Exemple d'implantation de fonction mathématique sur ST240

Guillaume Revy
Encadrants : Claude-Pierre Jeannerod et Gilles Villard

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

Réunion EVA-Flo

Perpignan, 18-19 octobre 2007

Contexte et objectifs

Bibliothèque FLIP

- Support logiciel pour l'arithmétique flottante simple précision aux processeurs entiers

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

- Fonctions mathématiques rapides et précises

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

Contexte et objectifs

Bibliothèque FLIP

- Support logiciel pour l'arithmétique flottante simple précision aux processeurs entiers

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

- Fonctions mathématiques rapides et précises

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

Objectifs

- 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)

Méthode générale

Soit x un nombre flottant simple précision normalisé positif (IEEE-754) :

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

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

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

Méthode générale

Soit x un nombre flottant simple précision normalisé positif (IEEE-754) :

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

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

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

Finalement : $\sqrt{x} = \ell \times 2^d$, 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 (IEEE-754) :

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

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

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

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

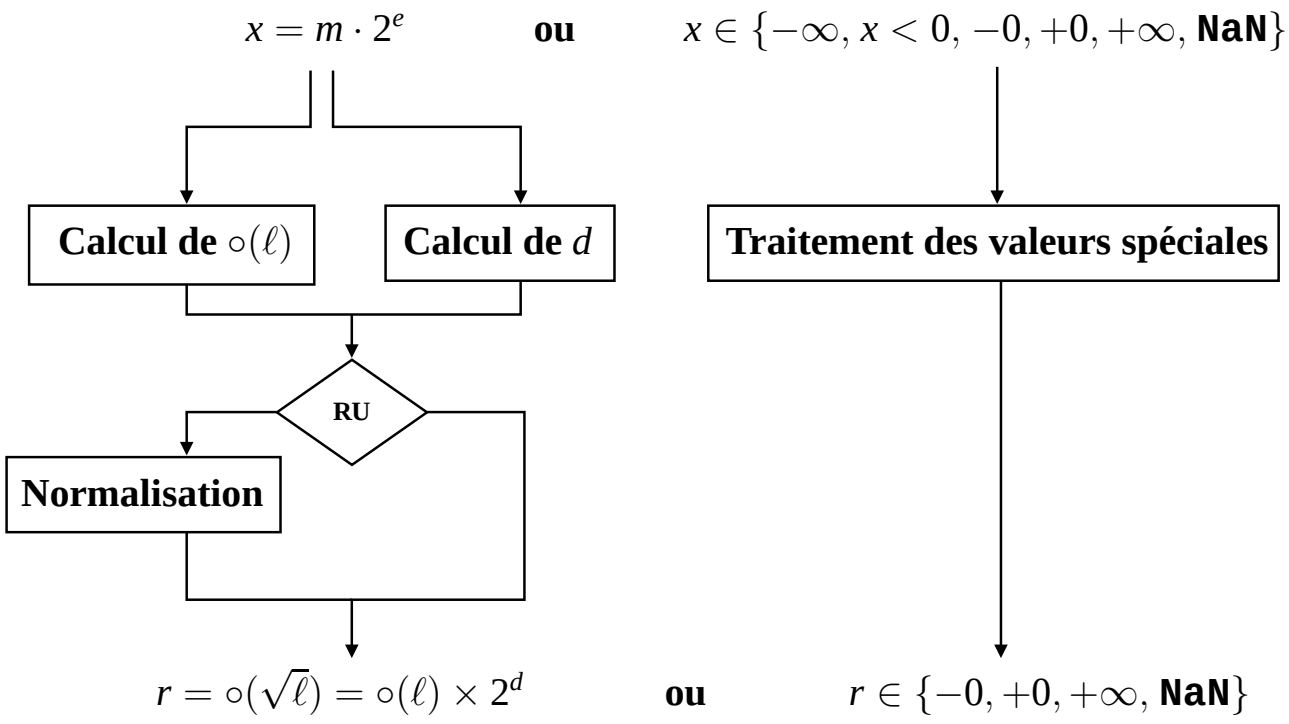
Avantage : pas de renormalisation en arrondi au plus près

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

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

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
- méthode précédente : 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
- méthode précédente : 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
- méthode précédente : 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 (avec coefficients structurés)
- évaluation du polynôme avec un schéma rapide

Quel schéma utiliser pour évaluer $\varphi a(t)$ sur ST240 ?

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 ST240 ?

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 ST240 ?

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 ST240 ?

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 ST240 ?

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 ST240 ?

