Support logiciel pour l'arithmétique flottante simple précision sur processeurs entiers

Guillaume Revy
Encadrants : Claude-Pierre Jeannerod et Gilles Villard

Équipe INRIA Arénaire
Laboratoire de l'Informatique du Parallélisme - ENS Lyon

Groupe de travail **Méthodes Ensemblistes pour l'Automatique**

ENSAM - Paris, 22 novembre 2007

Introduction

- Systèmes embarqués pour l'audio et la vidéo
  - ⇒ traitement du signal, de l'image, HD-IPTV, téléphonie mobile, ...
- Utilisation de processeurs entiers : pas d'unité flottante (FPU)
  - ⇒ diminution des coûts : petite surface
  - ⇒ rapidité / faible consommation d'énergie

⇒ Calcul entier ou virgule fixe

Introduction

- Systèmes embarqués pour l'audio et la vidéo
  - ⇒ traitement du signal, de l'image, HD-IPTV, téléphonie mobile, ...
- Utilisation de processeurs entiers : pas d'unité flottante (FPU)
  - ⇒ diminution des coûts : petite surface
  - ⇒ rapidité / faible consommation d'énergie

⇒ Calcul entier ou virgule fixe

- Mise à l'échelle *flottant* ⇔ *entier/virgule fixe*
  - ⇒ coûteuse / délicate (perte de précision)

⇒ Émulation d'une couche d'arithmétique flottante efficace

Introduction

Processeurs entiers (opérations entières : addition, multiplication, décalage...)

Introduction

Applications qui manipulent des nombres flottants

Processeurs entiers (opérations entières : addition, multiplication, décalage...)

Introduction

Applications qui manipulent des nombres flottants

valeurs flottantes manipulées

FLIP (support logiciel pour l'arithmétique flottante)

opérations entières

Processeurs entiers (opérations entières : addition, multiplication, décalage...)

Contexte et objectifs

Contexte

- Support logiciel pour l'arithmétique flottante simple précision (IEEE-754) aux processeurs entiers (FLIP)

  `https://lipforge.ens-lyon.fr/projects/flip/`

- Fonctions mathématiques rapides et précises

$$+, -, \times, /, \sqrt{}, 1/\sqrt{}, \ldots$$

- Collaboration avec STMicroelectronics Compilation Expertise Centre (Grenoble)

  $\Rightarrow$ FLIP : développée et optimisée pour les processeurs de la famille ST200
  $\Rightarrow$ utiliser au mieux l'architecture disponible

Contexte et objectifs

Objectifs pour chaque fonction de FLIP

- Qualité visée : arrondi correct au plus près, sans nombres dénormalisés
- Impact dû
  - $\longrightarrow$ à l'ajout d'autre modes d'arrondi
  - $\longrightarrow$ à la prise en compte des nombres dénormalisés
- Extension à d'autres formats : *medium/high precision* (OpenGL ES)

Plan de l'exposé

Plan de l'exposé

Bibliothèque FLIP

- Support logiciel pour l'arithmétique flottante simple précision (IEEE-754)
  - $\Rightarrow$ émulation d'une couche flottante
- Bibliothèque développée en langage C standard
  - $\Rightarrow$ nécessité d'un compilateur C
- Ensemble de fonctions
  - $\rightarrow$ avec ou sans nombres dénormalisés,
  - $\rightarrow$ correctement arrondies suivant les 4 modes d'arrondi (au plus près, vers $-\infty$, vers $+\infty$ et vers 0)

Utilisation de la bibliothèque FLIP

Application utilisant l'arithmétique flottante simple précision (IEEE-754)

$+, -, \times, /$ `flip_add`, `flip_sqrt`...

Transformation des opérations en fonctions

ie : transformation de $+$ en un appel à `__adds` (sur ST200)

Conversion des nombres flottants vers leur représentation binaire

FLIP : appel aux fonctions "entières" de la bibliothèque

ie : `__float32_flip_sqrt_float32`

Processeurs entiers

$\Rightarrow$ manipulation d'opérations entières : addition, multiplication, décalage, ...

Utilisation de la bibliothèque FLIP

Application utilisant l'arithmétique flottante simple précision (IEEE-754)

$+, -, \times, /$ `flip_add`, `flip_sqrt`...

↓

Transformation des opérations en fonctions

ie : transformation de $+$ en un appel à `__adds` (sur ST200)

Conversion des nombres flottants vers leur représentation binaire

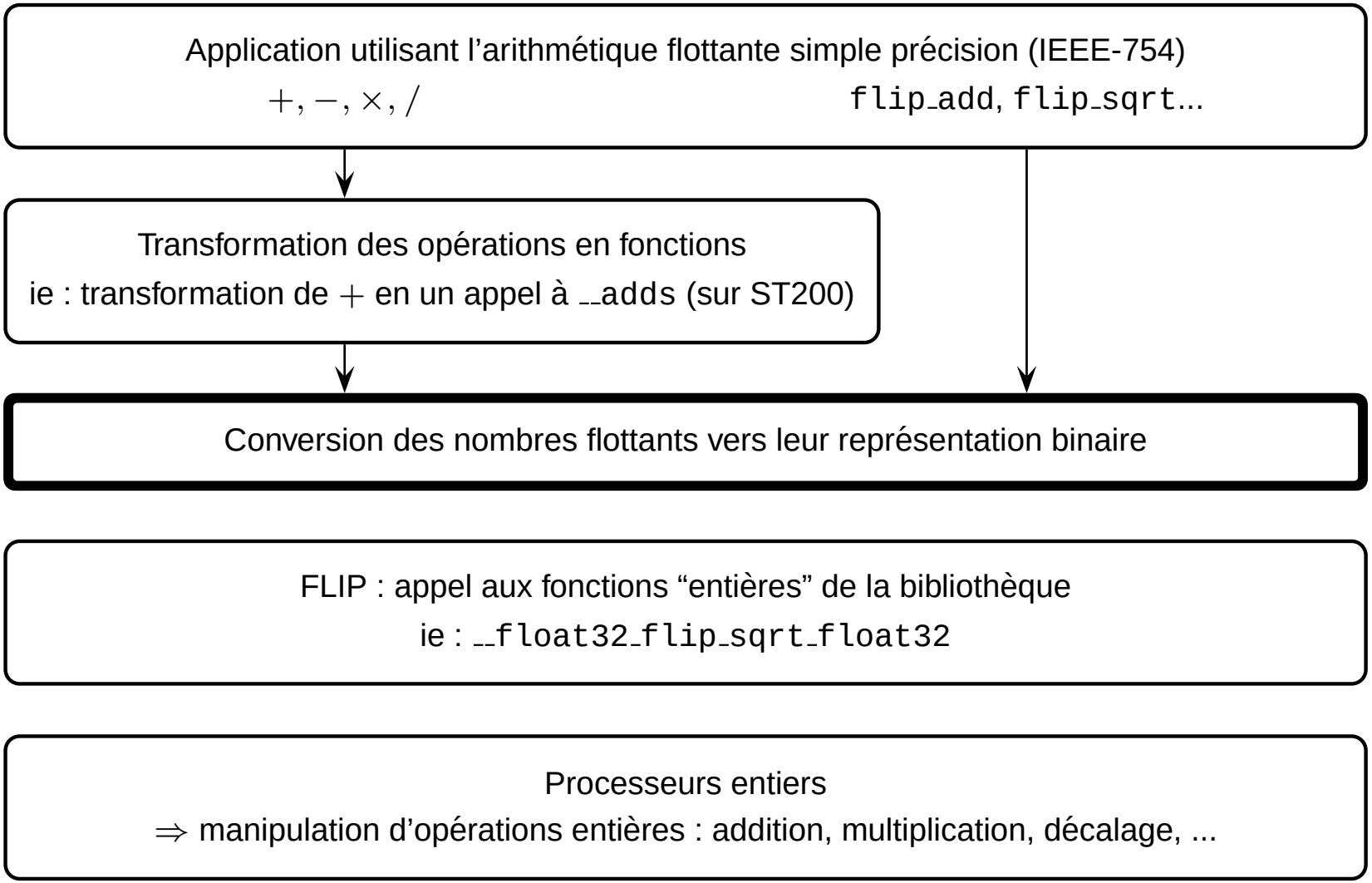
FLIP : appel aux fonctions "entières" de la bibliothèque

