Aalto University
School of Science

Elliptic curve cryptography on FPGAs: How fast can we go with a single chip?

Kimmo Järvinen

Department of Information and Computer Science
Aalto University, Finland
kimmo.jarvinen@aalto.fi

ERSA 2011, Las Vegas, NV, USA, July 18–21, 2011

Introduction

Contents

1. What we are computing
2. How we do it and what kind of optimizations are used
3. The results which show that this is the fastest published ECC implementation

Introduction

Contents

1. What we are computing
2. How we do it and what kind of optimizations are used
3. The results which show that this is the fastest published ECC implementation

- Explaining ECC requires some math but I try to keep it in minimum; See the paper for exact definitions and equations if you're interested

Introduction
Contents

1. What we are computing
2. How we do it and what kind of optimizations are used
3. The results which show that this is the fastest published ECC implementation

- Explaining ECC requires some math but I try to keep it in minimum; See the paper for exact definitions and equations if you're interested
- Work that is based on several of my previous publications. Especially, K. Järvinen: "Optimized FPGA-based elliptic curve cryptography processor for high-speed applications," Integration—the VLSI Journal, to appear

Introduction

Fast cryptography with FPGAs

- FPGAs very good platforms for cryptography[1]

Example

Architecture efficiency:

Reprogrammability allows optimizing for specific parameters because if parameters change, we can change the design

[1] Wollinger et al., ACM Trans. Embed. Comput. Syst. 3(3):534-574, 2004

Introduction

Fast cryptography with FPGAs

- FPGAs very good platforms for cryptography[1]

Example

Architecture efficiency:

Reprogrammability allows optimizing for specific parameters because if parameters change, we can change the design

- In this work, we fix the curve to a standardized curve NIST K-163 (both the curve and the underlying finite field is fixed)

[1] Wollinger et al., ACM Trans. Embed. Comput. Syst. 3(3):534-574, 2004

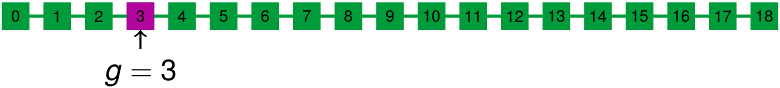
Preliminaries
(mathematical background)

Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

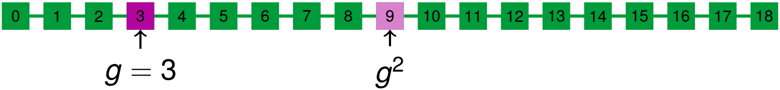


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

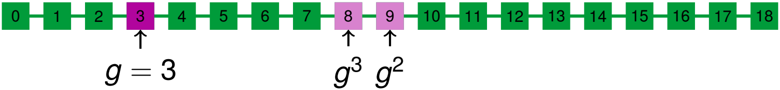


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

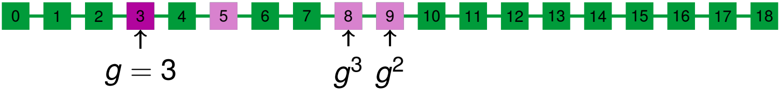


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

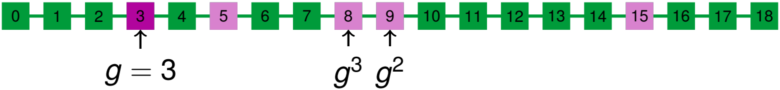


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

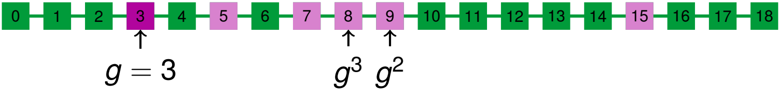


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

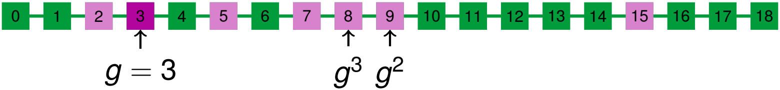


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

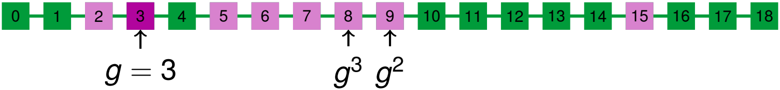


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

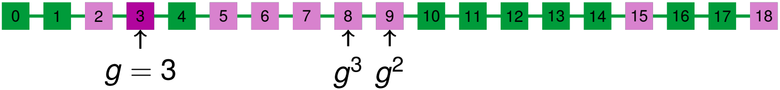


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

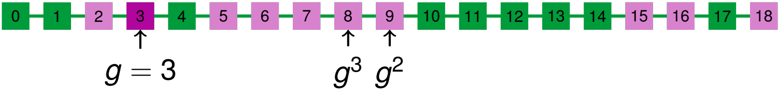


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

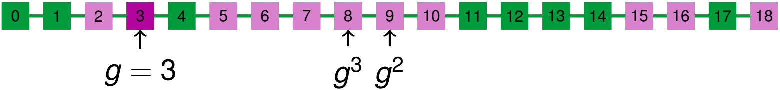


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

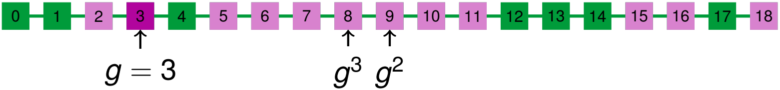


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

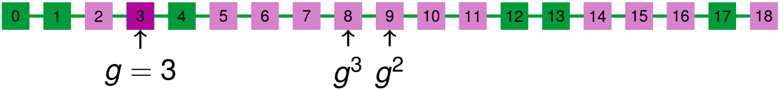


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

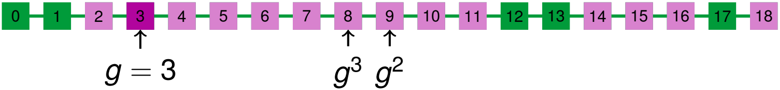


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

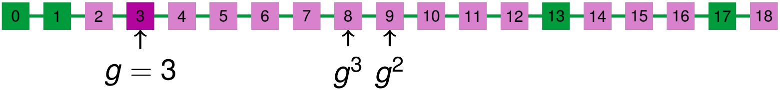


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

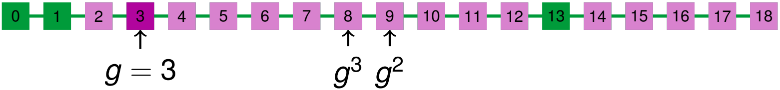


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

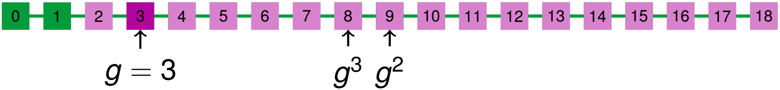


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

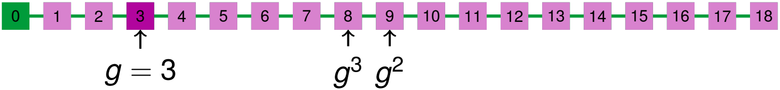


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

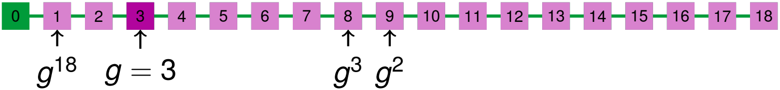


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$

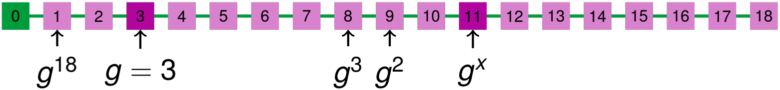


Public-key cryptography

- The encryption key y is derived from the decryption key x in a one-way manner
- Hence, y can be public if x remains private

Example

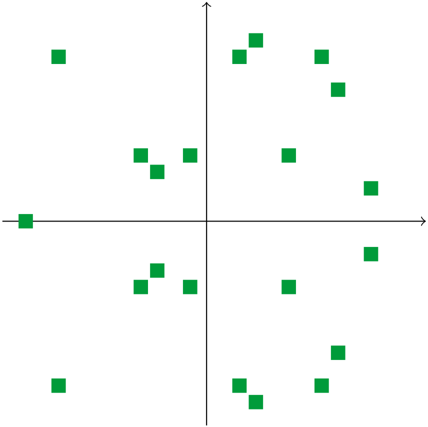
- Exponentiation: $y = g^x \bmod p$
- $y = 3^x \bmod 19$
- Discrete logarithm problem (DLP): $x = \log_g(y) \bmod p$



What is x?

Elliptic curve cryptography

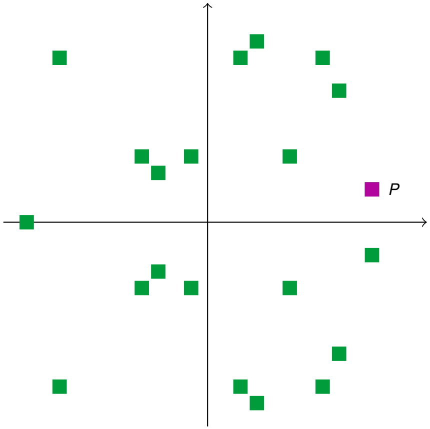
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$E : y^2 = x^3 + 2x - 4 \pmod{23}$

Elliptic curve cryptography

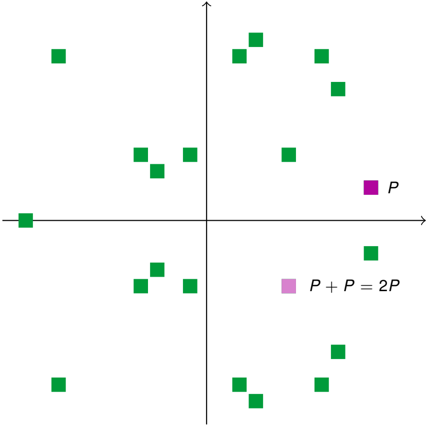
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

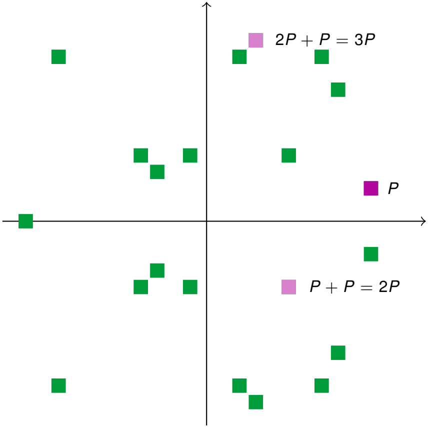
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

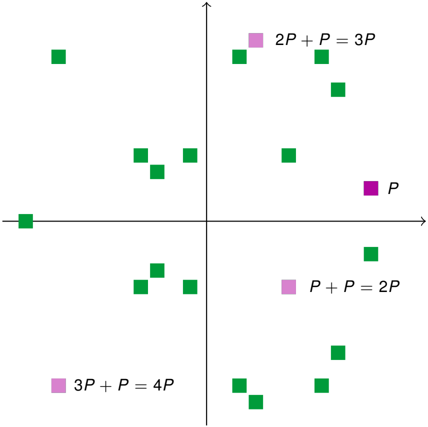
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

