CINVESTAV-IPN, November 2009

Fast hardware accelerator for the Tate pairing based on a fully parallel Karatsuba multiplier

Nicolas Estibals

CACAO project-team, LORIA

INRIA Nancy – Grand-Est

Nicolas.Estibals@loria.fr

Joint work with:

Jean-Luc Beuchat	LCIS, University of Tsukuba, Japan
Jérémie Detrey	CACAO, LORIA, Nancy, France
Eiji Okamoto	LCIS, University of Tsukuba, Japan
Francisco Rodríguez-Henríquez	CINVESTAV-IPN, Mexico City, Mexico

Outline of the talk

- Context
- Reduced Tate pairing
- Non-reduced Tate pairing
- Fully parallel Karatsuba multiplier
- Final Exponentiation
- Results & Conclusion
- Appendix

Pairing-based cryptography

- Origin of pairings in cryptography
  - attack against some elliptic curves
    - Menezes–Okamoto–Vanstone, 1993
    - Frey–Rück, 1994

Pairing-based cryptography

- Origin of pairings in cryptography
  - attack against some elliptic curves
    - Menezes–Okamoto–Vanstone, 1993
    - Frey–Rück, 1994
- Constructive properties
  - One-round three party key exchange
    - Joux, 2000
  - short digital signature
    - Boneh–Lynn–Shacham, 2001
    - Zang–Safavi–Naini–Susilo, 2004
  - identity-based encryption
    - Boneh–Franklin, 2001
    - Sakai–Kasahara, 2001
  - ...

Pairing-based cryptography

- Origin of pairings in cryptography
  - attack against some elliptic curves
    - Menezes–Okamoto–Vanstone, 1993
    - Frey–Rück, 1994
- Constructive properties
  - One-round three party key exchange
    - Joux, 2000
  - short digital signature
    - Boneh–Lynn–Shacham, 2001
    - Zang–Safavi–Naini–Susilo, 2004
  - identity-based encryption
    - Boneh–Franklin, 2001
    - Sakai–Kasahara, 2001
  - ...
- Standardization in progress
  - ISO/IEC 14888-3
  - IEEE P1363.3

Which pairing?

- Reduced Tate pairing
  - common choice for cryptographic applications

Which pairing?

- Reduced Tate pairing
  - common choice for cryptographic applications
- Pairing on supersingular curves
  - \+ easier arithmetic on the curve
  - − lower security
- Small characteristic
  - higher embedding degree
  - higher security

Which pairing?

- Reduced Tate pairing
  - common choice for cryptographic applications
- Pairing on supersingular curves
  - \+ easier arithmetic on the curve
  - − lower security
- Small characteristic
  - higher embedding degree
  - higher security
- Need for dedicated hardware coprocessor
  - area optimized (RFID, embedded systems, ...)
  - speed optimized (bank servers, ...)

Which pairing?

- Reduced Tate pairing
  - common choice for cryptographic applications
- Pairing on supersingular curves
  - \+ easier arithmetic on the curve
  - − lower security
- Small characteristic
  - higher embedding degree
  - higher security
- Need for dedicated hardware coprocessor
  - area optimized (RFID, embedded systems, . . .)
  - speed optimized (bank servers, . . .)

Outline of the talk

- Context
- Reduced Tate pairing
- Non-reduced Tate pairing
- Fully parallel Karatsuba multiplier
- Final Exponentiation
- Results & Conclusion
- Appendix

Pairing is a bilinear map

- $\mathbb{G}_1 = \langle P \rangle$: additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- $\mathbb{G}_2$: multiplicatively-written cyclic group of order $\#\mathbb{G}_2 = \#\mathbb{G}_1 = \ell$

Pairing is a bilinear map

- $\mathbb{G}_1 = \langle P \rangle$: additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- $\mathbb{G}_2$: multiplicatively-written cyclic group of order $\#\mathbb{G}_2 = \#\mathbb{G}_1 = \ell$
- $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear pairing iff:
  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$
  - bilinearity:
    - $\hat{e}(Q_1 + Q_2, R) = \hat{e}(Q_1, R) \cdot \hat{e}(Q_2, R)$
    - $\hat{e}(Q, R_1 + R_2) = \hat{e}(Q, R_1) \cdot \hat{e}(Q, R_2)$
  - computability: $\hat{e}$ can be efficiently computed

Pairing is a bilinear map

- $\mathbb{G}_1 = \langle P \rangle$: additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- $\mathbb{G}_2$: multiplicatively-written cyclic group of order $\#\mathbb{G}_2 = \#\mathbb{G}_1 = \ell$
- $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear pairing iff:
  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$
  - bilinearity:
    - $\hat{e}(Q_1 + Q_2, R) = \hat{e}(Q_1, R) \cdot \hat{e}(Q_2, R)$
    - $\hat{e}(Q, R_1 + R_2) = \hat{e}(Q, R_1) \cdot \hat{e}(Q, R_2)$
  - computability: $\hat{e}$ can be efficiently computed
- Important property for cryptographic applications:

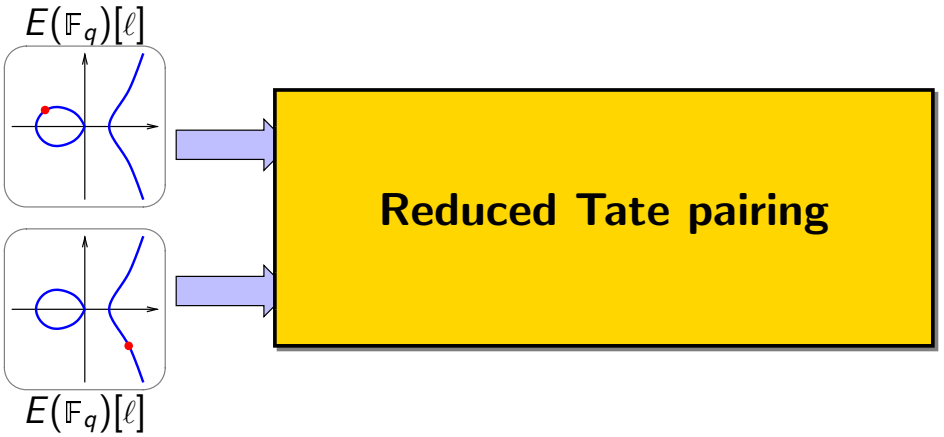
$$\hat{e}(k_1 P, k_2 P) = \hat{e}(k_2 P, k_1 P) = \hat{e}(P, P)^{k_1 k_2}$$

Combining secrets without having to reveal them!

Pairing over elliptic curve

Reduced Tate pairing

Pairing over elliptic curve



- Input: two points P and Q in $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell]$, where:
  - $q = p$, 2^m or 3^m
  - E is an elliptic curve over $\mathbb{F}_q$
  - ℓ is a large prime factor of $\#E(\mathbb{F}_q)$
  - $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell] = \{P \in E(\mathbb{F}_q) | \ell P = \mathcal{O}\}$

Pairing over elliptic curve

- Input: two points P and Q in $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell]$, where:
  - $q = p$, 2^m or 3^m
  - E is an elliptic curve over $\mathbb{F}_q$
  - ℓ is a large prime factor of $\#E(\mathbb{F}_q)$
  - $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell] = \{P \in E(\mathbb{F}_q) | \ell P = \mathcal{O}\}$
- Output: an ℓ-th root of unity
  - $\mathbb{G}_2 = \mu_\ell = \left\{U \in \overline{\mathbb{F}_q}^\times | U^\ell = 1\right\}$
  - k is the embedding degree: the smallest integer such that $\mu_\ell \subseteq \mathbb{F}_{q^k}^\times$

Security considerations

$$\hat{e} : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_q)[\ell] \to \mu_\ell \subseteq \mathbb{F}_{q^k}^\times$$

- Security should be enough in $\mathbb{G}_1$ and in $\mathbb{G}_2$
- Supersingular curves: k is bounded
- Some typical cases:

Base field $(\mathbb{F}_q)$	$\mathbb{F}_{2^m}$	$\mathbb{F}_{3^m}$	$\mathbb{F}_p$
Embedding degree (k)	4	6	2
Lower security $(\sim 2^{64})$	$m = 239$	$m = 97$	$\lvert p\rvert \sim 320$
Medium security $(\sim 2^{80})$	$m = 373$	$m = 163$	$\lvert p\rvert \sim 500$
Higher security $(\sim 2^{128})$	$m = 1103$	$m = 503$	$\lvert p\rvert \sim 1400$

Security considerations

$$\hat{e} : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_q)[\ell] \to \mu_\ell \subseteq \mathbb{F}_{q^k}^\times$$

- Security should be enough in $\mathbb{G}_1$ and in $\mathbb{G}_2$
- Supersingular curves: k is bounded
- Some typical cases:

Base field $(\mathbb{F}_q)$	$\mathbb{F}_{2^m}$	$\mathbb{F}_{3^m}$	$\mathbb{F}_p$
Embedding degree (k)	4	6	2
Lower security $(\sim 2^{64})$	$m = 239$	$m = 97$	$\lvert p\rvert \sim 320$
Medium security $(\sim 2^{80})$	$m = 373$	$m = 163$	$\lvert p\rvert \sim 500$
Higher security $(\sim 2^{128})$	$m = 1103$	$m = 503$	$\lvert p\rvert \sim 1400$

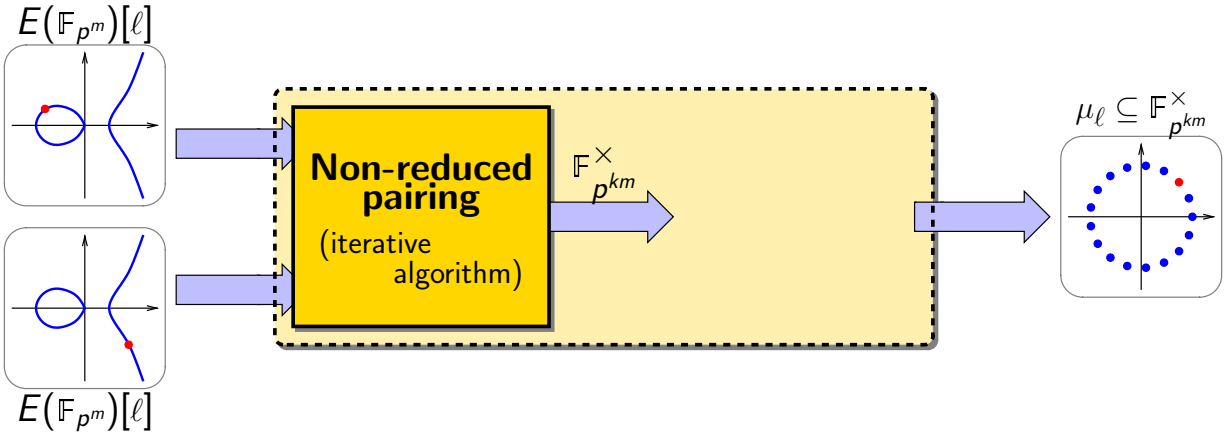
- $\mathbb{G}_2 = \mu_\ell \subseteq \mathbb{F}_{q^k}^\times$ is the bottleneck
- Low characteristic ($p = 2$ or 3) because of higher embedding degree

Reduced Tate pairing

Reduced Tate pairing

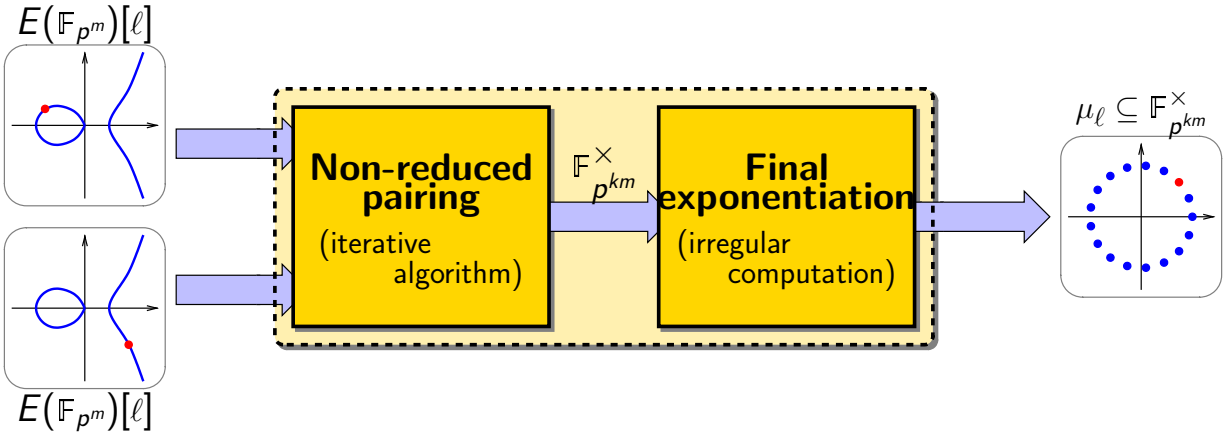
- Two very different steps:

Reduced Tate pairing



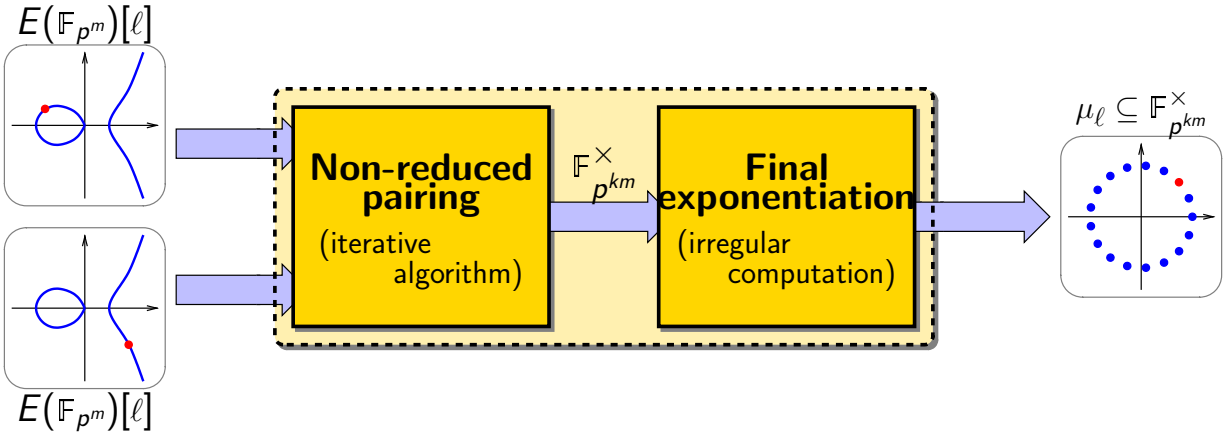
- Two very different steps:
  - non-reduced pairing: Miller's iterative algorithm

Reduced Tate pairing



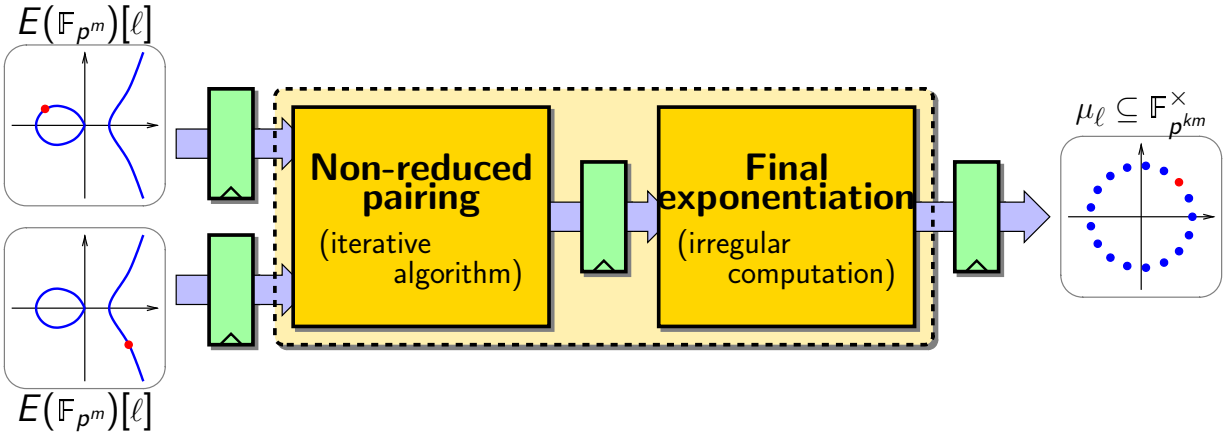
- Two very different steps:
  - non-reduced pairing: Miller's iterative algorithm
  - final exponention: irregular computation

Reduced Tate pairing



- Two very different steps:
  - non-reduced pairing: Miller's iterative algorithm
  - final exponention: irregular computation
- Idea: use two separate coprocessors

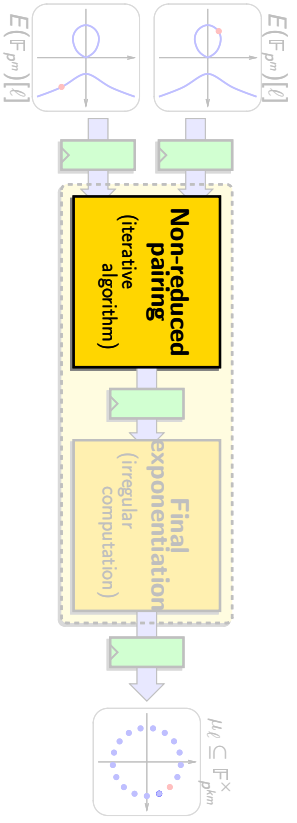
Reduced Tate pairing



- Two very different steps:
  - non-reduced pairing: Miller's iterative algorithm
  - final exponention: irregular computation
- Idea: use two separate coprocessors
  - pipeline the two computations
  - balance the latencies

Outline of the talk

- Context
- Reduced Tate pairing
- Non-reduced Tate pairing
- Fully parallel Karatsuba multiplier
- Final Exponentiation
- Results & Conclusion
- Appendix



A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

$$x_P \leftarrow \sqrt[3]{x_P} \quad ; \quad y_P \leftarrow \sqrt[3]{y_P}$$
$$x_Q \leftarrow x_Q^3 \quad ; \quad y_Q \leftarrow y_Q^3$$

$$t \leftarrow x_P + x_Q \qquad u \leftarrow y_P y_Q$$
$$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$$

$$R \leftarrow R \cdot S$$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

① $x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ — $2\sqrt[3]{\cdot}$
$x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$ — $2\,(\cdot)^3$

$t \leftarrow x_P + x_Q \quad u \leftarrow y_P y_Q$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$

$R \leftarrow R \cdot S$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates
  2. computation of line equation

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

① $x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ — $2\ \sqrt[3]{\cdot}$
$x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$ — $2\ (\cdot)^3$

② $t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ — $2\times, 2+$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$

$R \leftarrow R \cdot S$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates
  2. computation of line equation
  3. accumulation of the new factor

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$1\times\ (\mathbb{F}_{3^{6m}})$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates
  2. computation of line equation
  3. accumulation of the new factor
- Multiplication is critical
- Fully parallel, pipelined multiplier over $\mathbb{F}_{3^m}$

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$1\times\ (\mathbb{F}_{3^{6m}})$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates
  2. computation of line equation
  3. accumulation of the new factor
- Multiplication is critical
- Fully parallel, pipelined multiplier over $\mathbb{F}_{3^m}$
- Sparse multiplication over $\mathbb{F}_{3^{6m}}$

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$1\times\ (\mathbb{F}_{3^{6m}})$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates
  2. computation of line equation
  3. accumulation of the new factor
- Multiplication is critical
- Fully parallel, pipelined multiplier over $\mathbb{F}_{3^m}$
- Sparse multiplication over $\mathbb{F}_{3^{6m}}$
  - $12\times$ and $59+$ over $\mathbb{F}_{3^m}$ (Gorla *et al.*, SAC 2007)

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$12\times, 59+$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates
  2. computation of line equation
  3. accumulation of the new factor
- Multiplication is critical
- Fully parallel, pipelined multiplier over $\mathbb{F}_{3^m}$
- Sparse multiplication over $\mathbb{F}_{3^{6m}}$
  - $12\times$ and $59+$ over $\mathbb{F}_{3^m}$ (Gorla *et al.*, SAC 2007)
  - $15\times$ and $29+$ over $\mathbb{F}_{3^m}$ (Beuchat *et al.*, ARITH 18)

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$15\times, 29+$

end for

A closer look on Miller's loop (char. 3)

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  1. update of point coordinates
  2. computation of line equation
  3. accumulation of the new factor
- Multiplication is critical
- Fully parallel, pipelined multiplier over $\mathbb{F}_{3^m}$

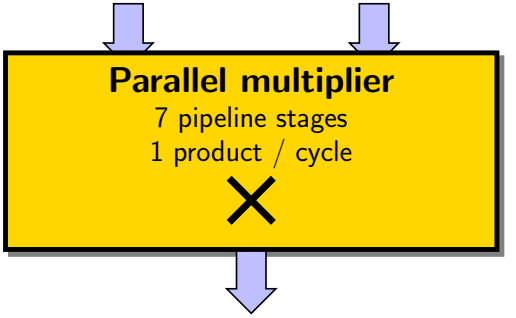
for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$15\times, 29+$

