CACAO seminar – March 2009

Hardware Operators for Pairing-Based Cryptography

— Part II: Because speed also matters —

Jérémie Detrey
CACAO project-team, LORIA
INRIA Nancy – Grand Est
Jeremie.Detrey@loria.fr

Joint work with:

Jean-Luc Beuchat	LCIS, University of Tsukuba, Japan
Nicolas Brisebarre	Arénaire, LIP, ÉNS Lyon, France
Nicolas Estibals	CACAO, LORIA, Nancy, France
Eiji Okamoto	LCIS, University of Tsukuba, Japan
Francisco Rodríguez-Henríquez	CSD, IPN, Mexico City, Mexico

Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

Bilinear pairings

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

Bilinear pairings

- $\mathbb{G}_1 = \langle P \rangle$: additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- $\mathbb{G}_2$: multiplicatively-written cyclic group of order $\#\mathbb{G}_2 = \#\mathbb{G}_1 = \ell$
- A bilinear pairing on $(\mathbb{G}_1, \mathbb{G}_2)$ is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$$

  that satisfies the following conditions:

  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$ (equivalently $\hat{e}(P, P)$ generates $\mathbb{G}_2$)
  - bilinearity:

    $$\hat{e}(Q_1 + Q_2, R) = \hat{e}(Q_1, R) \cdot \hat{e}(Q_2, R) \qquad \hat{e}(Q, R_1 + R_2) = \hat{e}(Q, R_1) \cdot \hat{e}(Q, R_2)$$

  - computability: $\hat{e}$ can be efficiently computed

Bilinear pairings

- $\mathbb{G}_1 = \langle P \rangle$: additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- $\mathbb{G}_2$: multiplicatively-written cyclic group of order $\#\mathbb{G}_2 = \#\mathbb{G}_1 = \ell$
- A bilinear pairing on $(\mathbb{G}_1, \mathbb{G}_2)$ is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$$

  that satisfies the following conditions:
  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$ (equivalently $\hat{e}(P, P)$ generates $\mathbb{G}_2$)
  - bilinearity:

$$\hat{e}(Q_1 + Q_2, R) = \hat{e}(Q_1, R) \cdot \hat{e}(Q_2, R) \qquad \hat{e}(Q, R_1 + R_2) = \hat{e}(Q, R_1) \cdot \hat{e}(Q, R_2)$$

  - computability: $\hat{e}$ can be efficiently computed
- Immediate property: for any two integers k_1 and k_2

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

Pairings in cryptography

- At first, used to attack supersingular elliptic curves
  - Menezes-Okamoto-Vanstone attack, 1993
  - Frey-Rück attack, 1994

Pairings in cryptography

- At first, used to attack supersingular elliptic curves
  - Menezes-Okamoto-Vanstone attack, 1993
  - Frey-Rück attack, 1994
- One-round three-party key agreement (Joux, 2000)
- Identity-based encryption
  - Boneh-Franklin, 2001
  - Sakai-Kasahara, 2001
- Short digital signatures
  - Boneh-Lynn-Shacham, 2001
  - Zang-Safavi-Naini-Susilo, 2004
- ...

The Tate pairing over supersingular elliptic curves

- We first define
  - $\mathbb{F}_{p^m}$, a finite field, with $p = 2$ or 3
  - E, a supersingular elliptic curve defined over $\mathbb{F}_{p^m}$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_{p^m})$

The Tate pairing over supersingular elliptic curves

- We first define
  - $\mathbb{F}_{p^m}$, a finite field, with $p = 2$ or 3
  - E, a supersingular elliptic curve defined over $\mathbb{F}_{p^m}$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_{p^m})$
- $\mathbb{G}_1 = E(\mathbb{F}_{p^m})[\ell]$, the $\mathbb{F}_{p^m}$-rational ℓ-torsion of E:

$$\mathbb{G}_1 = \{P \in E(\mathbb{F}_{p^m}) \mid \ell P = \mathcal{O}\}$$

The Tate pairing over supersingular elliptic curves

- We first define
  - $\mathbb{F}_{p^m}$, a finite field, with $p = 2$ or 3
  - E, a supersingular elliptic curve defined over $\mathbb{F}_{p^m}$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_{p^m})$

- $\mathbb{G}_1 = E(\mathbb{F}_{p^m})[\ell]$, the $\mathbb{F}_{p^m}$-rational ℓ-torsion of E:

$$\mathbb{G}_1 = \{P \in E(\mathbb{F}_{p^m}) \mid \ell P = \mathcal{O}\}$$

- $\mathbb{G}_2 = \mu_\ell$, the group of ℓ-th roots of unity in $\mathbb{F}_{p^{km}}^\times$:

$$\mathbb{G}_2 = \{U \in \mathbb{F}_{p^{km}}^\times \mid U^\ell = 1\}$$

The Tate pairing over supersingular elliptic curves

- We first define
  - $\mathbb{F}_{p^m}$, a finite field, with $p = 2$ or 3
  - E, a supersingular elliptic curve defined over $\mathbb{F}_{p^m}$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_{p^m})$
- $\mathbb{G}_1 = E(\mathbb{F}_{p^m})[\ell]$, the $\mathbb{F}_{p^m}$-rational ℓ-torsion of E:

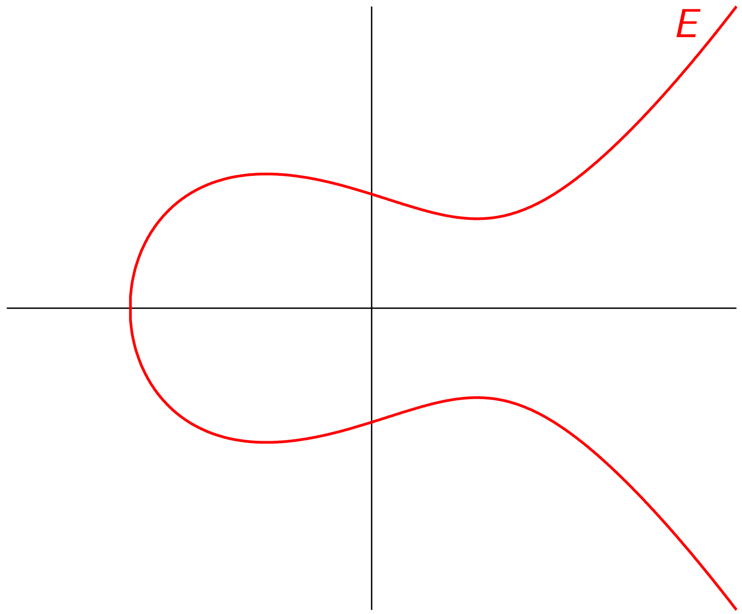
$$\mathbb{G}_1 = \{P \in E(\mathbb{F}_{p^m}) \mid \ell P = \mathcal{O}\}$$

- $\mathbb{G}_2 = \mu_\ell$, the group of ℓ-th roots of unity in $\mathbb{F}_{p^{km}}^\times$:

$$\mathbb{G}_2 = \{U \in \mathbb{F}_{p^{km}}^\times \mid U^\ell = 1\}$$

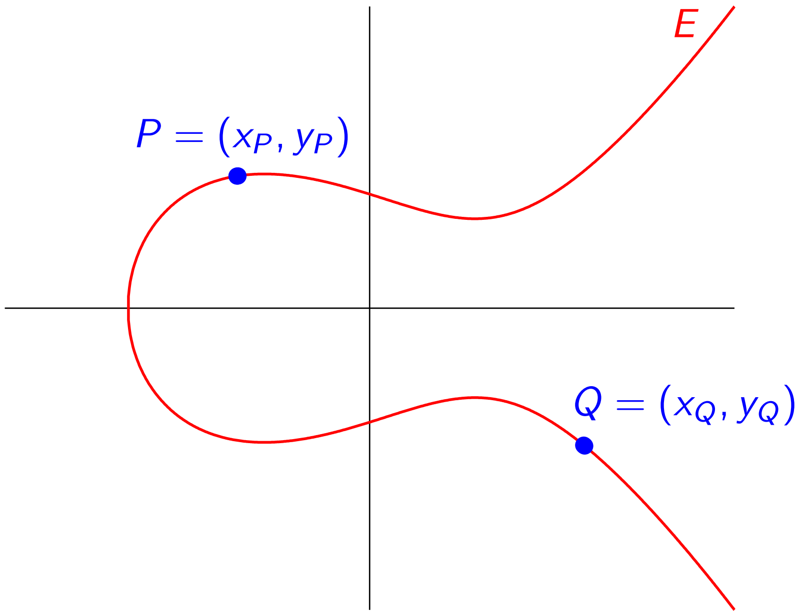
- k is the embedding degree, the smallest integer such that $\mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$
  - $k = 4$ in characteristic $p = 2$
  - $k = 6$ in characteristic $p = 3$

The Tate pairing over supersingular elliptic curves



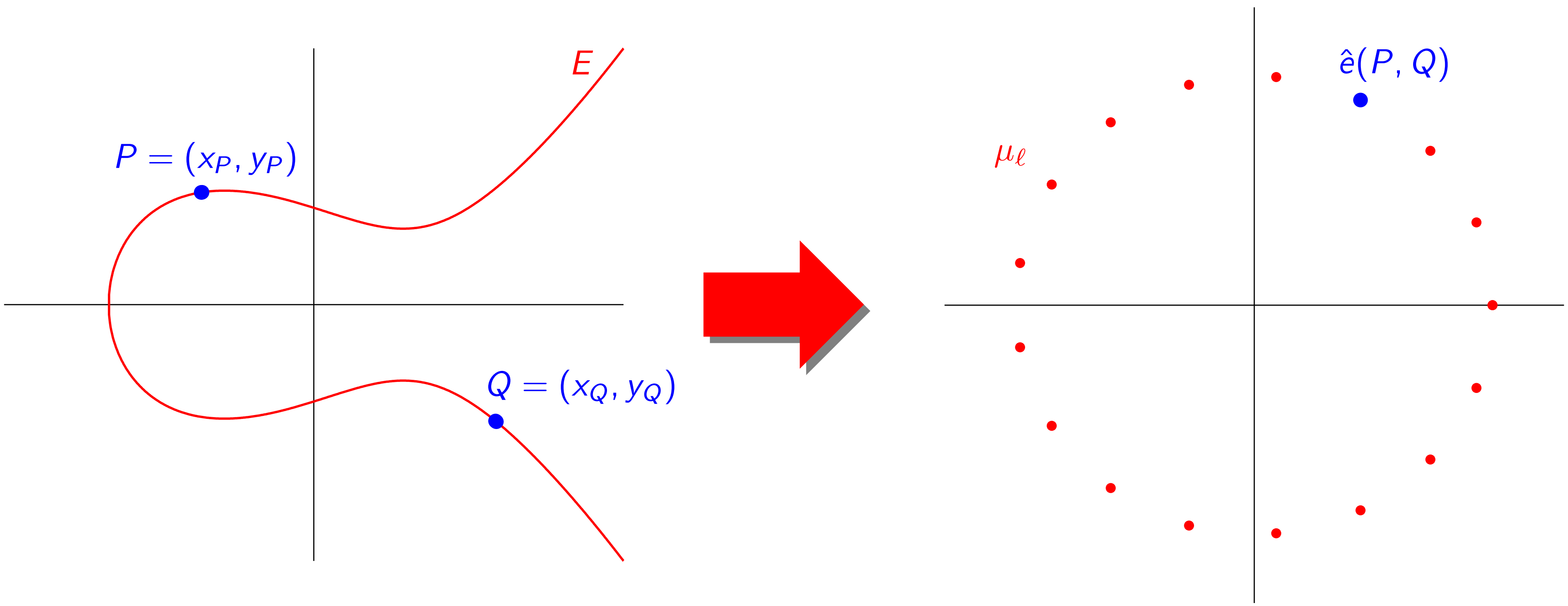
$\hat{e}$

The Tate pairing over supersingular elliptic curves



$$\begin{array}{cccc} \hat{e} & : & E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \\ & & (\quad P \quad , \quad Q \quad) \end{array}$$

The Tate pairing over supersingular elliptic curves



$$\begin{array}{rcccl} \hat{e} : & E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] & \longrightarrow & \mu_\ell & \subseteq \mathbb{F}_{p^{km}}^\times \\ & (\quad P \quad , \quad Q \quad) & \longmapsto & \hat{e}(P,Q) & \end{array}$$

Security considerations

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- The discrete logarithm problem should be hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$

Security considerations

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- The discrete logarithm problem should be hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$

Base field ($\mathbb{F}_{p^m}$)	$\mathbb{F}_{2^m}$	$\mathbb{F}_{3^m}$
Lower security ($\sim 2^{64}$)	$m = 239$	$m = 97$
Medium security ($\sim 2^{80}$)	$m = 373$	$m = 163$
Higher security ($\sim 2^{128}$)	$m = 1103$	$m = 503$

- $\mathbb{F}_{2^m}$: simpler finite field arithmetic
- $\mathbb{F}_{3^m}$: smaller field extension

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \longrightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$

- Arithmetic over $\mathbb{F}_{p^{km}}^\times$:
  - tower-field representation
  - only arithmetic over the underlying field $\mathbb{F}_{p^m}$

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \to \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$
- Arithmetic over $\mathbb{F}_{p^{km}}^\times$:
  - tower-field representation
  - only arithmetic over the underlying field $\mathbb{F}_{p^m}$
- Operations over $\mathbb{F}_{p^m}$:
  - $O(m)$ additions / subtractions
  - $O(m)$ multiplications
  - $O(m)$ Frobenius maps ($a \mapsto a^p$, *i.e.* squarings or cubings)
  - 1 inversion

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \to \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

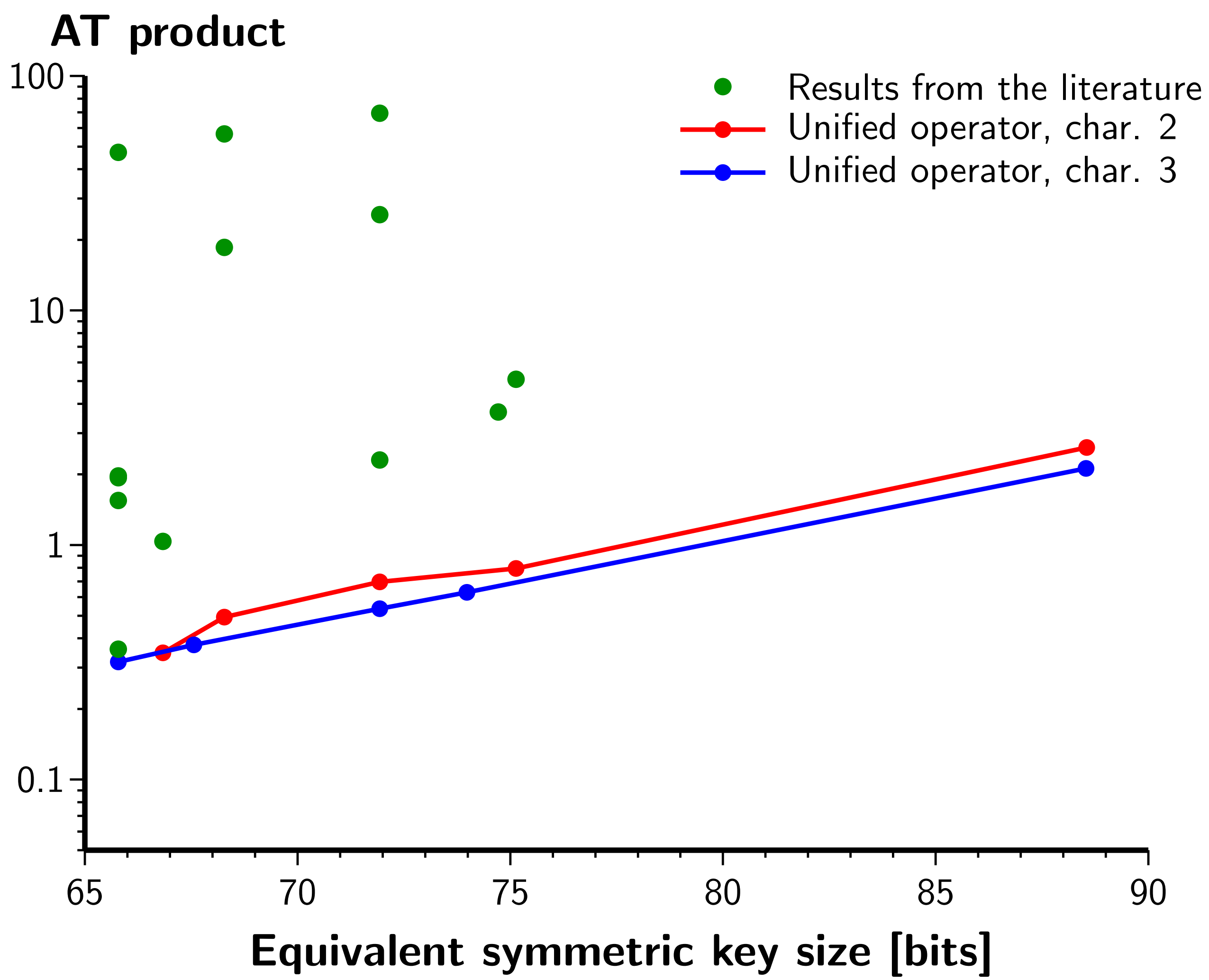
- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$

- Arithmetic over $\mathbb{F}_{p^{km}}^\times$:
  - tower-field representation
  - only arithmetic over the underlying field $\mathbb{F}_{p^m}$

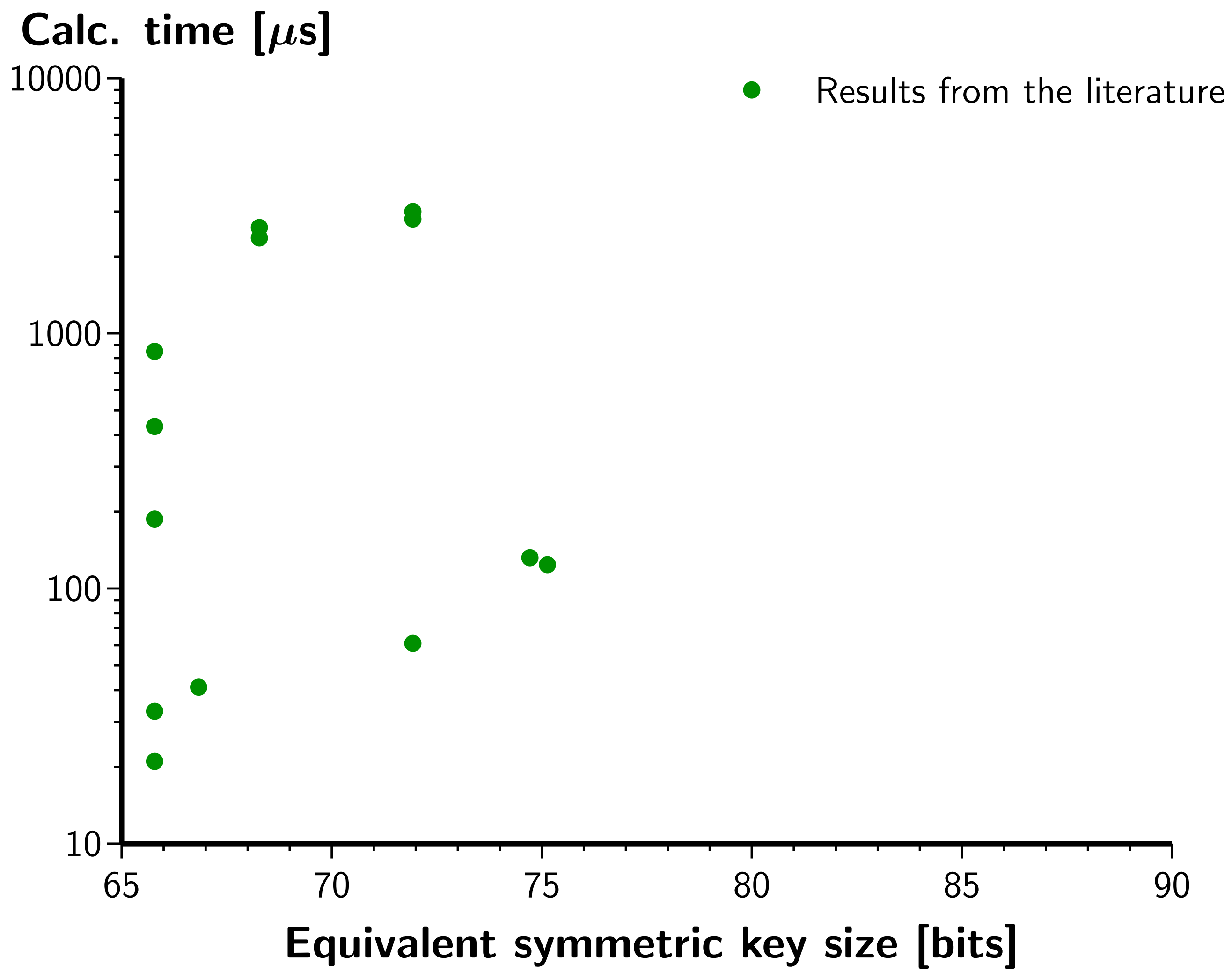
- Operations over $\mathbb{F}_{p^m}$:
  - $O(m)$ additions / subtractions
  - $O(m)$ multiplications
  - $O(m)$ Frobenius maps ($a \mapsto a^p$, *i.e.* squarings or cubings)
  - 1 inversion

- A first idea: an all-in-one unified operator:
  - shared resources
  - scalable architecture

The best area-time product of the literature...

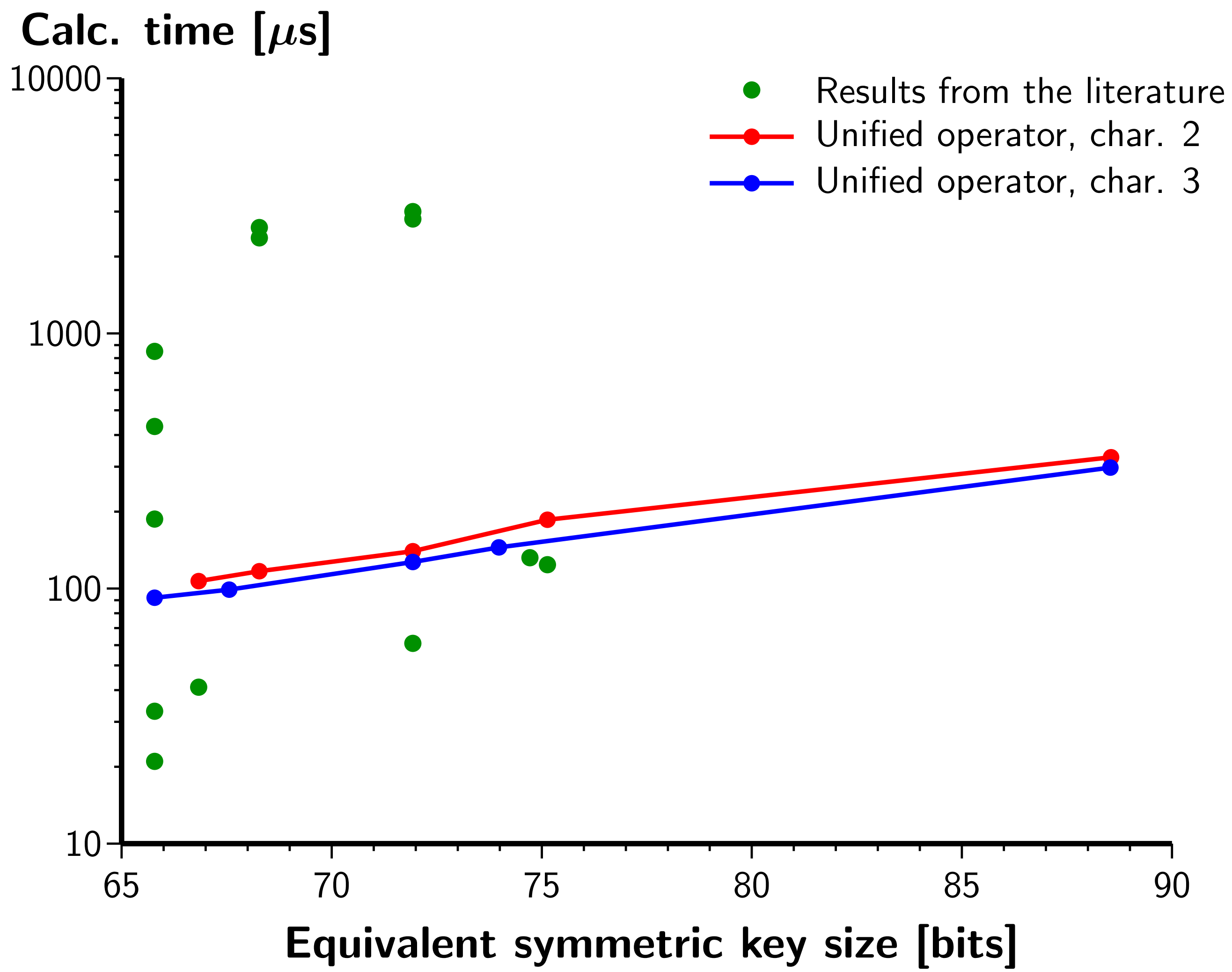


… But still quite slow



... But still quite slow

(or not the fastest, at least!)



Motivations

- High speed is more important than low resources for some cryptographic applications
- Explore the other end of the area vs. time tradeoff:
  - faster but larger than the unified operator
  - what about the area-time product?

Motivations

- High speed is more important than low resources for some cryptographic applications
- Explore the other end of the area vs. time tradeoff:
  - faster but larger than the unified operator
  - what about the area-time product?
- Accelerate the computation by extracting as much parallelism as possible...
- ... Without dramatically increasing the resource requirements

Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

Computation of the Tate pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P, Q)$$

Computation of the Tate pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P, Q) = \eta_T(P, Q)$$

- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m+1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}_{p^{km}}^{\times}$, with $N = \#E(\mathbb{F}_{p^m})$

Computation of the Tate pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P, Q) = \eta_T(P, Q)^M$$

- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m+1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}_{p^{km}}^{\times}$, with $N = \#E(\mathbb{F}_{p^m})$

- Final exponentiation by $M = (p^{km} - 1)/N$
  - required to obtain a unique value for each congruence class
  - example in characteristic 3 ($k = 6$ and $N = 3^m + 1 \pm 3^{(m+1)/2}$):

$$M = \frac{3^{6m} - 1}{3^m + 1 \pm 3^{(m+1)/2}} = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

  - exploit the special form of the exponent: *ad-hoc* algorithm

Computation of the Tate pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P,Q) = \eta_T(P,Q)^M$$

- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m+1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}_{p^{km}}^{\times}$, with $N = \#E(\mathbb{F}_{p^m})$

- Final exponentiation by $M = (p^{km} - 1)/N$
  - required to obtain a unique value for each congruence class
  - example in characteristic 3 ($k = 6$ and $N = 3^m + 1 \pm 3^{(m+1)/2}$):

$$M = \frac{3^{6m} - 1}{3^m + 1 \pm 3^{(m+1)/2}} = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

  - exploit the special form of the exponent: *ad-hoc* algorithm

- Two distinct computational requirements

Computation of the Tate pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P,Q) = \eta_T(P,Q)^M$$

- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m+1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}_{p^{km}}^{\times}$, with $N = \#E(\mathbb{F}_{p^m})$

- Final exponentiation by $M = (p^{km} - 1)/N$
  - required to obtain a unique value for each congruence class
  - example in characteristic 3 ($k = 6$ and $N = 3^m + 1 \pm 3^{(m+1)/2}$):

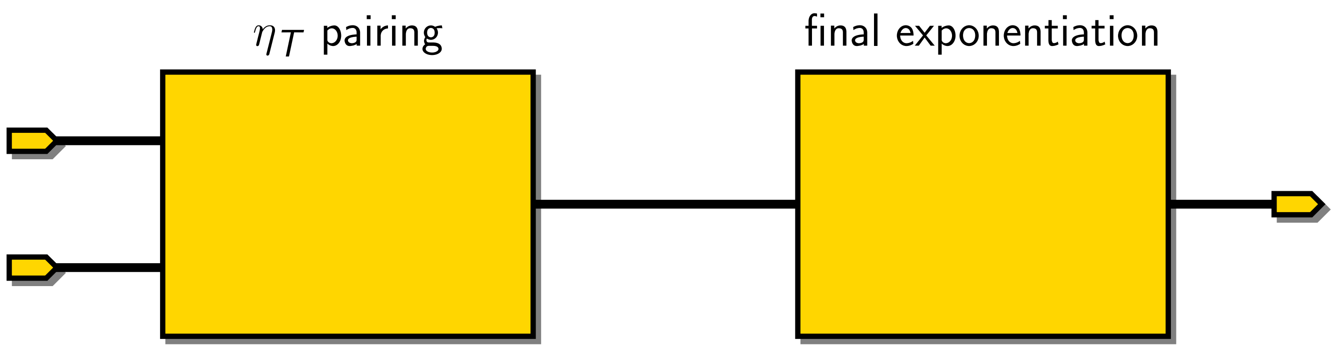
$$M = \frac{3^{6m} - 1}{3^m + 1 \pm 3^{(m+1)/2}} = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

  - exploit the special form of the exponent: *ad-hoc* algorithm

- Two distinct computational requirements $\Rightarrow$ use two distinct coprocessors

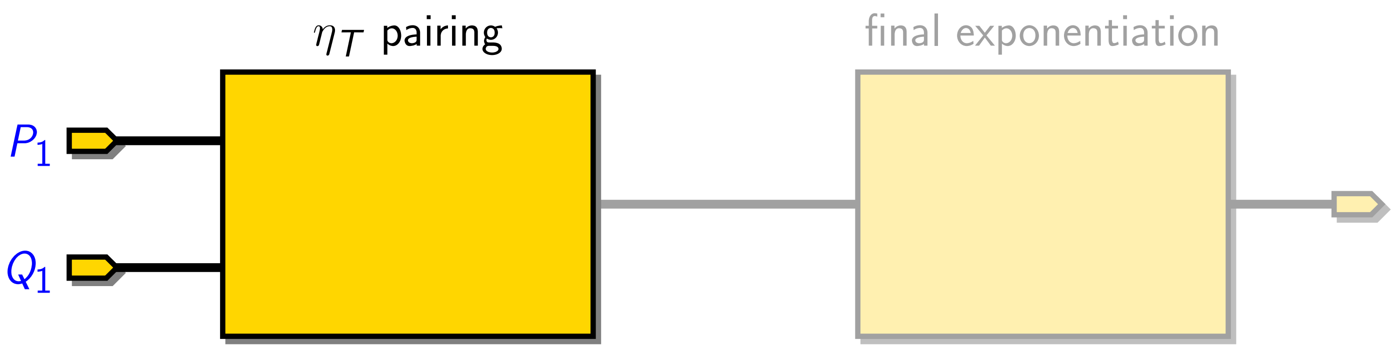
Two coprocessors for the Tate pairing

- The two operations are purely sequential



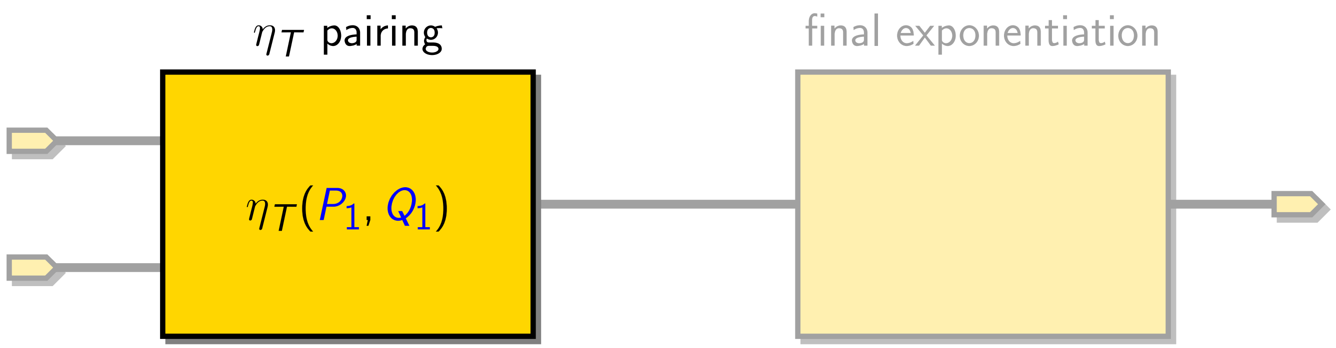
Two coprocessors for the Tate pairing

- The two operations are purely sequential



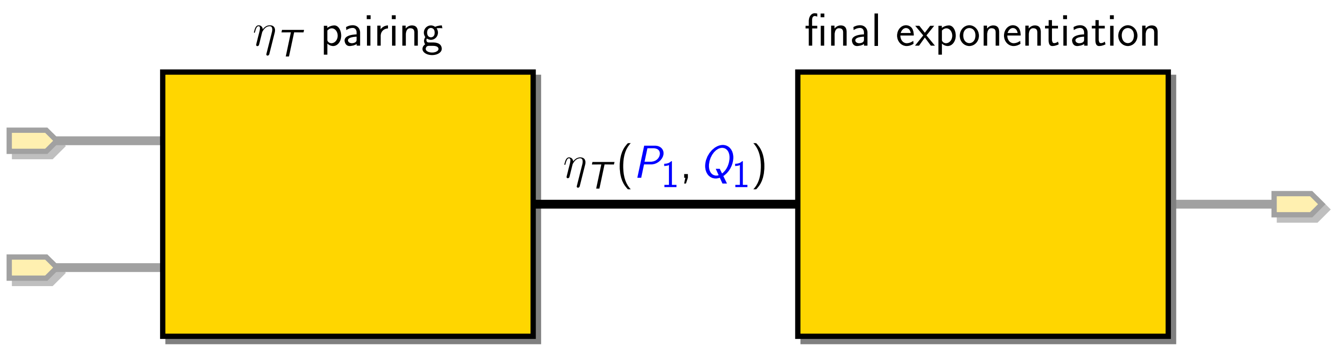
Two coprocessors for the Tate pairing

- The two operations are purely sequential



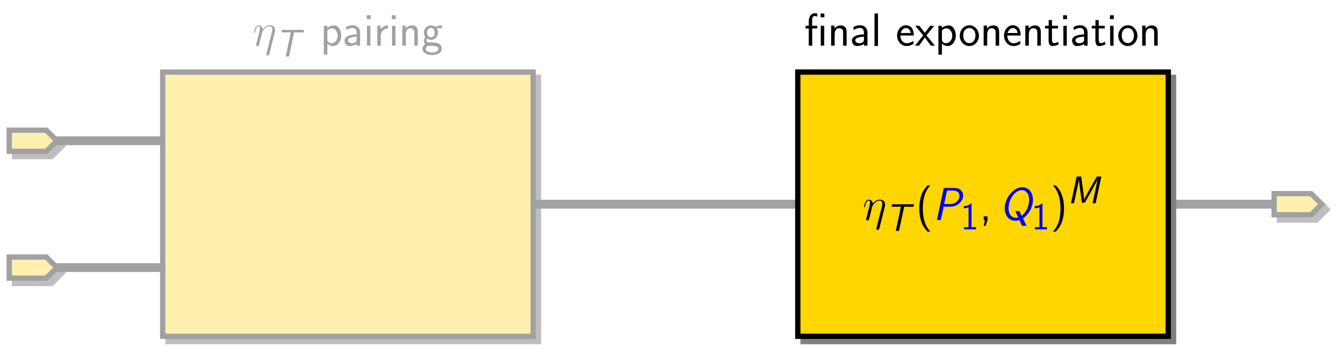
Two coprocessors for the Tate pairing

- The two operations are purely sequential



Two coprocessors for the Tate pairing

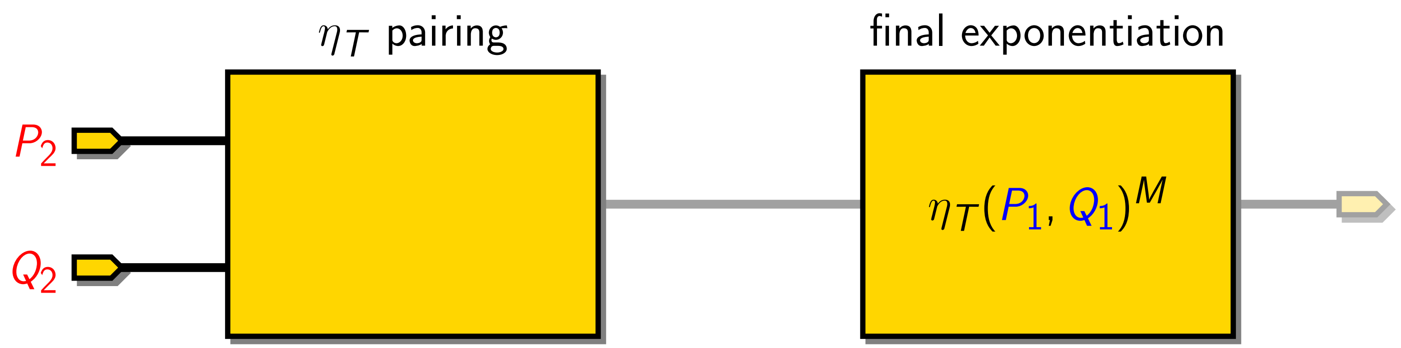
- The two operations are purely sequential