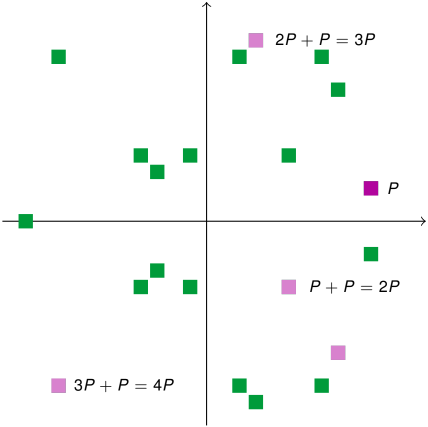
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

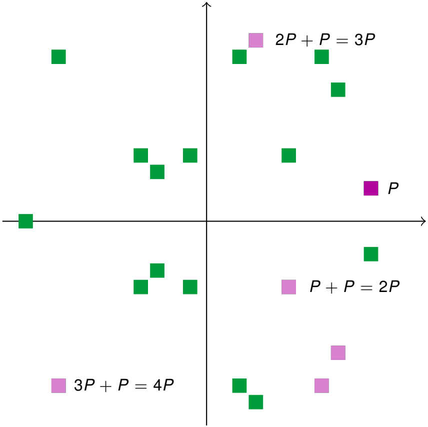
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

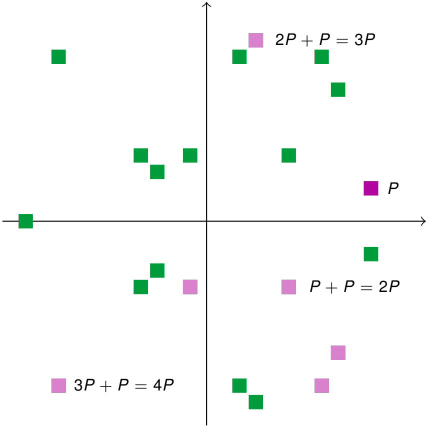
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

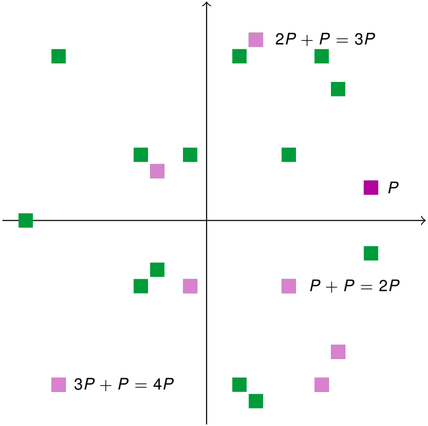
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

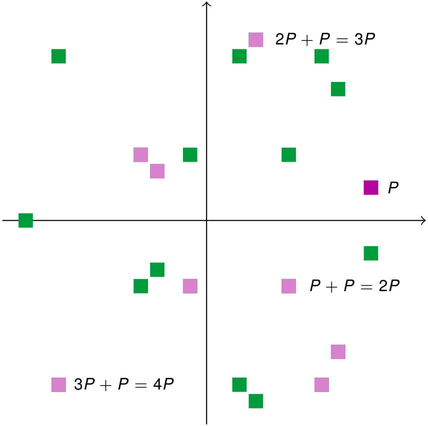
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$E : y^2 = x^3 + 2x - 4 \pmod{23}$

Elliptic curve cryptography

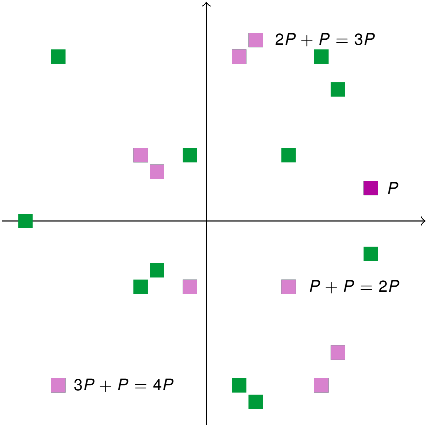
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

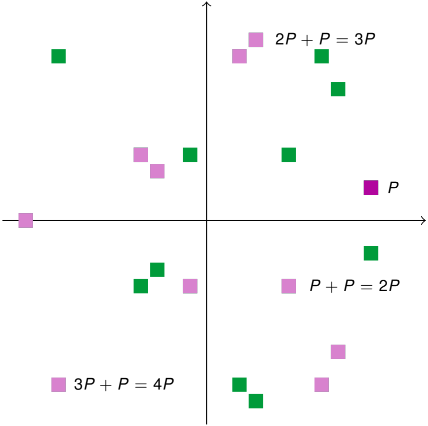
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

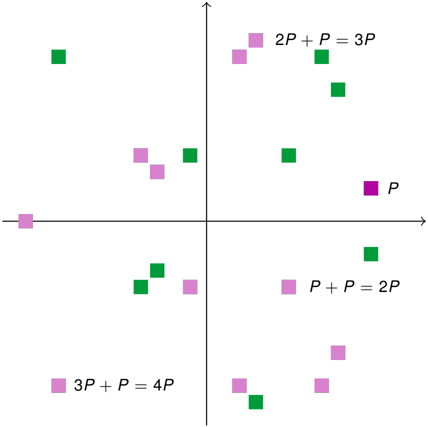
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

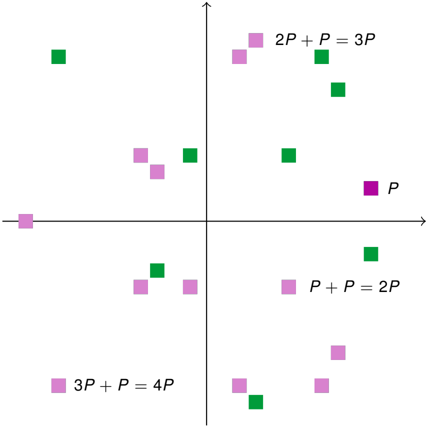
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