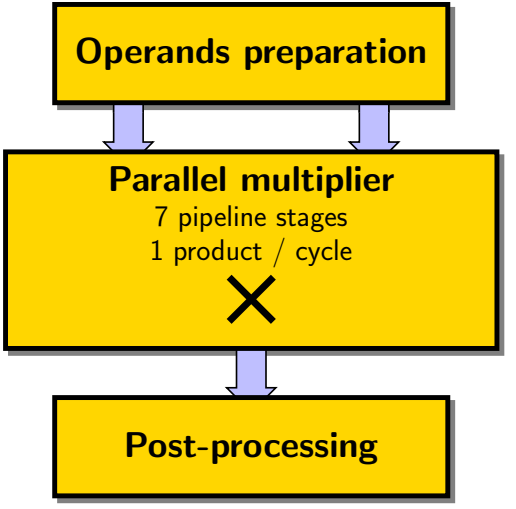
end for

- Sparse multiplication over $\mathbb{F}_{3^{6m}}$
  - $12\times$ and $59+$ over $\mathbb{F}_{3^m}$ (Gorla *et al.*, SAC 2007)
  - $15\times$ and $29+$ over $\mathbb{F}_{3^m}$ (Beuchat *et al.*, ARITH 18)
- Objective: keep the multiplier pipeline busy
  - 7-stages pipeline
  - one product per cycle
  - 17 cycles per iteration

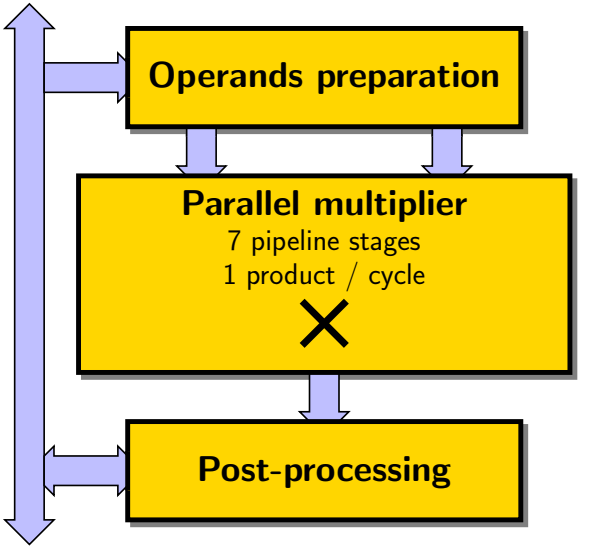
Coprocessor for the non-reduced pairing (char. 3)



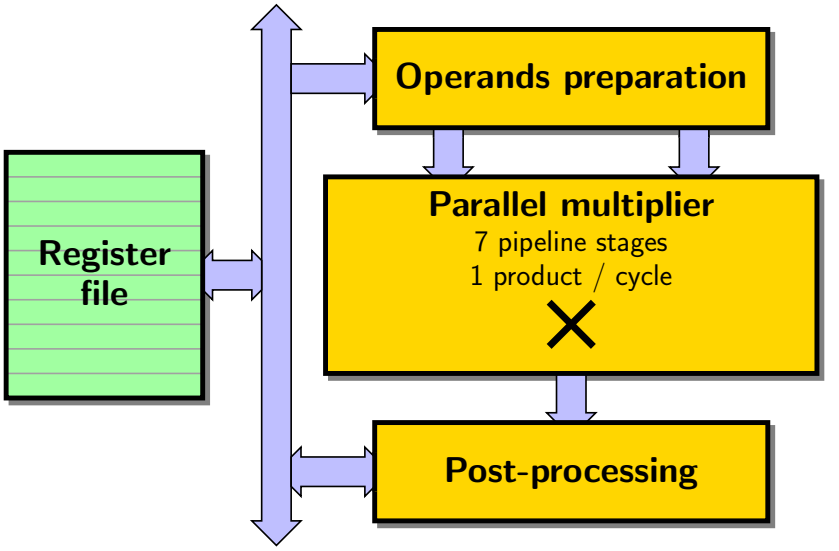
Coprocessor for the non-reduced pairing (char. 3)



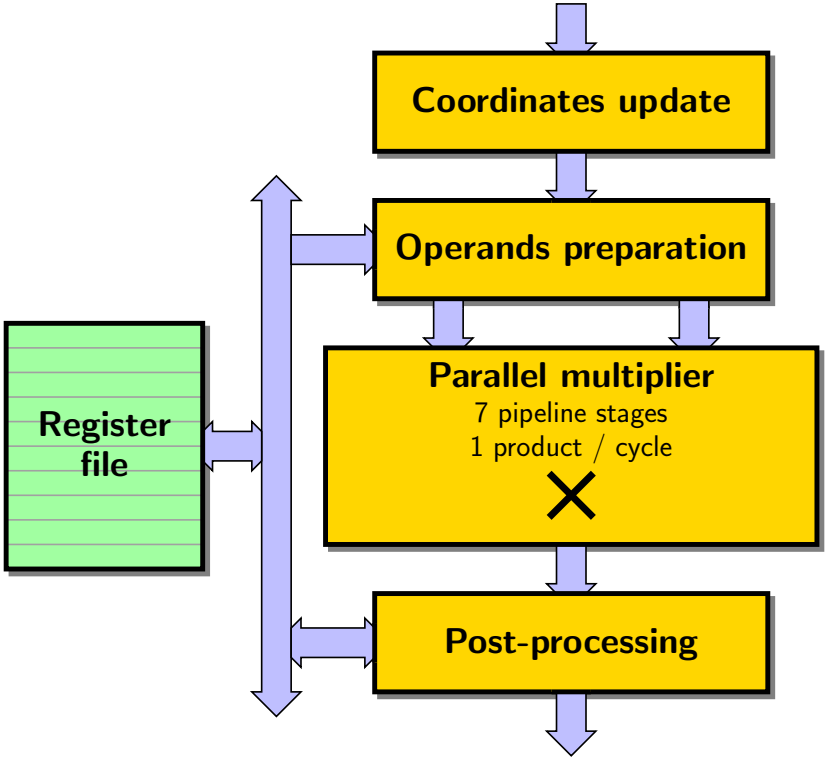
Coprocessor for the non-reduced pairing (char. 3)



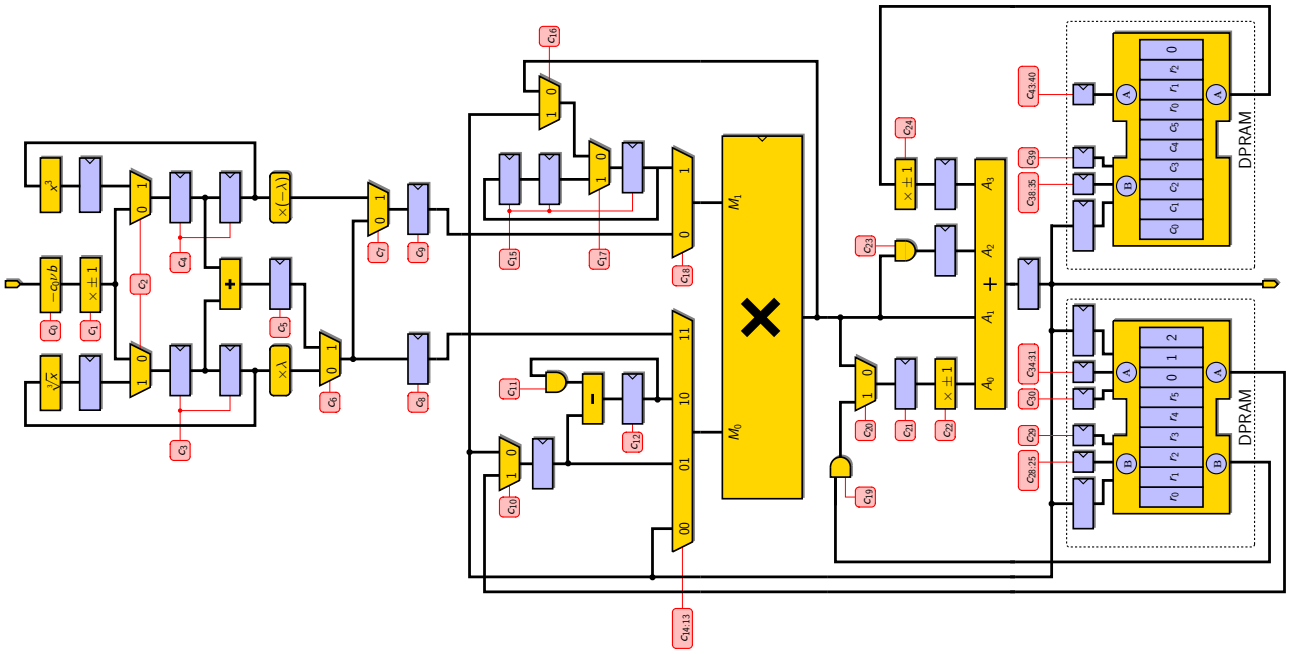
Coprocessor for the non-reduced pairing (char. 3)



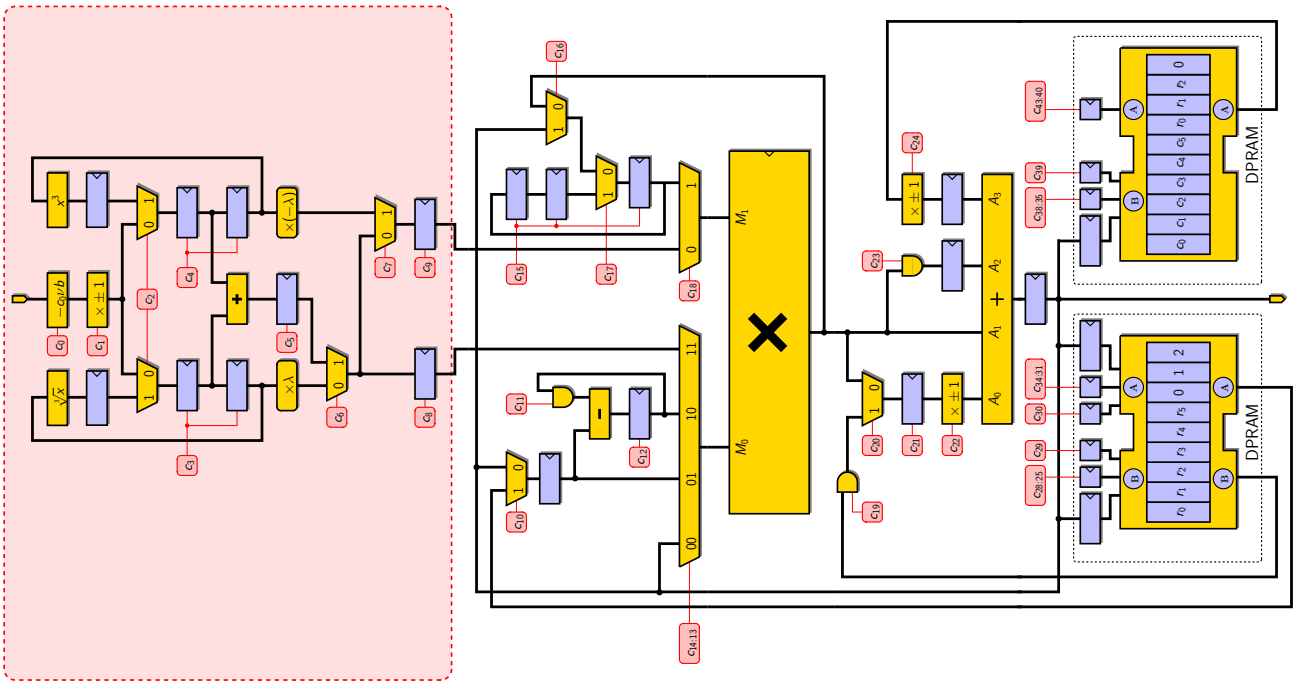
Coprocessor for the non-reduced pairing (char. 3)



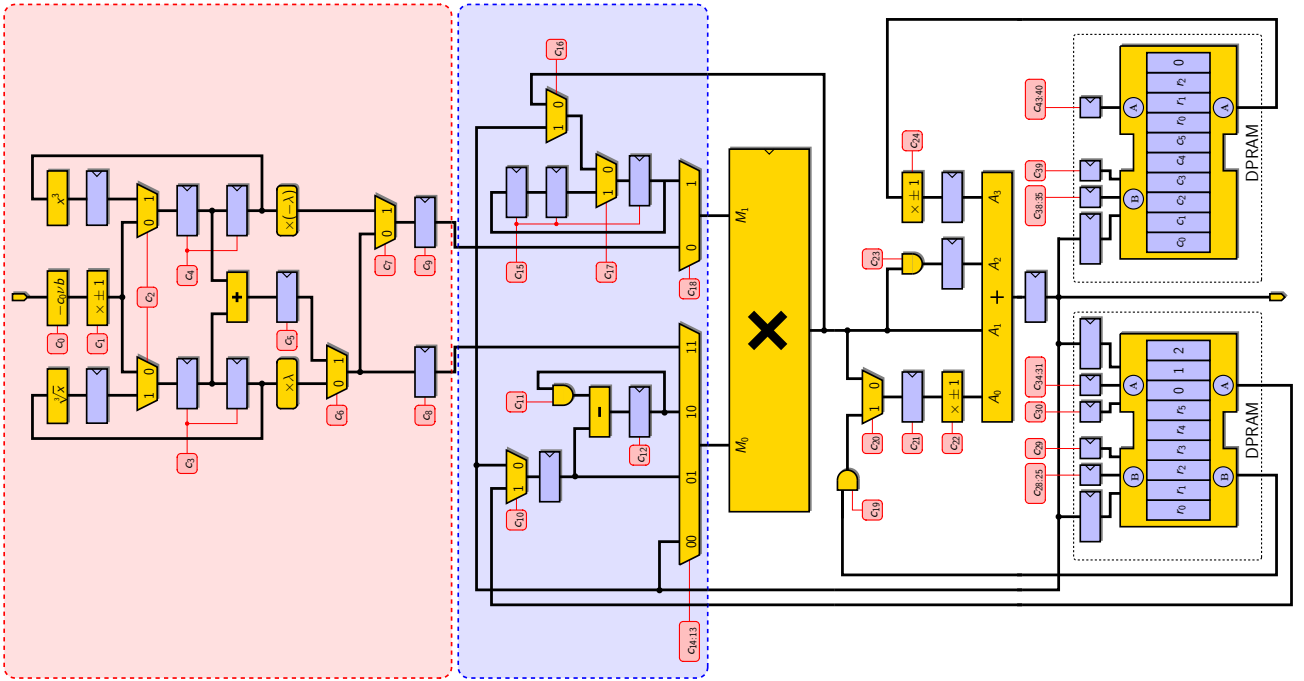
Detailed architecture of the coprocessor (char. 3)



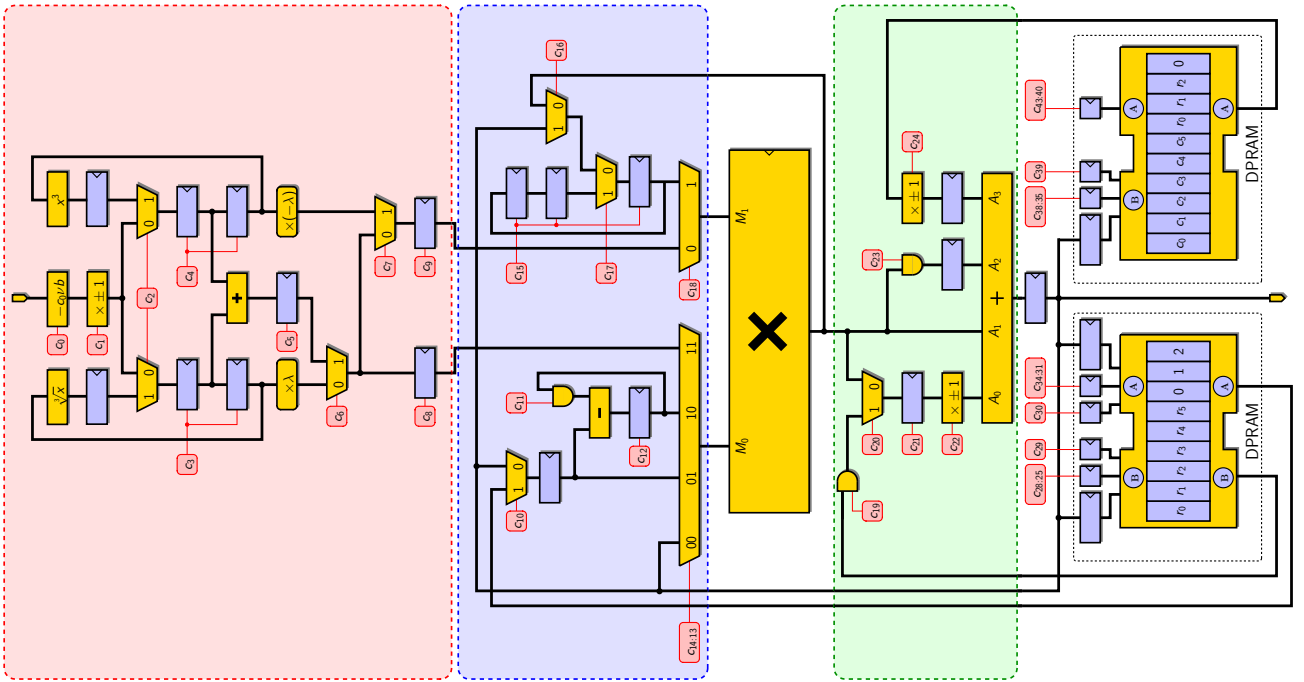
Detailed architecture of the coprocessor (char. 3)



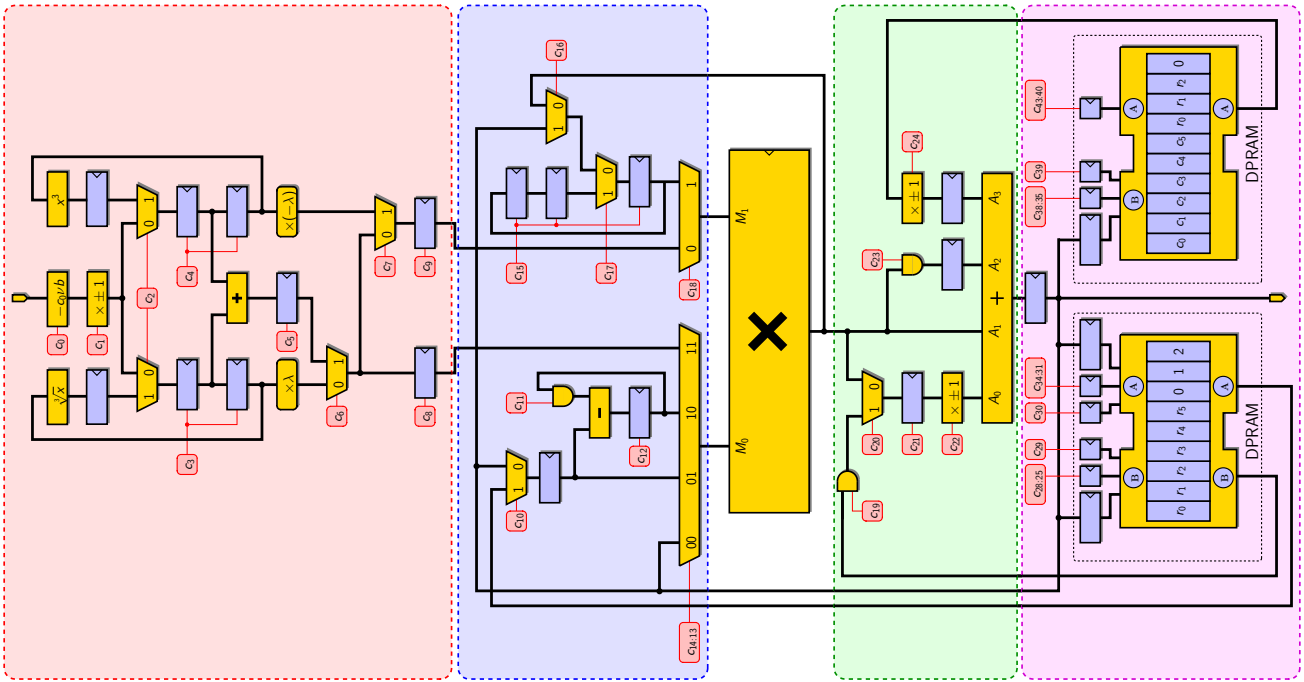
Detailed architecture of the coprocessor (char. 3)



Detailed architecture of the coprocessor (char. 3)

