EII2

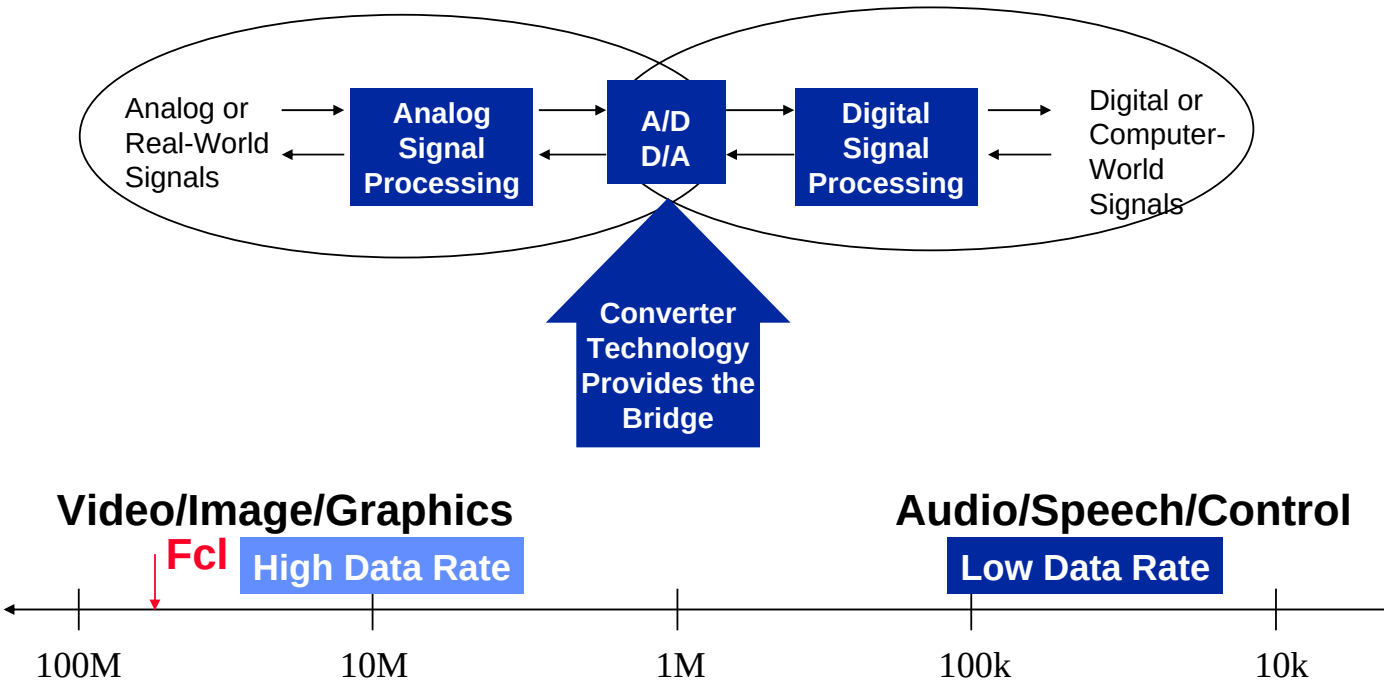
Traitement Numérique du Signal

Daniel MENARD
Olivier SENTIEYS
sentieys@enssat.fr

http://r2d2.enssat.fr/enseignements/Tns/Tns.php

Notes :

Traitement Numérique du Signal
(Digital Signal Processing)

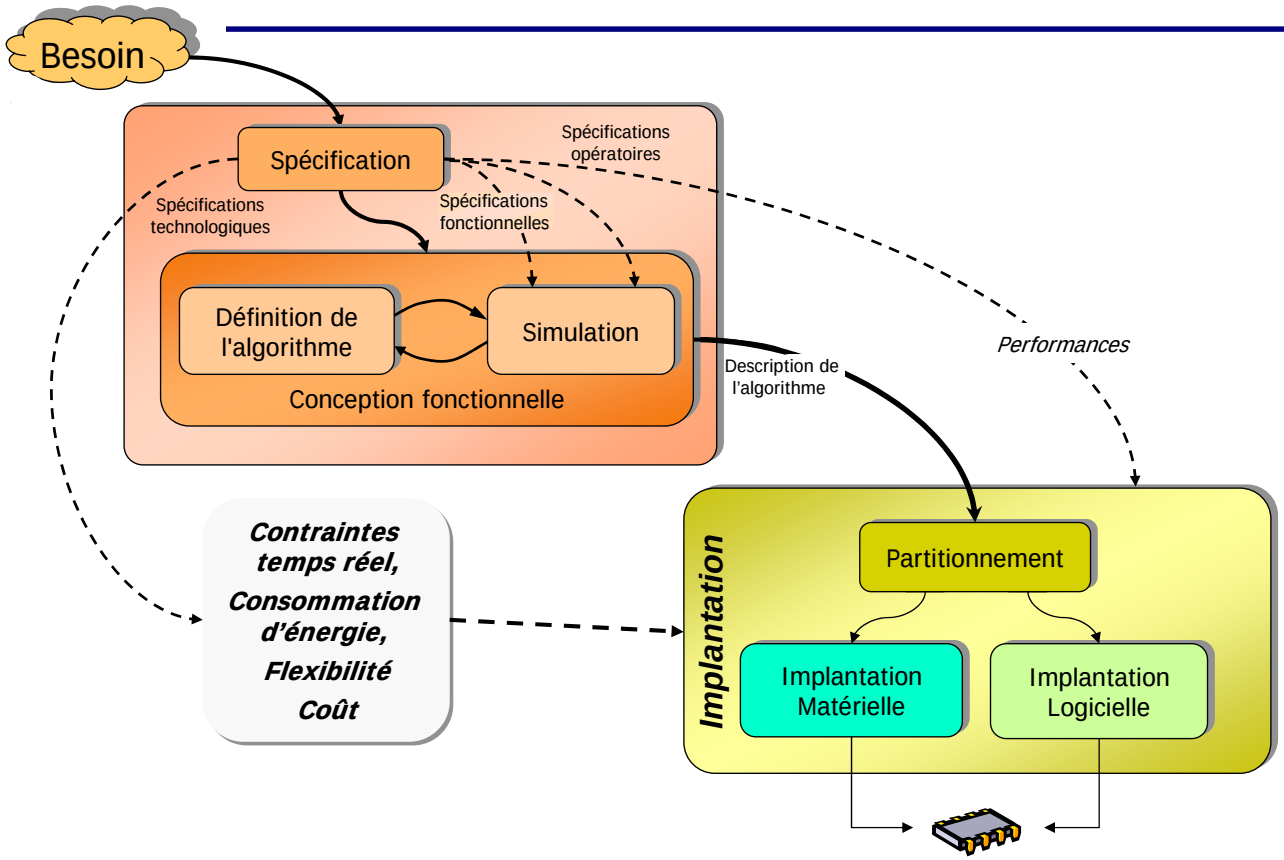


[ICE97] 2

Notes :

Digital signal processing (DSP) is a segment of the IC industry where advanced digital and analog technologies merge. The typical function of the DSP device is to perform real-time processing of a digitized analog signal, changing that signal using arithmetic algorithms, and then passing the signal on. The process is very math intensive and quite complicated. In fact, finding competent DSP designers and programmers is often a challenge for many DSP manufacturers.

Cycle de développement



Notes :

Contenu du cours de TNS et PTS

- Analyse et conception de systèmes de TNS
  - Analyse des filtres numériques
  - Transformées en TNS
  - Synthèse des filtres numériques RII
  - Synthèse des filtres numériques RIF
  - Analyse spectrale
  - Systèmes multi-cadences

- Implantation de systèmes de TNS
  - Arithmétique virgule fixe (codage - évaluation de la précision)
  - Implantation logicielle : processeurs de traitement du signal

    Implantation matérielle : module conception des circuits intégrés (S4)

Notes :

Plan du cours

I. Introduction

1. Introduction, problématique, caractéristiques, solutions architecturales
2. Caractéristiques des algorithmes
3. Applications typiques de TNS
4. Chaîne de traitement et problèmes temps réel

II. Processeurs de traitement du signal

I. Introduction
2. Description des différentes unités
3. Panorama des processeurs
4. Outils de développement

Notes :

Plan du cours (suite)

III. Arithmétique virgule fixe

1. Présentation de l'arithmétique virgule fixe
2. Processus de codage en virgule fixe
3. Exemple d'un filtre IIR cascadé

IV. Analyse des filtres numériques

1. Spécification, classification, représentation
2. Analyse fréquentielle
3. Structures des filtres RII et RIF

V. Transformées en TNS

1. TFD, convolution linéaire
2. TFR : Transformée de Fourier Rapide

Notes :

Plan du cours (suite)

VI. Quantification - évaluation de la précision

1. Quantification
2. Effets de la quantification en TNS

VII. Synthèse des filtres numériques RII

1. Invariance Impulsionnelle
2. Transformation Bilinéaire

VIII. Synthèse des filtres numériques RIF

1. Introduction
2. Filtres à Phase Linéaire
3. Méthode du Fenêtrage
4. Échantillonnage en Fréquence

Notes :

Notes

Notes :

Plan du cours (fin)

IX. Analyse spectrale

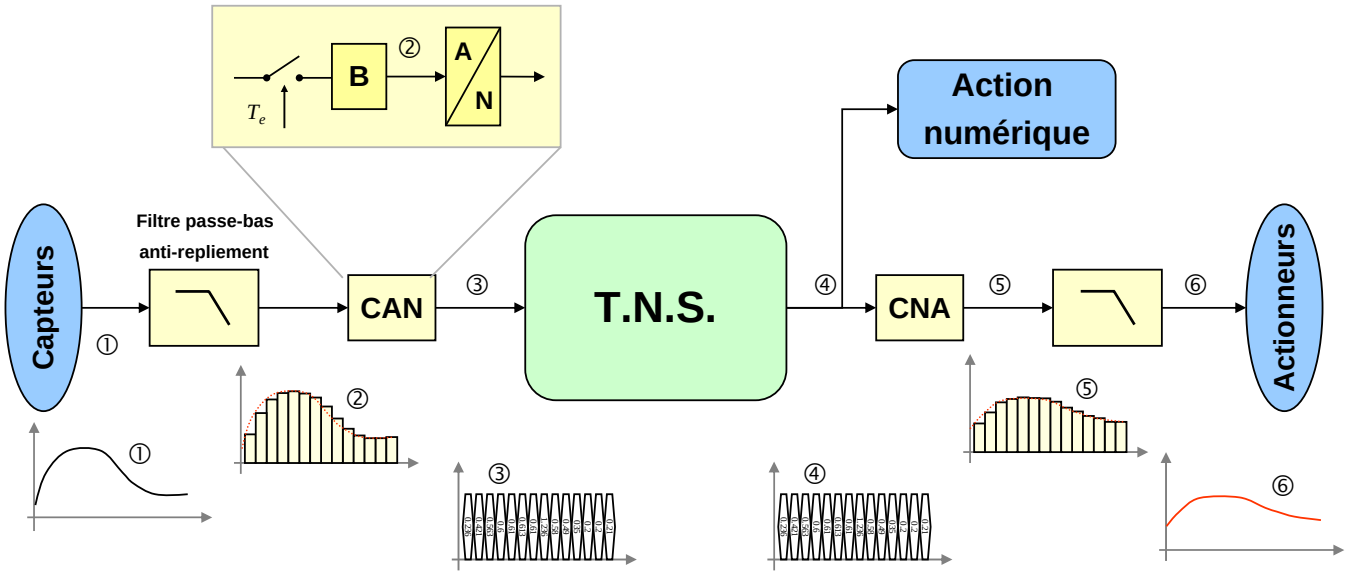
1. Effets de la troncature
2. Caractéristiques des fenêtres
3. Influence sur l'analyse

X. Systèmes multi-cadences

1. Définition
2. Décimation
3. Interpolation

Notes :

Chaîne de traitement



Notes :

I.1 Traitement Numérique du Signal

- Avantages
  - Pas de dérive : température, vieillissement, valeur des composants
  - Précision : garantie par le nombre de bits
  - Souplesse : plusieurs tâches simultanées possibles
  - Prédiction : simulation sur ordinateur
  - Prototypes : changements par modifications du logiciel
  - Performances : pas de distorsion de phase, filtrage adaptatif, etc.
  - Intégration : progrès des systèmes VLSI et DSP
- Inconvénients
  - Coût : élevé pour des réalisations simples
  - Vitesse : bande passante large = vitesse de calcul élevé
  - Complexité : réalisation à la fois matérielle et logicielle

Notes :

Quelles applications ?

- A la Maison
  - Télévision à la demande, Télévision Satellite, Jeu Vidéo et Réalité Virtuelle, Electroménager, Réseaux, ...
  - DVD, HDTV, CD, DAB, DVB

- Au Bureau
  - Vidéoconférence, Fax, Modems, Pagers, etc.
  - Réseaux rapides, Sans-fil (WLAN, WiFi, etc.)
  - ATM, ISDN, ADSL
- Sur la route
  - Téléphone cellulaires, Commande vocale, Radar et Sonar, GPS et traceur de route, Fax/Modems sans-fil, Véhicules intelligents, etc.