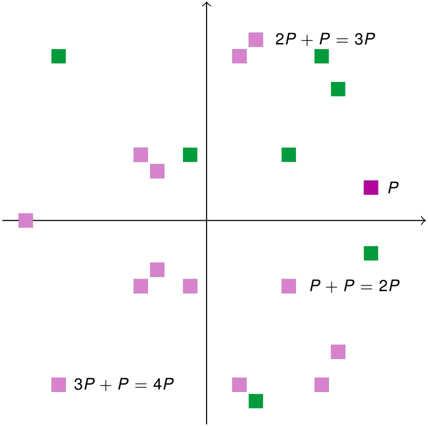
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

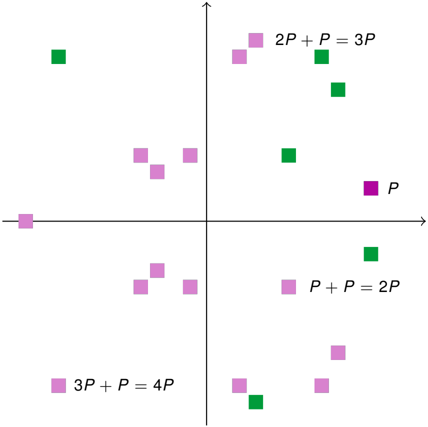
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

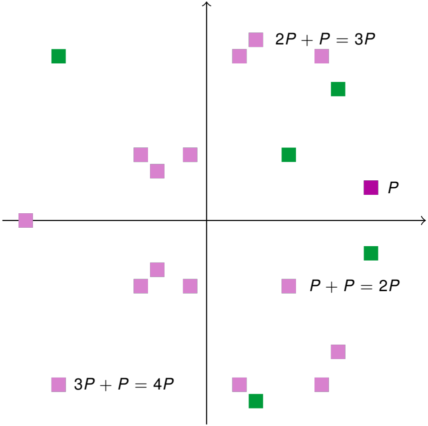
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

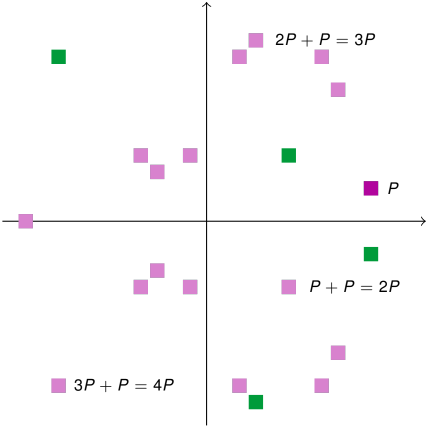
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

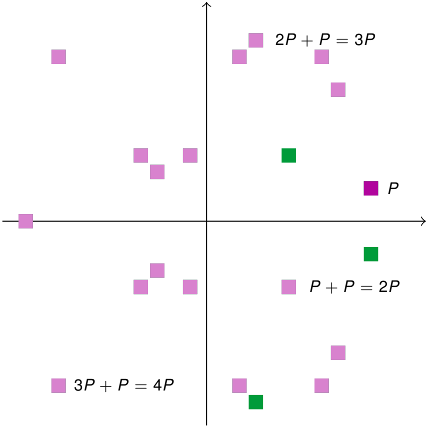
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

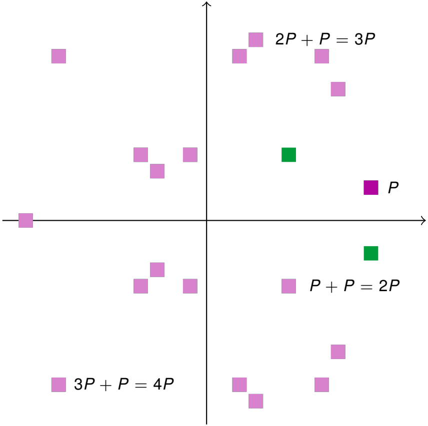
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

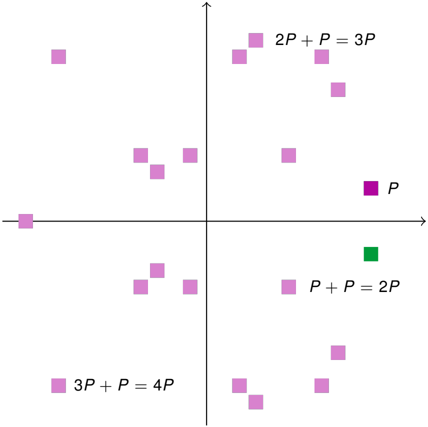
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

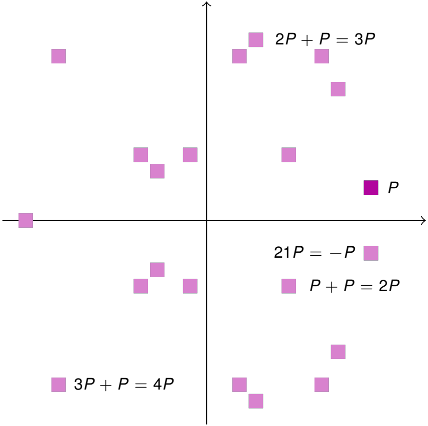
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

Elliptic curve cryptography

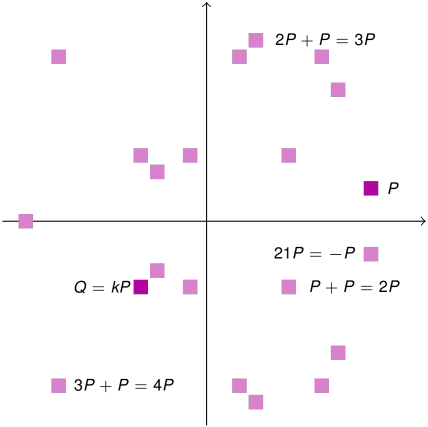
- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E



$E : y^2 = x^3 + 2x - 4 \pmod{23}$

Elliptic curve cryptography

- Points on E form an additive Abelian group
- We can compute additions: $R = P + Q$ so that P, Q, R are points on E
- EC DLP: Given P and $Q = kP$ determine k
- Elliptic curve cryptosystems analogous to systems based on DLP
- DLP is much harder in EC groups $\Rightarrow$ Shorter keys!



$$E : y^2 = x^3 + 2x - 4 \pmod{23}$$

What are we actually computing?

