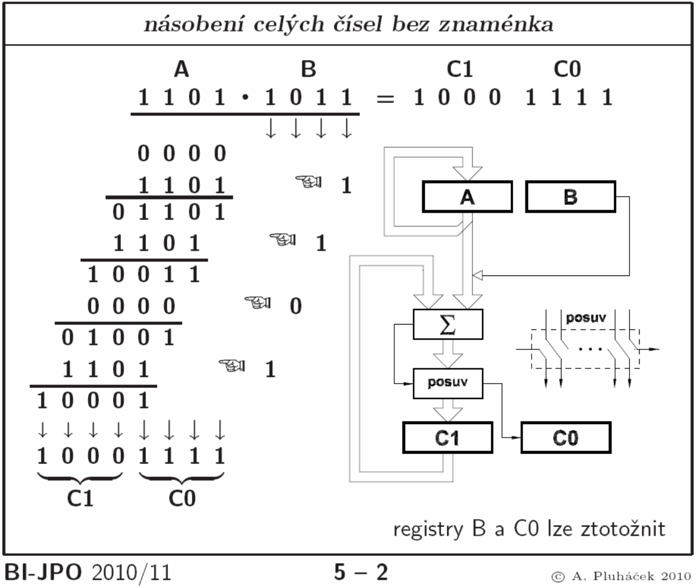
Addition over $\mathbb{F}_p$: $C = A + B \mod p$



Subtraction over $\mathbb{F}_p$: $C = A - B \mod p$



To remind you: „Classical“ LSB multiplier



Algorithm Double-and-Add — MSB multiplication

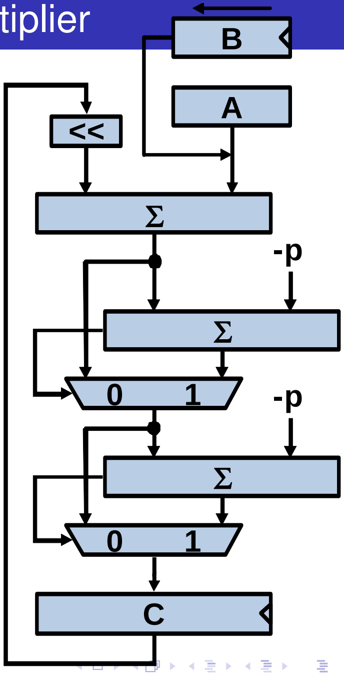
$$A \times 13 = A \times 1101_2 = (((\mathbf{1}A \times 2) + \mathbf{1}A) \times 2 + \mathbf{0}A) \times 2 + \mathbf{1}A$$

$$A \times 13 = A \times 1101_2 = \underbrace{\left(\underbrace{\left(\underbrace{\left(\underbrace{\mathbf{1}A}_{1 \times A} \times 2 \right) + \mathbf{1}A}_{3 \times A} \right) \times 2 + \mathbf{0}A}_{6 \times A} \right) \times 2 + \mathbf{1}A}_{13 \times A}$$

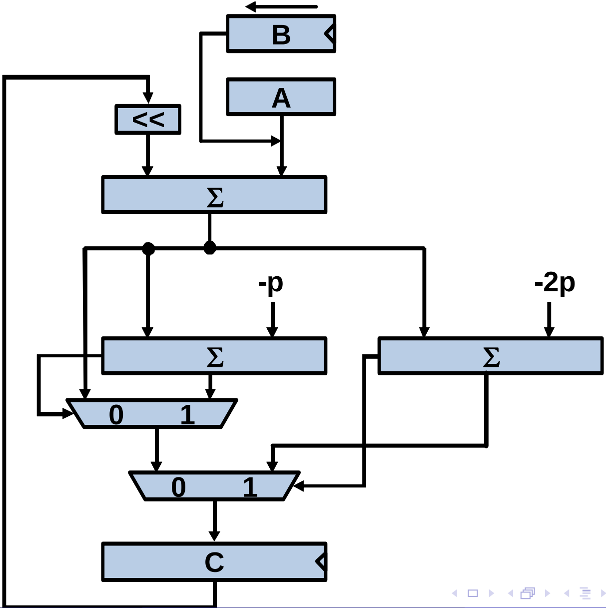
Multiplication over $\mathbb{F}_p$: MSB multiplier

Input: A, B, p, where $0 \leq A, B \leq p - 1$.
Output: $C = A \cdot B \mod p$
k: no. of bits B
b_i: i^{th} bit B

