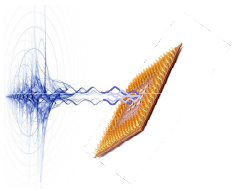
Conversion en virgule fixe pour les applications de traitement numérique du signal

ARCHI'09

Daniel MENARD
INIRA/IRISA
Projet CAIRN

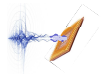


INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE — INRIA — centre de recherche RENNES - BRETAGNE ATLANTIQUE

UNIVERSITÉ DE RENNES 1

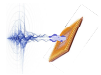
ENSSAT LANNION

Plan



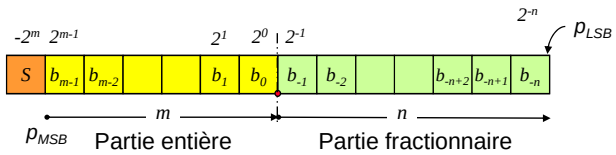
- Introduction
- Détermination de la position de la virgule
- Optimisation de la spécification virgule fixe
- Conclusion

Codage en virgule fixe complément à 2

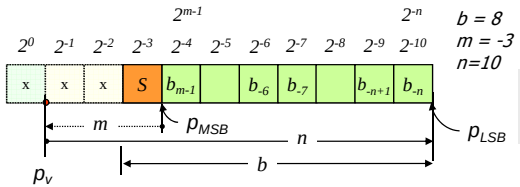


- Définition :

$$x = -2^m S + \sum_{i=-n}^{m-1} b_i 2^i$$



- m : distance (en nombre de bits) entre la position du bit le plus significatif p_{MSB} et la position de la virgule p_V
- n : distance entre la position de la virgule p_V et la position du bit le moins significatif p_{LSB}

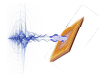


$b = m + n + 1$ bits
format: (b,m,n)

$b = 8$
$m = -3$
$n = 10$

ex: 0.09472 (8,-3,10)
01100001
0.0625+0.03125+2^{-10}

Codage en virgule fixe complément à 2



- Domaine de définition du codage : $\left[-2^m, 2^m - 2^{-n}\right]$
- Pas de quantification : $q = 2^{-n}$
- Exemple de codage :
  - (6,3,2), Q_2^6

0111.11	7.75
0111.10	7.5
0000.11	0.75
0000.10	0.5
0000.01	0.25
0000.00	0
1111.11	-0.25
1111.10	-0.5 ← -0.5 = -8+7. 5
1111.01	-0.75
1000.01	-7.75 ← -7.75 = -8+0.25
1000.00	-8

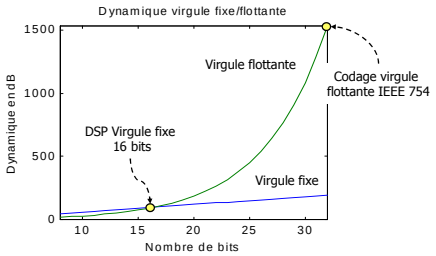
Comparaison virgule fixe / flottante

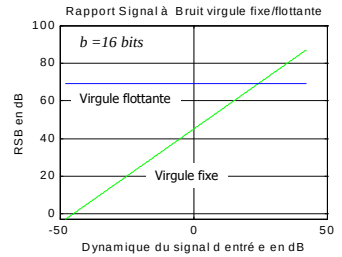
- Niveau de dynamique

$$D_{N\,(dB)} = 20.\log\left(\frac{\max(|x|)}{\min(|x|)}\right)$$

- Rapport Signal à Bruit de Quantification

$$\rho_{dB} = 10.\log\left(\frac{P_s}{P_e}\right)$$





Arithmétique virgule fixe

- Arithmétique :
  - Dynamique limitée : [-X_{max} et X_{max}]
    - Possibilité de débordement ⇨ nécessité de recadrer les données
- Développement :
  - Temps de développement plus long
    - Étude la dynamique des données, détermination du codage et des recadrages
- Architecture :
  - Opérateurs plus simples
  - Largeur des données b_{nat} : 16 bits
    - Efficacité énergétique plus importante, **consommation moins importante**
    - Processeur **plus rapide**
    - Processeur **moins cher** (surface du circuit moins importante)
- Marché : applications *grand public*
  - 95% des ventes en 96

TMS320C62x :
- f_{CLK} : 300 MHz (150 MHz - 300 MHz)
- On Chip Memory 72 Kbytes ⇨ 896 Kbytes
- Price : $9 ⇨ 102

TMS320C64x :
- f_{CLK} : 1 GHz (300 MHz - 1GHz)
- On Chip Memory 160 Kbytes ⇨ 1056 Kbytes
- Price : $18 ⇨ 219

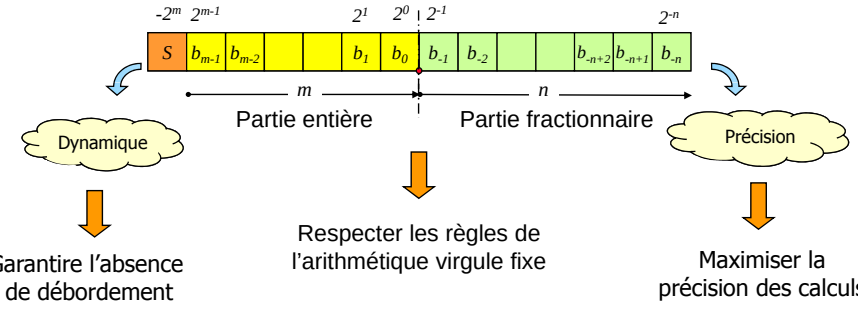
Arithmétique virgule flottante

- Arithmétique :
  - Dynamique importante : 1500 dB pour 32 bits
- Développement
  - Temps de développement plus court
    - Recadrage des données assuré par le processeur
    - Compilateur de langage de haut niveau plus efficace : plus grande portabilité
- Architecture :
  - Largeur des données : 32 bits
  - Opérateurs plus complexes (gestion de la mantisse et de l'exposant)
    - Processeur plus cher et consommant plus
- Marché
  - Applications nécessitant une grande dynamique : audionumérique
  - Applications avec des faibles volumes

TMS320C67x :
- f_{CLK} : 300 MHz (100 MHz - 300 MHz)
- On Chip Memory 72 Kbytes ⇨ 264 Kbytes
- Price : $14 ⇨ 105

Codage des données

- Codage des données en virgule fixe :
  - Définir la position de la virgule :
    - Nombre de bits pour la partie entière et pour la partie fractionnaire

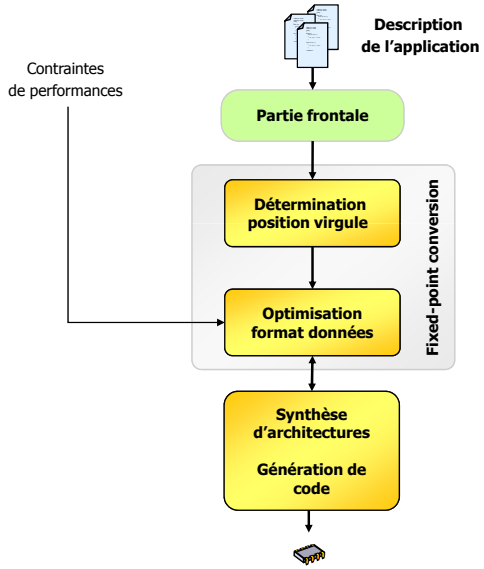


RSBQ (Rapport Signal à Bruit de Quantification)

Deux étapes pour la conversion

- Détermination de la position de la virgule m_{xi}
  - Déterminer le nombre de bits minimal pour représenter la partie entière permettant d'éviter les débordements
    - Nécessité de déterminer le domaine de définition de chaque variable
- Optimiser la spécification virgule fixe
  - Minimiser le cout de l'implantation
  - Garantir les performances de l'application
    - Limiter la dégradation de performances par rapport à la précision infinie.