- Scalar multiplication: $Q = kP$

What are we actually computing?

- Scalar multiplication: $Q = kP$
- Point operations: Point addition and point doubling
  Example (point addition): $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2; \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

What are we actually computing?

- Scalar multiplication: $Q = kP$
- Point operations: Point addition and point doubling
  Example (point addition): $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2; \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

- Field operations: Multiplication, addition, subtraction, inversion
  Example (multiplication)

$$c = a \times b \bmod p, \text{ where } a, b, c \in [0, p-1] \text{ and } p \text{ is prime}$$

The width of the operands is typically $[160, 600]$

What are we actually computing?

- Scalar multiplication: $Q = kP$
- Point operations: Point addition and point doubling
  Example (point addition): $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2; \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

- Field operations: Multiplication, addition, subtraction, inversion
  Example (multiplication)

$$c = a \times b \bmod p, \text{ where } a, b, c \in [0, p-1] \text{ and } p \text{ is prime}$$

  The width of the operands is typically $[160, 600]$
- Binary fields $\Rightarrow$ No carry!

Scalar multiplication

- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$

$\Rightarrow$

Scalar multiplication

- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$

$\Rightarrow \qquad P = P$

Scalar multiplication

- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$

$\Rightarrow \quad 2^3 P + P = 9P$

Scalar multiplication

- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$

$\Rightarrow \quad 2^7 P + 2^3 P + P = 137P$

Scalar multiplication

- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$

$\Rightarrow \quad 2^{10}P + 2^7 P + 2^3 P + P = 1161P$

Scalar multiplication

- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$

$\Rightarrow \quad 2^{11}P + 2^{10}P + 2^7 P + 2^3 P + P = 3209P$

Scalar multiplication

- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$
$\Rightarrow 2^{14}P + 2^{11}P + 2^{10}P + 2^7P + 2^3P + P = 19593P$

Scalar multiplication

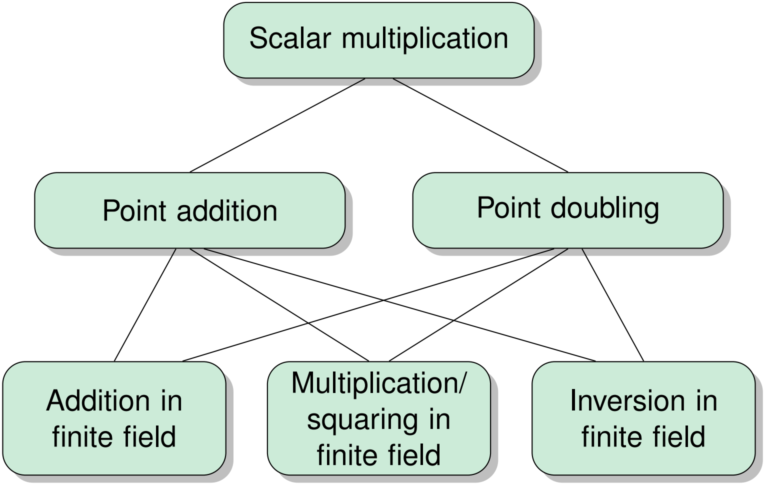
- Scalar multiplication is the main operation in all EC cryptosystem: $Q = kP$
- Double-and-add algorithm:
  Point doubling $P \leftarrow 2P$ for all k_i; and
  Point addition $Q \leftarrow Q + P$ if $k_i = 1$

Example

$Q = 19593P = (100110010001001)_2 P$
$\Rightarrow 2^{14}P + 2^{11}P + 2^{10}P + 2^{7}P + 2^{3}P + P = 19593P$
Only 14 point doublings and 5 point additions

- In practice, k is 100–600 bits long; in our case, $\log_2(k) \approx 160$ and we need $\sim$160 point doublings and $\sim$80 point additions

Hierarchy


Scalar multiplication
Point addition
Point doubling
Addition in finite field
Multiplication/ squaring in finite field
Inversion in finite field

Koblitz curves

- Special elliptic curves over binary fields (binary polynomials modulo an irreducible polynomial)

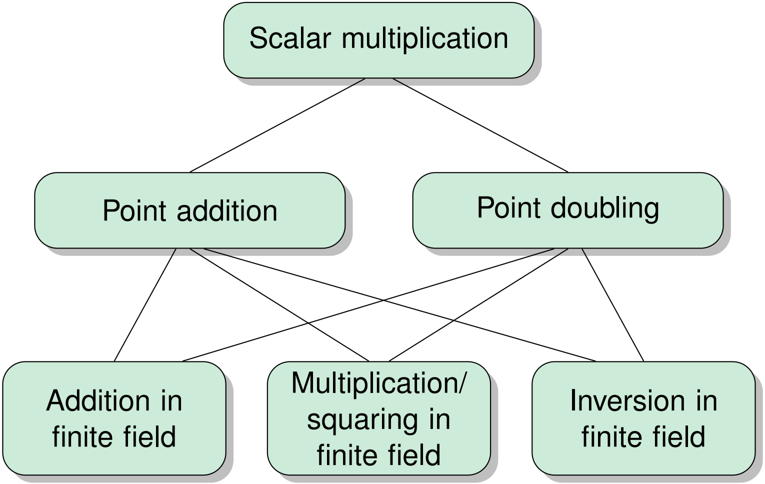
Koblitz curves

- Special elliptic curves over binary fields (binary polynomials modulo an irreducible polynomial)
- If the point (x, y) is on the curve, then also (x^2, y^2) is on the curve
  $\Rightarrow$ Point doublings can be replaced with $\phi(P) = (x^2, y^2)$ operations! Squaring is cheap in binary fields!
- But, k must be represented in τ-adic form: $k = \sum k_i \tau^i$ where $\tau \in \mathbb{C}$ (instead of binary form $\sum k_i 2^i$)