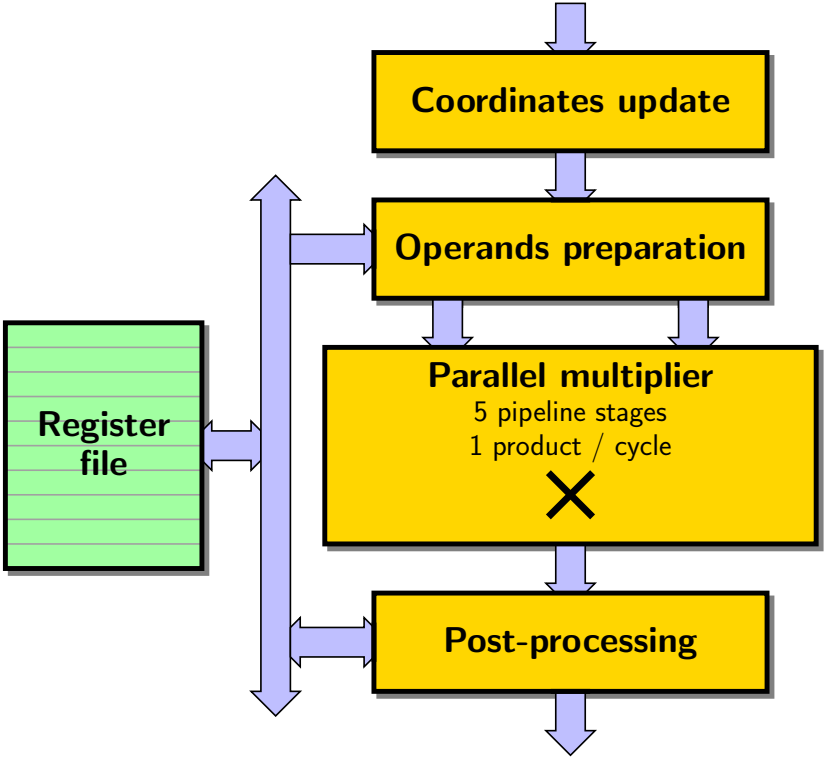


Detailed architecture of the coprocessor (char. 3)



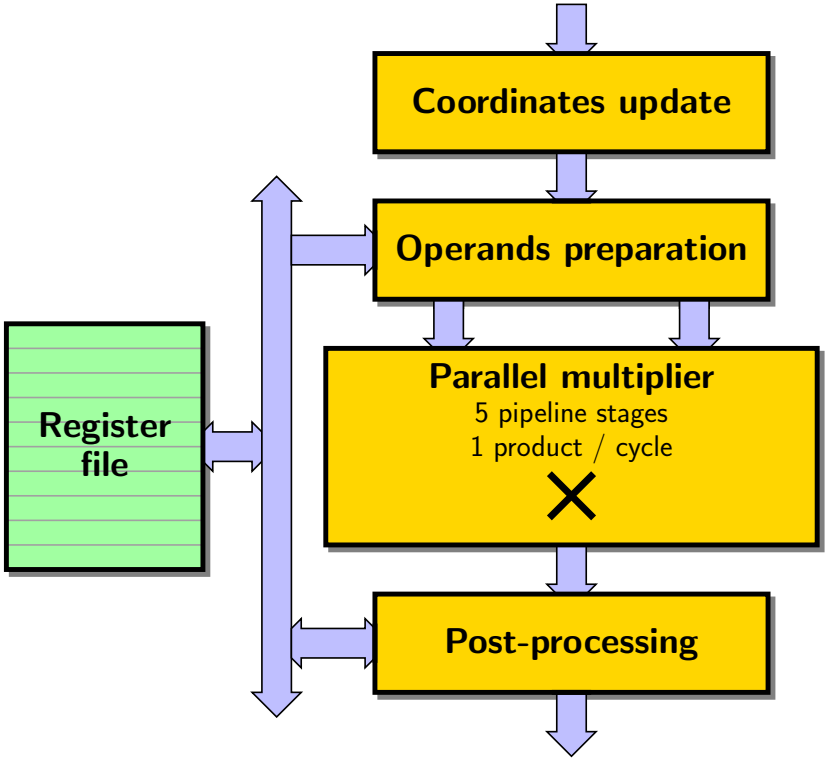
Coprocessor for the non-reduced pairing (char. 2)

- Similar algorithm
  - $\frac{m+1}{2}$ iterations



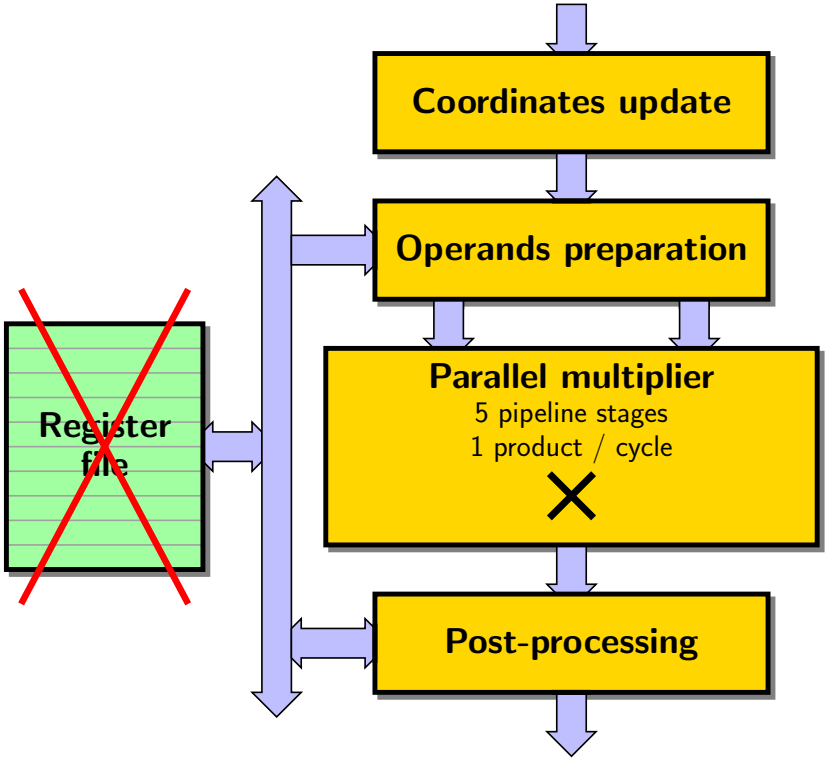
Coprocessor for the non-reduced pairing (char. 2)

- Similar algorithm
  - $\frac{m+1}{2}$ iterations
- But some differences:
  - only 6 products over $\mathbb{F}_{2^m}$ per iteration
  - different scheduling
  - 5-stages multiplier pipeline



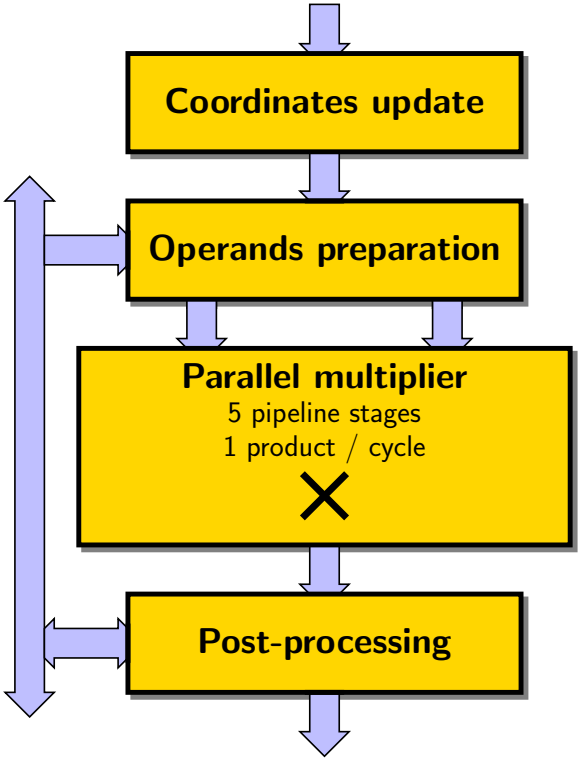
Coprocessor for the non-reduced pairing (char. 2)

- Similar algorithm
  - $\frac{m+1}{2}$ iterations
- But some differences:
  - only 6 products over $\mathbb{F}_{2^m}$ per iteration
  - different scheduling
  - 5-stages multiplier pipeline
- No need for a register file

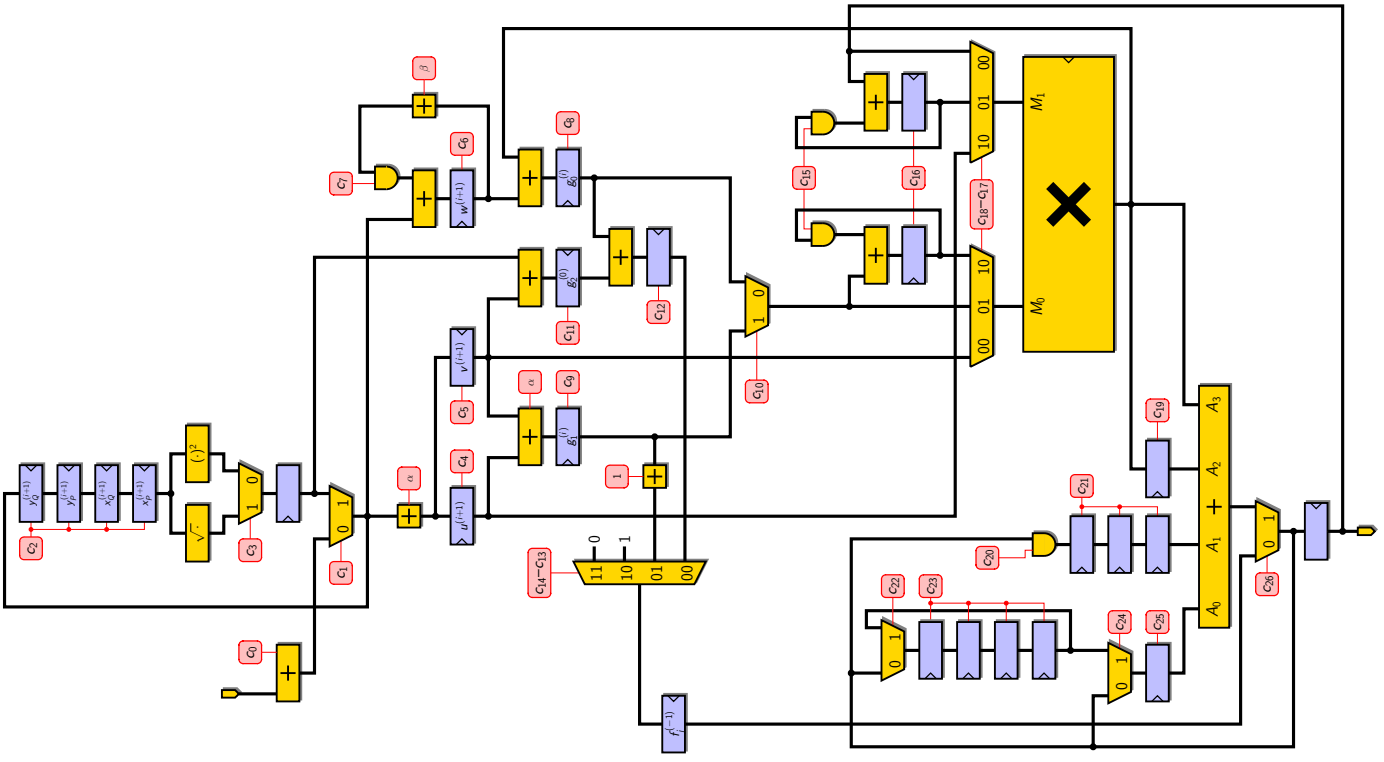


Coprocessor for the non-reduced pairing (char. 2)

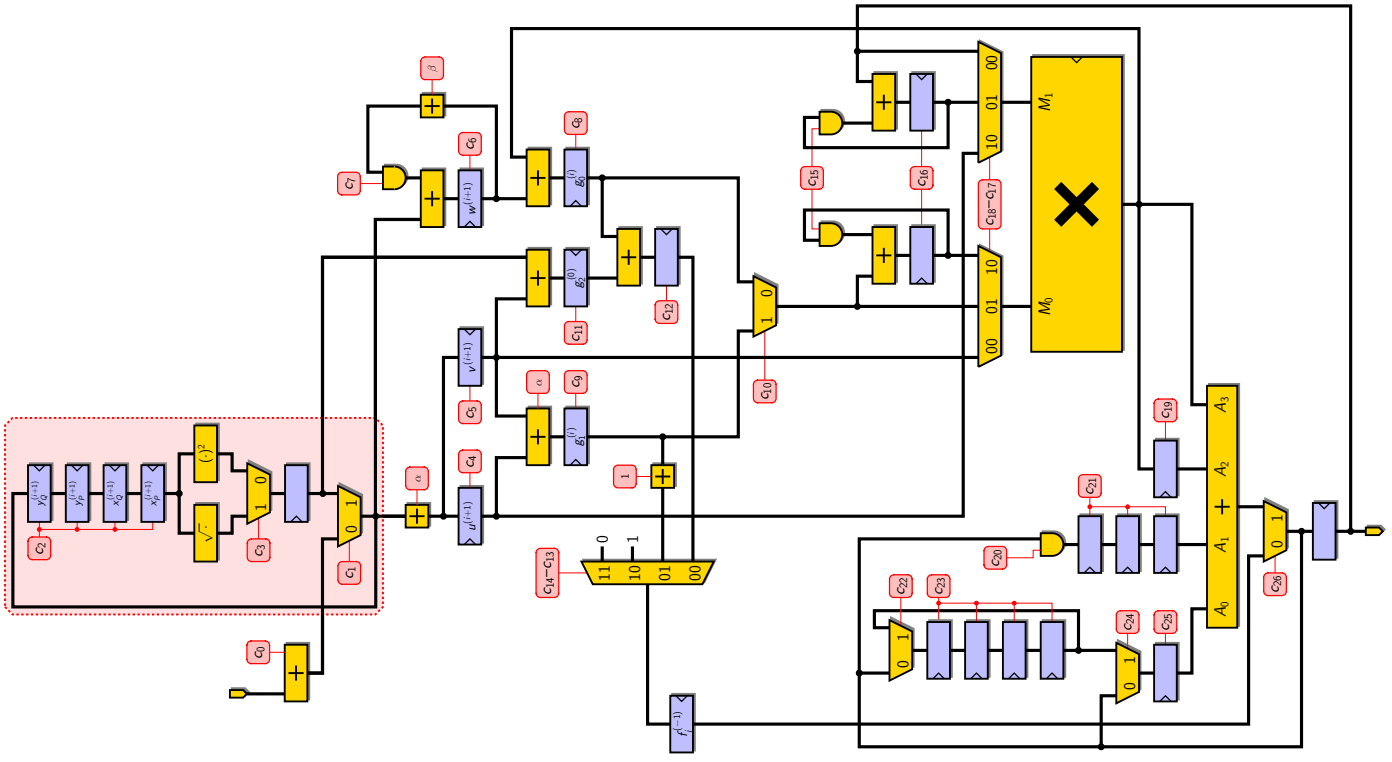
- Similar algorithm
  - $\frac{m+1}{2}$ iterations
- But some differences:
  - only 6 products over $\mathbb{F}_{2^m}$ per iteration
  - different scheduling
  - 5-stages multiplier pipeline
- No need for a register file
  - all data in shift register
  - more complex architecture



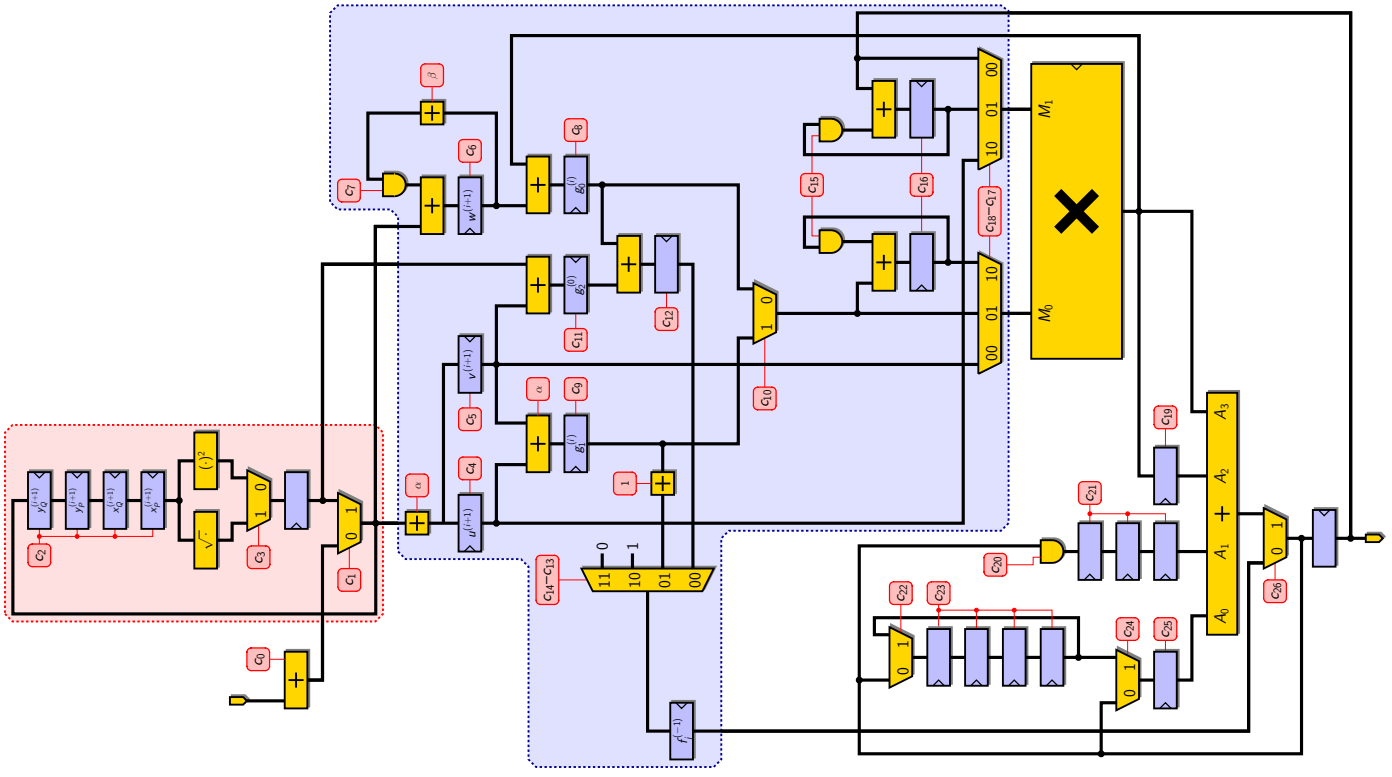
Detailed architecture of the coprocessor (char. 2)



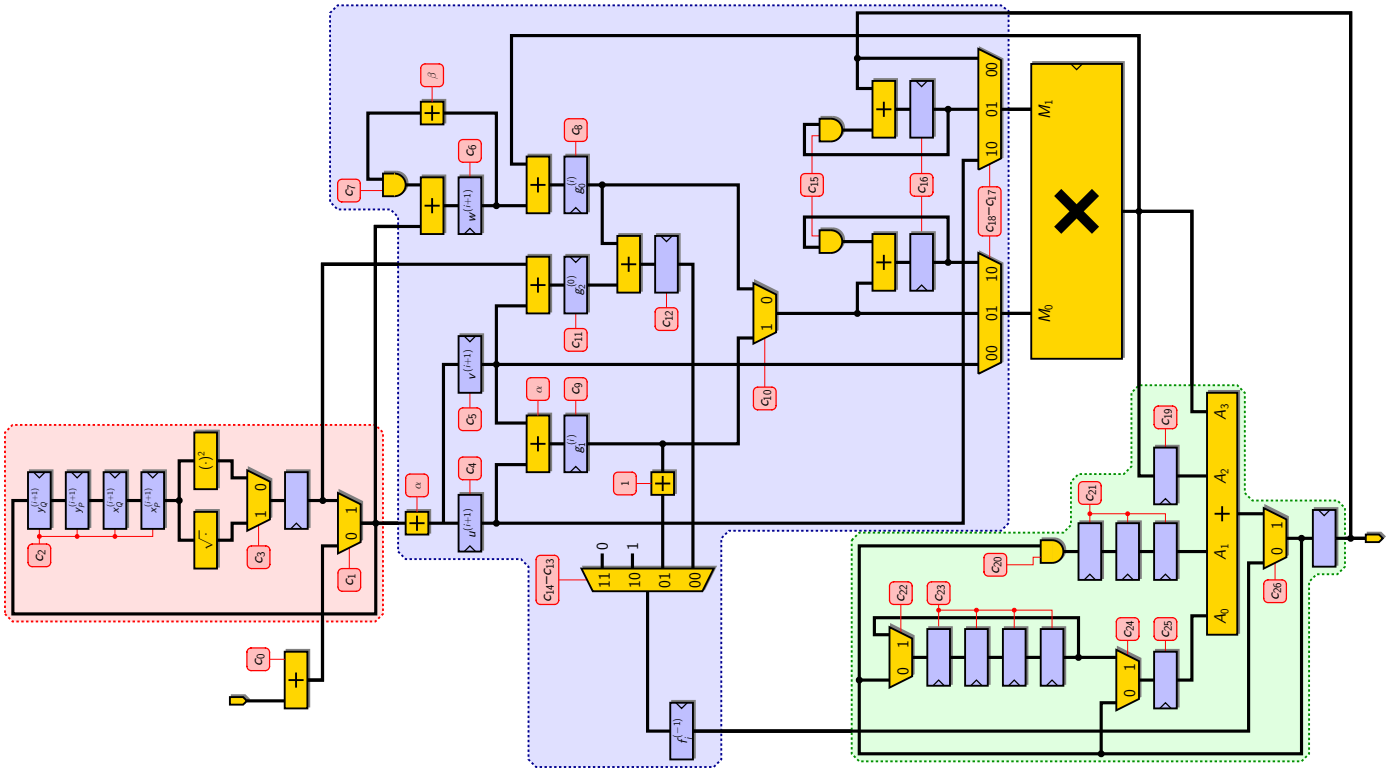
Detailed architecture of the coprocessor (char. 2)