Systèmes de l'âge de l'information
= fusion entre
Calculateur- Télécommunications- Consommateur

Notes :

Domaines d'application

- **Communication homme-machine**, synthèse, transformation texte-parole et inverse, reconnaissance de parole, identification et vérification du locuteur
- **Télécommunications**, codage et restauration de la parole, courrier vocal, télécopie, audionumérique (CD, DAB), TV numérique, compression et transmission d'images, cryptage et protection, transmission de données, télé informatique, annulation d'écho, codage à débit réduit, télé et visioconférence, téléphonie cellulaire, ...
- **Défense,** systèmes d'armes, surveillance, guidage, navigation
- **Biophysique**, génie biomédical, EEG, ECG, radiographie, tomographie, scintigraphie, gammagraphie, échographie, aide aux handicapés, ...
- **Acoustique**, aérienne, sous-marine, sonar, ultrasons, nuisances
- **Géophysique**, sismique, de surface, océanographique, télédétection
- **Electromagnétisme,** radar, radionavigation, optique, astrophysique
- **Automobile**, injection électronique, ABS, positionnement global, commande d'assiette adaptative
- **Musique**, numérique, MIDI, échantillonneurs (sampleurs), synthétiseurs, mélangeurs, réverbération et écho, effets spéciaux, filtrage, enregistrement (DAT)
- **Instrumentation**, capteurs, métrologie, analyse spectrale, génération de signaux, analyses de transitoires, DPLL
- **Graphisme et imagerie**, rotation 3D, vision, reconnaissance de formes, restauration d'images, stations de travail, animation, cartographie

Notes :

Glossaire du TNS

- Saisie, acquisition, conversion (A/N, N/A), codage
- Filtrage, FIR, IIR, convolution rapide, filtres spéciaux
- Représentation, modélisation, analyse spectrale, transformées
- Compression, approximation, extrapolation, codage de source, réduction de débit
- Modulation, codage de canal, protection contre les erreurs, cryptage, garantique
- Détection, réception optimale, démodulation, décodage, correction
- Estimation, paramétrique, estimation d'onde ou d'état, filtrage, prédiction, lissage
- Analyse de système, modèles de canaux, milieux de propagation
- Amélioration, réduction de bruit, annulation d'écho, compensation, égalisation
- Déconvolution, imagerie, résolution, détection de source, restauration
- Classification, reconnaissance, signatures
- Apprentissage, estimation séquentielle, adaptation, poursuite
- Analyse temps fréquence, non stationnarité, estimation de délais, de phase
- Traitement multi-fréquences, décimation, interpolation, filtrages en sous bandes
- Arithmétique bit fini, quantification, dépassement, virgule fixe, flottante, bruits de calcul, sensibilité des coefficients.
- Architecture des systèmes, DSP, ASIC, mémoire

Notes :

Caractéristiques algorithmiques

- Traitement temps réel

  Temps d'exécution de l'algorithme T_{ex} guidé par acquisition I/O

  Période d'échantillonnage T_e

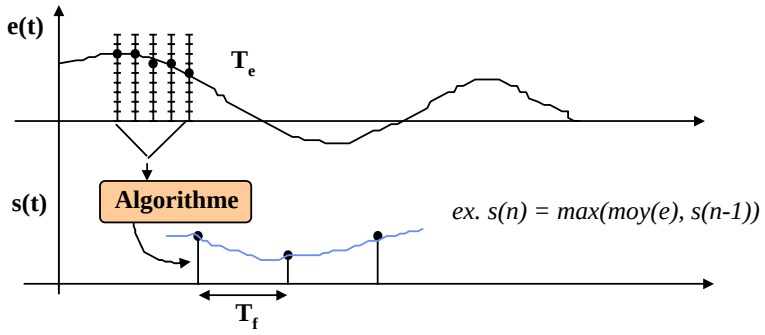
  Période des sorties T_f (frame period) > T_{ex}

  Ni plus vite ... ni plus lentement (not faster ... not slower)

- Signaux numériques : quantité importante de données

  Données scalaires, vectorielles, matricielles, multidimensionnelles

  Opérations I/O intensives par DMA



Notes :

Notes

Notes :

Caractéristiques algorithmiques

- Charge de calcul importante
  - multiplications-accumulations (convolution)
  - multiplications-additions (FFT, DCT, adaptation,...)
- Virgule Fixe ou Flottante
  - problèmes liés à la quantification !!!
- Calculs d'adressage complexes
  - accès linéaire, indexé
  - bit-reverse ou similaire (FFT)
  - vieillissement des données ...
- Boucles de traitement courtes

Notes :

Fonctions typiques de TS

- Filtrage, convolution

  y = y + x.h : MAC (multiplication-accumulation)

- Adaptation

  $y_n = y_{n-1} + x.h$: MAD (multiplication-addition)

- FFT, multiplication complexe

  xr = xr.wr - xi.wi

  xi = xr.wi + xi.wr

- Viterbi

  a1 = x1 + x2; a2 = y1 + y2;

  y = (a1>a2) ? a1 : a2 : ACS

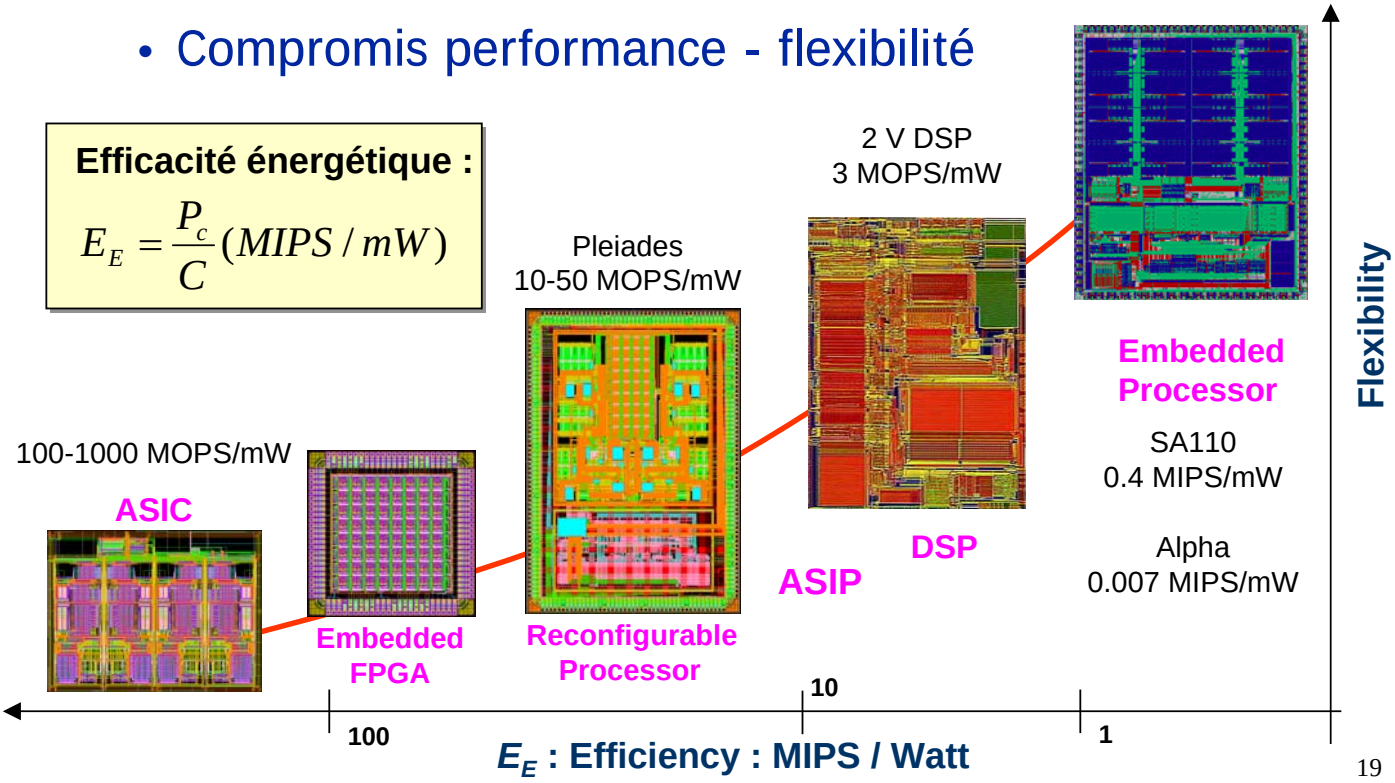
Notes :

Solutions architecturales

- Compromis performance - flexibilité

Efficacité énergétique :

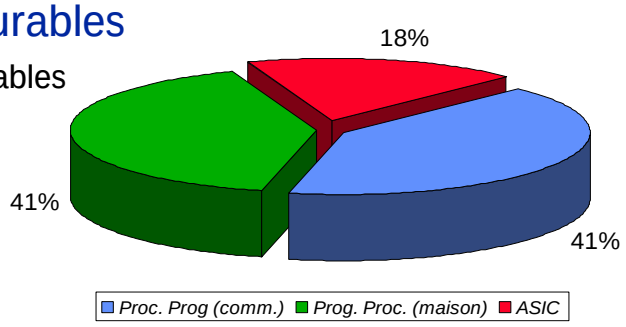
$$E_E = \frac{P_c}{C}(MIPS/mW)$$



Notes :

Solutions architecturales

- Processeurs programmables du commerce (ISP)
  - Processeurs généraux ***RISC, VLIW***
  - Micro-contrôleurs
  - Processeurs de Traitement du Signal (DSP)
  - Processeurs Multimédia
- Processeurs programmables “maison” (ASIP)
  - De type DSP ou µCtrl
- Processeurs et logique reconfigurables
  - FPGA enfouis, Processeurs reconfigurables
- Coprocesseurs ASIC



Notes :

Caractéristiques des DSP

- Applications embarquées et « grand public »
  - Très faible coût

    Le coût est proportionnel à la surface du circuit
  - Faible consommation

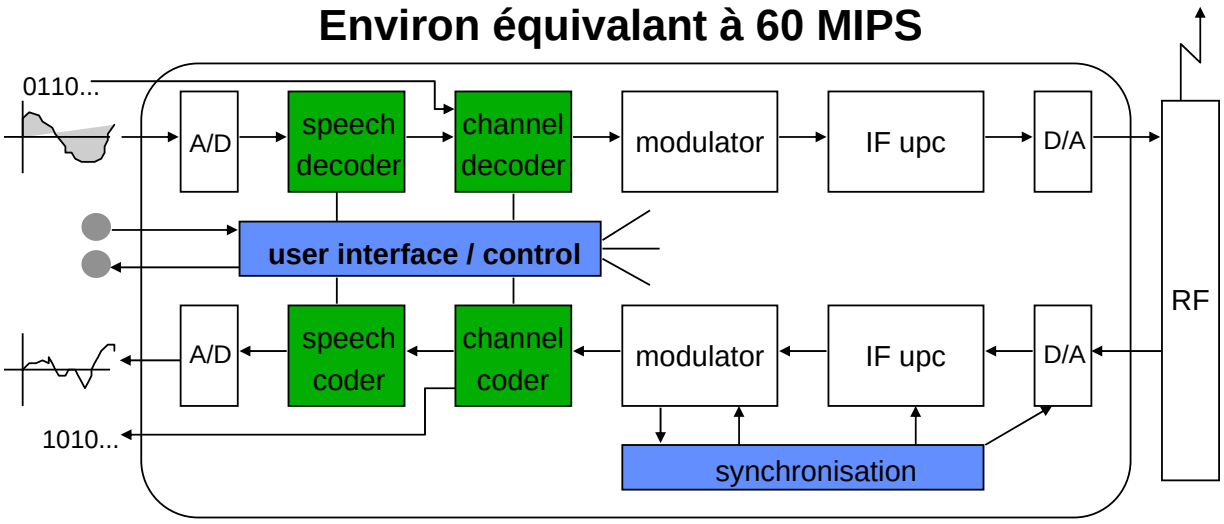
    Pmoy = α . C . V_{dd}^{2} . F
    - α: facteur d'activité défini comme le nombre moyen de transitions (0 **à** 1) pendant une période d'horloge.

    Une partie importante de la consommation est liée à la mémoire
    - Nécessité de minimiser la largeur des données et des instructions stockées en mémoire
  - Temps réel

    Implémentation efficace des applications de TS
  - Sûreté de fonctionnement

Notes :

Exemple : GSM

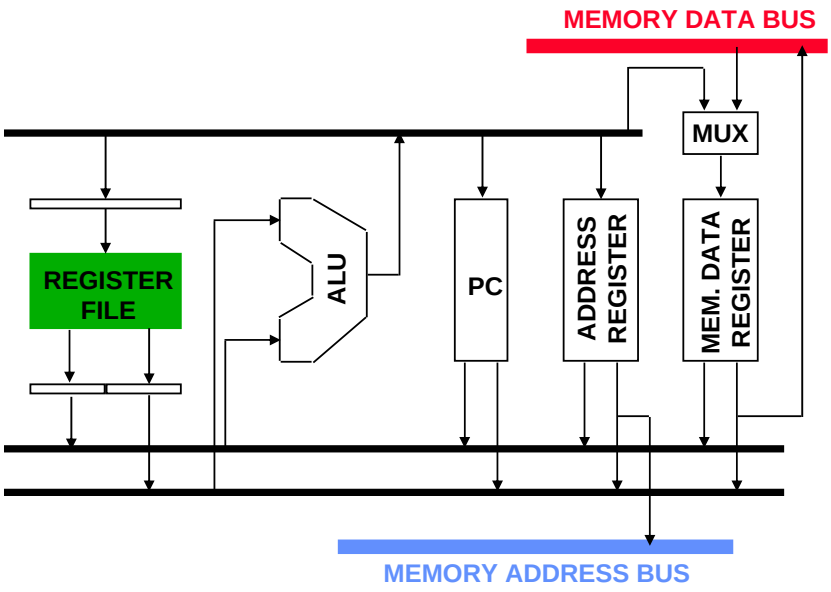


Coprocessor, Hardware
DSP, ASIP, ASP
µController

Notes :

Architecture Von Neumann

- Les processeurs RISC



Notes :

A typical data path of a RISC architecture is shown. It includes a register file with source and destination latches, an ALU (arithmetic and logic unit) and a program counter (PC). A RISC processor may also have additional registers for data and instruction addressing or other control realted functions.