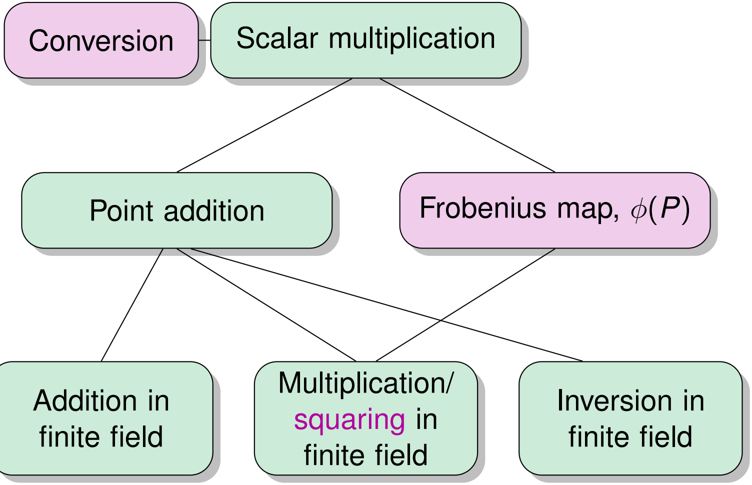
Koblitz curves

- Special elliptic curves over binary fields (binary polynomials modulo an irreducible polynomial)
- If the point (x, y) is on the curve, then also (x^2, y^2) is on the curve
  $\Rightarrow$ Point doublings can be replaced with $\phi(P) = (x^2, y^2)$ operations! Squaring is cheap in binary fields!
- But, k must be represented in τ-adic form: $k = \sum k_i \tau^i$ where $\tau \in \mathbb{C}$ (instead of binary form $\sum k_i 2^i$)
- Scalar multiplication: $k_{\ell-1}\phi^{\ell-1}P + \ldots + k_1\phi P + k_0 P$
  - Compute $\phi(P)$ (very cheap!) instead of point doublings
  - Otherwise, the algorithm is similar
- We need a converter!

Hierarchy (Normal curves)



Hierarchy (Koblitz curves)

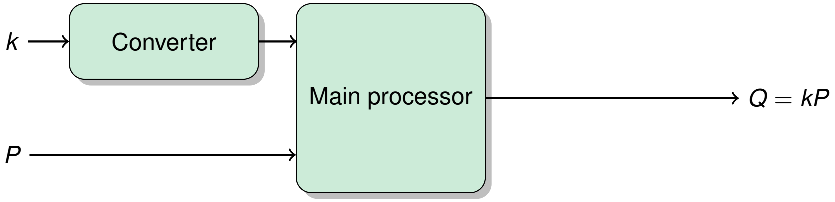


Processor architecture

Elliptic curve processor for Koblitz curves

Computations on Koblitz curves are performed with:

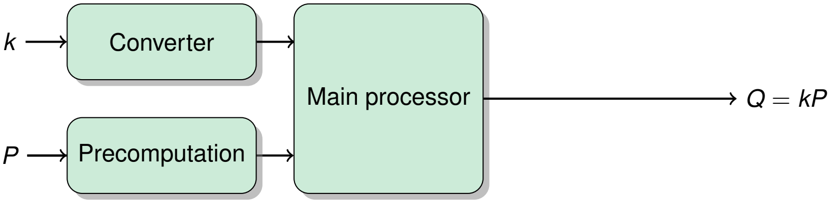
- Conversion for k
- Scalar multiplication



Elliptic curve processor for Koblitz curves

Computations on Koblitz curves are performed with:

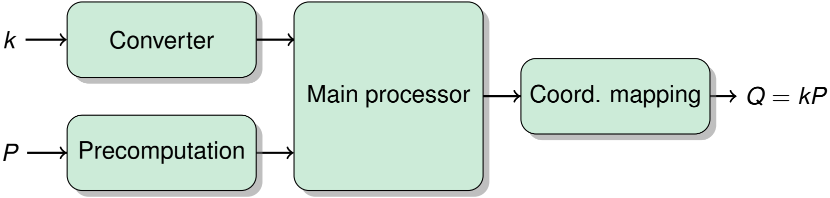
- Conversion for k
- Precomputations with the base point P
- Scalar multiplication with the precomputed points



Elliptic curve processor for Koblitz curves

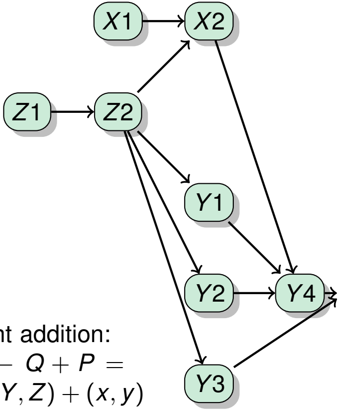
Computations on Koblitz curves are performed with:

- Conversion for k
- Precomputations with the base point P
- Scalar multiplication with the precomputed points in projective coordinates ($P = (X, Y, Z)$)
- Mapping from (X, Y, Z) back to (x, y)



Main processor: Interleaving point additions

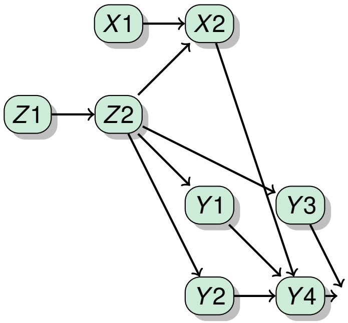
- 8 multiplications, critical path 4 multiplications