ie : `__float32_flip_sqrt_float32`

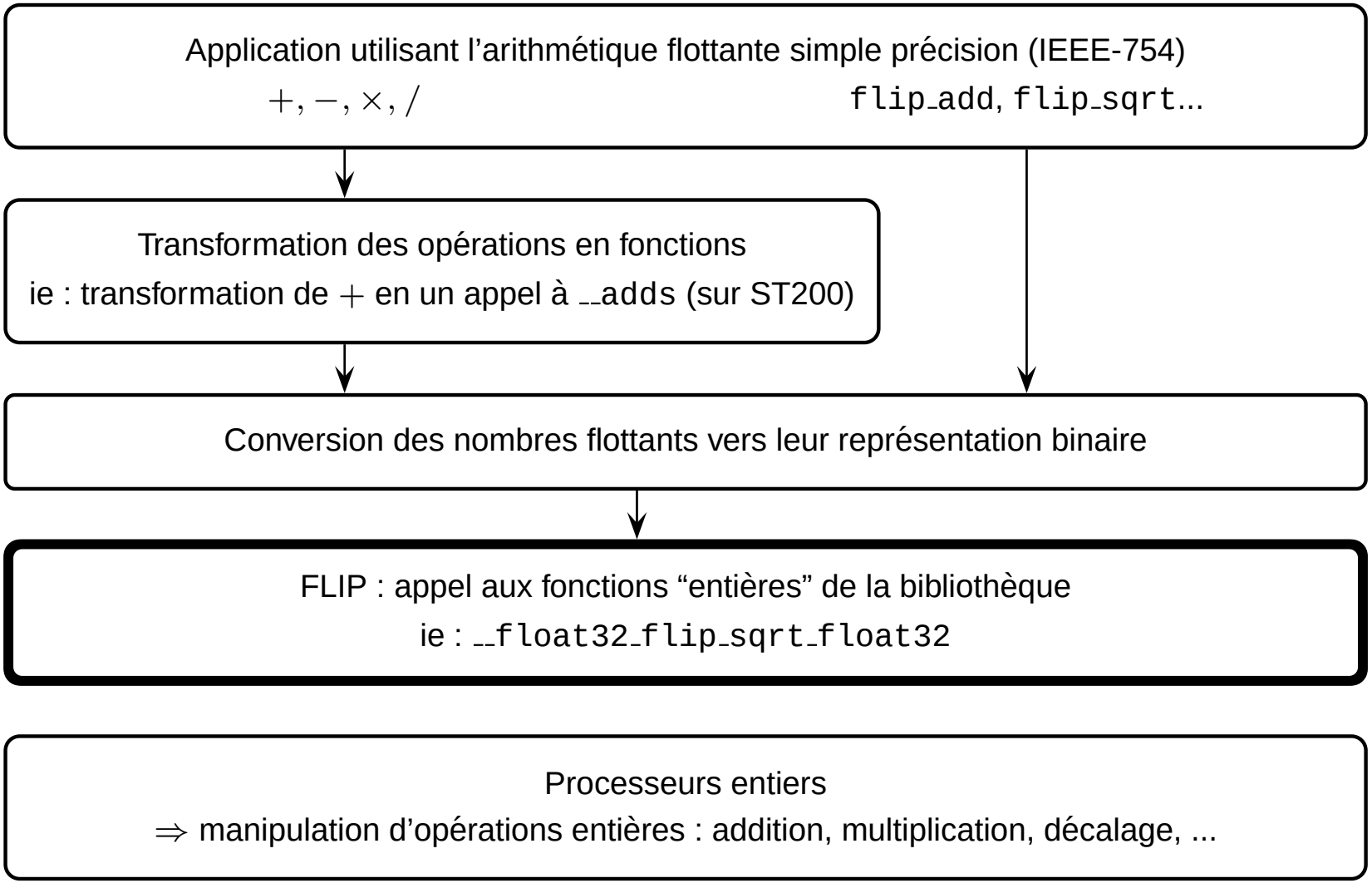
Processeurs entiers

$\Rightarrow$ manipulation d'opérations entières : addition, multiplication, décalage, ...