Detailed architecture of the coprocessor (char. 2)

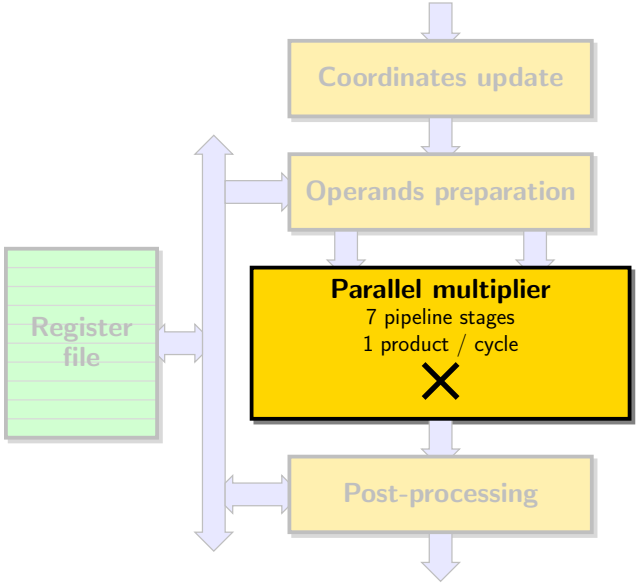


Detailed architecture of the coprocessor (char. 2)



Outline of the talk

- Context
- Reduced Tate pairing
- Non-reduced Tate pairing
- Fully parallel Karatsuba multiplier
- Final Exponentiation
- Results & Conclusion
- Appendix

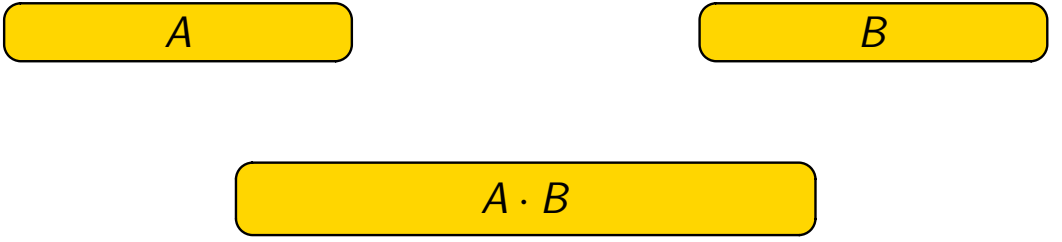


Finite field representation and Karatsuba's formula

- Polynomial basis:
  - $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ irreducible polynomial of degree m in $\mathbb{F}_p[x]$
  - $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]^{\leqslant(m-1)}$

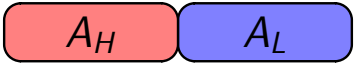
Finite field representation and Karatsuba's formula

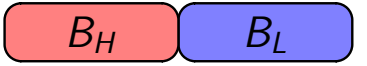
- Polynomial basis:
  - $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ irreducible polynomial of degree m in $\mathbb{F}_p[x]$
  - $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]^{\leqslant(m-1)}$
- Karatsuba algorithm for polynomials



Finite field representation and Karatsuba's formula

- Polynomial basis:
  - $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ irreducible polynomial of degree m in $\mathbb{F}_p[x]$
  - $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]^{\leqslant(m-1)}$
- Karatsuba algorithm for polynomials





$$A_H B_H X^{2n} + (A_H B_L + A_L B_H) X^n + A_L B_L$$

Finite field representation and Karatsuba's formula

- Polynomial basis:
  - $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ irreducible polynomial of degree m in $\mathbb{F}_p[x]$
  - $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]^{\leqslant(m-1)}$
- Karatsuba algorithm for polynomials

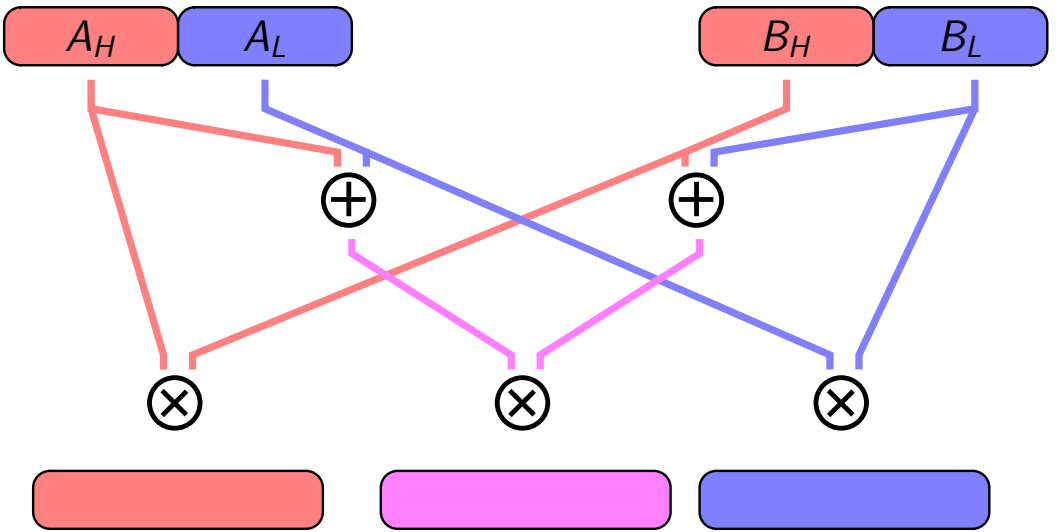
$$A_H B_H X^{2n} + (A_H B_L + A_L B_H) X^n + A_L B_L$$

$$ab' + a'b = (a + a')(b + b') - ab - a'b'$$

$$A_H B_H X^{2n} + ((A_H + A_L)(B_H + B_L) - A_H B_H - A_L B_L) X^n + A_L B_L$$

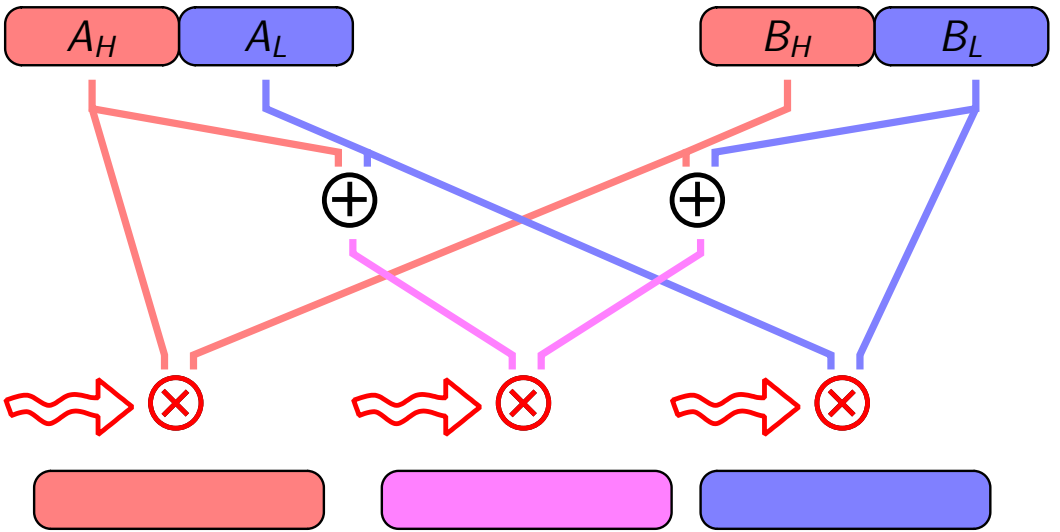
Finite field representation and Karatsuba's formula

- Polynomial basis:
  - $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ irreducible polynomial of degree m in $\mathbb{F}_p[x]$
  - $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]^{\leqslant(m-1)}$
- Karatsuba algorithm for polynomials



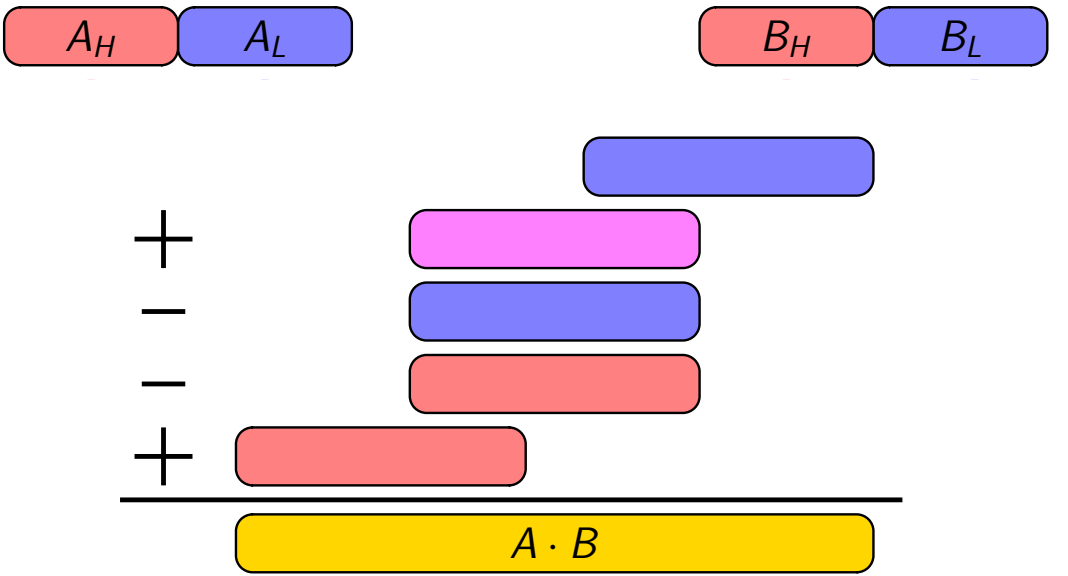
Finite field representation and Karatsuba's formula

- Polynomial basis:
  - $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ irreducible polynomial of degree m in $\mathbb{F}_p[x]$
  - $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]^{\leqslant(m-1)}$
- Karatsuba algorithm for polynomials

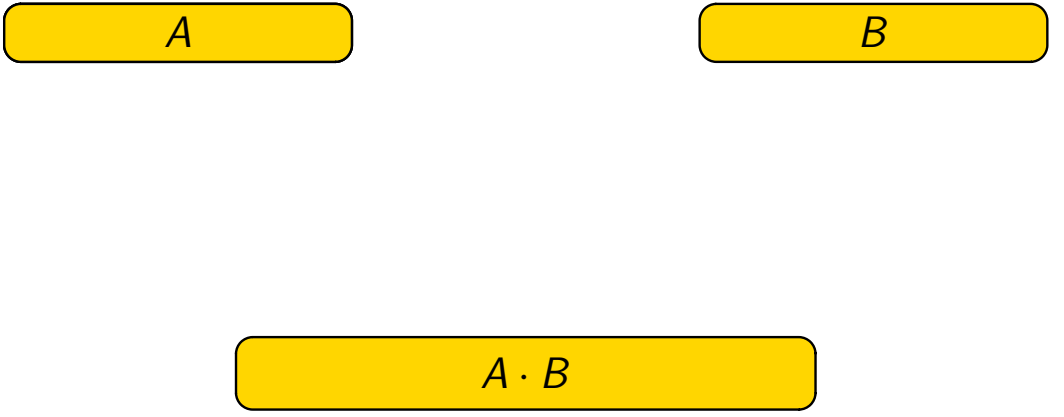


Finite field representation and Karatsuba's formula

- Polynomial basis:
  - $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$ irreducible polynomial of degree m in $\mathbb{F}_p[x]$
  - $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]^{\leqslant(m-1)}$
- Karatsuba algorithm for polynomials



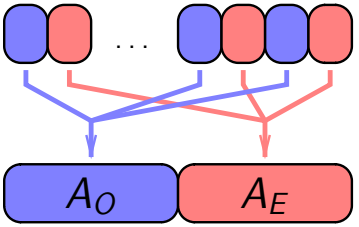
Odd–even split for Karatsuba multiplication

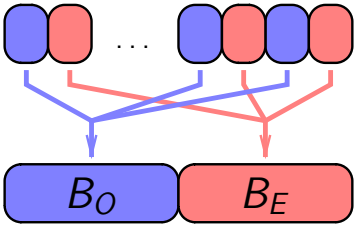


Odd–even split for Karatsuba multiplication

$$(A_O B_O X^2 + A_E B_E) + X(A_O B_E + A_E B_O)$$

Odd–even split for Karatsuba multiplication



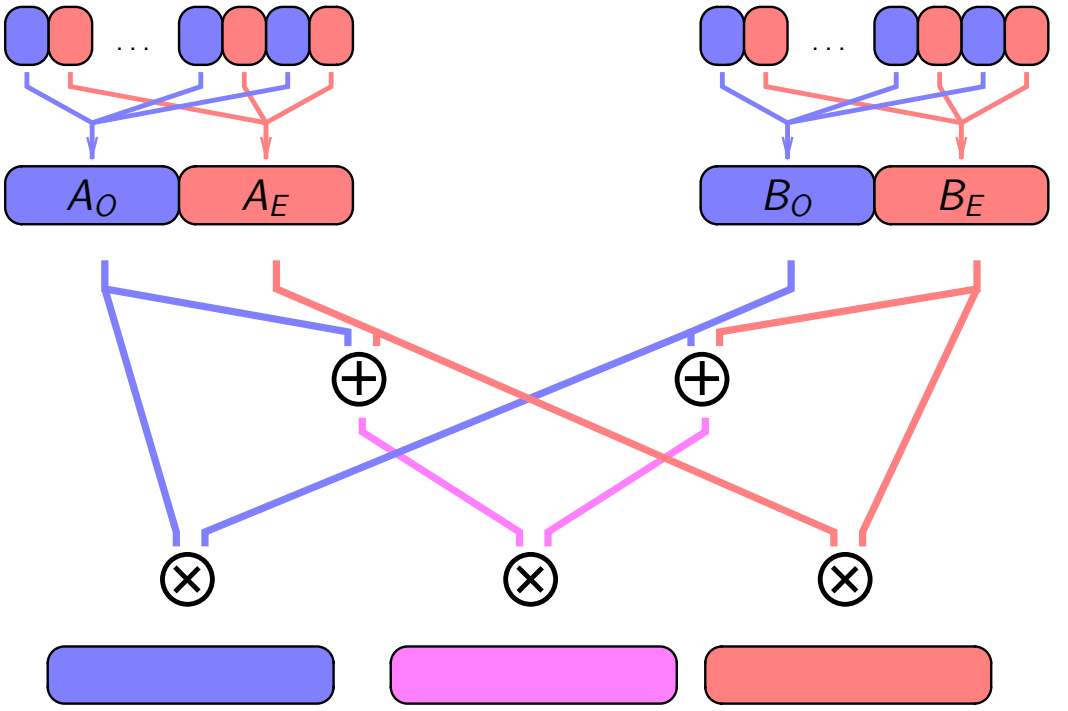


$$(A_O B_O X^2 + A_E B_E) + X(A_O B_E + A_E B_O)$$

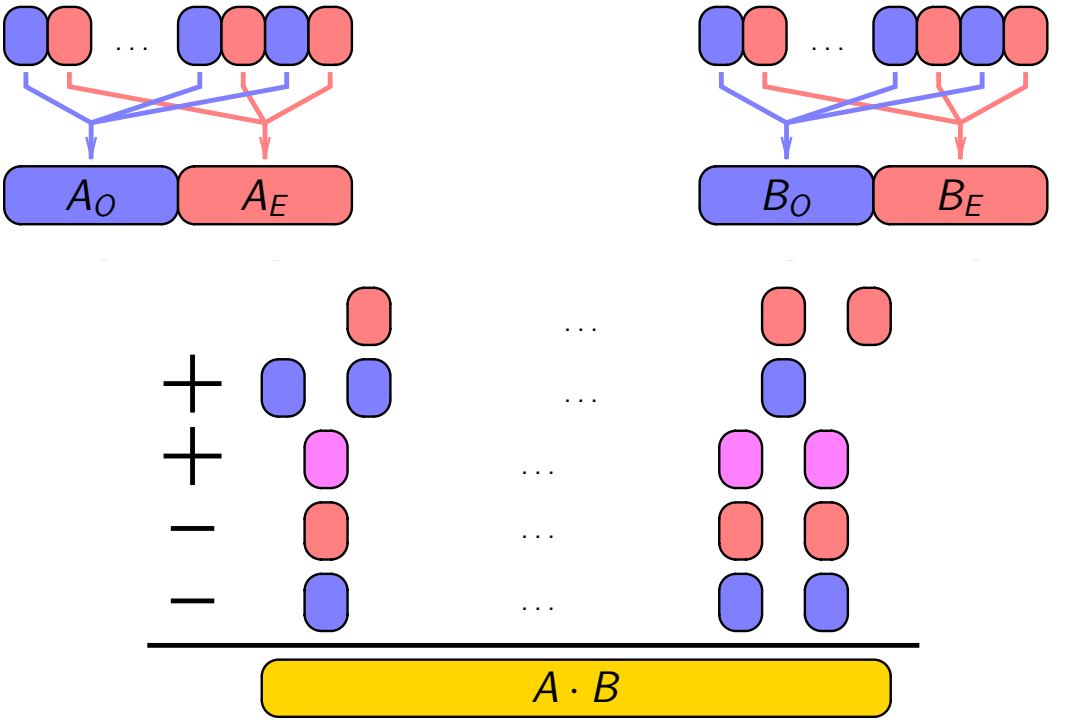
$$ab' + a'b = (a + a')(b + b') - ab - a'b'$$

$$(A_O B_O X^2 + A_E B_E) + X((A_O + A_E)(B_O + B_E) - A_O B_O - A_E B_E)$$

Odd–even split for Karatsuba multiplication



Odd–even split for Karatsuba multiplication



Some other subquadratic multiplication algorithms

- Karatsuba-like algorithms: detailed algorithm
  - original formula
  - 3-way Karatsuba (7 instead of 9 subproducts)
  - odd–even split
  - 3-way odd–even split-like

Some other subquadratic multiplication algorithms

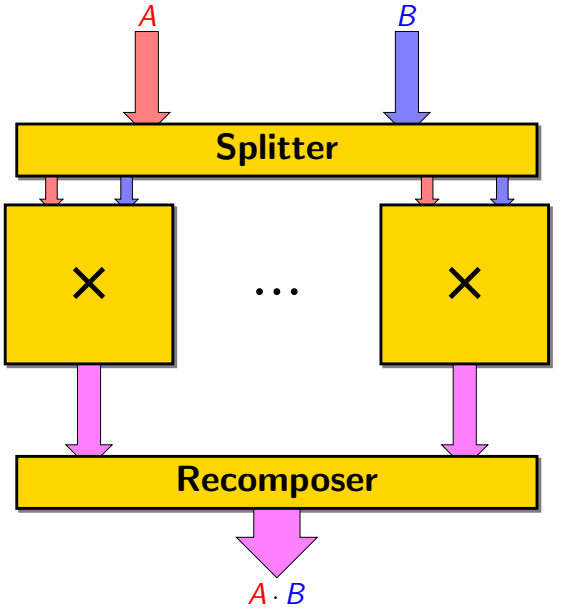
- Karatsuba-like algorithms: ‣ detailed algorithm
  - original formula
  - 3-way Karatsuba (7 instead of 9 subproducts)
  - odd–even split
  - 3-way odd–even split-like
- Toom-Cook algorithms:
  - evaluation–interpolation scheme
  - split operands in 3 or more parts
  - symmetric or asymmetric splitting
  - odd–even trick (work in progress)
- Montgomery's formulae
- ...