1. $C = 0$;
2. **for** $i = k - 1$ **downto** 0
3. $C = C \times 2 + b_i \cdot A$;
4. **if** $C \geq p$ **then**
5. $C = C - p$;
6. **end if**;
7. **if** $C \geq p$ **then**
8. $C = C - p$;
9. **end if**;
10. **end for**;



Multiplication over $\mathbb{F}_p$: MSB multiplier — modification



Inversion over $\mathbb{F}_p$

Extended Euclidean Algorithm (later)

Outline

- Elliptic Curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, Necessary Operations
- Arithmetic Operations over $\mathbb{F}_p$
- Arithmetic Operations over $\mathbb{F}_p$ in Montgomery Domain

Montgomery Domain

p-residuum

Let number $\bar{a} = |aR|_p$ be so-called p-residuum of number a.
Set of numbers $\bar{a}$ compose Montgomery Domain (Montgomery Field of Integers).
Radix R, $R > p$, is a least power of a base of positional number system.

Example: decimal system

Let module p be $p = 97$. Then radix $R = 100$.
p-residuum of $a = 21$ is $\bar{a} = |aR|_p = |21 \cdot 100|_{97} = 63$.
p-residuum of $b = 56$ is $\bar{b} = |bR|_p = |56 \cdot 100|_{97} = 71$.

Montgomery Domain — Addition/Subtraction I

Let c be a sum of two numbers a and b modulo p,

$$c = |a + b|_p.$$

Then its p-residuum $\bar{c}$ is

$$\bar{c} = |c \cdot R|_p = ||a + b|_p \cdot R|_p = |(a + b)R|_p.$$

p-residui of a and b are

$$\bar{a} = |a \cdot R|_p, \bar{b} = |b \cdot R|_p.$$

Sum of residui $\bar{a}$ and $\bar{b}$ is

$$\bar{a} + \bar{b} = |a \cdot R|_p + |b \cdot R|_p = |(a + b)R|_p = \bar{c}$$

$$\bar{a} + \bar{b} = \bar{c}.$$

Montgomery Domain — Addition/Subtraction II

Example: $a = 21$, $b = 56$, $p = 97$, $R = 100$, $\bar{a} = 63$, $\bar{b} = 71$

$$c = |a + b|_p = |21 + 56|_{97} = 77$$

$$\bar{c} = |77 \cdot 100|_{97} = 37$$

or

$$\bar{c} = \bar{a} + \bar{b} = |63 + 71|_{97} = |134|_{97} = 37$$

What about multiplication/division?

Montgomery Domain — Multiplication

Definition – Montgomery Product

Montgomery product of two p-residui $\bar{a}$ and $\bar{b}$ modulo p is defined as

$$\bar{c} = |\bar{a}\bar{b}R^{-1}|_p.$$

Proof ($\bar{c}$ is p-residuum of product $c = |a \cdot b|_p$):

$$\begin{aligned}
\bar{c} &= |cR|_p \\
&= |abR|_p \\
&= |aRbRR^{-1}|_p \\
&= |\bar{a}\bar{b}R^{-1}|_p
\end{aligned}$$

Montgomery Domain — Transformation

Transformation into Montgomery domain can be performed via Montgomery multiplication. We shall precompute the value of $|R^2|_p$.

Definition — Montgomery Tranformation by Multiplication

Let $a \in [0, p-1]$ be an integer and $b = |R^2|_p$. Then (according to definition of Montgomery product):

$$\bar{a} = |abR^{-1}|_p = |aR|_p.$$

Proof:

$$\bar{a} = |ab \cdot R^{-1}|_p = |aR^2R^{-1}|_p = |aR|_p.$$

Montgomery Domain — Inverse Transformation

Definition – Montgomery Inverse Transformation by Multiplication

Let $\bar{a} \in [0, p-1]$ be p-residuum, $\bar{b} = 1$. Then (according to definition of Montgomery product):

$$a = |\bar{a} \cdot 1 \cdot R^{-1}|_p.$$

Proof:

$$|\bar{a} \cdot 1 \cdot R^{-1}|_p = |aR \cdot 1 \cdot R^{-1}|_p = a.$$

Questions:

- What is the value of b?
- What is the value of p-residuum $\bar{x}$ of $x = 1$?

Montgomery Domain — Binary Multiplication

Let $R = 2^n$. Let $0 \leq \bar{b} < p$.