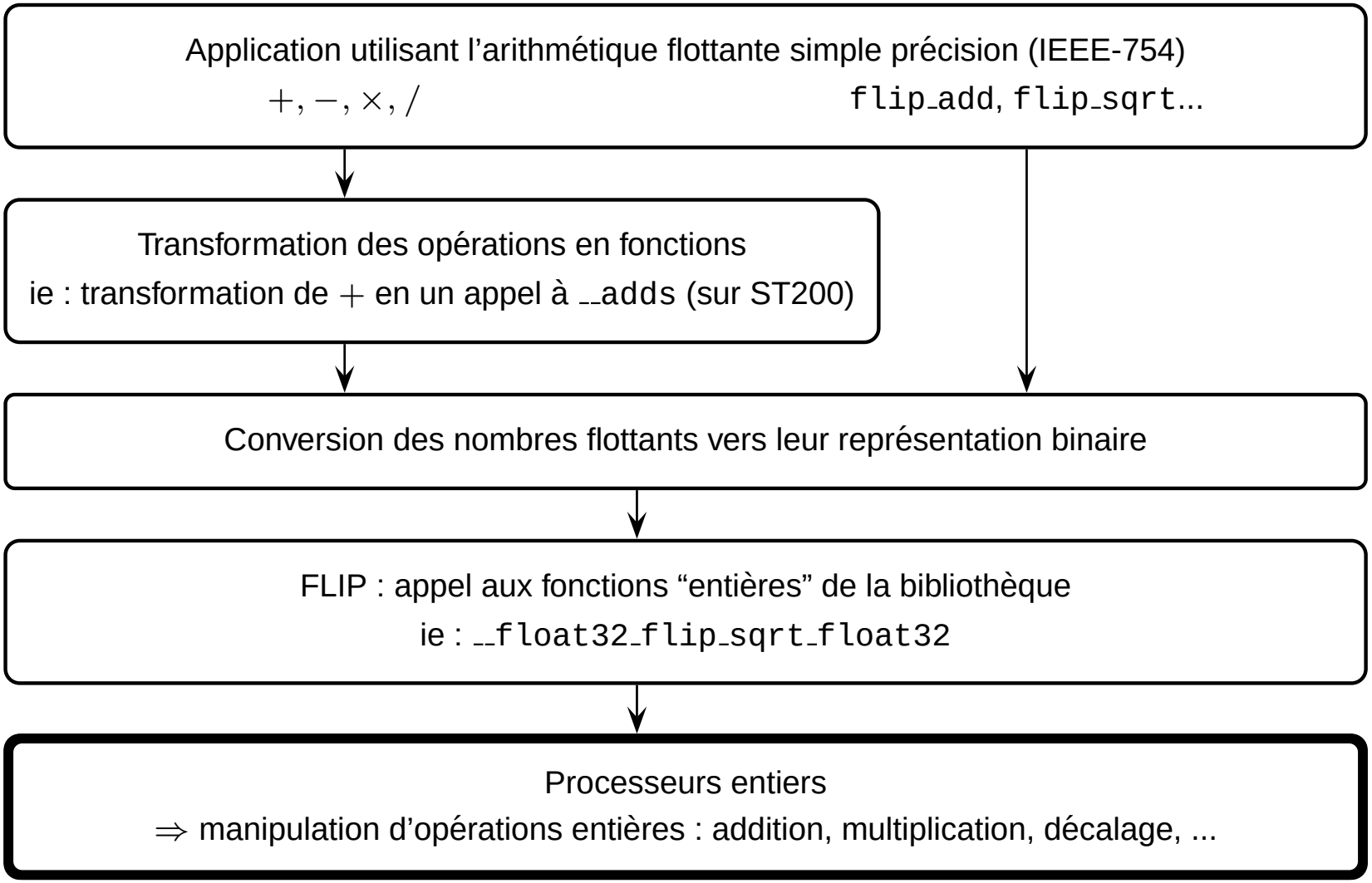
Utilisation de la bibliothèque FLIP



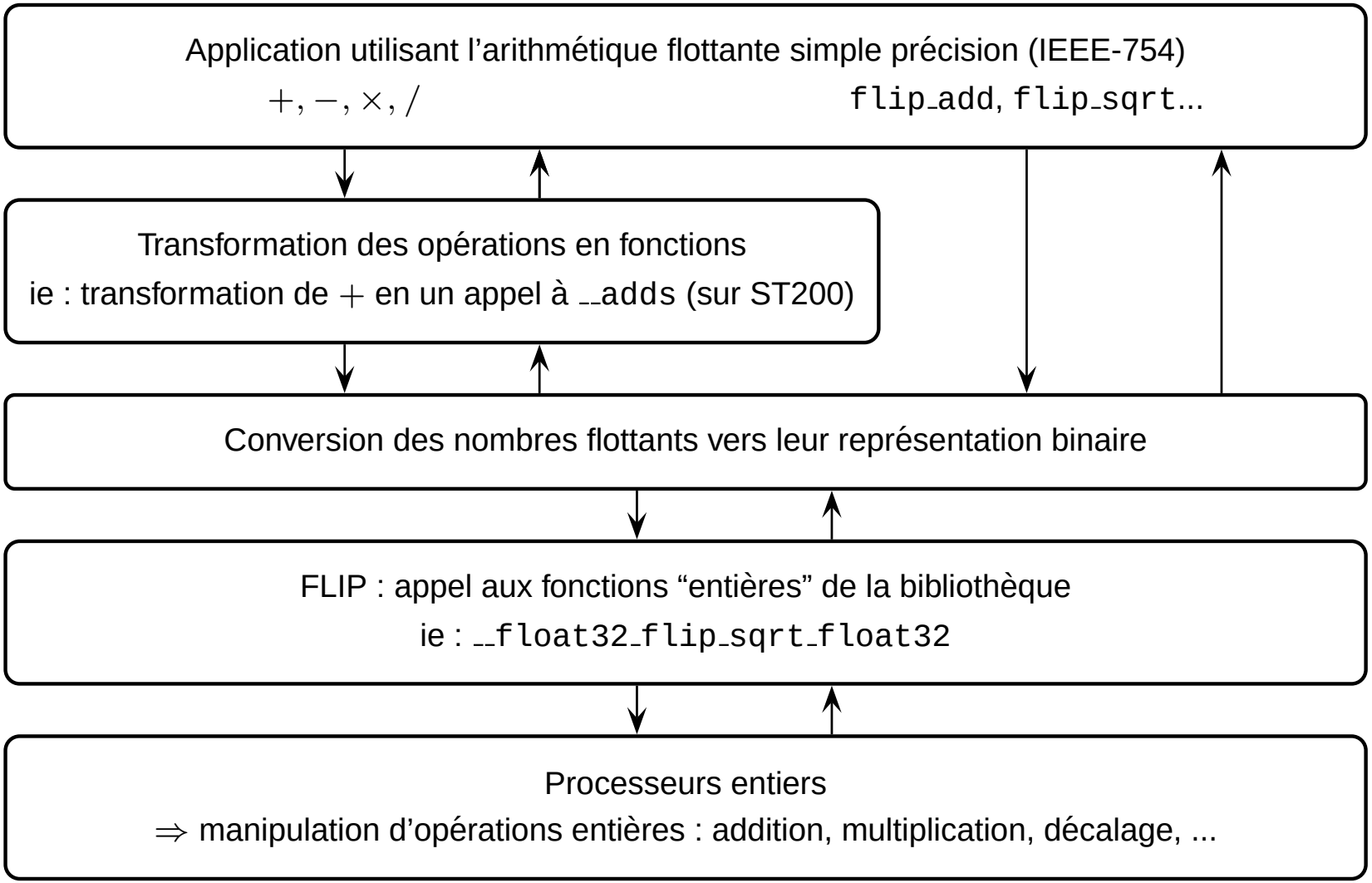
Utilisation de la bibliothèque FLIP



Utilisation de la bibliothèque FLIP



Utilisation de la bibliothèque FLIP



Nombres flottants et représentation binaire

Soit x un nombre flottant simple précision normalisé :

$$x = (-1)^s \cdot 2^e \cdot m \ ,$$

avec :

- $\rightarrow$ $s \in \{0, 1\}$,
- $\rightarrow$ $e \in \mathbb{Z} \cap [-126, 127]$ et $E = e + 127 \in \mathbb{Z} \cap [1, 254]$,
- $\rightarrow$ $m = 1.f_1 f_2 f_3 \ldots f_{23} \in [1, 2)$.

Nombres flottants et représentation binaire

Soit x un nombre flottant simple précision normalisé :

$$x = (-1)^s \cdot 2^e \cdot m \ ,$$

avec :

- $\rightarrow$ $s \in \{0, 1\}$,
- $\rightarrow$ $e \in \mathbb{Z} \cap [-126, 127]$ et $E = e + 127 \in \mathbb{Z} \cap [1, 254]$,
- $\rightarrow$ $m = 1.f_1 f_2 f_3 \ldots f_{23} \in [1, 2)$.

s	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}

Nombres flottants et représentation binaire

Soit x un nombre flottant simple précision dénormalisé :

$$x = (-1)^s \cdot 2^e \cdot m \ ,$$

avec :

- $\rightarrow$ $s \in \{0, 1\}$,
- $\rightarrow$ $e = -127$ et $E = e + 127 = 0$,
- $\rightarrow$ $m = 0.f_1 f_2 f_3 \ldots f_{23} \in [0, 1)$.

Nombres flottants et représentation binaire

Soit x un nombre flottant simple précision dénormalisé :

$$x = (-1)^s \cdot 2^e \cdot m \ ,$$

avec :

- $\rightarrow$ $s \in \{0, 1\}$,
- $\rightarrow$ $e = -127$ et $E = e + 127 = 0$,
- $\rightarrow$ $m = 0.f_1 f_2 f_3 \ldots f_{23} \in [0, 1)$.

s	0	0	0	0	0	0	0	0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}

Développement de fonctions dans FLIP

- Extraction des différents composants : signe / exposant / mantisse
  - ⇒ stockage dans des registres 32 bits

- Interprétation des différentes valeurs en virgule fixe

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

  - ⇒ 1.5 dans le format $Q1.31$

Développement de fonctions dans FLIP

- Extraction des différents composants : signe / exposant / mantisse
  - ⇒ stockage dans des registres 32 bits

- Interprétation des différentes valeurs en virgule fixe

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

  - ⇒ 6.0 dans le format $Q3.29$

Développement de fonctions dans FLIP

- Extraction des différents composants : signe / exposant / mantisse
  - ⇒ stockage dans des registres 32 bits
- Interprétation des différentes valeurs en virgule fixe

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

  - ⇒ 6.0 dans le format $Q3.29$
- Manipulation des différentes valeurs en virgule fixe via les opérations entières

Exemple détaillé

Plan de l'exposé

Méthode générale

Soit x un nombre flottant simple précision normalisé positif :

$$x = m \cdot 2^e,$$

avec $m = 1.f_1 f_2 f_3 \ldots f_{23} \in [1, 2)$, et $e \in \mathbb{Z} \cap [-126, 127]$.

$$\Rightarrow \sqrt{x} = \sqrt{m} \cdot 2^{\frac{e}{2}} = \begin{cases} \sqrt{m} \cdot 2^{\lfloor \frac{e}{2} \rfloor} & \text{si } e \text{ est pair,} \\ \sqrt{2m} \cdot 2^{\lfloor \frac{e}{2} \rfloor} & \text{si } e \text{ est impair.} \end{cases}$$

Méthode générale

Soit x un nombre flottant simple précision normalisé positif :

$$x = m \cdot 2^e,$$

avec $m = 1.f_1 f_2 f_3 \ldots f_{23} \in [1, 2)$, et $e \in \mathbb{Z} \cap [-126, 127]$.

$$\Rightarrow \sqrt{x} = \sqrt{m} \cdot 2^{\frac{e}{2}} = \begin{cases} \sqrt{m} \cdot 2^{\lfloor \frac{e}{2} \rfloor} & \text{si } e \text{ est pair,} \\ \sqrt{2m} \cdot 2^{\lfloor \frac{e}{2} \rfloor} & \text{si } e \text{ est impair.} \end{cases}$$

Finalement : $\sqrt{x} = \ell \cdot 2^d$, avec $\ell = \varphi\sqrt{m}$, $\varphi \in \{1, \sqrt{2}\}$ et $d = \lfloor \frac{e}{2} \rfloor$.

Méthode générale

Soit x un nombre flottant simple précision normalisé positif :

$$x = m \cdot 2^e,$$

avec $m = 1.f_1 f_2 f_3 \ldots f_{23} \in [1, 2)$, et $e \in \mathbb{Z} \cap [-126, 127]$.

$$\Rightarrow \sqrt{x} = \sqrt{m} \cdot 2^{\frac{e}{2}} = \begin{cases} \sqrt{m} \cdot 2^{\lfloor \frac{e}{2} \rfloor} & \text{si } e \text{ est pair,} \\ \sqrt{2m} \cdot 2^{\lfloor \frac{e}{2} \rfloor} & \text{si } e \text{ est impair.} \end{cases}$$

Finalement : $\sqrt{x} = \ell \cdot 2^d$, avec $\ell = \varphi\sqrt{m}$, $\varphi \in \{1, \sqrt{2}\}$ et $d = \lfloor \frac{e}{2} \rfloor$.

