Faster Pairing Hardware Accelerators

Junfeng Fan*

Joint work with Frederik Vercautere*, Gavin Yao†, Ray Cheung†, and Ingrid Verbauwhede*.

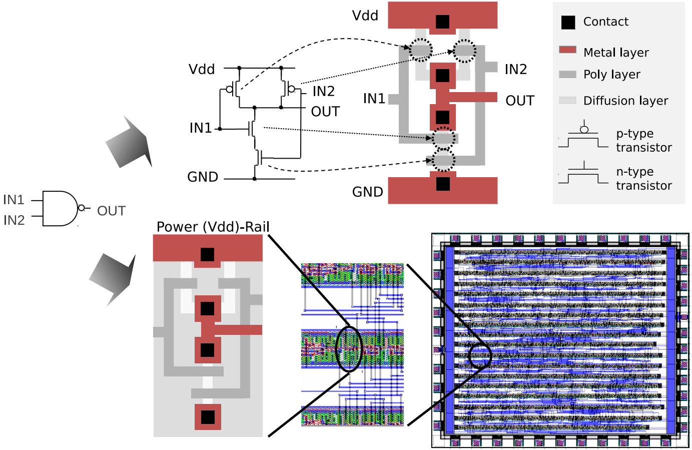
* ESAT/SCD-COSIC
KU Leuven, Belgium

† Dept. of Electronic Engineering
City University of Hong Kong

ECC 2012, Querétaro

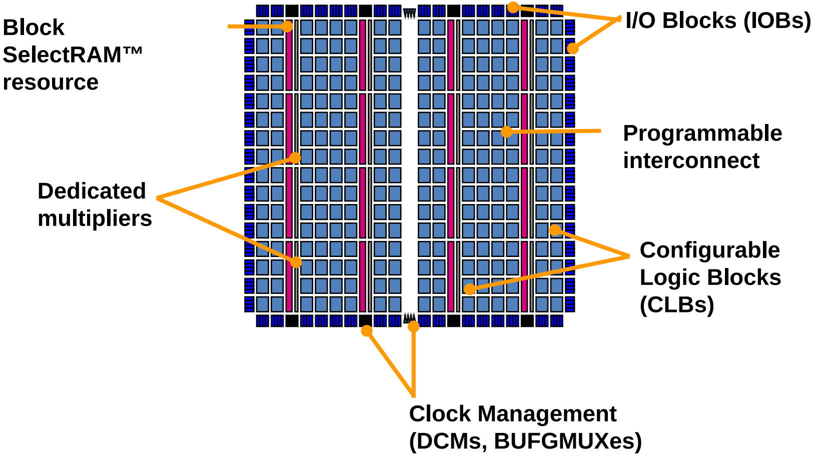
Agenda

Platform - ASIC



"source: Andrew B. Kahng et al."

Platform - FPGA

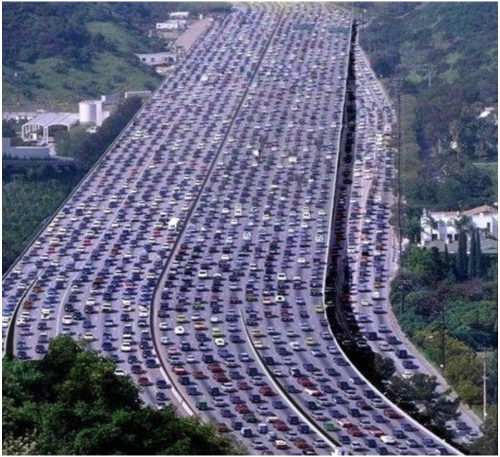


- So, how do we build fast hardware?

In the beginning...

We increased the frequency...

Beautiful parallelism



Datapath reuse

Unbalanced occupancy

Dynamic reconfiguration

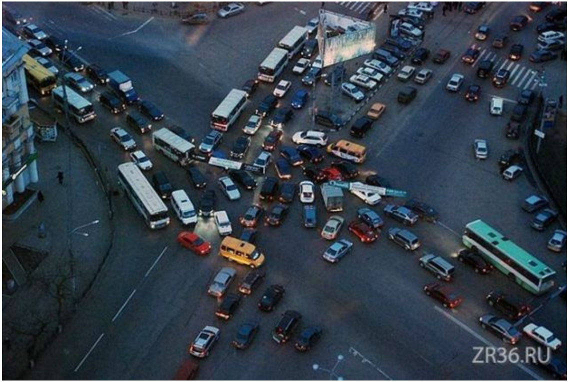
Complexity reduction

Carpooling

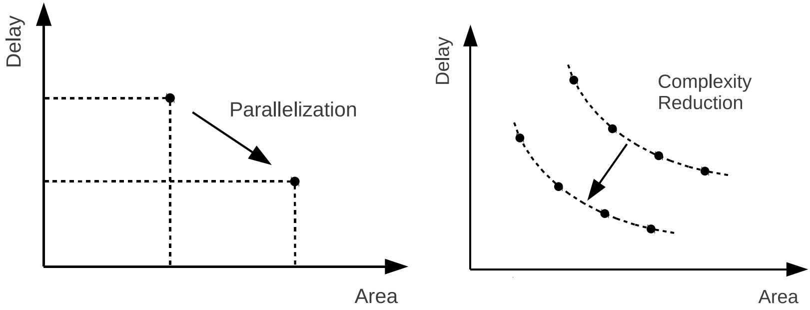
Ride More,
Drive Less.