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)

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

Ordonnancement du schéma d'évaluation sur ST240

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

	— 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 ST240

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

	— 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 (*Maple, Arenairetools*)
- erreur d'évaluation (*Gappa*)
- preuve sur papier des méthodes d'arrondi

Méthode 2 Validation *a posteriori*

- tests exhaustifs

$\Rightarrow$ envisageable uniquement pour les fonctions univariées

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

Performances sur ST240

- Pour toute entrée x (valeurs spéciales comprises) :

	RN	RU	RD/RZ	RF
Sans dénormalisés	22	22	22	20
Avec dénormalisés	25	25	25	22

TAB.: Timings (cycles) sur ST240, suivant différents modes d'arrondi

- Accélération $\approx 55\%$ par rapport à la version précédente
- 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 ST240

- Pour toute entrée x (valeurs spéciales comprises) :

	RN	RU	RD/RZ	RF
Sans dénormalisés	22	22	22	20
Avec dénormalisés	25	25	25	22

TAB.: Timings (cycles) sur ST240, suivant différents modes d'arrondi

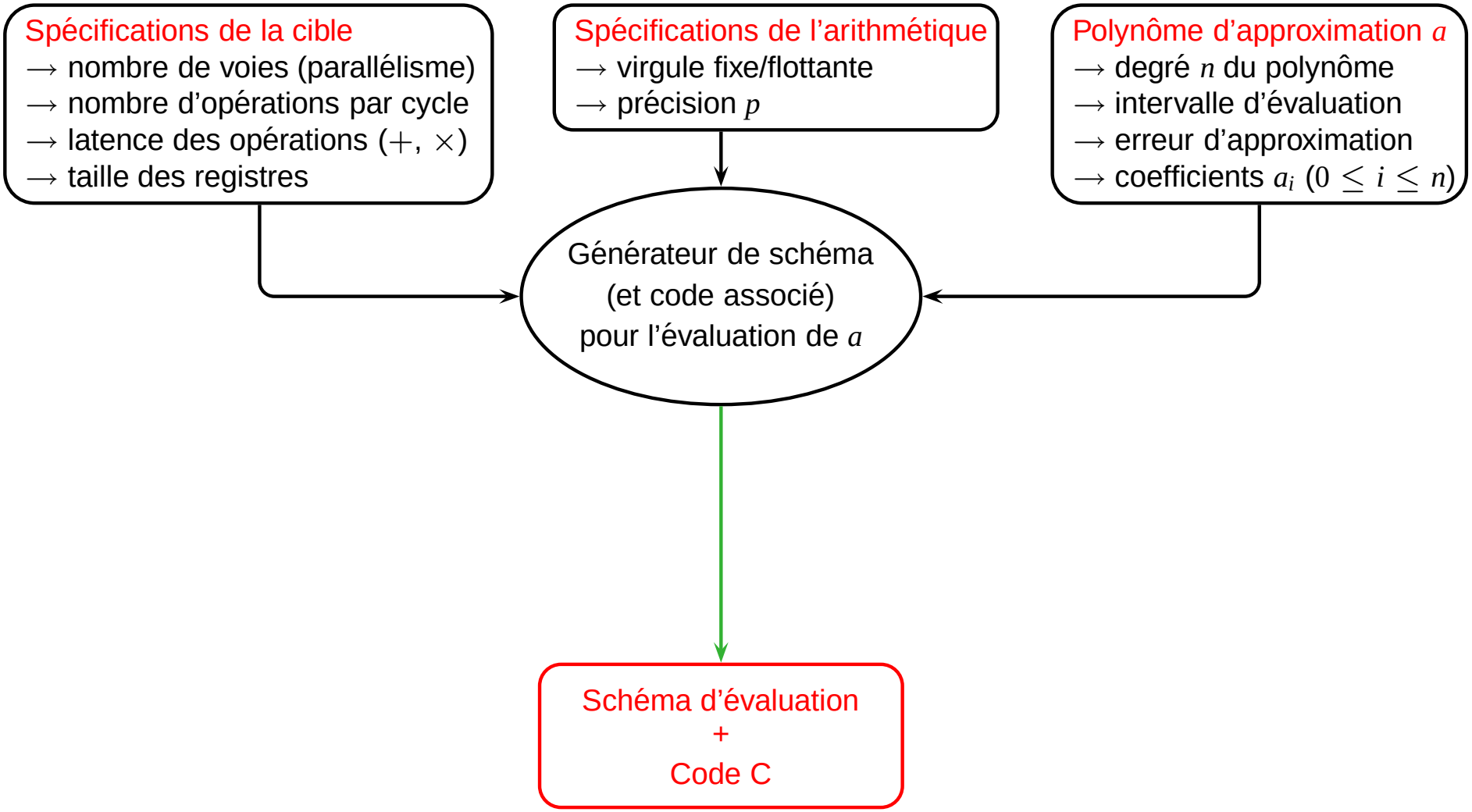
- Accélération $\approx 55\%$ par rapport à la version précédente
- Même latence pour les quatre modes d'arrondi
- Surcoût dû à la prise en compte des nombres dénormalisés = 3 cycles (2 cycles en théorie)
- Méthode facilement adaptable :
  - à l'arrondi fidèle (RF), mais polynôme *trop précis*
  - à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

Extension à d'autres fonctions...