Conversion en virgule fixe

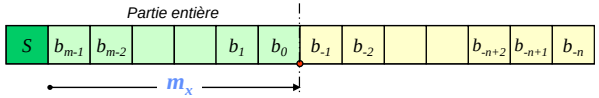


Détermination de la position de la virgule

1. **Détermination de la dynamique**
   1. **Objectifs**
   2. **Approches statistiques**
   3. **Approches analytiques**
2. **Détermination de la position de la virgule**

Objectifs

- Estimation du domaine de définition $[x_{min}, x_{max}]$ de x
  - En déduire la position de la virgule m_x



$$m_x = \lfloor \log_2 (\max (|x_{\min}|, |x_{\max}|)) \rfloor + 1$$

- Critères de qualité pour l'estimation
  - Précision : minimiser l'erreur d'estimation
    - Éviter la présence de bits non utilisés au niveau des bits les plus significatifs de la donnée
  - Qualité : garantir l'absence de débordement

Détermination de la position de la virgule

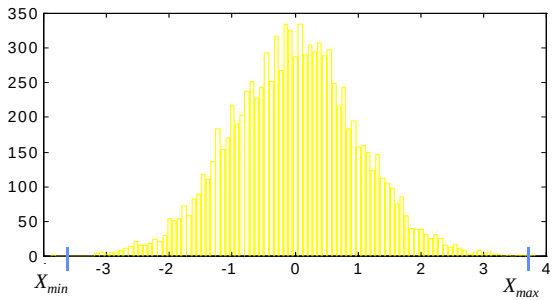
1. **Détermination de la dynamique**
   1. Objectifs
   2. **Approches statistiques**
   3. Approches analytiques
2. Détermination de la position de la virgule

Méthodes statistiques

- Objectifs :
  - Détermination de la dynamique d'une donnée à partir de ses paramètres statistiques obtenus par simulation
    - o Simulation de l'algorithme en virgule flottante
    - o Collecte des échantillons pour chaque donnée
    - o Détermination des paramètres statistiques
- Techniques :
  - o Instrumentation du code
    - Types C++ pour collecter les données

Méthodes statistiques

- Détermination des valeurs minimales X_{min} et maximales X_{max} **[Aam 01]**
  - Technique simple mais très sensible au choix des stimuli d'entrée



Méthodes statistiques

- Méthode basée sur les moments **[Kim98]**
  - Distribution uni-modale et symétrique

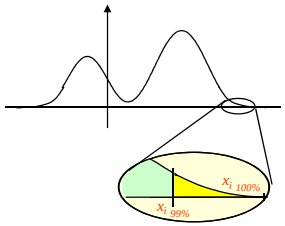
$$\tilde{x}_{i\max} = \left|\mu_{x_i}\right| + (k_{x_i} + 4)\sigma_{x_i}$$

- Kurtosis : $k_{x_i} = \frac{E\left[(x_i - \mu_{xi})^4\right]}{\sigma_{x_i}^4} - 3$ *(coefficient d'aplatissement)*
- Skewness : $s_{x_i} = \frac{E\left[(x_i - \mu_{xi})^3\right]}{\sigma_{x_i}^3}$ *(coefficient de dissymétrie)*
- Test uni-modale : $(-1.2 < k_{x_i} < 5)$
- Test symétrique : $s_{x_i} \approx 0$

- Distribution multi-modale ou non symétrique

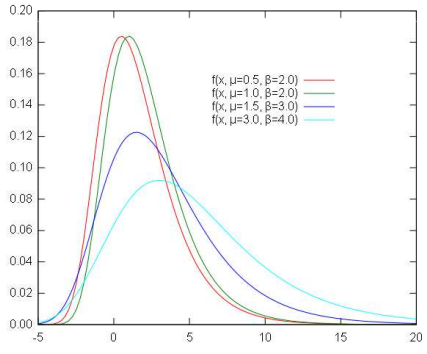
$$\begin{cases} |x_{i\,\min}| = |x_{i_{0\%}}| + r_R |x_{i_{0\%}} - x_{i_{1\%}}| \\ |x_{i\,\max}| = |x_{i_{100\%}}| + r_R |x_{i_{100\%}} - x_{i_{99\%}}| \end{cases}$$

$x_{i_{99\%}}$ défini tel que $P(x_i < x_{i_{99\%}}) = 0.99$



- Théorie des valeurs extrêmes **[OZER08]**
  - Réalisation de N simulations avec des stimuli différents
    - o N valeurs pour les minima et maxima
  - Distribution des minima et maxima : distribution de Gumbel

$$F(x) = e^{-e^{\frac{(x-\alpha)}{\beta}}}$$



- Estimation de la moyenne et de la variance
  - o Moyenne : $\mu = \alpha + \beta\gamma$
  - o Variance : $\sigma = \frac{\pi^2}{6}\beta^2$
    - γ : constant de Euler (0.57721)
- Probabilité de non-débordement :

$$P[X < A_{\max}] = c \qquad c = e^{-e^{\frac{A_{\max}-\alpha}{\beta}}}$$

Détermination de la position de la virgule

Méthodes analytiques

- Arithmétique d'intervalle [Kea 96] :
  - Propagation de la dynamique des entrées au sein de l'application

Opérations	*min (z)*	*max (z)*
z=x+y	*min(x)+min(y)*	*max(x)+max(y)*
z=x-y	*min(x)-max(y)*	*max(x)-min(y)*
z=x×y	*min(E)*	*max(E)*

E =(min(x)min(y),min(x)max(y), min(y)max(x),max(x)max(y))

  - o Traitement des structures non-récursives
  - o Estimation pessimiste
    - Absence de prise en compte de la corrélation entre les données

Méthodes analytiques

- Arithmétique affine :
  - Forme affine d'une donnée x :

$$x = x_0 + \sum_i x_i \varepsilon_i \qquad \varepsilon_i \in [-1;1]$$

  - o ε_i symbole d'incertitude
  - o Conservation de la source des erreurs
    - Prise en compte de la dépendance des données

Méthodes analytiques

- Opérations affines
  - o Addition :

$$x_f \pm y_f = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i)\varepsilon_i$$

    - Résultats sous forme affine

  - o Multiplication :

$$x_f y_f = x_0 y_0 + \sum_{i=1}^{n} (y_0 x_i + x_0 y_i)\varepsilon_i + \sum_{i=1}^{n} x_i \varepsilon_i \sum_{i=1}^{n} y_i \varepsilon_i$$

    - Utilisation d'une simplification pour obtenir une forme affine

$$x_f y_f \approx x_0 y_0 + \sum_{i=1}^{n} (y_0 x_i + x_0 y_i)\varepsilon_i + r\varepsilon_k$$

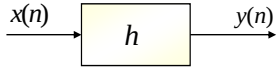
Comparaison

- Comparaison arithmétique d'intervalle et affine
  - o Exemple : données x, intervalle [0;1]

		Arithmétique d'intervalles	Arithmétique affine
• Cas 1 :	$y = x(x-1)$	$[-1;0]$	$\left[-\frac{1}{4};0\right]$
• Cas 2 :	$y = x^2 - x$	$[-1;1]$	$\left[-\frac{1}{4};0\right]$
• Cas 3 :	$y = \left(x-\frac{1}{2}\right)^2 - \frac{1}{4}$	$\left[-\frac{1}{4};0\right]$	$\left[-\frac{1}{4};0\right]$

Méthodes analytiques

- Normes pour systèmes linéaires



- Normes L1 :

$$y_{\max 1} = \max_n \left(|x(n)|\right). \sum_{m=-\infty}^{\infty} |h(m)|$$

  - Systèmes linéaires non-récursifs : résultats identiques à ceux obtenus avec l'arithmétique d'intervalle