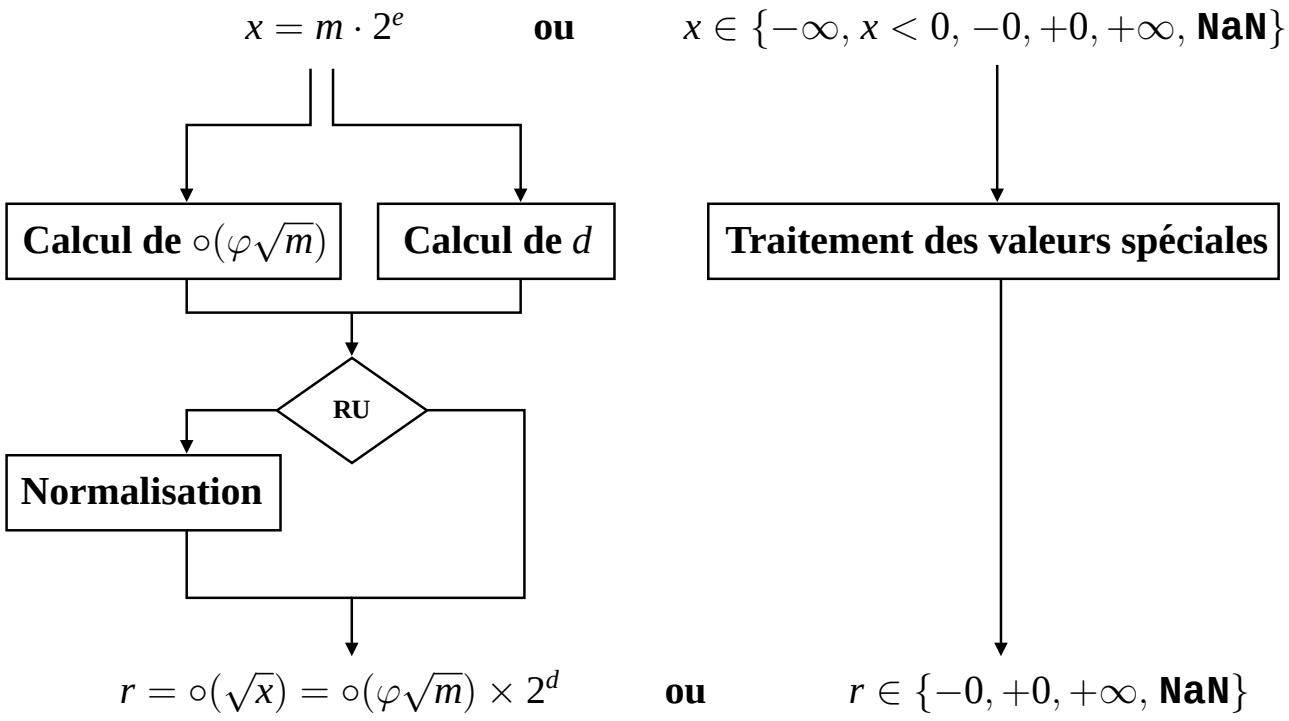
Avantage : pas de renormalisation en arrondi au plus près

$\rightarrow$ $m \in [1, 2)$ donc $\circ\left(\varphi\sqrt{m}\right) \in [1, 2)$

$\rightarrow$ $\circ(\sqrt{x}) = \circ\left(\varphi\sqrt{m}\right) \cdot 2^d$

Principales étapes

Entrée : un nombre flottant x simple précision normalisé ou une valeur spéciale, dans un registre 32 bits.



Sortie : arrondi correct au plus près de $\sqrt{x}$, ou une exception.

Quelle méthode utiliser pour calculer $\varphi\sqrt{m}$?

Méthodes directes restaurante / non-restaurante

- 1 bit du résultat calculé à chaque itération : 24 itérations
- méthodes lentes

Quelle méthode utiliser pour calculer $\varphi\sqrt{m}$?

Méthodes directes restaurante / non-restaurante

- 1 bit du résultat calculé à chaque itération : 24 itérations
- méthodes lentes

Méthodes itératives Newton-Raphson / Goldschmidt

- première approximation de $\sqrt{m}$ ou $\frac{1}{\sqrt{m}}$
- en gros, la précision double à chaque itération
- version précédente de FLIP : 1 itération de Goldschmidt

Quelle méthode utiliser pour calculer $\varphi\sqrt{m}$?

Méthodes directes restaurante / non-restaurante

- 1 bit du résultat calculé à chaque itération : 24 itérations
- méthodes lentes

Méthodes itératives Newton-Raphson / Goldschmidt

- première approximation de $\sqrt{m}$ ou $\frac{1}{\sqrt{m}}$
- en gros, la précision double à chaque itération
- version précédente de FLIP : 1 itération de Goldschmidt

Autres méthodes méthodes SRT, approximations par série entière, ...

Quelle méthode utiliser pour calculer $\varphi\sqrt{m}$?

Méthodes directes restaurante / non-restaurante

- 1 bit du résultat calculé à chaque itération : 24 itérations
- méthodes lentes

Méthodes itératives Newton-Raphson / Goldschmidt

- première approximation de $\sqrt{m}$ ou $\frac{1}{\sqrt{m}}$
- en gros, la précision double à chaque itération
- version précédente de FLIP : 1 itération de Goldschmidt

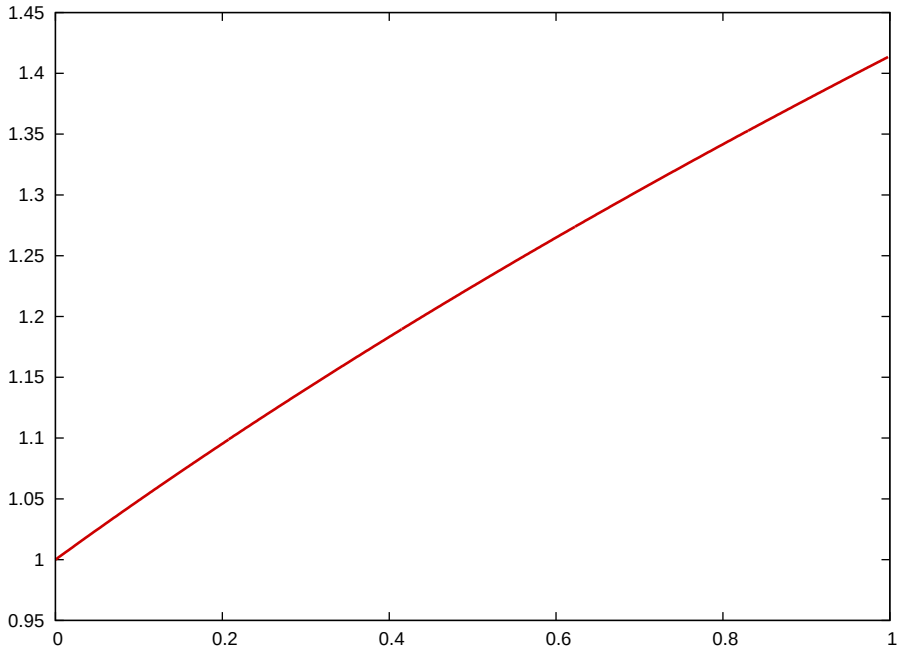
Autres méthodes méthodes SRT, approximations par série entière, ...

Notre approche méthode à base d'évaluation polynomiale

- approximation de $\sqrt{m}$ par un polynôme de degré 8
- évaluation du polynôme avec un schéma rapide

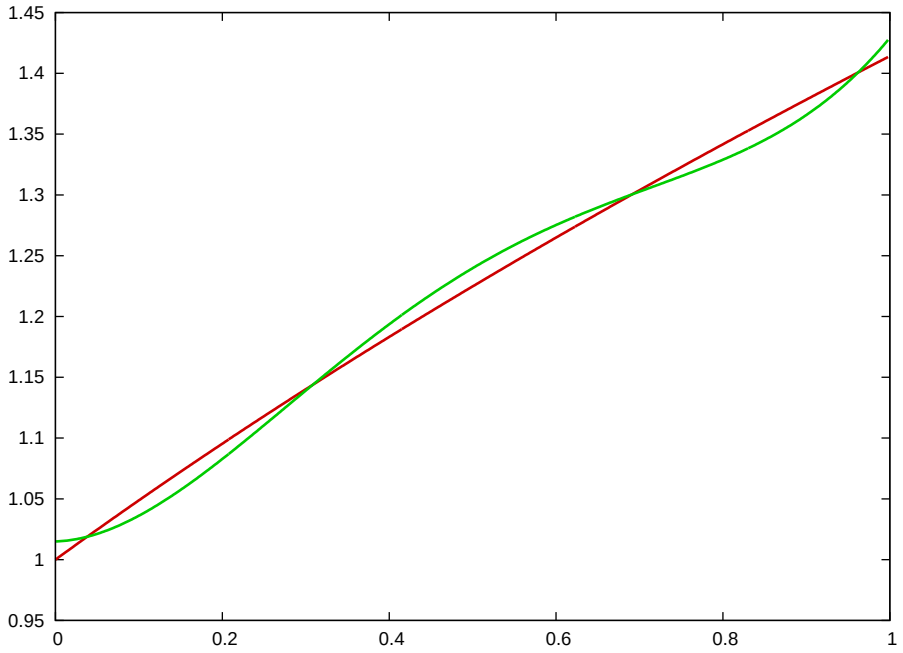
Approximation de $\sqrt{m}$ pour $m \in [1, 2)$

Pour $m \in [1, 2)$, $\sqrt{m} = \sqrt{1+t}$ avec $t \in [0, 1)$.