Some other subquadratic multiplication algorithms

- Karatsuba-like algorithms: ▸ detailed algorithm
  - original formula
  - 3-way Karatsuba (7 instead of 9 subproducts)
  - odd–even split
  - 3-way odd–even split-like
- Toom-Cook algorithms:
  - evaluation–interpolation scheme
  - split operands in 3 or more parts
  - symmetric or asymmetric splitting
  - odd–even trick (work in progress)
- Montgomery's formulae
- ...
- Select best method for each stage of recursion

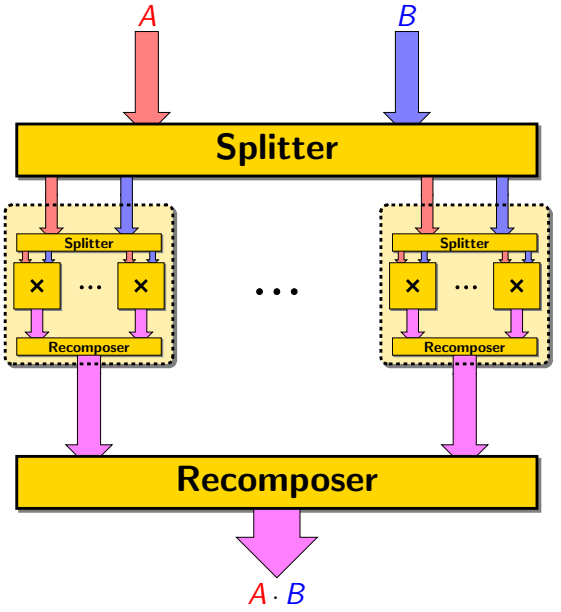
Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts



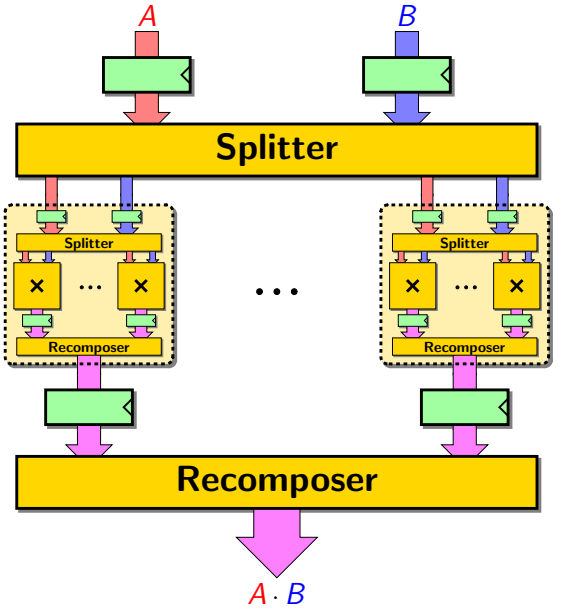
Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts
- Recursive scheme
  - eventually use different multiplication algorithms
  - end with the quadratic paper-and-pencil algorithm



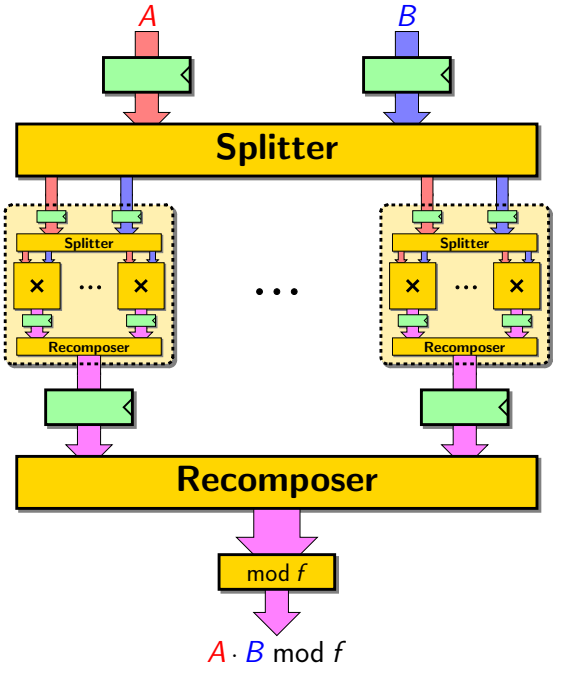
Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts
- Recursive scheme
  - eventually use different multiplication algorithms
  - end with the quadratic paper-and-pencil algorithm
- Pipelined
  - with the help of optional registers
  - cut the critical path
  - increase the frequency



Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts
- Recursive scheme
  - eventually use different multiplication algorithms
  - end with the quadratic paper-and-pencil algorithm
- Pipelined
  - with the help of optional registers
  - cut the critical path
  - increase the frequency
- Final reduction modulo f
  - small operator if f has low Hamming weight



Choice of the recursion and FPGA implementation

- Leading zeros problem
  - add them when the inputs are not perfectly splittable
  - increase the size of subproducts

Choice of the recursion and FPGA implementation

- Leading zeros problem
  - add them when the inputs are not perfectly splittable
  - increase the size of subproducts
  - correctly choose the recursion
  - use different kinds of multiplier for the different subproducts

Choice of the recursion and FPGA implementation

- Leading zeros problem
  - add them when the inputs are not perfectly splittable
  - increase the size of subproducts
  - correctly choose the recursion
  - use different kinds of multiplier for the different subproducts
- Form of addition trees
  - depends on characteristic
  - depends on FPGA technology
  - maximize LUTs utilization

Choice of the recursion and FPGA implementation

- Leading zeros problem
  - add them when the inputs are not perfectly splittable
  - increase the size of subproducts
  - correctly choose the recursion
  - use different kinds of multiplier for the different subproducts
- Form of addition trees
  - depends on characteristic
  - depends on FPGA technology
  - maximize LUTs utilization
- Why the odd–even trick does not always work
  - depends also on characteristic and FPGA
  - have to estimate the area in number of LUT not in number of additions

An example of multiplier over $\mathbb{F}_{3^{239}}$

- Polynomial basis

$$\mathbb{F}_{3^{239}} \cong \mathbb{F}_3[X]/(X^{239} - X^5 + 1)$$

An example of multiplier over $\mathbb{F}_{3^{239}}$

- Polynomial basis

$$\mathbb{F}_{3^{239}} \cong \mathbb{F}_3[X]/(X^{239} - X^5 + 1)$$

- Recursion choice

Polynomial size	Used algorithm
239	3-way Karatsuba with odd-even trick
80	2-way Karatsuba with odd-even trick
40	2-way Karatsuba with odd-even trick
20	2-way Karatsuba with odd-even trick
10	2-way Karatsuba with odd-even trick
5	2-way Karatsuba with odd-even trick
3	quadratic multiplication

An example of multiplier over $\mathbb{F}_{3^{239}}$

- Polynomial basis

$$\mathbb{F}_{3^{239}} \cong \mathbb{F}_3[X]/(X^{239} - X^5 + 1)$$

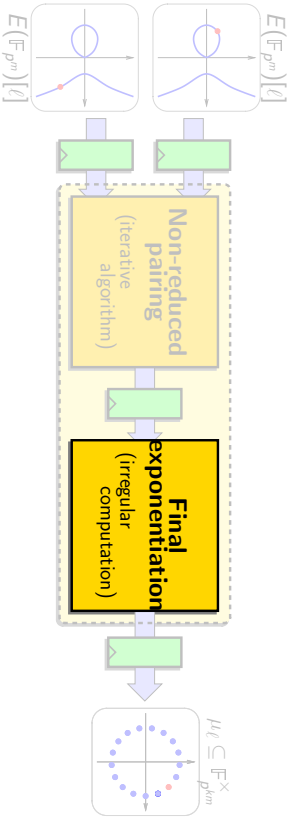
- Recursion choice

Polynomial size	Used algorithm
239	3-way Karatsuba with odd-even trick
80	2-way Karatsuba with odd-even trick
40	2-way Karatsuba with odd-even trick
20	2-way Karatsuba with odd-even trick
10	2-way Karatsuba with odd-even trick
5	2-way Karatsuba with odd-even trick
3	quadratic multiplication

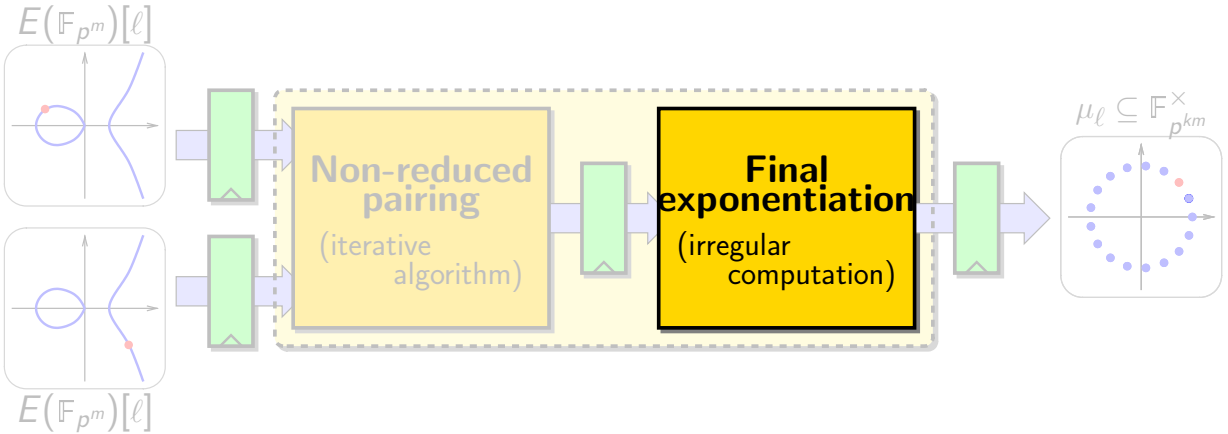
- Post-place-and-route estimation for Xilinx Virtex-II Pro
  - 49984 slices
  - 200 MHz

Outline of the talk

- Context
- Reduced Tate pairing
- Non-reduced Tate pairing
- Fully parallel Karatsuba multiplier
- Final Exponentiation
- Results & Conclusion
- Appendix



Final exponentiation



- Design rationale:
  - as small as possible
  - at least as fast as the computation of the non-reduced pairing

Coprocessor for the final exponentiation (char. 3)

- Highly sequential computation
- Very heterogeneous

Coprocessor for the final exponentiation (char. 3)

- Highly sequential computation
- Very heterogeneous

$\Rightarrow$ general-purpose finite-field arithmetic processor

Coprocessor for the final exponentiation (char. 3)

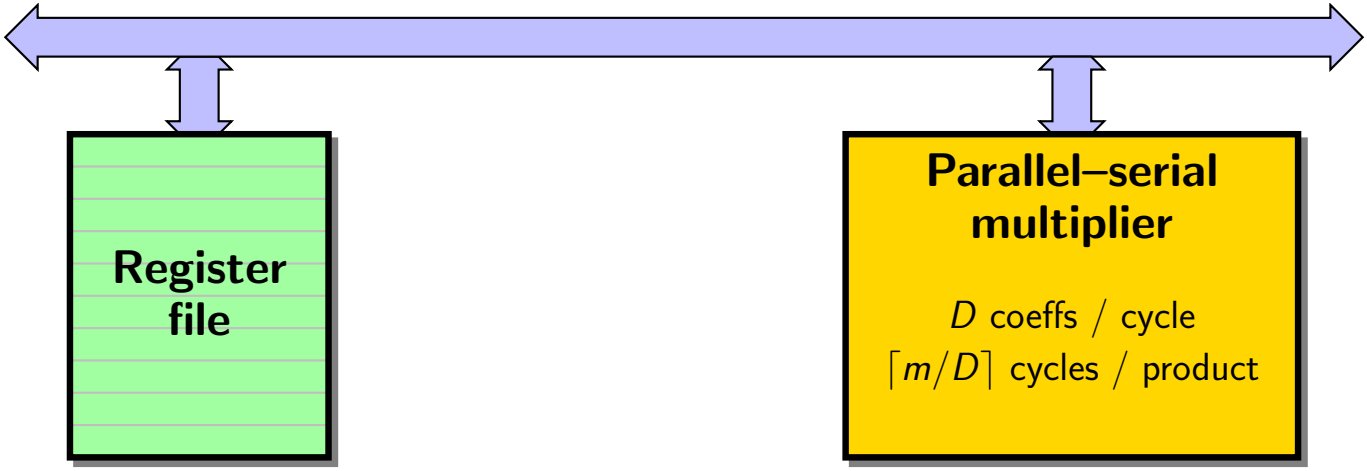
- Highly sequential computation
- Very heterogeneous

$\Rightarrow$ general-purpose finite-field arithmetic processor

Coprocessor for the final exponentiation (char. 3)

- Highly sequential computation
- Very heterogeneous

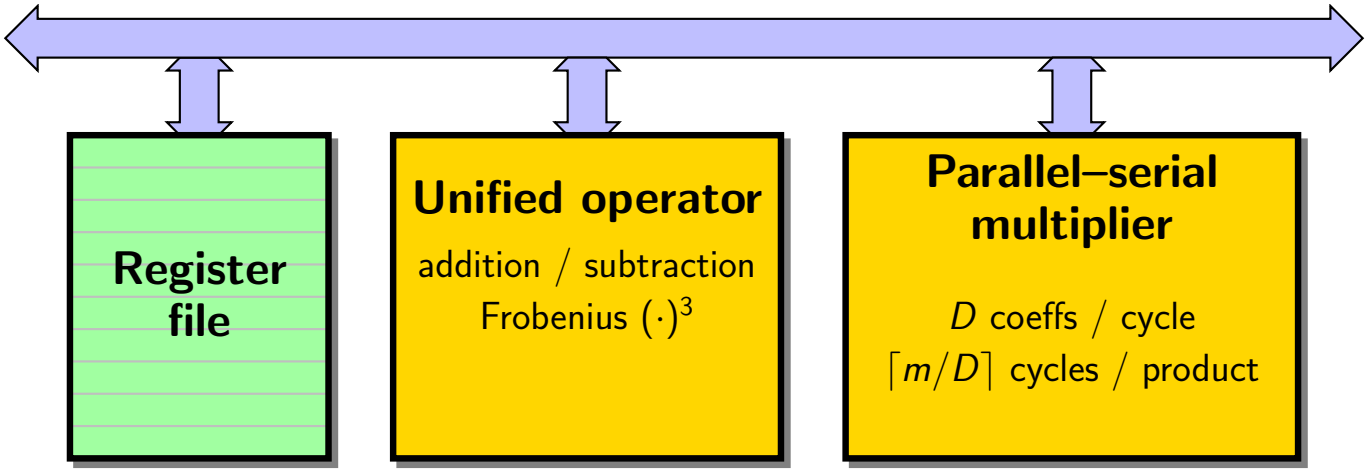
$\Rightarrow$ general-purpose finite-field arithmetic processor



Coprocessor for the final exponentiation (char. 3)

- Highly sequential computation
- Very heterogeneous

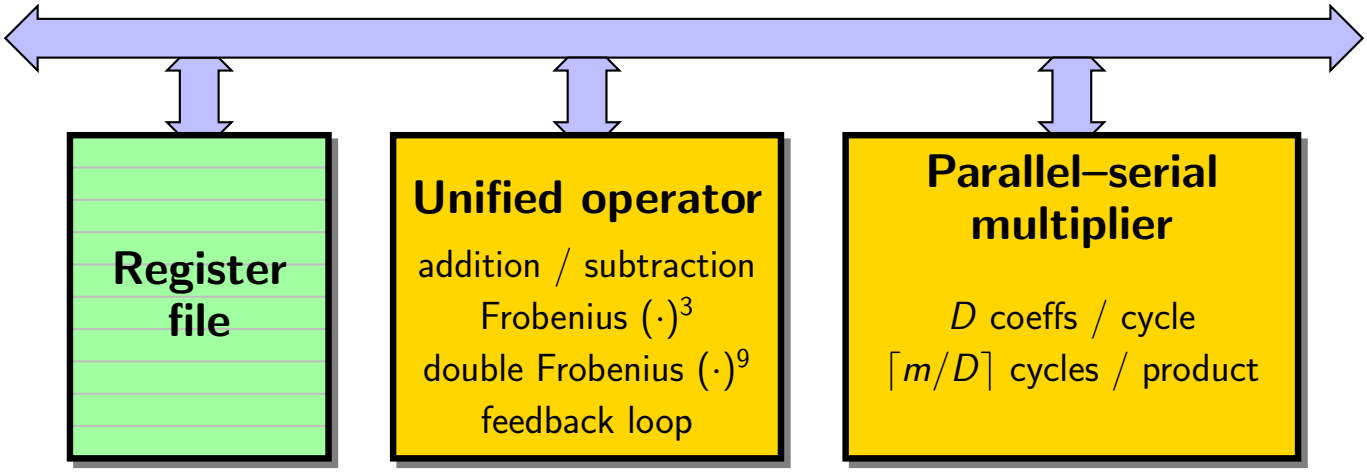
$\Rightarrow$ general-purpose finite-field arithmetic processor



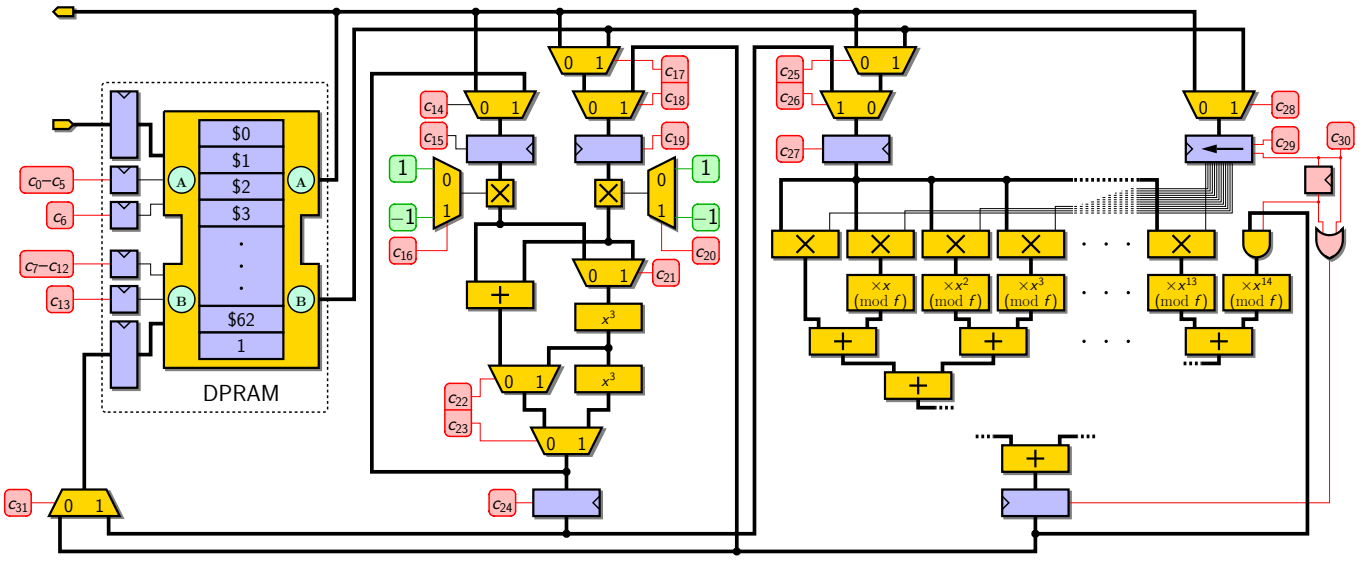
Coprocessor for the final exponentiation (char. 3)

- Highly sequential computation
- Very heterogeneous

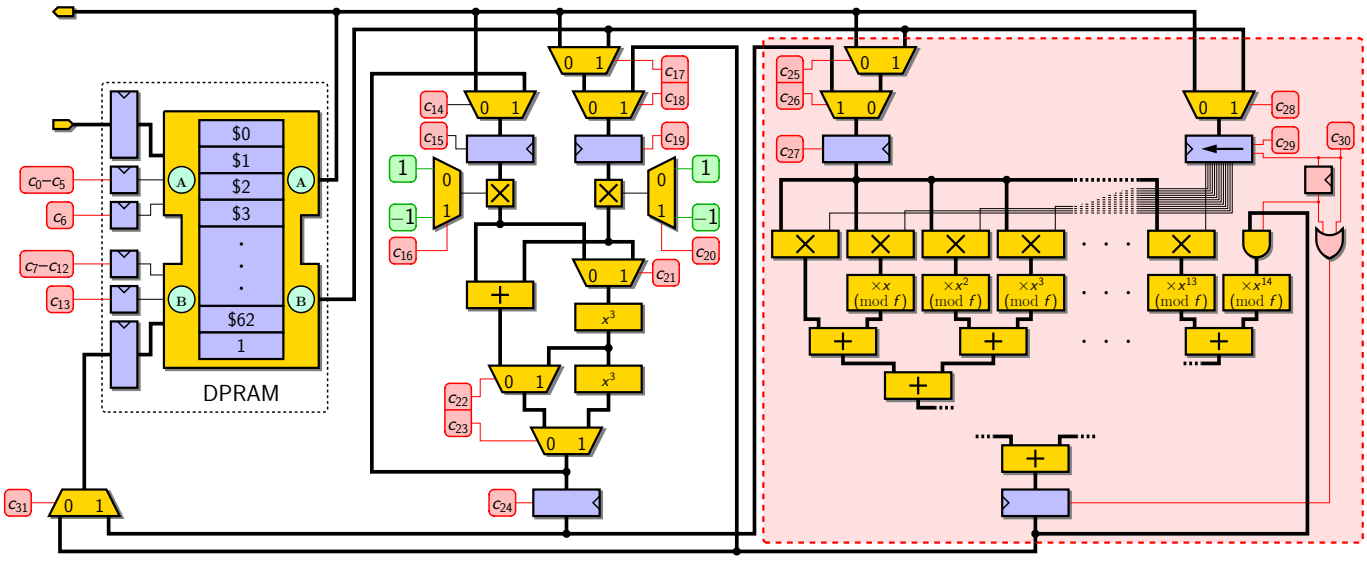
$\Rightarrow$ general-purpose finite-field arithmetic processor