- Norme Chebychev :

$$y_{\max 2} = \max_n \left(|x(n)|\right) \max_\omega \left(|H(\omega)|\right)$$

  - Signal d'entrée du type $x(n) = \cos(\omega.n.T)$

Comparaison des méthodes

	Méthode statistique	Méthode analytique
Précision	Erreur d'estimation faible	Méthode conservatrice (estimation dans le pire cas)
Qualité	Pas de garantie sur l'absence de débordement Fonction de la représentativité des signaux	Garantie sur l'absence de débordement
Structures traitées	Toutes	Structures linéaires et non-linéaires non-récursives
Connaissances nécessaires	Signaux d'entrée	Domaine de définition des entrées

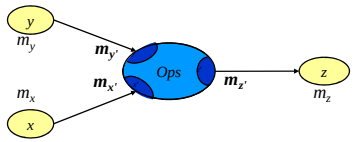
Détermination de la position de la virgule

Détermination de la position de la virgule

- Données

$$m_x = \left\lfloor \log_2 \left(\max_n \left(|x(n)| \right) \right) \right\rfloor + 1$$

- Opérations :



  - Multiplication

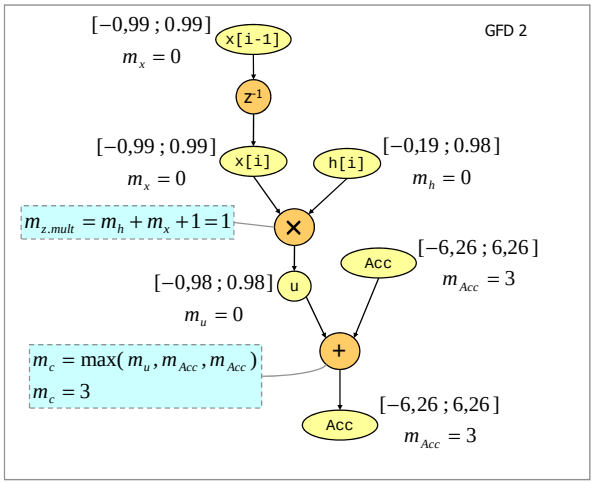
$$m_{z'} = m_x + m_y + 1$$

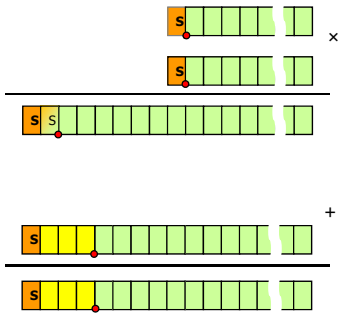
  - Addition **(sans bits de garde)**
    - Définition d'un format commun

$$m_c = \max(m_x, m_y, m_z)$$

Exemple filtre FIR

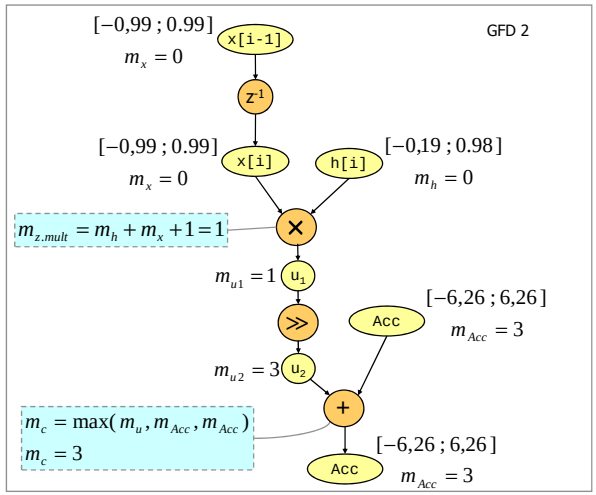
- Détermination de la position de la virgule

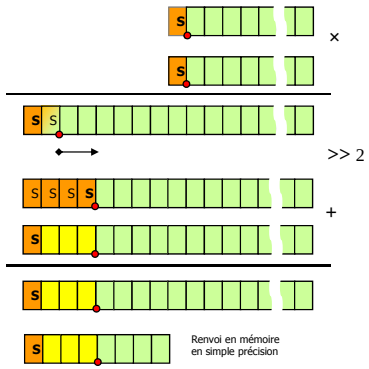




Exemple filtre FIR

- Insertion des opérations de recadrage





Optimisation de la spécification virgule fixe

1. **Objectifs**
2. Evaluation de la précision
3. Implantation matérielle
4. Implantation logicielle

Implantation matérielle

- Conception de l'architecture (ASIC – FPGA)
  - Sélection des opérateurs
    - Possibilité de choisir la largeur des opérateurs, des registres et de la mémoire
- Objectifs :
  - Minimiser le cout de l'architecture (C_{arch})
    - Surface du circuit
    - Consommation d'énergie
  - Maintenir les performances de l'application ($Perf$)

$$\underset{b_k \in N}{\text{Min}}\left(C_{arch}(b_k)\right) \quad \text{tel que} \quad Perf(b_k) \geq Perf_{seuil}$$

Implantation logicielle

- Mapping d'un algorithme sur une architecture figée (DSP – µP)
  - Sélection des instructions classiques, SWP, multi-précision
    - Différentes largeurs de données possibles $B = \{8,16,32\}$
  - Sélection de la localisation des recadrages
- Objectifs :
  - Minimiser le cout de l'implantation (C_{imp})
    - Temps d'exécution
    - Consommation d'énergie
    - Taille du code
  - Maintenir les performances de l'application ($Perf$)

$$\underset{b_k \in B}{\text{Min}}\left(C_{imp}(b_k)\right) \quad \text{tel que} \quad Perf(b_k) \geq Perf_{seuil}$$

Optimisation spécification virgule fixe

- Processus d'optimisation de la spécification virgule fixe
  - Evaluation des performances de l'application
    - Utilisation d'une métrique intermédiaire : précision des calculs
    - Nécessité de déterminer une contrainte de précision
  - Evaluation du coût de l'implantation
  - Algorithme d'optimisation

Optimisation de la spécification virgule fixe

Evaluation des performances

- Evaluation directe des performances
  - Méthodes basées sur la simulation
    - Temps d'évaluation très important
- Evaluation d'une métrique intermédiaire de précision
  - Objectifs :
    - Diminuer le temps nécessaire pour évaluer la métrique
    - Deux approches : analytique – basée sur la simulation
    - Nécessité de faire un lien entre les performances et la métrique de précision
  - Erreur de quantification associée à une donnée x
    - Différence entre la donnée en précision finie (virgule fixe) et la donnée en précision infinie (valeur exacte)

$$b_x = x_{\text{précision finie}} - x_{\text{précision infinie}}$$

Métrique d'évaluation de la précision

- Différents types de métrique :
  - o Nombre de bits significatifs
  - o Intervalle de l'erreur
  - o Moments de l'erreur
- Propriété de l'erreur de quantification
  - o Variable aléatoire (ergodique et stationnaire)
    - Caractérisée par sa puissance (moment d'ordre 2) P_{b_x}
- Métrique d'évaluation de la précision
  - o Rapport Signal à Bruit de Quantification
    - P_y : puissance du signal
    - P_{b_y} : puissance du bruit de quantification

$$RSBQ = \frac{P_y}{P_{b_y}}$$

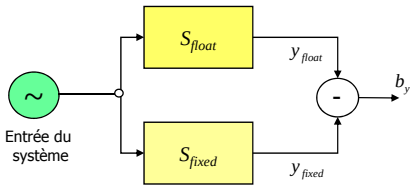
Approches disponibles

- Nombre de bits significatifs
  - Arithmétique stochastique
- Intervalle de l'erreur
  - Analyse min/max (simulation)
  - Arithmétique d'intervalle (analytique)
  - Arithmétique affine (analytique)
  - Théorie des valeurs extrêmes (simulation)
  - Théorie de la perturbation
- Moments du bruit
  - Théorie de la perturbation
  - Modèles de bruit

