`GPGPU Posit pour HPC` : Fomat de représentation des nombres flottants

Rodolphe Thienard

15 mars 2024 - 13 Septembre 2024

Sommaire

Introduction

EMG2

EMG2 est une entreprise spécialisées dans la distribution de technologies sur mesure dans les secteurs tels que le médical, militaire, aéronautique et spatial.

VIVIDSPARKS

VividSparks conçoit des semi-conducteurs spécialisée, offrant performance et faible consommation.

EMG2 est située à Villebon-sur-Yvette

Objectifs du stage

Validation technologique de l'intégration matérielle d'un cœur GPGPU POSIT sur FPGA pour applications HPC/HPDA et Deep learning

- Établir un état de l'art de la représentation Posit
- Evaluer les performances de Posit
- Valider l'implémentation de RacEr
- Évaluer la faisabilité de Posit dans l'industrie

Les nombres réels

Pourquoi les nombres réels :

- Permet de représenter des valeurs pour modéliser des phénomènes du monde réel.
- Indispensable pour les calculs précis en sciences, ingénierie, finances et modélisation graphique.
- Gère efficacement des nombres très petits ou très grands.

Standard IEEE 754

- Le **signe** indique si le nombre est positif ou négatif .
- L'**exposant** détermine l'ordre de grandeur du nombre.
- La **mantisse** ajuste la précision du nombre.

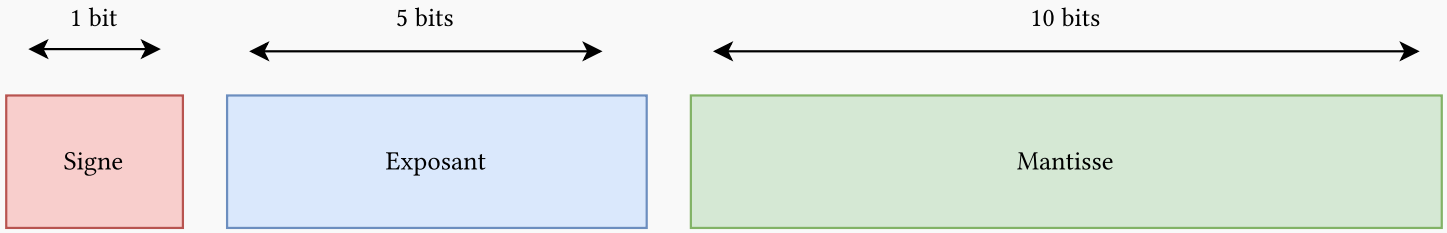


Figure 1: Représentation IEEE 754 16 bits

Posit

- Le **signe** indique si le nombre est positif ou négatif .
- Le **régime** détermine l'échelle de la valeur du nombre. Il permet de gérer plus efficacement une large gamme de valeurs.
- L'**exposant** ajuste la grandeur du nombre en fonction du régime.
- La **fraction** ajuste la précision du nombre avec une répartition des bits plus adaptative.

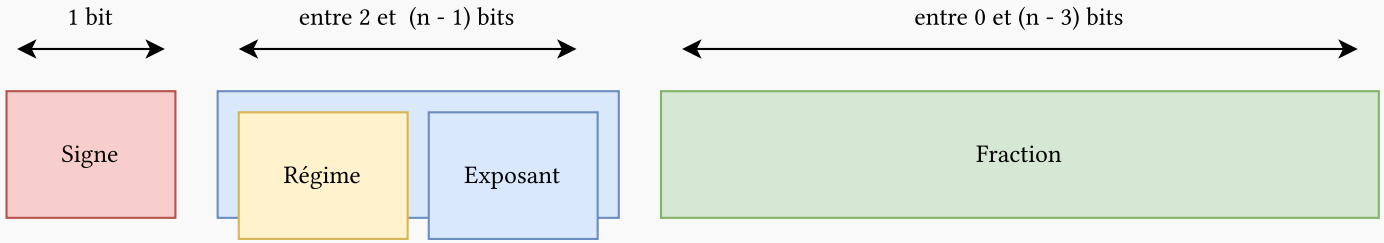


Figure 2: Représentation de Posit 16 bits

Problématique

Comment les limitations actuelles dans la représentation des nombres à virgule peuvent-elles être surmontées par l'utilisation du format Posit ?

Cas d'études

Inversion de matrice

En appliquant 2 fois une inversion sur une matrice, nous devrions obtenir la matrice d'entrée.

$$\left((A)^{-1}\right)^{-1} = A$$

Ainsi, chaque valeur qui diffère de A à la fin est due à une erreur de calcul :

$$\left((A)^{-1}\right)^{-1} = A + A_{\delta}$$

Mesure de l'erreur en utilisant la Root Mean Square (RMS) :

$$\text{RMS} = \sqrt{\left(\frac{1}{N}\right) * \sum (x_i^2 \text{ for } i \in 1..N)}$$

Inversion de matrice

i Float 32 bits IEEE

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401987 3.943966
      7.831672 7.984796 ]
rms = 0.000540
```

i Float 64 bits IEEE

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401877 3.943829
      7.830992 7.984400 ]
rms = 0.000000
```

Simulation Moléculaire

Application de simulation moléculaire dont la somme des forces doit etre égale à zéro.