- Only one active coprocessor at every moment

Two coprocessors for the Tate pairing

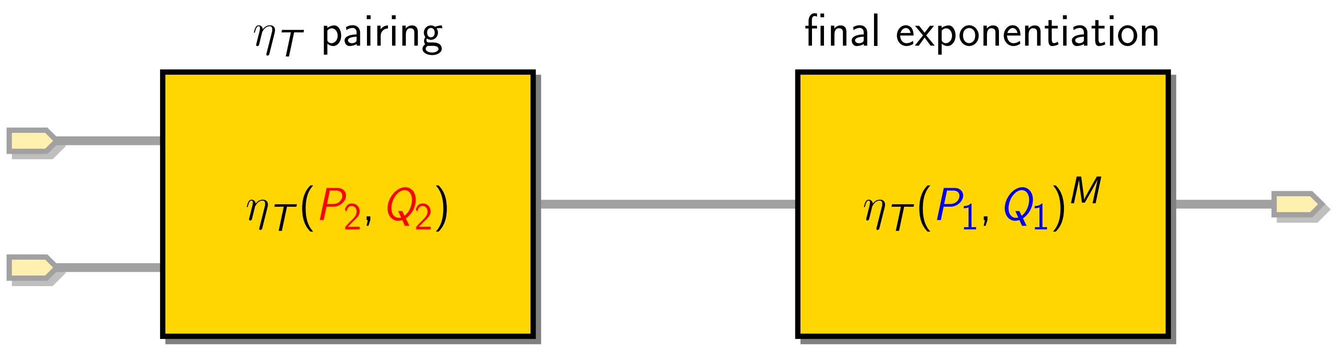
- The two operations are purely sequential



- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors

Two coprocessors for the Tate pairing

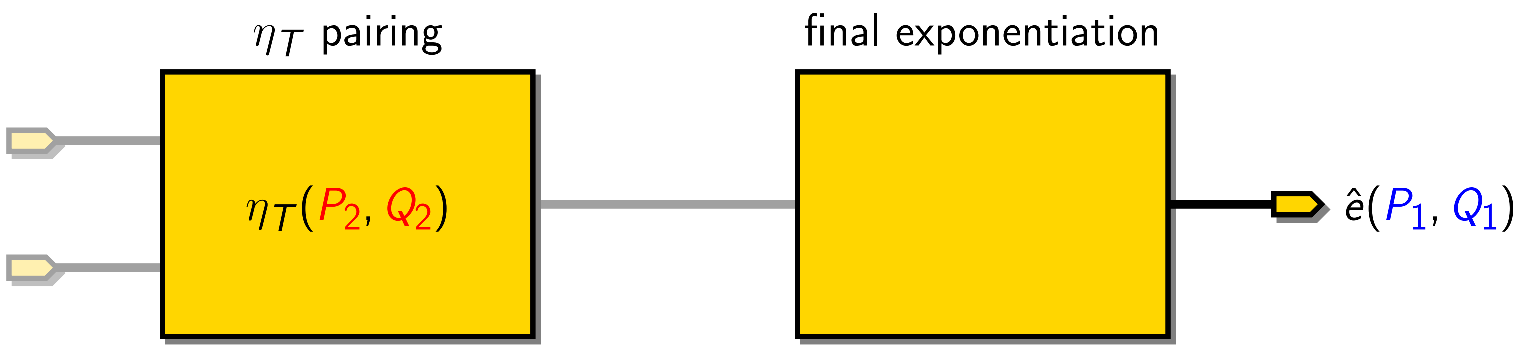
- The two operations are purely sequential



- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors

Two coprocessors for the Tate pairing

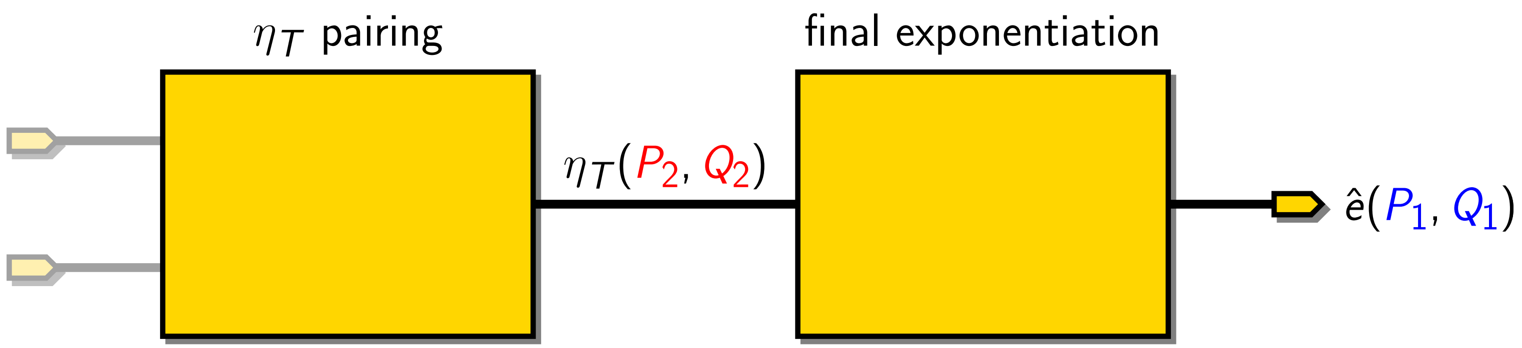
- The two operations are purely sequential



- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors

Two coprocessors for the Tate pairing

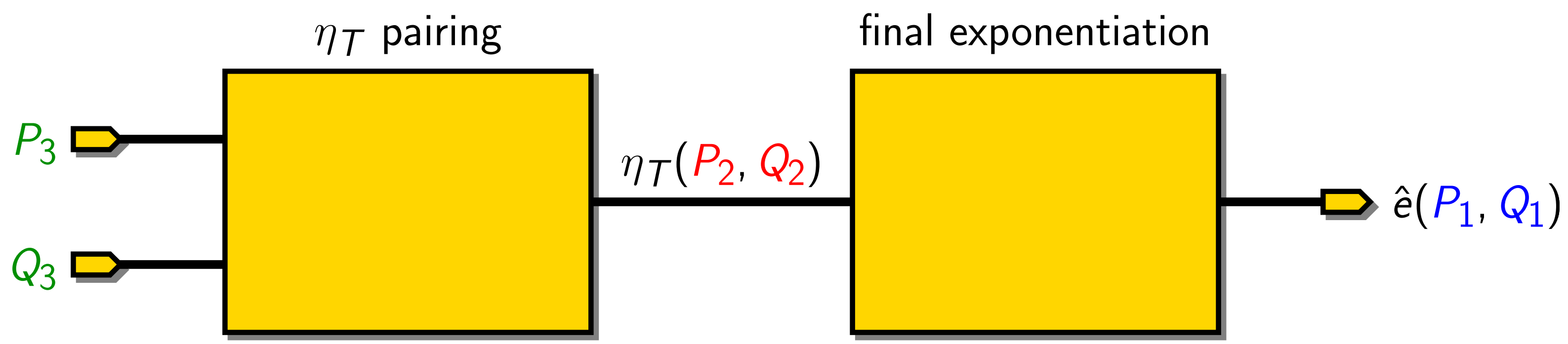
- The two operations are purely sequential



- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors

Two coprocessors for the Tate pairing

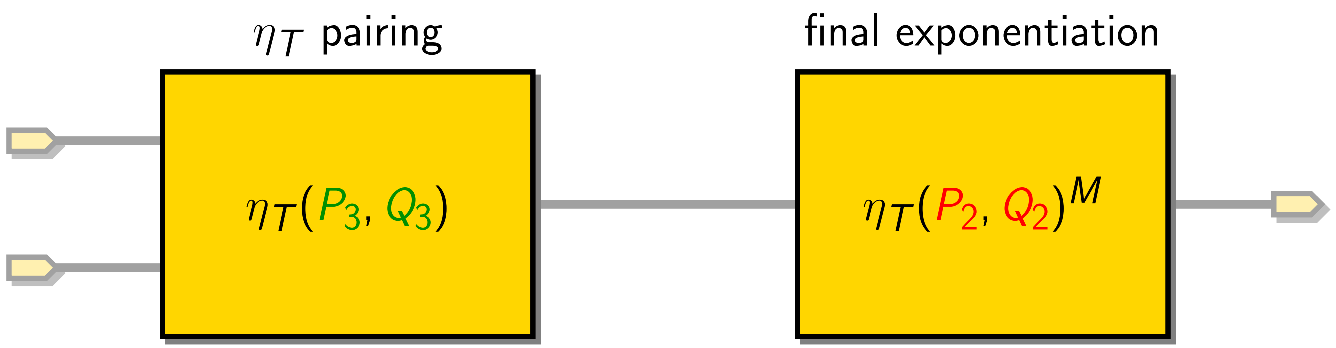
- The two operations are purely sequential



- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors

Two coprocessors for the Tate pairing

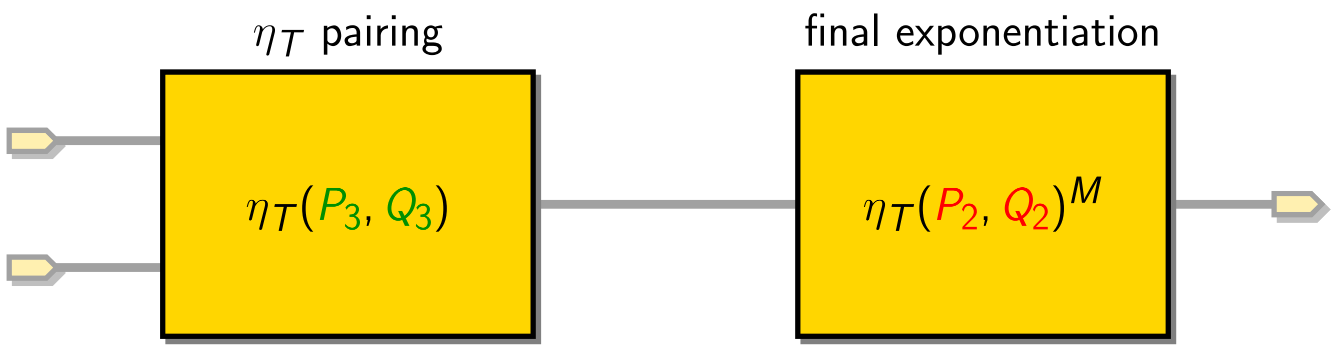
- The two operations are purely sequential



- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors
  - both of them are kept busy
  - higher throughput

Two coprocessors for the Tate pairing

- The two operations are purely sequential



- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors
  - both of them are kept busy
  - higher throughput
- Balance the computation time between the two coprocessors

Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing (in characteristic 3)
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \longrightarrow \mathbb{F}_{p^{km}}^{\times}$$

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^{\times}$$

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}^{\times}_{3^{6m}}$$

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

$x_Q \leftarrow x_Q^9 \pm 1 \;;\; y_Q \leftarrow -y_Q^9$

$t \leftarrow x_P + x_Q \;;\; u \leftarrow y_P y_Q$

$S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$

$R \leftarrow R \cdot S$

$R \leftarrow R^3$

end for

- Four tasks per iteration:

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}_{3^{6m}}^{\times}$$

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

① $x_Q \leftarrow x_Q^9 \pm 1 \; ; \; y_Q \leftarrow -y_Q^9$

$t \leftarrow x_P + x_Q \; ; \; u \leftarrow y_P y_Q$

$S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$

$R \leftarrow R \cdot S$

$R \leftarrow R^3$

end for

- Four tasks per iteration:

① update the coordinates

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}^{\times}_{3^{6m}}$$

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

① $x_Q \leftarrow x_Q^9 \pm 1 \;;\; y_Q \leftarrow -y_Q^9$

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

$R \leftarrow R \cdot S$

$R \leftarrow R^3$

end for

- Four tasks per iteration:

  ① update the coordinates
  ② compute the line equation

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}_{3^{6m}}^{\times}$$

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

① $x_Q \leftarrow x_Q^9 \pm 1 \;;\; y_Q \leftarrow -y_Q^9$

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

③ $R \leftarrow R \cdot S$

$R \leftarrow R^3$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}^{\times}_{3^{6m}}$$

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

① $x_Q \leftarrow x_Q^9 \pm 1 \;;\; y_Q \leftarrow -y_Q^9$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^3$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

① $x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$ 4 Frobenius, $2+$ $(\mathbb{F}_{3^m})$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^3$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}^{\times}_{3^{6m}}$$

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

①	$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$	4 Frobenius, 2 + $(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$	2 ×, 1 + $(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	
④	$R \leftarrow R^3$	

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

①	$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$	4 Frobenius, 2 $+$	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$	2 $\times$, 1 $+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	1 $\times$	$(\mathbb{F}_{3^{6m}})$
④	$R \leftarrow R^3$		

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

①	$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$	4 Frobenius, $2+$	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$1\times$	$(\mathbb{F}_{3^{6m}})$
④	$R \leftarrow R^3$	1 Frobenius	$(\mathbb{F}_{3^{6m}})$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

①	$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$	4 Frobenius, $2+$	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$1\times$	$(\mathbb{F}_{3^{6m}})$
④	$R \leftarrow R^3$	6 Frobenius, $6+$	$(\mathbb{F}_{3^m})$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

①	$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$	4 Frobenius, 2 $+$	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$	2 $\times$, 1 $+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	12 $\times$, 59 $+$	$(\mathbb{F}_{3^m})$
④	$R \leftarrow R^3$	6 Frobenius, 6 $+$	$(\mathbb{F}_{3^m})$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}_{3^{6m}}^{\times}$$

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

①	$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$	4 Frobenius, 2 +	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$	2 ×, 1 +	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	15 ×, 29 +	$(\mathbb{F}_{3^m})$
④	$R \leftarrow R^3$	6 Frobenius, 6 +	$(\mathbb{F}_{3^m})$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}^{\times}_{3^{6m}}$$

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

①	$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$	4 Frobenius, 2 $+$	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$	2 $\times$, 1 $+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	15 $\times$, 29 $+$	$(\mathbb{F}_{3^m})$
④	$R \leftarrow R^3$	6 Frobenius, 6 $+$	$(\mathbb{F}_{3^m})$

end for

- Four tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
  - ④ cube the partial product
- Total cost: 17 $\times$, 10 Frobenius and 38 $+$ over $\mathbb{F}_{3^m}$

Accelerating the η_T pairing

- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration

Accelerating the η_T pairing

- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$

Accelerating the η_T pairing

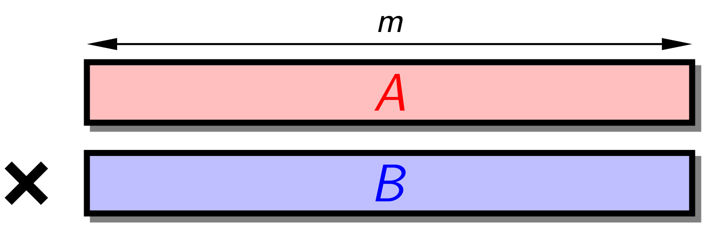
- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier

Accelerating the η_T pairing

- Total cost: $17 \times$, 10 Frobenius and $38 +$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba-Ofman

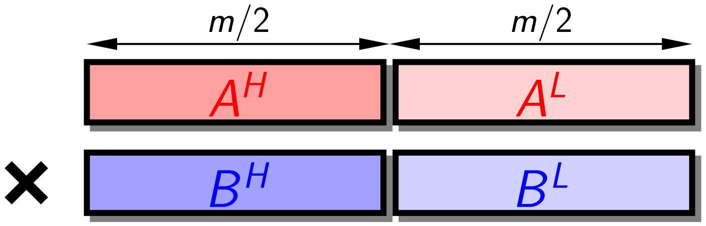
Accelerating the η_T pairing

- Total cost: $17\,\times$, 10 Frobenius and $38\,+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba-Ofman



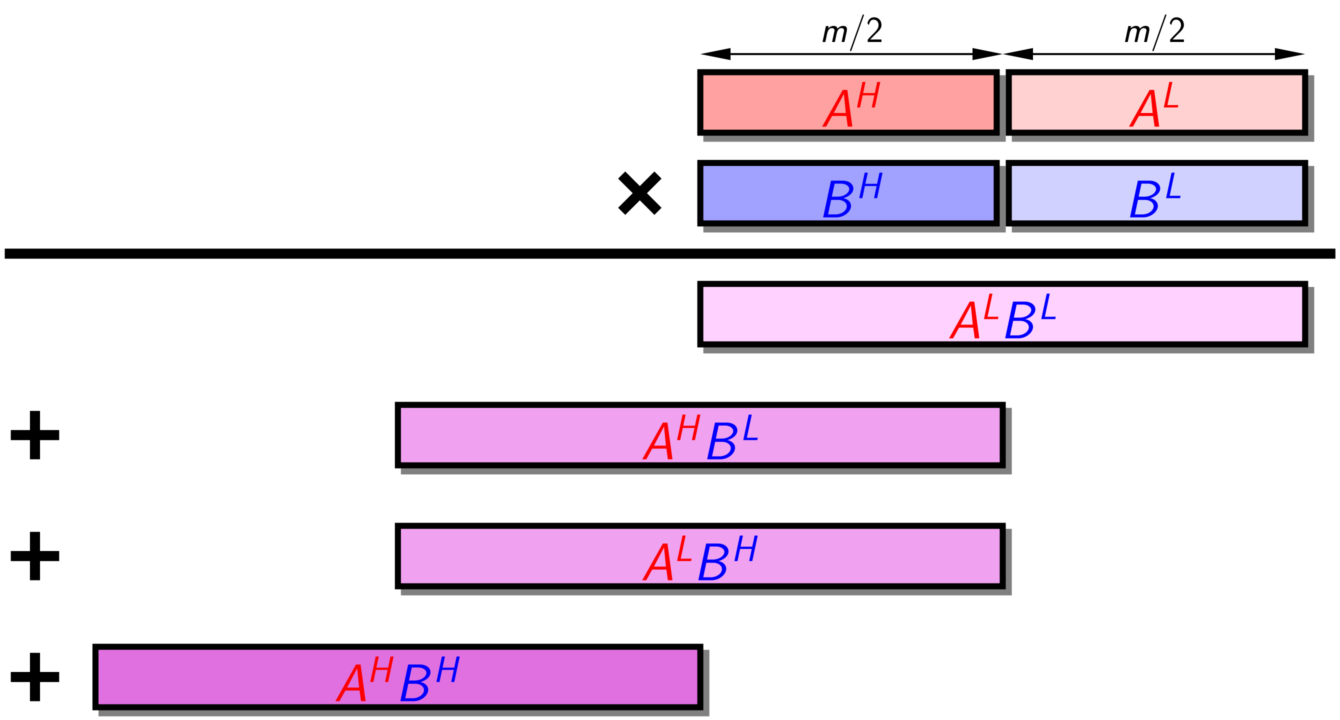
Accelerating the η_T pairing

- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba-Ofman



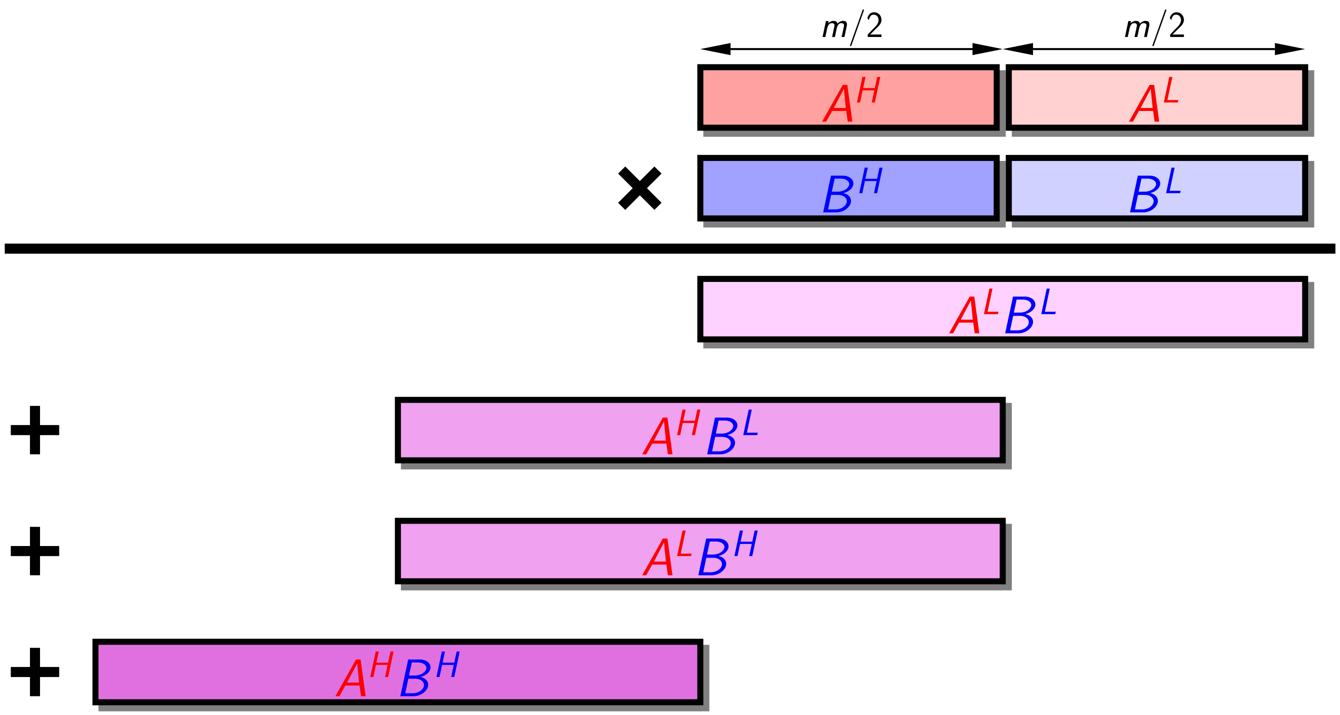
Accelerating the η_T pairing

- Total cost: 17 $\times$, 10 Frobenius and 38 $+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba-Ofman



Accelerating the η_T pairing

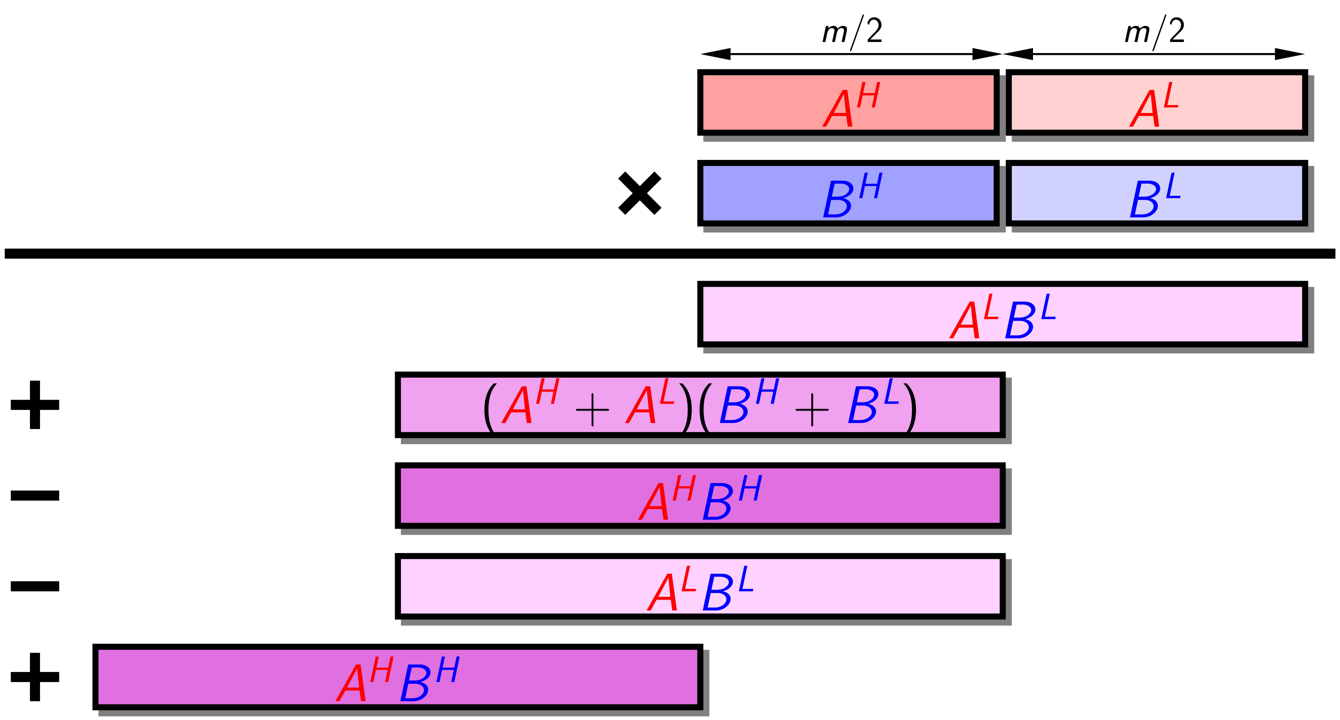
- Total cost: 17 $\times$, 10 Frobenius and 38 $+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba-Ofman



$$A^H B^L + A^L B^H = (A^H + A^L)(B^H + B^L) - A^H B^H - A^L B^L$$

Accelerating the η_T pairing

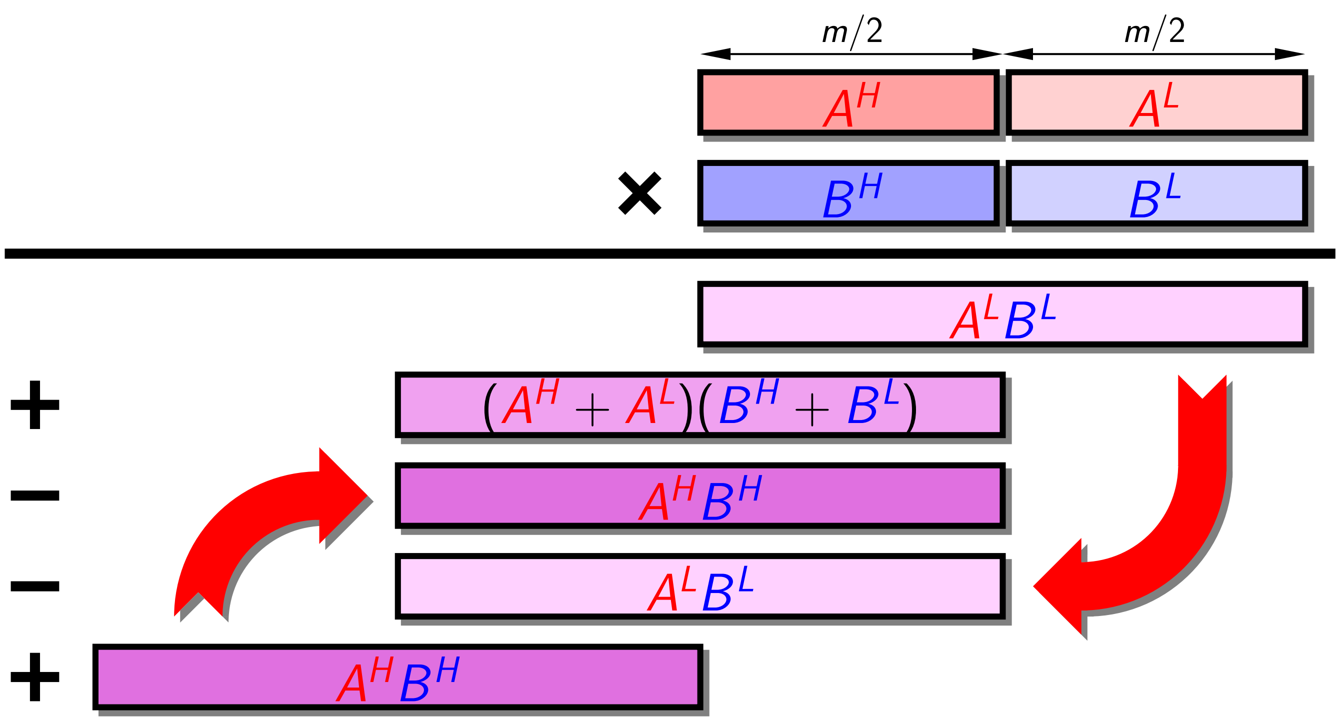
- Total cost: $17 \times$, 10 Frobenius and $38 +$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba-Ofman



$$A^H B^L + A^L B^H = (A^H + A^L)(B^H + B^L) - A^H B^H - A^L B^L$$

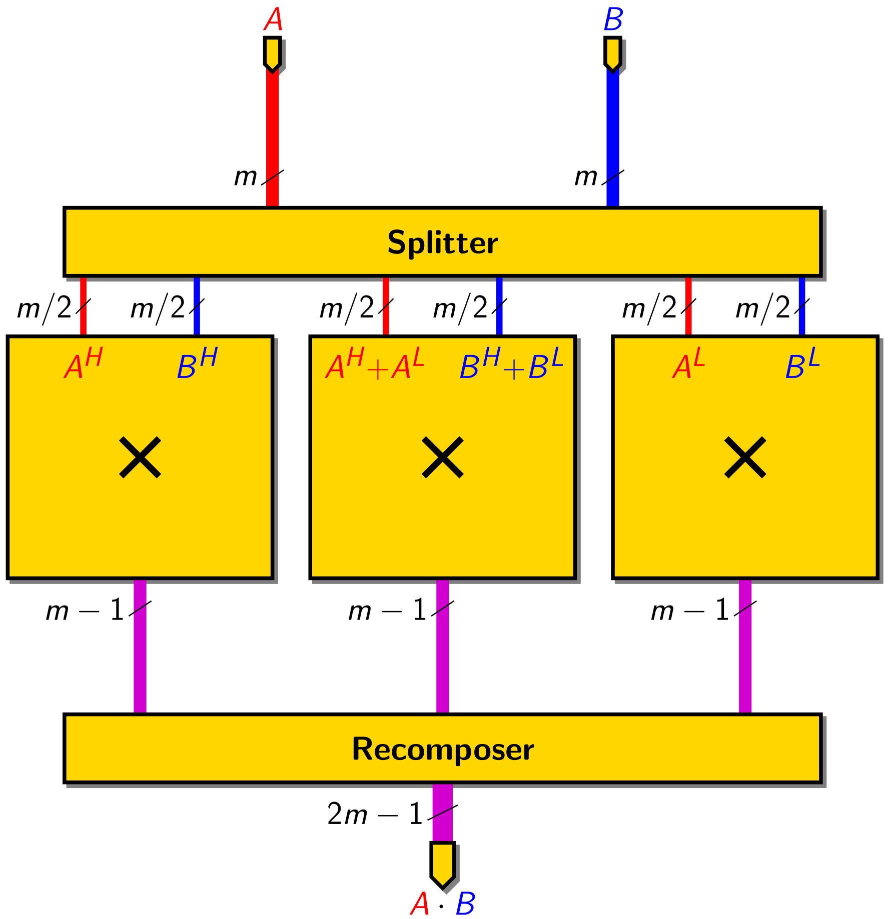
Accelerating the η_T pairing

- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba-Ofman



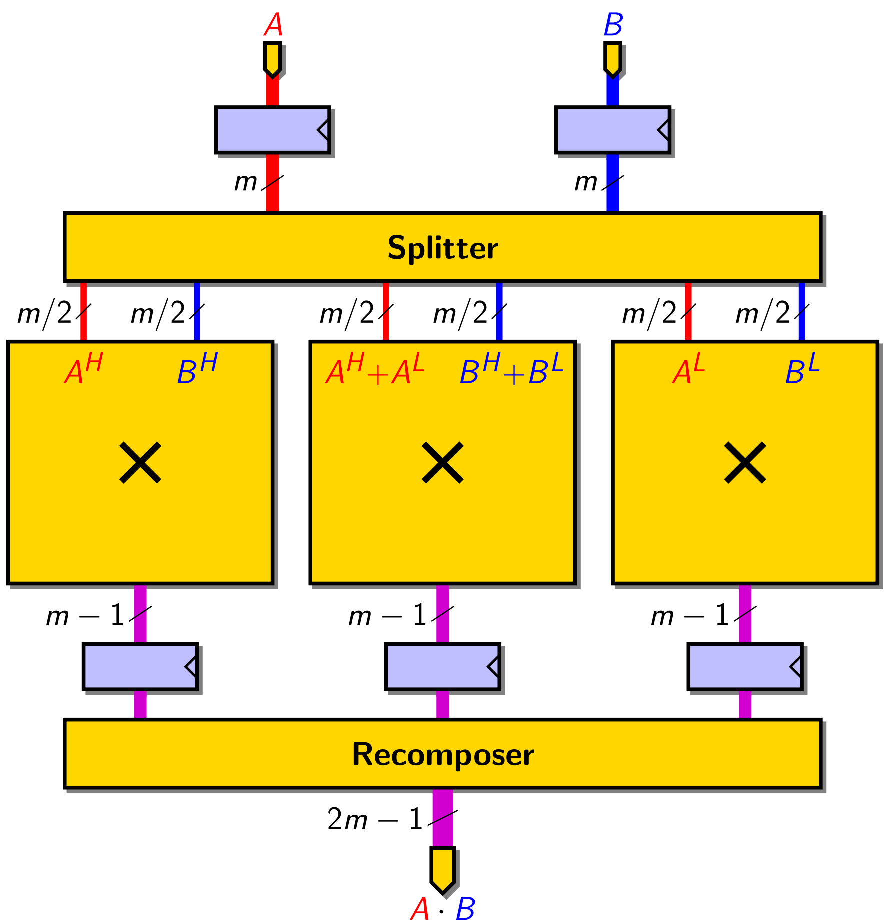
$$A^H B^L + A^L B^H = (A^H + A^L)(B^H + B^L) - A^H B^H - A^L B^L$$

A parallel Karatsuba-Ofman multiplier



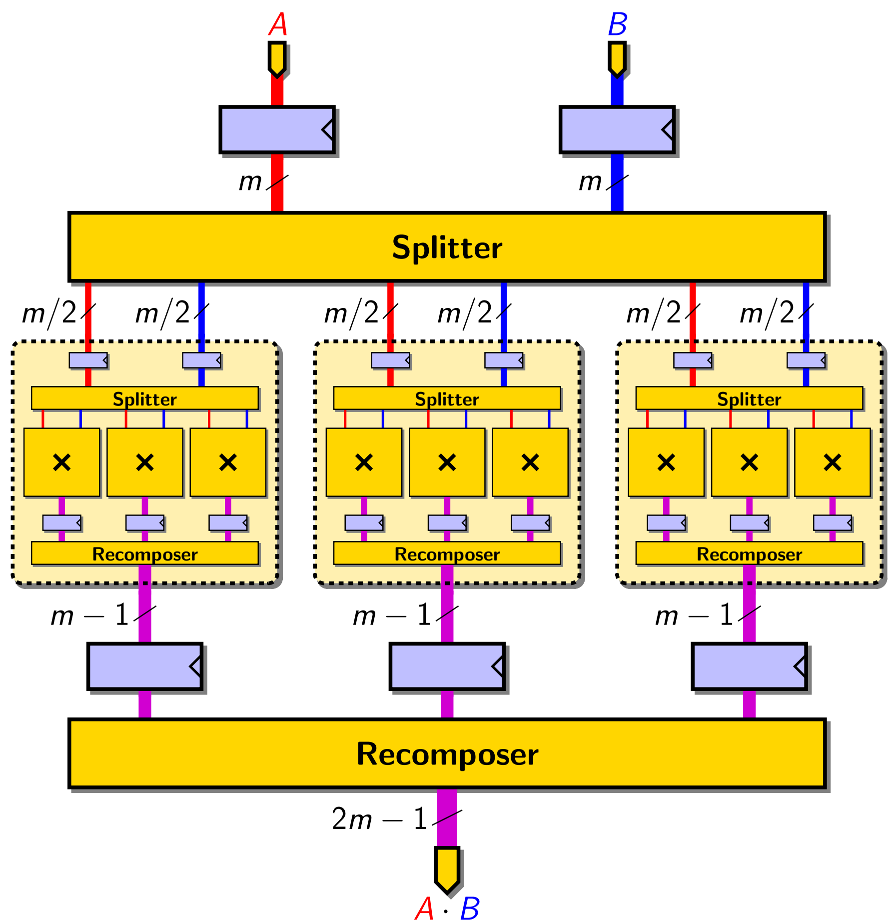
- fully parallel: all sub-products are computed in parallel