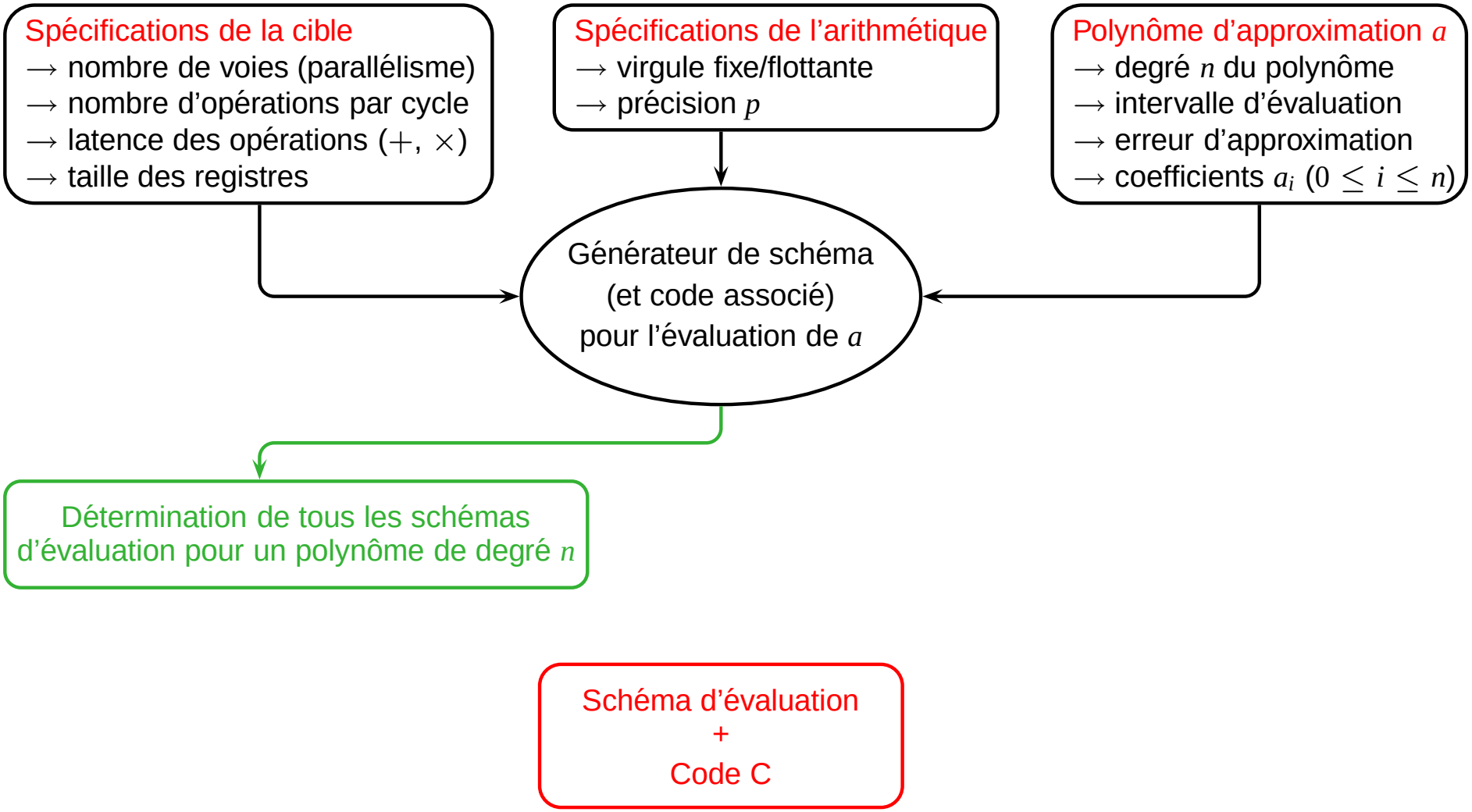
	Sans dénormalisés			Avec dénormalisés	
	FLIP 1.0	FLIP 0.3	Accélération	FLIP 1.0	Surcoût
$\sqrt{x}$	22	48	$\approx 55\%$	25	3
$x^{-\frac{1}{2}}$	29	60	$\approx 52\%$	31	2
$x^{-\frac{1}{4}}$	36	–	–	38	2
$1/x$	23	45	$\approx 49\%$	28	5