Let $\bar{a} = (\bar{a}_{n-1}\bar{a}_{n-2}\ldots\bar{a}_0)_2$, where $\bar{a}_i \in \{0,1\}$, s.t. $\bar{a} = \sum_{i=0}^{n-1} \bar{a}_i 2^i$.

Binary Montgomery Algorithm for Multiplication

Input: $\bar{a}, \bar{b}, p, n$
Output: $|\bar{a}\bar{b}R^{-1}|_p$

1. $x := 0, i := 0$
2. **while** $(i < n)$
3. $\quad x := x + \bar{a}_i \cdot \bar{b}$
4. $\quad x := (x + x_0 \cdot p)/2$
5. $\quad i := i + 1$
6. **if** $(x \geq p)$ **then**
7. $\quad x := x - p$
8. **return** $|\bar{a}\bar{b}R^{-1}|_p := x$

Montgomery Domain — Inversion I

Montgomery Modular Inverse

Input: $a, b \in \mathcal{Z}$, $a > b > 0$, a is n-bit odd number
Output: $\gcd(a, b) > 1$ or $|b^{-1}2^n|_a = |\overline{b}^{-1}|_a$

First stage

$u := a, v := b, r := 0, s := 1, k := 0$
while $(v > 0)$
  if (u even) **then**
    $u := u/2, s := 2s$
  else if (v even) **then**
    $v := v/2, r := 2r$
  else if $(u > v)$ **then**
    $u := (u - v)/2, r := r + s, s := 2s$
  else
    $v := (v - u)/2, s := r + s, r := 2r$
  $k := k + 1$
if $u \neq 1$ **then return** "Not relatively prime"

Montgomery Domain — Inversion II

if $r \geq a$ **then**
    $r := r - a$

Second stage

while$(k > n)$
    if r even **then**
        $r := r/2$
    else
        $r := (r + a)/2$
    $k := k - 1$
return $|b^{-1}2^n|_a := a - r$

Montgomery Domain - inverse I

Montgomeryho Modulární Inverse - vlastnosti

- Když $a > b > 0$, $\gcd(a, b) = 1$, a je liché a n je počet bitů a, potom
  - počet iterací v 1. fázi algoritmu MI je minimálně n a maximálně $2n$,
  - MI vrací $|b^{-1}2^n|_a$,
- Nechť $AMI(a, b) = |b^{-1}2^k|_a$ reprezentuje 1. fázi MI algoritmu, a funkce $MMI(a, b) = |b^{-1}2^n|_a = |\overline{b}^{-1}|_a$ reprezentuje celý algoritmus MI, kde n je počet bitů v a a $\overline{b} = |b2^n|_a$ je p-residuum. Potom je zřejmé, že MMI může být použitá k výpočtu MD inverze nějakého celého čísla.
- Pro výpočet celočíselné inverze z nějakého čísla z MD platí:

$$|MMI(p, \overline{b})|_p = |\overline{b}^{-1}2^n|_p = |(b2^n)^{-1}2^n|_p = |b^{-1}2^{-n}2^n|_p = |b^{-1}|_p$$

- Pro výpočet inverze v MD (MMI) z operandu z MD se musí 2. fáze algoritmu upravit. Namísto dělení výsledku 1. fáze (AMI) číslem 2^{k-n}, se musí výsledek násobit číslem 2^{2n-k}.

Montgomery Domain - inverse II

- To je provedené *násobením 2/posuvem vlevo* $(2n - k)$ - krát a redukcí modulo p.

$$|AMI(p, \overline{b}) \cdot 2^{2n-k}|_p = |(b^{-1}2^{-n}2^k)2^{2n-k}|_p = |\overline{b}^{-1}|_p$$

- Nasledující vztahy shrnují výpočet inverzí pro všechny možnosti:

$$\begin{aligned} |\overline{b}^{-1}|_p &= |MMI(p, b)|_p \\ |b^{-1}|_p &= |MMI(p, \overline{b})|_p \\ |\overline{b}^{-1}|_p &= |AMI(p, \overline{b}) \cdot 2^{2n-k}|_p. \end{aligned} \tag{1}$$

EEA pro výpočet INV

Euklidův algoritmus (EA)

Vstup: $x, y \in \mathcal{Z}$ a $x > y > 0$
Výstup: $\gcd(x, y)$

1. **while**$(y > 0)$
2. $\quad r \leftarrow x \bmod y$
3. $\quad x \leftarrow y$
4. $\quad y \leftarrow r$
5. **return** x

EA lze zapsat také pomocí rekurentního vztahu:
$r_i = r_{i-2} - q_i r_{i-1}$,
$q_i = \lfloor r_{i-2}/r_{i-1} \rfloor$,
kde $r_0 = x$ a $r_1 = y$ a
pro $r_{n-1} = 0$ je $\gcd(x, y) = r_n$.