A parallel Karatsuba-Ofman multiplier



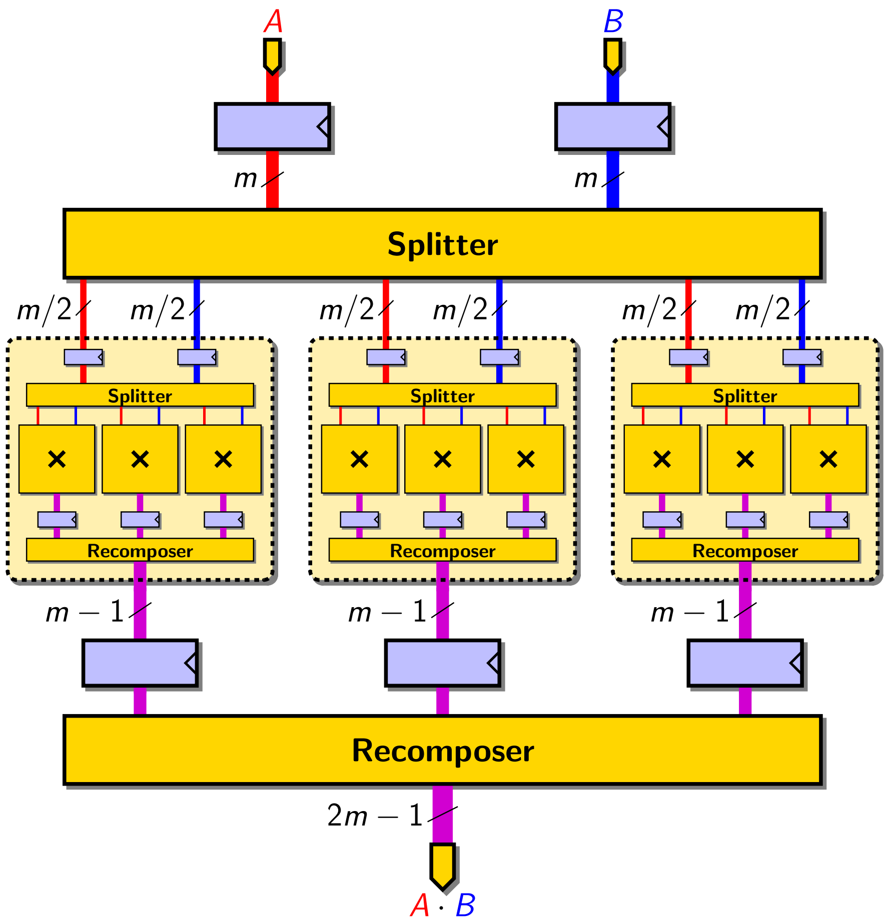
- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle

A parallel Karatsuba-Ofman multiplier



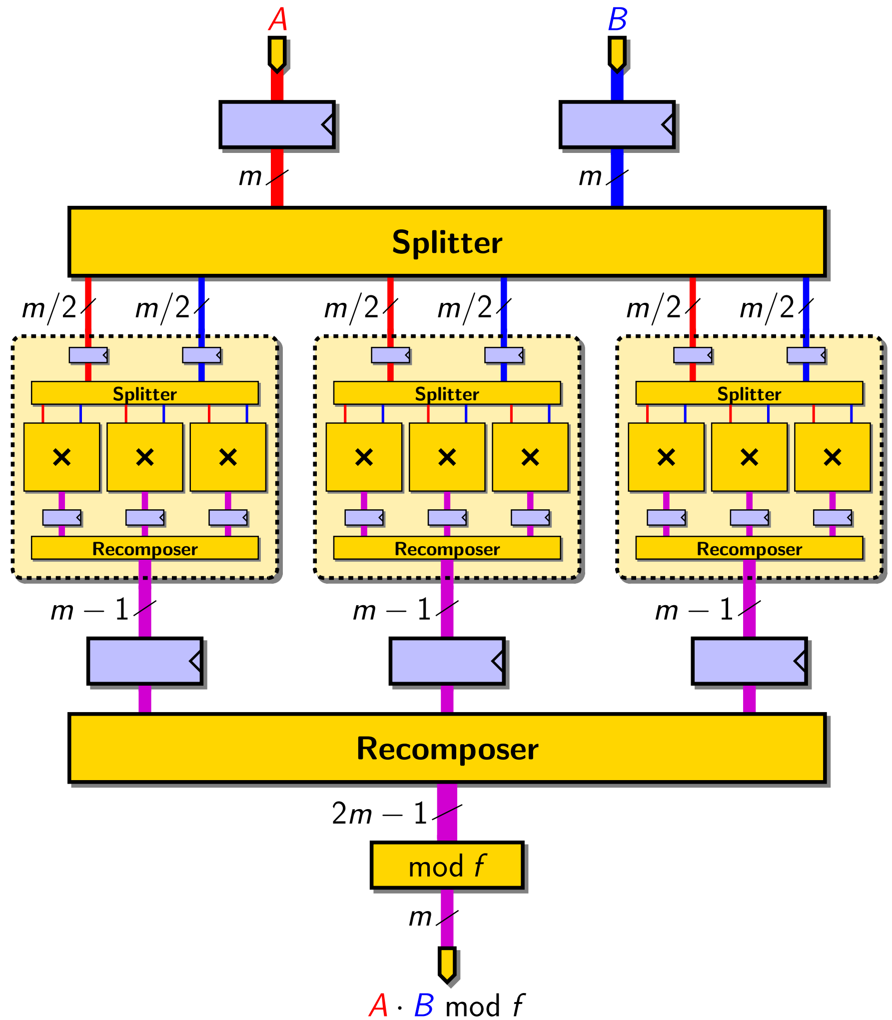
- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle
- sub-products recursively implemented as Karatsuba-Ofman multipliers

A parallel Karatsuba-Ofman multiplier



- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle
- sub-products recursively implemented as Karatsuba-Ofman multipliers
- support for other variants: odd-even split, 3-way split, ...

A parallel Karatsuba-Ofman multiplier



- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle
- sub-products recursively implemented as Karatsuba-Ofman multipliers
- support for other variants: odd-even split, 3-way split, ...
- final reduction modulo the irreducible polynomial f

Accelerating the η_T pairing

- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration
- η_T coprocessor based on a single large multiplier:
  - parallel Karatsuba-Ofman architecture
  - 7-stage pipeline
  - one product per cycle

Accelerating the η_T pairing

- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration
- η_T coprocessor based on a single large multiplier:
  - parallel Karatsuba-Ofman architecture
  - 7-stage pipeline
  - one product per cycle
- Challenge: keep the multiplier busy at all times

Accelerating the η_T pairing

- Total cost: $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration
- η_T coprocessor based on a single large multiplier:
  - parallel Karatsuba-Ofman architecture
  - 7-stage pipeline
  - one product per cycle
- Challenge: keep the multiplier busy at all times
- Careful scheduling to avoid pipeline bubbles (idle cycles):
  - ensure that multiplication operands are always available
  - avoid memory congestion issues

Data dependencies

① $x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^3$

Data dependencies

① $x_Q \leftarrow x_Q^9 \pm 1 \; ; \; y_Q \leftarrow -y_Q^9$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^3$

Data dependencies

① $x_Q \leftarrow x_Q^9 \pm 1 \;;\; y_Q \leftarrow -y_Q^9$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^3$

Data dependencies

① $x_Q \leftarrow x_Q^9 \pm 1 \;;\; y_Q \leftarrow -y_Q^9$

↓

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

↓

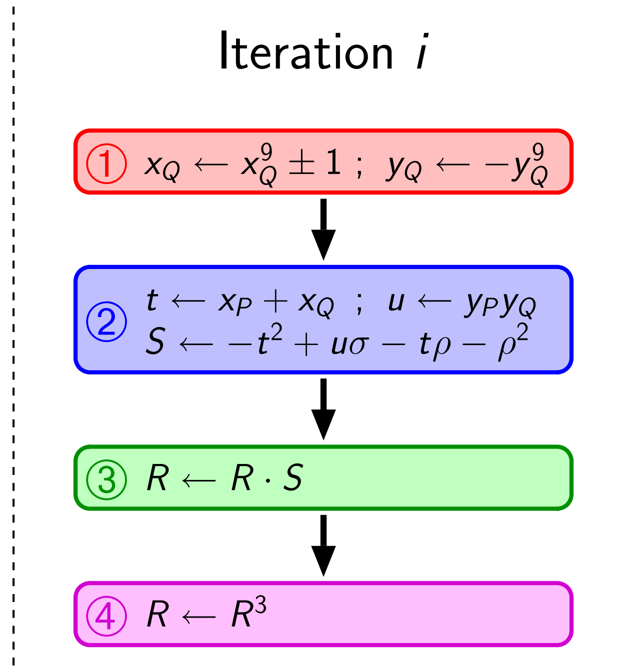
③ $R \leftarrow R \cdot S$

↓

④ $R \leftarrow R^3$

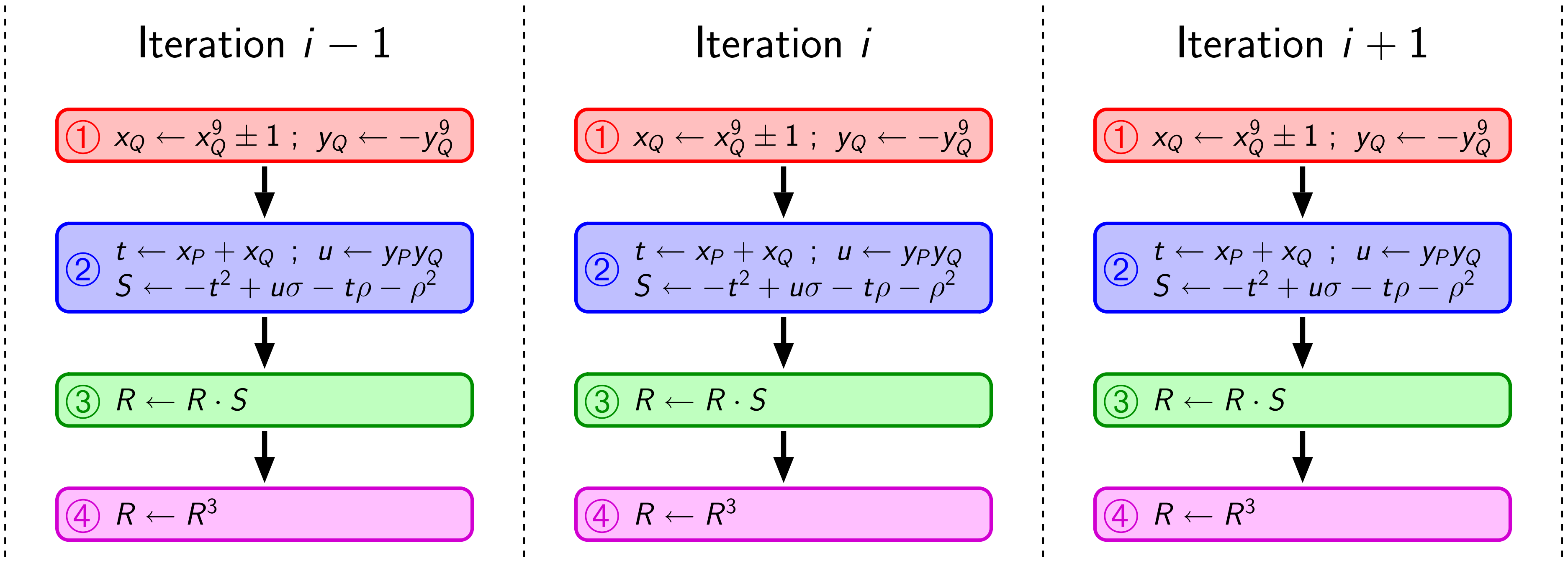
- Sequential dependencies between the tasks

Data dependencies



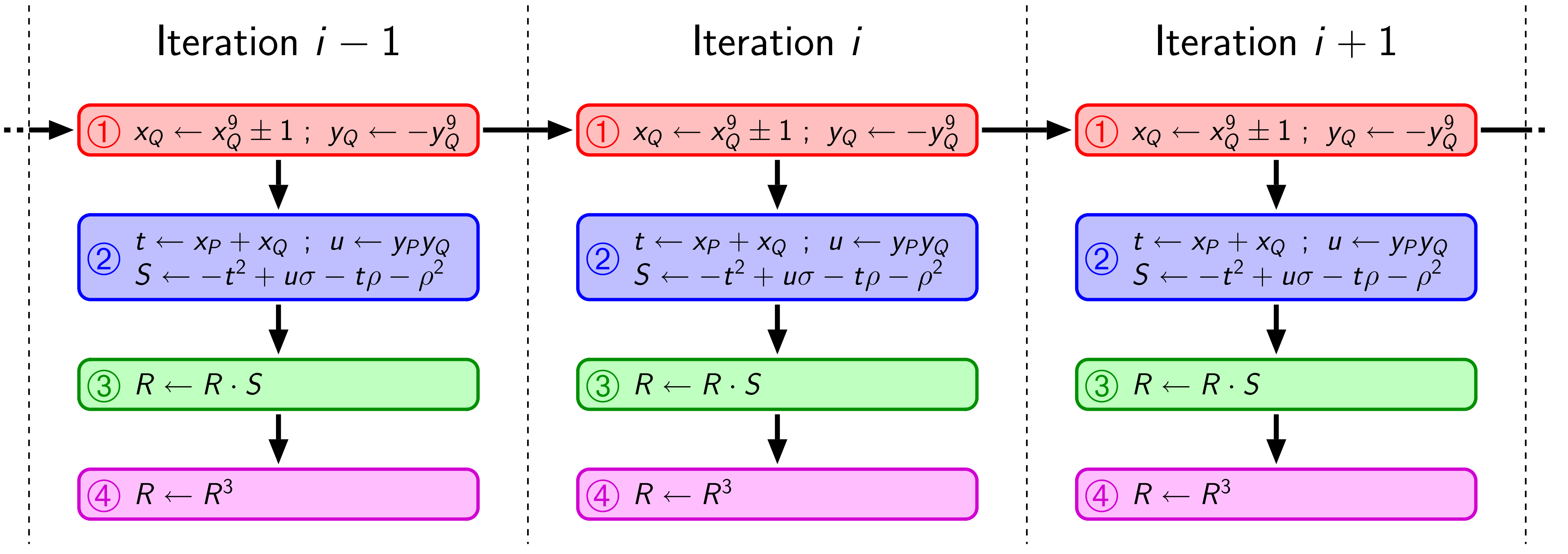
- Sequential dependencies between the tasks in each iteration

Data dependencies



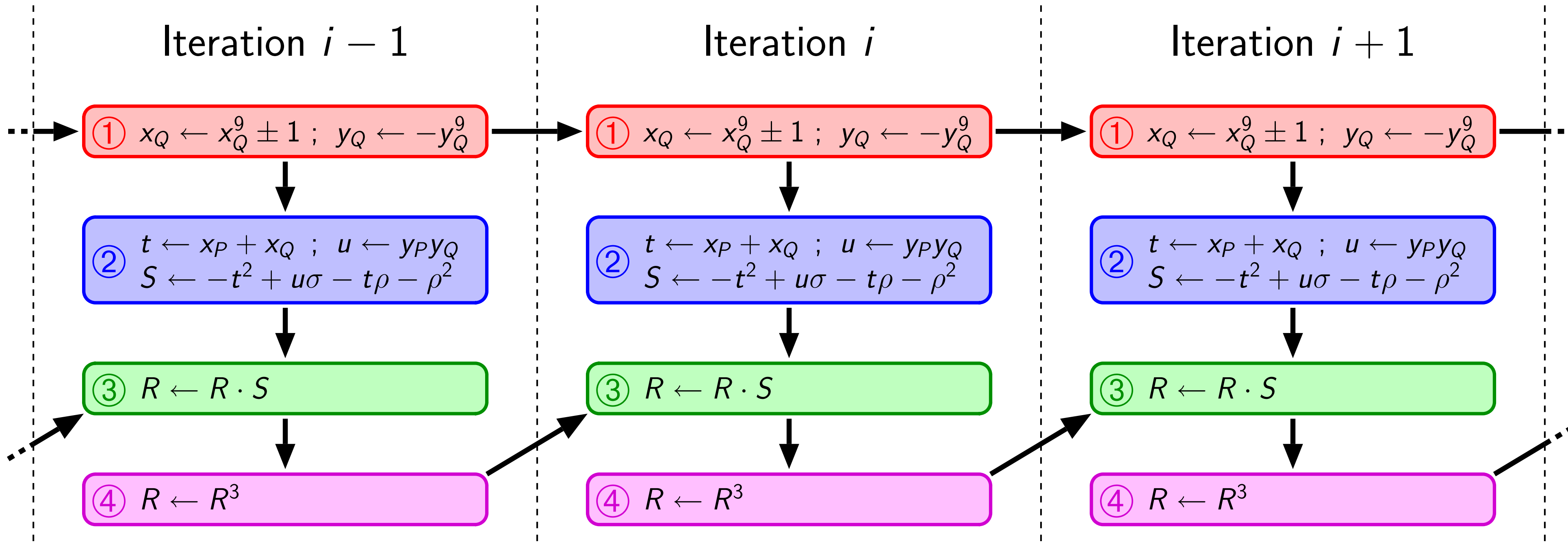
- Sequential dependencies between the tasks in each iteration

Data dependencies



- Sequential dependencies between the tasks in each iteration
- Dependencies between consecutive iterations

Data dependencies



- Sequential dependencies between the tasks in each iteration
- Dependencies between consecutive iterations

Task scheduling

Task scheduling

① $x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$

Task scheduling

① 4 Frobenius, 2 +

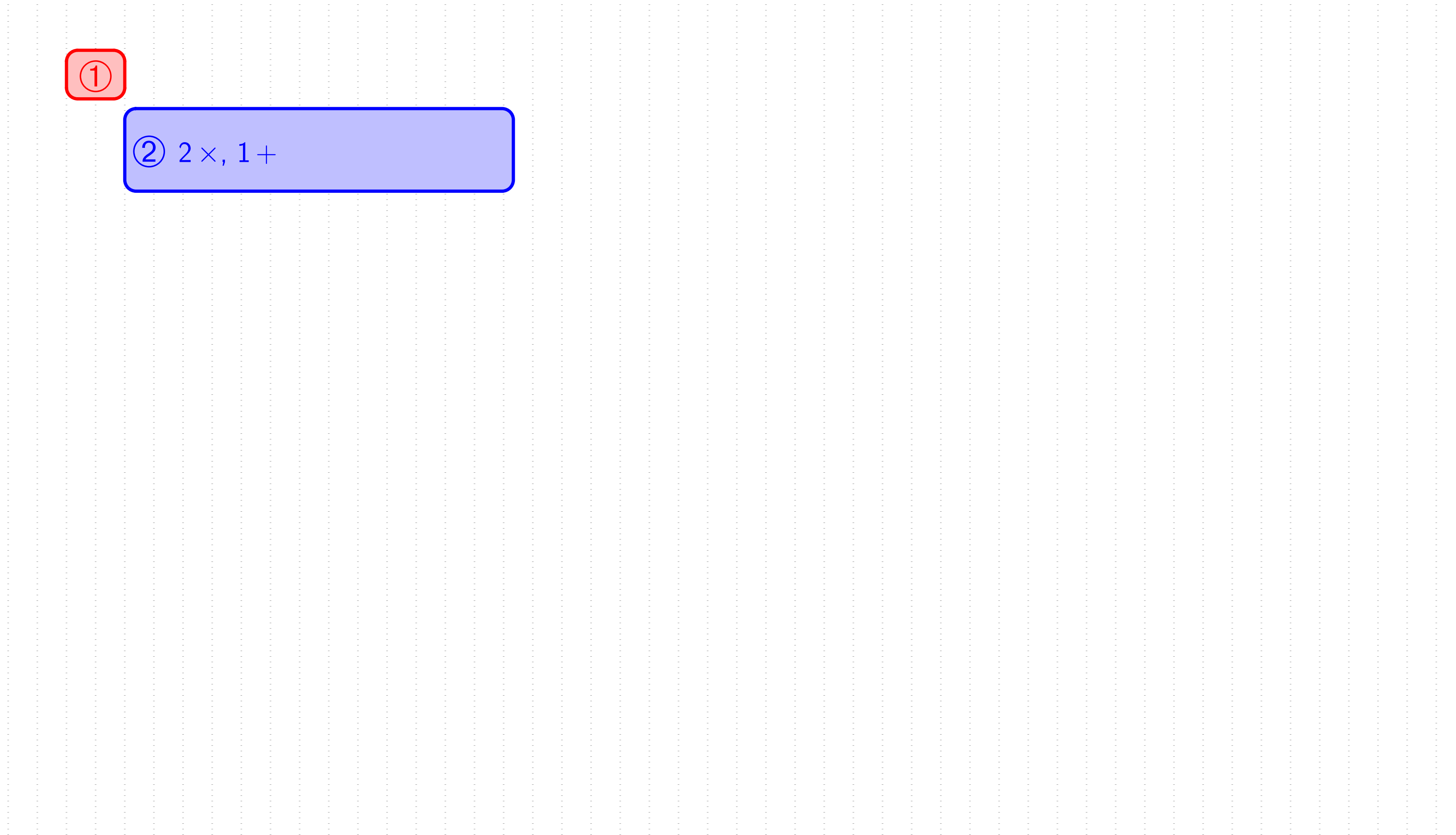
Task scheduling

Task scheduling

①

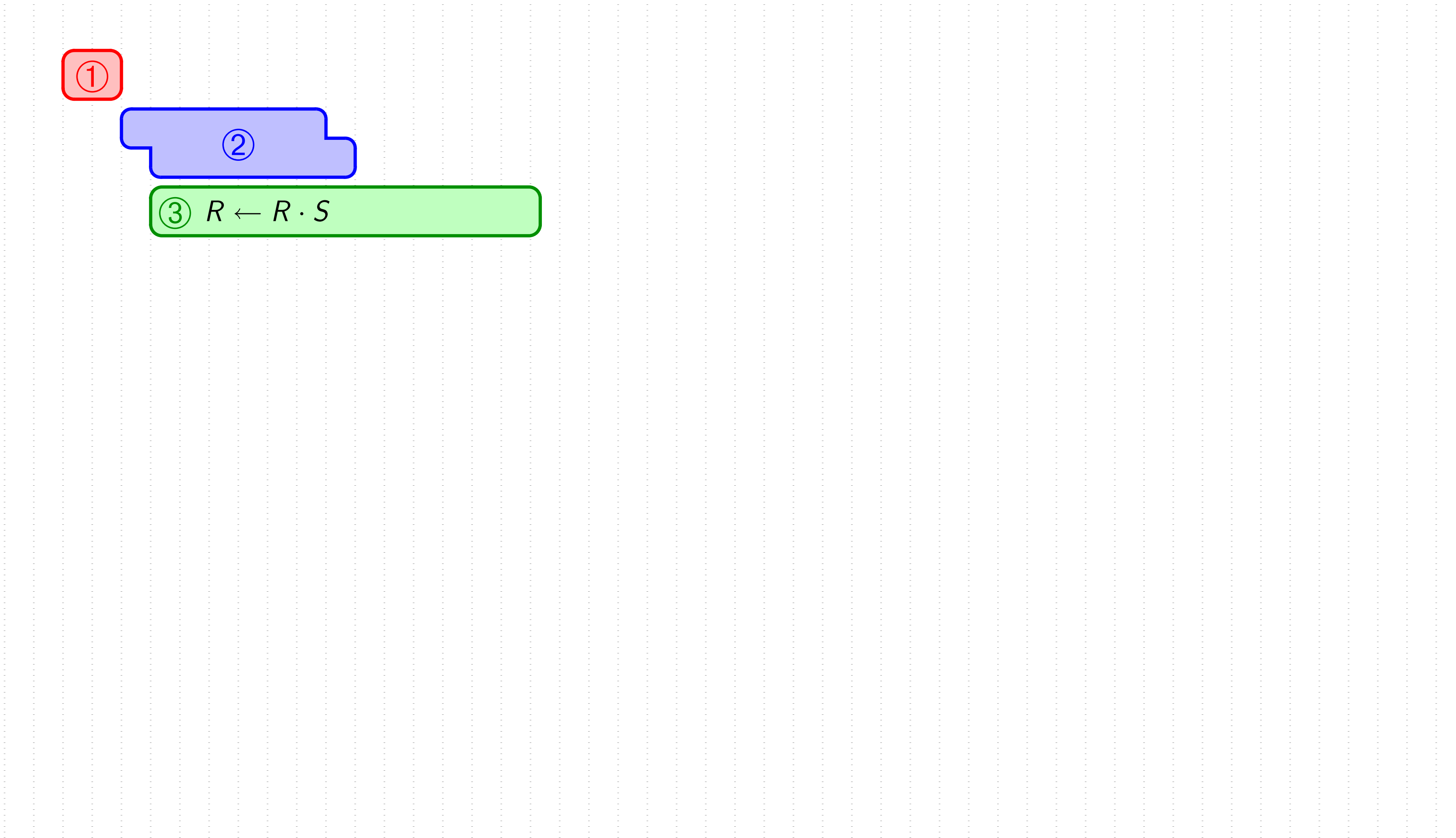
② $t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$
$S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$