Point addition:
$Q \leftarrow Q + P = (X, Y, Z) + (x, y)$

Main processor: Interleaving point additions

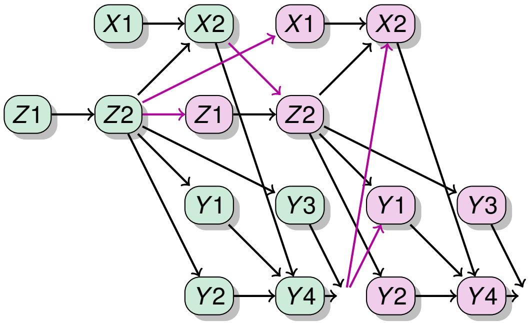
- 8 multiplications, critical path 4 multiplications



Point addition:
$Q \leftarrow Q + P = (X, Y, Z) + (x, y)$

Main processor: Interleaving point additions

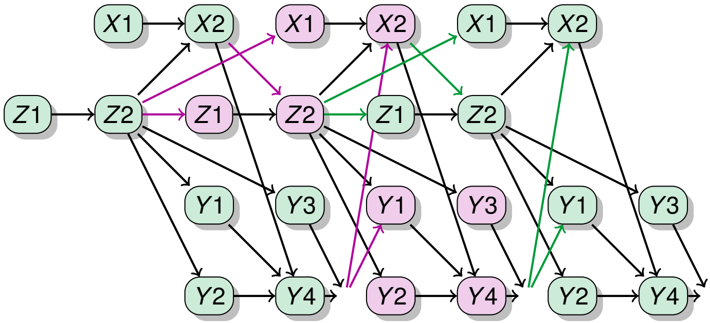
- 8 multiplications, critical path 4 multiplications
- With 4 multipliers, critical path 2 multiplications / p. addition



Point addition:
$Q \leftarrow Q + P =$
$(X, Y, Z) + (x, y)$

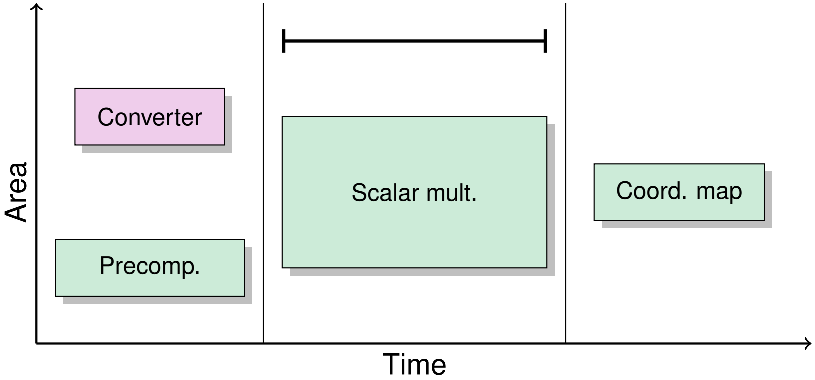
Main processor: Interleaving point additions

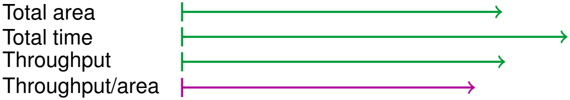
- 8 multiplications, critical path 4 multiplications
- With 4 multipliers, critical path 2 multiplications / p. addition



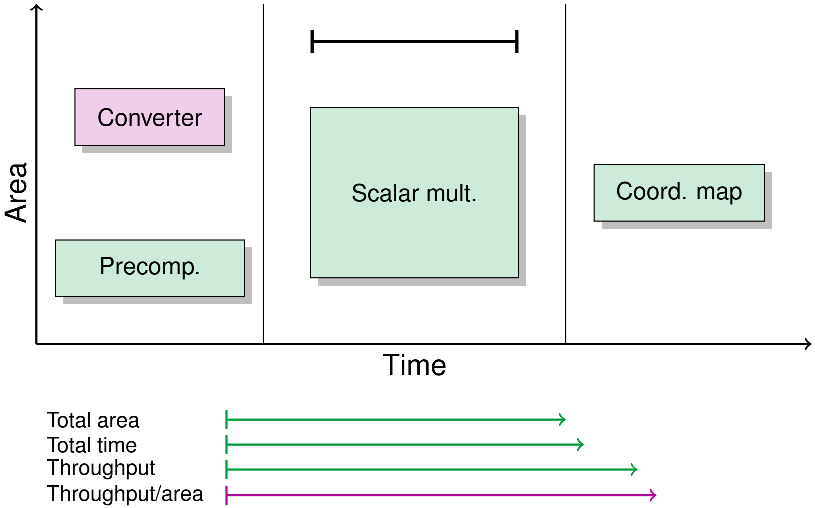
Point addition:
$Q \leftarrow Q + P = (X, Y, Z) + (x, y)$