Pour cette simulation, l'ordre de grandeur des variables est compris entre 0 et 1×10^4.

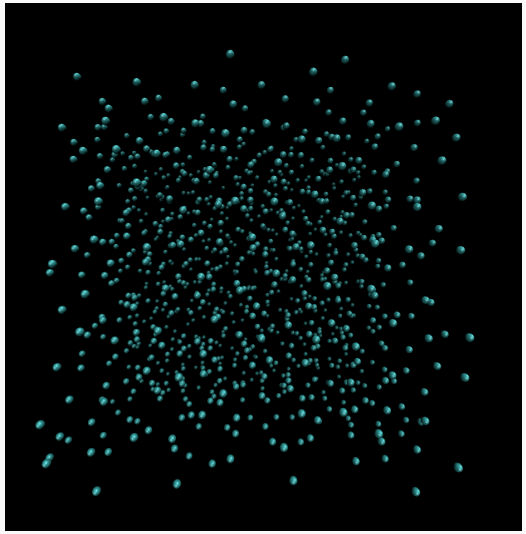


Figure 3: Debut de la simulation

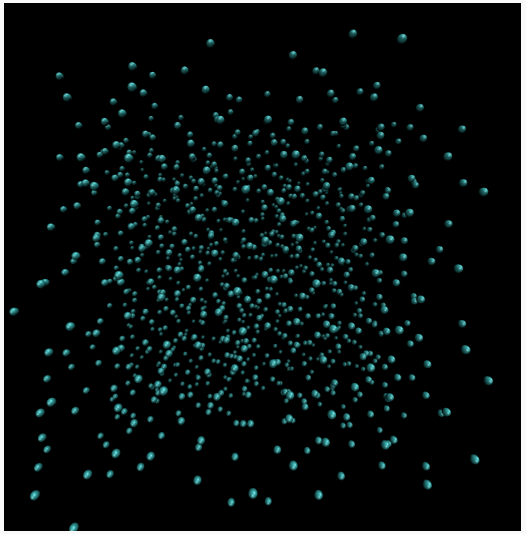


Figure 4: Fin de la simulation

Simulation Moléculaire

i Résultats IEEE 64 bits

```
Energy verlet: 758.659959 = -1085.330020 + 1843.989979  Temperature: 309.185229
Sum of verlet forces: -1.134026e-11
```

Validation de la littérature

Annonces de la littérature

- **Précision adaptative** : Le format posit offre une précision plus élevée pour les nombres proches de zéro.
- **Pas de NaN ou d'infinis** : Le format posit élimine la nécessité des représentations spéciales comme NaN (Not a Number) ou $\pm\infty$.
- **Résilience accrue aux erreurs** : Posit réduit les erreurs d'arrondi, offrant une plus grande stabilité numérique dans les calculs.
- **Format plus compact** : Les nombres posit peuvent être représentés de manière plus compacte tout en offrant une précision équivalente ou supérieure, ce qui économise de la mémoire.

Émulations

L'émulation consiste à reproduire sur une machine le comportement d'une application qui à été prévu pour un autre type de machine.

Émulations Posit utilisées :

1. Softposit
2. Ceralaine
3. Posit GCC
4. FastSigmoid

Émulations

L'émulation consiste à reproduire sur une machine le comportement d'une application qui à été prévu pour un autre type de machine.

Émulations Posit utilisées :

1. Softposit
   - Posit<8,0>
   - Posit<16,1>
   - Posit<32,2>
2. Ceralaine
3. Posit GCC
4. FastSigmoid

Émulations

L'émulation consiste à reproduire sur une machine le comportement d'une application qui à été prévu pour un autre type de machine.

Émulations Posit utilisées :

1. Softposit
2. Ceralaine
   - Posit<8,0>
   - Posit<16,1>
   - Posit<32,2>
   - Posit<64,3>
   - Posit<128,4>
3. Posit GCC
4. FastSigmoid

Émulations

L'émulation consiste à reproduire sur une machine le comportement d'une application qui à été prévu pour un autre type de machine.