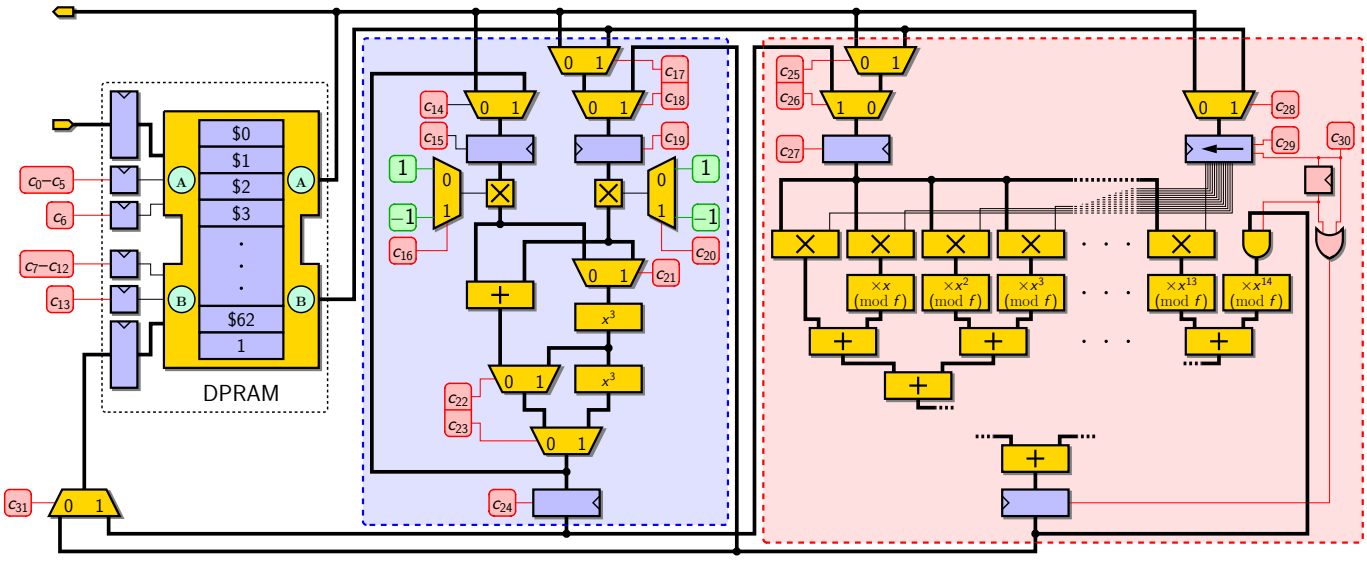
Detailed architecture of the coprocessor (char. 3)



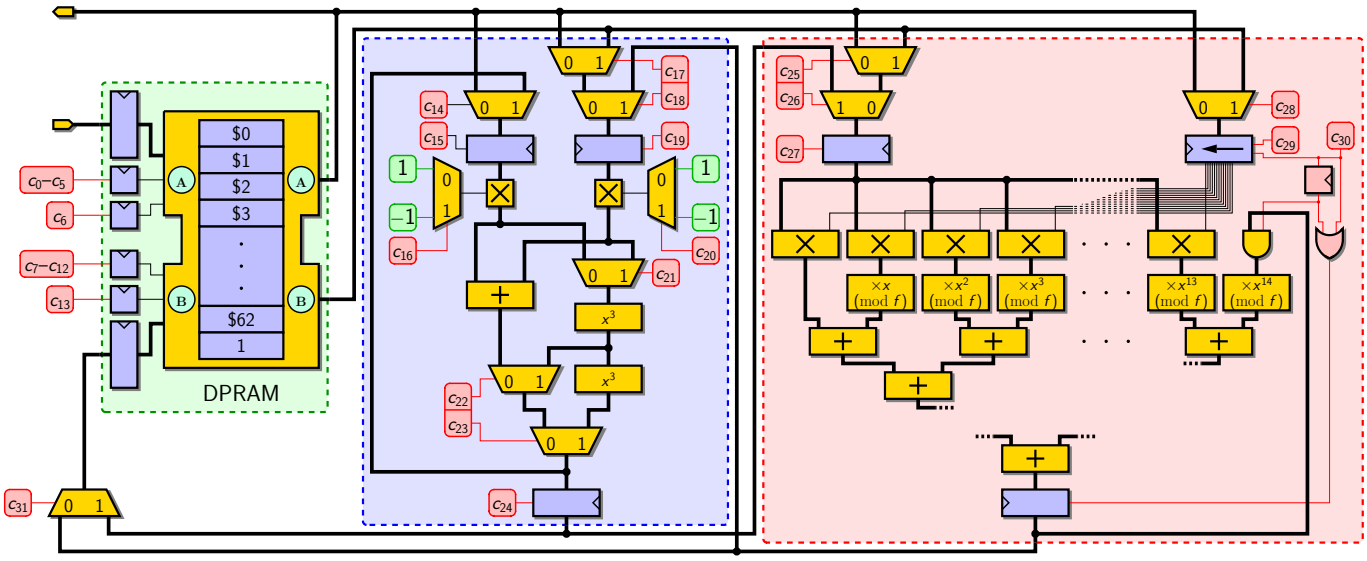
Detailed architecture of the coprocessor (char. 3)



Detailed architecture of the coprocessor (char. 3)



Detailed architecture of the coprocessor (char. 3)



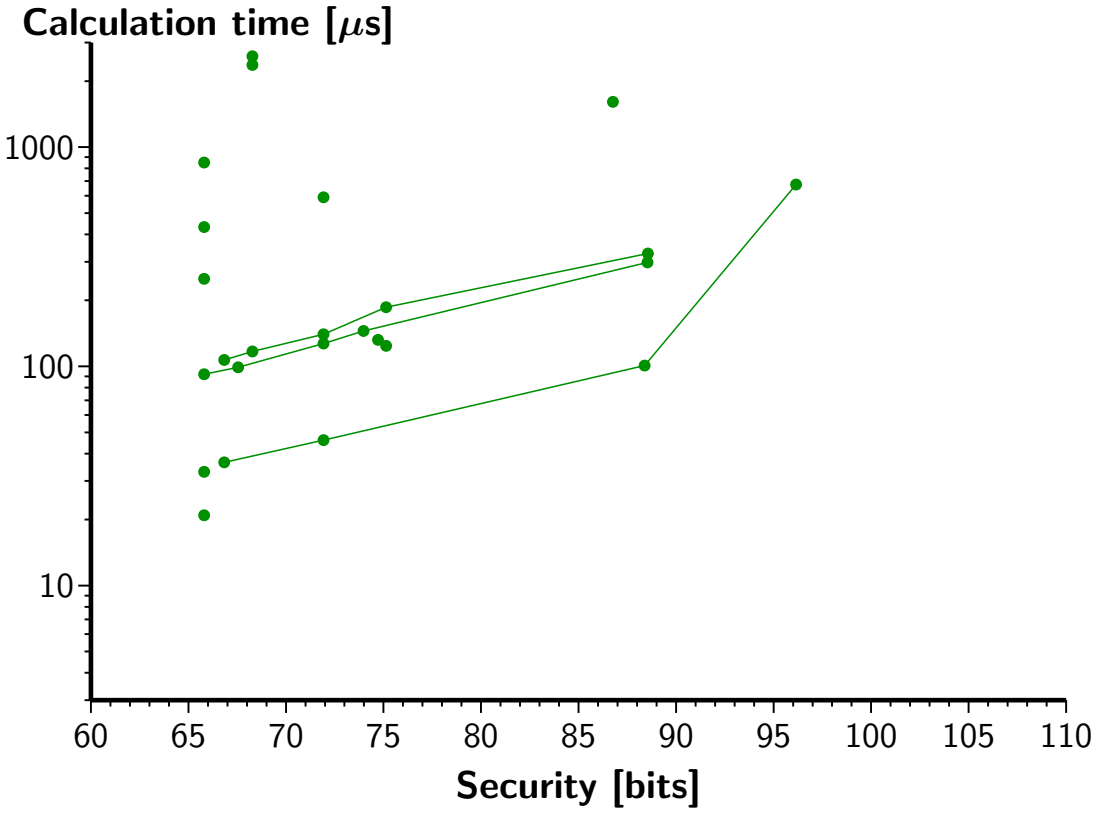
Outline of the talk

- Context
- Reduced Tate pairing
- Non-reduced Tate pairing
- Fully parallel Karatsuba multiplier
- Final Exponentiation
- Results & Conclusion
- Appendix

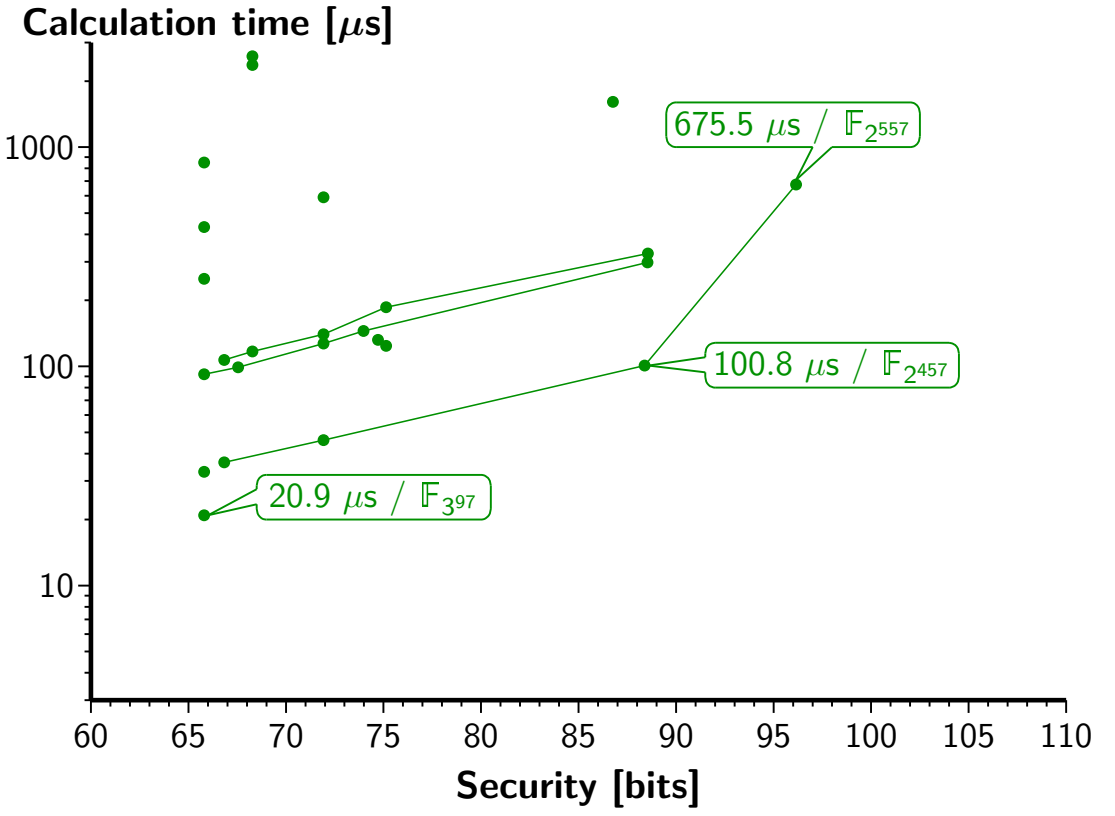
Experimental setup

- Full Tate pairing computation:
  - non-reduced pairing and
  - final exponentiation
- Prototyped on Xilinx Virtex-II Pro and Virtex-4 LX FPGAs
- Post-place-and-route timing and area estimations