Most RISC designs use the same ALU to compute both algebraic operations and memory addresses for load and store operations. The justification for such a design is that because during load and store operations the ALU is not busy, such an implementation does not cause any performance penalty.

[Bhaskaran 95]

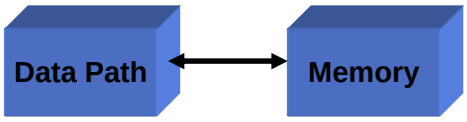
Notes

Notes :

FIR sur machine Von Neumann

- Problèmes :
  - Bande passante avec la mémoire et gestion des pointeurs d'adresse
  - Code pour le contrôle
  - Multiplication lente

```
loop:
   mov *r0,x0      } Lecture des opérandes sources
   mov *r1,x1      }
   mpy x0,x1,a     } Opération MAC
   add a,b         }
   mov x1,*r2      } Vieillissement de l'échantillon
   inc r0          }
   inc r1          } Gestion des pointeurs d'adresse
   inc r2          }
   dec ctr         }
   tst ctr         } Gestion de la boucle
   jnz loop        }
```

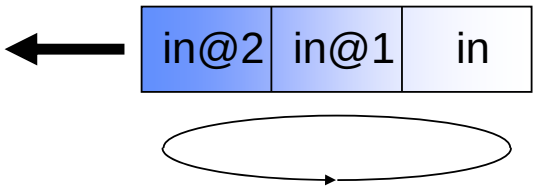


Exécution en 15 à 20 cycles

Notes :

Architecture Harvard

- Bus et mémoires données/instructions séparées
- Unité de traitement de type mpy-acc
- Registres distribués (≠ RISC register file)
  - Chaque module (ALU) possède ses propres registres locaux
- Génération adresses efficaces (AGUs)
  - Modes d'adressage spéciaux : auto incr-decr, circular buffering (delay line) ...



Notes :

Because many DSP algorithms involve performing repetitive computations, most DSP processors provide special support for efficient looping. Often, a special loop or repeat instruction is provide which allows the programmer to implement a for-next loop without expending any instruction cycles for updating and testing the loop counter or branching back to the top of the loop.

Finally, to allow low-cost, high-performance input and output, most DSP processors incorporate One Or more serial or parallel I/O interfaces, and specialized I/O handling mechanisms such as low-overhead interrupts and direct memory access (DMA) to allow data transfers to proceed with little or no intervention from the rest of the processor.

In some cases, system designers may prefer to use a general-purpose processor over a DSP processor. Although general-purpose processor architectures often require several instructions to perform operations done with just one DSP processor instruction, general-purpose processors sometimes compensate by running at extremely fast clock speeds. If the designer needs to perform non-DSP processing, then using a using a general-purpose processor for both DSP and non-DSP processing could reduce the system parts count and lower costs versus using a separate DSP processor and general-purpose microprocessor. Furthermore, some popular general-purpose processors feature a tremendous selection of application development tools.

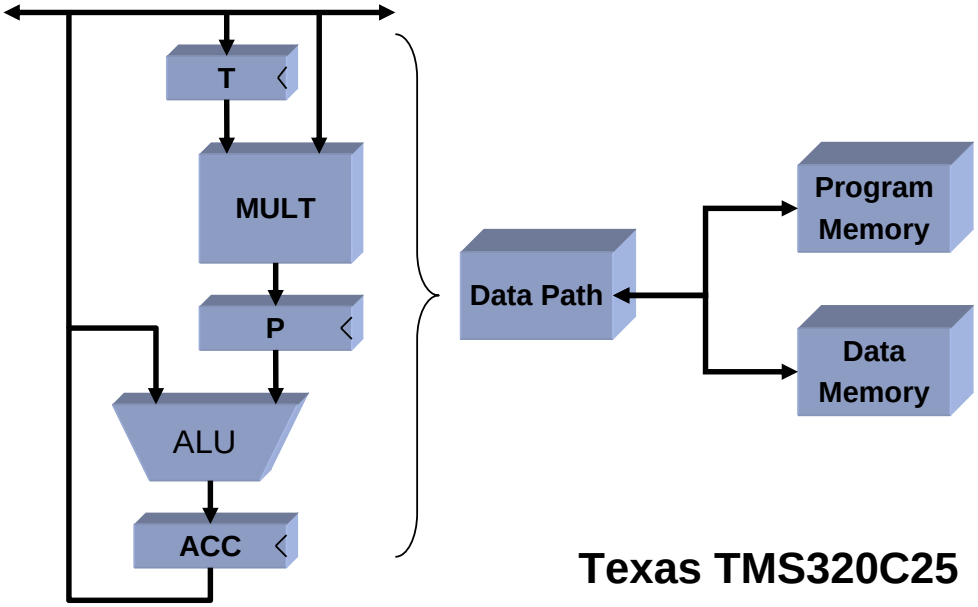
On the other hand, because general-purpose processor architectures lack eatures that simplify DSP programming, software development is sometimes more tedious than on DSP processors and can result in awkward code that's difficult to maintain. Moreover, if general-purpose processors are used only for signal processing, they are rarely cost-effective compared to DSP chips designed specifically for the task. Thus, at least in the short run, we believe that system designers will continue to use traditional DSP processors for the majority of DSP intensive applications.

FIR sur DSP conventionnel

- Jeu d'instructions complexe

```
loop:
   acc=acc+P
// P=T*PM(*r0+)
// T=DM(*r1+)
   jnz loop
```

Exécution en 1 cycle



Texas TMS320C25

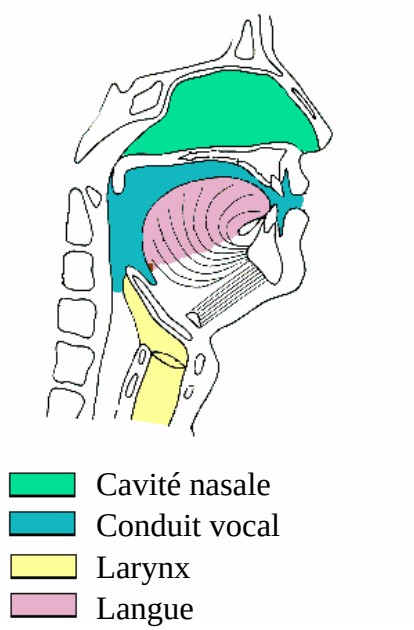
Notes :

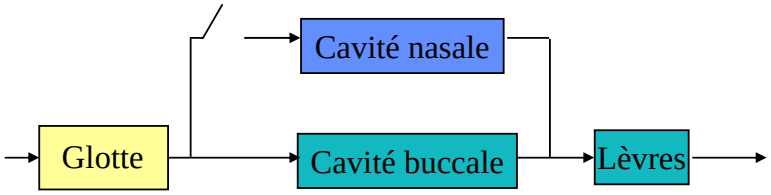
- Signaux médicaux
  - EEG, ECG, Images IRM, Images scanner, ...
- Signaux sismiques
- Données
  - Statistiques, Bourse, ...
- Signal de parole
- Sons
- Images
- ...

Notes :

Signal de Parole

- Processus de phonation
  - Génération d'une énergie ventilatoire (poumons+trachée)
  - Vibration des cordes vocales
  - Réalisation d'un dispositif articulatoire (conduit vocal)



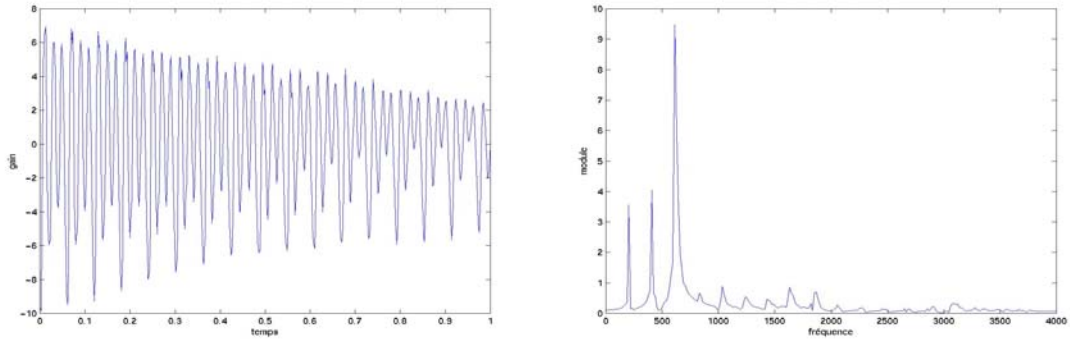


Modèle source filtre

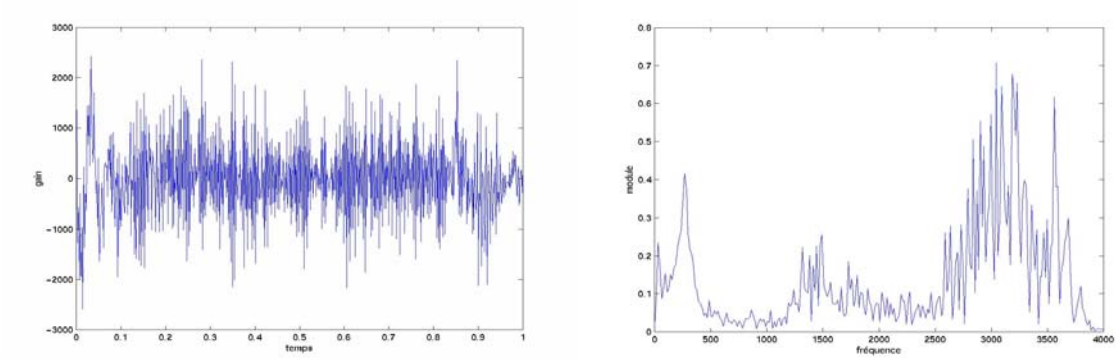
Notes :

Le signal de parole est réalisé au moyen d'un appareil phonatoire non initialement prévu pour cela. Il faut tout d'abord une énergie ventilatoire pour être l'initiateur du mouvement d'air qui crée les ondes acoustiques et qui crée par la même occasion le mouvement oscillatoire des cordes vocales ou au contraire qui les écarte pour créer du bruit; c'est le rôle de la **soufflerie**. Elle est composée des poumons qui sont le générateur d'air et du conduit trachéo-bronchique. Ce souffle passe le **larynx** (où siégent les cordes vocales) pour former l'onde glottique. Un dispositif écarte ou rapproche les cordes vocales selon qu'elles doivent vibrer ou non pour obtenir le son. Si les cordes vocales sont rapprochées, elles vibrent et donnent un son dit voisé (80% de la phonation), sinon on obtient un son dit non voisé. Ensuite, le pharynx, la langue et les parois se modifient et forment un filtre possédant une certaine fonction de transfert qui modifie l'onde glottique par convolution, ce qui donne, après rayonnement au niveau des lèvres, le signal de parole .

Signal de parole



Un son voisé et son spectre (son “ eu ”)



Un son non voisé et son spectre (son “ ch ”)

Notes :

On peut distinguer la nature du son (voisé ou non) par son allure (temporelle et fréquentielle). Un son voisé possède un signal pseudo périodique : son spectre contient des harmoniques (énergie présente dans les différentes fréquences). Le premier harmonique, qui reflète la fréquence de vibration des cordes vocales est appelé la **fréquence fondamentale (F0)** ou en anglais le **pitch** (de 80 à 100 Hz chez l’homme, de 175 à 300 Hz pour la femme et de 200 à 600 Hz chez l’enfant). L’évolution de la fréquence fondamentale détermine la **mélodie** de la parole [MAL,01].

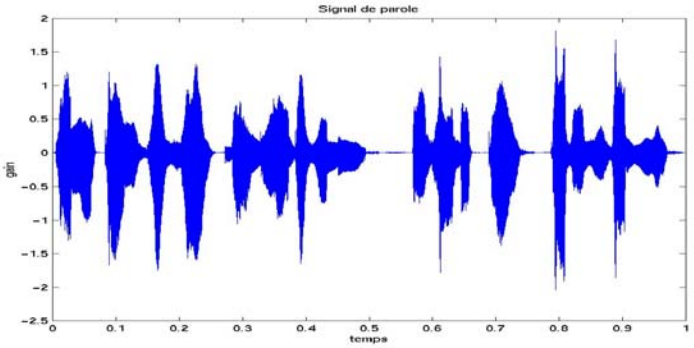
Pour un son non voisé, on obtient un signal qui ressemble à un bruit possédant beaucoup de hautes fréquences : le fait de filtrer le signal par un passe-bas pour respecter le théorème de Shannon fait qu’il devient plus difficile de distinguer des sons chuintants comme “ ch ”et “ sh ” car leurs différences se situent justement dans les hautes fréquences.