Optimisation de la spécification virgule fixe

Méthodes basées sur la simulation

- Principe
  - Détermination de la puissance du bruit de quantification à partir de la simulation du système en virgule fixe (y_{fixed}) et en virgule flottante (y_{float})
    - o La sortie en virgule flottante est considérée comme la référence
      - Hypothèse valide si l 'erreur liée à l 'arithmétique virgule flottante est négligeable par rapport celle liée à la virgule fixe
        - – La largeur des données en virgule fixe doit rester faible



$$P_{b_y} = \frac{1}{N_{pts}} \sum_{n=0}^{N_{pts}} \left(y_{float} - y_{fixed} \right)^2$$

Méthodes basées sur la simulation

- Utilisation de librairies pour émuler la virgule fixe
  - Utilisation de classe C++ : systemC, gFix [Kim 98]
    - Temps de simulation élevés
  - Utilisation des caractéristiques de la machine hôte pour accélérer la simulation en virgule fixe
    - Utilisation de types optimisés : pFix [Kim 98]
    - Génération d'un code optimisé : FRIDGE [Ked 01]
      - Réduction des temps de simulation
      - Augmentation du temps nécessaire pour générer le code utilisé pour la simulation

☞ Temps d'optimisation du format des données très élevé [Sun 95]
  ☞ Une nouvelle simulation en virgule fixe est requise dès que le format d'une donnée est modifié

Méthodes basées sur la simulation

- Exemples de temps de simulation
  - Filtre IIR d'ordre 4, (PC pentium 100 MHz) [Sun 95]

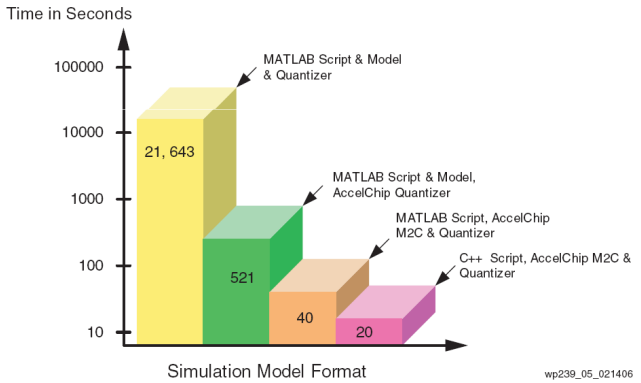
Type	Flottant	gFix	pFix	VHDL	SPW
Temps de simulation (s)	2.19	340	16.3	181	60
Rapport fixe / flottant	1	155	7.4	82.6	27.4

  - Comparaison des temps de simulation en virgule fixe et en virgule flottante pour 6 applications [Ked 01]
    - FIR, DCT, IIR, FFT, auto-corrélation, produit de matrice

Type	Flottant	SystemC	SystemC précision limitée	Code optimisé
Rapport fixe / flottant	1	540	120	3.6

Méthodes basées sur la simulation

- Outil AccelChip (Xilinx) [Hill06]



FFT Example Simulation Run Times

Optimisation de la spécification virgule fixe

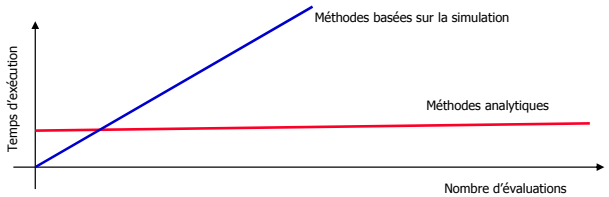
1. Objectifs
2. **Evaluation de la précision**
   1. Introduction
   2. Approches basées sur la simulation
   3. **Approches analytiques**
   4. Détermination de la contrainte de précision
3. Implantation matérielle
4. Implantation logicielle

Approche analytique

- Objectifs :
  - Génération d'une fonction mathématique définissant l'expression de la précision en fonction du format des données

$$P_{b_y}(\mathbf{b}) = \sum_{i=0}^{Ne} K_i \sigma_i^2 + \sum_{i=0}^{Ne}\sum_{j=0}^{Ne} G_{ij}\mu_i\mu_j$$

  - Accélérer l'évaluation de la précision

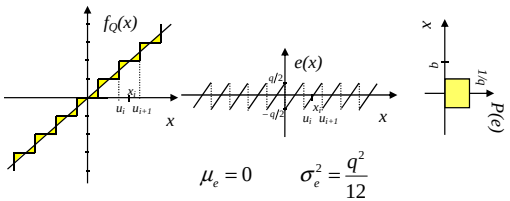


Optimisation de la spécification virgule fixe

Modèle de bruit de quantification généré

. Arrondi

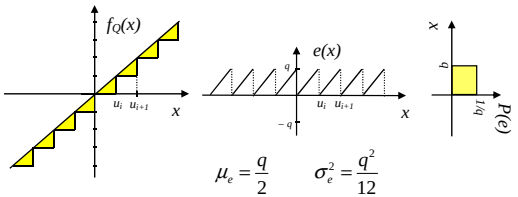
$Q(x) = k.q$ si $(k-1/2).q \le x < (k+1/2).q$



$$\mu_e = 0 \qquad \sigma_e^2 = \frac{q^2}{12}$$

. Troncature

$Q(x) = k.q$ si $k.q \le x < (k+1).q$



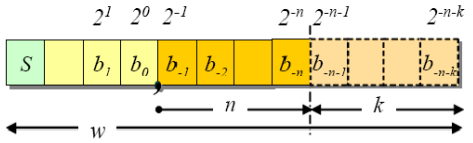
$$\mu_e = \frac{q}{2} \qquad \sigma_e^2 = \frac{q^2}{12}$$

Etude statistique

- $\{e(n)\}$ est une séquence d'un processus aléatoire continu et stationnaire
- $\{e(n)\}$ est décorrélée de $\{x(n)\}$
- $\{e(n)\}$ est un bruit blanc additif
- la distribution de probabilité de $\{e(n)\}$ est uniforme sur l'intervalle de quantification
- ergodicité : moyennes temporelles = moyennes statistiques
- moyenne μ_e = moyenne temporelle
- variance σ_e^2 = puissance du bruit
  variance = $q^2/12$

Quantification signal discret

- Bruit lié à l'élimination de k bits



- Pas de quantification **avant** élimination des k bits

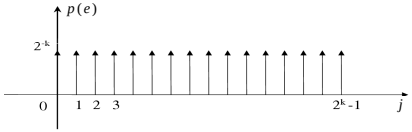
$$\Delta = 2^{-(n+k)}$$

Modèle de bruit de quantification généré

- Troncature :
  - Définition $x_Q = \lfloor x \cdot q^{-1} \rfloor \cdot q$

    $x_Q = kq \quad \forall x \in [k \cdot q; (k+1)q[$
  - Densité de probabilité

$$p_{e_q}(x) = \frac{1}{2^k} \sum_{j=0}^{2^k-1} \delta(x - j \cdot \Delta)$$



  - Moments : $\mu_b = \frac{q}{2}\left(1 - 2^{-k}\right)$ $\sigma_b^2 = \frac{q^2}{12}\left(1 - 2^{-2k}\right)$

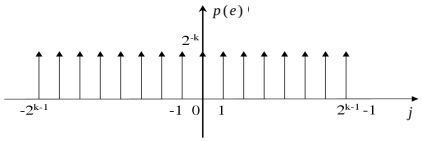
Modèle de bruit de quantification généré