Approximation de $\sqrt{m}$ pour $m \in [1, 2)$

Pour $m \in [1, 2)$, $\sqrt{m} = \sqrt{1+t}$ avec $t \in [0, 1)$.

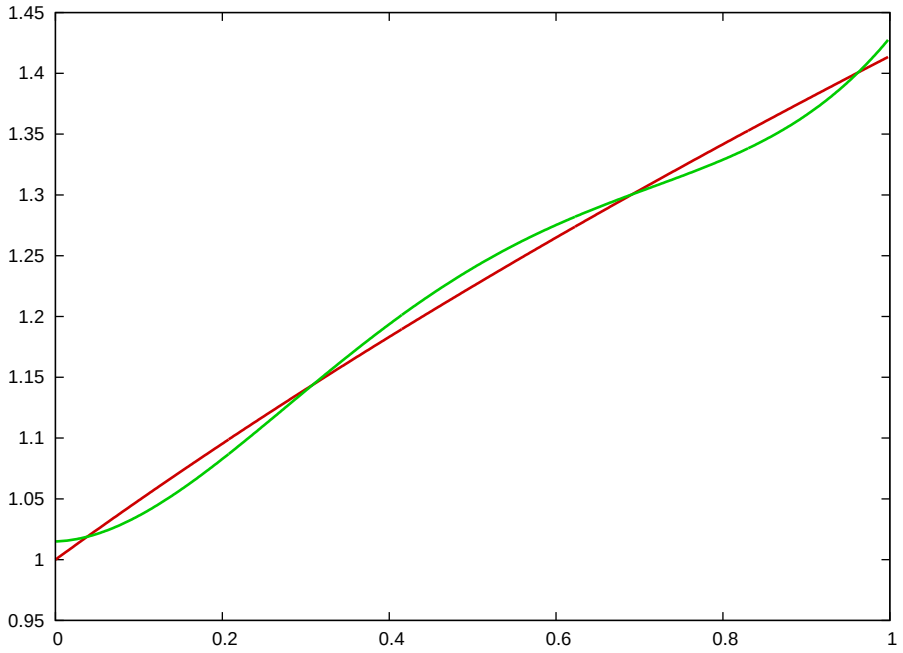


- approximation de $\sqrt{1+t}$ par $a(t)$, pour $t \in [0, 1)$
- $a(t)$: polynôme de degré 8
- obtenu avec un `minimax` (tronqué)

Cf. travaux de N. Brisebarre, S. Chevillard, C. Lauter, JM. Muller et S. Torres (Équipe INRIA Arénaire)

Approximation de $\sqrt{m}$ pour $m \in [1, 2)$

Pour $m \in [1, 2)$, $\sqrt{m} = \sqrt{1+t}$ avec $t \in [0, 1)$.

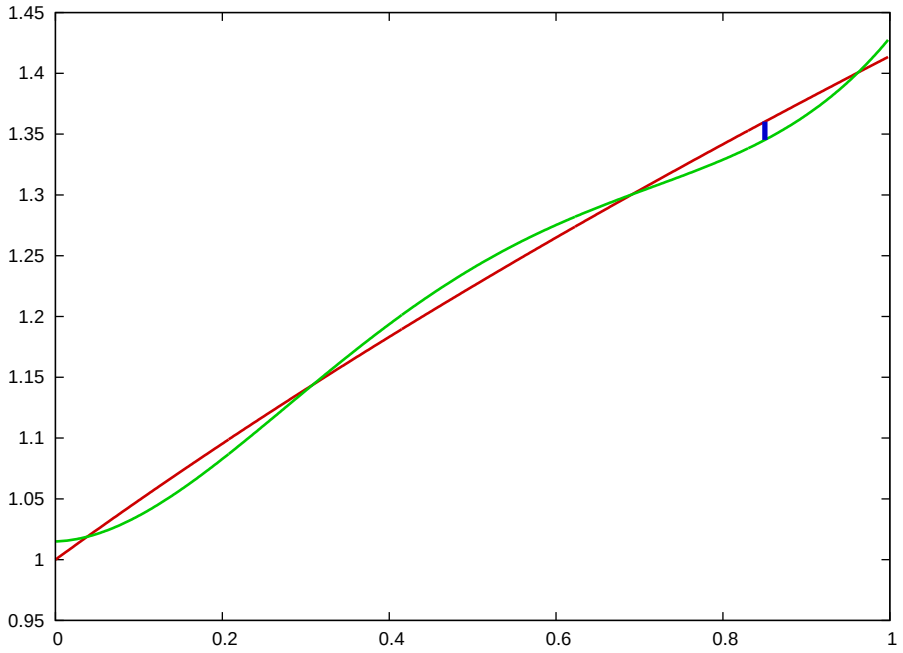


- approximation de $\sqrt{1+t}$ par $a(t)$, pour $t \in [0, 1)$
- $a(t)$: polynôme de degré 8
- obtenu avec un `minimax` (tronqué)
- coefficients structurés :
  - $\rightarrow a_0 = 1$ $\rightarrow a_1 = 1/2$
  - $\rightarrow a_8 = 1/2^{10}$

Cf. travaux de N. Brisebarre, S. Chevillard, C. Lauter, JM. Muller et S. Torres (Équipe INRIA Arénaire)

Approximation de $\sqrt{m}$ pour $m \in [1, 2)$

Pour $m \in [1, 2)$, $\sqrt{m} = \sqrt{1+t}$ avec $t \in [0, 1)$.



- approximation de $\sqrt{1+t}$ par $a(t)$, pour $t \in [0, 1)$
- $a(t)$: polynôme de degré 8
- obtenu avec un `minimax` (tronqué)
- coefficients structurés :
  - $\rightarrow a_0 = 1$ $\rightarrow a_1 = 1/2$
  - $\rightarrow a_8 = 1/2^{10}$
- erreur d'approximation pour tout $t \in [0, 1)$

$$\delta(t) = |a(t) - \sqrt{1+t}| < 2^{-26.8}$$

Cf. travaux de N. Brisebarre, S. Chevillard, C. Lauter, JM. Muller et S. Torres (Équipe INRIA Arénaire)

Quel schéma utiliser pour évaluer $\varphi a(t)$ sur ST231 ?

Quelques caractéristiques

- VLIW 4 voies $\rightarrow$ 4 opérations/cycle (ou 2 multiplications)
- latences : 3 cycles/multiplication et 1 cycle/addition

Quel schéma utiliser pour évaluer $\varphi a(t)$ sur ST231 ?

Quelques caractéristiques

- VLIW 4 voies $\rightarrow$ 4 opérations/cycle (ou 2 multiplications)
- latences : 3 cycles/multiplication et 1 cycle/addition

Schéma de Horner $\varphi a(t) = \varphi\Big[\Big(\cdots\big((a_8 t + a_7)t + a_6\big)\ldots\Big)t + a_0\Big]$

- schéma séquentiel

Quel schéma utiliser pour évaluer $\varphi a(t)$ sur ST231 ?

Quelques caractéristiques

- VLIW 4 voies $\rightarrow$ 4 opérations/cycle (ou 2 multiplications)
- latences : 3 cycles/multiplication et 1 cycle/addition

Schéma de Horner $\varphi a(t) = \varphi\Big[\Big(\cdots\big((a_8 t + a_7)t + a_6\big)\ldots\Big)t + a_0\Big]$

- schéma séquentiel

	— Voie 1 —	— Voie 2 —	— Voie 3 —	— Voie 4 —
cycle 1	$a_8 \times t$			
cycle 2				
cycle 3				
cycle 4	$a_8 t + a_7$			
cycle 5	$(a_8 t + a_7) \times t$			
cycle 6				
cycle 7				
cycle 8	$(a_8 t + a_7) \times t + a_6$			
cycle 9	...			

Quel schéma utiliser pour évaluer $\varphi a(t)$ sur ST231 ?

Quelques caractéristiques

- VLIW 4 voies $\rightarrow$ 4 opérations/cycle (ou 2 multiplications)
- latences : 3 cycles/multiplication et 1 cycle/addition

Schéma de Horner $\varphi a(t) = \varphi\Big[\Big(\cdots\big((a_8 t + a_7)t + a_6\big)\ldots\Big)t + a_0\Big]$

- schéma séquentiel

	— Voie 1 —	— Voie 2 —	— Voie 3 —	— Voie 4 —
cycle 1	$a_8 \times t$			
cycle 2	$a_8 t + a_7$			
cycle 3	$(a_8 t + a_7) \times t$			
cycle 4				
cycle 5				
cycle 6	$(a_8 t + a_7) \times t + a_6$			
cycle 7	...			
cycle 8				
cycle 9				

Quel schéma utiliser pour évaluer $\varphi a(t)$ sur ST231 ?