Balancing the pipeline

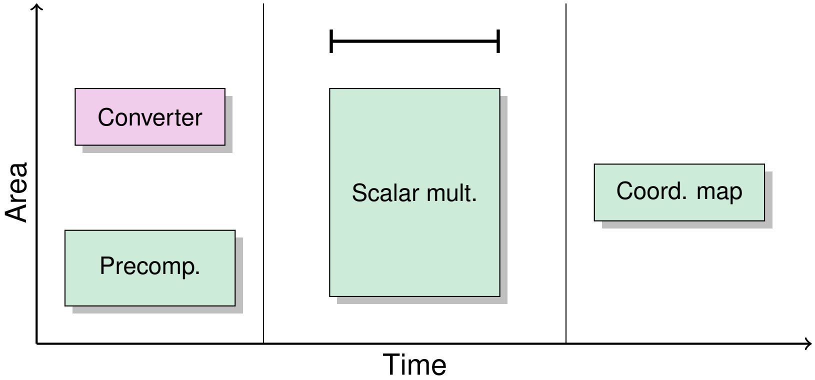


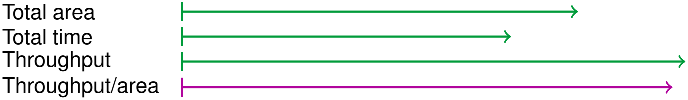


Balancing the pipeline

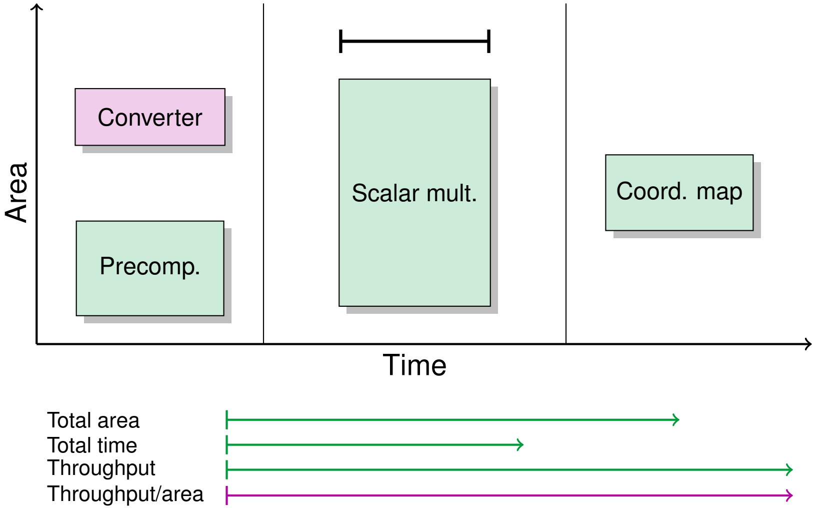


Balancing the pipeline

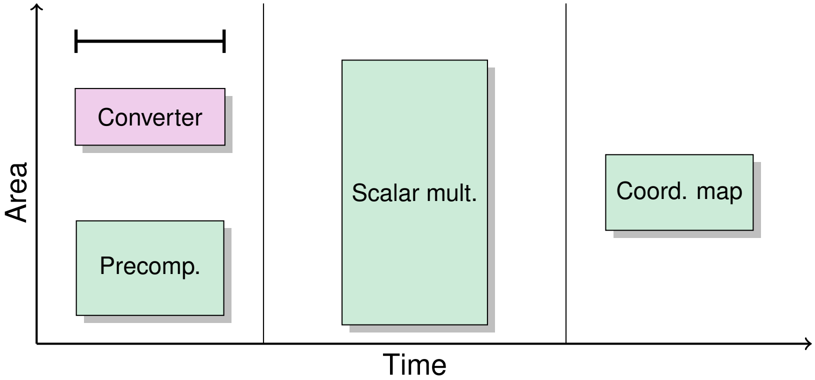




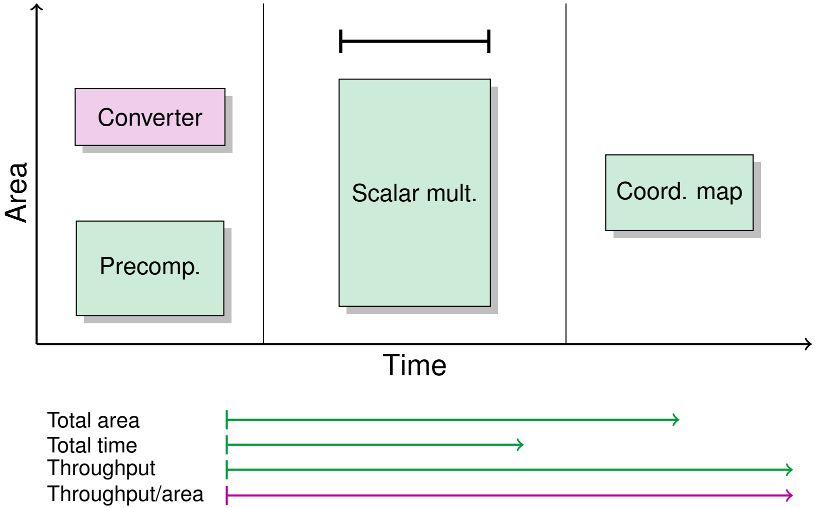
Balancing the pipeline



Balancing the pipeline

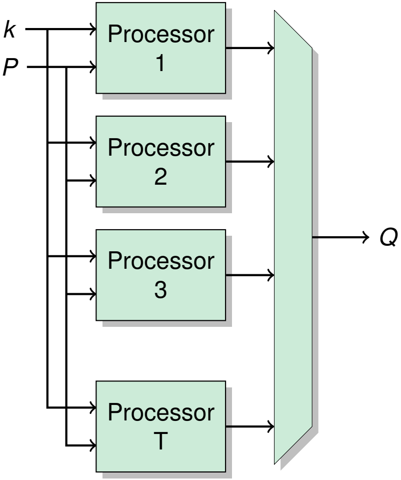


Balancing the pipeline



Parallelization

- The strategy is to replicate T processors with maximum throughput / area ratio
- For example, we can fit $T = 5$ processors in a Stratix IV GX230 FPGA



Results & conclusions

Results

Table: Results on Stratix IV GX EP4SGX230KF40C2.

ALUTs	78,695 (43 %)
Regs	61,871 (34 %)
ALMs	74,750 (82 %)
M9K	105 (9 %)
Clock, converter	120 MHz
Clock, others	266 MHz
Time	8.3 μs
Throughput	1,693,000

Summary

FPGA-based implementation of ECC that…

1. …is the fastest published implementation
   (almost 1,700,000 scalar multiplication / sec.)
2. …is optimized for a specific curve on every level
3. …uses a lot of parallelism
4. …relies on reprogrammability of FPGAs
   (fixed and highly optimized implementations would be impractical without reprogrammability)

Thank you![2]
Questions?

[2]... and thanks to Emil Aaltonen Foundation for a grant covering the expenses of this trip!