One-Child policy

Scheduling is critical



The design space



Complexity vs Implementations

- Computational: how many bit-operations.
- Software implementation
  - Typical measurement: no. of cycles, code size
  - Depends on: platform, compiler, programmer, etc.
- Hardware implementation
  - Typical measurement: area, throughput, power
  - Depends on:
    - platform: ASIC vs FPGA
    - Architecture: Low-area vs High-speed
    - EDA tools: synthesis, P&A, etc.
    - Designers.

Algorithm-Architecture co-optimization

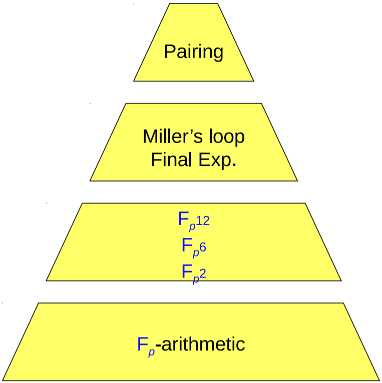
- Optimize your algorithm and architecture together
  - Step 1: analyse and optimize the algorithm
  - Step 2: Map the algorithm to hardware
  - Step 3: Optimize the architecture
    - data-path reuse
    - reduce pipeline bubbles
    - optimize the memory structure
    - achieve higher frequency
  - Step 4: Optimize the algorithm based on Step 3
  - Step 3: Go to Step 3

Pairing computation

Algorithm 3. Computing the Tate pairing for $E_3/\mathbb{F}_p$

INPUT: $P \in G_1$ and $Q \in G_2$.
OUTPUT: $t_r(P, Q)$.

1. Write r in binary: $r = \sum_{i=0}^{L-1} r_i 2^i$.
2. $T \leftarrow P$, $f \leftarrow 1$.
3. For i from $L-2$ downto 0 do: {Miller operation}
   - 3.1 Let ℓ be the tangent line at T.
   - 3.2 $T \leftarrow 2T$.
   - 3.3 $f \leftarrow f^2 \cdot \ell(Q)$.
   - 3.4 If $r_i = 1$ and $i \neq 0$ then
     Let ℓ be the line through T and P.
     $T \leftarrow T + P$.
     $f \leftarrow f \cdot \ell(Q)$.
4. Compute $f^{(p^{12}-1)/r}$ as follows: {Final exponentiation}
   - 4.1 $f \leftarrow f^{p^6-1}$.
   - 4.2 $f \leftarrow f^{p^2+1}$.
   - 4.3 $a \leftarrow f^{-(6z+5)}$, $b \leftarrow a^p$, $b \leftarrow a \cdot b$.
   - 4.4 Compute f^p, f^{p^2}, f^{p^3}.
   - 4.5 $f \leftarrow f^{p^3} \cdot [b \cdot (f^p)^2 \cdot f^{p^2}]^{6z^2+1} \cdot b \cdot (f^p \cdot f)^9 \cdot a \cdot f^4$.
5. Return(f).



Faster Pairing Computation?

- Speed up $\mathbb{F}_p$ multiplications

Agenda

1. Introduction
2. Hybrid Montgomery
3. RNS Montgomery

Modular multiplication

- Target: compute $ab \bmod p$
- Fast reduction method
  - Use pseudo-Mersenne number

$$p = 2^m - s$$

  - Montgomery reduction
  - Barrett reduction
  - Chung-Hasan
    - if $p = f(t)$, where $f(t)$ is monic.

Montgomery Mult.

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1: $T \leftarrow AB$
2: $\mu \leftarrow T \bmod R$
3: $q \leftarrow \mu \cdot (p') \bmod R$
4: $S \leftarrow (T + qp)/R$
5: $S \leftarrow S - p$ if $S > p$

Return: S

Montgomery Mult.

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1: $T \leftarrow AB$
2: $\mu \leftarrow T \bmod R$
3: $q \leftarrow \mu \cdot (p') \bmod R$
4: $S \leftarrow (T + qp)/R$
5: $S \leftarrow S - p$ if $S > p$

Return: S

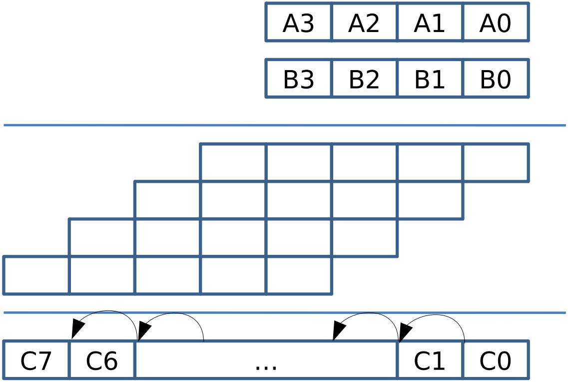
Complexity

Let $\log_2(|p|)+1=nw$

n^2 **MUL**$_w$

n^2 **MUL**$_w$
n^2 **MUL**$_w$

Long integer multiplication: carry propagation



Pairing on BN curves

- Barreto-Naehrig Curves:

$$y^2 = x^3 + b \;\; over \;\; \mathbb{F}_p,$$

where

$$\begin{aligned} p(z) &= 36z^4 + 36z^3 + 24z^2 + 6z + 1, \\ r(z) &= 36z^4 + 36z^3 + 18z^2 + 6z + 1, \\ t(z) &= 6z^2 + 1, \\ k &= 12 \end{aligned}$$

Some observations

- $p = 36z^4 + 36z^3 + 24z^2 + 6z + 1$
- p can not be psudo-Mersenne number
- However,
  - $p(z)$ has small coefficients
  - $p^{-1}(z) = 324z^4 - 36z^3 - 12z^2 + 6z - 1 \bmod z^5$
  - $p^{-1}(z) = -1 \bmod z$

Polynomial based reduction

Montgomery Mult.