Task scheduling

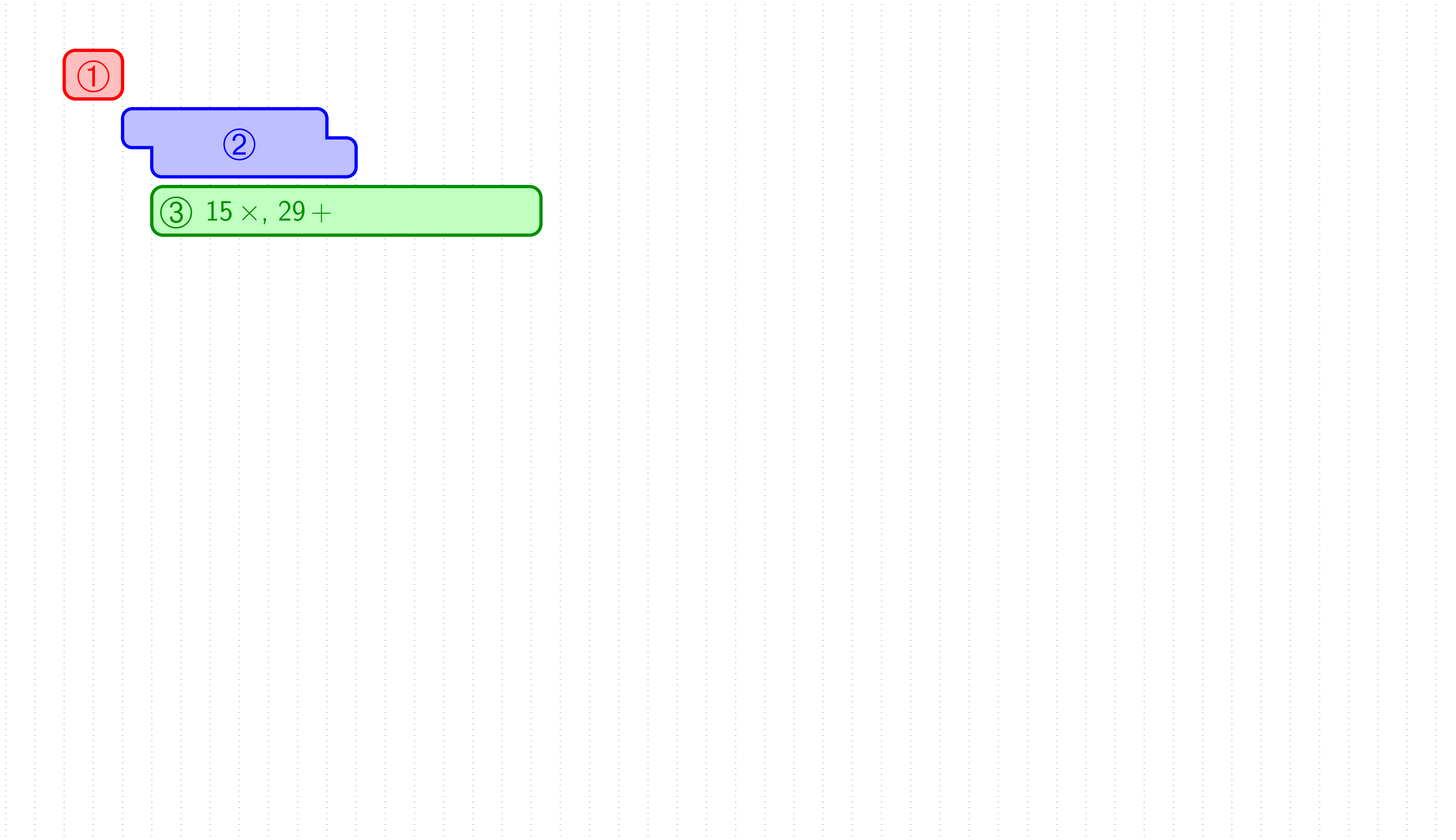


Task scheduling

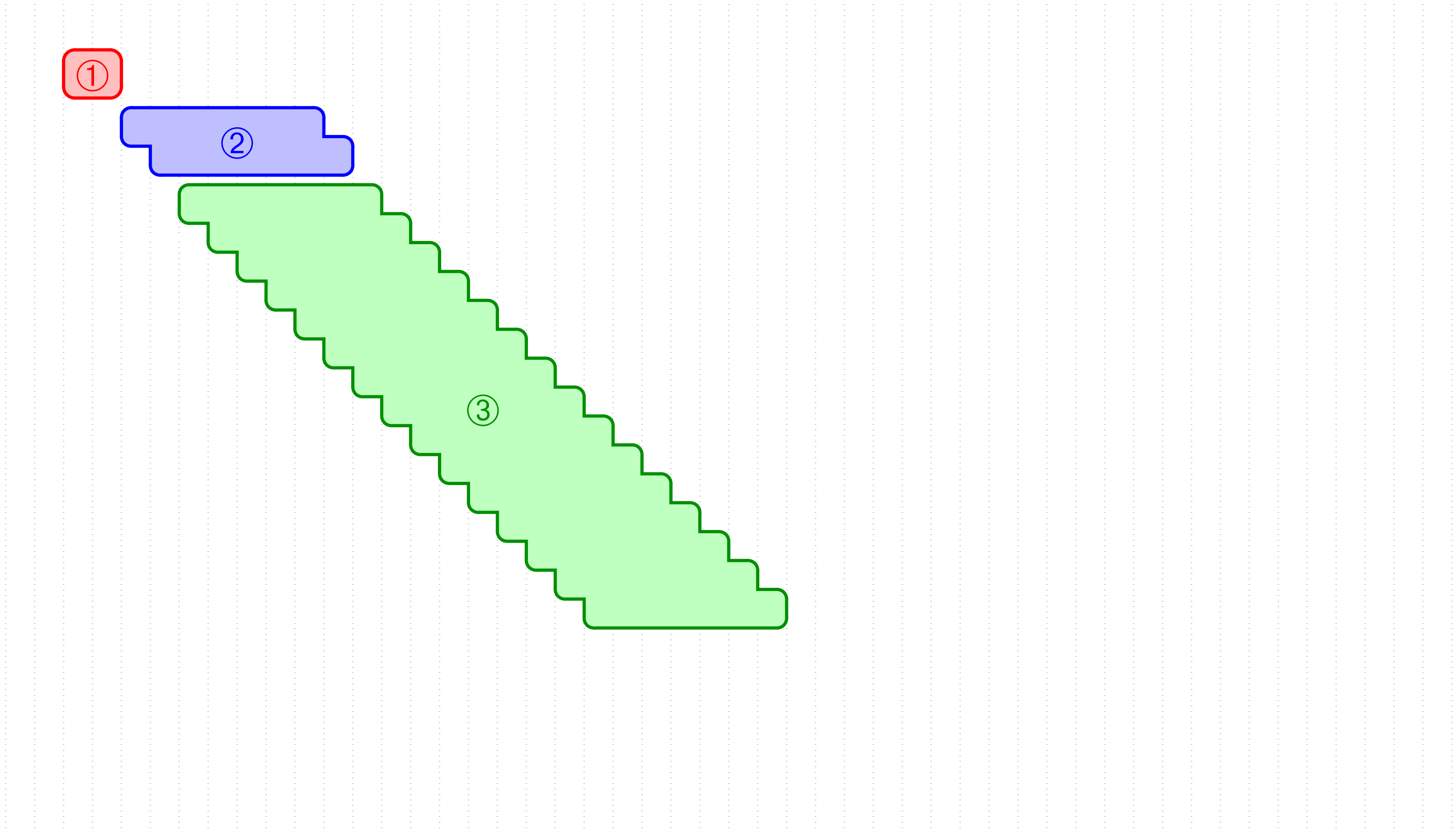
Task scheduling



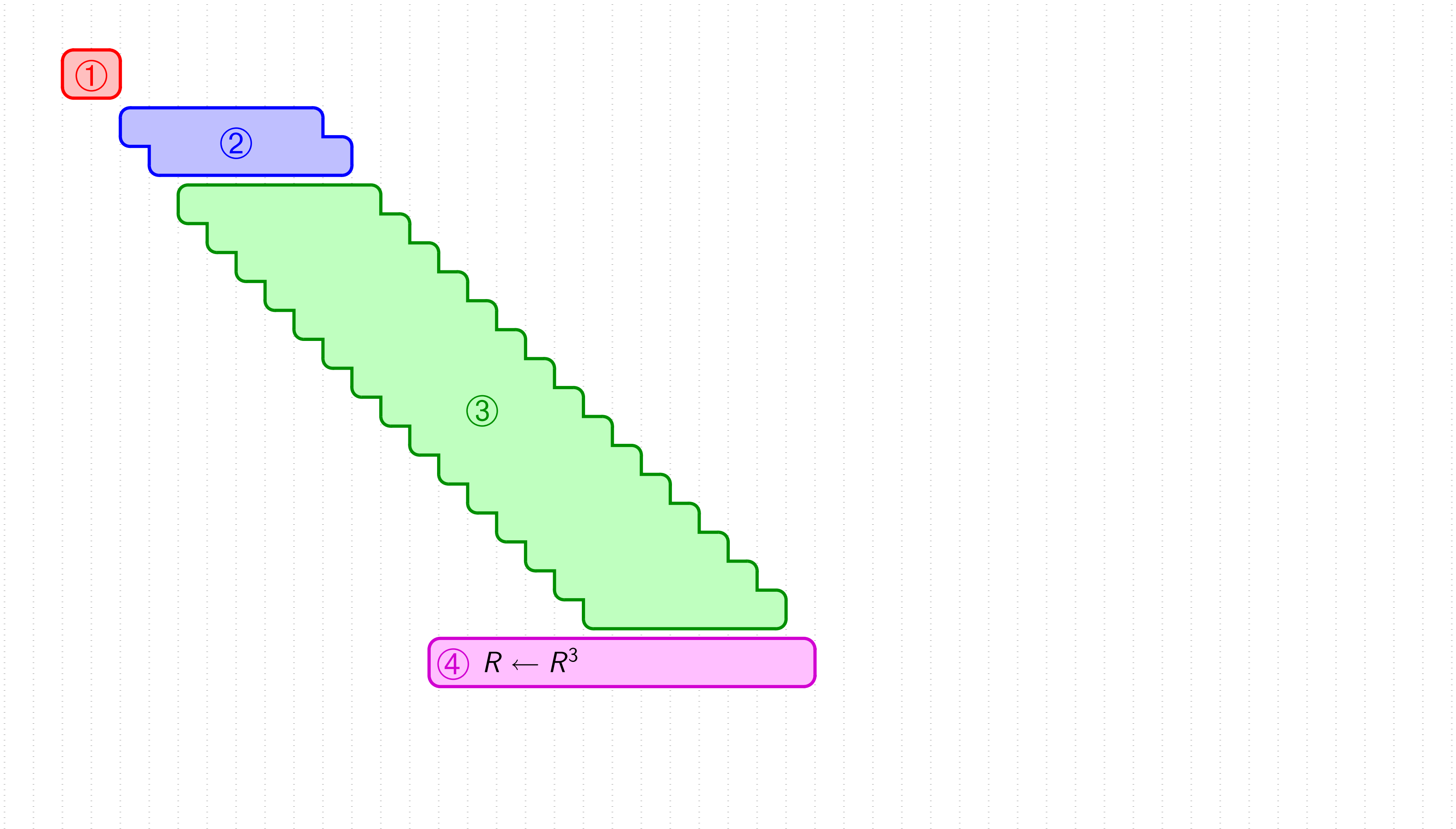
Task scheduling



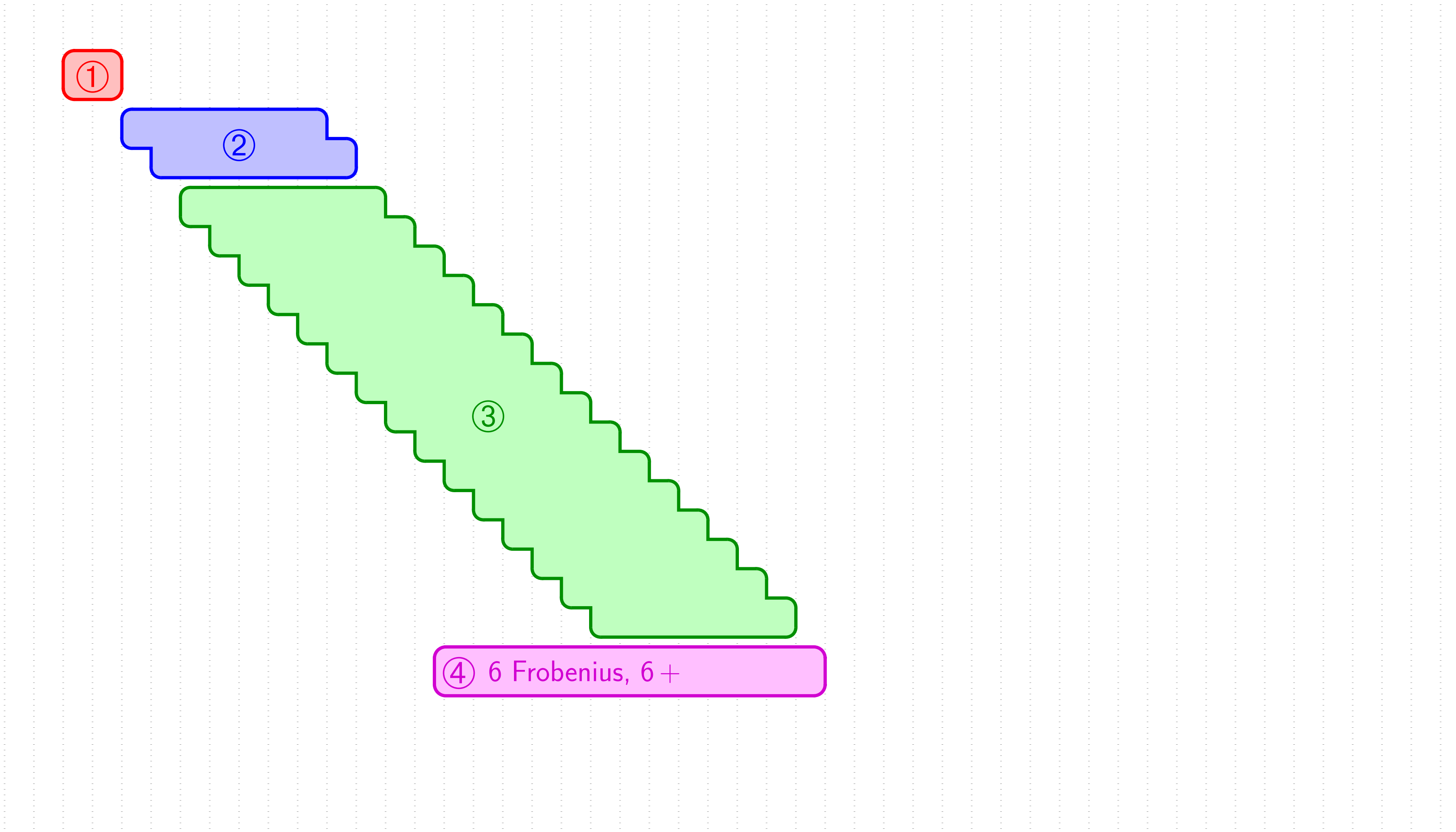
Task scheduling



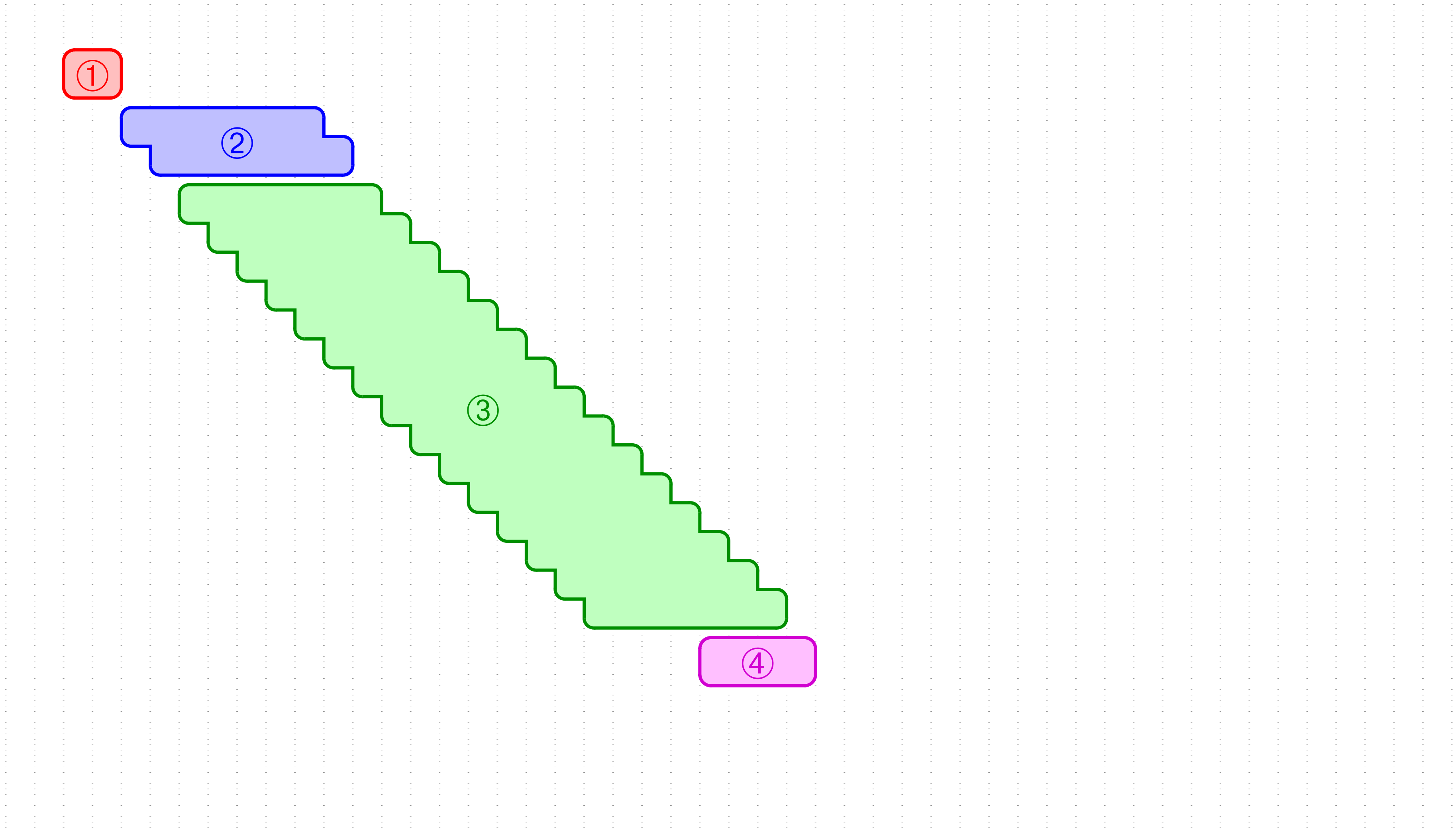
Task scheduling



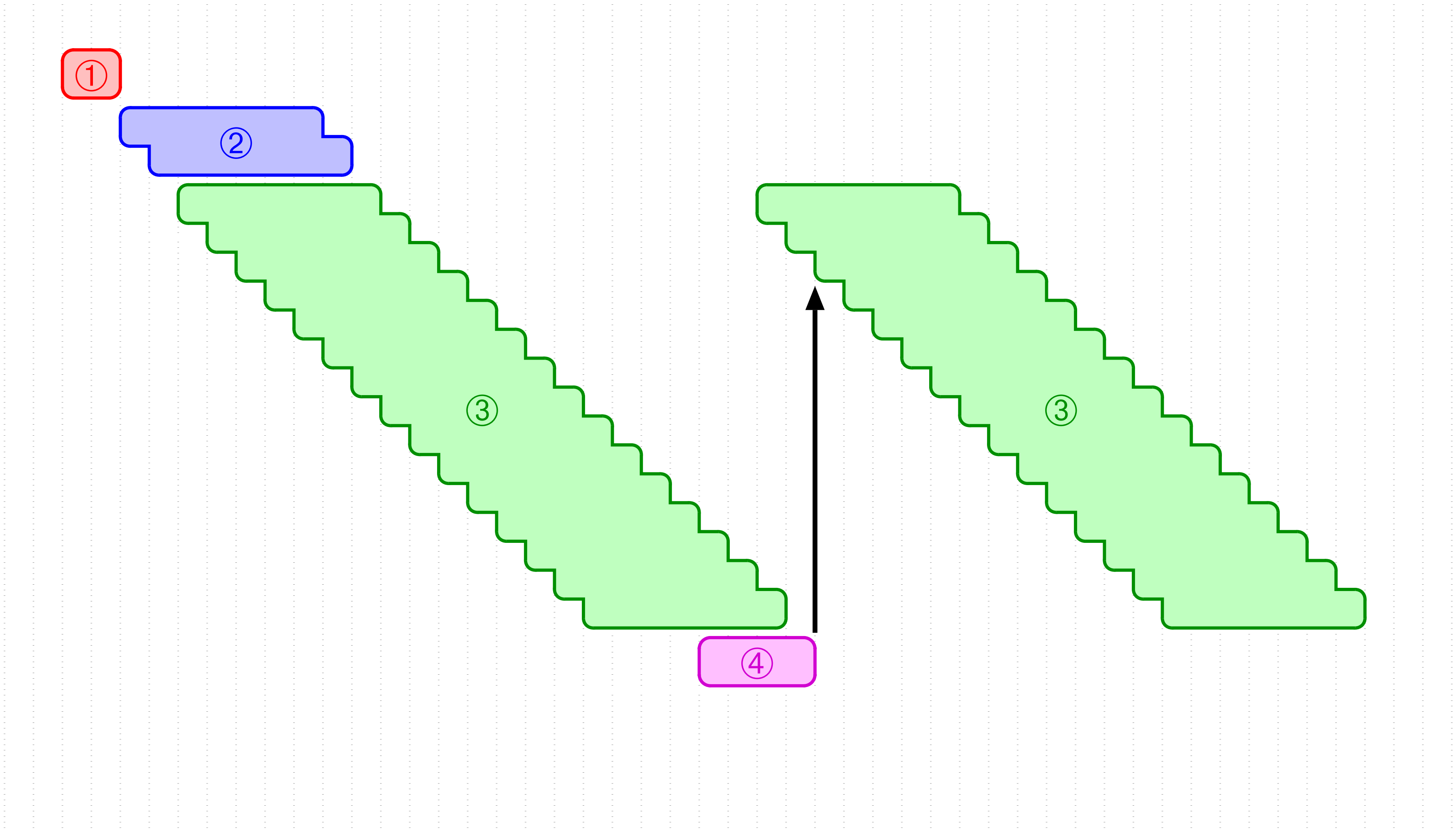
Task scheduling



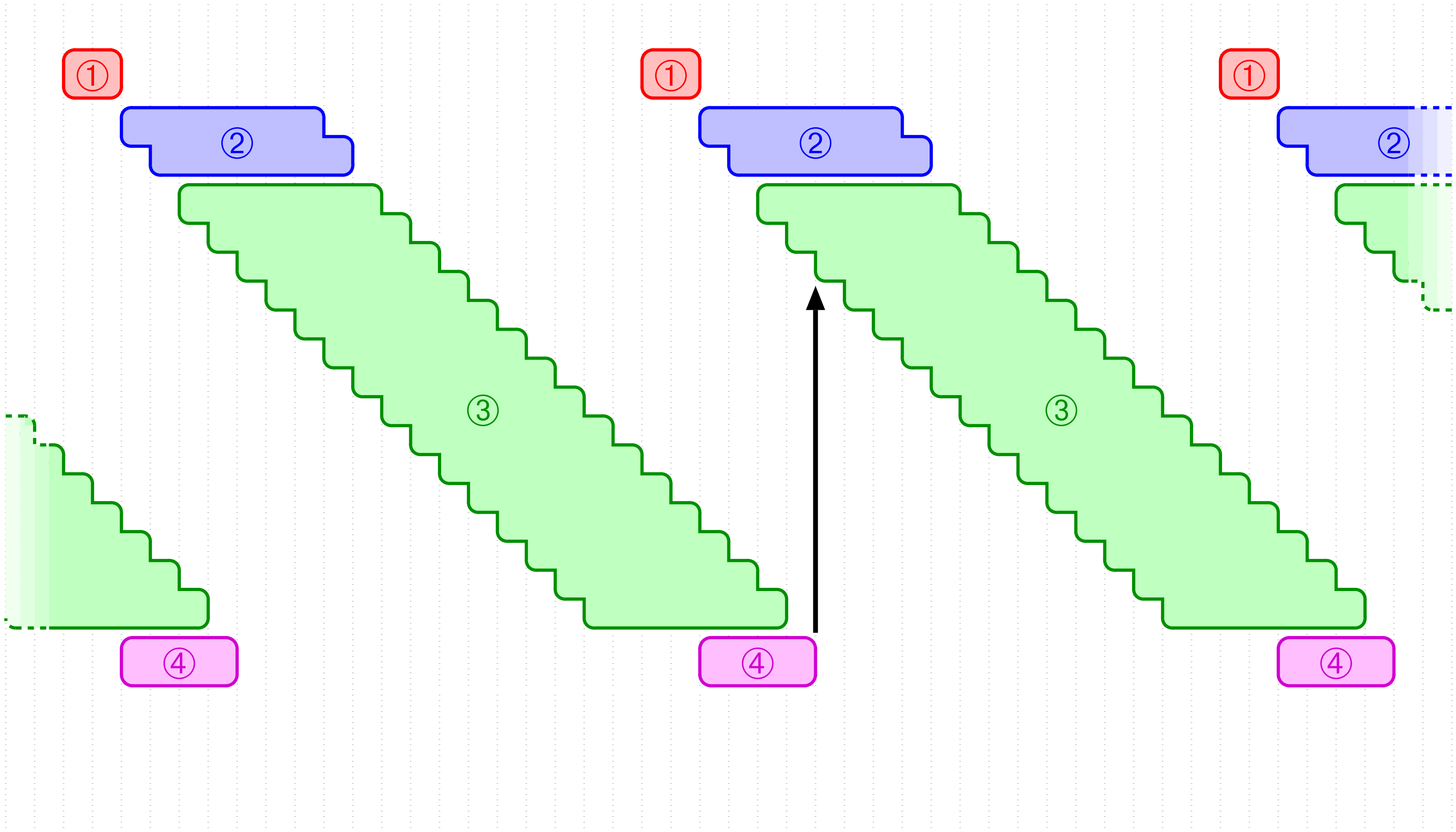
Task scheduling



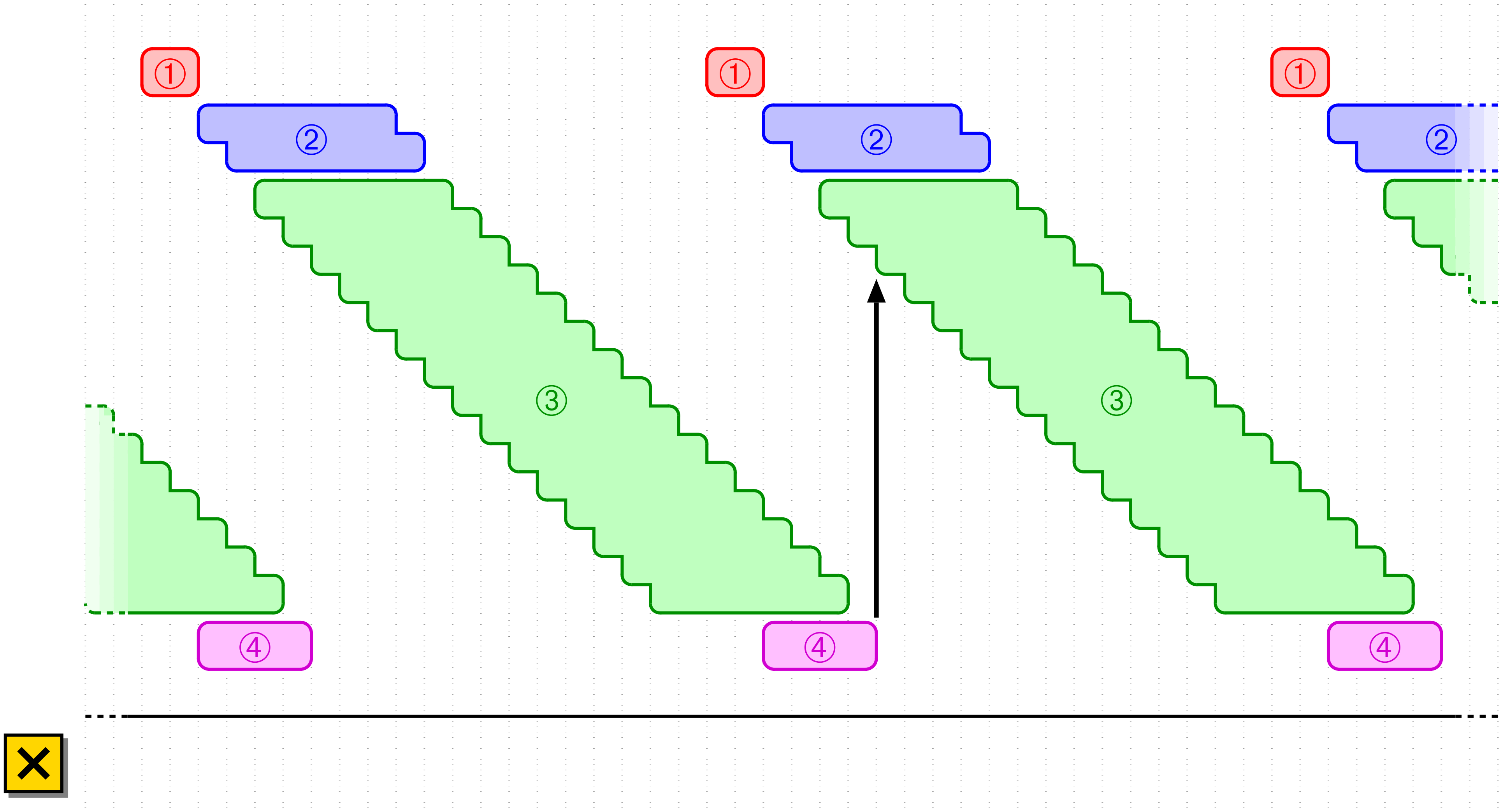
Task scheduling



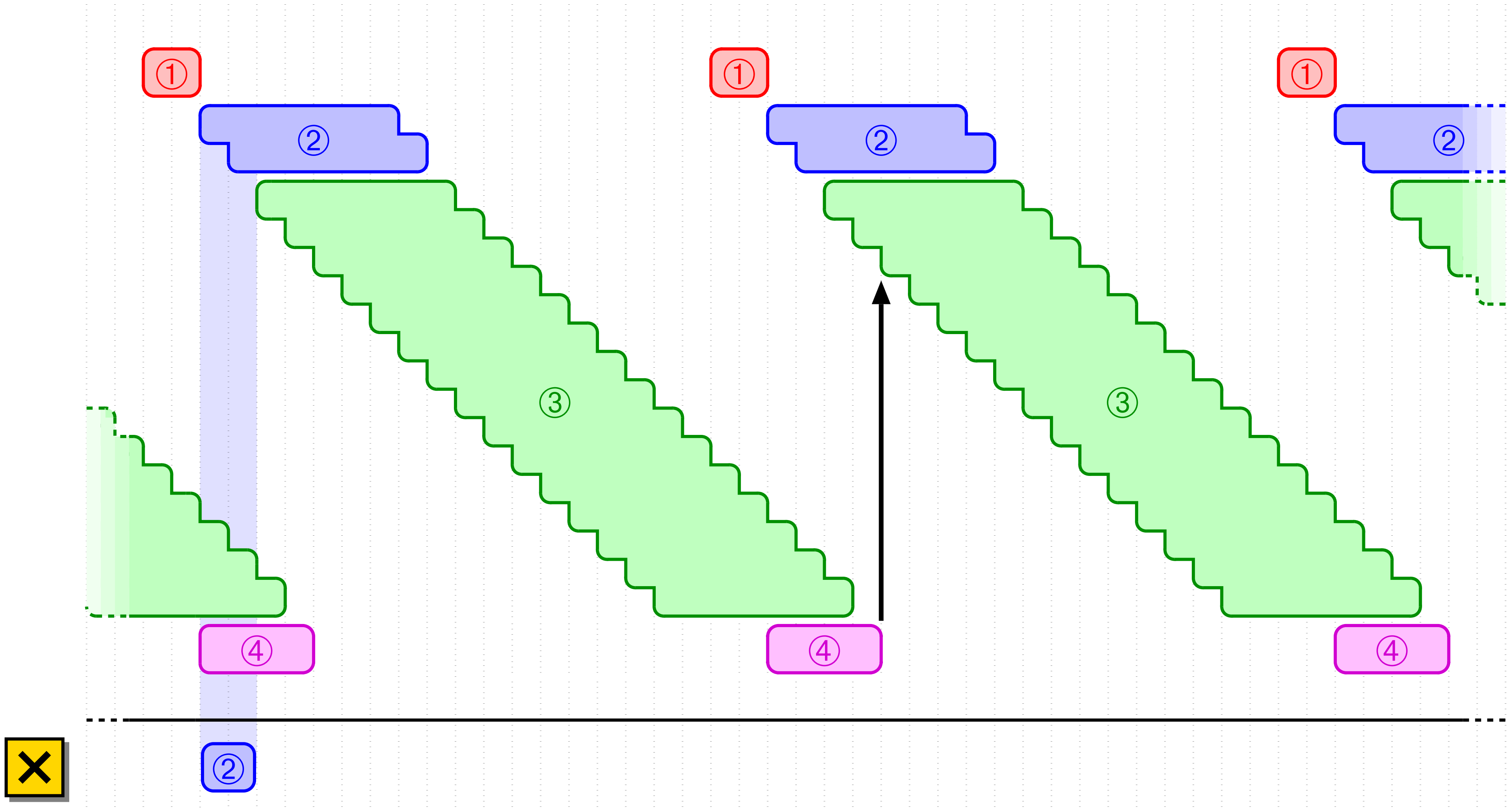
Task scheduling



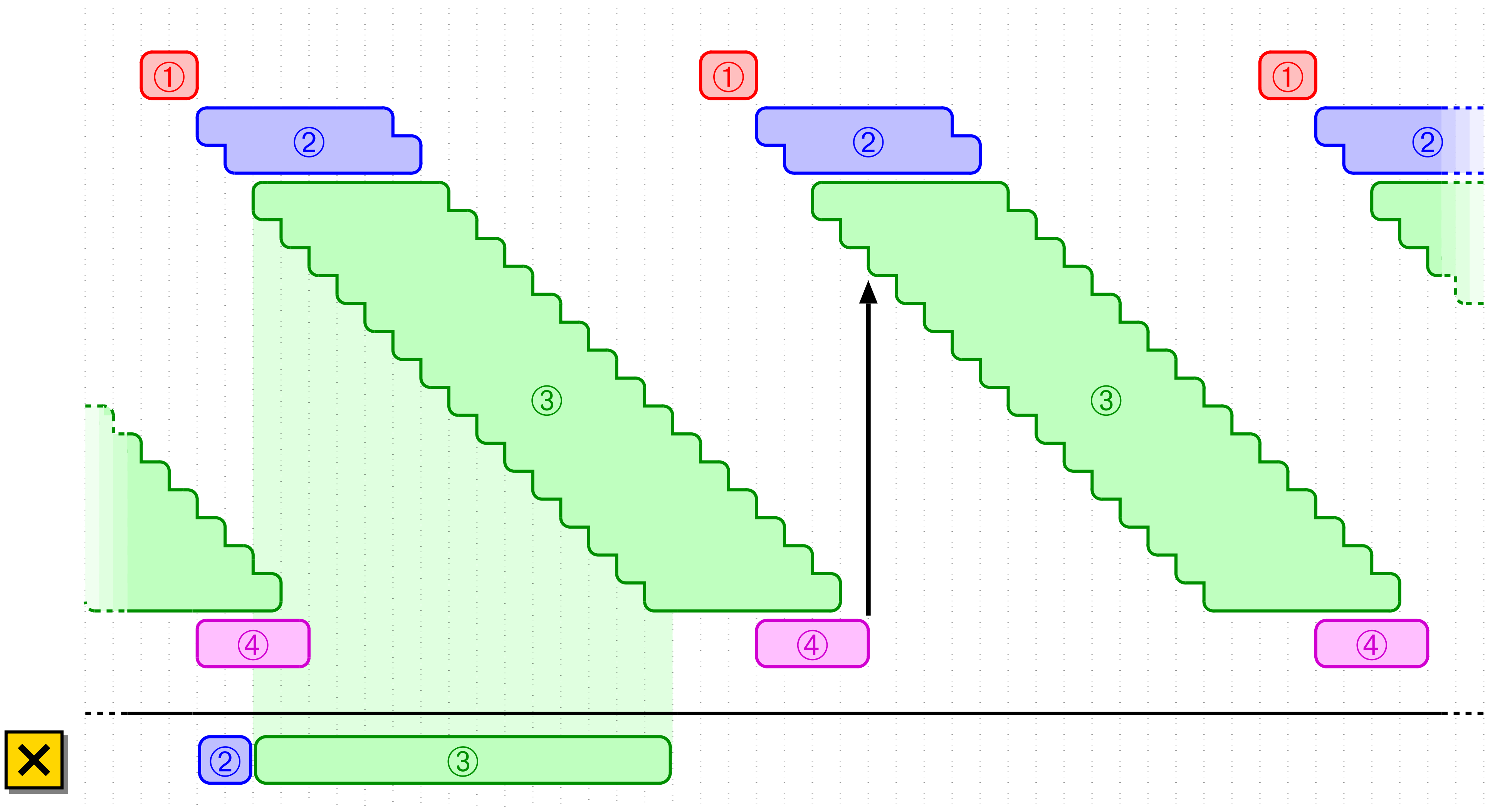
Task scheduling



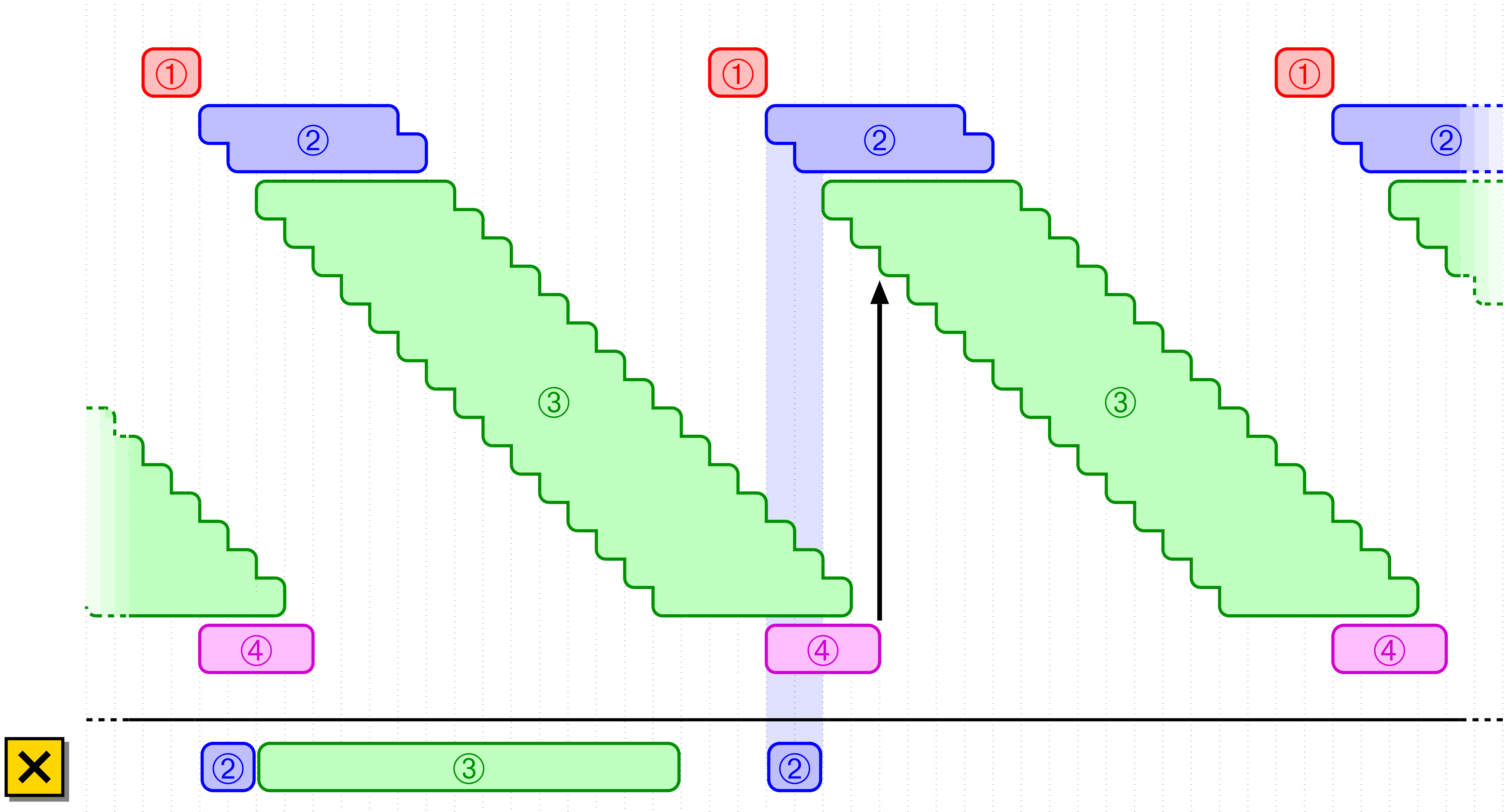
Task scheduling



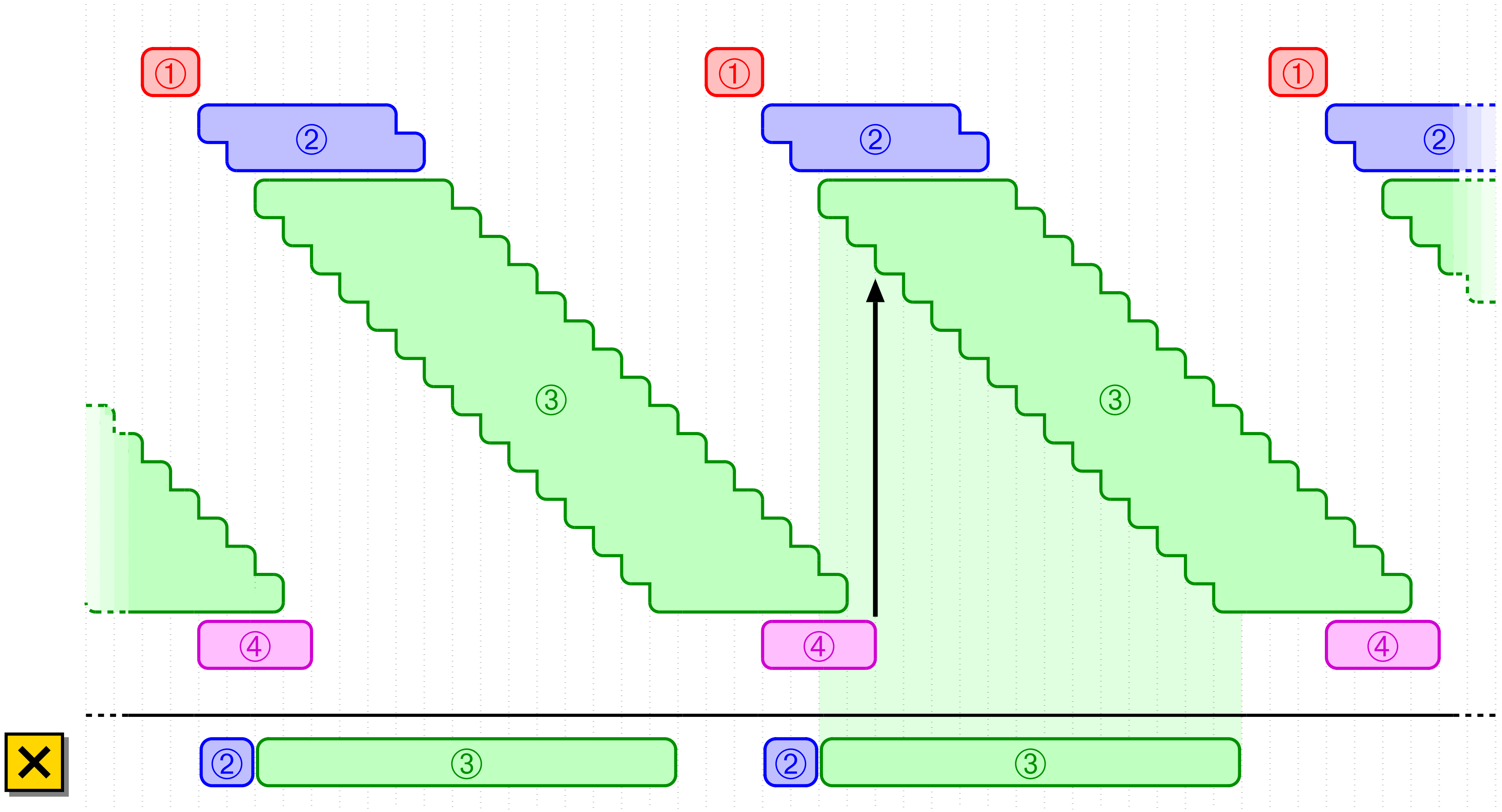
Task scheduling



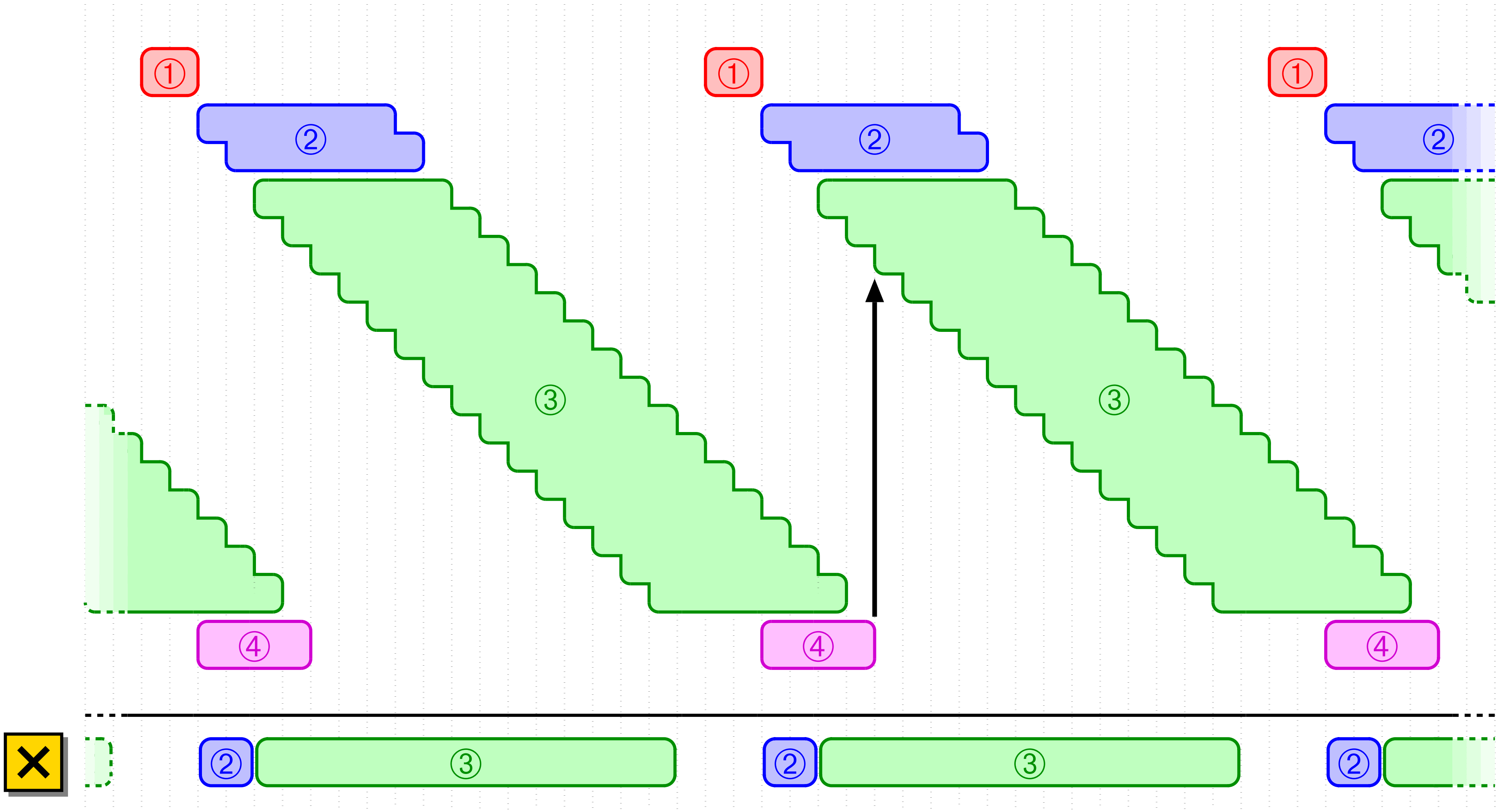
Task scheduling



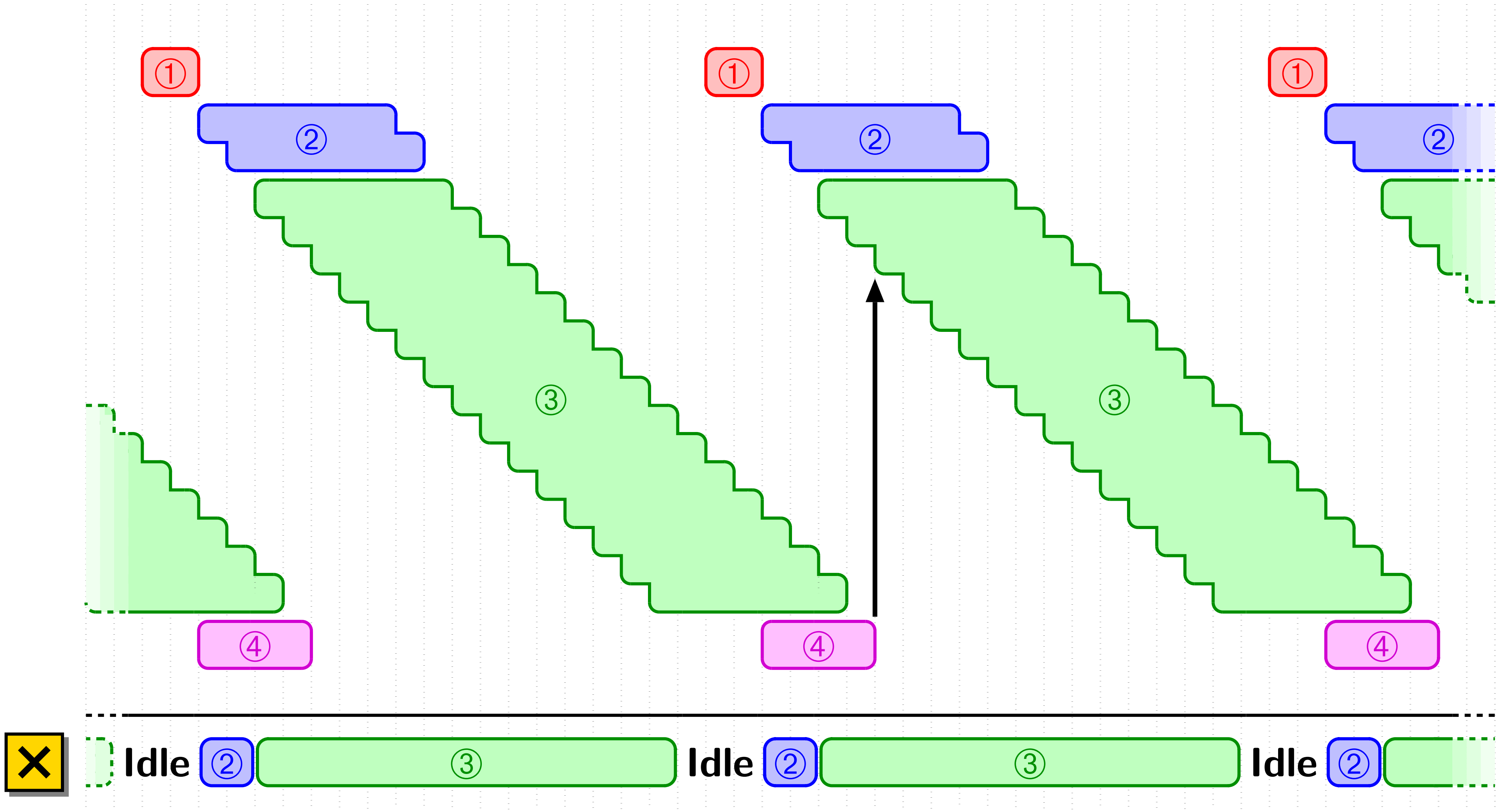
Task scheduling



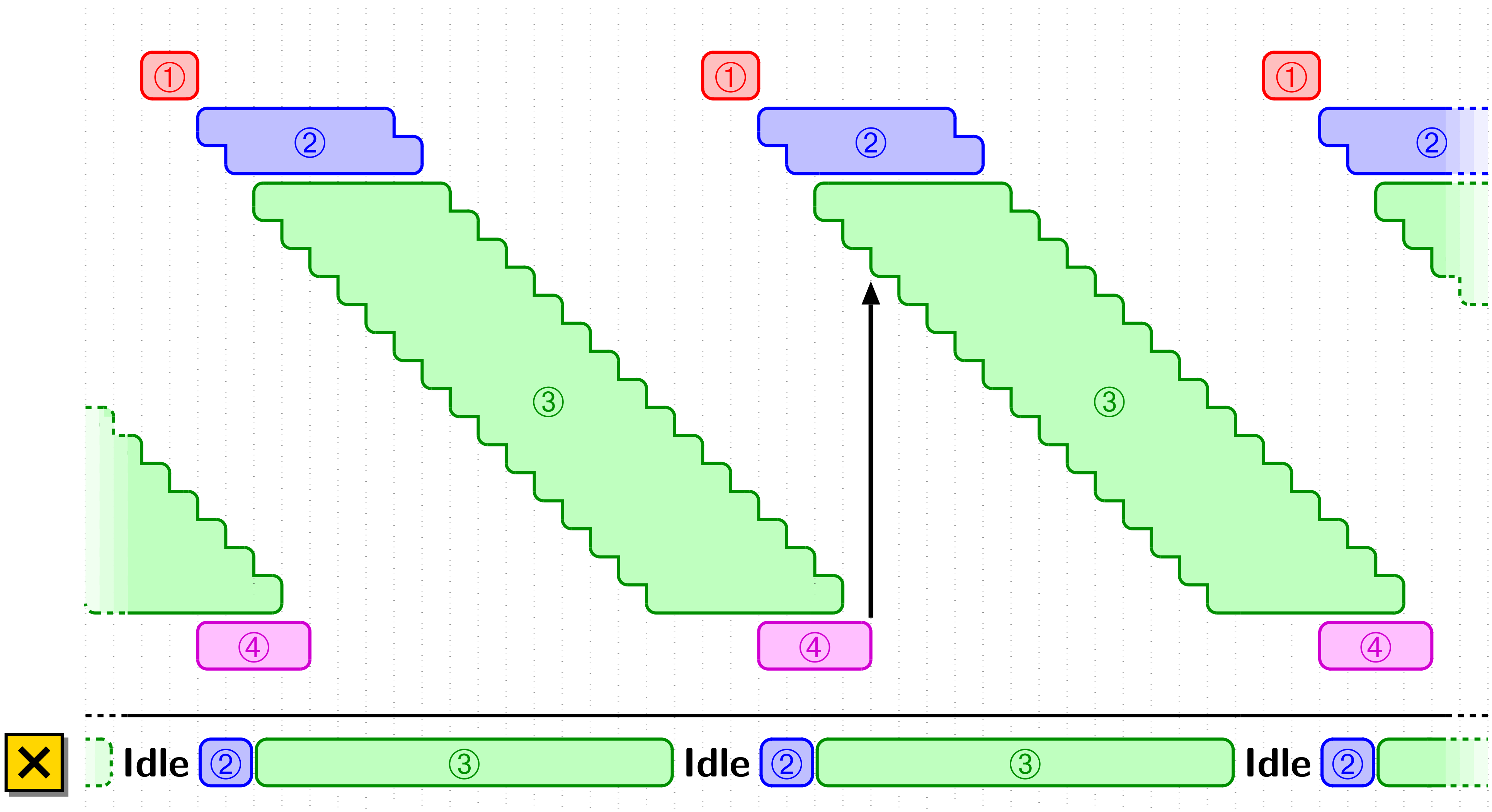
Task scheduling



Task scheduling

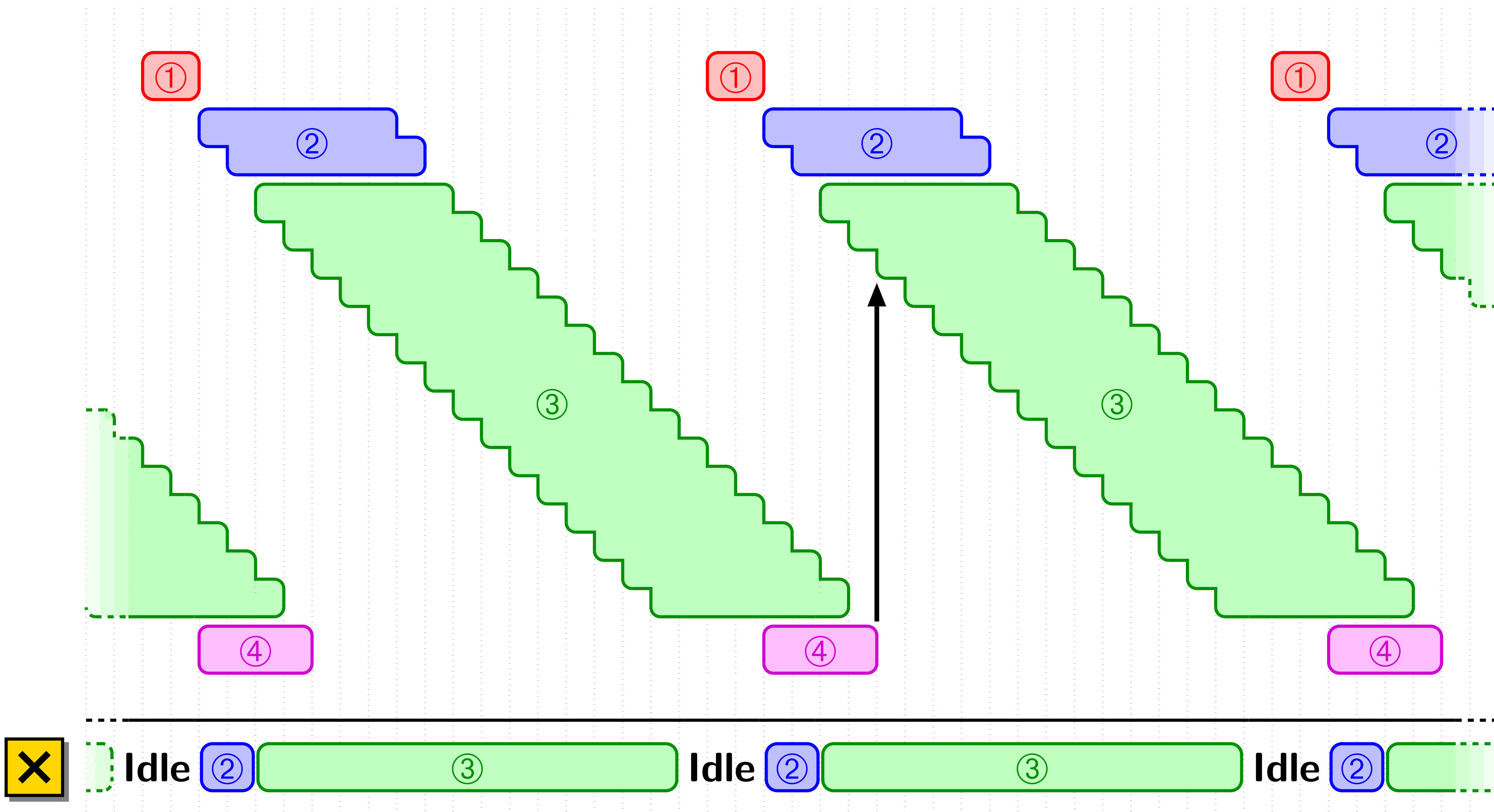


Task scheduling



- The task dependency ④ → ③ forces idle cycles into the multiplier pipeline

Task scheduling



- The task dependency ④ → ③ forces idle cycles into the multiplier pipeline
- The considered η_T pairing algorithm is not suited for parallel implementation

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

① $x_Q \leftarrow x_Q^9 \pm 1 \; ; \; y_Q \leftarrow -y_Q^9$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^3$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}_{3^{6m}}^{\times}$$

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

① $x_P \leftarrow x_P$; $y_P \leftarrow y_P$
$x_Q \leftarrow x_Q^9 \pm 1$; $y_Q \leftarrow -y_Q^9$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^3$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^\times$$

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

① $x_P \leftarrow x_P$; $y_P \leftarrow y_P$
$x_Q \leftarrow x_Q^{9} \pm 1$; $y_Q \leftarrow -y_Q^{9}$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^{3}$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

① $x_P \leftarrow x_P$; $y_P \leftarrow y_P$
$x_Q \leftarrow x_Q^{\mathbf{9}} \pm 1$; $y_Q \leftarrow -y_Q^{\mathbf{9}}$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^{\mathbf{3}}$

end for

⇒

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

① $x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$
$x_Q \leftarrow x_Q^{\mathbf{3}}$; $y_Q \leftarrow y_Q^{\mathbf{3}}$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^\times$$

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

① $x_P \leftarrow x_P$; $y_P \leftarrow y_P$
$x_Q \leftarrow x_Q^{9} \pm 1$; $y_Q \leftarrow -y_Q^{9}$

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

③ $R \leftarrow R \cdot S$

④ $R \leftarrow R^{3}$

end for

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

③ $R \leftarrow R \cdot S$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

③ $R \leftarrow R \cdot S$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}_{3^{6m}}^{\times}$$

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 inv. Frobenius
2 Frobenius
$(\mathbb{F}_{3^m})$

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

③ $R \leftarrow R \cdot S$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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 inv. Frobenius $(\mathbb{F}_{3^m})$