- Arrondi conventionnel :
  - Définition $x_Q = \left\lfloor \left(x + \frac{1}{2}q\right) \cdot q^{-1} \right\rfloor \cdot q$

$$x_Q = \begin{cases} kq & \forall x \in [k \cdot q; (k+\frac{1}{2})q[\\ (k+1)q & \forall x \in [(k+\frac{1}{2})q; (k+1)q] \end{cases}$$

  - Densité de probabilité

$$p_{e_q}(x) = \frac{1}{2^k} \sum_{j=-2^{k-1}}^{2^{k-1}-1} \delta(x - j.\Delta)$$



  - Moments : $\mu_b = -\frac{q}{1}2^{-k}$ $\sigma_b^2 = \frac{q^2}{12}\left(1 - 2^{-2k}\right)$

Modèle de bruit de quantification généré

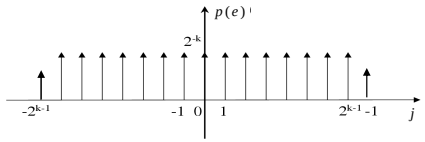
- Arrondi convergent :
  - Définition

$$x_Q = \begin{cases} kq & \forall\, x \in [k.q; (k+\frac{1}{2})q[\\ (k+1)q & \forall\, x \in](k+\frac{1}{2})q; (k+1)q] \\ kq & \forall\, x = q_{1/2} \quad \text{and} \quad b_{-n} = 0 \\ (k+1)q & \forall\, x = q_{1/2} \quad \text{and} \quad b_{-n} = 1 \end{cases}$$

$q_{1/2} = (k+\frac{1}{2})q$

  - Densité de probabilité

$$p_{e_q}(x) = \frac{1}{2^k} \sum_{j=-2^{k-1}+1}^{2^{k-1}-1} \delta(x - j.\Delta) + \frac{1}{2^{k+1}}\left(\delta(x - 2^{k-1}\Delta) + \delta(x + 2^{k-1}\Delta)\right)$$



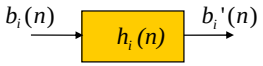
  - Moments : $\mu_b = 0$ $\sigma_b^2 = \frac{q^2}{12}\left(1 - 2^{-2k}\right)$

Modélisation du bruit propagé

- Modèle de propagation du bruit dans une opération arithmétique :
  - Addition $b_Z = b_X + b_Y$
  - Soustraction $b_Z = b_X - b_Y$
  - Multiplication $b_Z = y.b_X + x.b_Y$
  - Division $b_Z = b_X . \frac{1}{y} - b_Y \frac{x}{y^2}$

Puissance du bruit de quantification b_y

- Cas des systèmes linéaires [Men02b]



  - Propriétés : les sources de bruit b_i sont des bruits blancs

$$G_i = \sum_{n=-\infty}^{+\infty} h_i(n) = H_i\left(e^{j0}\right) \qquad K_i = \sum_{n=-\infty}^{+\infty} |h_k(n)|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|H_k\left(e^{j\Omega}\right)\right|^2 d\Omega$$

- Cas des systèmes à base d'opérations arithmétiques [Roc06]

$$b_i' = \mathbf{A_i} b_i \mathbf{D_i}$$

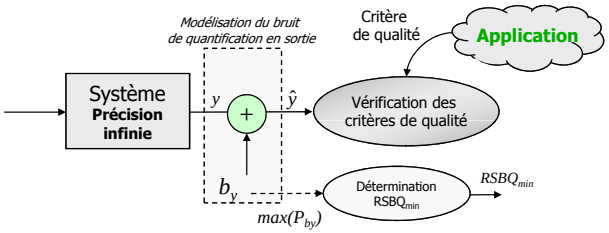
$$G_{ij} = \sum_{k=0}^{\infty}\sum_{m=0}^{\infty} E\left(\mathbf{A_i}(k)1\mathbf{D_i}(k)\mathbf{D_i^t}(k)1\mathbf{A_i^t}(k)\right)$$

$$K_i = \sum_{k=0}^{n} E\left(Tr\left[\mathbf{D_i}(k)\mathbf{D_i^t}(k)\right]\mathbf{A_i}(k)\mathbf{A_i^t}(k)\right)$$

Optimisation de la spécification virgule fixe

Détermination contrainte de précision

- Modélisation du comportement en virgule fixe



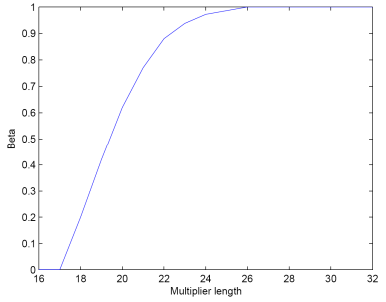
  - Recherche de la puissance de bruit maximale permettant de maintenir les performances de l'application

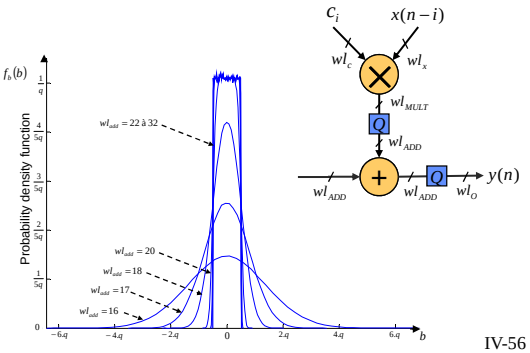
Détermination contrainte de précision

- Modèle de bruit en sortie d'un système en virgule fixe

$$b_p = v(\beta \times b_u + (1-\beta) \times b_n).$$

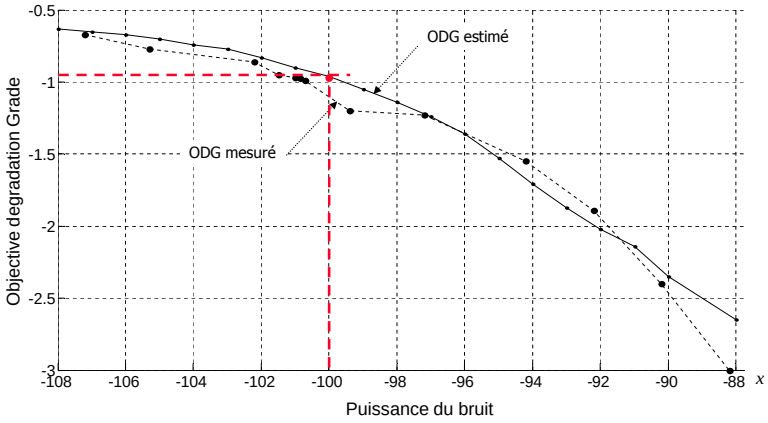
  - Exemple filtre FIR





Détermination contrainte de précision

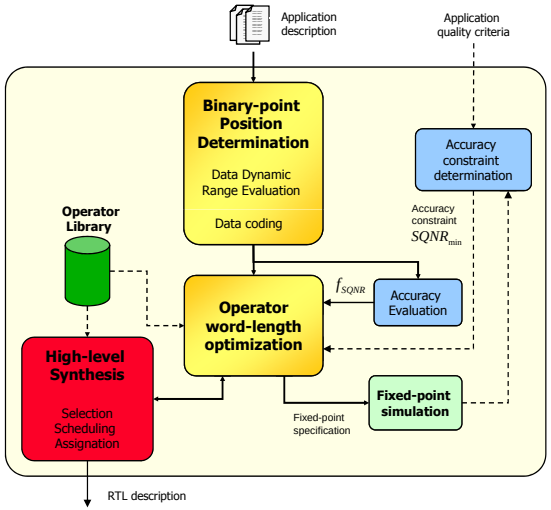
- Codeur/Décodeur MP3
  - Métrique de mesure de la qualité de compression : ODG