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1: $T \leftarrow AB$
2: $\mu \leftarrow T \bmod R$
3: $q \leftarrow \mu \cdot (p') \bmod R$
4: $S \leftarrow (T + qp)/R$
5: $S \leftarrow S - p$ if $S > p$

Return: S

Polynomial based reduction

Montgomery Mult.

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1: $T \leftarrow AB$
2: $\mu \leftarrow T \bmod R$
3: $q \leftarrow \mu \cdot (p') \bmod R$
4: $S \leftarrow (T + qp)/R$
5: $S \leftarrow S - p$ if $S > p$

Return: S

Montgomery Mult. using poly.

Input: $A(z)$ and $B(z)$

Output: $T = A(z)B(z)R^{-1}(z) \bmod p(z)$

1: $T(z) \leftarrow A(z)B(z)$
2: $\mu(z) \leftarrow T(z) \bmod R(z)$
3: $q(z) \leftarrow \mu(z) \cdot (p'(z)) \bmod R(z)$
4: $S(z) \leftarrow (T(z) + q(z)p(z))/R(z)$

Return: $S(z)$

- Note: $R = z^5$, $p'(z) = 324z^4 - 36z^3 - 12z^2 + 6z - 1$, $p(z) = (36z^4 + 36z^3 + 24z^2 + 6z + 1)$.

Coefficient reduction

- There is one problem: coefficient grows
  - Input: $a(z) = 35z^4 + 36z^3 + 7z^2 + 6z + 103$,
    $b(z) = 5z^4 + 136z^3 + 34z^2 + 9z + 5$
    Select $z = 137$,
  - Compute
    - step 1: $c(z) \leftarrow a(z)b(z)$
    - step 2: $\mu(z) \leftarrow c(z) \bmod z^5$
    - step 3: $q(z) \leftarrow \mu(z)p'(z) \bmod z^5$
    - step 4: $r(z) \leftarrow (c(z) + q(z)p(z)/z^5$
  - Result:
    $r(z) = 2243z^4 - 820648z^3 - 964511z^2 - 616127z - 173978$
    Thus, we need to reduce the coefficient s.t. $r_i < z$
    $r(z) = -28z^5 + 37z^4 + 32z^3 + 120z^2 + 62z + 12$.

Selection of z

- We need division by z
- Choose $z = 2^m + s$, where s is small.
- For BN-curves, we also need
  - prime $p(z) = 36z^4 + 36z^3 + 24z^2 + 6z + 1$,
  - prime $r(z) = 36z^4 + 36z^3 + 18z^2 + 6z + 1$
- Example: $z = 2^{63} + 857$ to achieve 128-bit security.

Complexity analysis

Montgomery Mult. using poly.

Input: $A(z)$ and $B(z)$

Output: $T = A(z)B(z)R^{-1} \bmod p(z)$

1: $T(z) \leftarrow A(z)B(z)$
2: Coefficient reduction.
3: $\mu(z) \leftarrow T(z) \bmod R$
4: $q(z) \leftarrow \mu(z) \cdot (p'(z)) \bmod R$
5: $S(z) \leftarrow (T(z) + q(z)p(z))/R$

Return: $S(z)$

Complexity

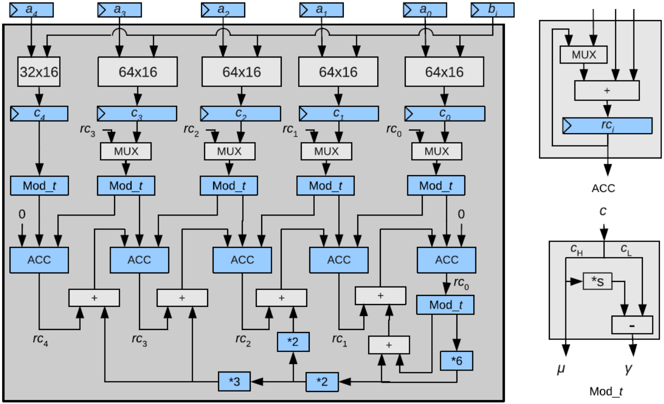
Let n=deg($p(z)$)+1, w=log$|z|$+1

n^2 **MUL**$_w$
$2n - 1$ **CoRedc()**

Easy
Easy

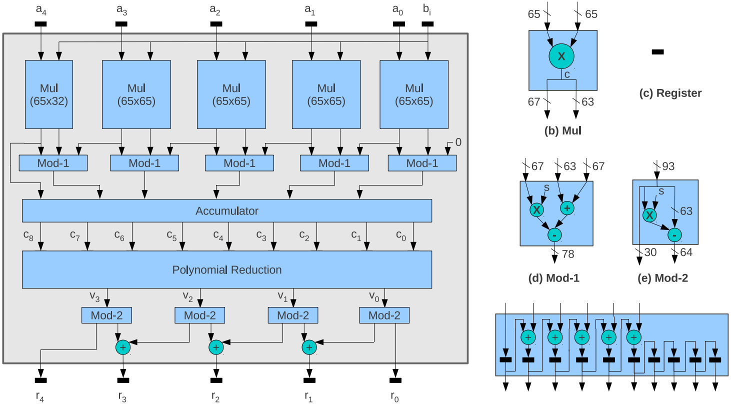
Digit-serial Hybrid Multiplier

- Digit-serial implementations ($n = 4$, $w = 64$)