Les spectres des figures 2.3 et 2.4 sont obtenus par le passage de l’onde glottique dans la fonction de transfert du conduit vocal. On définit un **formant** comme le maximum de la fonction de transfert du conduit vocal, mais les maxima de la fonction de transfert sont excités par les composantes spectrales du signal glottique. Donc, les maxima du spectre de la figure 2.3 correspondent à peu près aux formants.

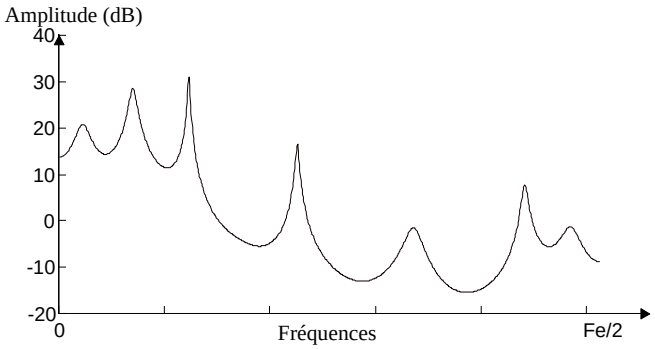
Pour cataloguer et référencer les sons afin de les étudier, on utilise une base de donnée qui possède tous les sons “ simples ” : les éléments de cette base sont appelés **phonèmes**. Un phonème se définit rigoureusement comme étant la plus petite unité susceptible de changer un mot en un autre : par exemple, [k] de “ car ” et [p] de “ par ” sont des phonèmes. Enfin, la mélodie de la parole liée à la durée et l’intensité des syllabes sont les éléments qui caractérisent la **prosodie**, qui sert entre autre à la compréhension syntaxique de la phrase (augmenter de façon significative la valeur du pitch à la fin d’une phrase interrogative).

Signal de parole

- Signal temporel
  - Aspects statistiques
  - Variabilité intra-individuelle
  - Variabilité inter-individuelle
  - Masquage temporel
  - Prosodie



- Signal fréquentiel
  - Structure formantique
    - Fondamental (pitch)
    - Harmoniques
  - Masquage fréquentiel

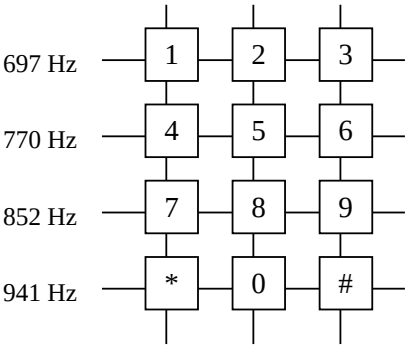


Notes :

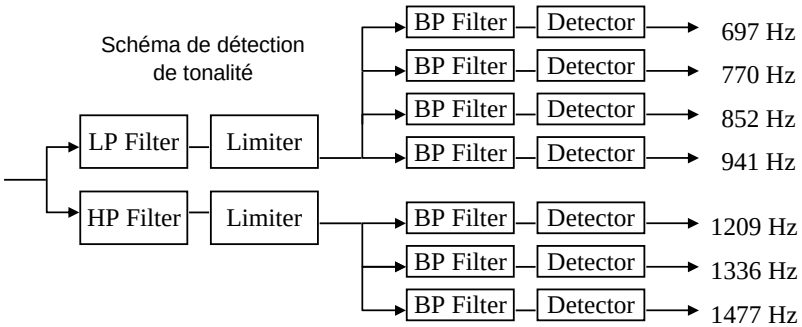
Notes

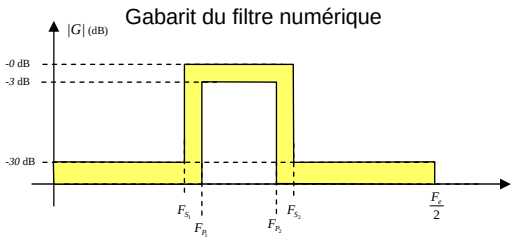
Notes :

- Télécommunications : détection de tonalité



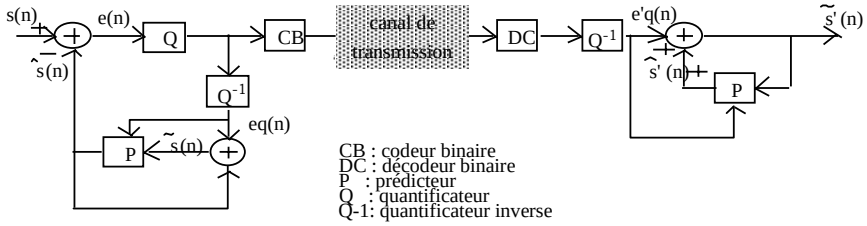
Fréquences de tonalité du clavier numérique





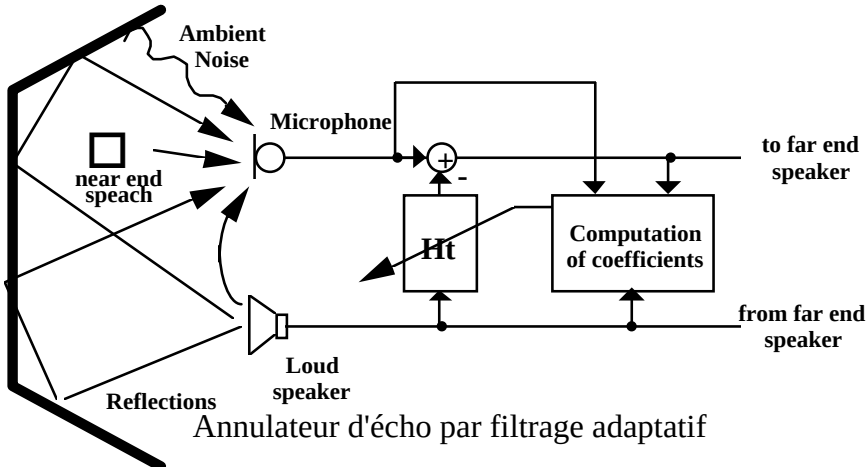
Notes :

- Voie MIC (PCM)
  - Parole échantillonnée à 8 kHz en bande limitée à 300-3400Hz sur 8 bits par une loi logarithmique (Rec. G711 du CCITT)
  - Débit normalisé de 64 kbit/s par voie numérique (MIC)
- Codage de la parole
  - Le codage permet : soit d'augmenter le nombre de signaux par voie (multiplexage temporel), soit d'élargir la bande codée (7kHz pour audio et visioconférence)
  - ADPCM : 32 kbit/s sans dégradation audible



Notes :

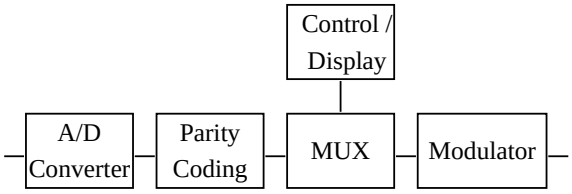
- Annulation d'écho
  - Réseau téléphonique utilisant les satellites géostationnaires (540ms)
  - Téléphone main libre en voiture (écho + bruit)
  - Téléconférence
    - Réponse impulsionnelle de la salle
    - Effets : écho, Larsen, réverbération
    - Problème de déconvolution



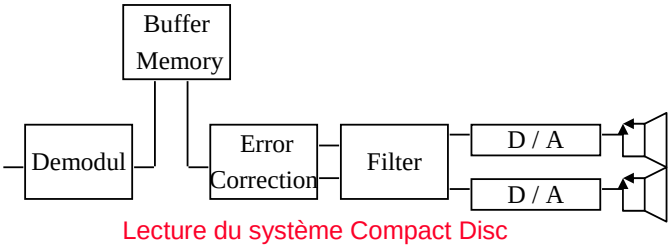
Annulateur d'écho par filtrage adaptatif

Notes :

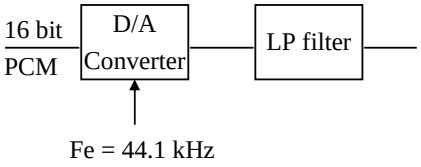
- Compact Disc Audio
  - Echantillonnage à 44,1 kHz sur 16 bits des deux voies : 1,41 Mbit/s
  - Information + correction d'erreurs, contrôle et affichage : 4,32 Mbit/s
  - 90 dB de rapport Signal à Bruit et de séparation stéréo (contre 60 et 30 dB)



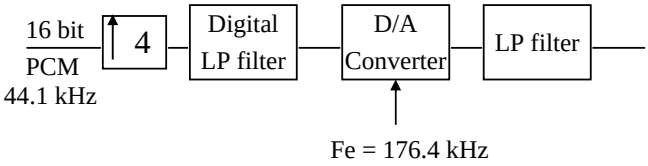
Encodeur du système Compact Disc



Lecture du système Compact Disc



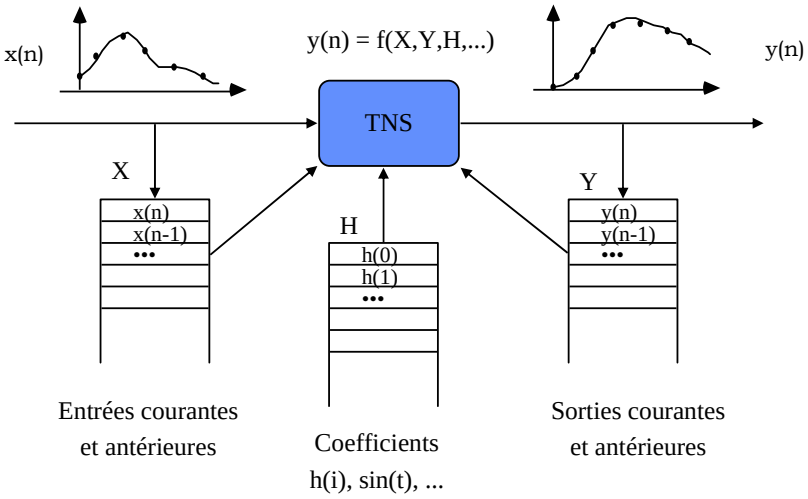
Conversion N/A classique



En pratique : Suréchantillonnage

Notes :

- Traitement par ligne
  - Flot de données ininterrompu
  - En entrée : flot d'échantillons sur N bits à N.Fe bit/s par entrée
  - En sortie : flot d'échantillons sur N' bits à N'.F'e bit/s par sortie

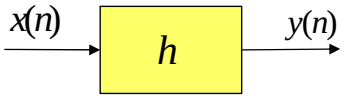


Notes :

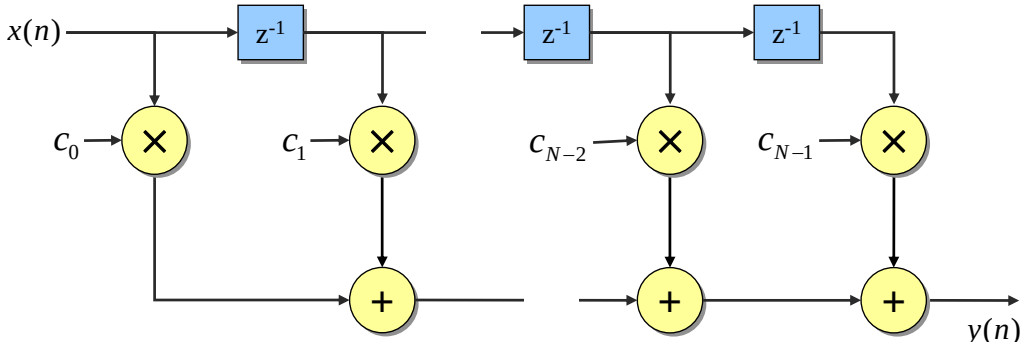
Exemple filtre FIR

- Équation aux différences

$$y(n) = \sum_{i=0}^{N-1} c_i . x(n-i) = x(n) * h(n) \quad \textbf{avec} \quad \begin{cases} h(i) = c_i & \forall i \in [0, N-1] \\ h(i) = 0 & \textbf{ailleurs} \end{cases}$$



- Graphe Flot de Signal



Notes :

Exemple filtre FIR

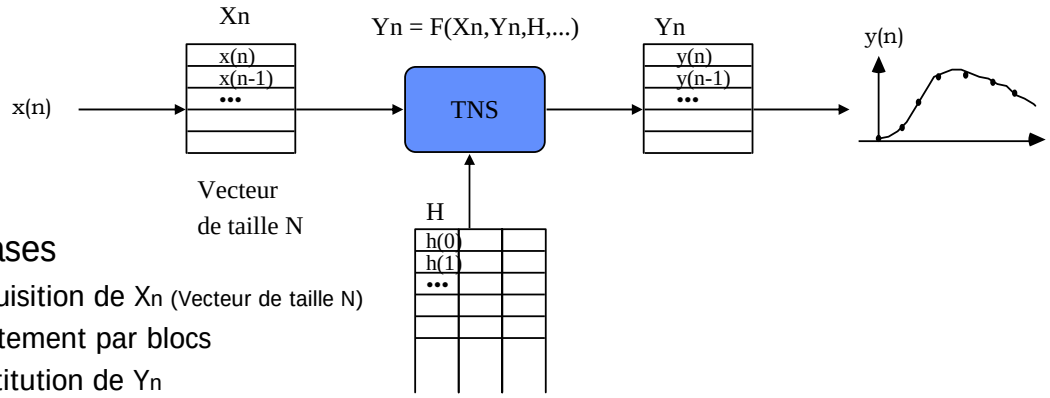
- Traitement par ligne : code C ...