Émulations Posit utilisées :

1. Softposit
2. Ceralaine
3. Posit GCC
   - Posit<32,2>
   - Posit<64,3>
4. FastSigmoid

Émulations

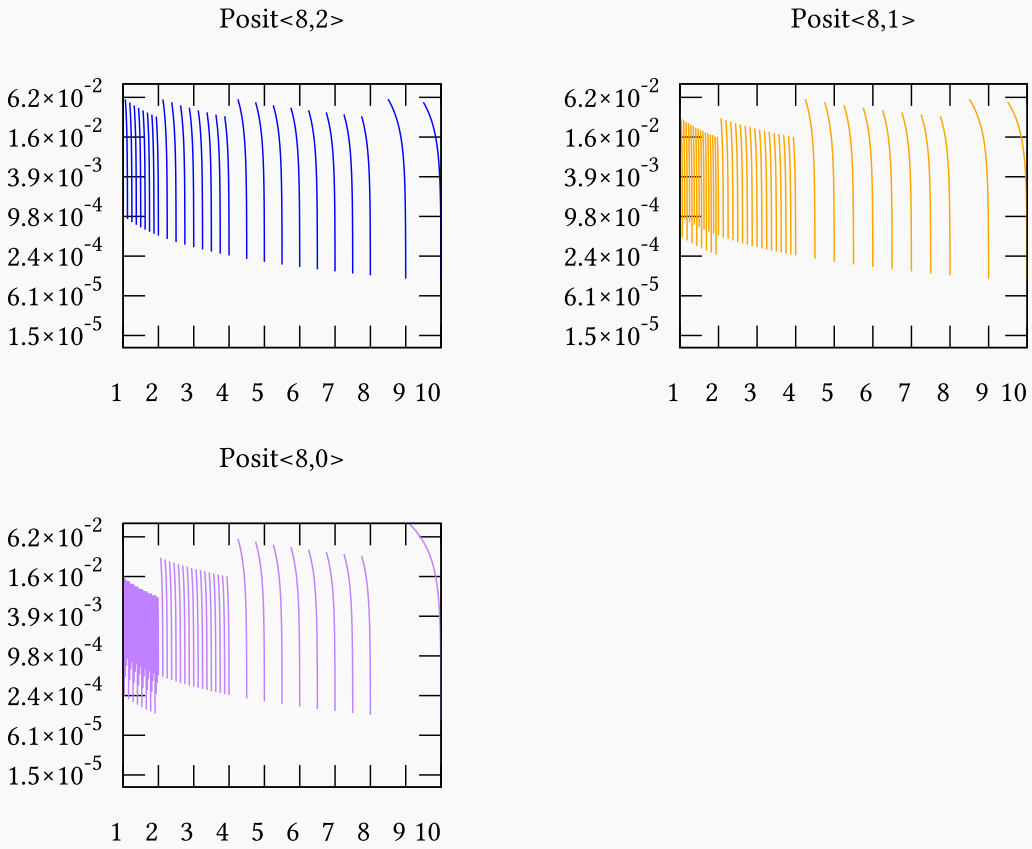
L'émulation consiste à reproduire sur une machine le comportement d'une application qui à été prévu pour un autre type de machine.

Émulations Posit utilisées :

1. Softposit
2. Ceralaine
3. Posit GCC
4. FastSigmoid
   - Posit<8,0> à Posit<8,2>
   - Posit<16,0> à Posit<16,2>
   - Posit<32,0> à Posit<32,3>