Another Digit-serial Hybrid Multiplier

- Digit-serial implementations ($n = 4$, $w = 64$)



(a) HMM multiplier

(b) Mul

(c) Register

(d) Mod-1

(e) Mod-2

(f) Accumulator

Discussion

- Advantages
  - Reduced complexity
  - Easy to parallelize
- Disadvantages
  - Only work for specific polynomial form primes
  - Doesn't make use of lazy reduction

Agenda

RNS representation

RNS is defined by n pairwise coprime integer constants:

$$\mathfrak{B} = \{b_1, b_2, \cdots, b_n\}.$$

$$M_{\mathfrak{B}} := \prod_{i=1}^{n} b_i, b_i \in \mathfrak{B}$$

RNS representation

RNS is defined by n pairwise coprime integer constants:

$$\mathfrak{B} = \{b_1, b_2, \cdots, b_n\}.$$

$$M_{\mathfrak{B}} := \prod_{i=1}^{n} b_i, b_i \in \mathfrak{B}$$

Any integer $X, 0 \leqslant X < M_{\mathfrak{B}}$, X is uniquely represented by:

$$\mathfrak{X} = \{|X|_{b_1}, |X|_{b_2}, \cdots, |X|_{b_n}\},$$

RNS representation

RNS is defined by n pairwise coprime integer constants:

$$\mathfrak{B} = \{b_1, b_2, \cdots, b_n\}.$$

$$M_{\mathfrak{B}} := \prod_{i=1}^{n} b_i, b_i \in \mathfrak{B}$$

Any integer $X, 0 \leqslant X < M_{\mathfrak{B}}$, X is uniquely represented by:

$$\mathfrak{X} = \{|X|_{b_1}, |X|_{b_2}, \cdots, |X|_{b_n}\},$$

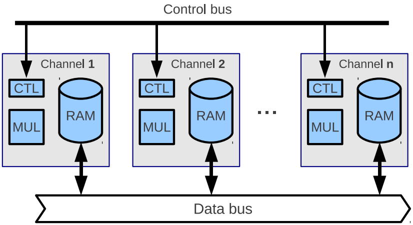
Example:

- X= 42711290161816493313599693852409489022
- $\mathfrak{X}$= $\{103164, 4142, 38734, 34062, 26238, 30586, 117182, 113538\}$
  base: $\{2^{17} - 9, 2^{17} - 7, 2^{17} - 3, 2^{17} - 1, 2^{17}, 2^{17} + 1, 2^{17} + 5, 2^{17} + 9\}$

Arithmetic operations using RNS

Arithmetic operations using RNS ($\mathbb{Z}/M_{\mathfrak{B}}\mathbb{Z}$)

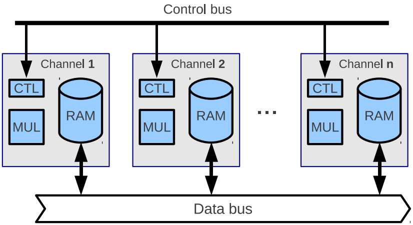
Normal	RNS	
$R = X \pm Y \bmod M_{\mathfrak{B}}$	$\mathfrak{R} = \mathfrak{X} \pm \mathfrak{Y}$,	where $r_i = x_i \pm y_i \mod b_i$
$R = XY \bmod M_{\mathfrak{B}}$	$\mathfrak{R} = \mathfrak{X} \odot \mathfrak{Y}$,	where $r_i = x_i y_i \mod b_i$
$R = X/Y \bmod M_{\mathfrak{B}}$ if $\gcd(Y, M_{\mathfrak{B}}) = 1$	$\mathfrak{R} = \mathfrak{X} \odot \mathfrak{Y}^{-1}$,	where $r_i = x_i y_i^{-1} \mod b_i$



Arithmetic operations using RNS

Arithmetic operations using RNS ($\mathbb{Z}/M_{\mathfrak{B}}\mathbb{Z}$)

Normal	RNS	
$R = X \pm Y \bmod M_{\mathfrak{B}}$	$\mathfrak{R} = \mathfrak{X} \pm \mathfrak{Y}$,	where $r_i = x_i \pm y_i \mod b_i$
$R = XY \bmod M_{\mathfrak{B}}$	$\mathfrak{R} = \mathfrak{X} \odot \mathfrak{Y}$,	where $r_i = x_i y_i \mod b_i$
$R = X/Y \bmod M_{\mathfrak{B}}$	$\mathfrak{R} = \mathfrak{X} \odot \mathfrak{Y}^{-1}$,	where $r_i = x_i y_i^{-1} \mod b_i$
if $\gcd(Y, M_{\mathfrak{B}}) = 1$		