Optimisation de la spécification virgule fixe

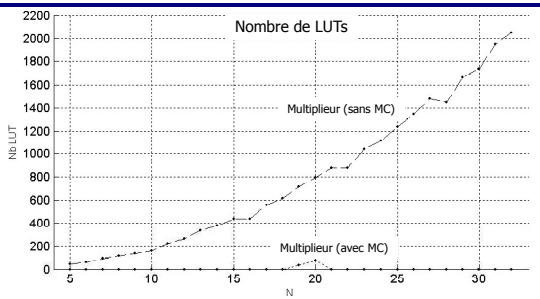
1. Objectifs
2. Evaluation de la précision
3. **Implantation matérielle**
4. Implantation logicielle

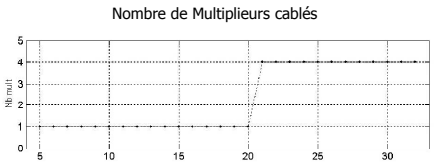
Méthode

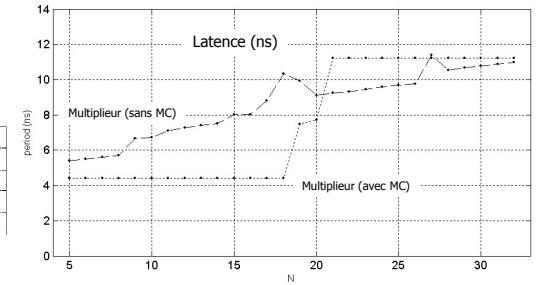


Opérateurs arithmétiques

- Virtex 4 (Xilinx)

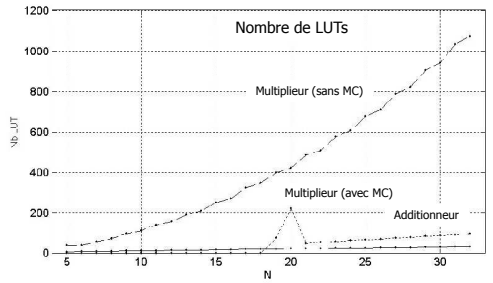


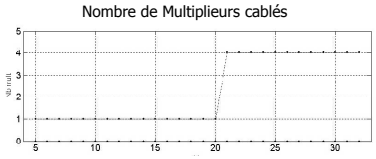


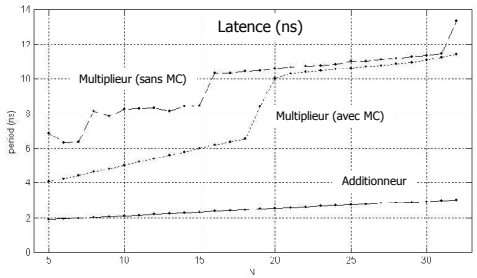


Opérateurs arithmétiques

- Stratix (Altera)



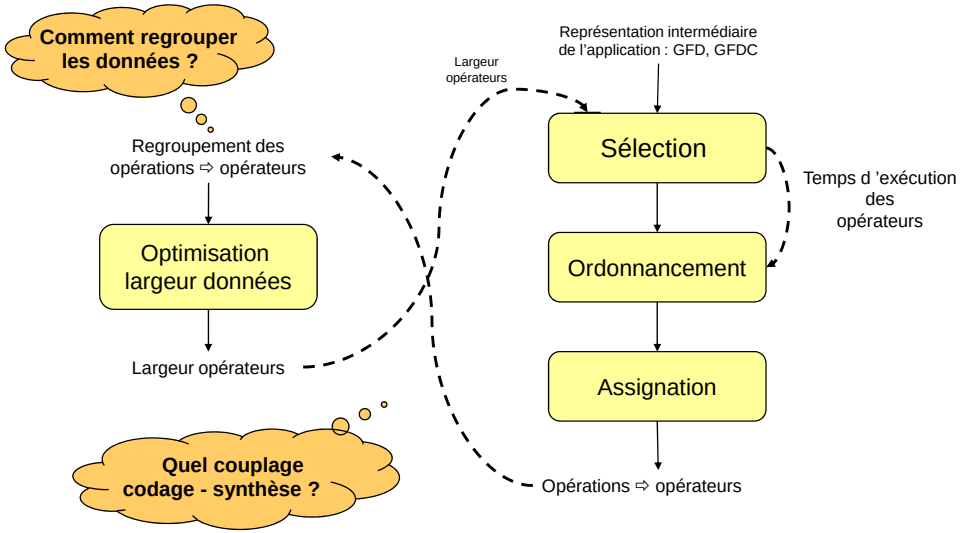




Niveau de regroupement des données

- Algorithme : toutes les données ont la même largeur
  - Le processus d'optimisation de la largeur des données peut être réalisé avant la synthèse d'architecture [Mar 01]
    - Qualité de l'implantation en terme de surface ?
- Bloc : toutes les données d'un bloc ont la même largeur
    - Limitation du nombre de variables pour l'optimisation
  - Regroupement dirigé par
    - L'analyse du graphe de l'application [Sun 95]
    - La synthèse : couplage du codage et de la synthèse [Kum 01]
- Donnée : chaque donnée possède sa propre largeur
  - Codage puis synthèse de l'architecture [Kum 98], [Ked 98]
    - Affectation d'opérations de largeur différente sur un même opérateur [Con 01]

Lien entre le codage et la synthèse

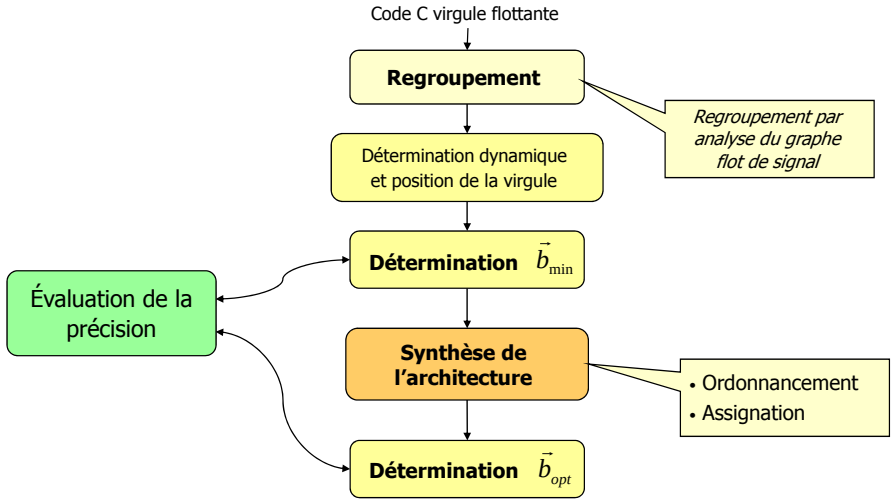


Méthode FPO (Université de Séoul)

- Différentes méthodes proposées
  - Regroupement au niveau donnée [Kum 98]
  - Regroupement au niveau bloc par analyse du graphe flot de signal de l'application [Sun 95]
    - Regroupement réalisé par l'utilisateur
    - Regroupement automatique
      - Données connectées par un délai ou un multiplexeur
      - Entrées et sortie d'un additionneur
  - Regroupement au niveau bloc : couplage des processus de codage en virgule fixe et de synthèse [Kum 01]
    - Regroupement des opérations réalisées sur un même opérateur