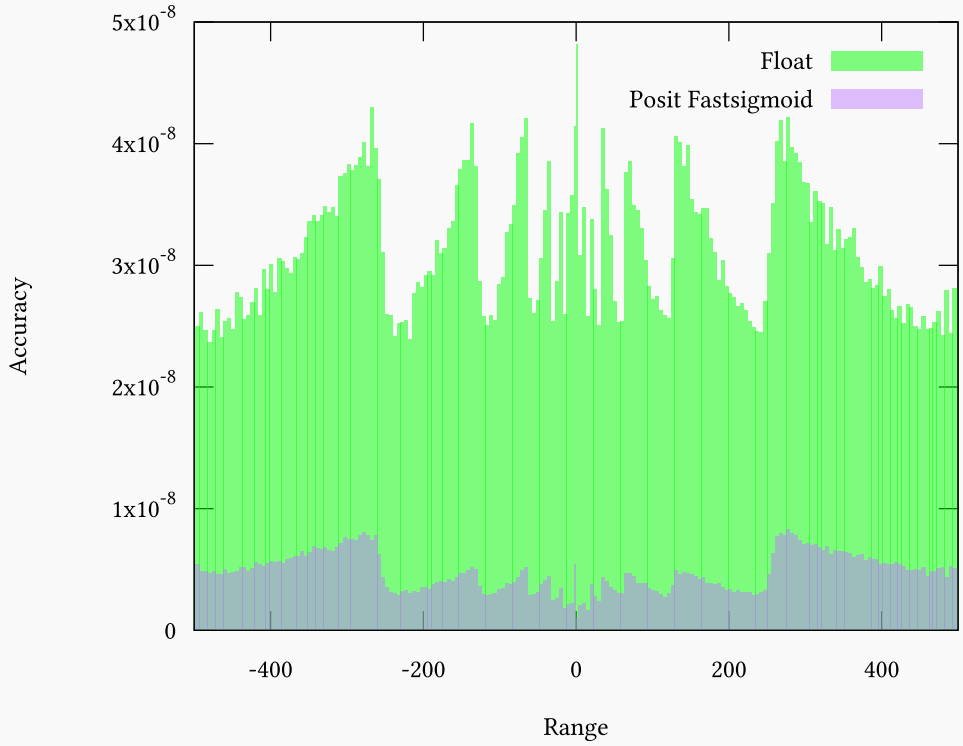
Impact de l'exposant

- **Précision vs Plage** : Plus l'exposant est important, plus la plage de valeur le sera
- **Granularité des valeurs** : Au contraire, un petit exposant permettra une plus grande définition des nombres autour de 1