V případě výpočtu INV je $r_0 = p$ a $r_1 = a$, kde p je prvočíslo, $p > a > 0$ a $\gcd(p, a) = 1$.

Při výpočtu INV přechází EA na EEA $\Longrightarrow r_i = f_i p + g_i a$ a pro $r_n = 1$ je $g_n = a^{-1} \bmod p$.

Rozšířený Euklidův algoritmus – EEA pro výpočet MI

$a \in \mathbb{Z}_p$, kde p je prvočíslo a platí, že $\gcd(p, a) = 1$ a $p > a > 0$

Určující podmínky

$$\begin{array}{rcl|lll|}
r_0 & = & p & & & \\
r_1 & = & a & & & \\
r_2 & = & r_0 - r_1 q_2 & 0 < r_2 < r_1 & q_2 & = \mathrm{f}(r_0, r_1) \\
 & \vdots & & & & \\
r_i & = & r_{i-2} - r_{i-1} q_i & 0 < r_i < r_{i-1} & q_i & = \mathrm{f}(r_{i-2}, r_{i-1}) \\
 & \vdots & & & & \\
r_n & = & r_{n-2} - r_{n-1} q_n & r_n = 1 & q_n & = \mathrm{f}(r_{n-2}, r_{n-1}) \\
0 & = & r_{n-1} - r_n q_{n+1}, & & q_{n+1} & = r_{n-1}
\end{array}$$

Diofantické rovnice

$r_i = r_{i-2} - q r_{i-1} \Leftrightarrow r_i = f_i p + g_i a$

potom lze stanovit rekurentní vztahy pro výpočet MI

EEA pro výpočet modulární inverze III

vstupy
určující podmínky
rekurentní vztahy
výstup:

$$
\begin{aligned}
r_0 &= p \\
r_1 &= a \\
q_i &= \lfloor r_{i-2}/r_{i-1} \rfloor \\
r_i &= r_{i-2} - q_i r_{i-1} \\
0 &< r_i < r_{i-1} \\
f_i &= f_{i-2} - q_i f_{i-1} \\
g_i &= g_{i-2} - q_i g_{i-1} \\
g_n &= a^{-1} \bmod p
\end{aligned}
$$

EEA algoritmus

vstup: p je prvočíslo $p > a > 0$
výstup: a^{-1}mod

1. $r_1 := p, r_2 := a, g_1 := 0, g_2 := 1$
2. **while** $(r_2 > 0)$
3. $\quad q := \lfloor r_1/r_2 \rfloor$
4. $\quad g := g_2,\ r := r_2$
5. $\quad r_2 := r_1 - qr_2$
   $\quad f_2 := f_1 - qf_2$
   $\quad g_2 := g_1 - qg_2$
6. $\quad g_1 := g,\ r_1 := r$
7. **return** $a^{-1} \bmod p = g$

Penkova inverze - algoritmus I

Input: $a \in [1, p-1]$ and p
Output: $r \in [1, p-1]$ and k, where $r = a^{-1} \bmod p$
and $n \leq k \leq 2n$

$u := p, v := a, r := 0, s := 1$
$k := 0$
while $(v > 0)$
    if (u is even) then
        if (r is even) then
            $u := u/2$, $r := r/2$, $k := k+1$
        else
            $u := u/2$, $r := (r+p)/2$, $k := k+1$
    else if (v is even) then
        if (s is even) then
            $v := v/2$, $s := s/2$, $k := k+1$
        else
            $v := v/2$, $s := (s+p)/2$, $k := k+1$
    else $x := (u - v)$

Penkova inverze - algoritmus II

```
        if (x > 0) then
            u := x, r := r − s
            if (r < 0) then
                r := r + p
        else
            v := −x, s := s − r
            if (s < 0) then
                s := s + p
if (r > p) then
    r := r − p
if (r < 0) then
    r := r + p
return r and k.
```