Quelques caractéristiques

- VLIW 4 voies $\rightarrow$ 4 opérations/cycle (ou 2 multiplications)
- latences : 3 cycles/multiplication et 1 cycle/addition

Schéma de Horner $\varphi a(t) = \varphi\Big[\Big(\cdots\big((a_8 t + a_7)t + a_6\big)\ldots\Big)t + a_0\Big]$

- schéma séquentiel
- utilisation d'une seule des 4 voies
- latence = 29 cycles

Quel schéma utiliser pour évaluer $\varphi a(t)$ sur ST231 ?

Quelques caractéristiques

- VLIW 4 voies $\rightarrow$ 4 opérations/cycle (ou 2 multiplications)
- latences : 3 cycles/multiplication et 1 cycle/addition

Schéma de Horner $\varphi a(t) = \varphi\Big[\Big(\cdots\big((a_8 t + a_7)t + a_6\big)\ldots\Big)t + a_0\Big]$

- schéma séquentiel
- utilisation d'une seule des 4 voies
- latence = 29 cycles

Schéma rapide proposé

- coefficients positifs et valeurs intermédiaires positives
- incorporation de la multiplication par φ et ajout de 2^{-25} (utile pour l'arrondi)

$$\varphi a(t)+2^{-25} = \Big(\big(\varphi(a_0+a_1 t)+2^{-25}\big)-\big((a_2-a_3 t)+(a_4-a_5 t)t^2\big)\varphi t^2\Big)-\Big(\big((a_6-a_7 t)+(a_8 t^2)\big)t^4\Big)\varphi t^2$$

Ordonnancement du schéma d'évaluation sur ST231

$$\varphi a(t)+2^{-25} = \Big(\big(\varphi(a_0+a_1t)+2^{-25}\big)-\big((a_2-a_3t)+(a_4-a_5t)t^2\big)\varphi t^2\Big)-\Big(\big((a_6-a_7t)+(a_8t^2)\big)t^4\Big)\varphi t^2$$

	— Voie 1 —	— Voie 2 —	— Voie 3 —	— Voie 4 —
cycle 1	$t^2 = t \times t$	$a_5 \times t$		
cycle 2	$a_7 \times t$	$a_1 \times t$		
cycle 3	$a_3 \times t$			
cycle 4	$t^4 = t^2 \times t^2$	$a_{45} = a_4 - a_5t$		
cycle 5	$a_{45} \times t^2$	$a_8 \times t^2$	$a_{67} = a_6 - a_7t$	
cycle 6	$a_{01} = a_0 + a_1t$	$\varphi \times t^2$	$a_{68} = a_{67} + a_8t^2$	
cycle 7	$a_{23} = a_2 - a_3t$	$a_{68} \times t^4$		
cycle 8	$a_{01} \times \varphi$	$a_{25} = a_{23} + a_{45}t^2$		
cycle 9	$a_{25} \times \varphi t^2$			
cycle 10	$(a_{68}t^4) \times \varphi t^2$			
cycle 11	$a'_{01} = a_{01}\varphi + 2^{-25}$			
cycle 12	$a_{05} = a'_{01} - a_{25}\varphi t^2$			
cycle 13	$\varphi a(t) = a_{05} - (a_{68}t^4)\varphi t^2$			

- Évaluation en 13 cycles (≈ 2.2 fois plus rapide que Horner)

Ordonnancement du schéma d'évaluation sur ST231

$$\varphi a(t)+2^{-25} = \Big(\big(\varphi(a_0+a_1t)+2^{-25}\big)-\big((a_2-a_3t)+(a_4-a_5t)t^2\big)\varphi t^2\Big)-\Big(\big((a_6-a_7t)+(a_8t^2)\big)t^4\Big)\varphi t^2$$

	— Voie 1 —	— Voie 2 —	— Voie 3 —	— Voie 4 —
cycle 1	$t^2 = t \times t$	$a_5 \times t$	–	
cycle 2	$a_7 \times t$	$a_1 \times t$	–	
cycle 3	$a_3 \times t$	–		
cycle 4	$t^4 = t^2 \times t^2$	$a_{45} = a_4 - a_5t$	–	
cycle 5	$a_{45} \times t^2$	$a_8 \times t^2$	$a_{67} = a_6 - a_7t$	
cycle 6	$a_{01} = a_0 + a_1t$	$\varphi \times t^2$	$a_{68} = a_{67} + a_8t^2$	–
cycle 7	$a_{23} = a_2 - a_3t$	$a_{68} \times t^4$	–	
cycle 8	$a_{01} \times \varphi$	$a_{25} = a_{23} + a_{45}t^2$	–	–
cycle 9	$a_{25} \times \varphi t^2$	–	–	
cycle 10	$(a_{68}t^4) \times \varphi t^2$	–	–	
cycle 11	$a'_{01} = a_{01}\varphi + 2^{-25}$	–	–	
cycle 12	$a_{05} = a'_{01} - a_{25}\varphi t^2$	–	–	
cycle 13	$\varphi a(t) = a_{05} - (a_{68}t^4)\varphi t^2$	–	–	

- Évaluation en 13 cycles (≈ 2.2 fois plus rapide que Horner)

Comment vérifier l'arrondi correct ?

Méthode 1 Validation *a priori*

- erreur d'approximation (*Sollya*[1])
- erreur d'évaluation (*Gappa* [2])
- preuve sur papier des méthodes d'arrondi

Méthode 2 Validation *a posteriori*

- tests exhaustifs

$\Rightarrow$ envisageable uniquement pour les fonctions univariées

[1] Sollya : `https://gforge.inria.fr/projects/sollya/`
[2] Gappa : `http://lipforge.ens-lyon.fr/www/gappa/`

Performances sur ST231

- Pour toute entrée x (valeurs spéciales comprises) :

	RN	RU	RD/RZ
Sans dénormalisés	22	22	22
Avec dénormalisés	25	25	25

TAB.: Timings (cycles) sur ST231, suivant différents modes d'arrondi

- Accélération $\approx 55\%$ par rapport à la version précédente de FLIP
- Même latence pour les quatre modes d'arrondi
- Surcoût dû à la prise en compte des nombres dénormalisés = 3 cycles (2 cycles en théorie)

Performances sur ST231

- Pour toute entrée x (valeurs spéciales comprises) :

	RN	RU	RD/RZ
Sans dénormalisés	22	22	22
Avec dénormalisés	25	25	25

TAB.: Timings (cycles) sur ST231, suivant différents modes d'arrondi

- Accélération $\approx 55\%$ par rapport à la version précédente de FLIP
- Même latence pour les quatre modes d'arrondi
- Surcoût dû à la prise en compte des nombres dénormalisés = 3 cycles (2 cycles en théorie)
- Méthode facilement adaptable :
  - à l'arrondi fidèle (RF), mais polynôme *trop précis*
  - à d'autres formats (*medium/high precision*)

Plan de l'exposé

État actuel de FLIP

FLIP 1.0

Fonctions correctement arrondies suivant les 4 modes d'arrondis, avec ou sans nombres dénormalisés : $\sqrt{x}$, $\frac{1}{x}$, $x^{-\frac{1}{2}}$, $x^{-\frac{1}{4}}$

Fonction correctement arrondie au plus près, sans nombres dénormalisés : $\frac{x}{y}$

Timings sur ST231

État actuel de FLIP

FLIP 1.0

Fonctions correctement arrondies suivant les 4 modes d'arrondis, avec ou sans nombres dénormalisés : $\sqrt{x}$, $\frac{1}{x}$, $x^{-\frac{1}{2}}$, $x^{-\frac{1}{4}}$

Fonction correctement arrondie au plus près, sans nombres dénormalisés : $\frac{x}{y}$

Timings sur ST231

État actuel de FLIP

FLIP 1.0

Fonctions correctement arrondies suivant les 4 modes d'arrondis, avec ou sans nombres dénormalisés : $\sqrt{x}$, $\frac{1}{x}$, $x^{-\frac{1}{2}}$, $x^{-\frac{1}{4}}$

Fonction correctement arrondie au plus près, sans nombres dénormalisés : $\frac{x}{y}$

Timings sur ST231

État actuel de FLIP

FLIP 1.0

Fonctions correctement arrondies suivant les 4 modes d'arrondis, avec ou sans nombres dénormalisés : $\sqrt{x}$, $\frac{1}{x}$, $x^{-\frac{1}{2}}$, $x^{-\frac{1}{4}}$

Fonction correctement arrondie au plus près, sans nombres dénormalisés : $\frac{x}{y}$

Timings sur ST231

- Addition / Soustraction / Multiplication : disponible dans FLIP-0.3

État actuel de FLIP

FLIP 1.0

Fonctions correctement arrondies suivant les 4 modes d'arrondis, avec ou sans nombres dénormalisés : $\sqrt{x}$, $\frac{1}{x}$, $x^{-\frac{1}{2}}$, $x^{-\frac{1}{4}}$

Fonction correctement arrondie au plus près, sans nombres dénormalisés : $\frac{x}{y}$

Timings sur ST231

- Addition / Soustraction / Multiplication : disponible dans FLIP-0.3

\+ Évaluation de toutes ces fonctions suivant le même schéma

– Mode d'arrondi et prise en compte des dénormalisés = statiques
  ⇒ utilisation d'options de compilation

Plan de l'exposé

Résumé de l'approche

Comment implanter une fonction ?