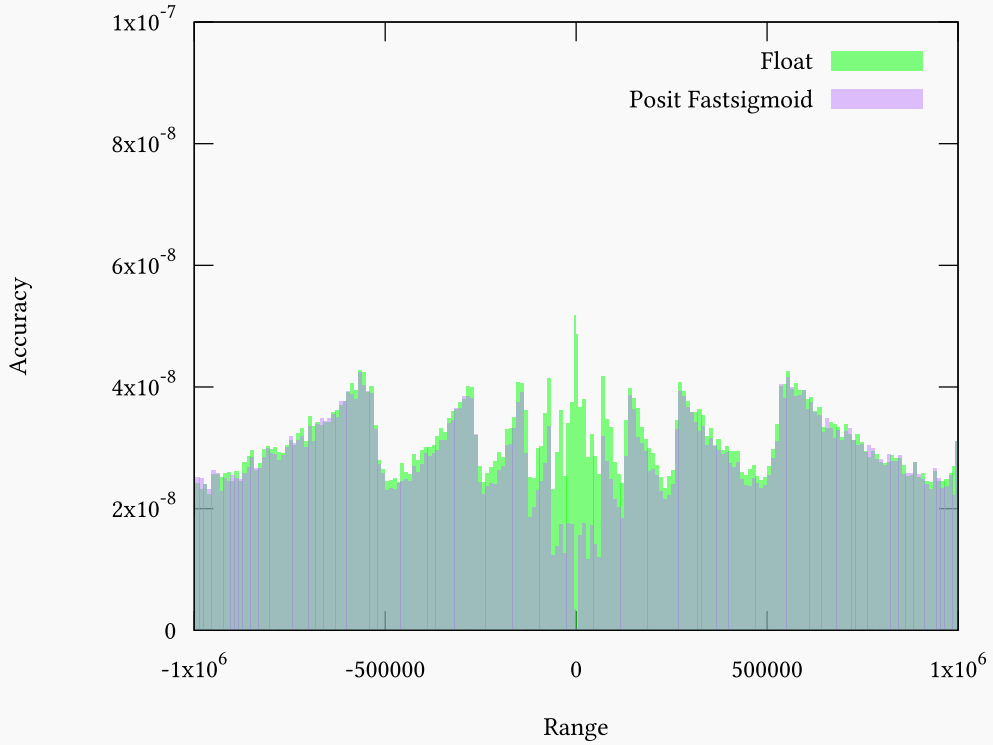
Précision

- **La précision autour de 1**



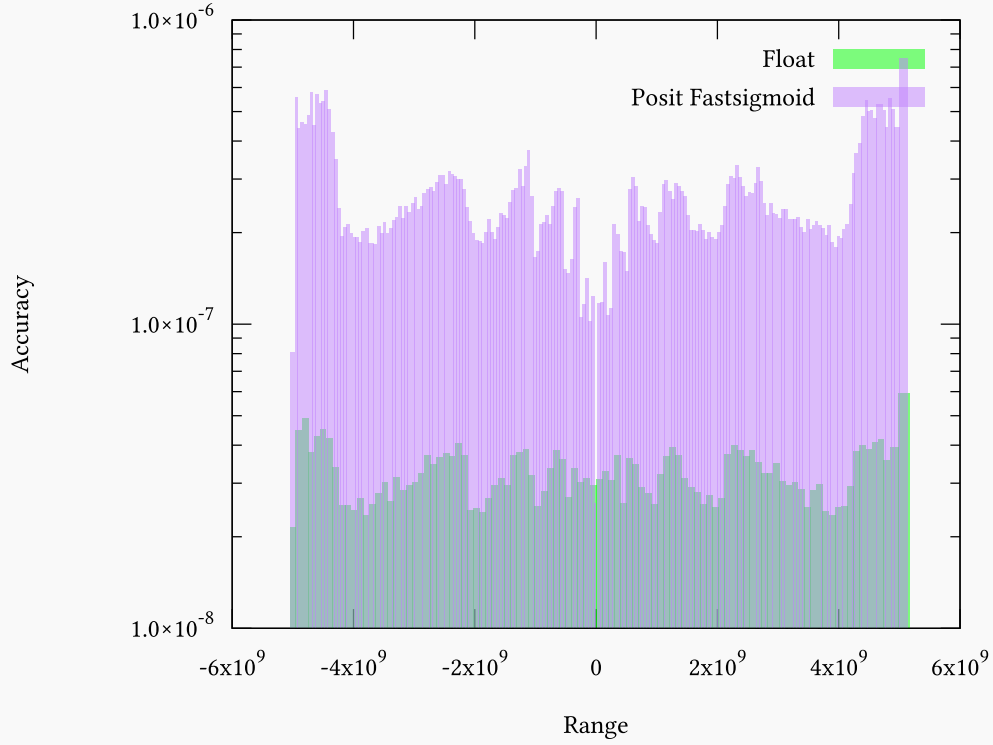
Précision

- **La précision autour de 1**
- **La précision dans l'intervalle de -1×10^6 à 1×10^6**



Précision

- **La précision autour de 1**
- **La précision dans l'intervalle de** -1×10^6 **à** 1×10^6
- **La précision au-delà de** 1×10^6 **et en dessous de** -1×10^6



Zone de couverture

MaxPos Posit

- Pas d'arrondi silencieux pour Posit GCC et Softposit

MaxPos Litterature	MaxPos FastSigmoid	MaxPos Ceralaine	MaxFloat
$2.126765e+37$	$2.126765e+37$	$1.329228e+36$	$3.402823e+38$

- Avantage : Code plus résiliant
- Inconvenient : Impossibilité de connaitre la taille de l'arrondi

Cas d'études résultat avec Posit

Inversion de matrice

i Float 32 bits IEEE

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401987 3.943966
      7.831672 7.984796 ]
rms = 0.000540
```

Inversion de matrice

i Float 32 bits IEEE

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401987 3.943966
      7.831672 7.984796 ]
rms = 0.000540
```

i Fastsigmoid Posit 32 bits ES 2

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401946 3.943851
      7.830993 7.984435 ]
rms = 0.000044
```

Inversion de matrice

i Float 32 bits IEEE

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401987 3.943966
      7.831672 7.984796 ]
rms = 0.000540
```

i Fastsigmoid Posit 32 bits ES 2

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401946 3.943851
      7.830993 7.984435 ]
rms = 0.000044
```

i Ceralaine Posit 32 bits ES 2

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401852 3.943811
      7.830964 7.984383 ]
rms = 0.000025
```

Inversion de matrice

i Float 32 bits IEEE

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401987 3.943966
      7.831672 7.984796 ]
rms = 0.000540
```

i Fastsigmoid Posit 32 bits ES 2

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401946 3.943851
      7.830993 7.984435 ]
rms = 0.000044
```

i Ceralaine Posit 32 bits ES 2

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401852 3.943811
      7.830964 7.984383 ]
rms = 0.000025
```

i Posit 32 bits ES 0

```
A = [ 8.401877 3.943829
      7.830992 7.984400 ]
B = [ 8.401877 3.943829
      7.830992 7.984400 ]
rms = 0.000000
```

Simulation Moléculaire

i Résultats IEEE 64 bits

```
Energy verlet: 758.659959 = -1085.330020 + 1843.989979  Temperature: 309.185229
Sum of verlet forces: -1.134026e-11
```

Simulation Moléculaire

i Résultats IEEE 64 bits