Notes :

Notes

Notes :

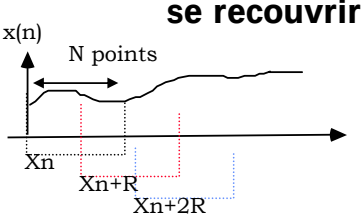
- Traitement par blocs



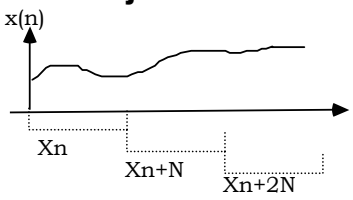
– 3 phases
  Acquisition de X_n (Vecteur de taille N)
  Traitement par blocs
  Restitution de Y_n

– Les phases d'acquisition d'un traitement au suivant peuvent soit

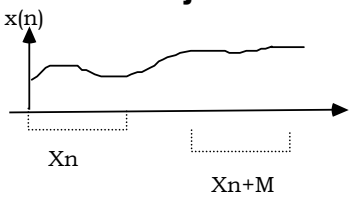
se recouvrir



être jointives



être disjointes



Notes :

Exemple filtre FIR

- Traitement par blocs : code C ...

Notes :

Cadence des calculs

- N_v : nombre de voies à traiter en parallèle
- N_{op} : quantité d'opérations élémentaires nécessaires au TNS
- T_e : périodicité du calcul

- Puissance de calcul de la machine :

$$P_{calcul} = \frac{N_V . N_{op}}{T_e}$$ (en MIPS, MOPS ou MFlOps)

$$P_{calcul} = 2.B.N_V.N_{op}$$ B est la bande du signal

Notes :

Exemple : Convolution

Traitement ligne

$$y(n) = \sum_{k=0}^{N-1} h(k).x(n-k)$$

- Complexité pour un point y(n) : ??
- Architecture réalisant une multiplication accumulation en 1 cycle

Traitement blocs avec recouvrement de N-R échantillons

$$Y_n = H \cdot X_n$$

$$\begin{bmatrix} y(n) \\ y(n-1) \\ \vdots \\ y(n-N+1) \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & \cdots & h(N-1) \\ h(N-1) & h(0) & h(1) & \vdots \\ \vdots & & & h(1) \\ h(1) & \cdots & h(N-1) & h(0) \end{bmatrix} \cdot \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix}$$

- Complexité pour un vecteur Yn :
- Même Architecture

Notes :

I. Processeurs de traitement du signal

1. Introduction

2. Description des différentes unités

2.1. Unité de traitement

2.2. Unité de mémorisation

2.3. Unité de contrôle

2.4. Unité de communication

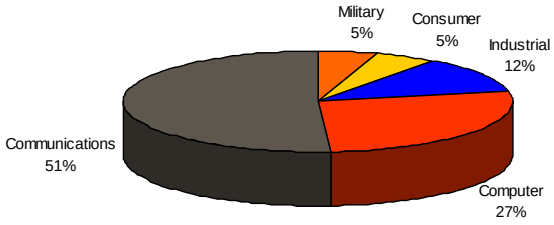
3. Panorama des processeurs

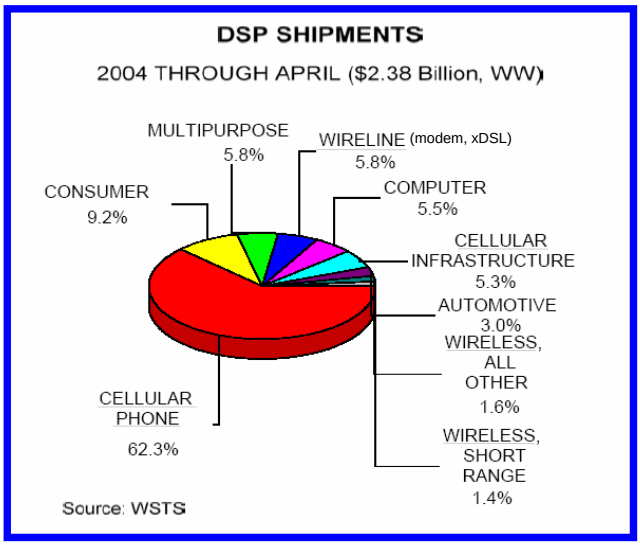
4. Outils de développement

Notes :

Le marché des processeurs DSP

- Les communications dominent : 76,4% des revenus en 2004
- Systèmes *grand public* : DVD, TV et radio numérique, ...
- Ordinateurs : asservissement des moteurs pour les disques durs, ...





[ICE97] *B. McCleanICE, "Status 1997: A Report on the Integrated Circuit Industry", Integrated Circuit Engineering Corporation (ICE), Scottsdale, 1997*

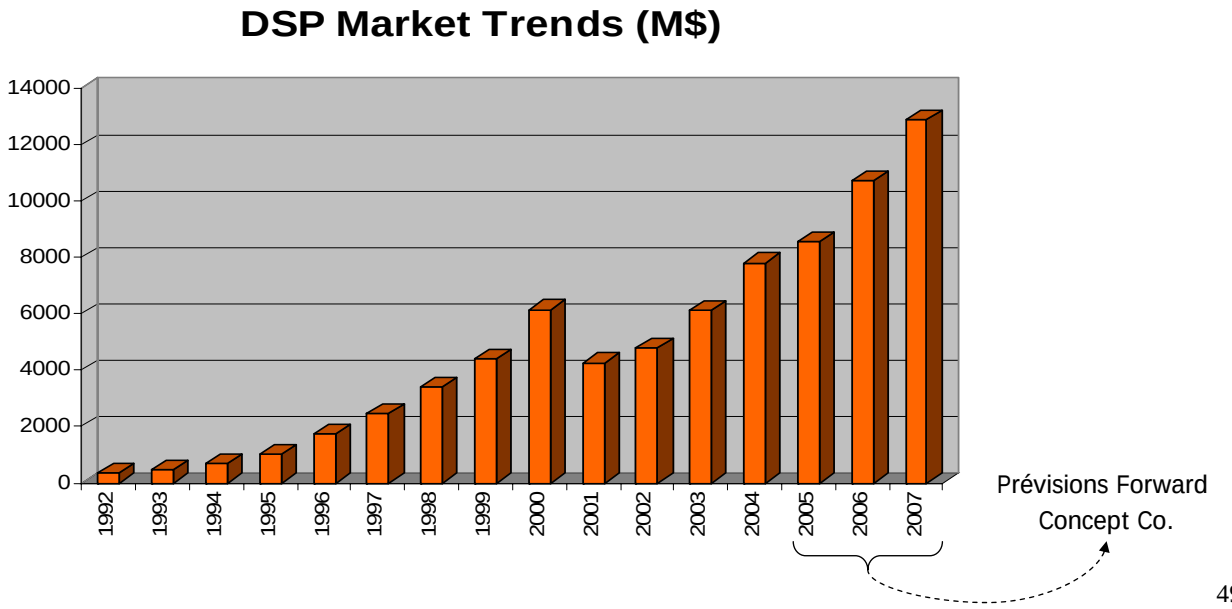
Notes :

Notes

Notes :

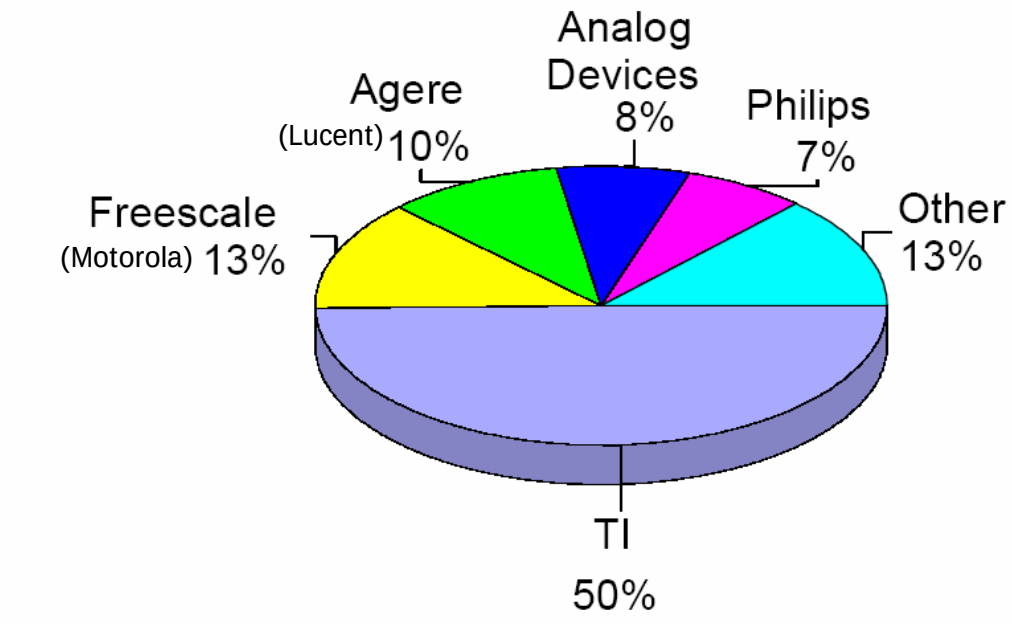
Le marché des processeurs DSP

- En haut de la liste des plus fortes croissances du marché de l'industrie des semi-conducteurs.



Notes :

Le marché des processeurs DSP

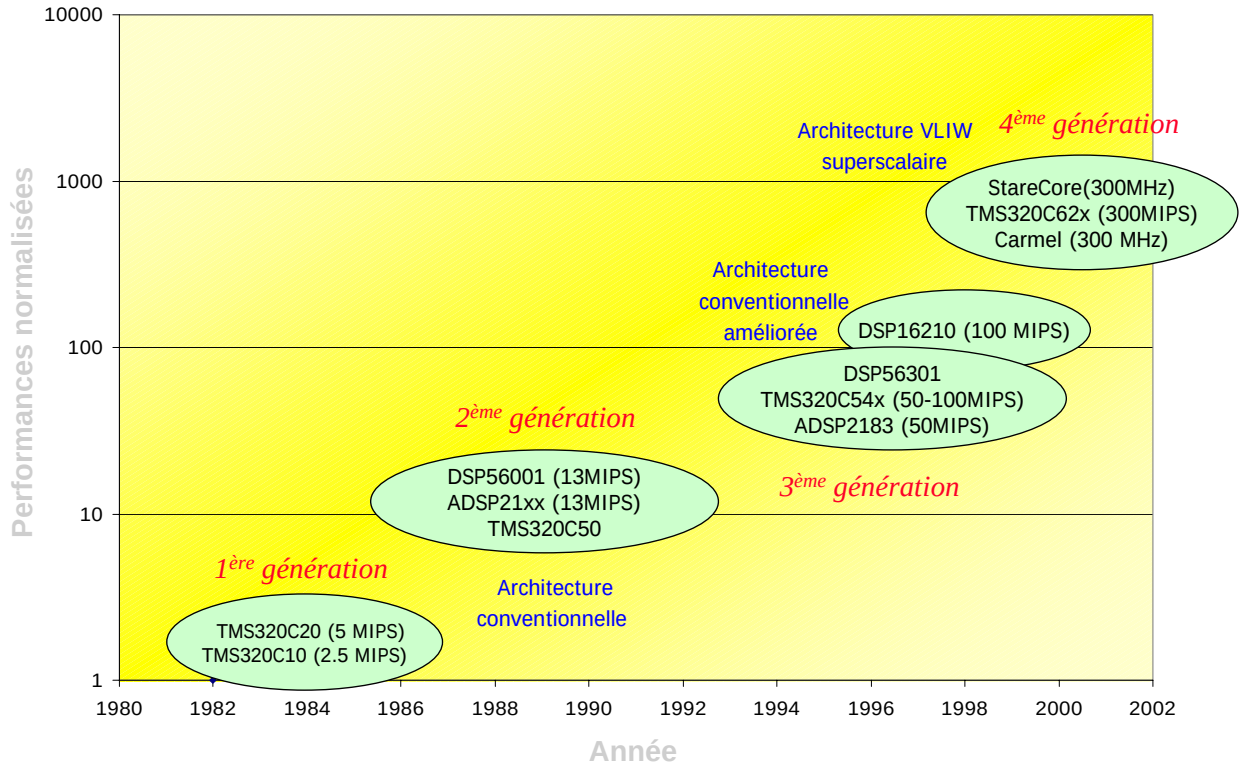


Tendance du marché :

vers des solutions à intégration de cœurs de DSP

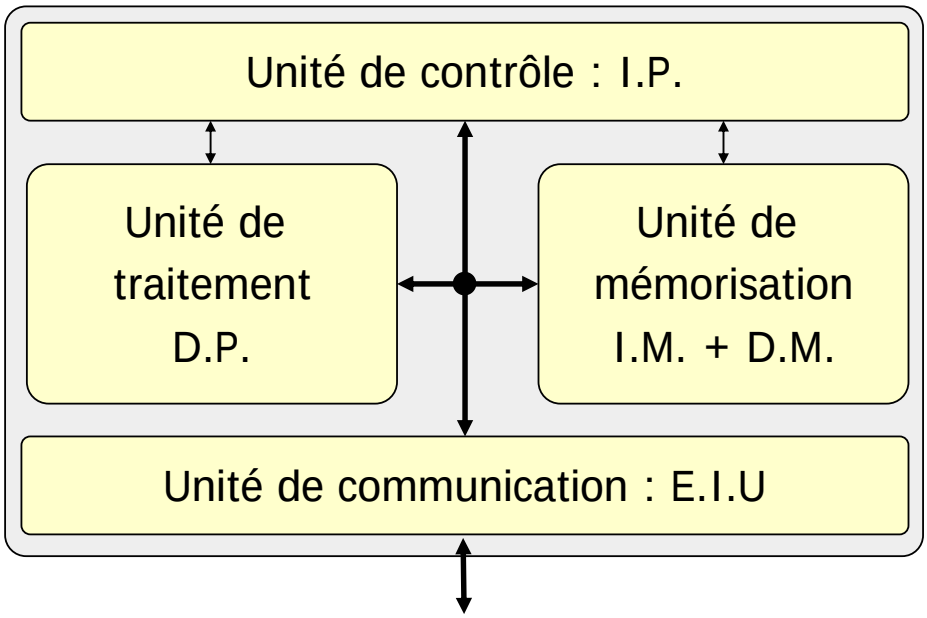
Notes :