Méthode Fixed-Point Optimizer

- Regroupement au niveau bloc et couplage codage synthèse **[Kum 01]**



Synthèse de l'architecture

- Ordonnancement par liste
  - Une liste pour chaque opérateur de largeur différente
  - Traitement en priorité des opérations de largeur élevée
  - Assignation d'une opération à un opérateur de largeur plus élevée si celui-ci est libre
- Processus itératif pour l'ordonnancement et l'optimisation de la largeur des opérateurs
  - 1. Estimation du nombre minimal d'opérateurs nécessaires
  - 2. La largeur des opérateurs est fixée à la valeur maximale
  - 3. Ordonnancement de l'application
  - 4. Réduction de la largeur d'un des opérateurs tant que la contrainte de temps est satisfaite lors de l'ordonnancement

Optimisation de la spécification virgule fixe

1. Objectifs
2. Evaluation de la précision
3. Implantation matérielle
4. **Implantation logicielle**

Caractéristiques des DSP

- Largeur naturelle du processeur (b_{nat})
  - DSP: largeur fixe : 16 ou 24 bits
  - Cœur de DSP et ASIP : largeur paramétrable
- Largeur des données au sein de l'unité de traitement
  - Instructions classiques
    - Calcul d'une multiplication-addition sans perte de précision
    - Bits de garde au niveau de l'accumulateur
  - Instructions double précision
    - Augmentation de la précision : données stockées en mémoire en double précision
    - Augmentation des temps de calcul
  - Instructions SWP (Sub-Word Parallelism)
    - Utilisation du parallélisme au niveau des données
    - Réduction du temps d'exécution
- Loi de quantification

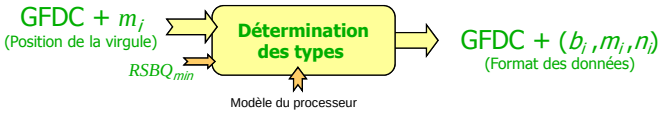
Architecture des DSP

- Instructions SWP Exemple C64x+

Instructions	b_{in-1}	b_{in-2}	b_{out}	k	T_{lat}	T_{tpt}
ADD4	8	8	8	4	1	1
ADD2	16	16	16	2	1	1
ADD	32	32	32	1	1	1
MPY4	8	8	16	4	4	1
MPY2	16	16	32	2	4	1
MPY2IR	16	32	32	2	4	1
MPYx	16	16	32	1	1	1
MPYxI	16	32	48	1	4	1
MPYxIR	16	32	32	1	4	1
MPY32	32	32	32	1	4	1

Table 2: TMS320C64x+ SWP instruction set

Détermination du type des données



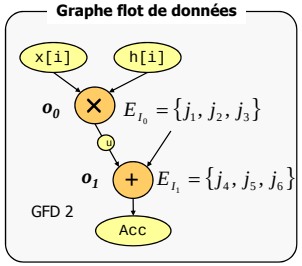
- Détermination de la largeur des données
  - Prise en compte des différents types manipulés par le DSP :
    - Instructions classiques
    - Instructions double précision
    - Instructions SWP
- Sélection de la suite d'instructions permettant de
  - Minimiser le temps d'exécution global
  - Satisfaire la contrainte de précision ($RSBQ_{min}$)

$$\min_{\vec{b} \in B} \left(T(\vec{b}) \right) \quad \text{tel que} \quad RSBQ(\vec{b}) \geq RSBQ_{\min}$$

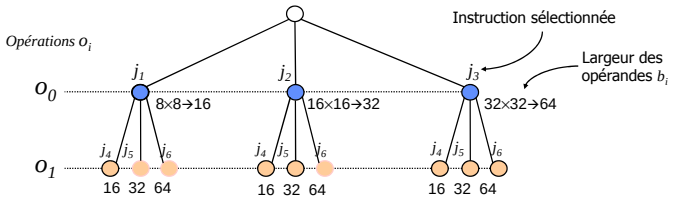
Exemple opération MAC

- Jeu d'instructions du processeur

Instruction j_k	Fonction γ_k	Temps d'execution t_k	Largeur des opérandes E/S b_{e1}	b_{e2}	b_s
j_1	MULT	0.25	8	8	16
j_2	MULT	0.5	16	16	32
J_3	MULT	1	32	32	64
j_4	ADD	0.25	16	16	16
j_5	ADD	0.5	32	32	32
j_6	ADD	1	64	64	64

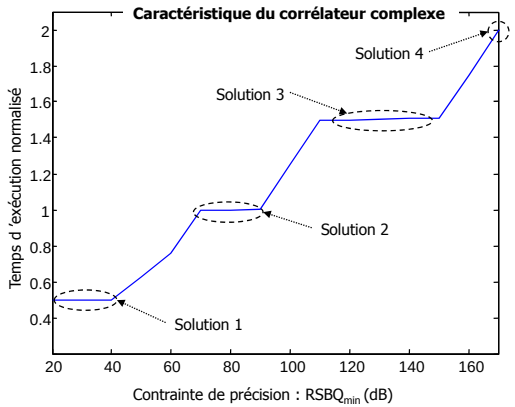


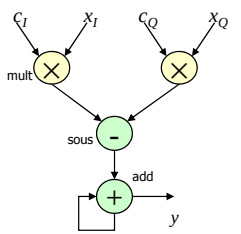
- Modélisation des solutions sous forme d'arbre



Compromis Coût / précision

- Caractéristiques corrélateur complexe sur DSP C64x

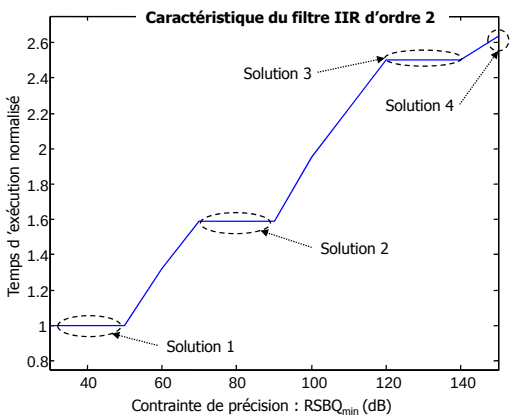


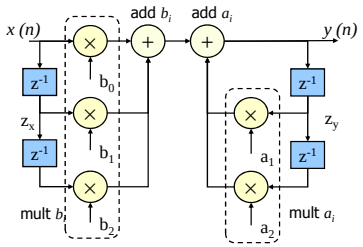


Solutions	Types des opérations et données (bits) mult	sous	add	y
1	8×8→16	16+16→16	16+16→16	8
2	16×16→32	32+32→32	32+32→32	16
3	32×16→32	32+32→32	32+32→32	32
4	32×16→64	64+64→64	64+64→64	32

Compromis Coût / précision

- Caractéristique filtre IIR sur DSP C54x

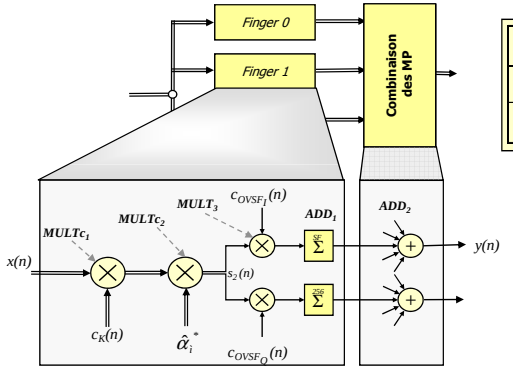




Solutions	Types des opérations et données (bits)						
	x, z_x	mult b_i	add b_i	mult a_i	add a_i	z_y	y
1	16	16×16→32	32+32→32	16×16→32	32+32→32	16	16
2	16	16×16→32	32+32→32	32×32→64	32+32→32	32	16
3	32	32×32→64	32+32→32	32×32→64	32+32→32	32	32
4	32	32×32→64	64+64→64	32×32→64	64+64→64	32	32

Optimisation du coût

- Spécification pour une contrainte de RSBQ de 12,5 dB



T_{exec} (cycles)	
Instructions classiques	Instructions SWP
3133	= ./ 2.64 ⟹ 1185