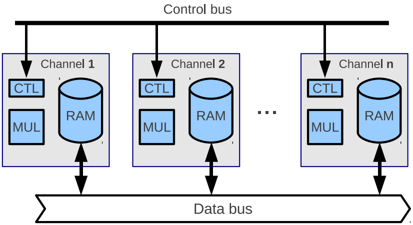


Arithmetic operations using RNS

Arithmetic operations using RNS ($\mathbb{Z}/M_{\mathfrak{B}}\mathbb{Z}$)

Normal	RNS	
$R = X \pm Y \bmod M_{\mathfrak{B}}$	$\mathfrak{R} = \mathfrak{X} \pm \mathfrak{Y}$,	where $r_i = x_i \pm y_i \mod b_i$
$R = XY \bmod M_{\mathfrak{B}}$	$\mathfrak{R} = \mathfrak{X} \odot \mathfrak{Y}$,	where $r_i = x_i y_i \mod b_i$
$R = X/Y \bmod M_{\mathfrak{B}}$ if $\gcd(Y, M_{\mathfrak{B}}) = 1$	$\mathfrak{R} = \mathfrak{X} \odot \mathfrak{Y}^{-1}$,	where $r_i = x_i y_i^{-1} \mod b_i$

$b_i = 2^w \pm t$, and t is a small integer.



Complexity

Complexity of Montgomery Mult.

$2n^2 + n$ $\mathbf{MUL}_w$

Complexity

Complexity of Montgomery Mult.

$2n^2 + n$ **MUL**$_w$

Complexity of Hybrid Montgomery Mult.

n^2 **MUL**$_w$ +
PolyRedc() +
$2n - 1$ **CoRedc()**

Complexity

Complexity of Montgomery Mult.

$2n^2 + n$ **MUL**$_w$

Complexity of Hybrid Montgomery Mult.

n^2 **MUL**$_w$ +
PolyRedc() +
$2n - 1$ **CoRedc()**

RNS Mult.

n **MUL**$_w$ +
n **CoRedc()**

RNS Montgomery algorithm [Kawamura *et al.* 00]

Montgomery – R

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1:$T \leftarrow AB$
2:$Q \leftarrow \left| |T|_R \cdot (-p)^{-1} \right|_R$
3:$S \leftarrow (T + Qp)/R$

RNS Montgomery algorithm [Kawamura *et al.* 00]

Montgomery – R

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1:$T \leftarrow AB$
2:$Q \leftarrow \left| |T|_R \cdot (-p)^{-1} \right|_R$

3:$S \leftarrow (T + Qp)/R$

RNS Montgomery – $M_{\mathfrak{B}}$

Input: $A = aM_{\mathfrak{B}} \bmod p$ and
$B = bM_{\mathfrak{B}} \bmod p$

Output: $T = abM_{\mathfrak{B}} \bmod p$

in $\mathfrak{B}$
1: $T_{\mathfrak{B}} \leftarrow A_{\mathfrak{B}} B_{\mathfrak{B}}$
2: $Q_{\mathfrak{B}} \leftarrow T_{\mathfrak{B}} \cdot (-p)^{-1}$

3: $S_{\mathfrak{B}} \leftarrow (T + Q_{\mathfrak{B}} p)/M_{\mathfrak{B}}$

RNS Montgomery algorithm [Kawamura *et al.* 00]

Montgomery – R

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1:$T \leftarrow AB$
2:$Q \leftarrow \left| |T|_R \cdot (-p)^{-1} \right|_R$

3:$S \leftarrow (T + Qp)/R$

RNS Montgomery – $M_{\mathfrak{B}}$

Input: $A = aM_{\mathfrak{B}} \bmod p$ and
$B = bM_{\mathfrak{B}} \bmod p$

Output: $T = abM_{\mathfrak{B}} \bmod p$

in $\mathfrak{B}$
1: $T_{\mathfrak{B}} \leftarrow A_{\mathfrak{B}} B_{\mathfrak{B}}$
2: $Q_{\mathfrak{B}} \leftarrow T_{\mathfrak{B}} \cdot (-p)^{-1}$

3: $S_{\mathfrak{B}} \leftarrow (T + Q_{\mathfrak{B}} p)/M_{\mathfrak{B}}$

$M_{\mathfrak{B}}^{-1}$ does not exist in $\mathfrak{B}$.

Introduce a new base $\mathfrak{C}$ to perform division by $M_{\mathfrak{B}}$.

RNS Montgomery algorithm [Kawamura *et al.* 00]

Montgomery – R

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1: $T \leftarrow AB$
2: $Q \leftarrow \left|\,|T|_R \cdot (-p)^{-1}\right|_R$

3: $S \leftarrow (T + Qp)/R$

RNS Montgomery – $M_{\mathfrak{B}}$

Input: $A = aM_{\mathfrak{B}} \bmod p$ and
$B = bM_{\mathfrak{B}} \bmod p$

Output: $T = abM_{\mathfrak{B}} \bmod p$

	in $\mathfrak{B}$	in $\mathfrak{C}$
1:	$T_{\mathfrak{B}} \leftarrow A_{\mathfrak{B}} B_{\mathfrak{B}}$	$T_{\mathfrak{C}} \leftarrow A_{\mathfrak{C}} B_{\mathfrak{C}}$
2:	$Q_{\mathfrak{B}} \leftarrow T_{\mathfrak{B}} \cdot (-p)^{-1}$	
3:		$S_{\mathfrak{C}} \leftarrow (T_{\mathfrak{C}} + Q_{\mathfrak{C}} p)(M_{\mathfrak{B}}^{-1})_{\mathfrak{C}}$

$M_{\mathfrak{B}}^{-1}$ does not exist in $\mathfrak{B}$.