Les différentes générations



Notes :

Modélisation d'un processeur



Notes :

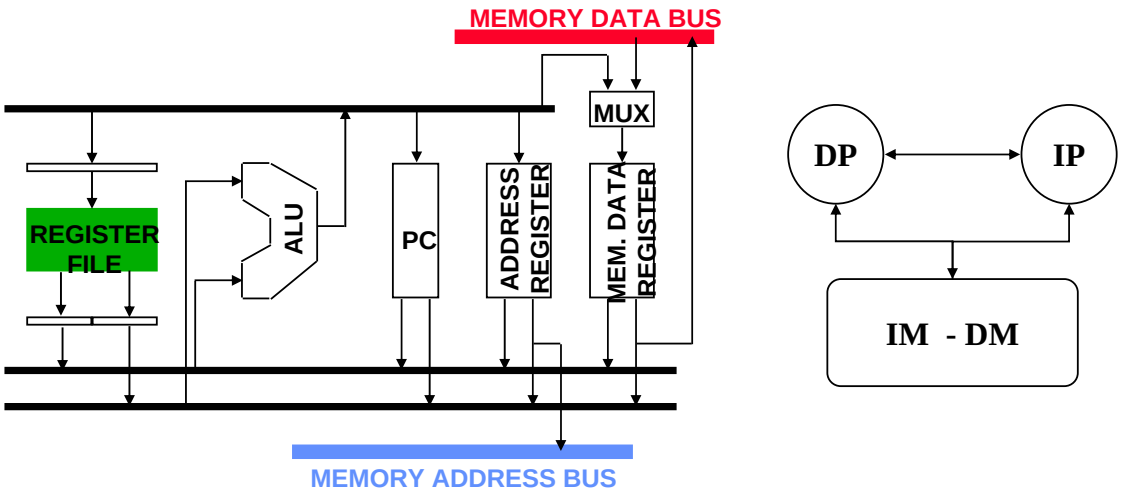
Modélisation d'un processeur

- Unité de contrôle (IP : *Instruction Processor)* : unité fonctionnelle (UF) qui interprète les instructions et commande les autres UF
- Unité de traitement (DP : *Data Processor)* : UF qui modifie ou transforme les données
- Unité de mémorisation:
  - IM : *Instruction Memory* : stocke les instructions
  - DM : *Data Memory* : stocke les données traitées par le DP
- Unité de communication (EIU : *External Interface Unit)* : contrôle les accès aux données ou instructions externes, ou à d'autres processeurs

Notes :

Architecture Von Neumann

- Les processeurs RISC

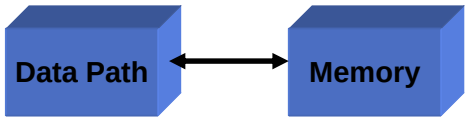


Notes :

FIR sur machine Von Neumann

- Problèmes :
  - Bande passante avec la mémoire et gestion des pointeurs d'adresse
  - Code pour le contrôle (gestion de la boucle)
  - Multiplication lente

```
loop:
   mov *r0,x0      } Lecture des opérandes sources
   mov *r1,x1      }
   mpy x0,x1,a     } Opération MAC
   add a,b         }
   mov x1,*r2      } Vieillissement de l'échantillon
   inc r0          } Gestion des pointeurs d'adresse
   inc r1          }
   inc r2          }
   dec ctr         } Gestion de la boucle
   tst ctr         }
   jnz loop        }
```



Exécution en 15 à 20 cycles

Notes :

Notes

Notes :

Objectifs

- Unité de traitement :
  - Réaliser efficacement les traitements typiques en TNS
    1 MAC par cycle
- Unité de mémorisation :
  - Alimenter efficacement en données l'unité de traitement afin de ne pas la ralentir
  - Gérer efficacement les pointeurs d'adresse
- Unité de contrôle :
  - Gestion efficace des structures de contrôle utilisées en TNS : boucles
  - Encodage des instructions
    - Minimiser la taille des instructions
    - Encoder le maximum de parallélisme

Notes :

Caractéristiques des DSP

Propriétés du traitement numérique du signal	Conséquences sur les architectures
Calculs intensifs et répétitifs	♦ Fonctionnement pipeline ♦ Architecture Harvard ♦ Structures de contrôle évoluées
Primitives simples	♦ Unités de traitement spécialisées câblées ♦ Gestion complexe de l'adressage

Le tout dans un environnement temps réel

Notes :

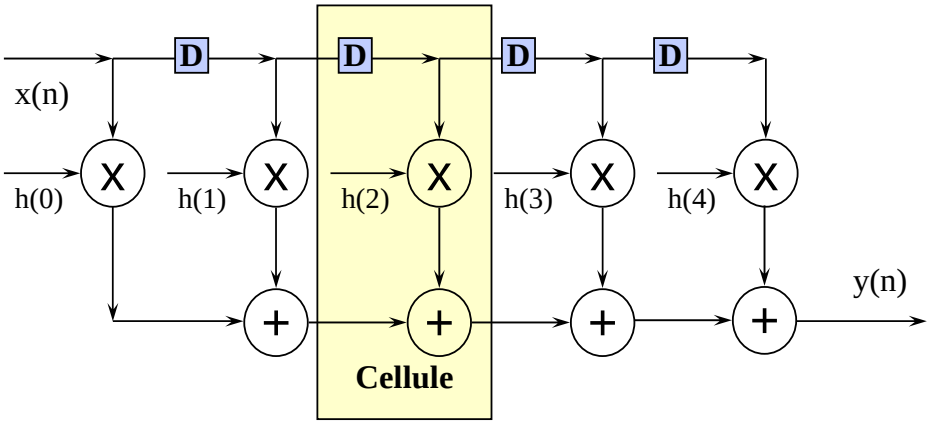
I. Processeurs de traitement du signal

Notes :

Exemple *Fil Rouge*

- Filtre numérique FIR sur N points

$$y(n) = \sum_{i=0}^{N-1} h(i).x(n-i) = x(n) * h(n)$$

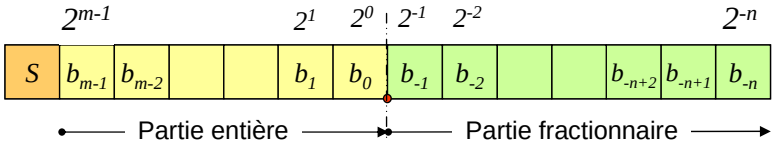


⇨ Objectif : traitement d'une cellule par cycle

Notes :

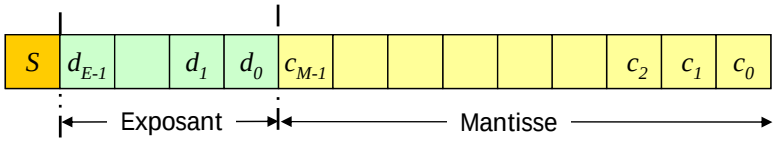
Représentation des données

- Virgule fixe
  - Le format d'une donnée ne varie pas au cours du temps
  - Représentation : partie entière - partie fractionnaire



$$x = (-2)^m S + \sum_{i=-n}^{m-1} b_i 2^i \quad CA2$$

- Virgule flottante
  - Représentation : exposant - mantisse



$$x = 2^u (-1)^{S_E} \left(\frac{1}{2} + \sum_{i=1}^{M} C_i 2^{i-1} \right)$$

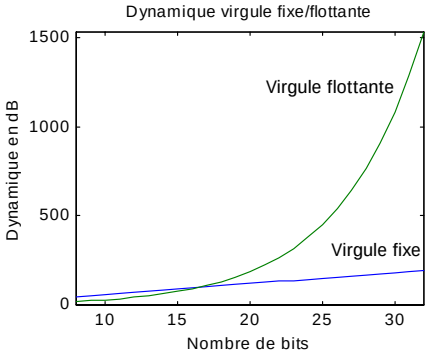
$$avec \quad u = (-1)^{S_E} \sum_{i=1}^{E-1} d_i 2^i$$

Notes :

Comparaison fixe - flottant

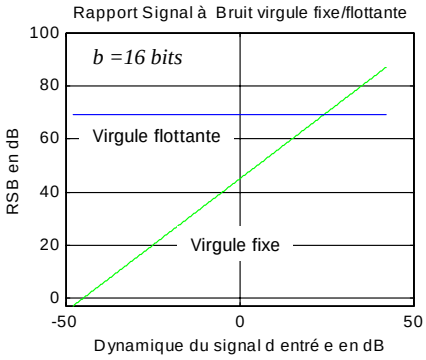
- Niveau de dynamique

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



- Rapport Signal à Bruit de Quantification

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



Notes :

DSP virgule fixe

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 :
  - 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

Notes :

Notes

Notes :

DSP virgule flottante

- Arithmétique :
  - Dynamique importante : 1500 dB pour 32 bits

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

- 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

Notes :

Eléments de l'UT

- Opérateurs
  - Multiplieur câblé

    Multiplication en 1 cycle ou *pipelinée* (1 résultat de multiplication par cycle)

    Le résultat est fourni directement à l'UAL ou il est stocké dans un registre (P register)

    Largeur des opérandes source : b_{nat}

    résultat : $b_{mult} = 2.b_{nat}$
  - U.A.L.

    Opérations arithmétiques : addition, soustraction, incrémentation, négation

    Opérations logiques : and, or, not

    Largeur des opérandes sources et destination identique $b_{add} \geq 2.b_{nat}$
  - Additionneur indépendant de l'UAL
  - Registres à décalage (recadrage des données)

    spécialisé : réalisation de quelques décalages prédéfinis en //

    en barillet : réalisation d 'un décalage quelconque en 1 cycle

Notes :

Éléments de l'UT

- Unités de saturation ou d'arrondi
- Unités spécifiques

  Unité de manipulation de bit, Viterbi, ...