$x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$ — 2 Frobenius

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

③ $R \leftarrow R \cdot S$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \longrightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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 inv. Frobenius 2 Frobenius	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$15\times$, $29+$	$(\mathbb{F}_{3^m})$

end for

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}_{3^{6m}}^{\times}$$

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 inv. Frobenius 2 Frobenius	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$15\times$, $29+$	$(\mathbb{F}_{3^m})$

end for

- Modified algorithm: $17\times$, 2 Frobenius, 2 inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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 inv. Frobenius 2 Frobenius	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$15\times$, $29+$	$(\mathbb{F}_{3^m})$

end for

- Modified algorithm: $17\times$, 2 Frobenius, 2 inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$
- Previous algorithm: $17\times$, 10 Frobenius and $38+$

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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 inv. Frobenius 2 Frobenius	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$15\times$, $29+$	$(\mathbb{F}_{3^m})$

end for

- Modified algorithm: $17\times$, 2 Frobenius, 2 inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$
- Previous algorithm: $17\times$, 10 Frobenius and $38+$
- Cost of the inverse Frobenius?

A modified algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

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 inv. Frobenius 2 Frobenius	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$15\times$, $29+$	$(\mathbb{F}_{3^m})$

end for

- Modified algorithm: $17\times$, 2 Frobenius, 2 inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$
- Previous algorithm: $17\times$, 10 Frobenius and $38+$
- Cost of the inverse Frobenius? Same as the Frobenius

Data dependencies

Iteration $i-1$

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

↓

③ $R \leftarrow R \cdot S$

Iteration i

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

↓

③ $R \leftarrow R \cdot S$

Iteration $i+1$

① $x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$
$x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$

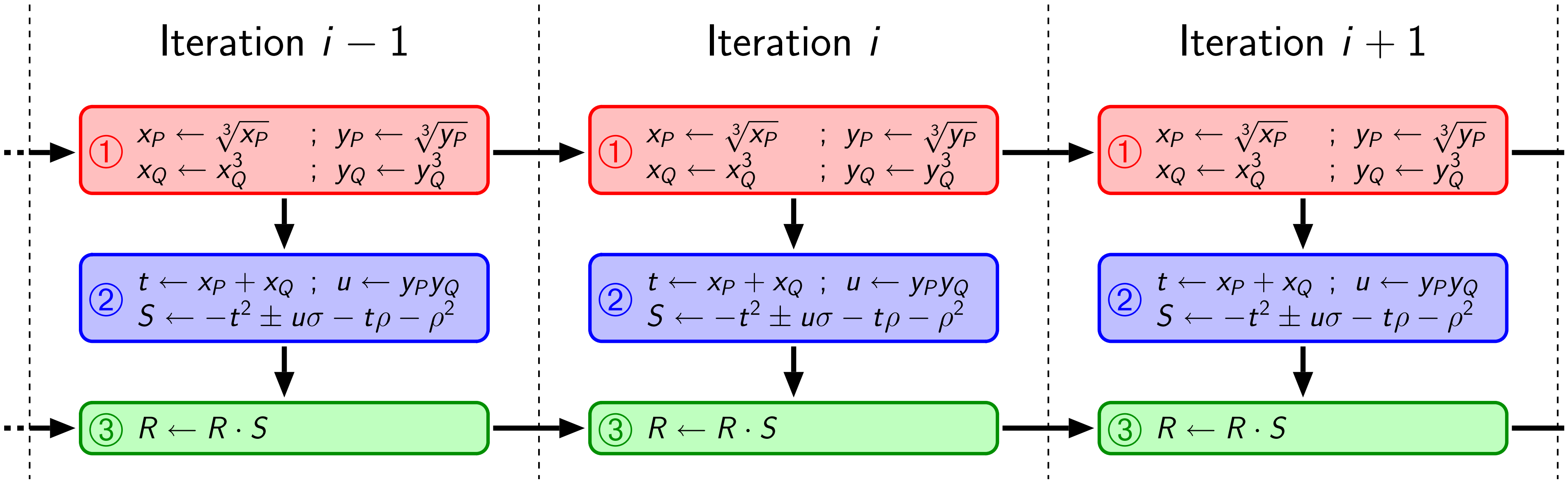
↓

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

↓

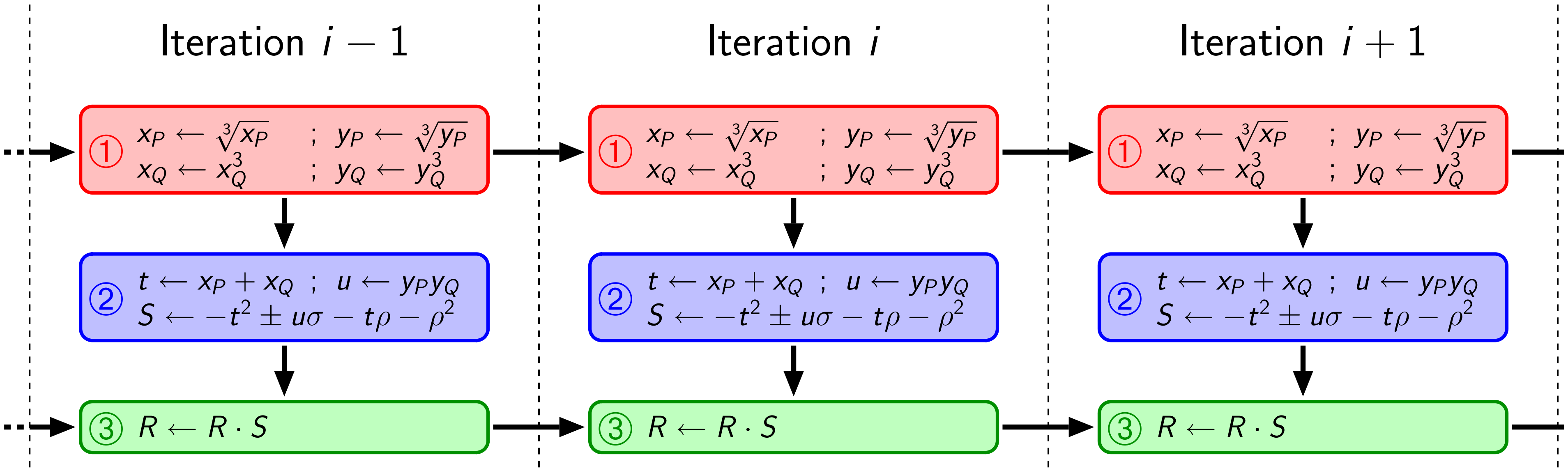
③ $R \leftarrow R \cdot S$

Data dependencies



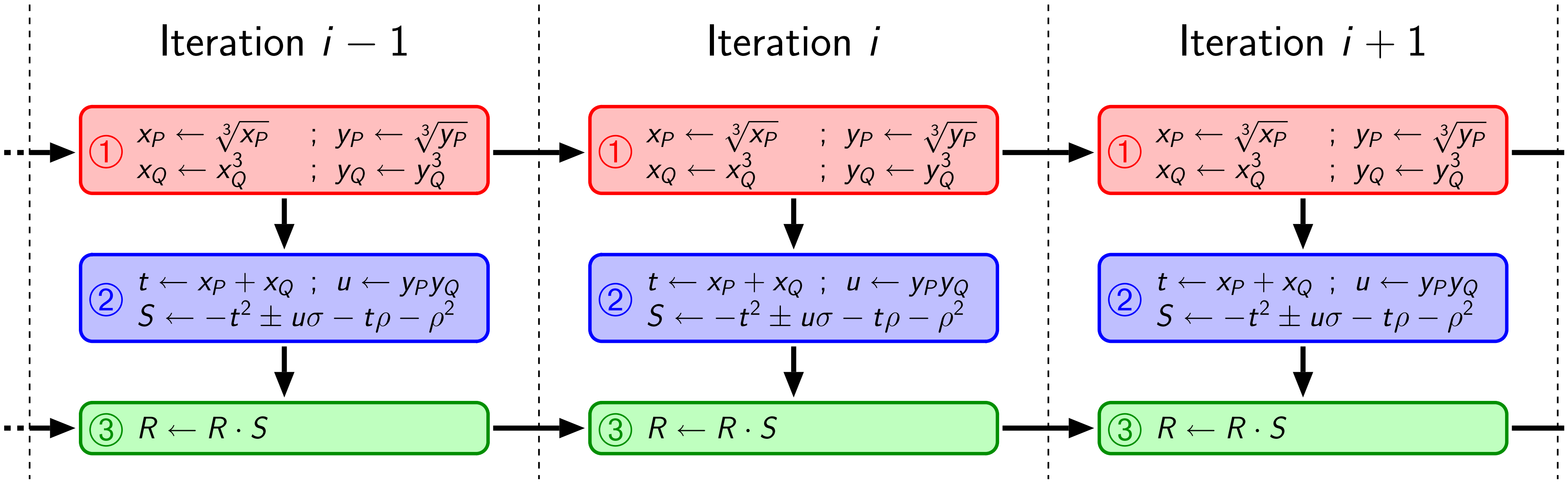
- Direct dependency ③ $\rightarrow$ ③ between consecutive iterations

Data dependencies



- Direct dependency ③ → ③ between consecutive iterations
  - avoid the scheduling bottleneck of task ④
  - better overlapping of successive tasks ③

Data dependencies



- Direct dependency ③ $\rightarrow$ ③ between consecutive iterations
  - avoid the scheduling bottleneck of task ④
  - better overlapping of successive tasks ③
  - hopefully tighter scheduling

Task scheduling

Task scheduling

① $x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$
$x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$