TAB.: Timings en arrondi au plus près sur ST240

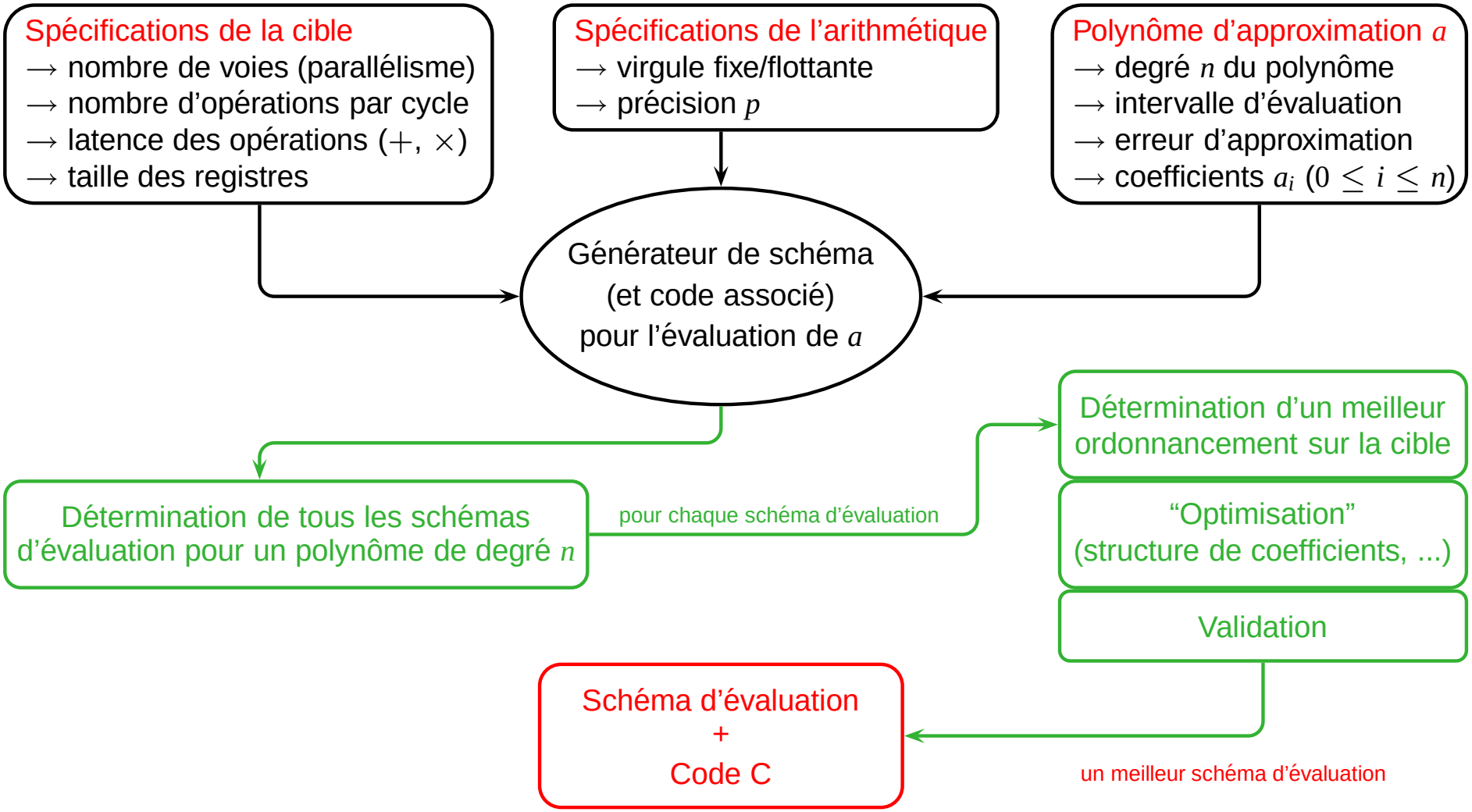
Vers l'automatisation...



Vers l'automatisation...



Vers l'automatisation...



Conclusions

- Racine carrée simple précision correctement arrondie
- Implantation à base d'évaluation polynomiale
- Efficace sur ST240 : accélération de $\approx 55\%$

- Efficacité de cette méthode pour l'implantation d'autres fonctions algébriques : racine carrée inverse, racine quatrième inverse...
- Extension aux autres fonctions de FLIP (inverse, division, ...)

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