```
Energy verlet: 758.659959 = -1085.330020 + 1843.989979  Temperature: 309.185229
Sum of verlet forces: -1.134026e-11
```

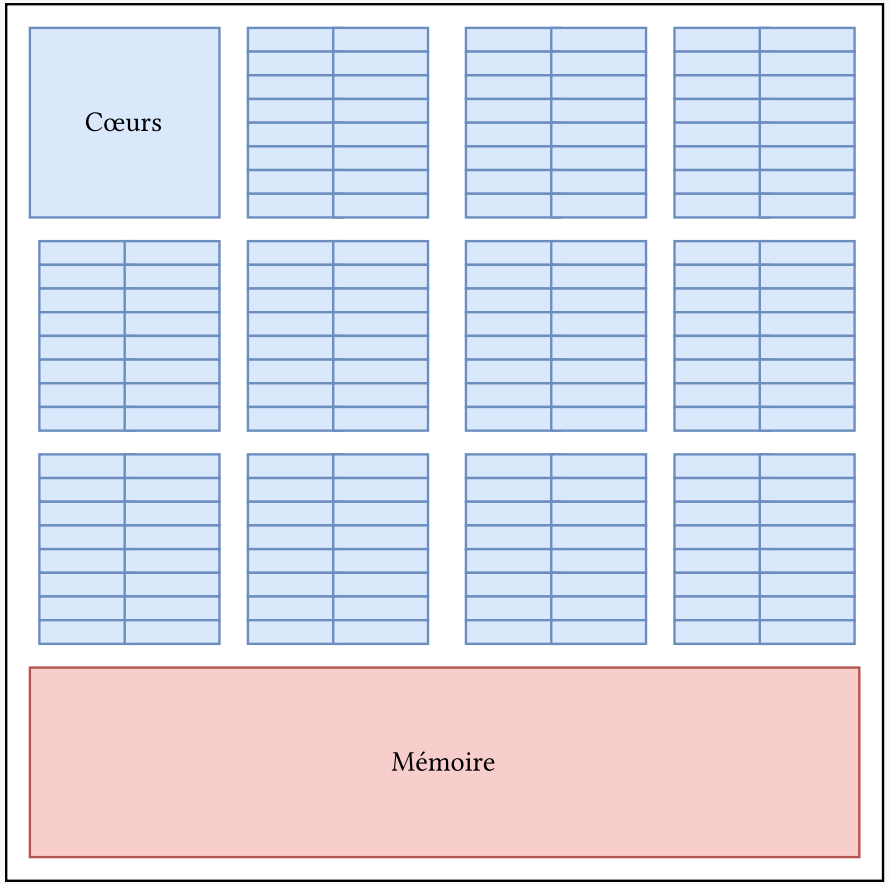
i Résultats Posit 64 bits