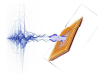
Types des opérations et données (bits)								
	$MULTc_1$		$MULTc_2$		$MULT_3$	ADD_1	ADD_2	
x	*mult*	*add*	*mult*	*add*	*mult*	*add*	*add*	y
8	8×8→16	16+16→16	16×16→32	16+16→16	16×16→32	16+16→16	16+16→16	16

Conclusion

Conclusion

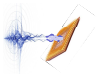
- Évaluation de la dynamique
  - Méthodes statistiques :
    - Estimation précise mais pas ne garantissant pas l'absence de débordement
  - Méthodes analytiques
    - Estimation pessimiste garantissant l'absence de débordement
- Évaluation de la précision
  - Méthodes basées sur la simulation :
    - Augmentent le temps d'exécution du processus d'optimisation
      - Mise en œuvre d'heuristiques pour réduire l'espace de recherche
  - Méthodes analytiques :
    - Toutes les structures ne sont pas prises en comptes
      - Définition de nouvelles méthodes

Conclusion



- Détermination de la contrainte de précision
  - Nécessité de bien maitriser l'application
- Implantation matérielle
  - Nécessité de couplage entre les processus de conversion en virgule fixe et de synthèse d'architecture
  - Mise en œuvre d'un algorithme efficace d'optimisation sous contrainte
- Implantation logicielle
  - Nécessité de prise en compte de l'architecture
  - Nécessité de couplage entre les processus de conversion en virgule fixe et de génération de code

Bibliographie (1)



[Aam00] T. Aamodt, Floating-point To Fixed-point Compilation and Embedded Architectural Support, Master Thesis, University of Toronto, January 2001

[Cac02] D. Cachera, T. Risset *Advaces in Bit Width Selection Methodology*. Proceedinngs of the IEEE Internationnal Conference on Application-Specific Systems, Architectures, and Processirs (ASAP 02), Jul 02.

[Coo01] M. Coors, H. Keding, O. Luthje and H. Meyr, *Integer Code Generation For the TI TMS320C62x*, ICASSP-01, May 2001, Sate Lake City, US.

[Con99] G. Constantinides, P. Cheung, W Luk, *Truncation noise in fixed-point SFG*, IEE Electronics Letters, 35(23), November 1999.

[Con01] G. Constantinides, P. Cheung, W Luk, Heuristic Datapath Allocation for Multiple Wordlength Systems, DATE 2001, Mars 2001.

[Hill06] T. Hill *AccelDSP Synthesis Tool Floating-Point to Fixed-Point Conversion of MATLAB Algorithms Targeting FPGAs* Xilinx Whitepapers WP239 (v1.0) April 19, 2006

[Kea96] R. Kearfott, *Interval Computations: Introduction, Uses, and Resources*, Euromath Bulletin, vol 2 (1), 1996, p95-112.

[Ked01] H. Keding , M. Coors, O. Luthje, H. Meyr, *Fast Bit-True Simulation*, Design Automation Conference 2001, (DAC-01), Jun 01, Las Vegas, US.

[Ked98a] H. Keding, M. Willems, M. Coors, and H. Meyr. FRIDGE: A Fixed-Point Design And Simulation Environment. Design, Automation and Test in Europe 1998 (DATE-98), Mar 98.

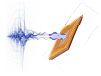
[Ked98b] H. Keding and F. Hurtgen and M. Willems and M. Coors, *Transformation of Floating-Point into Fixed-Point Algorithms by Interpolation Applying a Statistical Approach*, 9th International Conference on Signal Processing Applications and Technology, ICSPAT'98, 98.

[Kim98] S. Kim, K. Kum, S. Wonyong. *Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs,* IEEE Transactions on Circuits and Systems II, 45(11), Nov 98.

[Kum00] K. Kum, J.Y. Kang and W.Y. Sung, *AUTOSCALER for C: An optimizing floating-point to integer C program converter for fixed-point DSP* ,IEEE Transactions on Circuits and Systems II, pp 840-848, September 2000.

[Kum01] K. Kum, and W.Y. Sung, *Combined Word-length Optimization and High-level Synthesis of Digital Signal Processing Systems*,IEEE Transactions on Computer Aided Design of circuits and Systems II, pp 921-930, 20(8), August 2000.

Bibliographie (2)



[Liu71] B. Liu. Effect of Finite Word Length on the Accuracy of Digital Filters - A Review (Invited Paper). IEEE Transaction on Circuit Theory, 18(6), Nov 71.

[Mar01] E. Martin, C. Nouet, JM. Toureilles. Conception optimisée d'architectures en précision finie pour les applications de traitement du signal.Traitement du Signal 2001, Vol 18 (1), 2001.

[Men01] D. Menard, O. Sentieys, *Influence du modèle de l'architecture des DSPs virgule fixe sur la précision des calculs*, Dix huitième colloque GRETSI sur le traitement du signal et des images, Toulouse, Sep 01.

[Men02a] D. Menard and O. Sentieys. *Automatic Evaluation of the Accuracy of Fixed-point Algorithms*, IEEE/ACM Conference on Design, Automation and Test in Europe 2002 (DATE-02), Paris, Mar 02.

[Men02b] D. Menard and O. Sentieys. *A methodology for evaluating the precision of fixed-point systems*. International Conference on Acoustics, Speech and Signal Processing 2002 (ICASSP 2002), Orlando, May 02.

[Men02c] D. Menard, T. Saidi, D. Chillet, O. Sentieys. *Implantation d'algorithmes spécifiés en virgule flottante dans les DSP virgule fixe*. 8ème Symposium en Architectures nouvelles de machines (SYMPA), Hammamet, Tunisie, Apr 02.

[Men02d] D. Menard, P. Quemerais, O. Sentieys. *Influence of fixed-point DSP architecture on computation accuracy*. XI European Signal Processing Conference (EUSIPCO 2002), Toulouse, Sep 02.

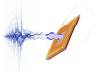
[Men02e] D. Menard, D. Chillet, F.Charot, O. Sentieys. *Automatic Floating-point to Fixed-point Conversion for DSP Code Generation*. ACM International Conference on Compilers, Architectures and Synthesis for Embedded Systems 2002 (CASES 2002), Grenoble, Oct 02.

[Men02f] D. Menard, T. Saidi, D. Chillet, O. Sentieys. *Implantation d'algorithmes spécifiés en virgule flottante dans les DSP virgule fixe*. Sélectionné pour numéro spécial de TSI Architectures des systèmes embarqués.

[Men03a] D. Menard, M. Guitton, S. Pillement, O. Sentieys. Design and Implementation of WCDMA Platforms Challenges and Trade-offs. Accepted for the International Signal Processing Conference, Dallas, April 03.

[Par87] T.W. Parks and C.S. Burrus, *Digital Filter Design*, 1987, Jhon Willey and Sons Inc

Bibliographie (3)



[Ozer08] Özer, E., Nisbet, A. P., and Gregg, D. 2008. A stochastic bitwidth estimation technique for compact and low-power custom processors. *Trans. on Embedded Computing Sys.* 7, 3 (Apr. 2008), 1-30. DOI= http://doi.acm.org/10.1145/1347375.1347387

[Sun95] W. Sung, K. Kum. *Simulation-Based Word-Length Optimization Method for Fixed-Point Digital Signal Processing Systems.* IEEE Transactions on Signal Processing, 43(12), Dec. 1995.

[Tou99] J. Toureilles. *Conception d'architectures pour le traitement du signal en précision finie*. PhD thesis, Université de Rennes I, Jan 99.

[Wil97] M. Willems and V. Bursgens and H. Meyr, *FRIDGE: Floating-Point Programming of Fixed-Point Digital Signal Processors* Internartionnal Conférence On Signal Processing Applications and Technology, (ICSPAT'97), 1997.