Introduce a new base $\mathfrak{C}$ to perform division by $M_{\mathfrak{B}}$.

The overhead is two base extensions.

RNS Montgomery algorithm [Kawamura *et al.* 00]

Montgomery – R

Input: $A = aR \bmod p$ and
$B = bR \bmod p$

Output: $T = abR \bmod p$

1: $T \leftarrow AB$
2: $Q \leftarrow \left| |T|_R \cdot (-p)^{-1} \right|_R$
3: $S \leftarrow (T + Qp)/R$

RNS Montgomery – $M_{\mathfrak{B}}$

Input: $A = aM_{\mathfrak{B}} \bmod p$ and
$B = bM_{\mathfrak{B}} \bmod p$

Output: $T = abM_{\mathfrak{B}} \bmod p$

	in $\mathfrak{B}$		in $\mathfrak{C}$
1:	$T_{\mathfrak{B}} \leftarrow A_{\mathfrak{B}} B_{\mathfrak{B}}$		$T_{\mathfrak{C}} \leftarrow A_{\mathfrak{C}} B_{\mathfrak{C}}$
2:	$Q_{\mathfrak{B}} \leftarrow T_{\mathfrak{B}} \cdot (-p)^{-1}$		
3:		$Q_{\mathfrak{B}} \longrightarrow Q_{\mathfrak{C}}$	
4:			$S_{\mathfrak{C}} \leftarrow (T_{\mathfrak{C}} + Q_{\mathfrak{C}} p)(M_{\mathfrak{B}}^{-1})_{\mathfrak{C}}$
5:		$S_{\mathfrak{B}} \longleftarrow S_{\mathfrak{C}}$	

$M_{\mathfrak{B}}^{-1}$ does not exist in $\mathfrak{B}$.

Introduce a new base $\mathfrak{C}$ to perform division by $M_{\mathfrak{B}}$.

The overhead is two base extensions.

Base extension

$$\begin{matrix} \mathfrak{X}_{\mathfrak{B}} = \{x_1, x_2, \cdots, x_n\} \\ x_i = X \bmod b_i \end{matrix} \xrightarrow{\textit{Base Extension}} \begin{matrix} \mathfrak{X}_{\mathfrak{C}} = \{x'_1, x'_2, \cdots, x'_n\} \\ x'_i = X \bmod c_i \end{matrix}$$

Base extension

$$\mathfrak{X}_{\mathfrak{B}} = \{x_1, x_2, \cdots, x_n\} \qquad \mathfrak{X}_{\mathfrak{C}} = \{x'_1, x'_2, \cdots, x'_n\}$$

$$x_i = X \bmod b_i \quad \xrightarrow{\textit{Base Extension}} \quad x'_i = X \bmod c_i$$

Chinese Remainder Theorem (CRT):

$$X = \left| \sum_{i=1}^{n} B_i \cdot \xi_i \right|_{M_{\mathfrak{B}}}$$

where

$$\xi_i = \left| x_i \cdot |B_i^{-1}|_{b_i} \right|_{b_i}, B_i = \prod_{k=1, k \neq i}^{n} b_k$$

Base extension

$$\mathfrak{X}_{\mathfrak{B}} = \{x_1, x_2, \cdots, x_n\} \quad\quad \mathfrak{X}_{\mathfrak{C}} = \{x'_1, x'_2, \cdots, x'_n\}$$

$$x_i = X \bmod b_i \quad \xrightarrow{\textit{Base Extension}} \quad x'_i = X \bmod c_i$$

Chinese Remainder Theorem (CRT):

$$X = \left| \sum_{i=1}^{n} B_i \cdot \xi_i \right|_{M_{\mathfrak{B}}} = \sum_{i=1}^{n} B_i \cdot \xi_i - \lambda \cdot M_{\mathfrak{B}}$$

where

$$\xi_i = \left| x_i \cdot |B_i^{-1}|_{b_i} \right|_{b_i}, B_i = \prod_{k=1, k \neq i}^{n} b_k$$

Base extension

$$\begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix} = \begin{pmatrix} |B_1|_{c_1} & |B_2|_{c_1} & \cdots & |B_n|_{c_1} \\ |B_1|_{c_2} & |B_2|_{c_2} & \cdots & |B_n|_{c_2} \\ \vdots & \vdots & \ddots & \vdots \\ |B_1|_{c_n} & |B_2|_{c_n} & \cdots & |B_n|_{c_n} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix} - \lambda \begin{pmatrix} |M_{\mathfrak{B}}|_{c_1} \\ |M_{\mathfrak{B}}|_{c_2} \\ \vdots \\ |M_{\mathfrak{B}}|_{c_n} \end{pmatrix}$$

where

$$|B_i|_{c_j} = \left| \prod_{k=1, k\neq i}^{n} b_k \right|_{c_j}$$

Base extension

$$\begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix} = \begin{pmatrix} |B_1|_{c_1} & |B_2|_{c_1} & \cdots & |B_n|_{c_1} \\ |B_1|_{c_2} & |B_2|_{c_2} & \cdots & |B_n|_{c_2} \\ \vdots & \vdots & \ddots & \vdots \\ |B_1|_{c_n} & |B_2|_{c_n} & \cdots & |B_n|_{c_n} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix} - \lambda \begin{pmatrix} |M_{\mathfrak{B}}|_{c_1} \\ |M_{\mathfrak{B}}|_{c_2} \\ \vdots \\ |M_{\mathfrak{B}}|_{c_n} \end{pmatrix}$$