```
Energy verlet: 797.305560 = -1046.684419 + 1843.989979  Temperature: 309.185229
Sum of verlet forces: 1.932676e-12
```

Posit sur matériel GPGPU : RacEr

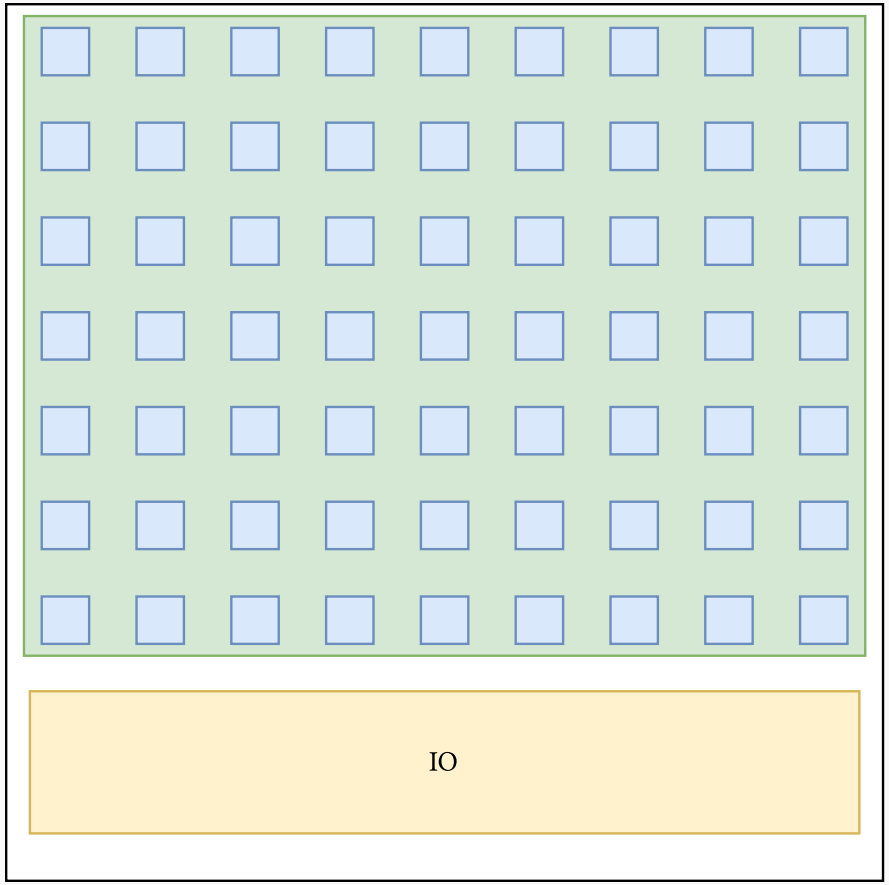
GPGPU

- **GPGPU** (General-Purpose Graphics Processing Unit) :
  - Processeur spécialisé
    - Les calculs massivement parallèles.
    - Dédié à certains types de calculs.
    - Nombre de coeurs très élevé



FPGA

- **FPGA** (Field-Programmable Gate Array) :
  - Circuit intégré reconfigurable
  - Offre une grande flexibilité pour l'implémentation d'architectures.



Variation des valeurs en fonction du matériel

- Les mesures effectuées montrent que la précision numérique n'est pas directement influencée par le matériel utilisé (GPU, CPU, FPGA), mais est principalement déterminée par la manière dont les nombres sont représentés et traités dans les calculs.

Précision

Perte de 1 chiffre significatif dans la golden zone par rapport au standard IEEE

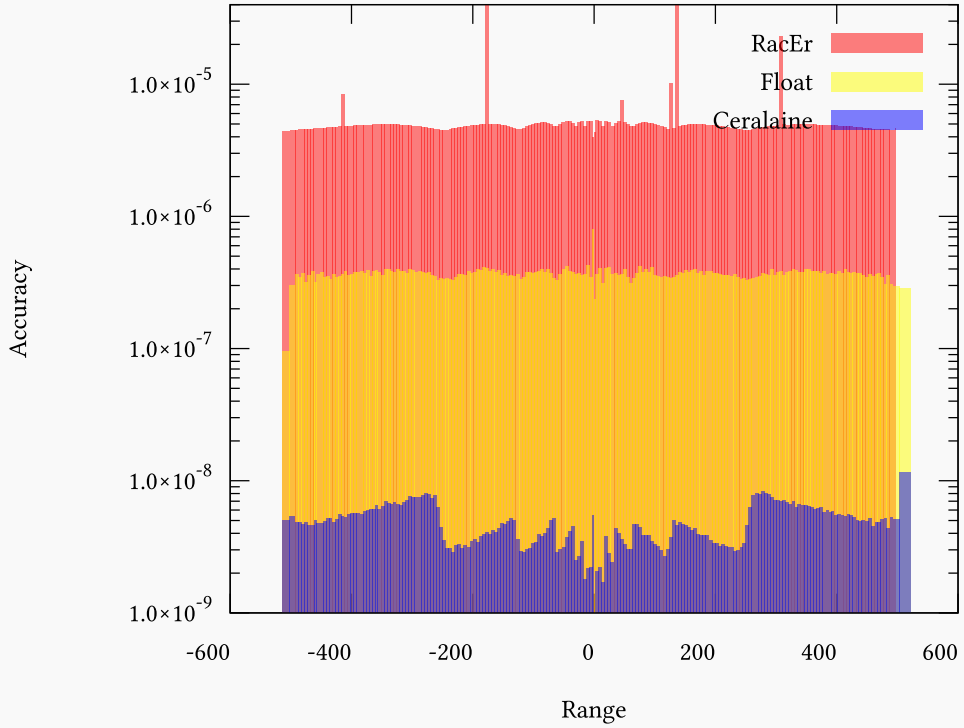


Figure 9: Précision de RacEr dans l'intervalle −500 à 500

Zone de couverture

Inversion des zones :

- Golden zone théorique moins précise
- Zone de couverture plus précise

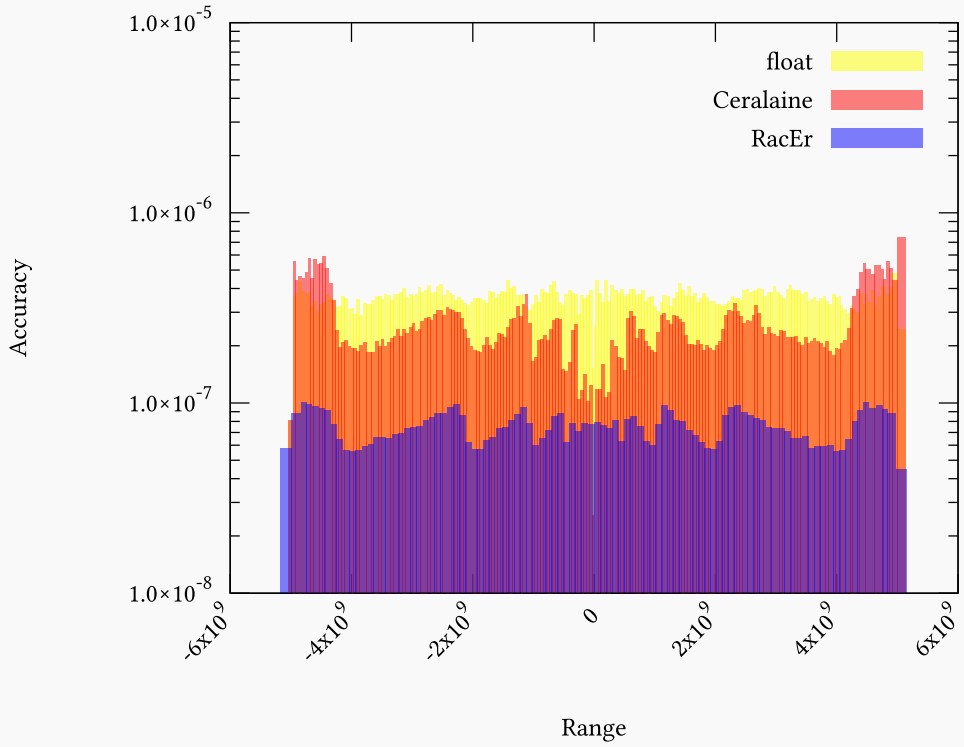


Figure 10: Précision de RacEr dans l'intervalle $-5e^9$ à $5e^9$

Limite de l'implémentation matérielle

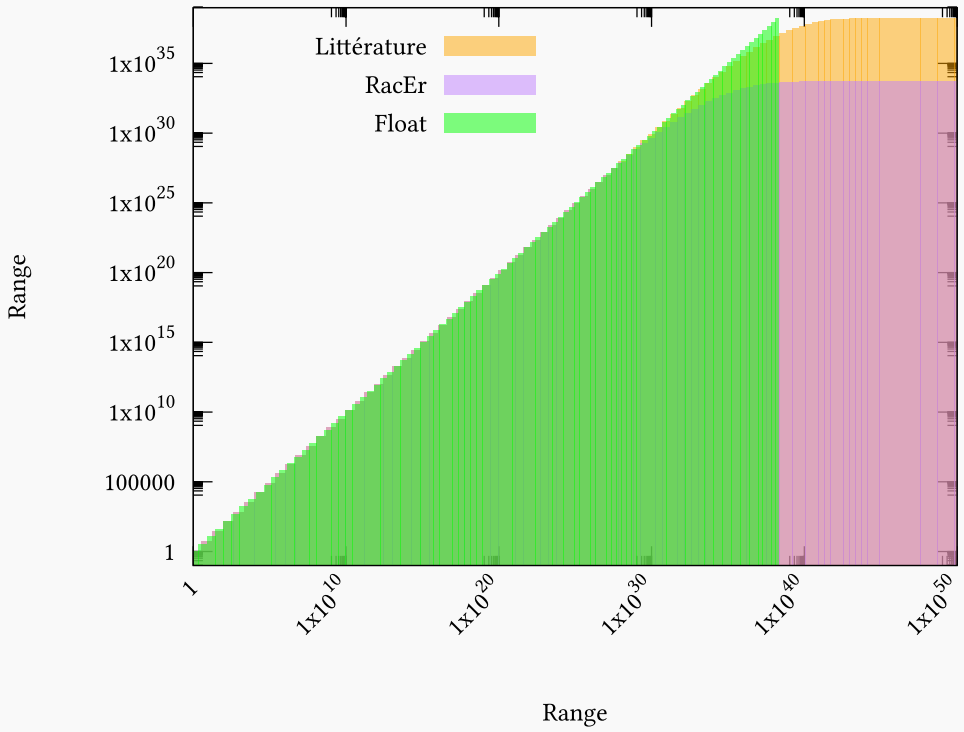


Figure 11: Limite de couverture de RacEr

L'implémentation RacEr permet l'arrondi silencieux mais sur une valeur limite de $5.192297e+33$

MaxPos Litterature	MaxPos Ceralaine	MaxPos RacEr
$2.126765e+37$	$1.329228e+36$	$5.192297e+33$

Accès 2D

Exemple

Nous avons remarqué que cette méthode entraînait un décalage de 1.

En utilisant un code de copie-colle mémoire de base, nous avons obtenu ce résultat : :
Entrée A[4] = {1,2,3,4}
Sortie B[4] = {0,1,2,3}