Task scheduling

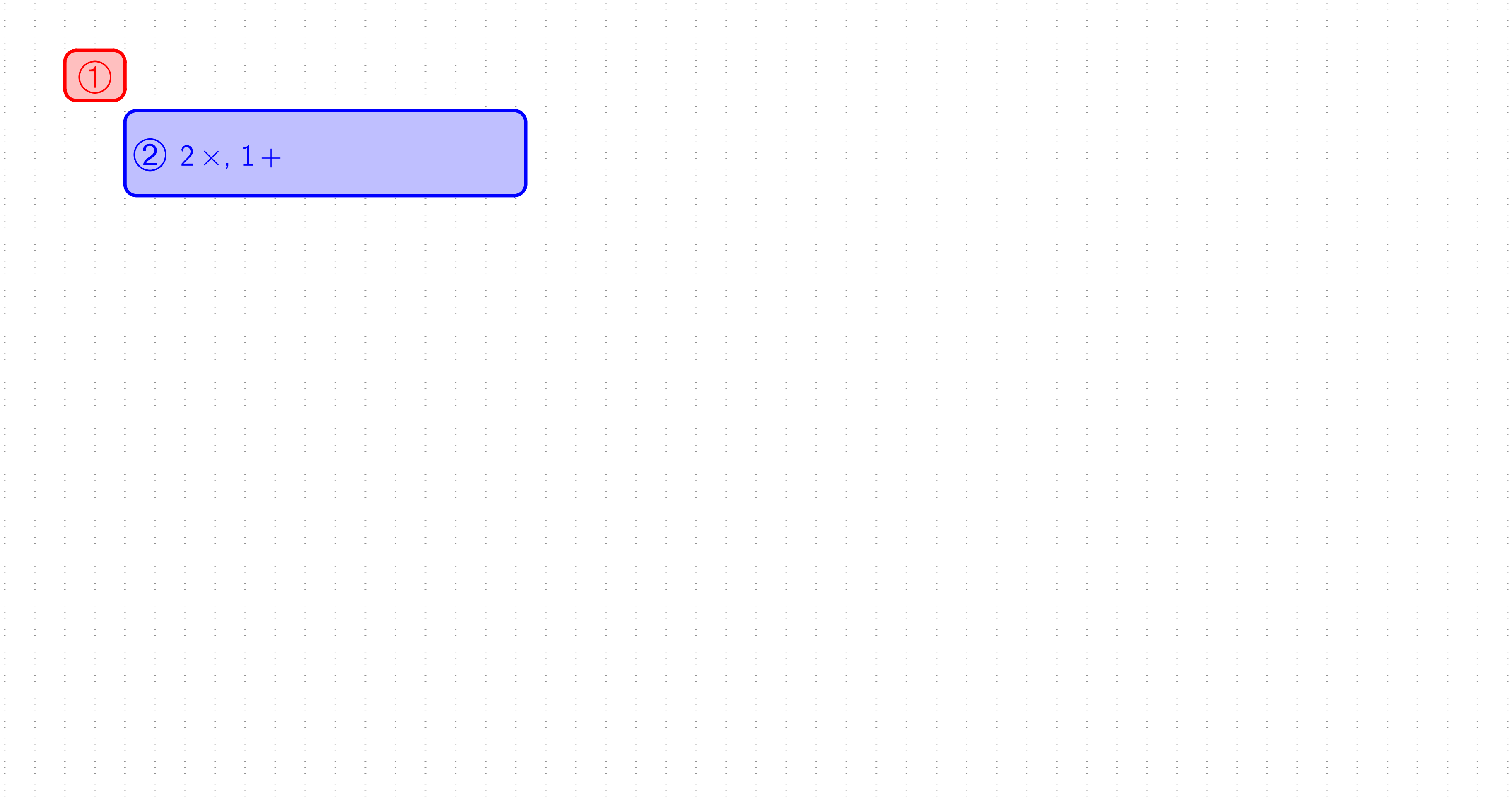
Task scheduling

Task scheduling

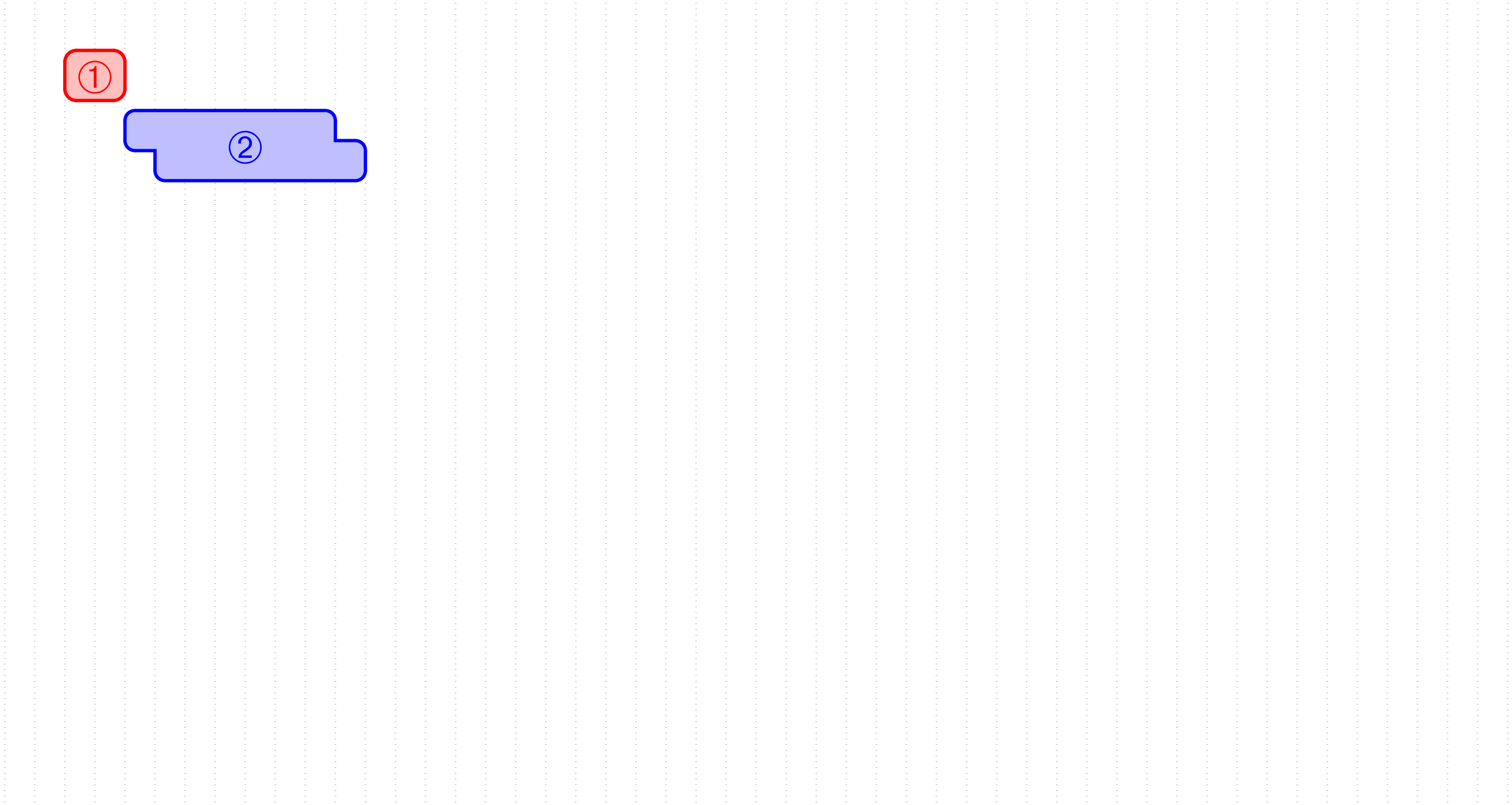
①

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

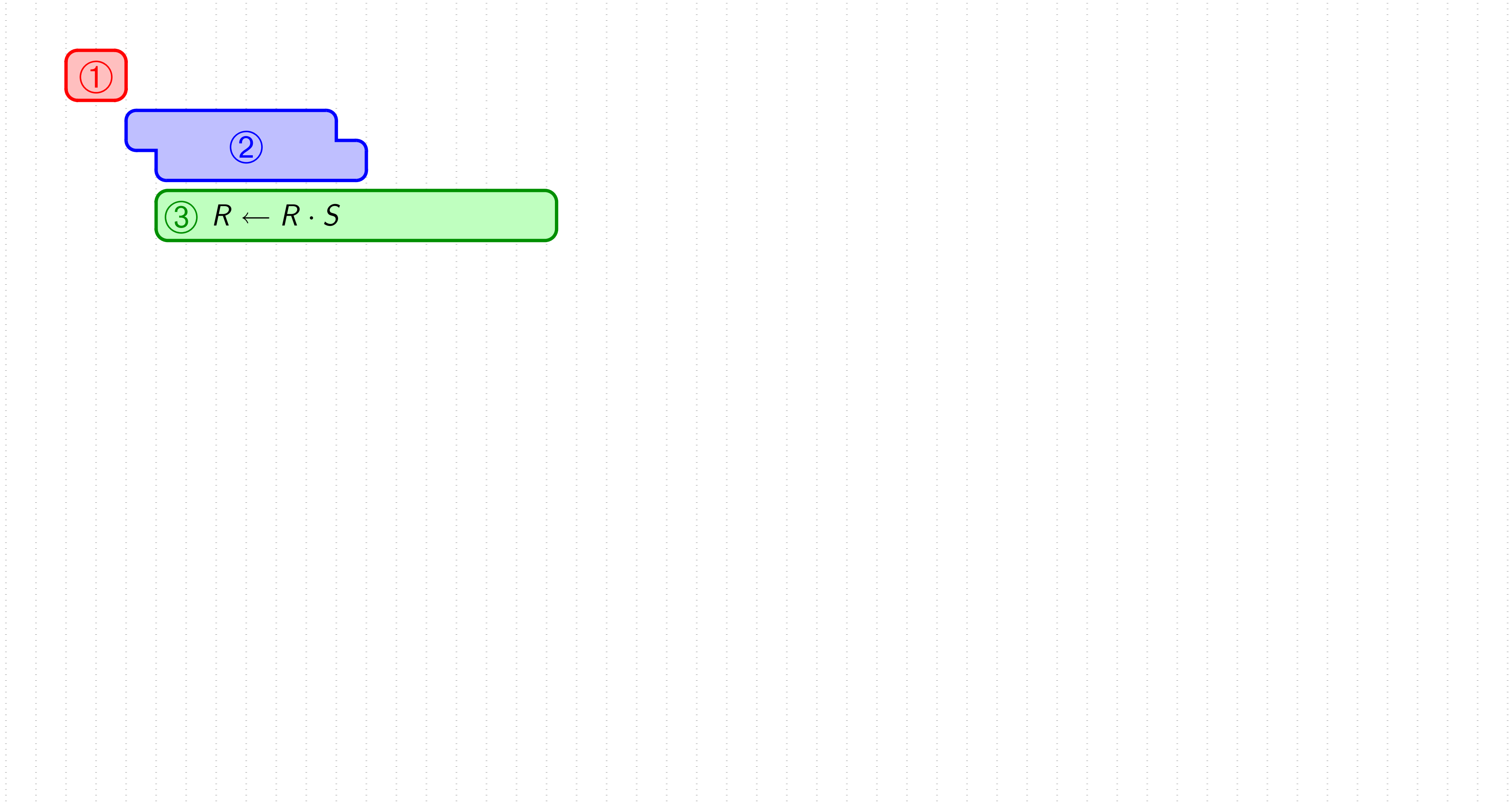
Task scheduling



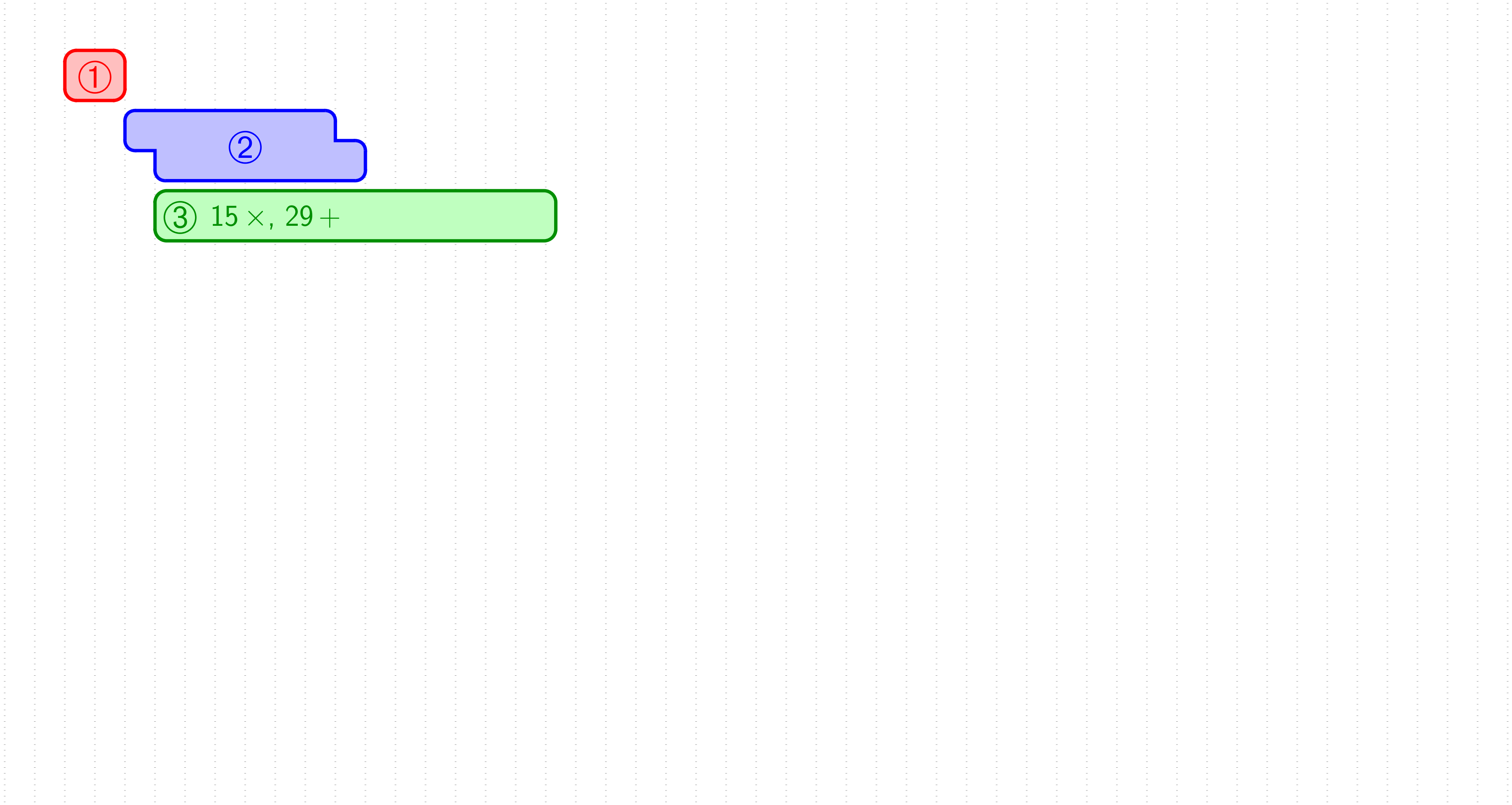
Task scheduling



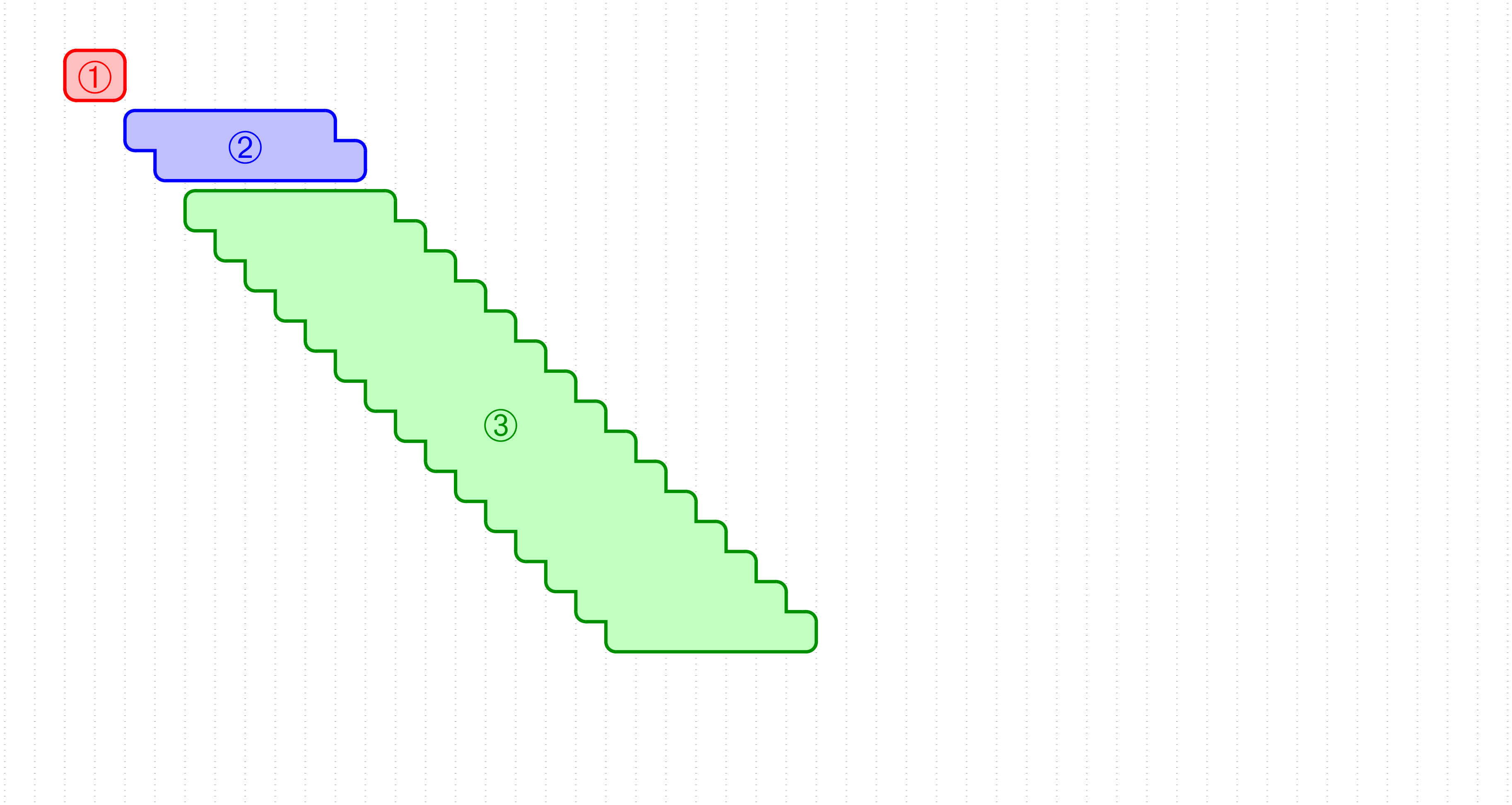
Task scheduling



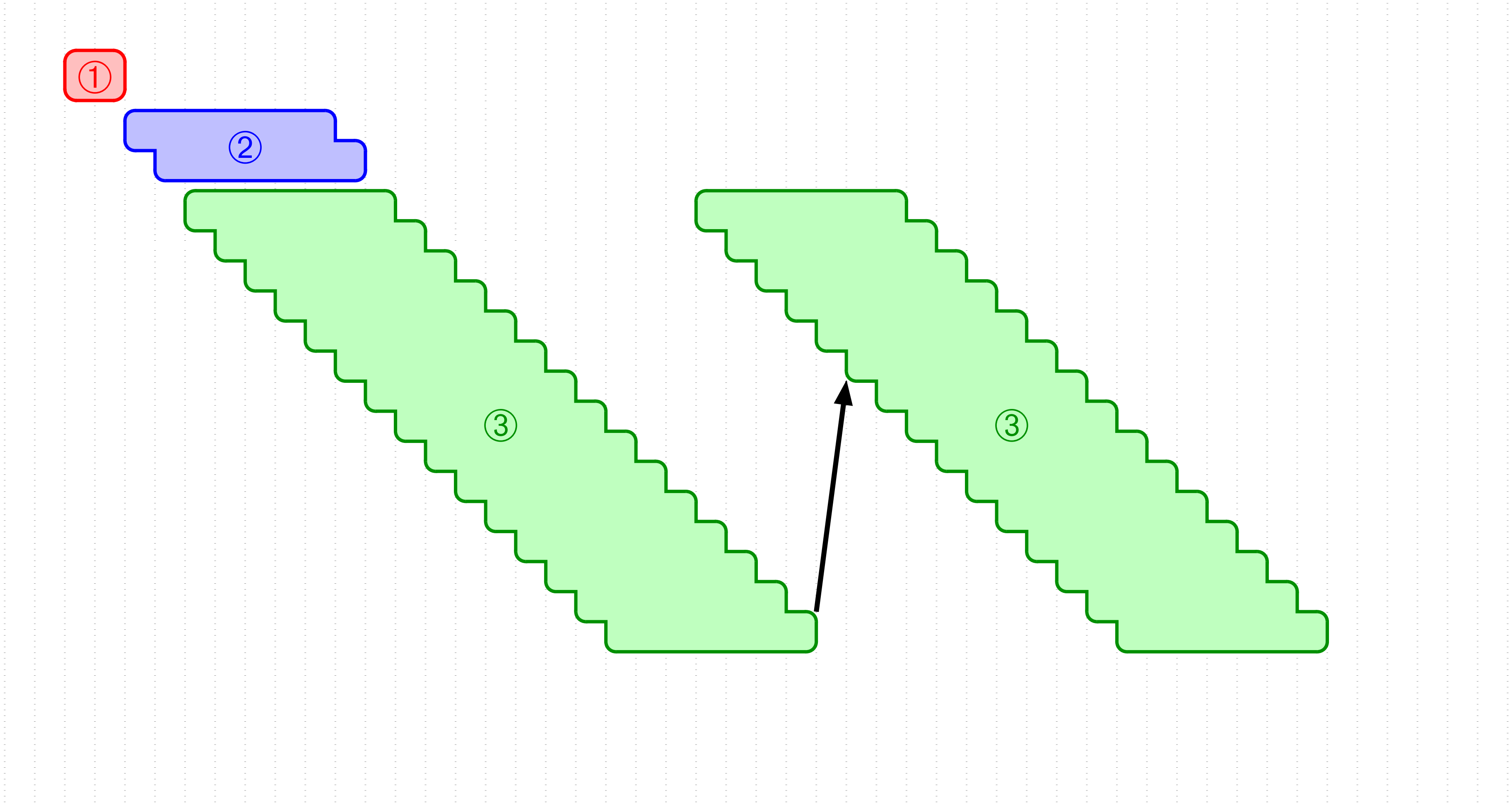
Task scheduling



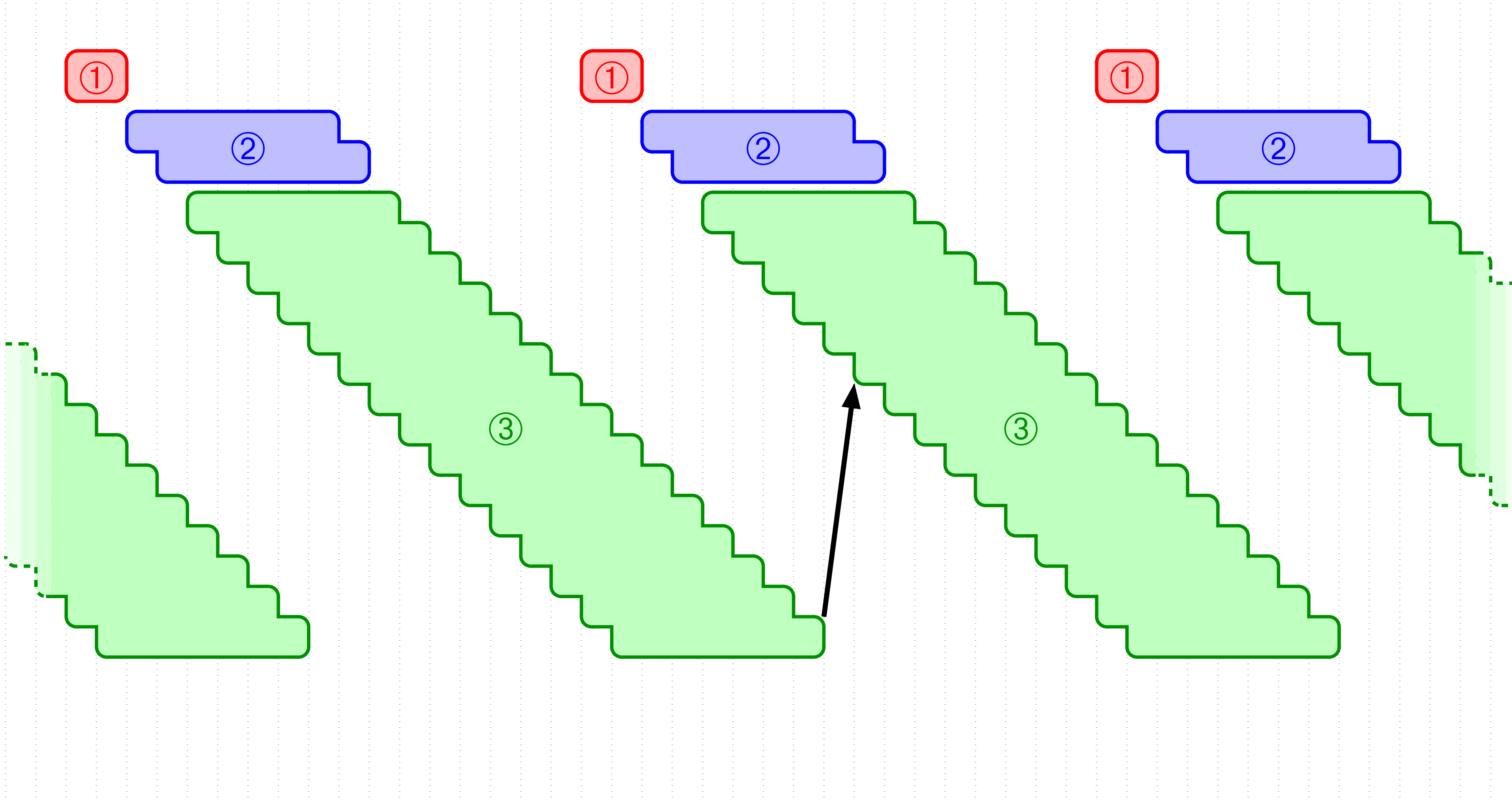
Task scheduling



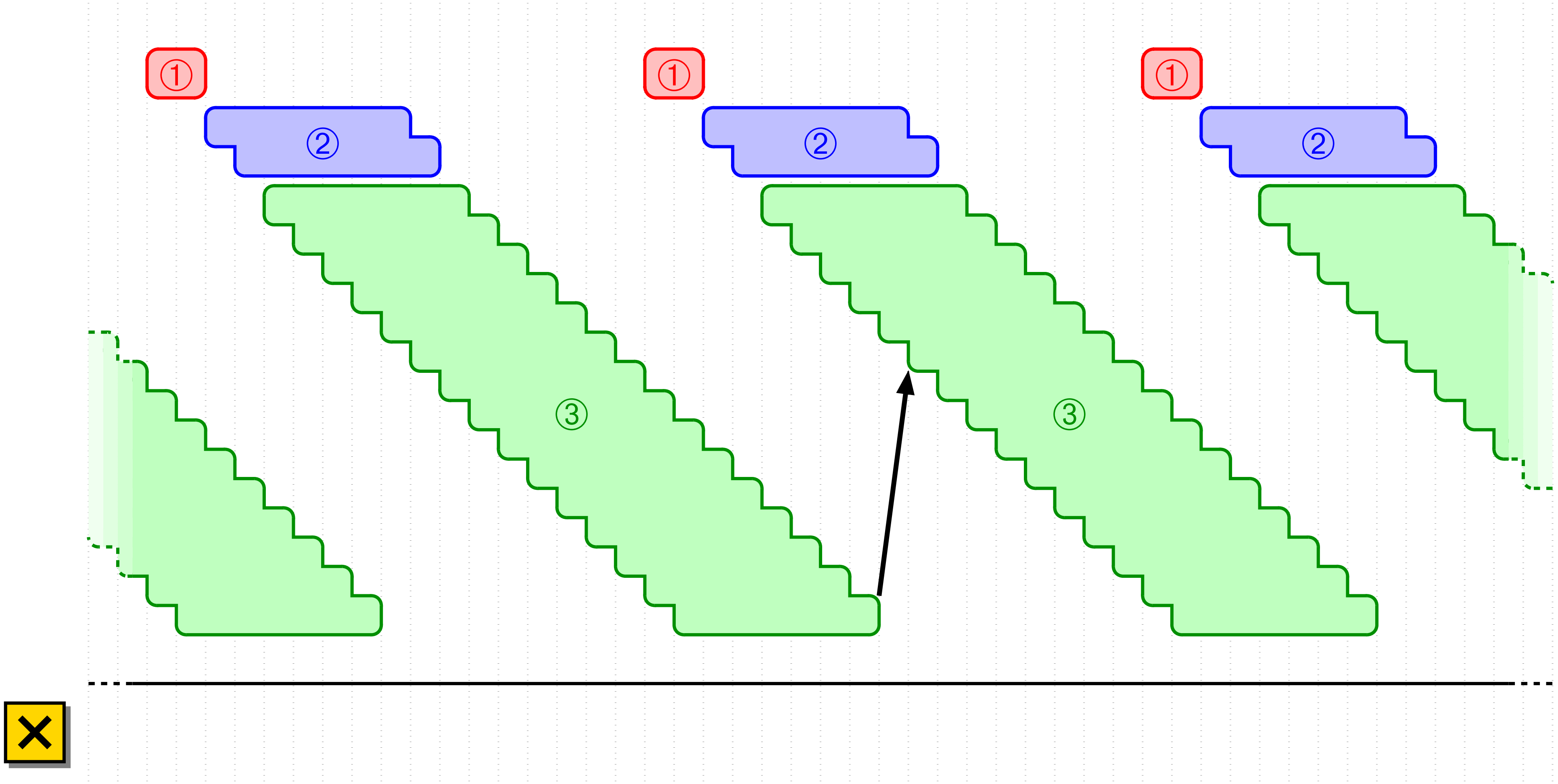
Task scheduling



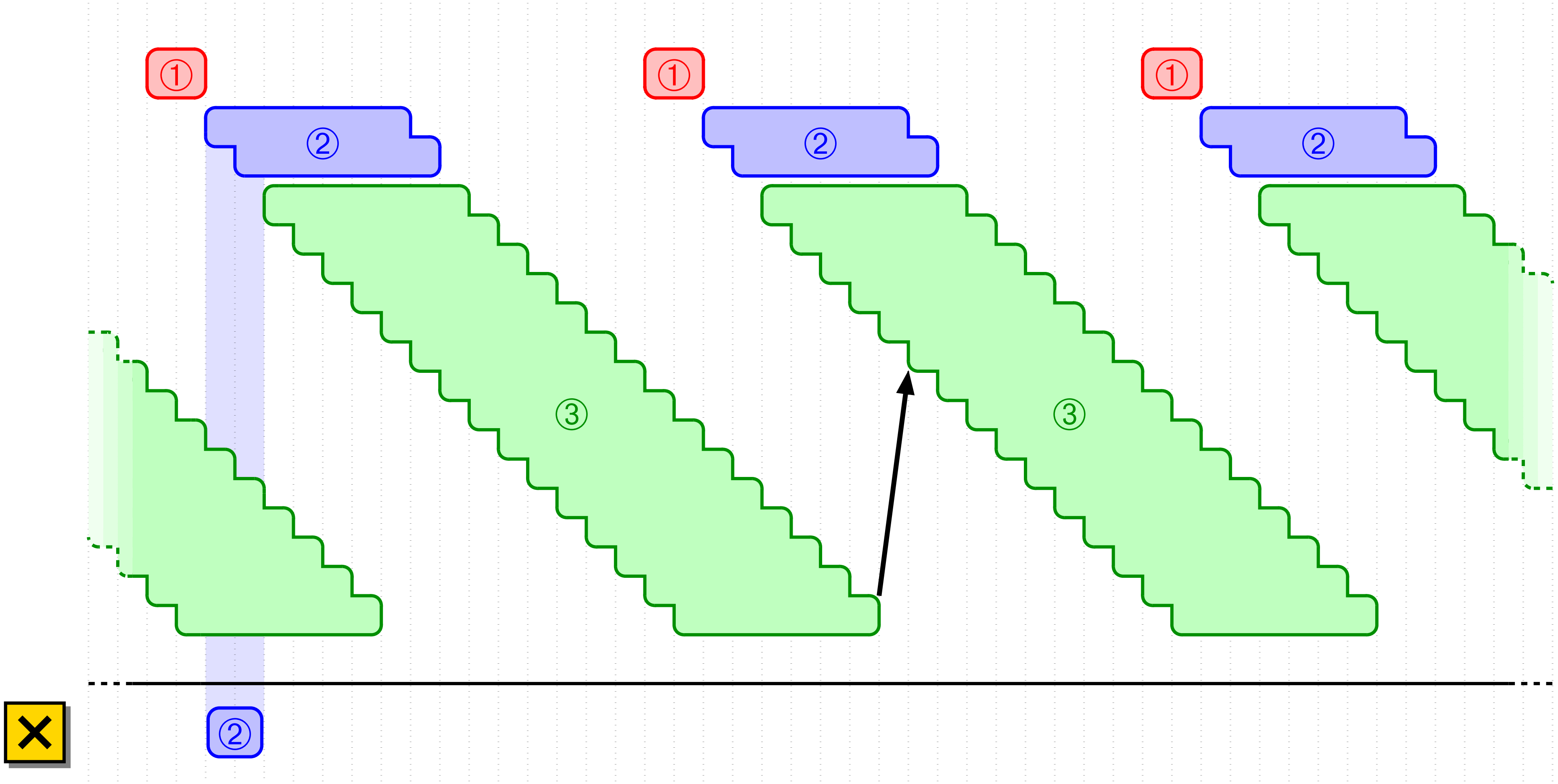
Task scheduling



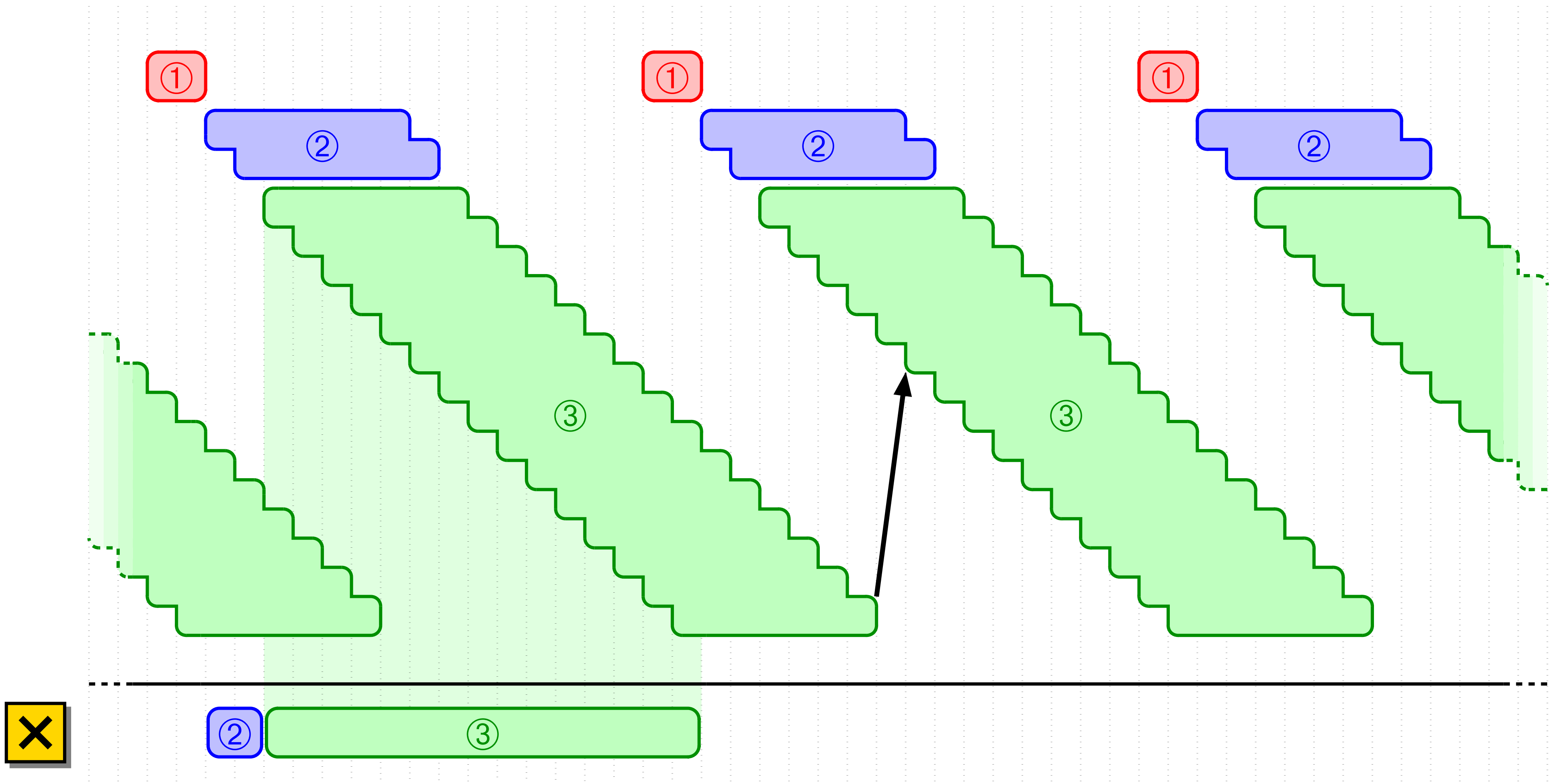
Task scheduling



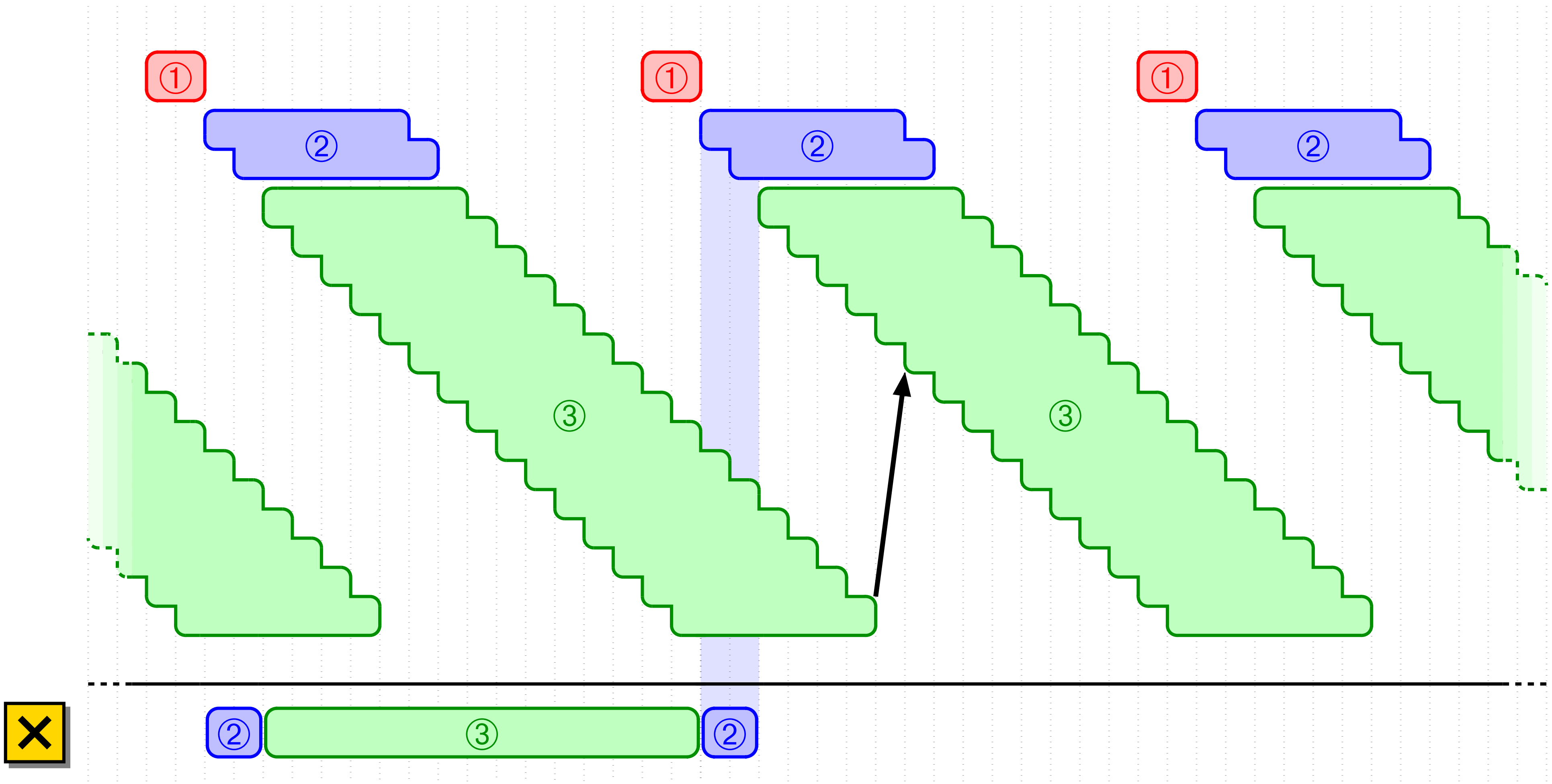
Task scheduling



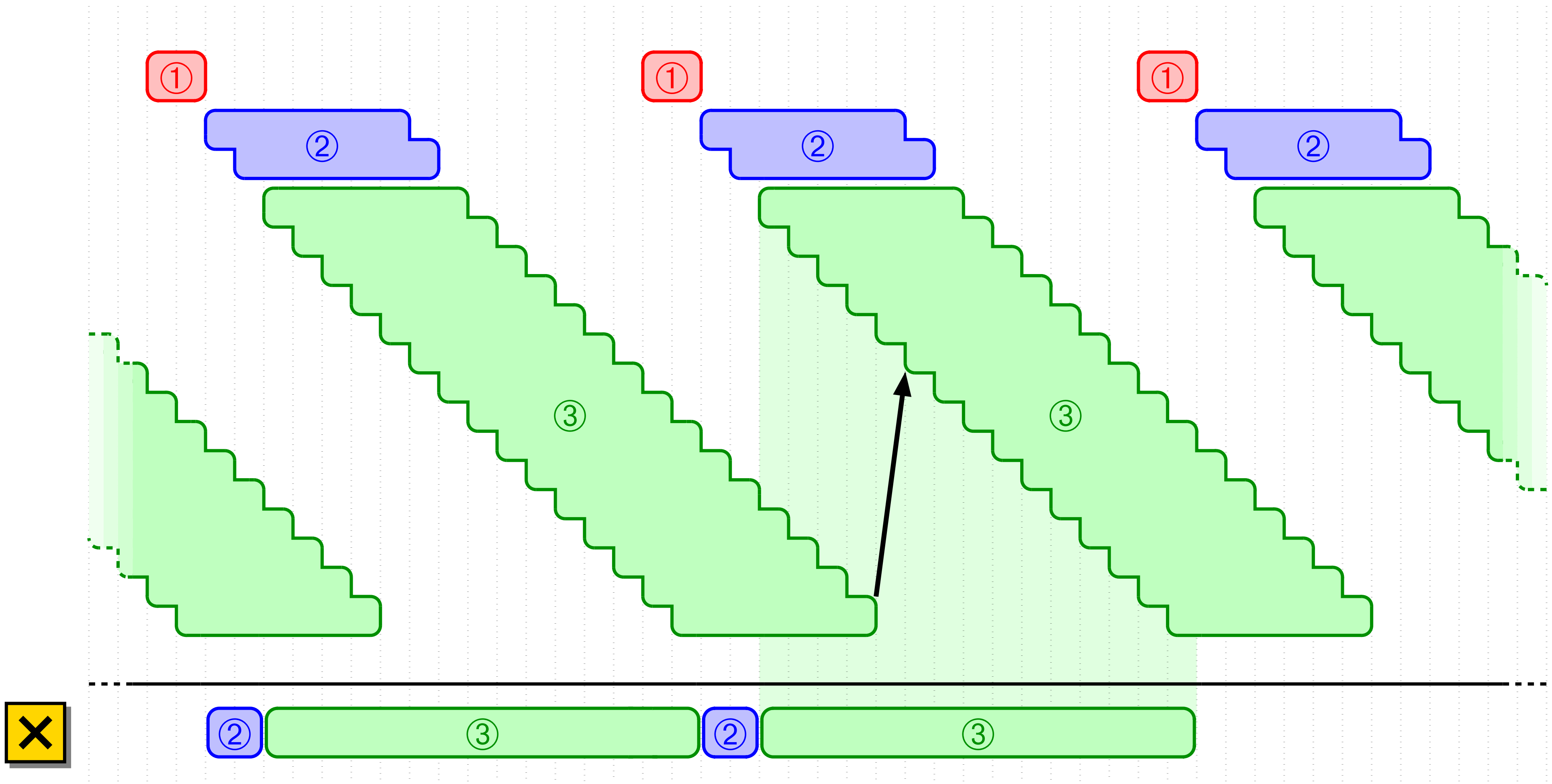
Task scheduling



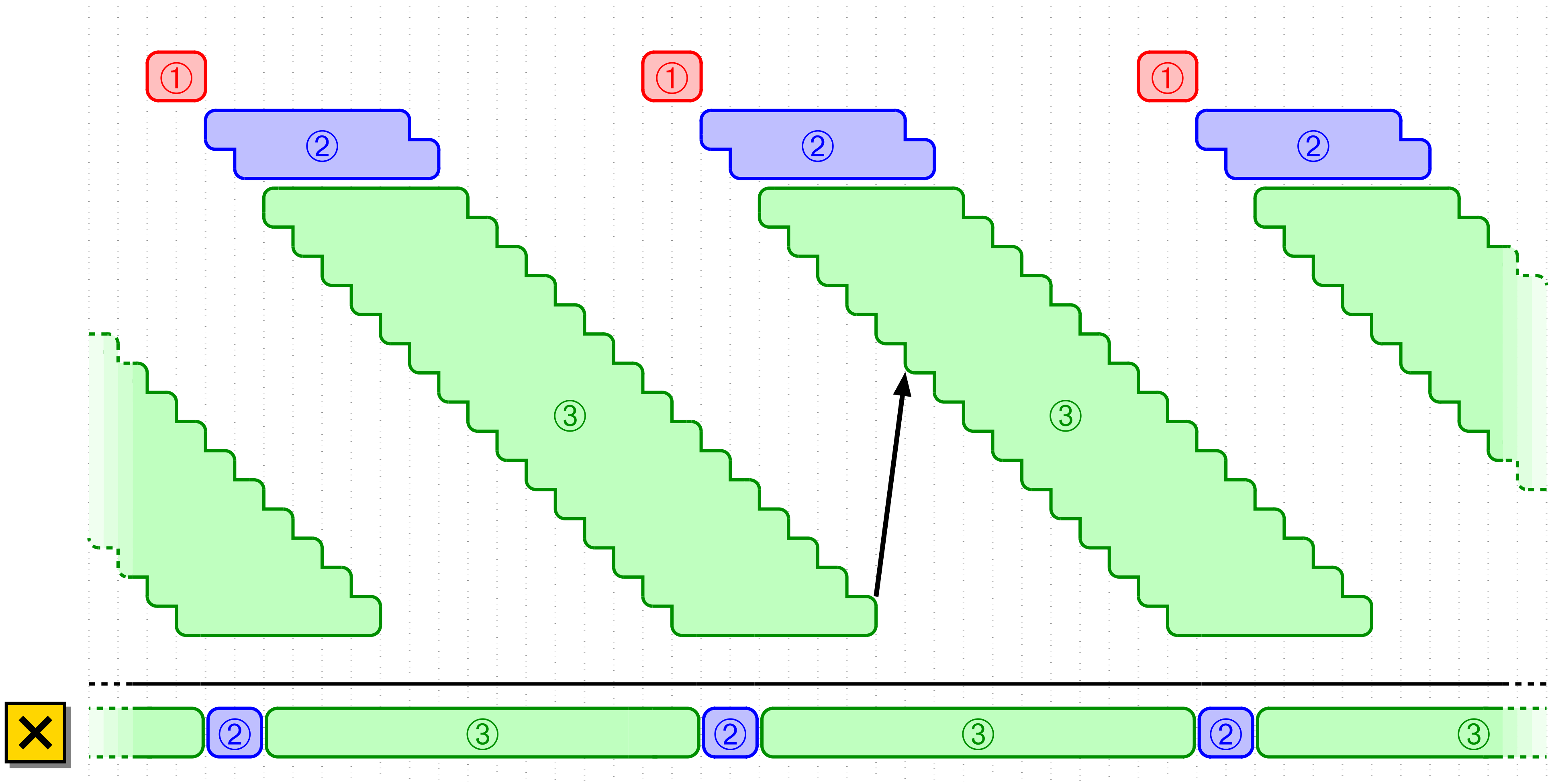
Task scheduling



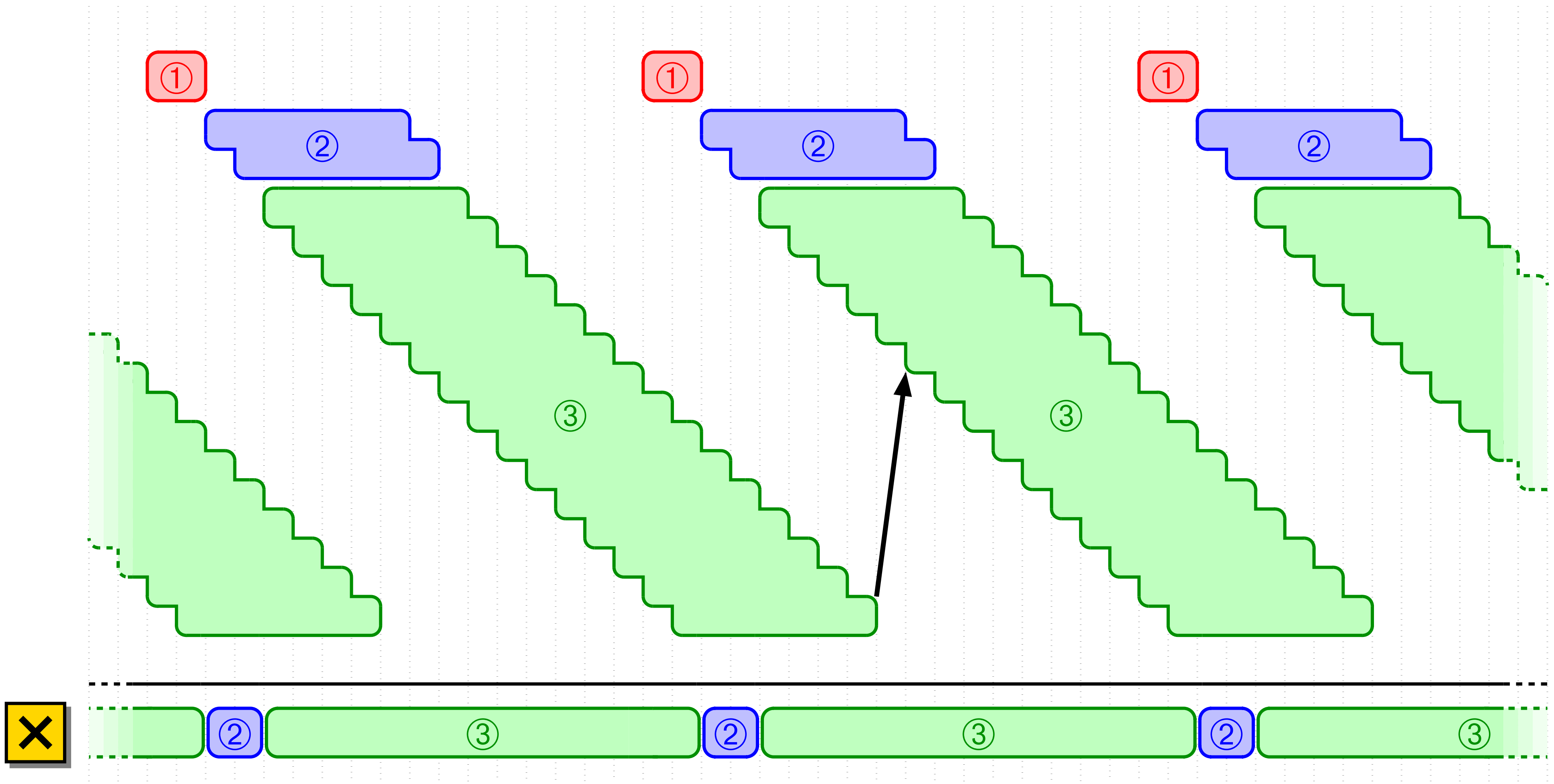
Task scheduling



Task scheduling

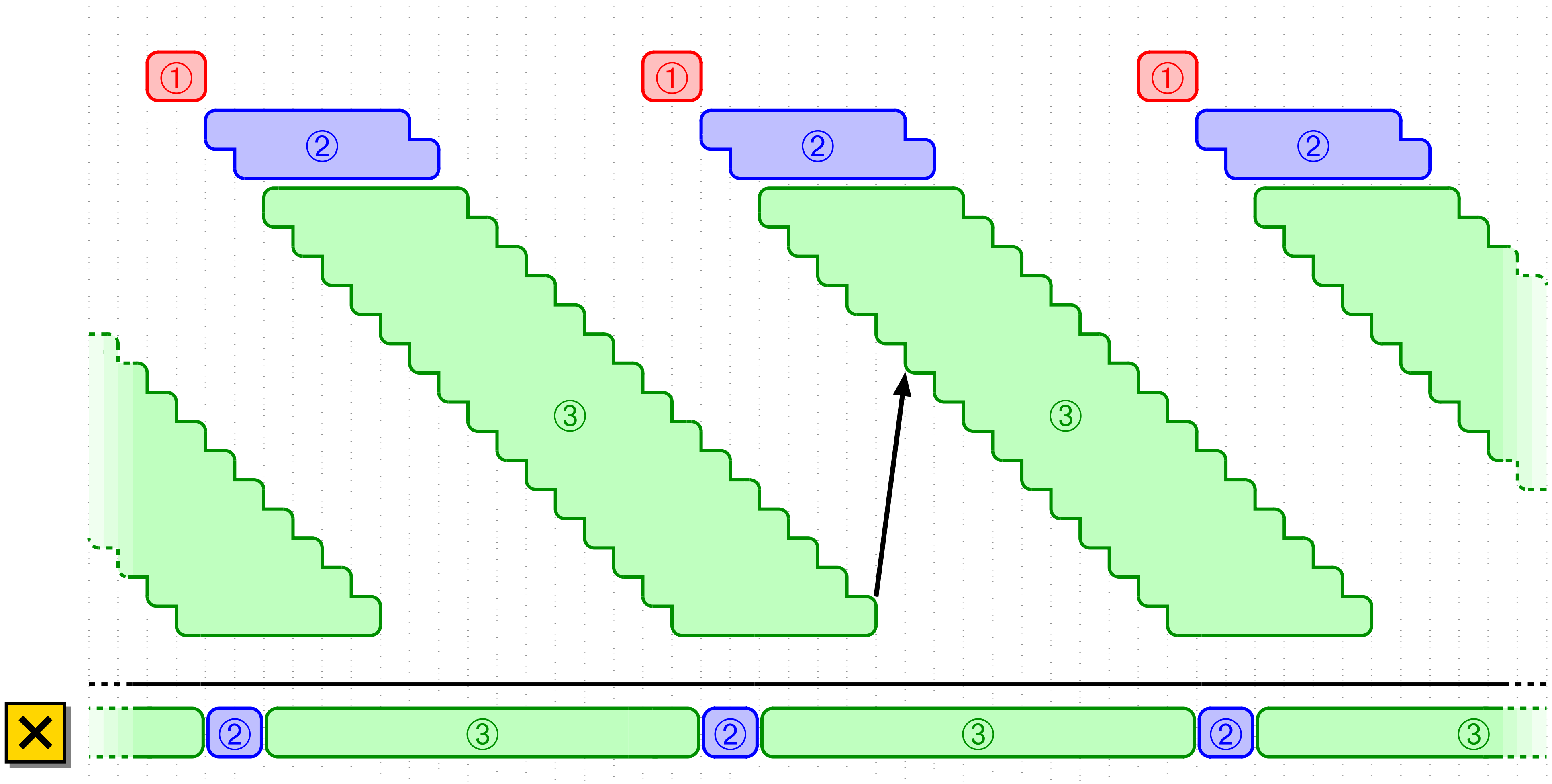


Task scheduling



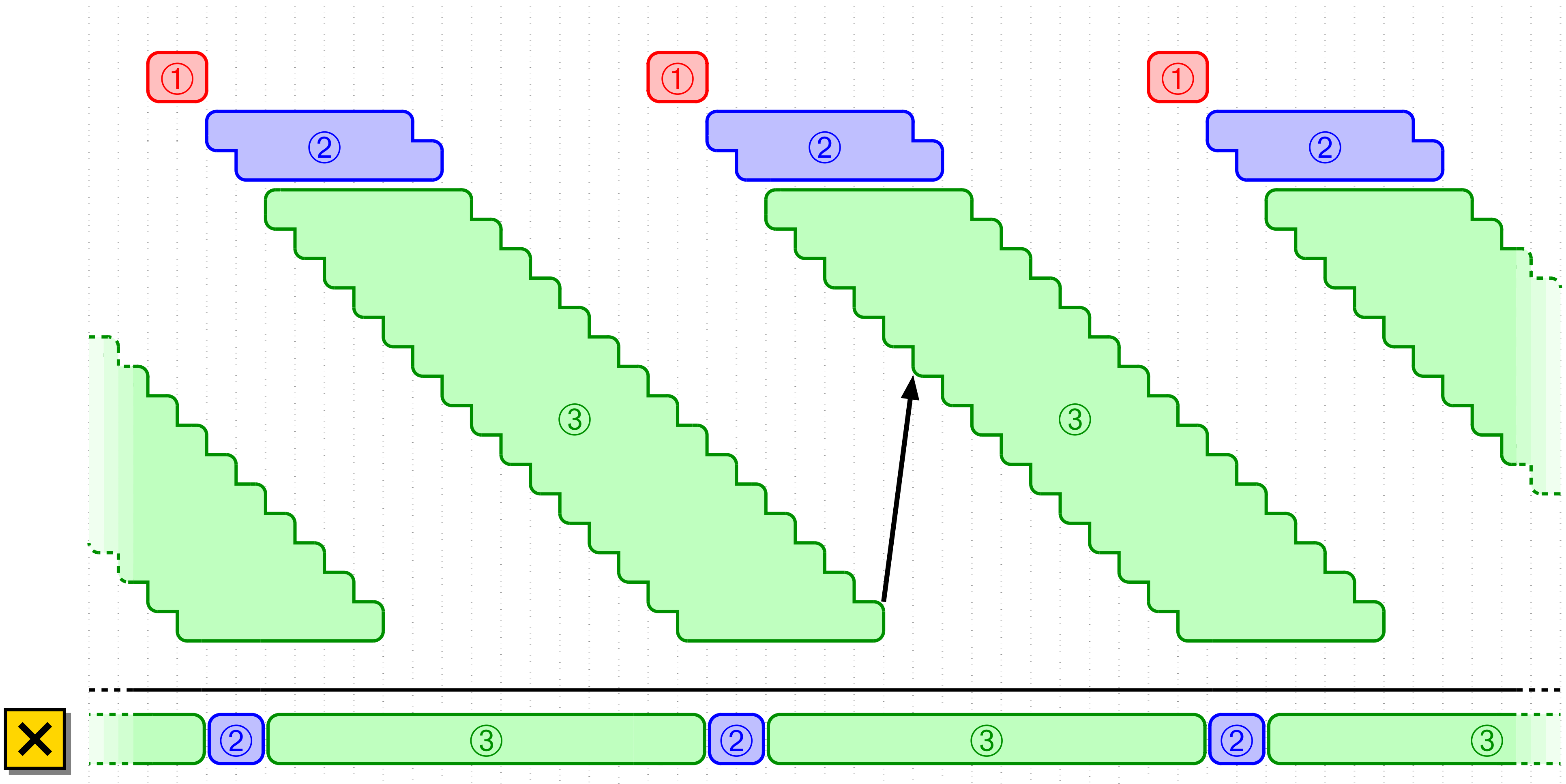
- Perfectly tight scheduling: no idle cycle

Task scheduling



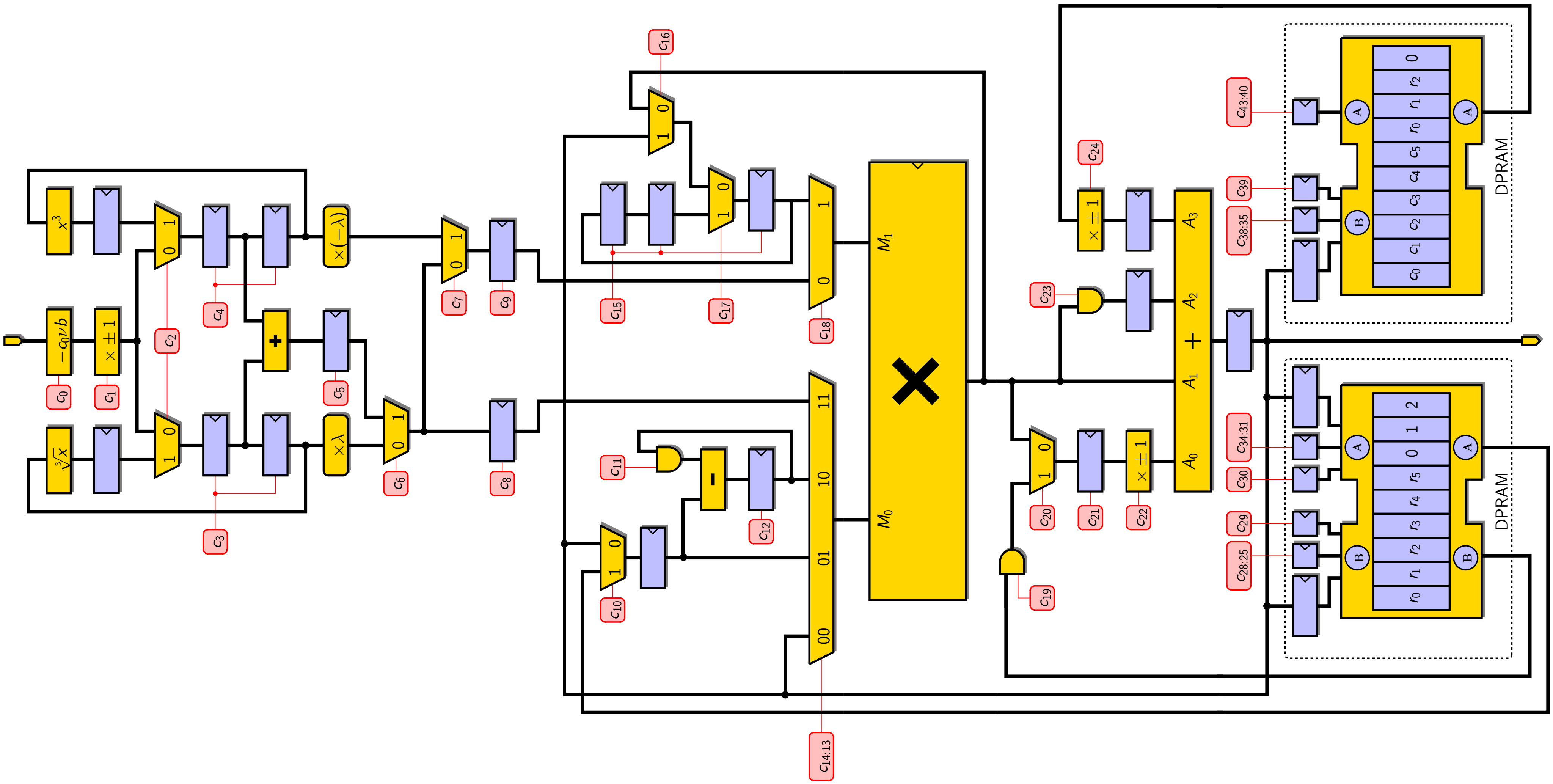
- Perfectly tight scheduling: no idle cycle
- 17 clock cycles per iteration

Task scheduling

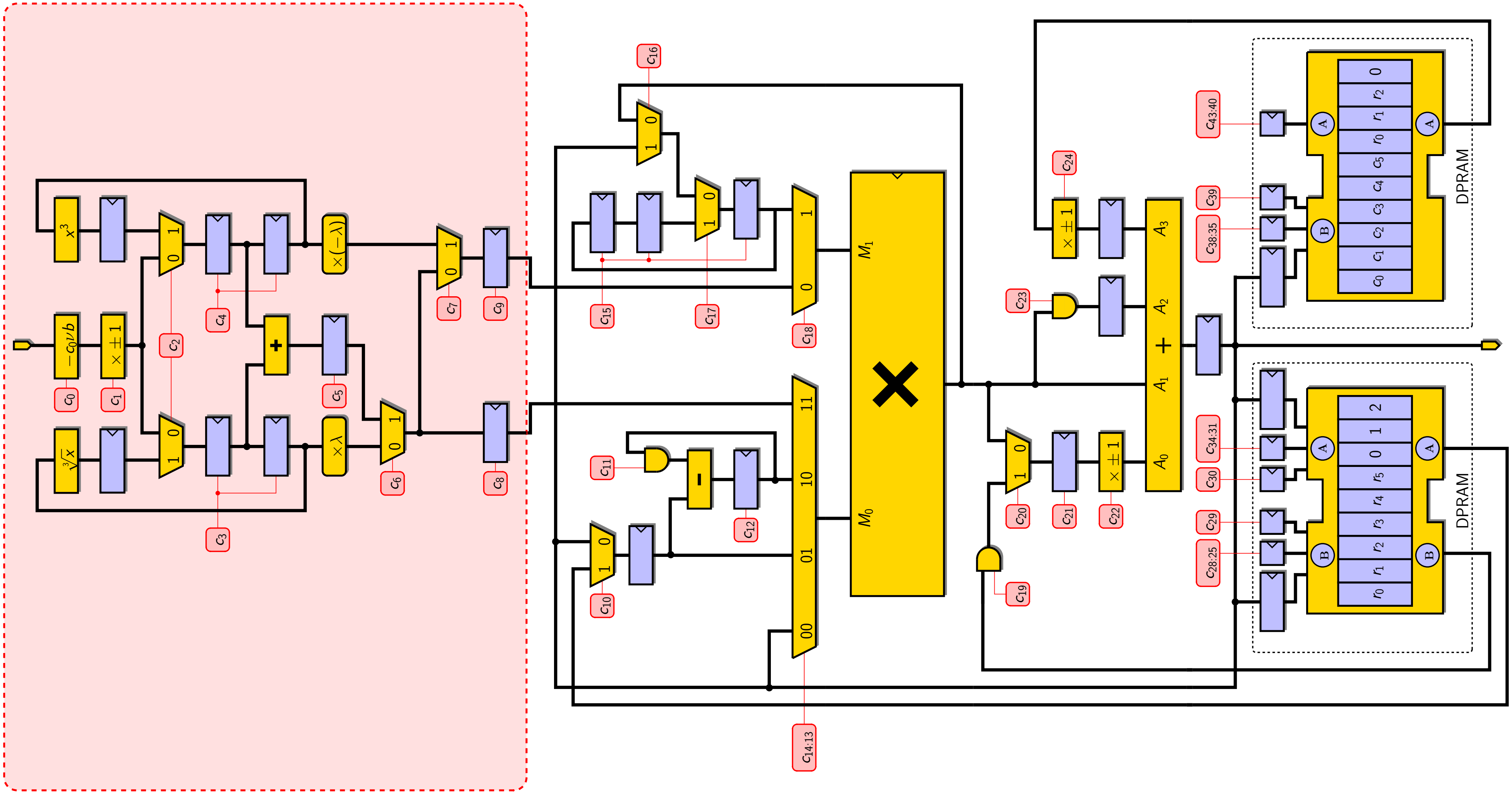


- Perfectly tight scheduling: no idle cycle
- 17 clock cycles per iteration ⇒ $17(m+1)/2$ cycles for the complete η_T pairing

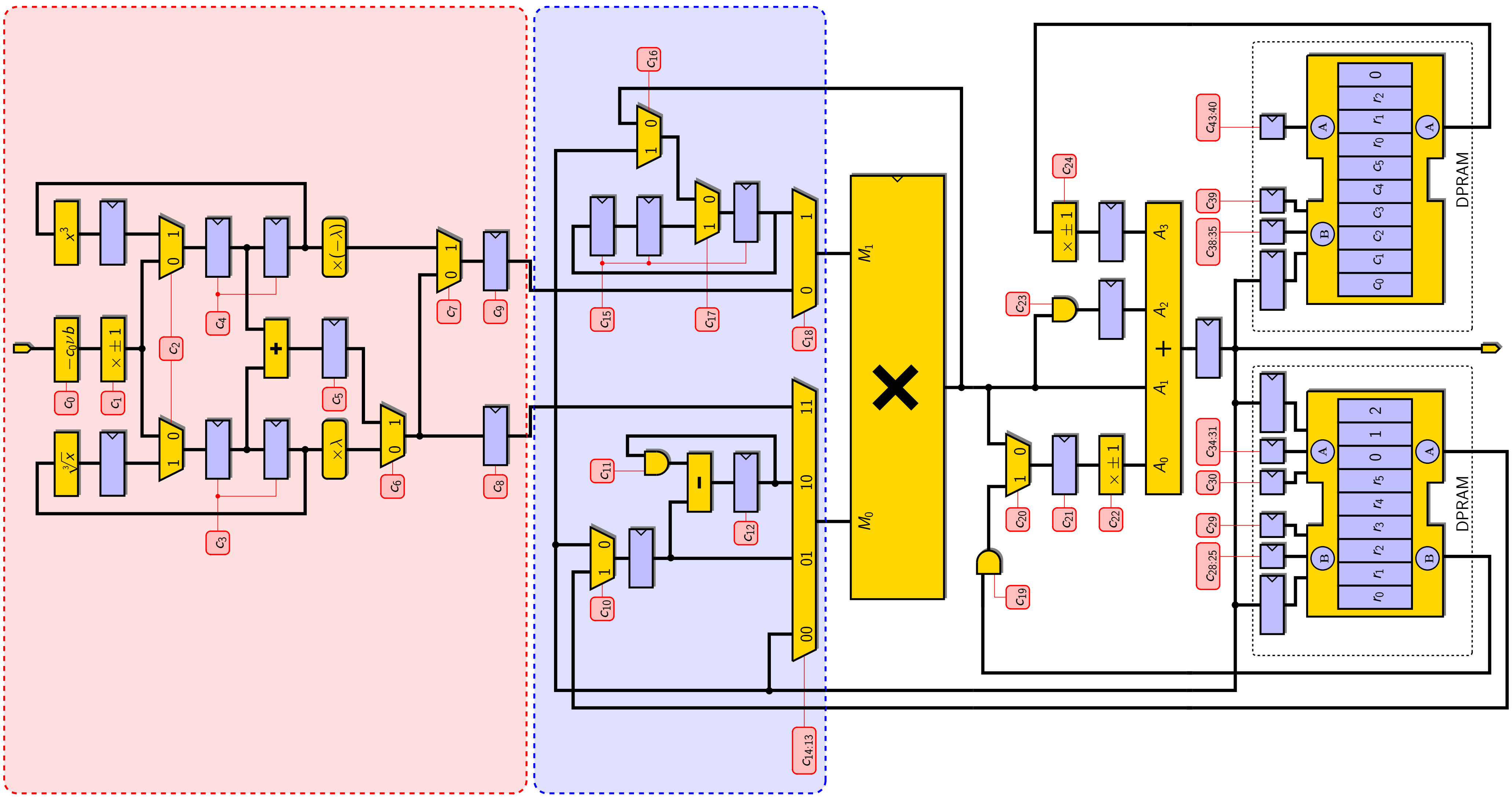
A parallel operator for the η_T pairing



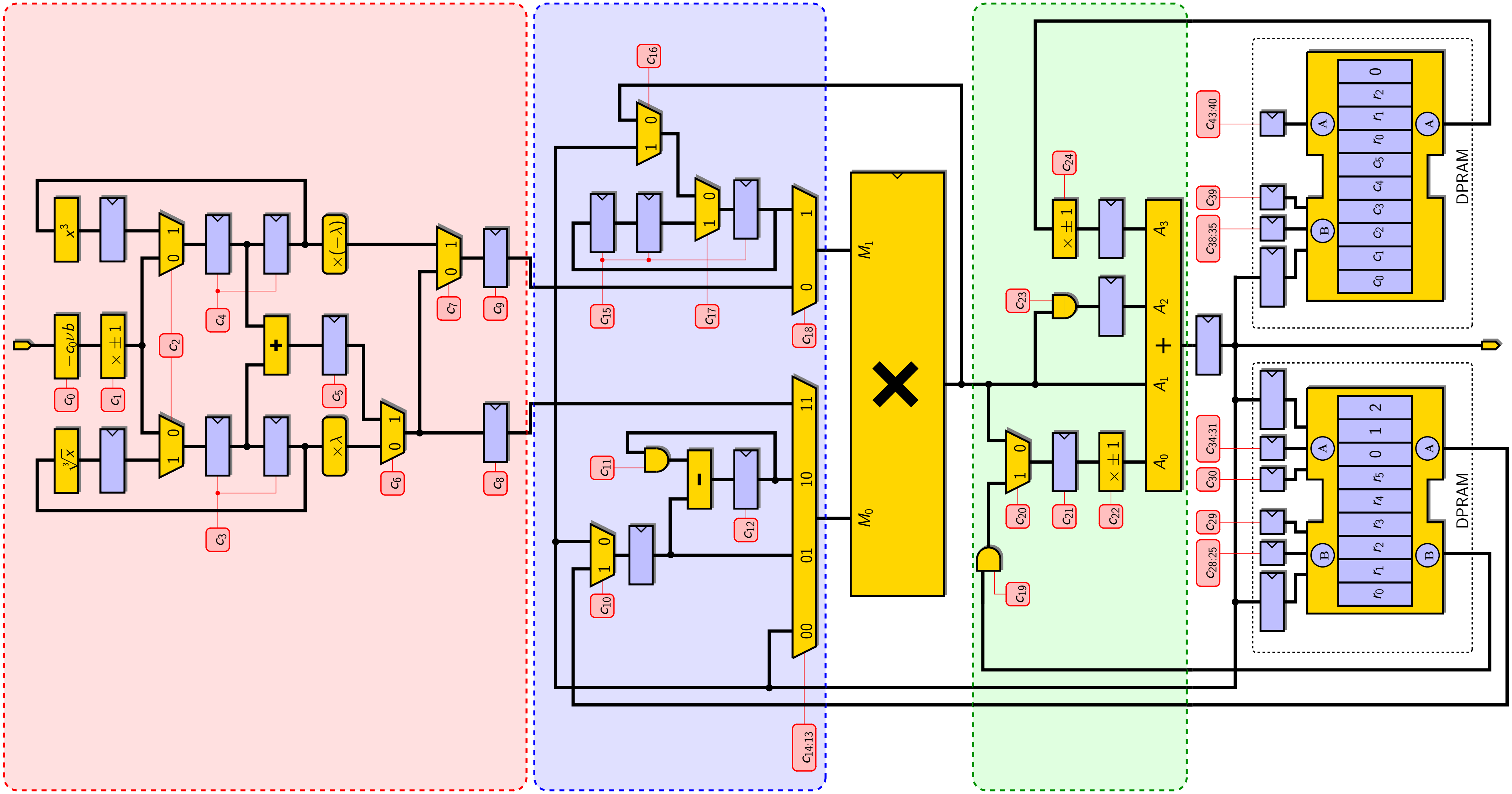
A parallel operator for the η_T pairing



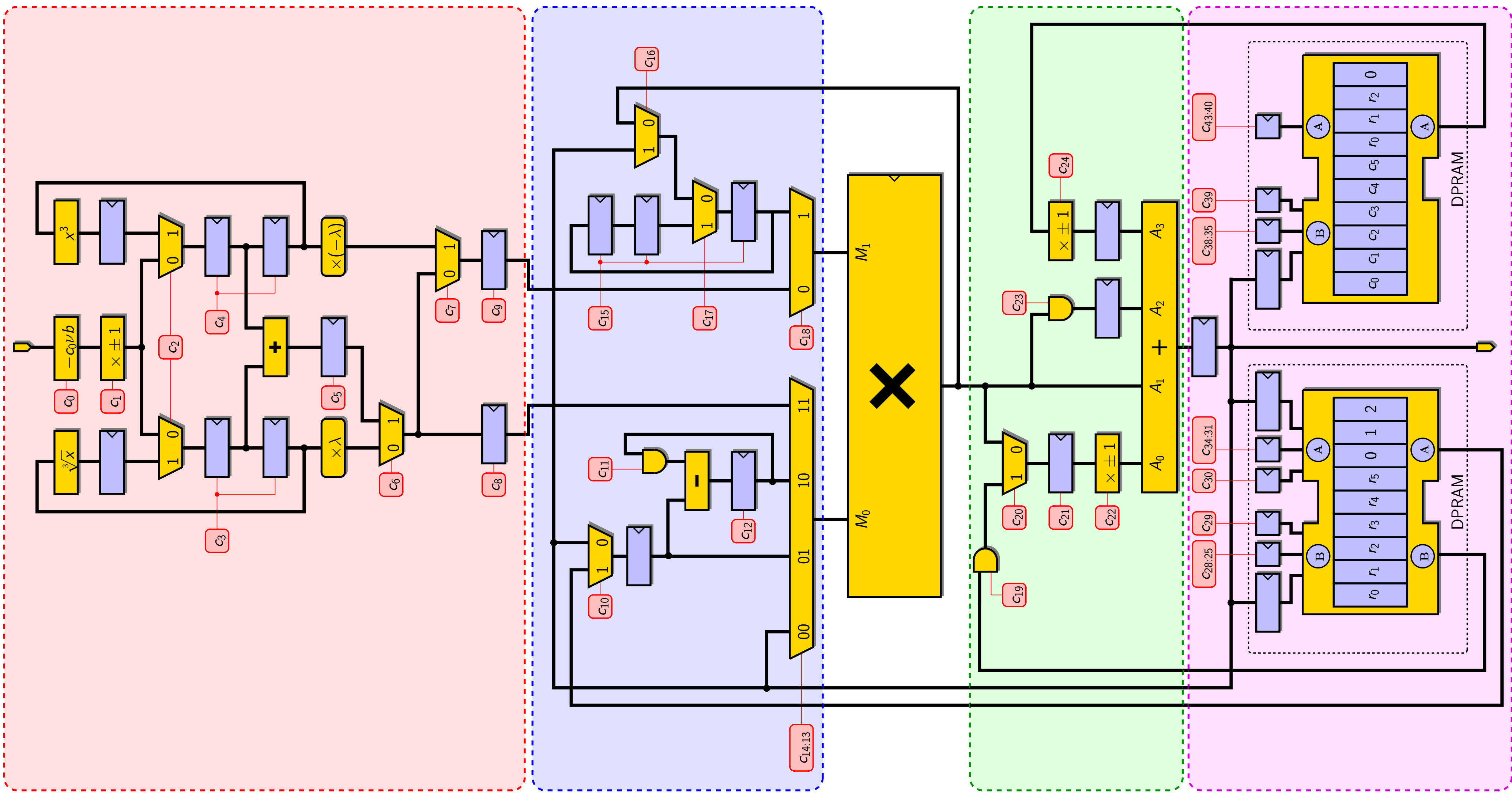
A parallel operator for the η_T pairing



A parallel operator for the η_T pairing



A parallel operator for the η_T pairing



Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation (in characteristic 3)
- Implementation results
- Concluding thoughts

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^{\times}$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^{\times}$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73\,\times$, $3m + 3$ Frobenius, $3m + 175\,+$, and 1 inversion