Left-shift inverze - algoritmus I

Input: $a \in [1, p-1]$ and p
Output: $r \in [1, p-1]$, where $r = a^{-1} \pmod p$, c_u, c_v
and $0 < c\_v + c\_u \leq 2n$

1. $u := p, v := a, r := 0, s := 1$
2. $c\_u = 0$, $c\_v = 0$
3. while($u \neq \pm 2^{c\_u}$ & $v \neq \pm 2^{c\_v}$)
4. if $(u_n, u_{n-1} = 0)$ *or* $(u_n, u_{n-1} = 1$ & $\mathrm{OR}(u_{n-2}, \ldots, u_0) = 1)$ then
5. if $(c\_u \geq c\_v)$ then
6. $u := 2u$, $r := 2r$, $c\_u := c\_u + 1$
7. else
8. $u := 2u$, $s := s/2$, $c\_u := c\_u + 1$
9. else if $(v_n, v_{n-1} = 0)$ *or* $(v_n, v_{n-1} = 1$ & $\mathrm{OR}(v_{n-2}, \ldots, v_0) = 1)$ then
10. if $(c\_v \geq c\_u)$ then
11. $v := 2v$, $s := 2s$, $c\_v := c\_v + 1$
12. else
13. $v := 2v$, $r := r/2$, $c\_v := c\_v + 1$
14. else

```
        if (v_n = u_n) then
            oper = " − "
        else
            oper = " + "
        if (c_u ≤ c_v) then
            u := u oper v, r := r oper s
        else
            v := v oper u, s := s oper r
if (v = ±2^{c_v}) then
    r := s, u_n := v_n
if (u_n = 1) then
    if (r < 0) then
        r := −r
    else
        r := p − r
if (r < 0) then
    r := r + p
return r, c_u, and c_v.
```

Left-shift inverze - algoritmus III

Příklad

I	operations	values of registers	tests
0		$u^{(0)} = (13)_{10} = (01010.)_2$ $v^{(0)} = (10)_{10} = (01010.)_2$ $r^{(0)} = (0)_{10} = (00000.)_2$ $s^{(0)} = (1)_{10} = (00001.)_2$	$u^{(0)} \neq \pm 2^0$ $v^{(0)} \neq \pm 2^0$
1	$u^{(1)} = u^{(0)} - v^{(0)}$ $r^{(1)} = r^{(0)} - s^{(0)}$	$u^{(1)} = (3)_{10} = (00011.)_2$ $v^{(1)} = (10)_{10} = (01010.)_2$ $r^{(1)} = (-1)_{10} = (11111.)_2$ $s^{(1)} = (1)_{10} = (00001.)_2$	$u^{(1)} \neq \pm 2^0$ $v^{(1)} \neq \pm 2^0$
2	$u^{(2)} = 4u^{(1)}$ $r^{(2)} = 4r^{(1)}$	$u^{(2)} = (12)_{10} = (011.00)_2$ $v^{(2)} = (10)_{10} = (01010.)_2$ $r^{(2)} = (-4)_{10} = (111.00)_2$ $s^{(2)} = (1)_{10} = (00001.)_2$	$u^{(2)} \neq \pm 2^2$ $v^{(2)} \neq \pm 2^0$
3	$v^{(3)} = v^{(2)} - u^{(2)}$ $s^{(3)} = s^{(2)} - r^{(2)}$	$u^{(3)} = (12)_{10} = (011.00)_2$ $v^{(3)} = (-2)_{10} = (11110.)_2$ $r^{(3)} = (-4)_{10} = (111.00)_2$ $s^{(3)} = (5)_{10} = (00101.)_2$	$u^{(3)} \neq \pm 2^2$ $v^{(3)} \neq \pm 2^0$
4	$v^{(4)} = 4v^{(3)}$ $r^{(4)} = r^{(3)}/4$	$u^{(4)} = (12)_{10} = (011.00)_2$ $v^{(4)} = (-8)_{10} = (110.00)_2$ $r^{(4)} = (-1)_{10} = (11111.)_2$ $s^{(4)} = (5)_{10} = (00101.)_2$	$u^{(4)} \neq \pm 2^2$ $v^{(4)} \neq \pm 2^2$
5	$u^{(5)} = u^{(4)} + v^{(4)}$ $r^{(5)} = r^{(4)} + s^{(4)}$	$u^{(5)} = (4)_{10} = (001.00)_2$ $r^{(5)} = (4)_{10} = (00100.)_2$	$u^{(5)} = 2^2$

HW implementace Left-shift algoritmu