```
int start_y = __RacEr_tile_group_id_y
            * block_size_y;
int start_x = __RacEr_tile_group_id_x
            * block_size_x;
int end_y = start_y + block_size_y;
int end_x = start_x + block_size_x;
for (int iter_y = start_y +
     __RacEr_y; iter_y < end_y;
     iter_y += RacEr_tiles_Y)
    for (int iter_x = start_x +
          __RacEr_x; iter_x < end_x;
          iter_x += RacEr_tiles_X)
          B[iter_y * n + iter_x] =
           A[iter_y * n + iter_x];
```

Autres difficultés rencontrées

- Fonctions essentiels au débuggage erronnées
- Manque de documentation
- Impossible d'utiliser des type custom

Exemple

```
typedef struct
{
  posit x;
  posit y;
  int cluster;
} Point;
```

Programmation sur FPGA

Modèle de carte

Xilinx Kria KV260

- ARM Cortex-A53
- 230 000 LUTs
  - 435 LUTs pour addition IEEE
  - 719 LUTs pour addition Posit

Conclusion

Conclusion

- Freins à l'utilisation :
  - **Aucun matériel implémentant Posit nativement**
  - **Implémentations très aléatoire de Posit**
- Éléments prometteurs :
  - **Précision accrue**
  - **Flexibilité des représentations**
  - **Réduction des erreurs d'arrondi**
  - **Optimisation pour des applications spécifiques**
  - **Gain en espace**