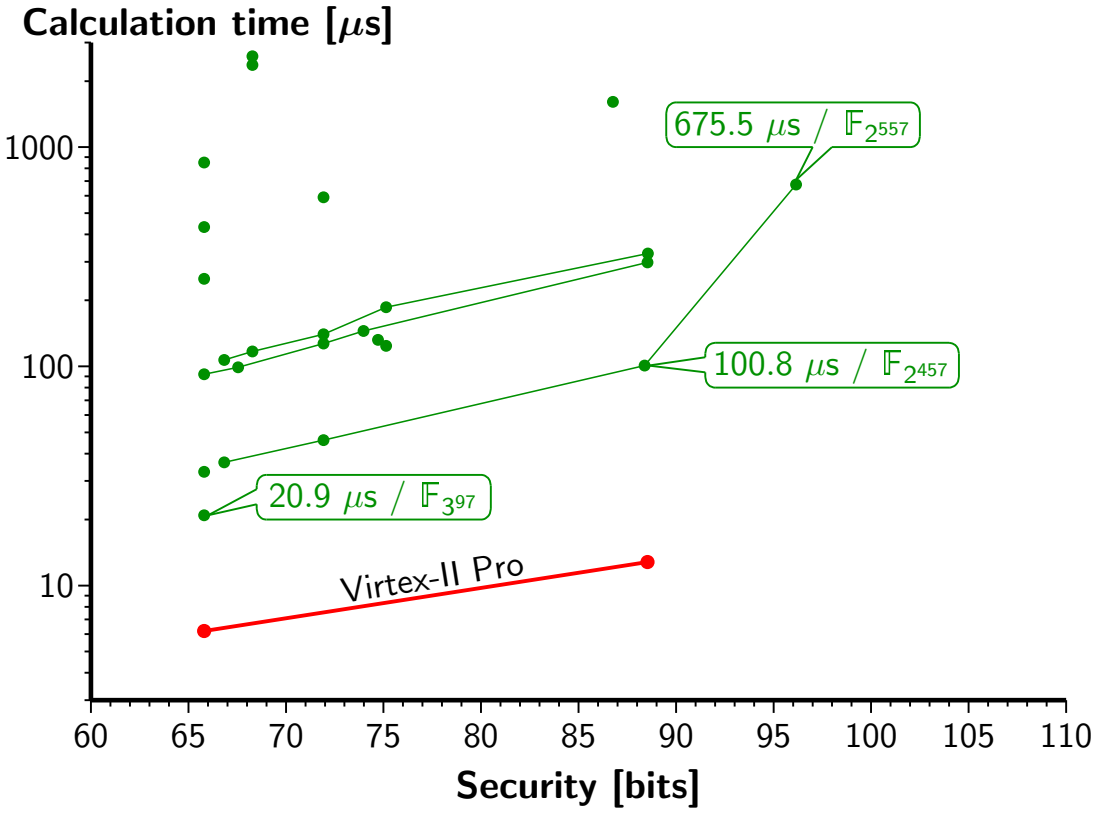
Calculation time



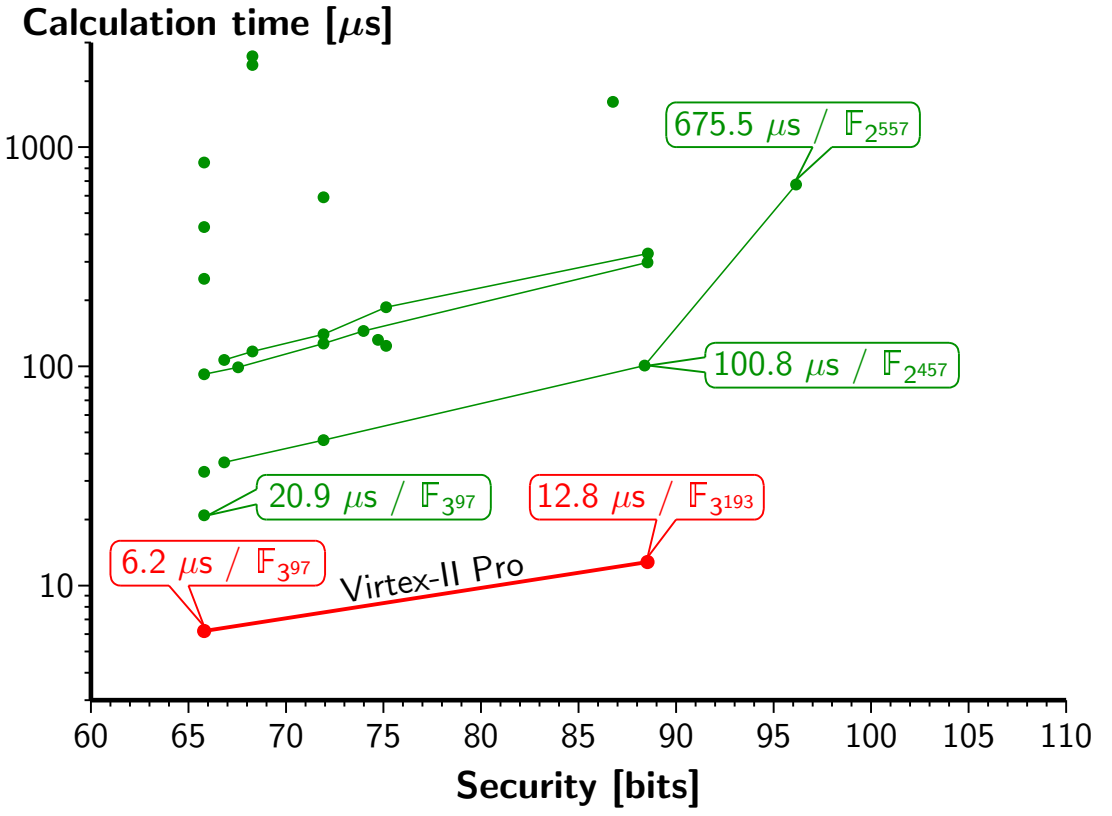
Calculation time



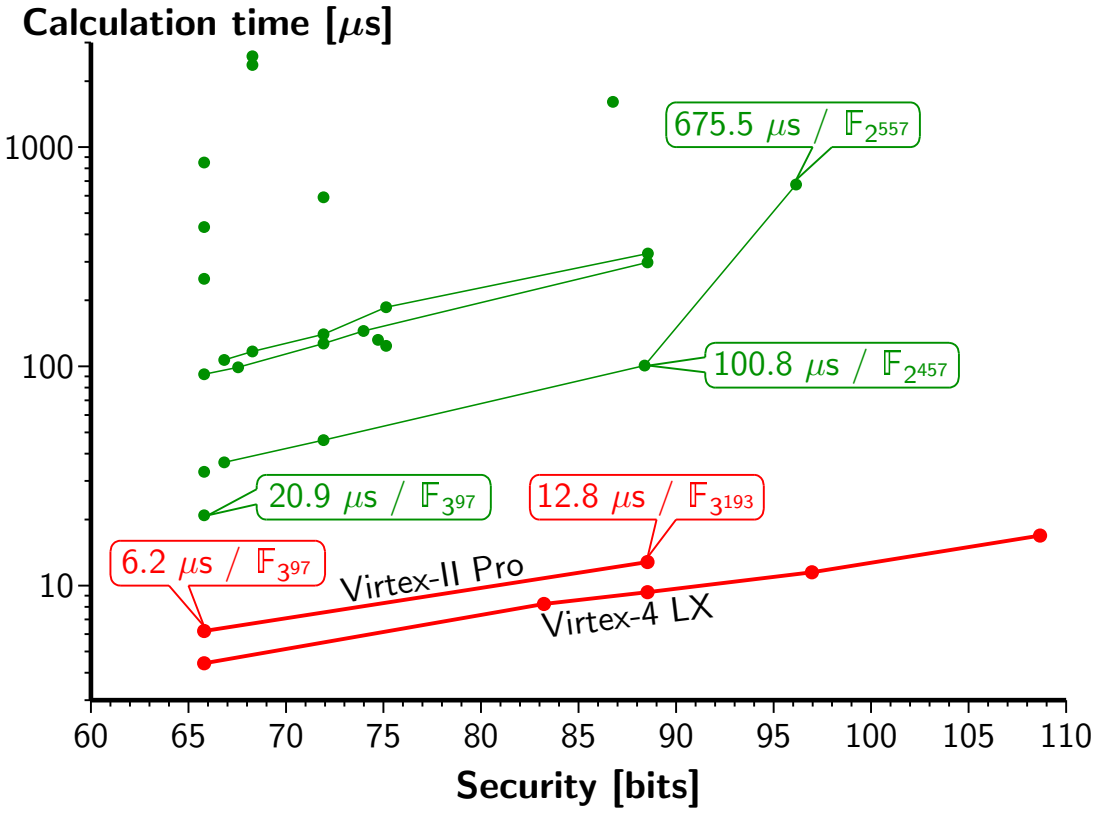
Calculation time



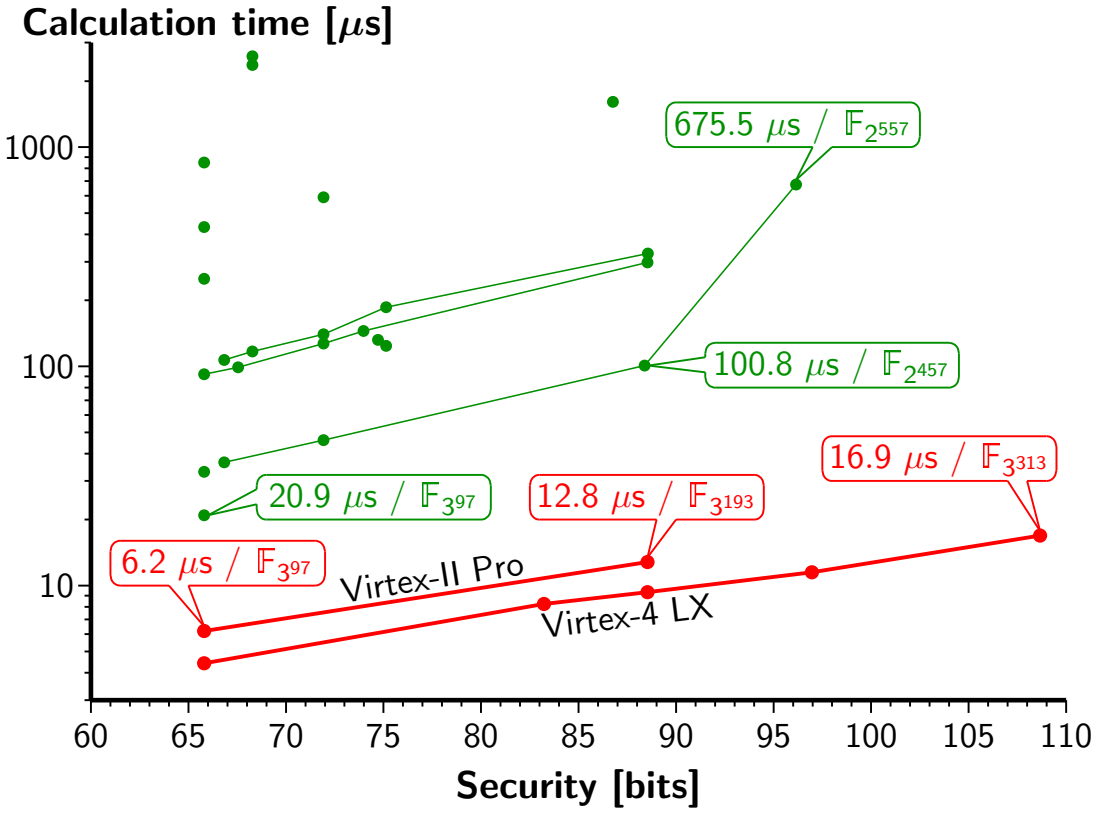
Calculation time



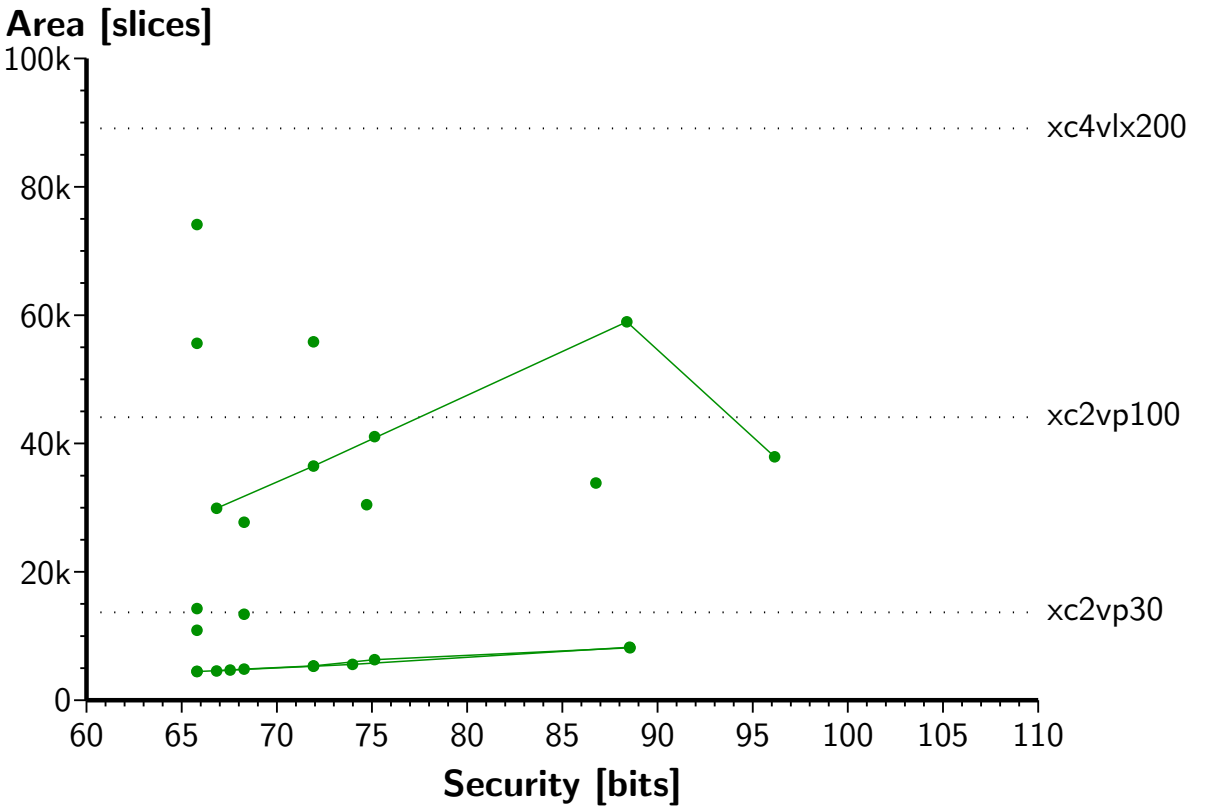
Calculation time



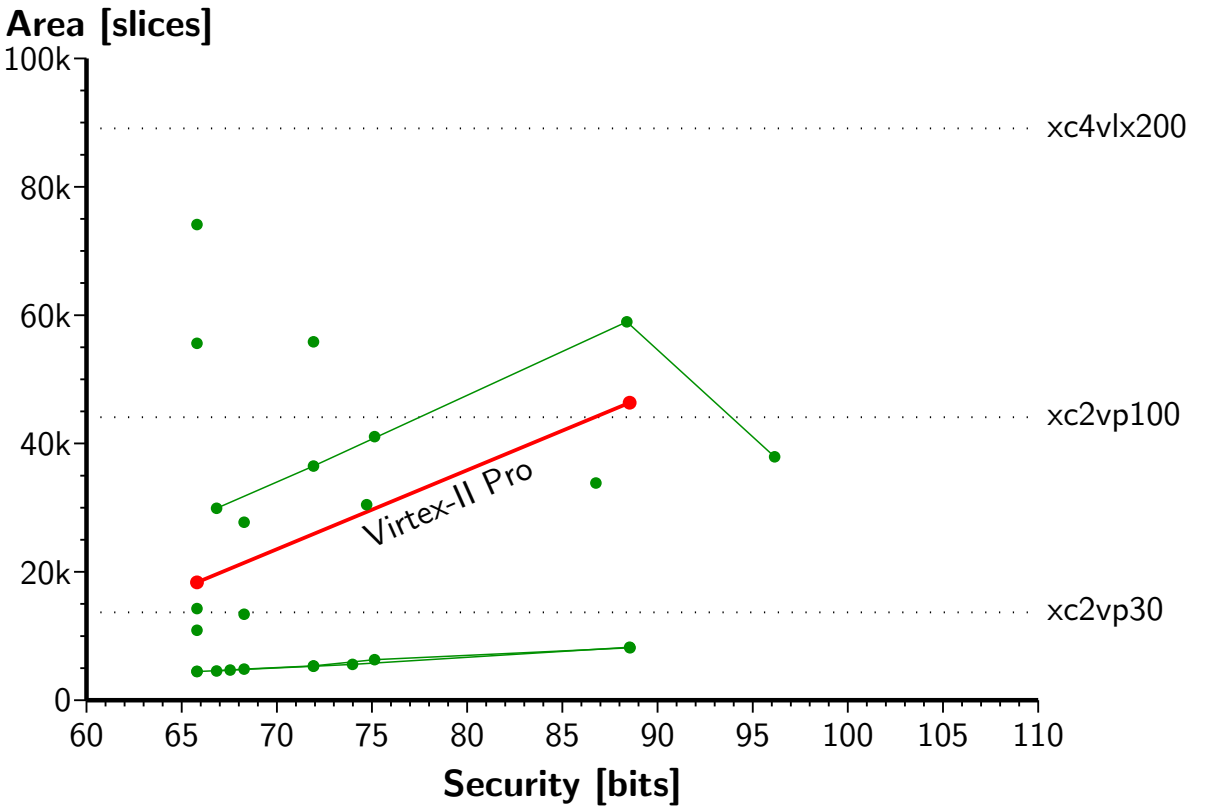
Calculation time



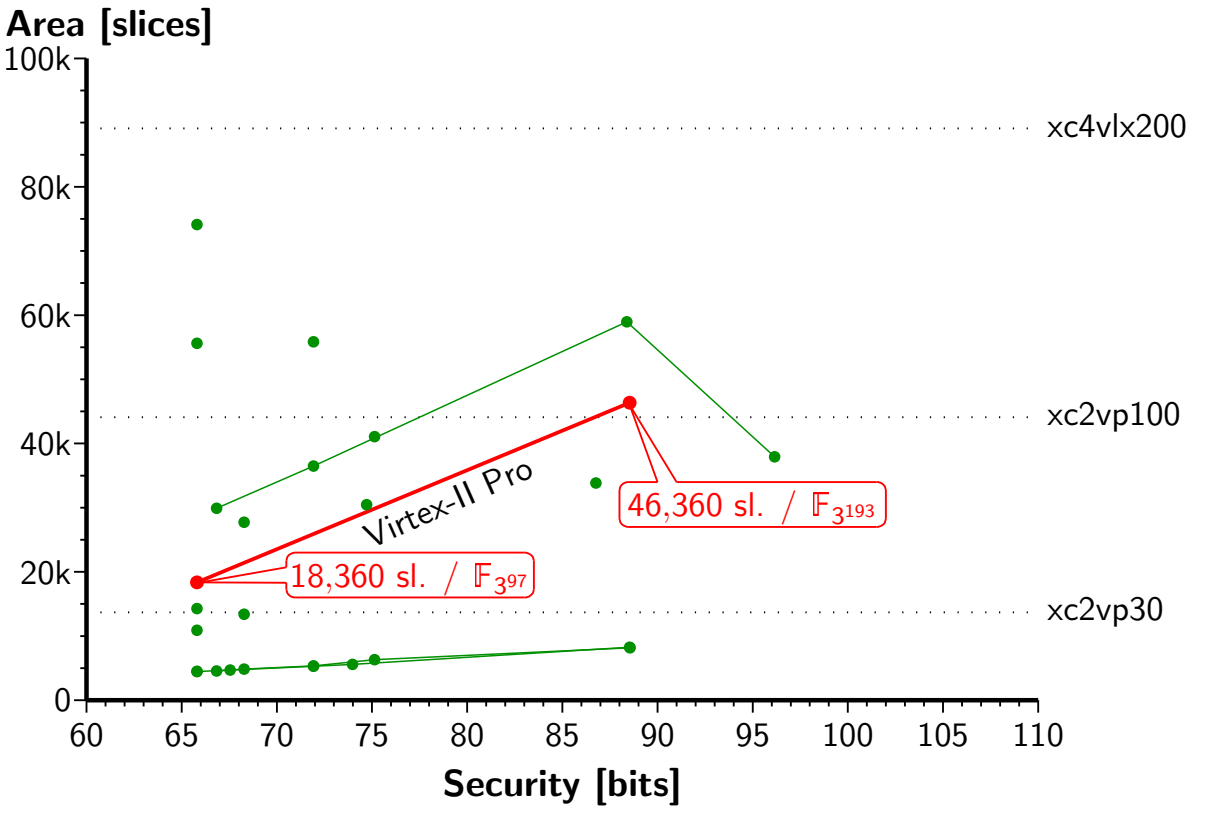
Area


Area [slices]
100k
80k
60k
40k
20k
0
xc4vlx200
xc2vp100
xc2vp30
60
65
70
75
80
85
90
95
100
105
110
Security [bits]

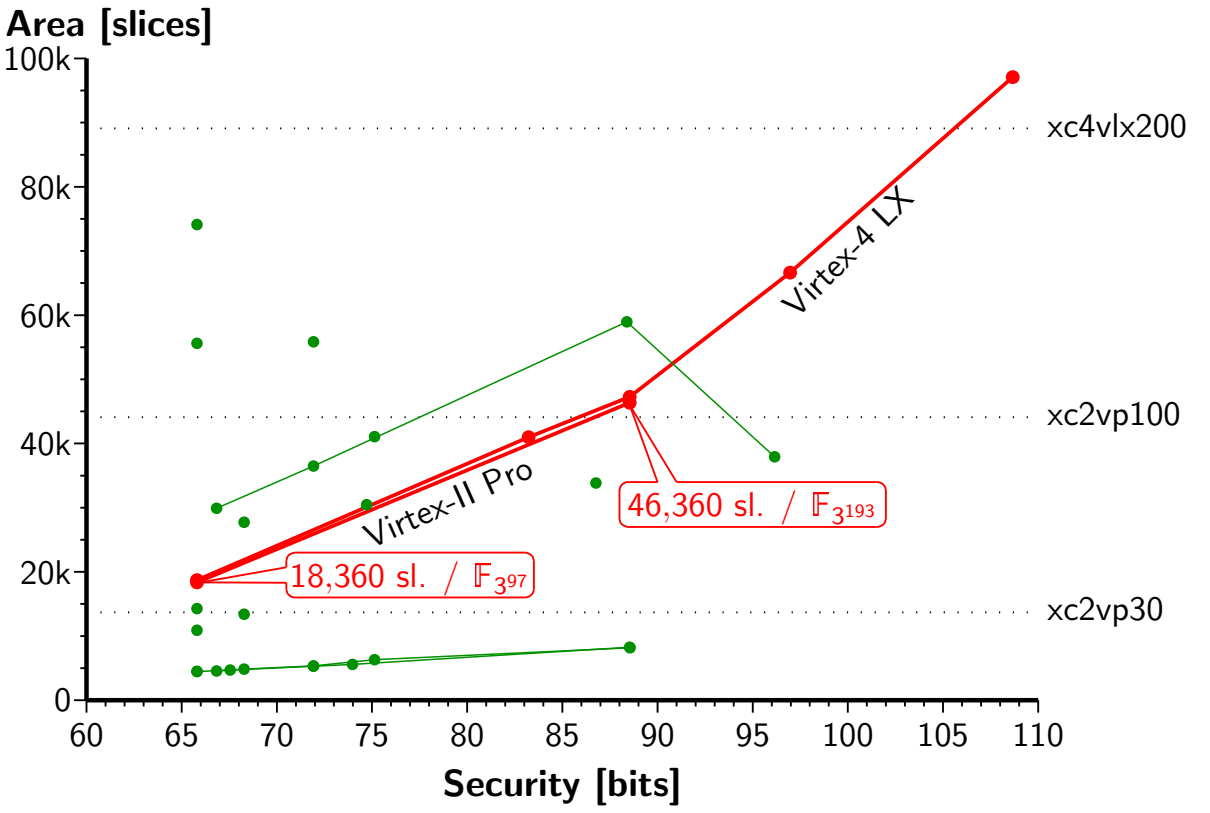
Area


Area [slices]
100k
80k
60k
40k
20k
0
xc4vlx200
xc2vp100
xc2vp30
Virtex-II Pro
60
65
70
75
80
85
90
95
100
105
110
Security [bits]

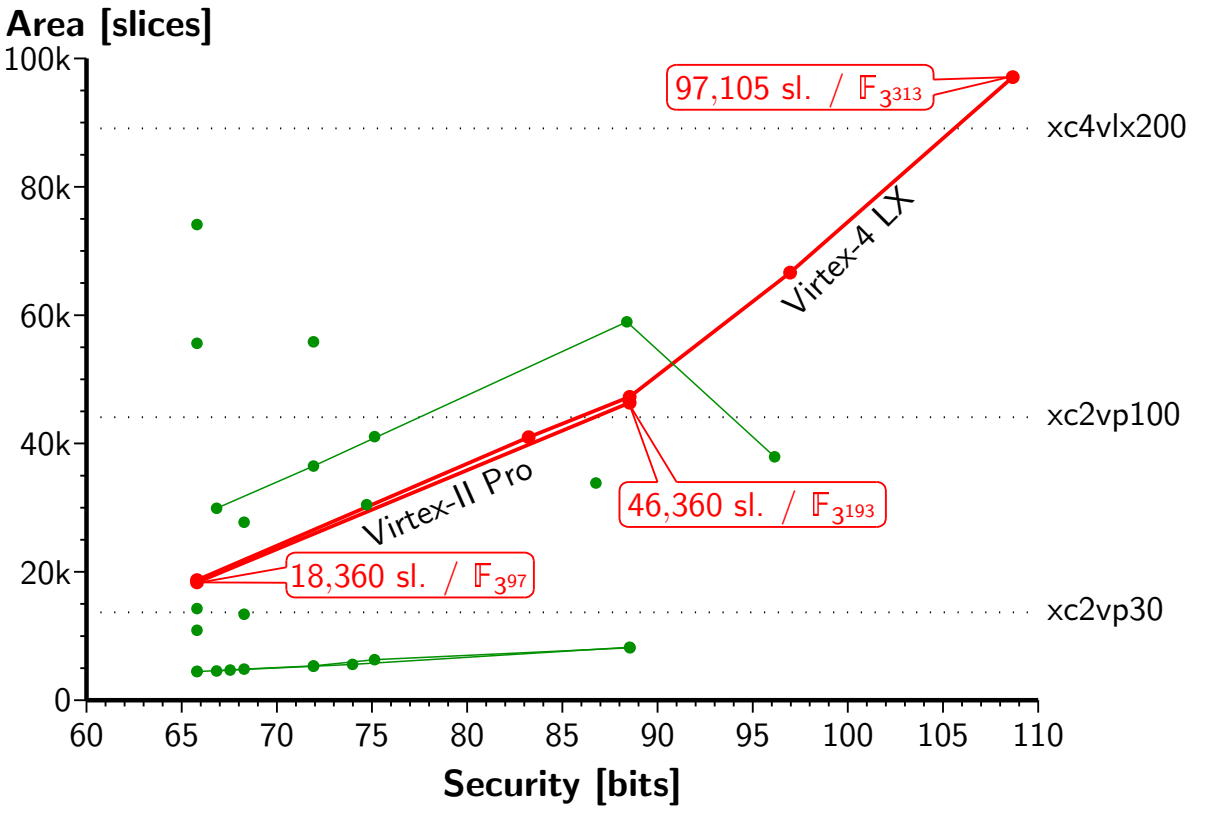
Area


Area [slices]
100k
80k
60k
40k
20k
0
xc4vlx200
xc2vp100
xc2vp30
Virtex-II Pro
46,360 sl. / $\mathbb{F}_{3^{193}}$
18,360 sl. / $\mathbb{F}_{3^{97}}$
60
65
70
75
80
85
90
95
100
105
110
Security [bits]

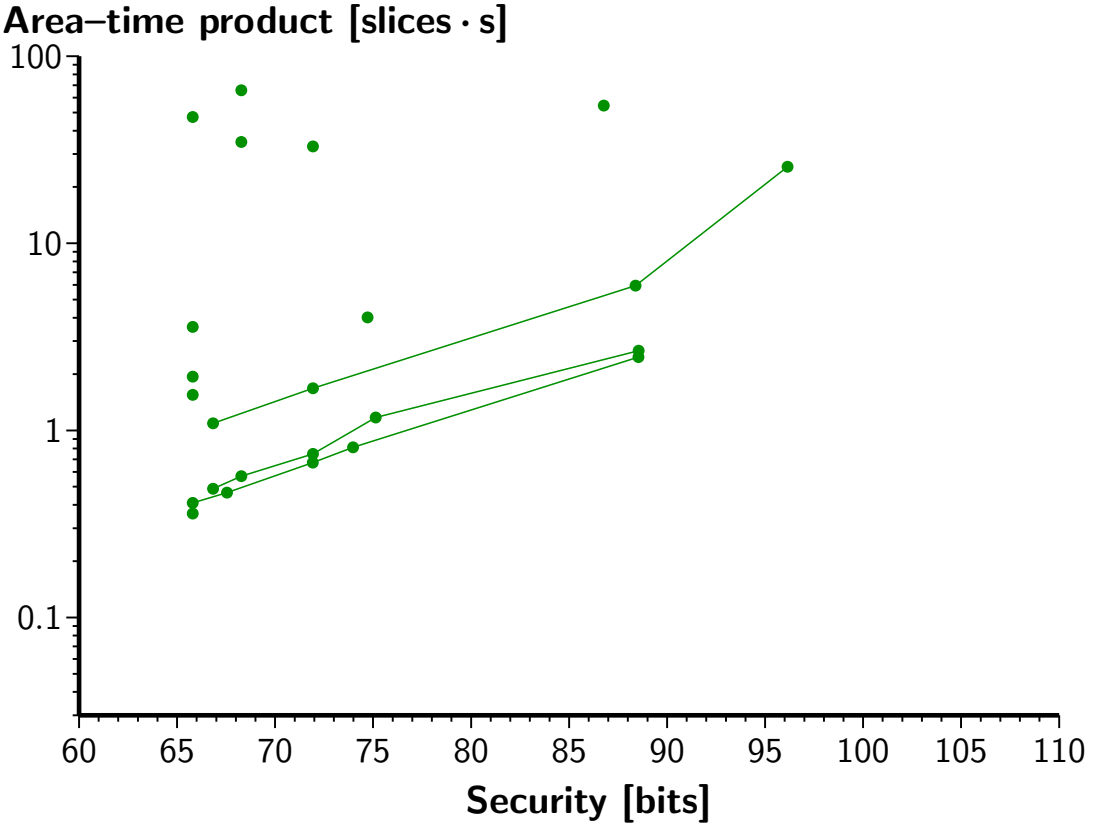
Area


Area [slices]
100k
80k
60k
40k
20k
0
60
65
70
75
80
85
90
95
100
105
110
Security [bits]
xc4vlx200
xc2vp100
xc2vp30
Virtex-4 LX
Virtex-II Pro
46,360 sl. / $\mathbb{F}_{3^{193}}$
18,360 sl. / $\mathbb{F}_{3^{97}}$