- Calculer les coefficients d'un bon polynôme d'approximation
  - → quelle fonction sur quel intervalle ? quel degré ? quelle structure de coefficients ?
- Proposer un code d'évaluation de ce polynôme en virgule fixe
  - → quel degré de parallélisme ? quelles latences ?
  - → quel schéma d'évaluation ?
  - → analyse numérique du schéma d'évaluation avec Gappa

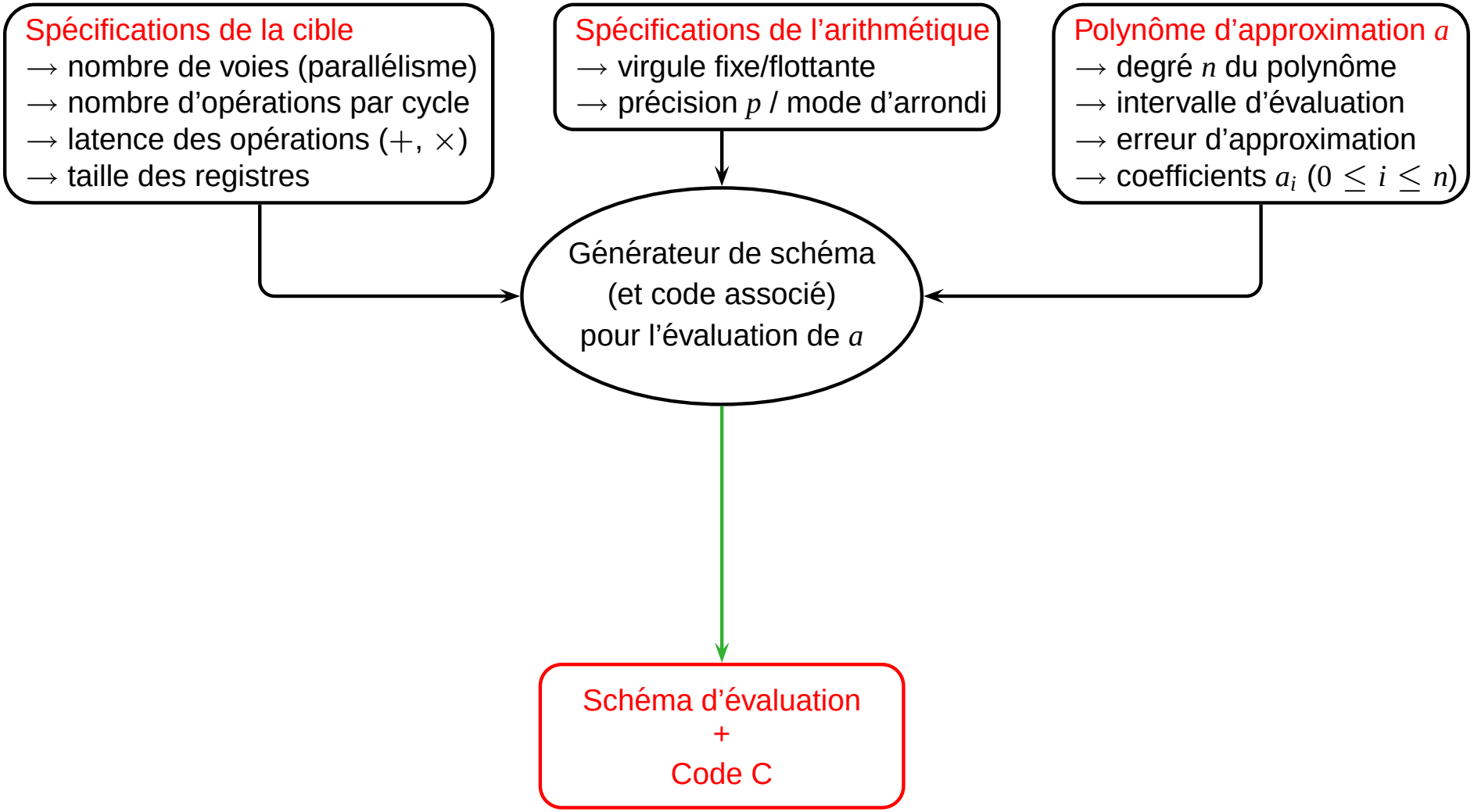
Résumé de l'approche

Comment implanter une fonction ?

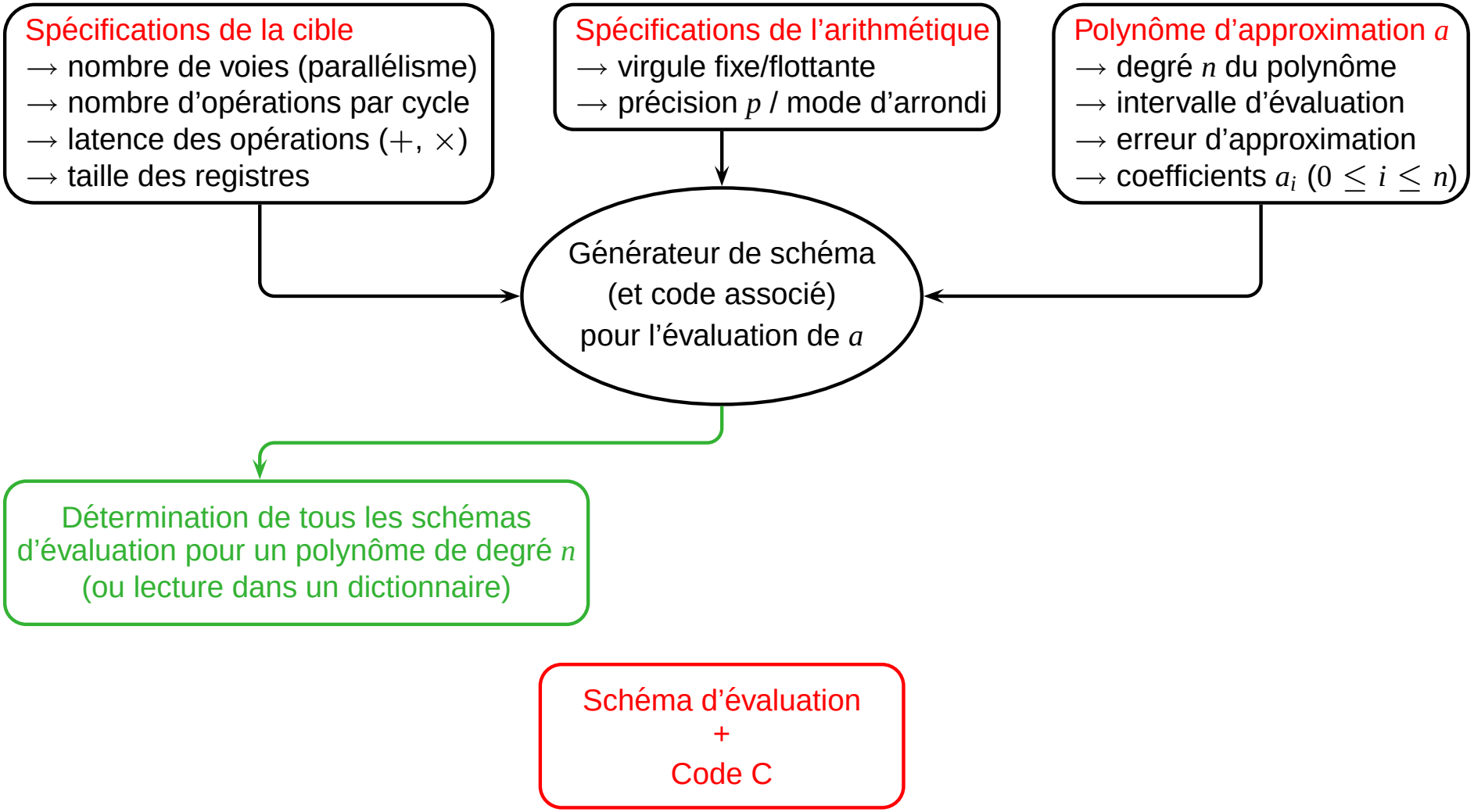
- Calculer les coefficients d'un bon polynôme d'approximation
  - → quelle fonction sur quel intervalle ? quel degré ? quelle structure de coefficients ?
- Proposer un code d'évaluation de ce polynôme en virgule fixe
  - → quel degré de parallélisme ? quelles latences ?
  - → quel schéma d'évaluation ?
  - → analyse numérique du schéma d'évaluation avec Gappa