where

$$|B_i|_{c_j} = \left| \prod_{k=1, k \neq i}^{n} b_k \right|_{c_j}$$

Base extension

$$\begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix} = \begin{pmatrix} |B_1|_{c_1} & |B_2|_{c_1} & \cdots & |B_n|_{c_1} \\ |B_1|_{c_2} & |B_2|_{c_2} & \cdots & |B_n|_{c_2} \\ \vdots & \vdots & \ddots & \vdots \\ |B_1|_{c_n} & |B_2|_{c_n} & \cdots & |B_n|_{c_n} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix} - \lambda \begin{pmatrix} |M_{\mathfrak{B}}|_{c_1} \\ |M_{\mathfrak{B}}|_{c_2} \\ \vdots \\ |M_{\mathfrak{B}}|_{c_n} \end{pmatrix}$$

where

$$|B_i|_{c_j} = \left| \prod_{k=1,k\neq i}^{n} b_k \right|_{c_j} = \left| \prod_{k=1,k\neq i}^{n} (b_k - c_j) \right|_{c_j}$$

Base extension

$$\begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix} = \begin{pmatrix} |B_1|_{c_1} & |B_2|_{c_1} & \cdots & |B_n|_{c_1} \\ |B_1|_{c_2} & |B_2|_{c_2} & \cdots & |B_n|_{c_2} \\ \vdots & \vdots & \ddots & \vdots \\ |B_1|_{c_n} & |B_2|_{c_n} & \cdots & |B_n|_{c_n} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix} - \lambda \begin{pmatrix} |M_{\mathfrak{B}}|_{c_1} \\ |M_{\mathfrak{B}}|_{c_2} \\ \vdots \\ |M_{\mathfrak{B}}|_{c_n} \end{pmatrix}$$

where

$$|B_i|_{c_j} = \left| \prod_{k=1,k\neq i}^{n} b_k \right|_{c_j} = \left| \prod_{k=1,k\neq i}^{n} (b_k - c_j) \right|_{c_j}$$

$$\tilde{B}_{ij} := \prod_{k=1,k\neq i}^{n} (b_k - c_j) \ll 2^w$$

if $b_1, b_2, \ldots, b_n, c_1, c_2, \ldots, c_n$ are close enough.

Previously on Parameter Selection

$n = 8, w = 33$

$$
\begin{array}{lllllr}
\mathfrak{B} = \{ & 2^w - 1 & 2^w - 9 & 2^w + 3 & 2^w + 11 & \\
 & 2^w + 5 & 2^w + 9 & 2^w - 31 & 2^w + 15 & \} \\
\mathfrak{C} = \{ & 2^w & 2^w + 1 & 2^w - 3 & 2^w + 17 & \\
 & 2^w - 13 & 2^w - 21 & 2^w - 25 & 2^w - 33 & \}
\end{array}
$$

Consequently,

$$
\lceil \log_2 \tilde{B}_{ij} \rceil = \begin{pmatrix}
23 & 20 & 21 & 20 & 21 & 20 & 18 & 19 \\
20 & 18 & 20 & 18 & 19 & 18 & 16 & 17 \\
22 & 20 & 20 & 19 & 20 & 19 & 18 & 19 \\
18 & 17 & 18 & 19 & 18 & 19 & 16 & 21 \\
23 & \color{red}{24} & 22 & 22 & 22 & 22 & 22 & 21 \\
22 & 23 & 22 & 21 & 21 & 21 & 23 & 21 \\
23 & \color{red}{24} & 23 & 23 & 23 & 23 & \color{red}{24} & 23 \\
20 & 20 & 20 & 19 & 20 & 20 & \color{red}{24} & 19
\end{pmatrix}
$$

RNS Parameter Selection

$n = 4, w = 67$

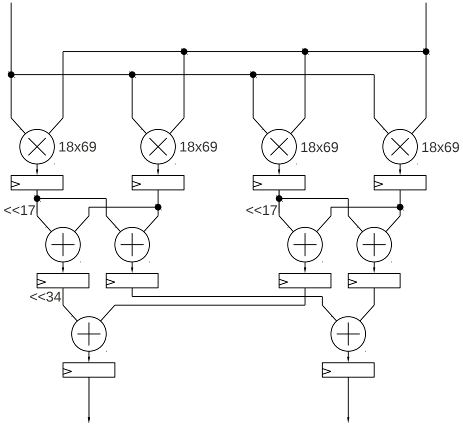
$$\mathfrak{B} = \{2^w - 1, 2^w - 7, 2^w - 9, 2^w - 15\},$$
$$\mathfrak{C} = \{2^w - 0, 2^w - 3, 2^w - 5, 2^w - 31\}.$$

Consequently,

$$\lceil \log_2 \tilde{B}_{ij} \rceil = \begin{pmatrix} 10 & 8 & 7 & 6 \\ 9 & 8 & 7 & 6 \\ 7 & 8 & 7 & 6 \\ 14 & 14 & 14 & 14 \end{pmatrix}, \lceil \log_2 \tilde{C}_{ij} \rceil = \begin{pmatrix} 8 & 7 & 6 & 4 \\ 8 & 9 & 10 & 6 \\ 10 & 10 & 11 & 8 \\ 11 & 12 & 12 & 11 \end{pmatrix}.$$