• Unités de stockage de l'UT

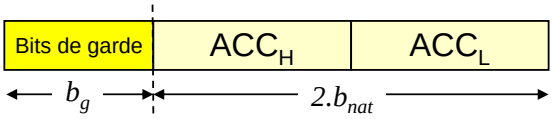
- Registres opérandes

  Stockage des opérandes sources ou des résultats intermédiaires

- Registres d'accumulation

  Stockage du résultat de l 'additionneur
  Nombre de registres d'accumulation limité (1 à 4)
  Données stockées en double précision
  Possibilité de bits de garde pour stocker les bits supplémentaires issus d 'accumulations successives

  $b_{add} = b_{mult} + b_g$



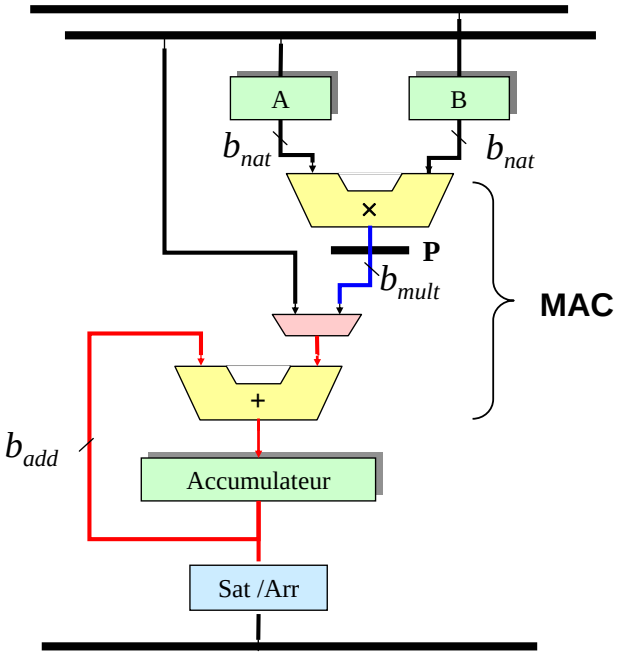
Notes :

Structure de l'UT de type MAC

Opérateurs	N_{bits} Entrées	N_{bits} Sortie
Multiplieur	b_{nat}	$2\ b_{nat}$
Additionneur/ UAL	$2\ b_{nat}$ $2\ b_{nat} + b_g$	$2\ b_{nat}$ $2\ b_{nat} + b_g$
Saturation/Arrondi	$2\ b_{nat}$ $2\ b_{nat} + b_g$	b_{nat} b_{nat}

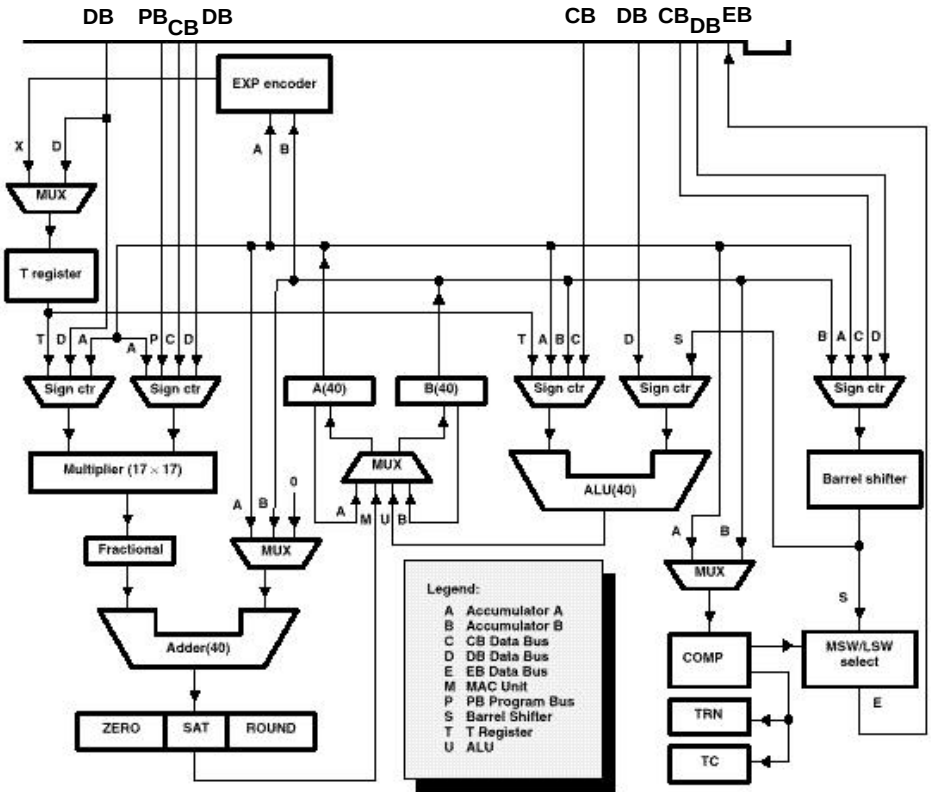
Registre	N_{bits}
Opérande source	b_{nat}
Accumulateur	$2\ b_{nat}$ $2\ b_{nat} + b_g$

Interconnexions registres - opérateurs spécialisées
⇨ Structure hétérogène
⇨ Bonnes performances en terme de consommation et de surface



Notes :

Exemple : TMS320C54x



- 1 multiplieur 16*16 bits
  - Op source 1 : registre T
  - Op source 2 : mémoire
  - Op destination :
- 1 additionneur 40 bits
- 1 ALU (40 bits)
- 2 registres d 'accumulation 40 bits
- 1 registre à décalage en barillet
- 1 unité dédiée à l 'algorithme de Viterbi

Notes :

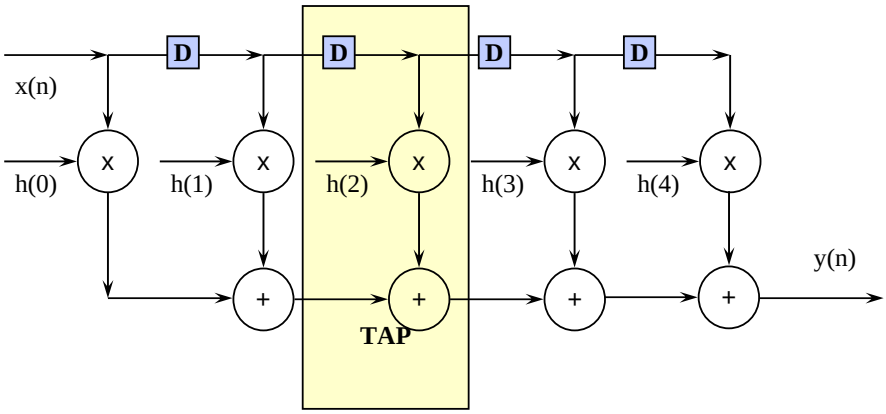
I. Processeurs de traitement du signal

1. Introduction
2. **Description des différentes unités**
   - 2.1. Unité de traitement
   - **2.2. Unité de mémorisation**
   - 2.3. Unité de contrôle
   - 2.4. Unité de communication
3. Panorama des processeurs
4. Outils de développement

Notes :

Exemple FIR *Fil Rouge*

- Les différents accès à la mémoire:
  - Recherche de l'instruction
  - Lecture de la donnée x_{n-k}
  - Lecture du coefficient h_k
  - Vieillissement des données $x_{n-k-1} = x_{n-k}$



Notes :

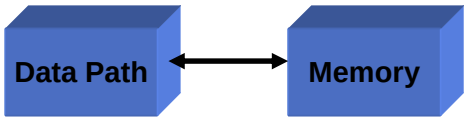
Notes

Notes :

FIR sur machine Von Neumann

- Problèmes :
  - Bande passante avec la mémoire
    - Lecture des échantillons
    - Vieillissement des échantillons
  - Gestion des pointeurs d'adresse

```
loop:
   mov *r0,x0      } Lecture des opérandes sources
   mov *r1,x1      }
   mpy x0,x1,a     } Opération MAC
   add a,b         }
   mov x1,*r2      } Vieillissement de l'échantillon
   inc r0          }
   inc r1          } Gestion des pointeurs d'adresse
   inc r2          }
   dec ctr         }
   tst ctr         } Gestion de la boucle
   jnz loop        }
```



Exécution en 15 à 20 cycles

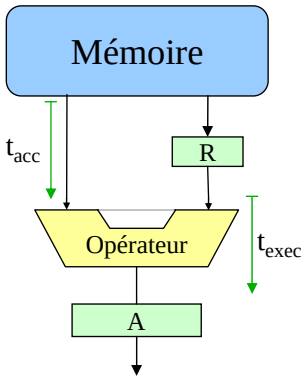
Notes :

Localisation des opérandes

- Modèle registre-mémoire
  - Opérandes situées en mémoire et dans les registres

    Temps d 'exécution de l 'instruction

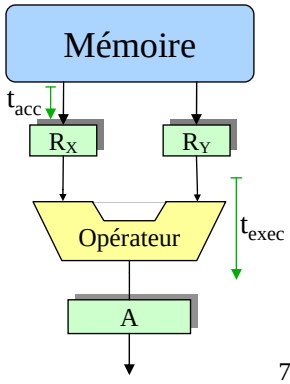
$$t_{inst} = t_{acc} + t_{exec}$$



- Modèle « Load-Store »
  - Opérandes situées uniquement dans des registres

    Temps d 'exécution de l 'instruction

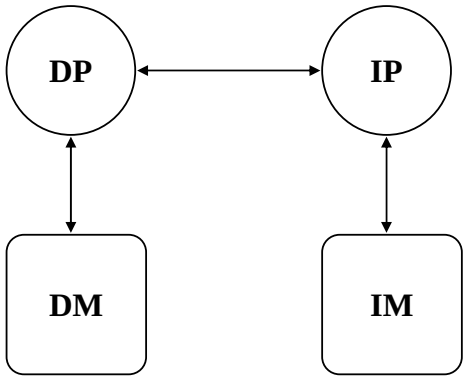
$$t_{inst} = max(t_{exec}, t_{acc})$$



Notes :

Architecture Harvard de base

- Architecture Harvard : séparation de la mémoire données et de la mémoire programme
  - ⇨ *Fetch* d'instruction pipeliné avec *fetch* opérande



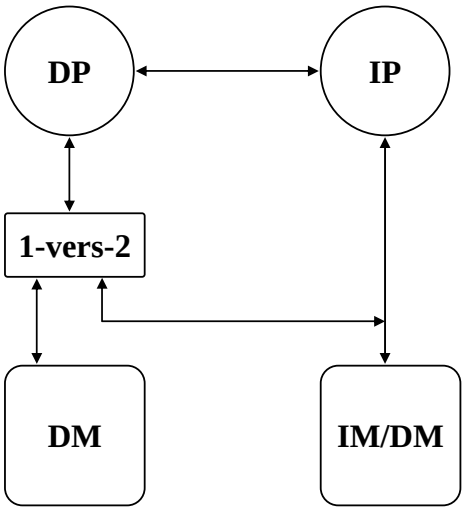
Ex: TMS320C10

Classification de E. Lee (1989)

Notes :

Modification 1

- Autorisation de mémorisation de données dans l'IM
- En un cycle si 2 accès mémoire par cycle ($t_{acc} = 1/2.t_{inst}$) :
  - *fetch* de l'instruction
  - *fetch* deux opérandes de la mémoire
  - exécution d'un MAC
  - écriture du résultat en I/O ou mémoire

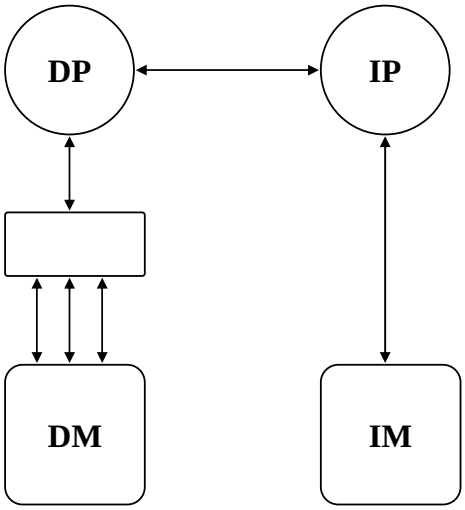


AT&T DSP32 et DSP32C

Notes :

Modification 2

- DM est une mémoire multi-ports, plusieurs accès aux données par cycle
- Utilisable pour des mémoires internes au CI

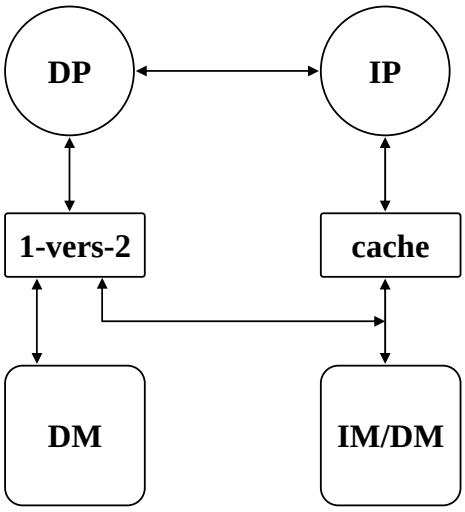


Fujitsu MB86232 (3 ports en mémoire interne)

Notes :

Modification 3

- Cache pour charger les instructions fréquentes
- Évite les conflits d'accès données et instructions de la modification 1

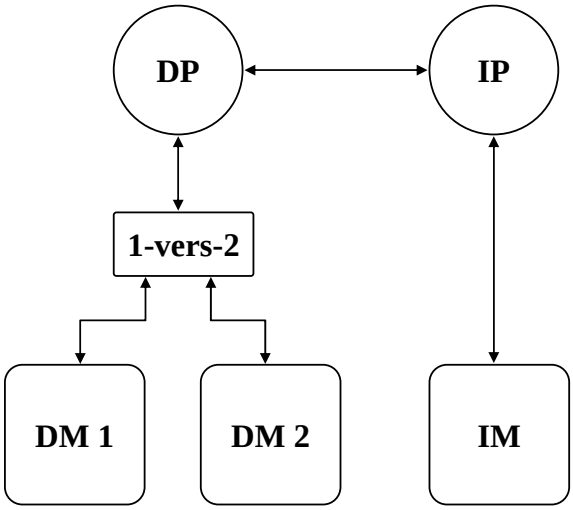


TMS320C25 : **cache 1 instruction pour les boucles**
DSP16 : **cache 15 instructions**
ADSP-2100 : **cache 16 instructions**

Notes :

Modification 4

- Deux mémoires données DM séparées
- En un cycle :
  - *fetch* de l'instruction
  - *fetch* de deux opérandes (si les temps d 'accès aux mémoires DM1, DM2 et IM sont identiques)