BESOIN D'AUTOMATISATION

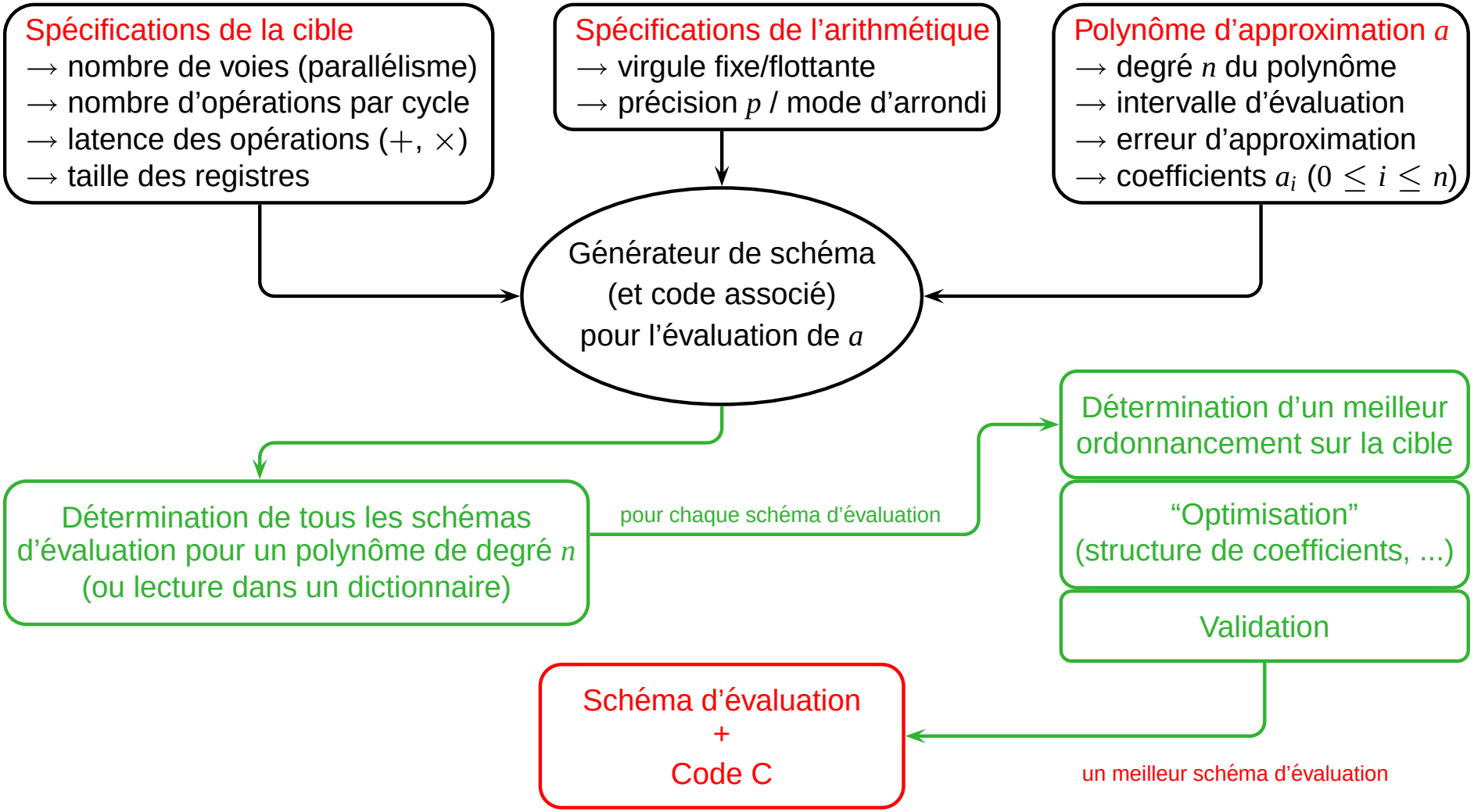
Vers l'automatisation de la conception d'une fonction



Vers l'automatisation de la conception d'une fonction



Vers l'automatisation de la conception d'une fonction



Conclusions

- Support logiciel pour l'arithmétique flottante simple précision pour processeurs entiers : bibliothèque FLIP

  `https://lipforge.ens-lyon.fr/projects/flip/`

- Ensemble de fonctions algébriques correctement arrondies, avec ou sans nombres dénormalisés

Conclusions

- Support logiciel pour l'arithmétique flottante simple précision pour processeurs entiers : bibliothèque FLIP

  `https://lipforge.ens-lyon.fr/projects/flip/`

- Ensemble de fonctions algébriques correctement arrondies, avec ou sans nombres dénormalisés

- Implantation de toutes les fonctions sur le même schéma : à base d'évaluation polynomiale
- Efficace sur ST231 : accélération de $\approx 50\%$ pour la plupart des fonctions

Conclusions

- Support logiciel pour l'arithmétique flottante simple précision pour processeurs entiers : bibliothèque FLIP

  `https://lipforge.ens-lyon.fr/projects/flip/`

- Ensemble de fonctions algébriques correctement arrondies, avec ou sans nombres dénormalisés

- Implantation de toutes les fonctions sur le même schéma : à base d'évaluation polynomiale

- Efficace sur ST231 : accélération de $\approx 50\%$ pour la plupart des fonctions

- Besoin d'automatiser la conception :
  - $\longrightarrow$ des schémas d'évaluation et des procédures d'arrondi,
  - $\longrightarrow$ et plus généralement de fonctions.

Exemple détaillé

Calcul de $\circ(\sqrt{17})$: $17 = 1.0625 \times 2^4$

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sortie $\circ(\sqrt{x})$	0		

Retour à l'exposé

Exemple détaillé

Calcul de l'exposant du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	
Sortie $\circ(\sqrt{x})$	0																																

Retour à l'exposé

Exemple détaillé

Calcul de l'exposant du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
$E + 127$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
Sortie $\circ(\sqrt{x})$	0																																

Retour à l'exposé

Exemple détaillé

Calcul de l'exposant du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1		
$E + 127$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0		
$D = \frac{E+127}{2}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
Sortie $\circ(\sqrt{x})$	0																																	

Retour à l'exposé

Exemple détaillé

Calcul de l'exposant du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1		
$E + 127$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0		
$D = \frac{E+127}{2}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
Sortie $\circ(\sqrt{x})$	0	1	0	0	0	0	0	0	1																									

Retour à l'exposé

Exemple détaillé

Calcul de la fraction du résultat

Entrée $x = 17$

0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sortie $\circ\left(\sqrt{x}\right)$

0	1	0	0	0	0	0	0	1	

Retour à l'exposé

Exemple détaillé

Calcul de la fraction du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	• 0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sortie $\circ(\sqrt{x})$	0	1	0	0	0	0	0	0	1																								

Retour à l'exposé

Exemple détaillé

Calcul de la fraction du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	•	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v	0	1•	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1	0	0		
Sortie $\circ(\sqrt{x})$	0	1	0	0	0	0	0	0	1																									

Retour à l'exposé

Exemple détaillé

Calcul de la fraction du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	•	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v	0	1•	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1	0	0		
$\circ(\varphi\sqrt{m})$	0	1•	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0		
Sortie $\circ(\sqrt{x})$	0	1	0	0	0	0	0	0	1																									

Retour à l'exposé

Exemple détaillé

Calcul de la fraction du résultat

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	.	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v	0	1.	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1	0	0		
$\circ(\varphi\sqrt{m})$	0	1.	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0		
Sortie $\circ(\sqrt{x})$	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1		

Retour à l'exposé

Exemple détaillé

Calcul de $\circ(\sqrt{17})$: $17 = 1.0625 \times 2^4$

Entrée $x = 17$	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sortie $\circ(\sqrt{x})$	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1

$$\approx 4.123105... \approx \circ(\sqrt{17})$$

Retour à l'exposé

Timings pour d'autres fonctions

	Sans dénormalisés			Avec dénormalisés	
	FLIP 0.3	FLIP 1.0	Accélération	FLIP 1.0	Surcoût
$\sqrt{x}$	48	22	$\approx 55\%$	25	3
$x^{-\frac{1}{2}}$	60	29	$\approx 52\%$	31	2
$x^{-\frac{1}{4}}$	–	36	–	38	2
$1/x$	45	22	$\approx 51\%$	29	7
x/y	50	37	$\approx 26\%$	–	–

TAB.: Timings en arrondi au plus près sur ST231

Retour à l'exposé