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^{\times}$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73\,\times$, $3m + 3$ Frobenius, $3m + 175\,+$, and 1 inversion ($\sim \log m\,\times$ and $m - 1$ Frobenius)

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^\times$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73\times$, $3m + 3$ Frobenius, $3m + 175+$, and 1 inversion ($\sim \log m \times$ and $m - 1$ Frobenius)

- Cost of the η_T pairing:
  - $(m + 1)/2$ iterations
  - $17\times$, 10 Frobenius and $38+$ over $\mathbb{F}_{3^m}$ per iteration

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^\times$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73\,\times$, $3m + 3$ Frobenius, $3m + 175\,+$, and 1 inversion ($\sim \log m\,\times$ and $m - 1$ Frobenius)

- Cost of the η_T pairing:
  - $(m + 1)/2$ iterations
  - $17\,\times$, 10 Frobenius and $38\,+$ over $\mathbb{F}_{3^m}$ per iteration

- The final exponentiation is much cheaper than the η_T pairing

- Challenge for the final exponentiation:
  - computation in the same time as the η_T pairing
  - ... using as few resources as possible

The final exponentiation

- First idea: use the unified operator
  - the smallest architecture supporting all the required operations over $\mathbb{F}_{3^m}$
  - purely sequential scheduling

The final exponentiation

- First idea: use the unified operator
  - the smallest architecture supporting all the required operations over $\mathbb{F}_{3^m}$
  - purely sequential scheduling
- Example for $m = 97$:
  - computation in 1430 clock cycles

The final exponentiation

- First idea: use the unified operator
  - the smallest architecture supporting all the required operations over $\mathbb{F}_{3^m}$
  - purely sequential scheduling
- Example for $m = 97$:
  - computation in 1430 clock cycles
  - ... but 833 clock cycles for the η_T pairing

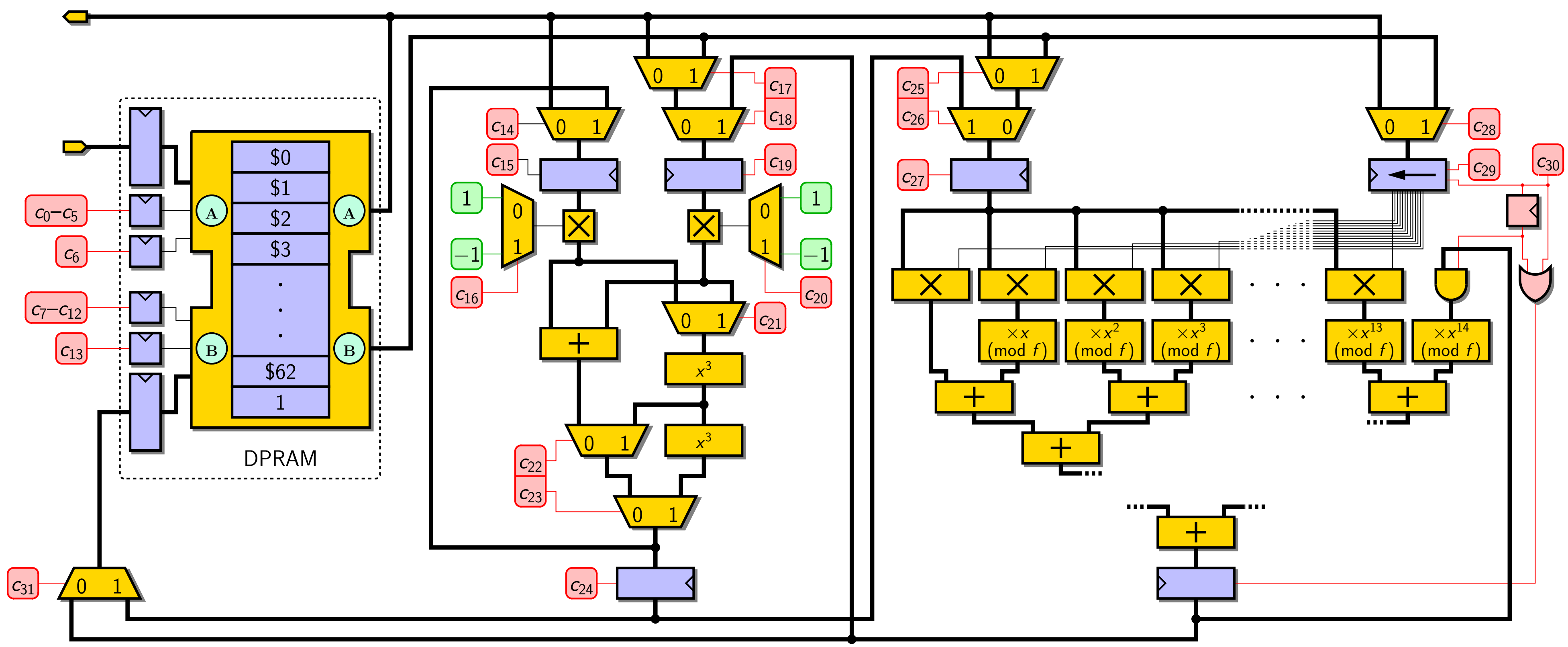
The final exponentiation

- First idea: use the unified operator
  - the smallest architecture supporting all the required operations over $\mathbb{F}_{3^m}$
  - purely sequential scheduling
- Example for $m = 97$:
  - computation in 1430 clock cycles
  - ... but 833 clock cycles for the η_T pairing
- Some parallelism is required