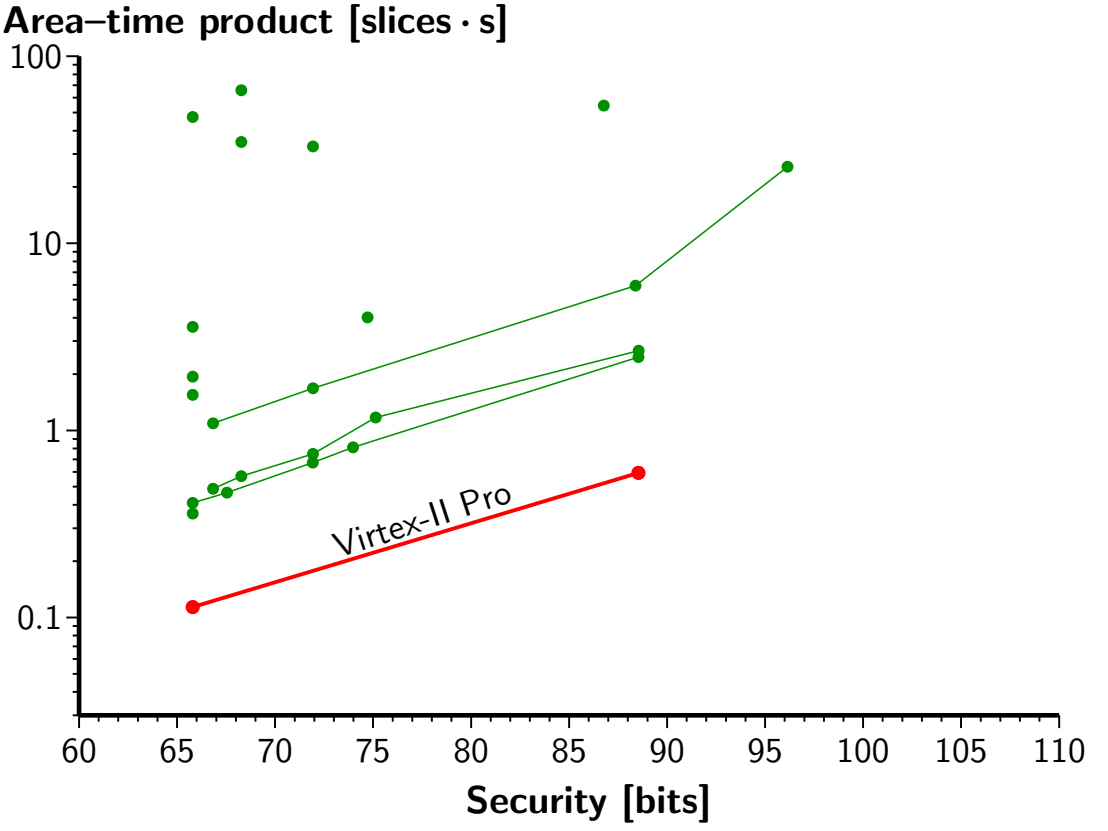
Area


Area [slices]
100k
80k
60k
40k
20k
0
60
65
70
75
80
85
90
95
100
105
110
Security [bits]
97,105 sl. / $\mathbb{F}_{3^{313}}$
46,360 sl. / $\mathbb{F}_{3^{193}}$
18,360 sl. / $\mathbb{F}_{3^{97}}$
Virtex-4 LX
Virtex-II Pro
xc4vlx200
xc2vp100
xc2vp30

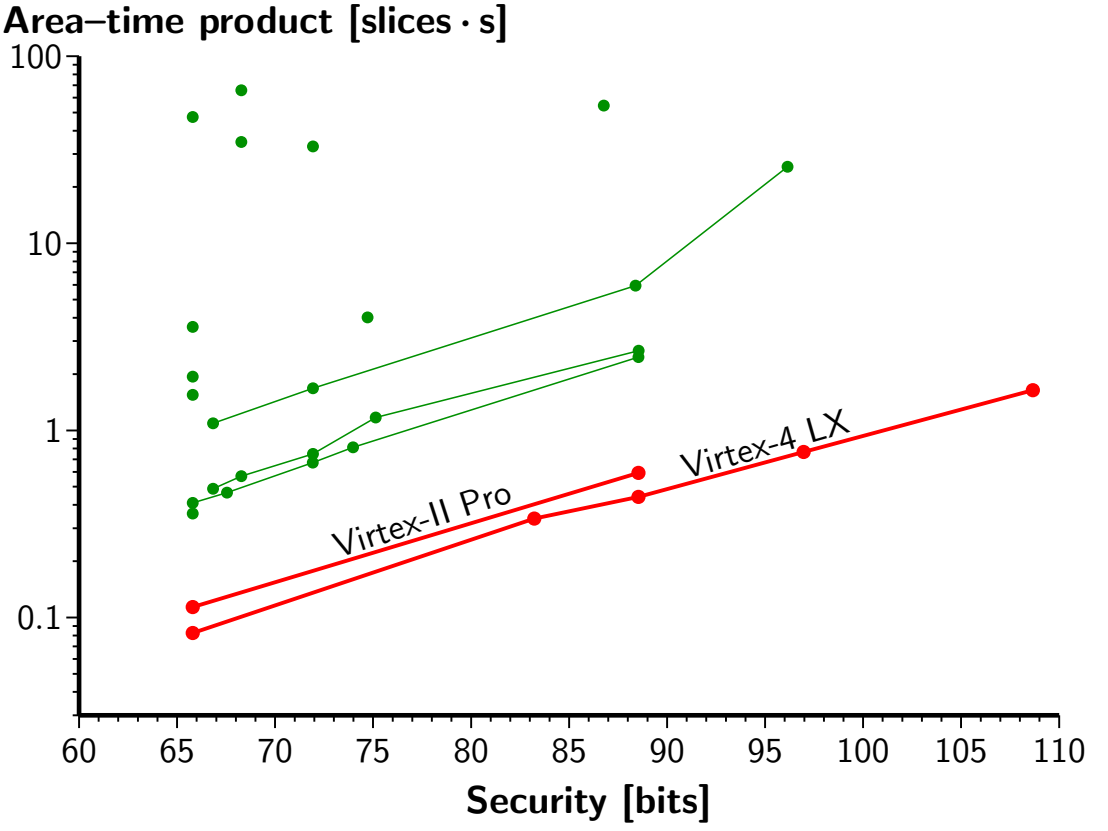
Area–Time product



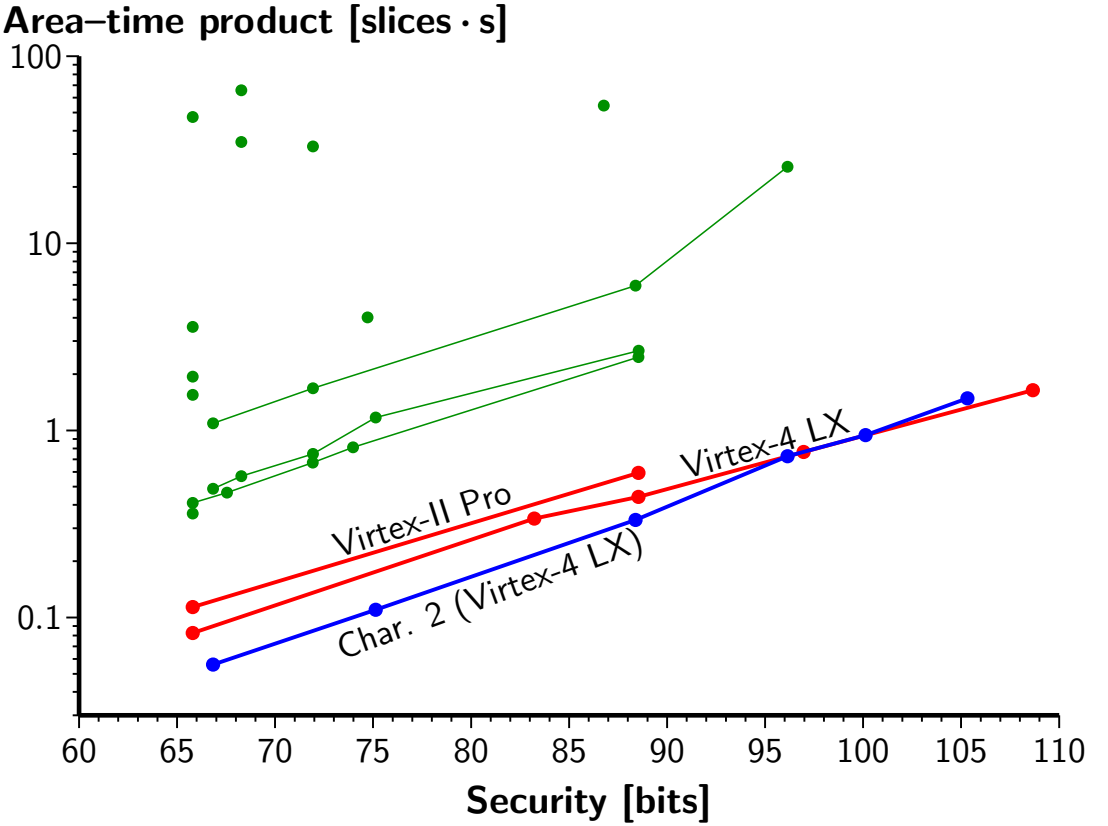
Area–Time product



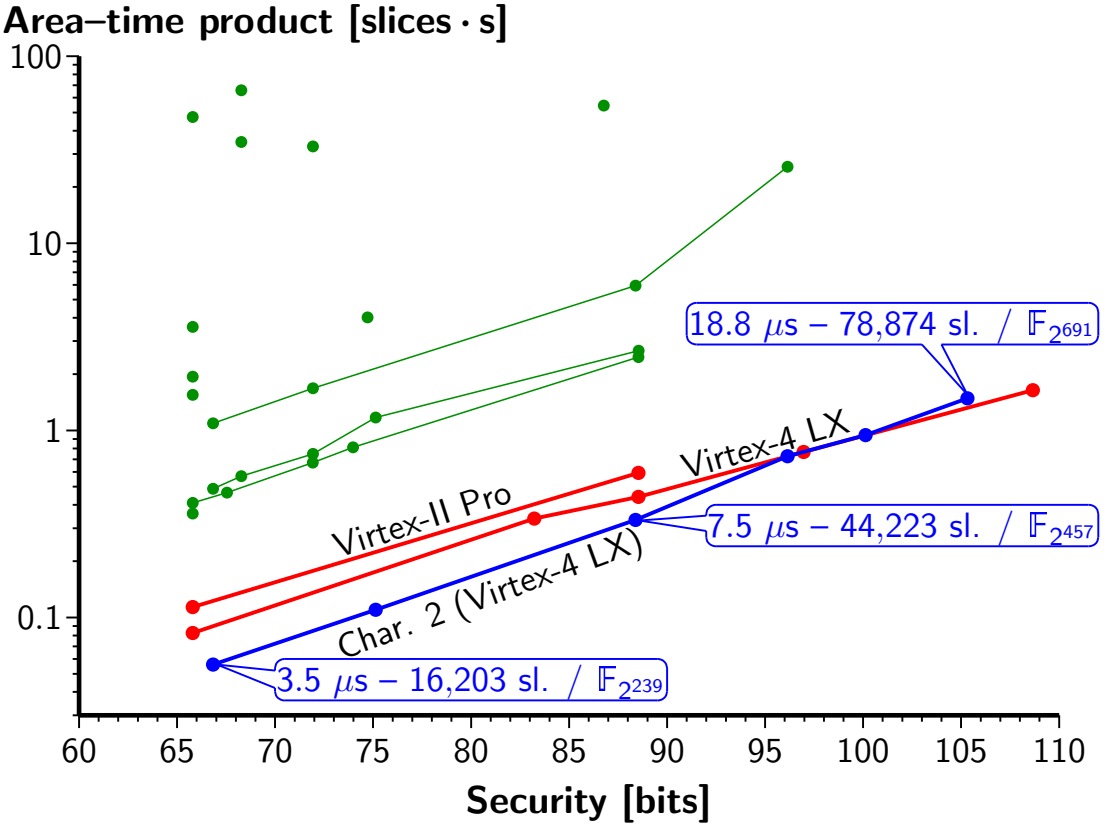
Area–Time product



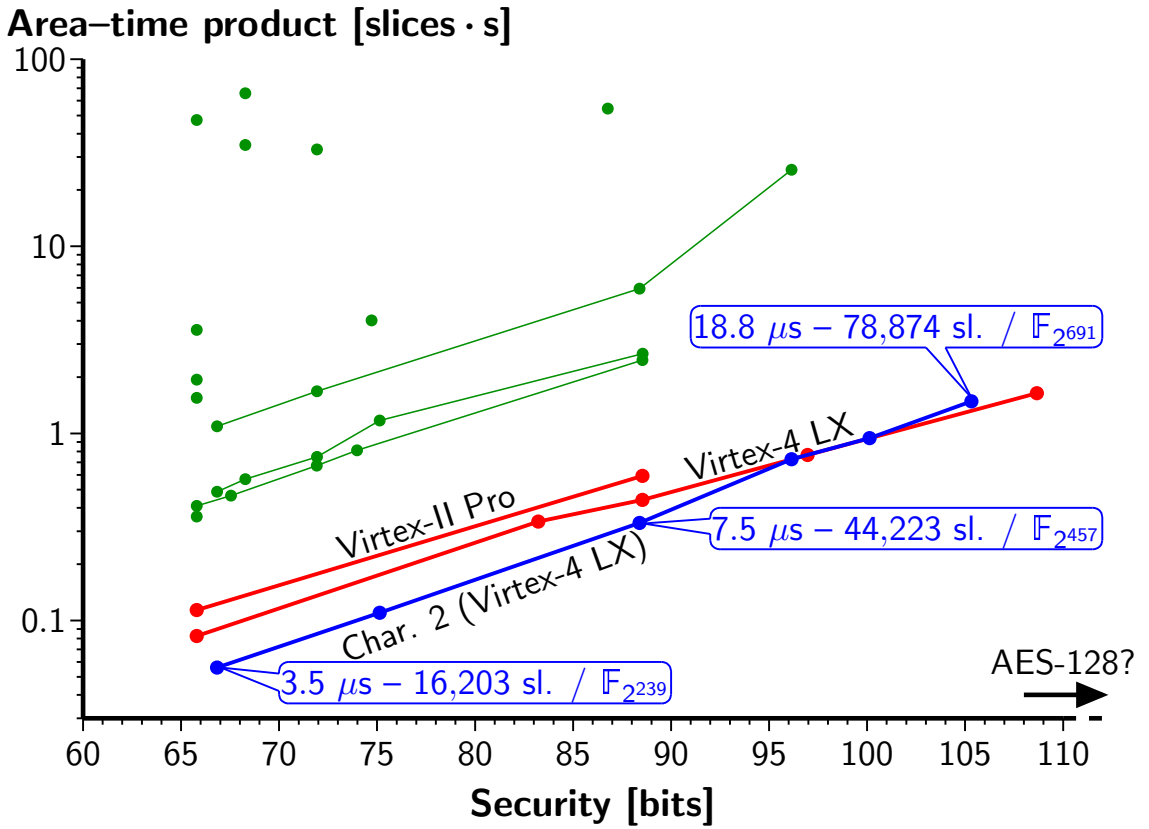
Area–Time product



Area–Time product



Area–Time product



Conclusion

- A new architecture for pairing computation
  - two specialized coprocessors
  - bet on parallelizing multiplier
  - based on Karatsuba multiplication scheme
  - importance of architecture–algorithm co-design
  - careful bubble-free scheduling of Miller's loop

Conclusion

- A new architecture for pairing computation
  - two specialized coprocessors
  - bet on parallelizing multiplier
  - based on Karatsuba multiplication scheme
  - importance of architecture–algorithm co-design
  - careful bubble-free scheduling of Miller's loop
- High-performance accelerator
  - the fastest coprocessor (17 μs for 109 bits of security)
  - the best area–time trade-off
  - scales to higher security levels

Future work

- Fully parallel multipliers
  - tune finely Karatsuba algorithm and multiplier architecture
  - try other algorithms: Toom–Cook, Montgomery's formulae
  - try less parallel multipliers: slower but smaller

Future work

- Fully parallel multipliers
  - tune finely Karatsuba algorithm and multiplier architecture
  - try other algorithms: Toom–Cook, Montgomery's formulae
  - try less parallel multipliers: slower but smaller
- Final-exponentiation coprocessor
  - full-featured finite-field processor
  - compute the full pairing with it (promising first experimental results)

Future work

- Fully parallel multipliers
  - tune finely Karatsuba algorithm and multiplier architecture
  - try other algorithms: Toom–Cook, Montgomery's formulae
  - try less parallel multipliers: slower but smaller
- Final-exponentiation coprocessor
  - full-featured finite-field processor
  - compute the full pairing with it (promising first experimental results)
- Toward AES-128 security level
  - explore supersingular pairing over $\mathbb{F}_{2^{mm'}}$ and $\mathbb{F}_{3^{mm'}}$ (work in progress)
  - genus-2 supersingular curves in characteristic 2 (work in progress)
  - Barreto–Naehrig curves

Thank you for your attention

Questions?

Outline of the talk

- Context
- Reduced Tate pairing
- Non-reduced Tate pairing
- Fully parallel Karatsuba multiplier
- Final Exponentiation
- Results & Conclusion
- Appendix

Detailed Karatsuba algorithms

Algorithm	Splitting	Subproducts	Recomposition
2-way split	$A \to A_L + X^{\lceil m/2 \rceil} A_H$ $B \to B_L + X^{\lceil m/2 \rceil} B_H$ $A_M \leftarrow A_H + A_L$ $B_M \leftarrow B_H + B_L$	$p_H \leftarrow A_H * B_H$ $p_M \leftarrow A_M * B_M$ $p_L \leftarrow A_L * B_L$	$S \leftarrow p_H X^{2\lceil m/2 \rceil}$ $+ (p_M - p_H - p_L) X^{\lceil m/2 \rceil}$ $+ p_L$
3-way split	$A \to A_0 + X^{\lceil m/3 \rceil} A_1 + X^{2\lceil m/3 \rceil} A_2$ $B \to B_0 + X^{\lceil m/3 \rceil} B_1 + X^{2\lceil m/3 \rceil} B_2$ $A_{S_0} \leftarrow A_1 + A_2$ $A_{S_1} \leftarrow A_0 + A_2$ $A_{S_2} \leftarrow A_1 + A_0$ $B_{S_0} \leftarrow B_1 + B_2$ $B_{S_1} \leftarrow B_0 + B_2$ $B_{S_2} \leftarrow B_1 + B_0$	$p_0 \leftarrow A_0 * B_0$ $p_1 \leftarrow A_1 * B_1$ $p_2 \leftarrow A_2 * B_2$ $p'_0 \leftarrow A_{S_0} * B_{S_0}$ $p'_1 \leftarrow A_{S_1} * B_{S_1}$ $p'_2 \leftarrow A_{S_2} * B_{S_2}$	$S \leftarrow p_2 X^{4\lceil m/3 \rceil}$ $+ (p'_0 - p_1 - p_2) X^{3\lceil m/3 \rceil}$ $+ (p'_1 - p_0 + p_1 - p_2) X^{2\lceil m/3 \rceil}$ $+ (p'_2 - p_1 - p_0) X^{\lceil m/3 \rceil}$ $+ p_0$

Detailed odd–even split Karatsuba algorithm

Algorithm	Splitting	Subproducts	Recomposition
2-way split	$A \to A_E(X^2) + XA_O(X^2)$ $B \to B_E(X^2) + XB_O(X^2)$ $A_M \leftarrow A_O + A_E$ $B_M \leftarrow B_O + B_E$	$p_O \leftarrow A_O * B_O$ $p_M \leftarrow A_M * B_M$ $p_E \leftarrow A_E * B_E$	$S \leftarrow (p_E + Xp_O)(X^2)$ $+ X(p_M - p_E - p_O)(X^2)$
3-way split	$A \to A_0(X^3) + XA_1(X^3) + X^2A_2(X^3)$ $B \to B_0(X^3) + XB_1(X^3) + X^2B_2(X^3)$ $A_{S_0} \leftarrow A_1 + A_2$ $A_{S_1} \leftarrow A_0 + A_2$ $A_{S_2} \leftarrow A_1 + A_0$ $B_{S_0} \leftarrow B_1 + B_2$ $B_{S_1} \leftarrow B_0 + B_2$ $B_{S_2} \leftarrow B_1 + B_0$	$p_0 \leftarrow A_0 * B_0$ $p_1 \leftarrow A_1 * B_1$ $p_2 \leftarrow A_2 * B_2$ $p'_0 \leftarrow A_{S_0} * B_{S_0}$ $p'_1 \leftarrow A_{S_1} * B_{S_1}$ $p'_2 \leftarrow A_{S_2} * B_{S_2}$	$S \leftarrow (p_0 + X(p'_0 - p_1 - p_2))(X^3)$ $+ X(p'_2 - p_0 - p_1 + Xp_2)(X^3)$ $+ X^2(p_1 + p'_1 - p_2 - p_0)(X^3)$

Multiplication algorithms