Dual Mode Multiplier

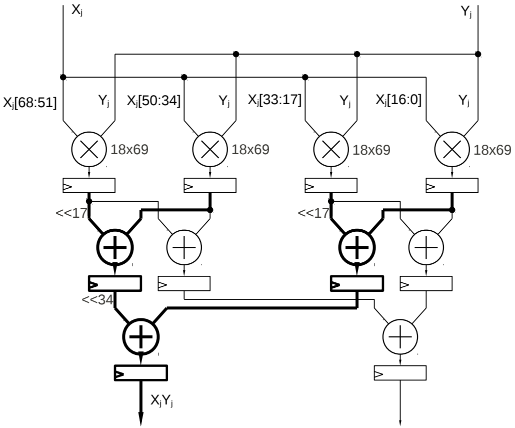
Four 18×69-bit multipliers



Dual Mode Multiplier

Four 18×69-bit multipliers

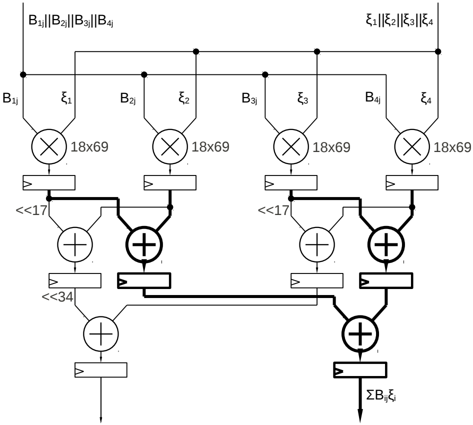
- Mode I
  - $X_j Y_j$
  - One 69×69 multiplier



Dual Mode Multiplier

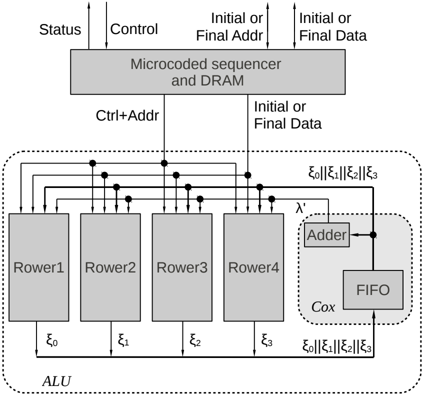
Four 18×69-bit multipliers

- Mode I
  - $X_j Y_j$
  - One 69×69 multiplier
- Mode II
  - $(\tilde{B}_{1j}\ \tilde{B}_{2j}\ \tilde{B}_{3j}\ \tilde{B}_{4j})\begin{pmatrix}\xi_1\\ \xi_2\\ \xi_3\\ \xi_4\end{pmatrix}$
  - Four 18×69 multipliers



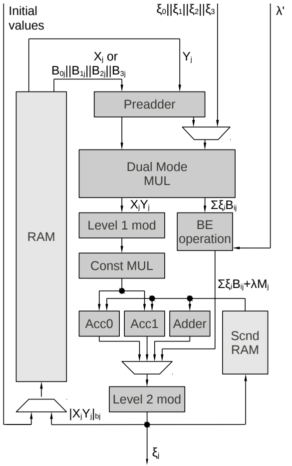
Cox-Rower Architecture [Kawamura *et al.* 00]

- 4 rowers
  - MUL: 2 cycles
  - RED: 6 cycles
- One rower, one dual-mode mul
- Microcoded sequencer



Rower Design

- Dual mode multiplier
- Preadder
- 2 accumulators
- Small-constant multiplier
- 3-port RAMs
- 2-level channel reduction
- Same data path
    - MUL
    - RED



Comparison

Design	Pairing/ Security[bit]	Platform	Algorithm	Area	Freq. [MHz]	Cycle	Delay [ms]
This Work	optimal ate 126	Xilinx (Virtex-6)	RNS (Parallel)	5237 slices 64 DSPs	230	77,769	**0.338**
Cheung+11	optimal ate 126	Xilinx (Virtex-6)	RNS (Parallel)	7032 slices 32 DSPs	250	143,111	**0.573**
Fan+11	ate/128	Xilinx (Virtex-6)	HMM (Parallel)	4014 slices 42 DSPs	210	336,366	1.60
	opt. ate/128					245,430	1.17
Estibals'10	Tate $\mathbb{F}_{3^{5\cdot97}}$ 128	Xilinx (Virtex-4)	-	4755 Slices 7 BRAMs	192	428,853	2.23
Ghosh+11	η_T over $\mathbb{F}_{2^{1223}}$ 128	Xilinx FPGA (Virtex-6)	-	15167 Slices	250	47,610	**0.19**
Beuchat+10	optimal ate 126	Core i7	Montgomery	-	2800	2,330,000	0.83
Aranha+11	optimal ate 126	Phenom II	Montgomery	-	3000	1,562,000	**0.52**
Aranha+10	η_T over $\mathbb{F}_{2^{1223}}$ 128	Xeon (8 cores)	-	-	2000	3,020,000	1.51
Aranha+10	opt. Eta $\mathbb{F}_{2^{367}}$ 128	Core i5	-	-	2530	2,440,000	0.96

Thank you!