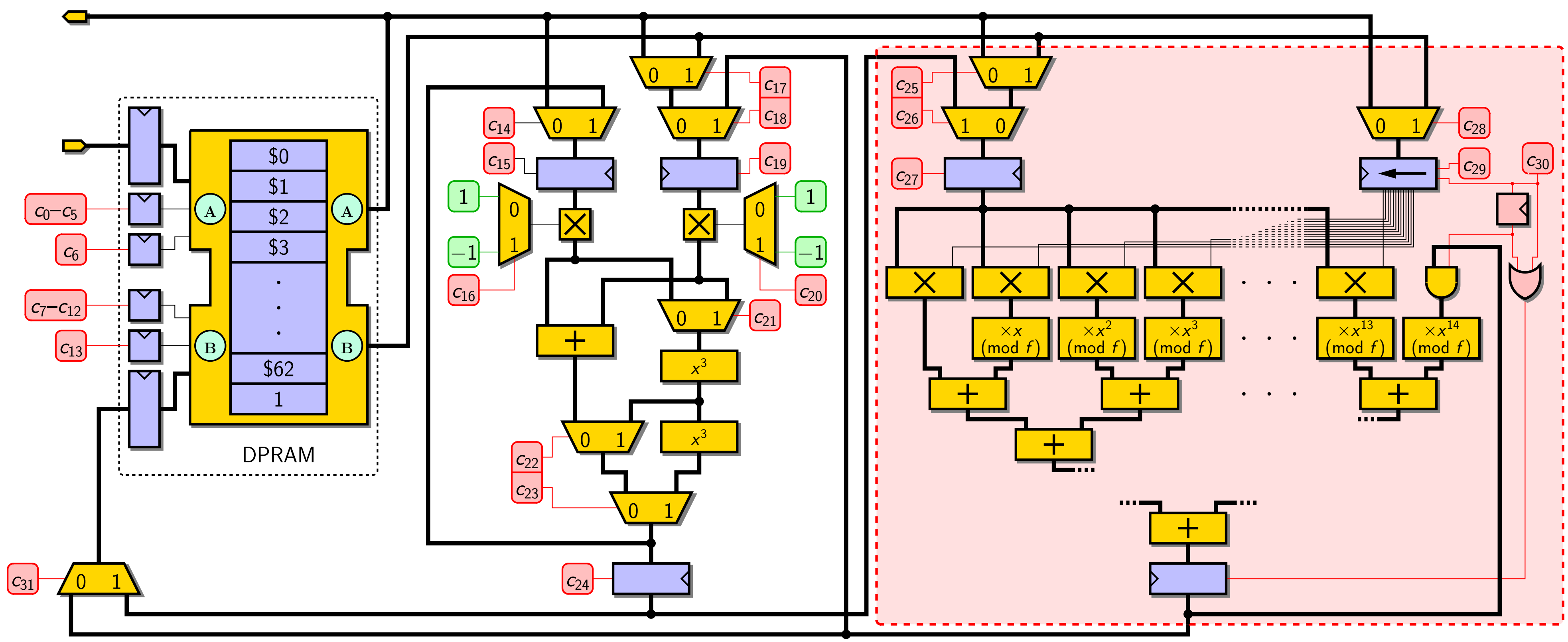
The final exponentiation

- First idea: use the unified operator
  - the smallest architecture supporting all the required operations over $\mathbb{F}_{3^m}$
  - purely sequential scheduling
- Example for $m = 97$:
  - computation in 1430 clock cycles
  - ... but 833 clock cycles for the η_T pairing
- Some parallelism is required
- New coprocessor with two arithmetic units:
  - a standalone multiplier, based on a parallel-serial scheme
  - a unified operator supporting addition/subtraction, Frobenius map and double Frobenius map

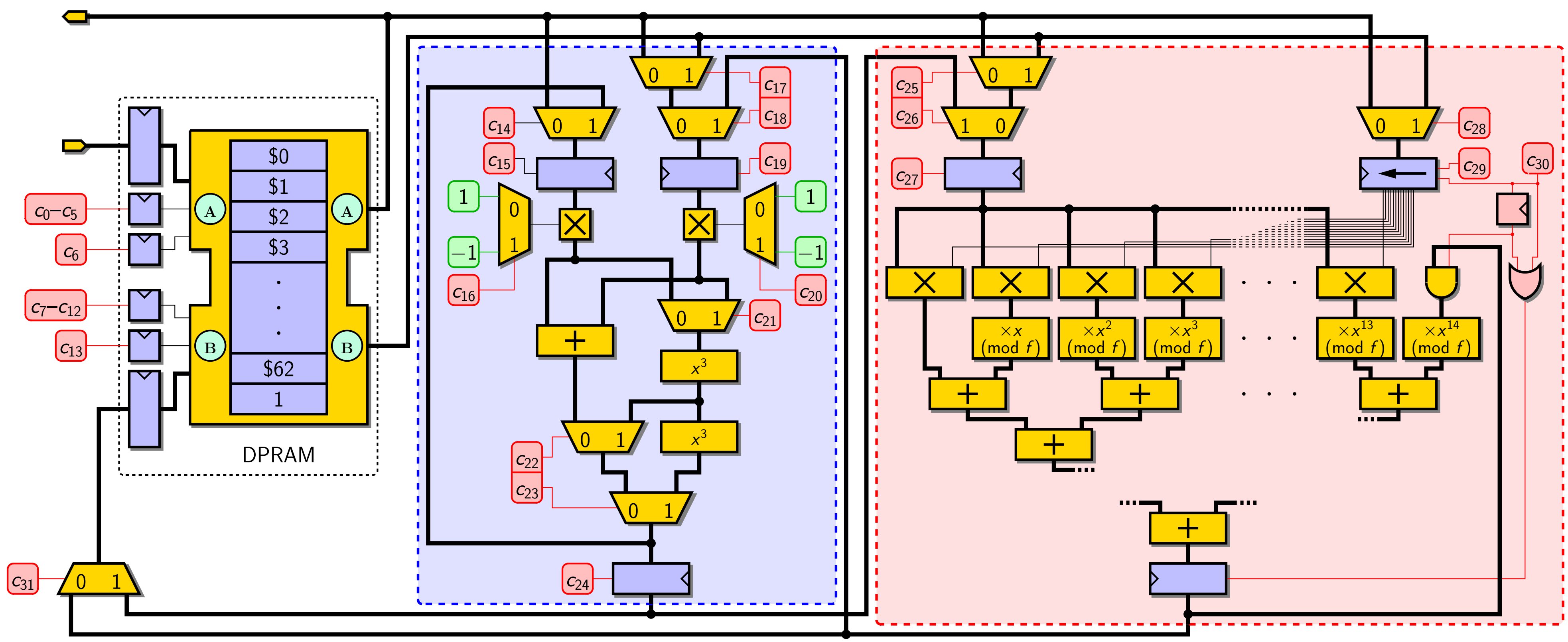
A coprocessor for the final exponentiation



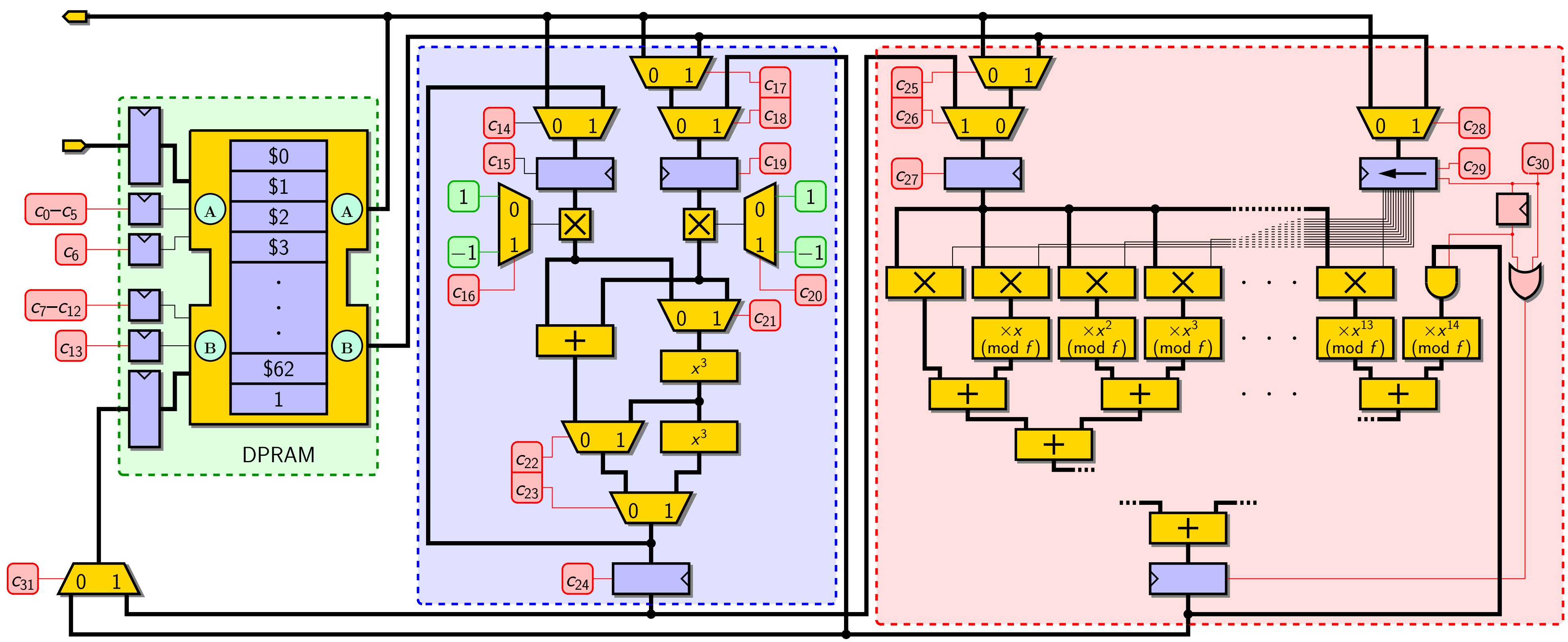
A coprocessor for the final exponentiation



A coprocessor for the final exponentiation



A coprocessor for the final exponentiation



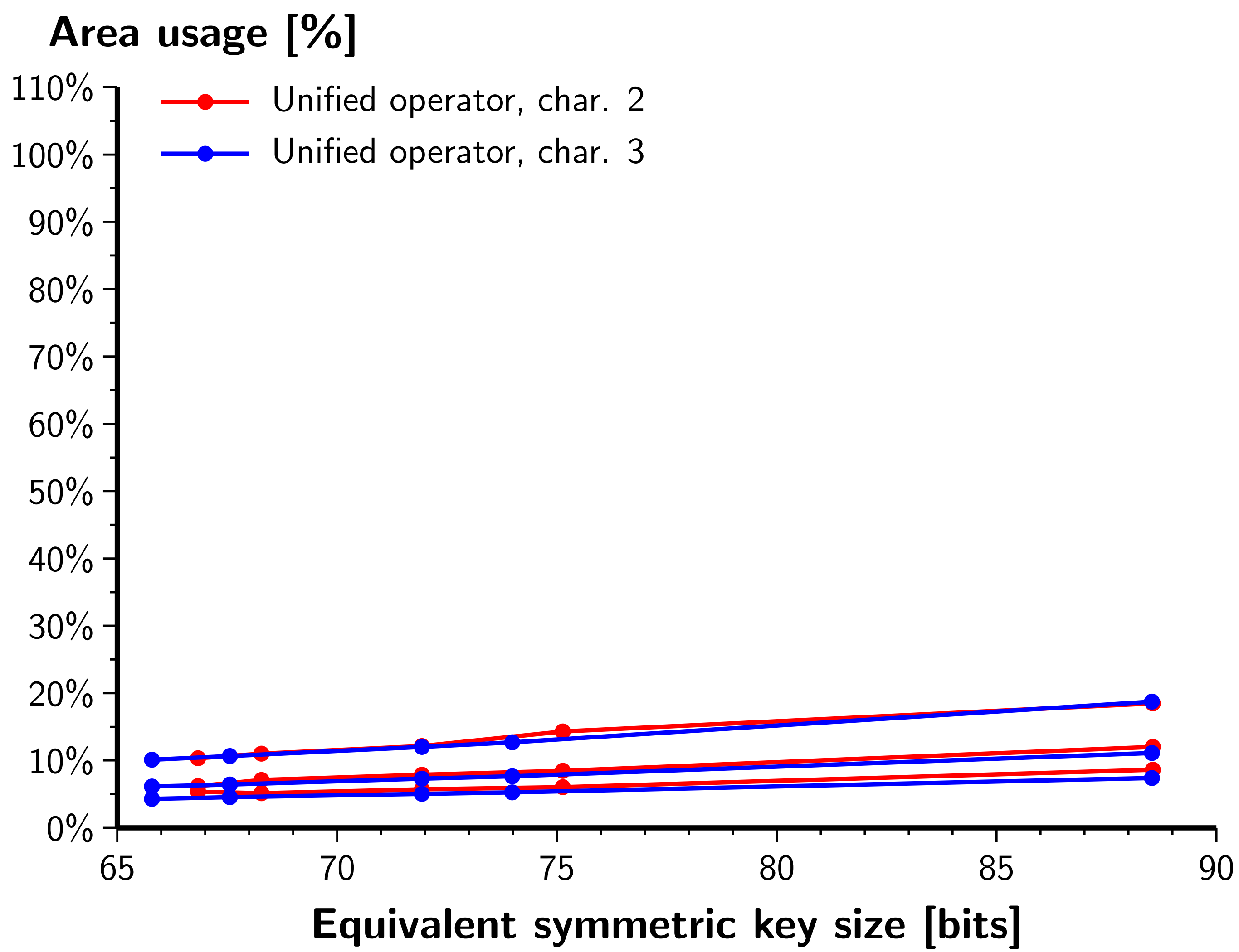
Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

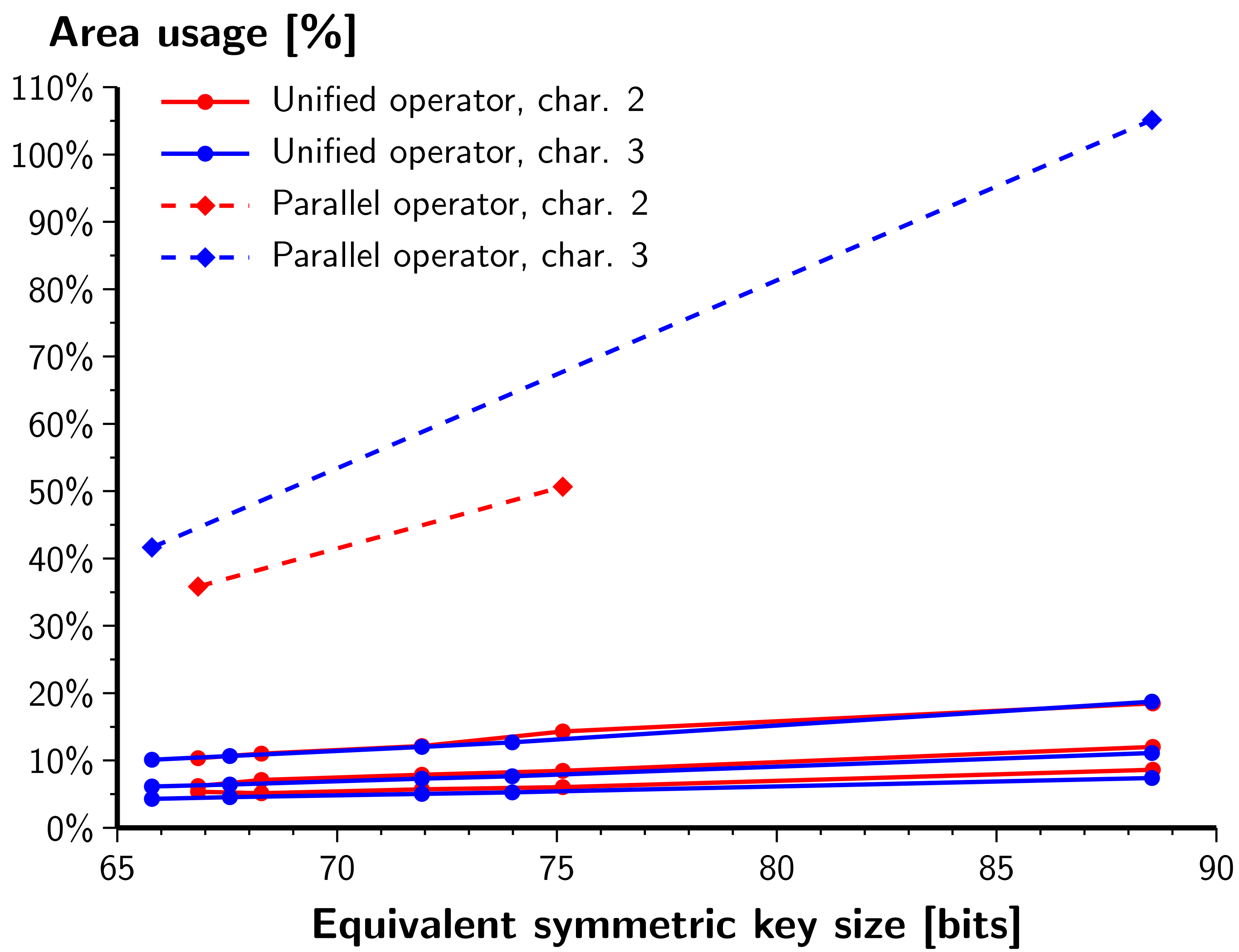
Experimental setup

- Full coprocessor for computation of the Tate pairing
- Architecture based on the two parallel accelerators
- Prototyped on a Xilinx Virtex-II Pro 50 FPGA (larger model)
- Post place-and-route results: area, computation time, AT product

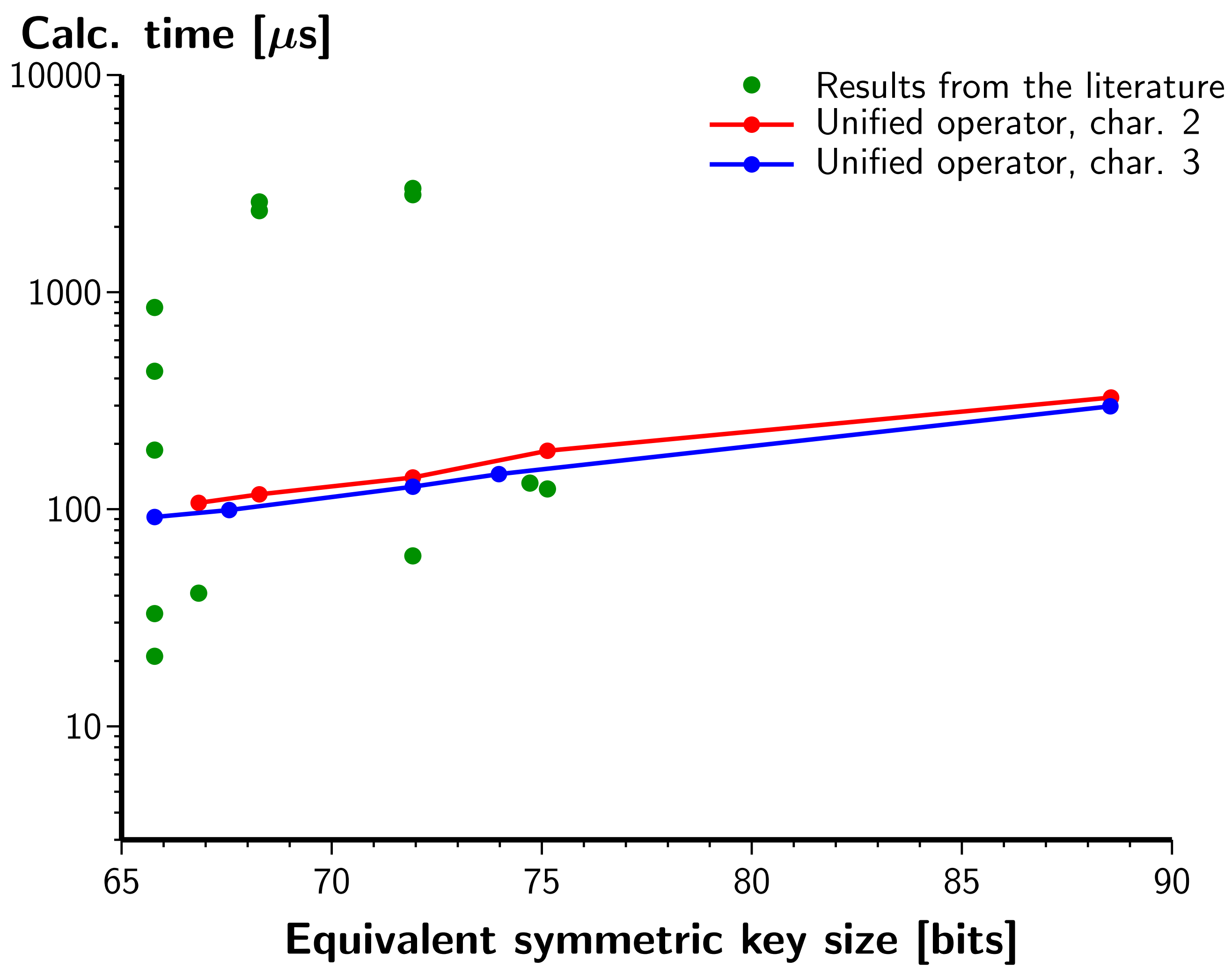
Coprocessor area



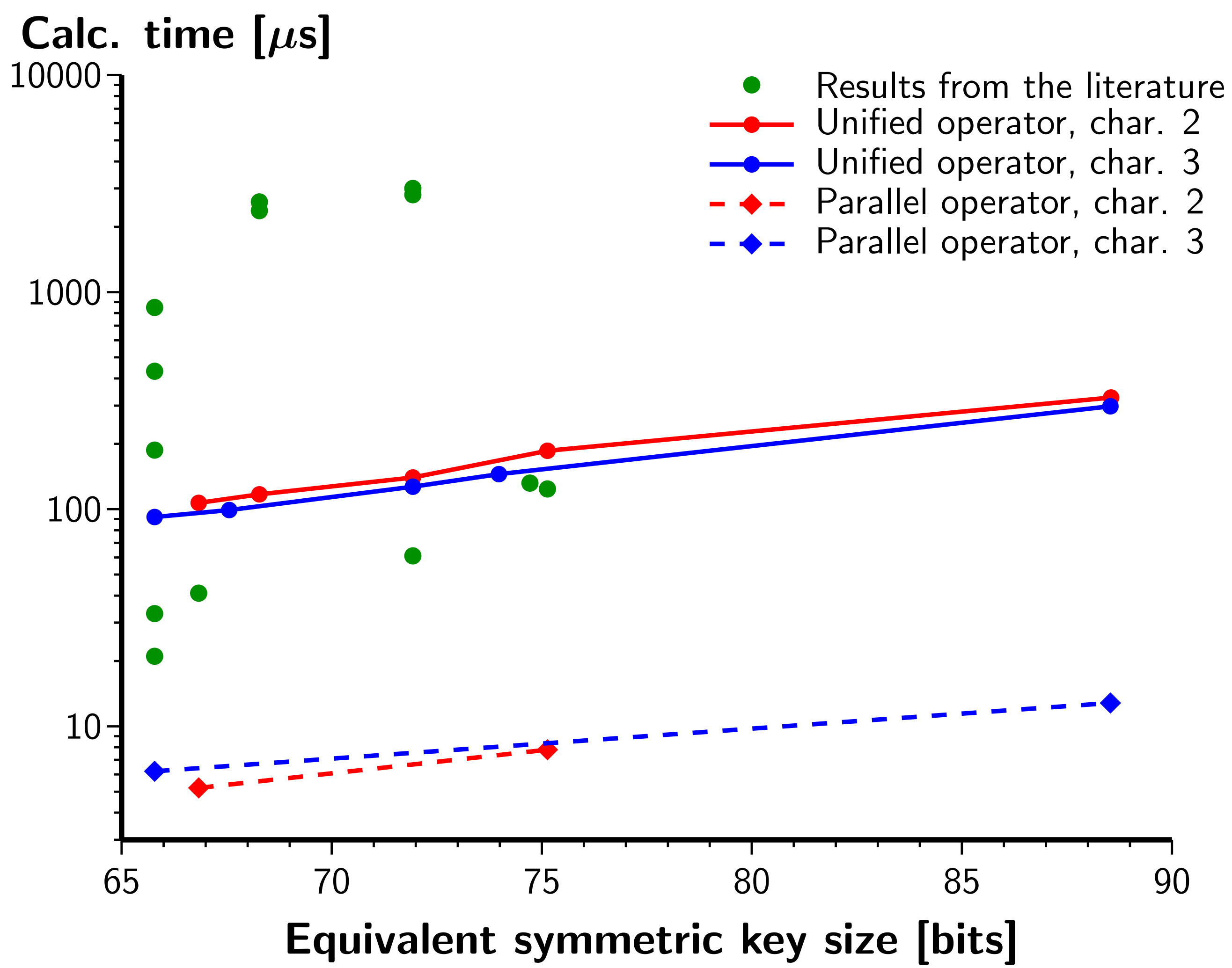
Coprocessor area



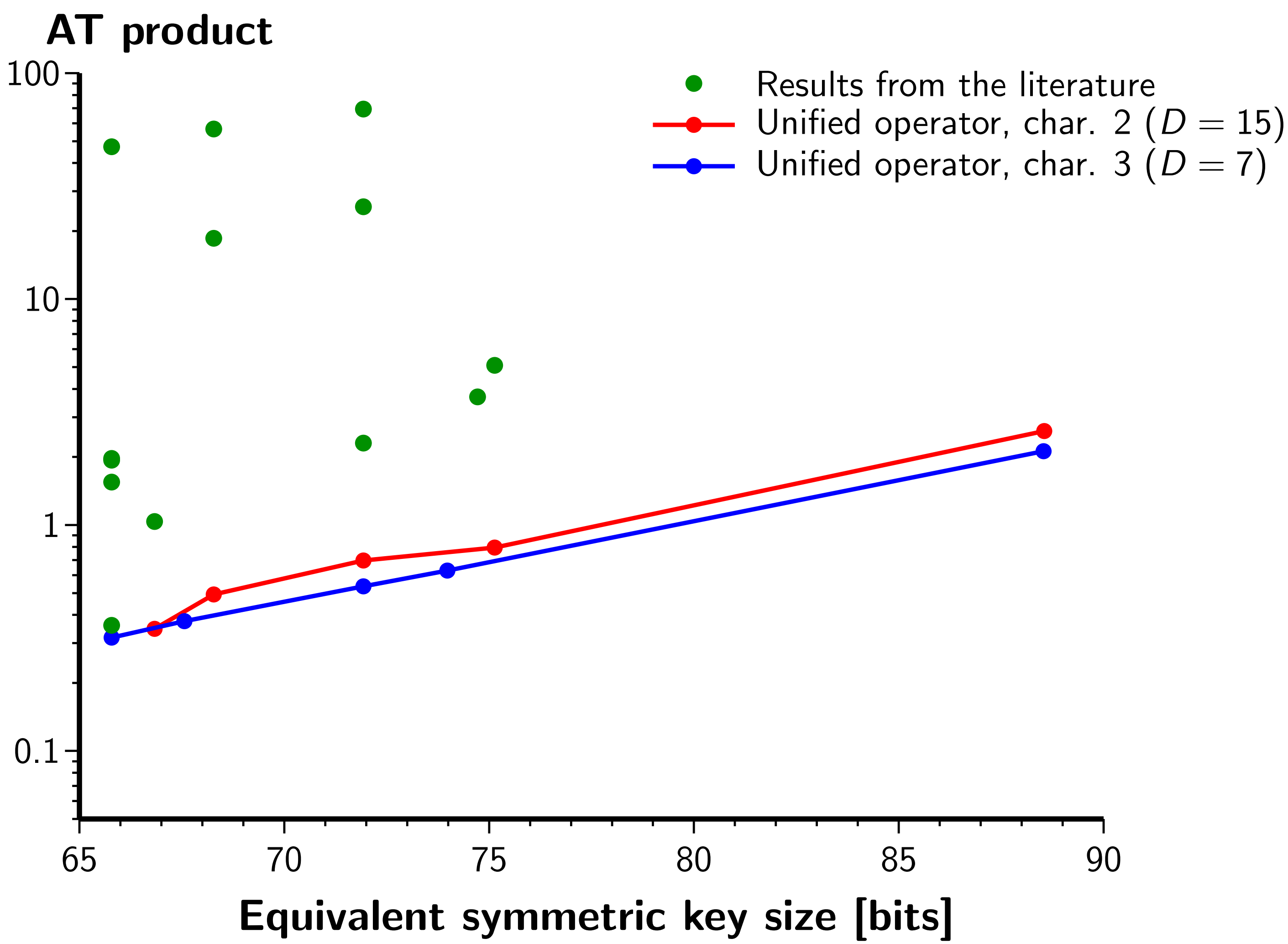
Calculation time



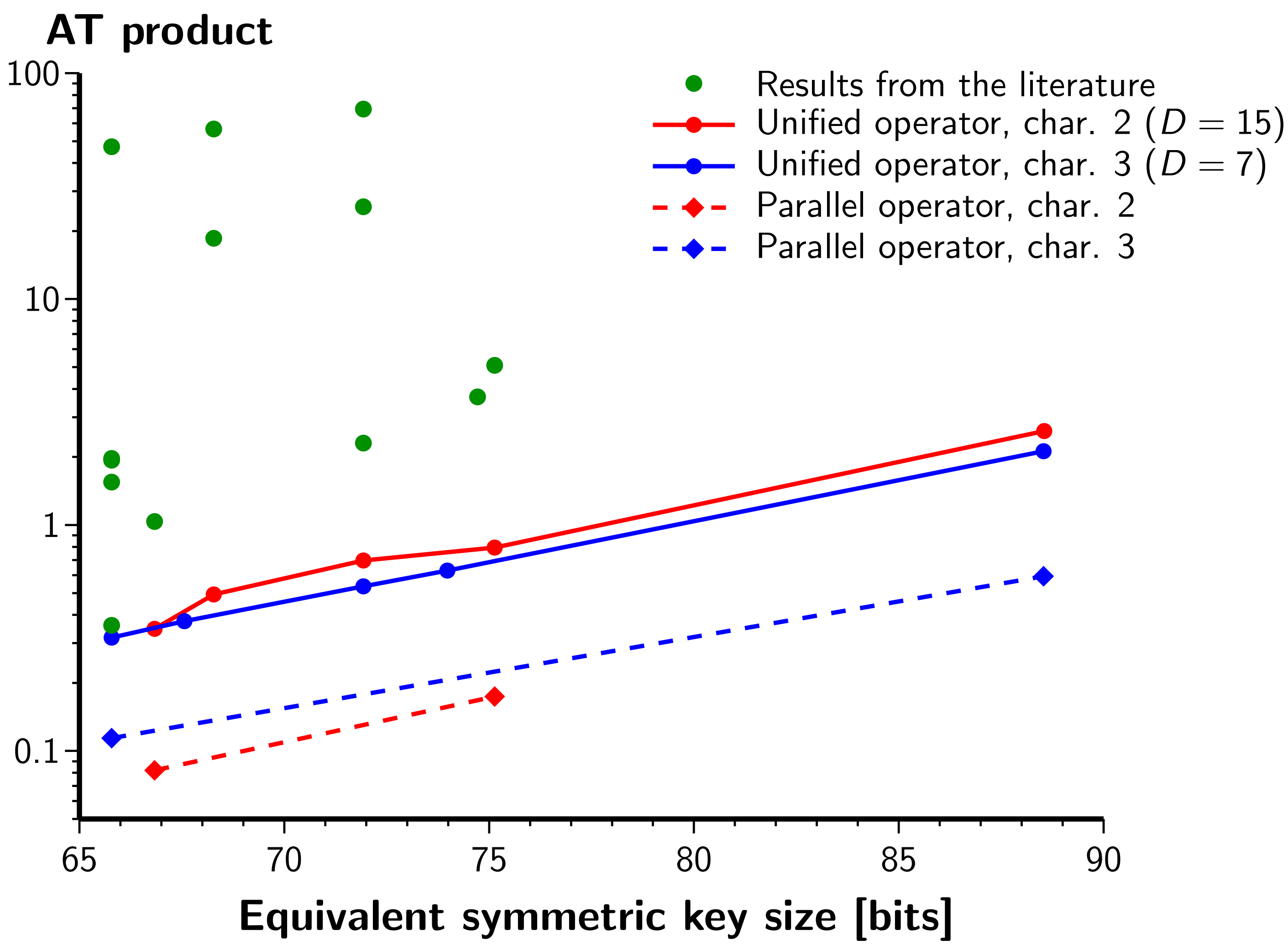
Calculation time



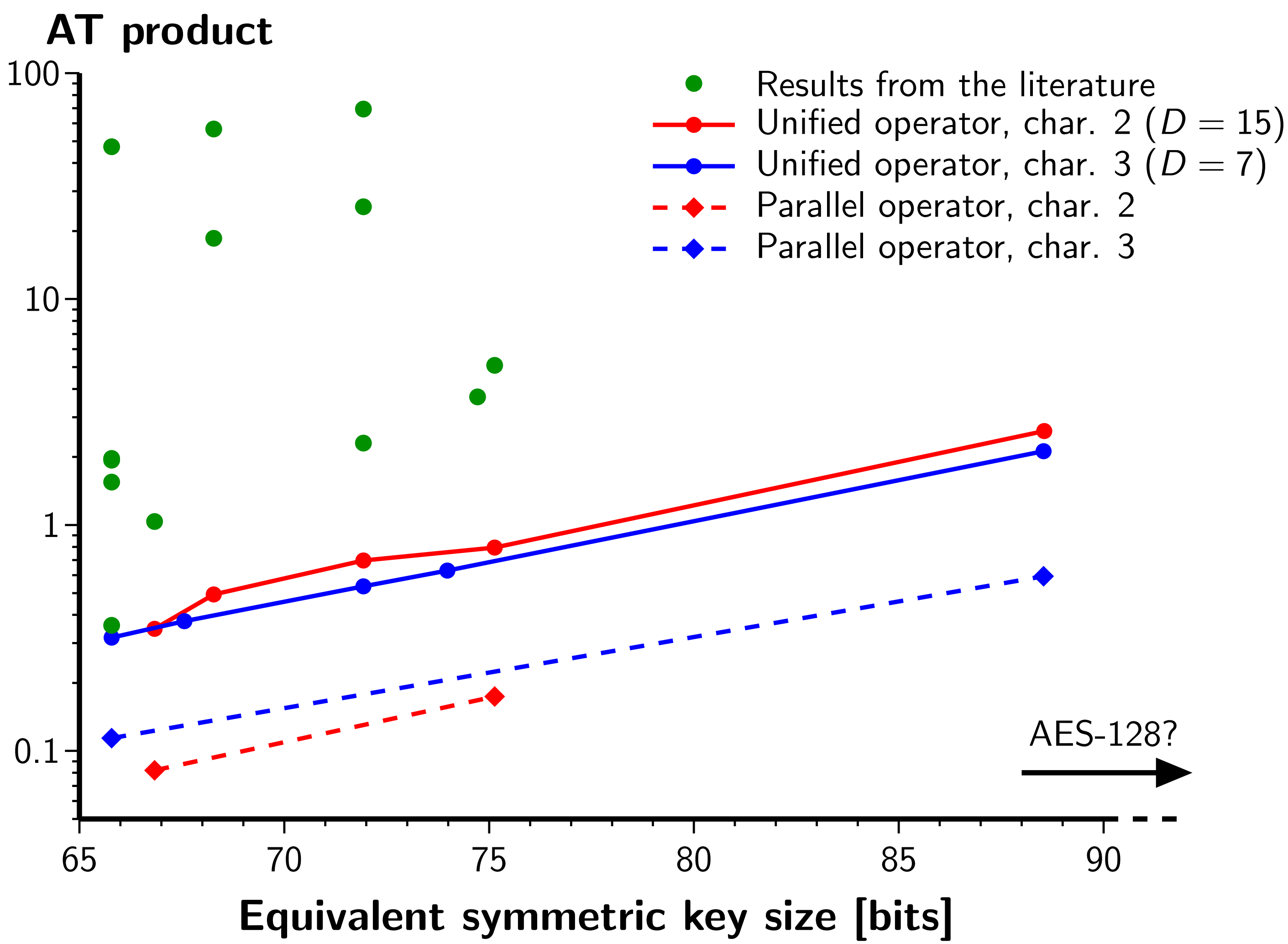
Area-time product



Area-time product



Area-time product



Outline of the talk

- Previously in the Jean-Luc Beuchat Tour
- A closer look at the algorithm
- Accelerating the η_T pairing
- Accelerating the final exponentiation
- Implementation results
- Concluding thoughts

Conclusion and perspectives

- The fastest implementation of the literature!

Conclusion and perspectives

- The fastest implementation of the literature!
- Importance of the adequation between algorithm and architecture

Conclusion and perspectives

- The fastest implementation of the literature!
- Importance of the adequation between algorithm and architecture
- Scalability? **AES-128?**

With thanks to our sponsor

Thank you for your attention

Questions?