Motorola DSP 56001 et 96002
TMS320C30 et C40

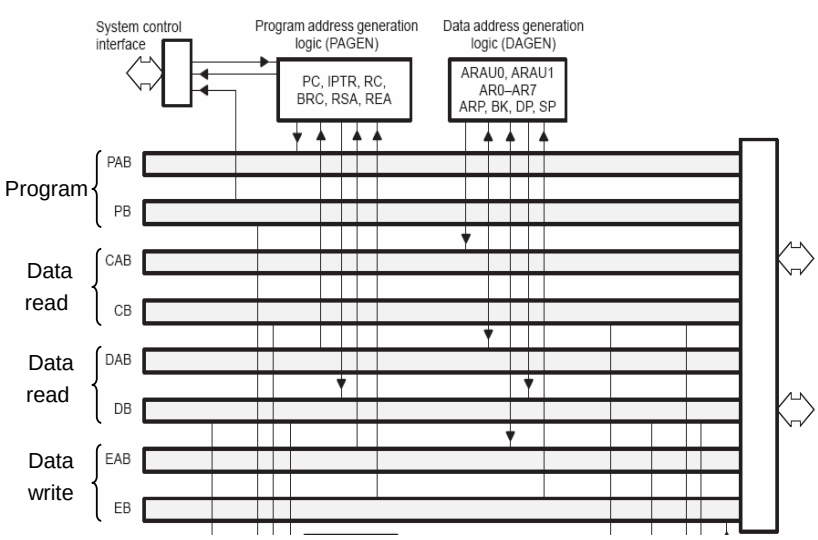
Notes :

Notes

Notes :

Modèle du C54x

- Bus et mémoires internes :



Mémoire interne

ROM :

- Bootloader
- Utilisation dans l'espace *data* ou *program*

DARAM (Dual Acces RAM)

- 1 lecture + 1 écriture par cycle

SARAM (Single Acces RAM)

- 1 lecture ou 1 écriture par cycle

Notes :

Modes d'adressage

- Adressage immédiat : la donnée est stockée directement dans l'instruction
  - Exemple C54x : LD #75h, A

    A = 75h

    Adressage court (instruction sur 1 mot) : valeur spécifiée sur 3,4,8 ou 9 bits

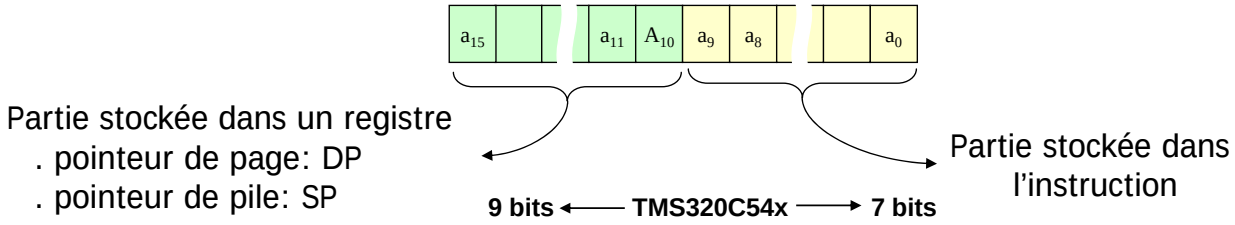
    Adressage long (instruction sur 2 mots) : valeur spécifiée sur 16 bits
  - Utilisé pour l 'initialisation des registres
  - Inconvénient : augmentation du temps d 'exécution et de la taille du code

- Adressage registre directe : les données sont stockées dans des registres
  - Exemple C54x SUB A, B

Notes :

Modes d’adressage

- Adressage mémoire directe : l’adresse de la donnée est stockée dans l’instruction
  - Adressage absolu : l ’adresse complète est stockée dans l’instruction
    C54x les adresses sont sur 16 bits
    L ’instruction doit être codée sur plusieurs mots
  - Adressage paginé : pour limiter le nombre de bits stockés dans l ’instruction l’adresse est composée de deux parties :



Notes :

Modes d'adressage

- Adressage indirecte par registre :
  - Registre d'adresse (AR) pointant sur les données

LD *AR1, A

addr = AR1 (A⇦ *AR1)

  - Possibilités de post modifications :

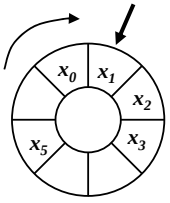
linéaire : AR:= AR ± 1 LD *AR1+, A

addr = AR1
AR1 = AR1 + 1

indexé : AR:= AR ± MR LD *AR1+0, A

addr = AR1
AR1 = AR1 + AR0

    - MR: registre d'index



modulo : (AR:= AR ± 1)$_N$ LD *AR1+% ,A

addr = AR1
AR1 = (AR1 + 1) modulo BK

(BK) specifies the size of the circular buffer.

bit-reverse : FFT LD *AR1+0B ,A

addr = AR1
AR1 = bitrev(AR1 + AR0)

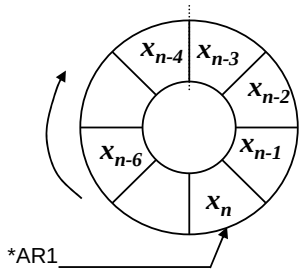
After access, AR0 is added to ARx with reverse carry (rc) propagation.

Notes :

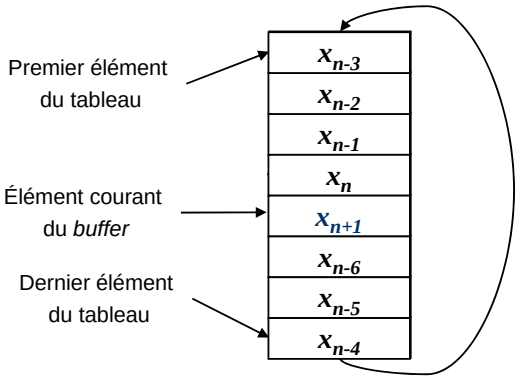
Buffer circulaire et *bit-reverse*

– *Buffer* circulaire :

Vieillissement automatique des données



N=8



– Bit-Reverse :

. Adressage des données en entrée ou en sortie de la FFT

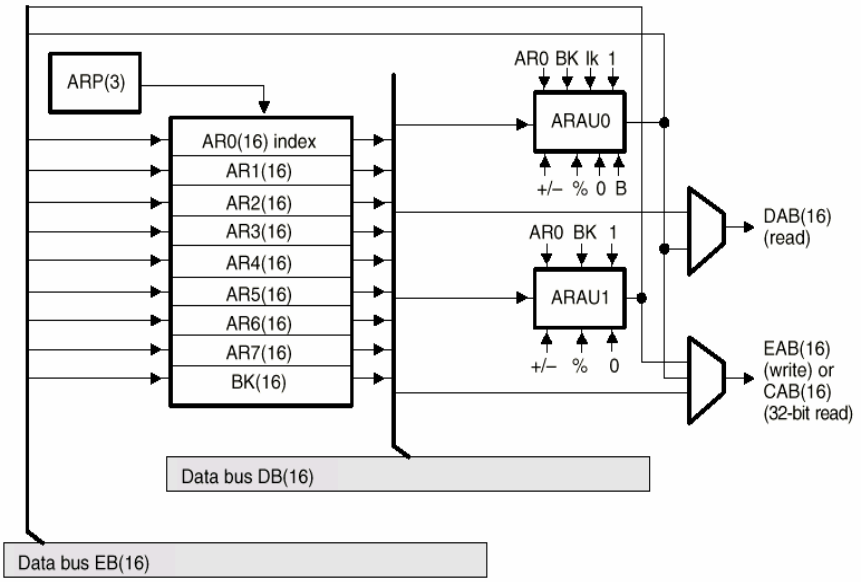
. Attention à l 'adresse de début du tableau : xxxx0000

Index	bit	bit reverse	index bit reverse
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Notes :

Unité de gestion des adresses

- Unité de gestion des adresses du TMS320C54x



- 8 registres auxiliaires (AR0…AR7)
- 2 unités de calcul ARAU
- registres spécifiques
  - BK : taille du buffer circulaire
  - AR0 : registre d 'index

Notes :

I. Processeurs de traitement du signal

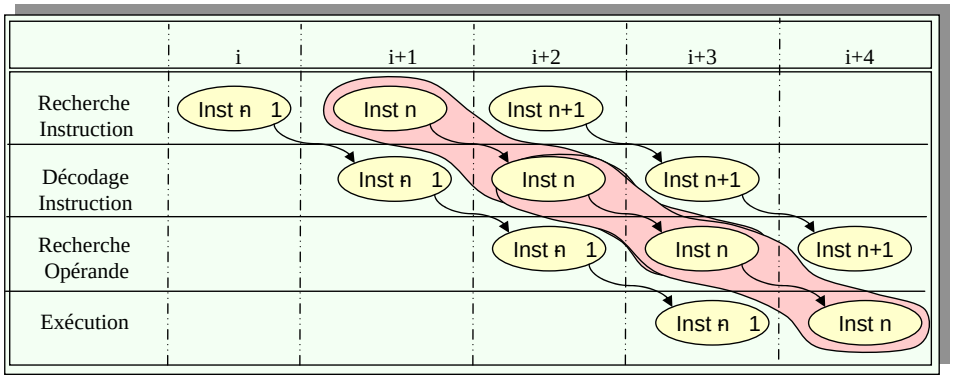
Notes :

Notes

Notes :

Pipeline

- Exécution de l'instruction en plusieurs phases :
  - 1. Recherche de l'instruction
  - 2. Décodage de l'instruction
  - 3. Recherche des opérandes
  - 4. Exécution

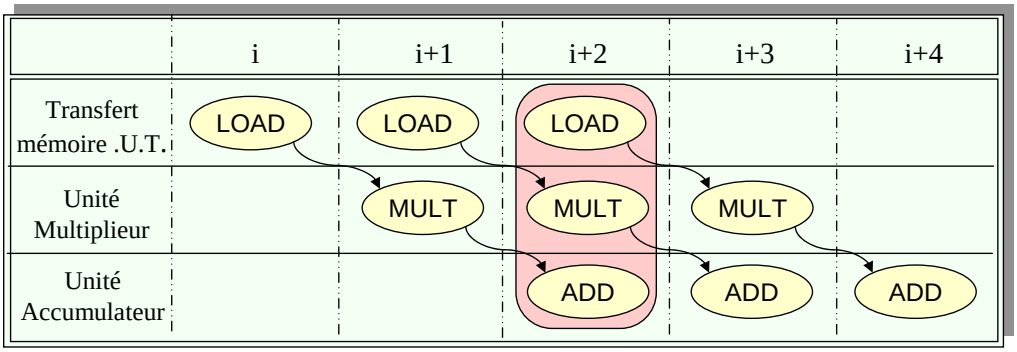


Notes :

Codage des instructions

- Stationnarité temporelle:
  - Une instruction définit l'ensemble des opérations à réaliser pour chaque unité fonctionnelle

```
MULT R1,R2,T     ADD T,A,A     LD R1,AR1     LD R2,AR2
```

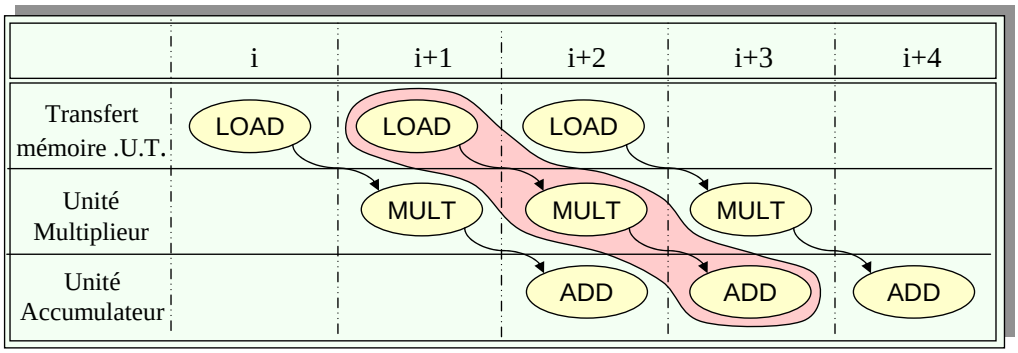


Notes :

Codage des instructions

- Stationnarité des données:
  - Une instruction définit une séquence complète d'opérations à réaliser sur un ensemble de données.

```
MAC *AR1,*AR2
```



Notes :

Format des instructions

Compromis

Minimiser la largeur des instructions
⇩
Diminuer la consommation et la surface mémoire & bus
⇩
Abaisser les coûts

Maximiser le parallélisme
⇩
Augmenter l'efficacité du processeur

- Format encodé: *largeur minimale (16 à 32 bits)*
  - Restrictions:
    - nombre d 'opérations disponibles
    - modes d 'adressage
    - opérandes sources et destination
  - Utilisation de bits de mode

Jeu d'instructions hétérogène

Notes :

Structures de contrôle

- Boucle matérielle
  - Optimiser le traitement des boucles de petite taille
    - Initialisation des paramètres de la boucle en 1 instruction
    - Pas d'instructions supplémentaires pour la gestion de la fin de la boucle
  - Exemple boucle mono-instruction

Boucle logicielle

```
        MOVE  #16,B
LOOP    MAC   (R0)+,(R4)+,A
        DEC   B
        JNE   LOOP
```

```
        RPT   #16
        MAC   (R0)+,(R4)+,A
```

Boucle matérielle

  - Exemple boucle multi-instruction

DSP 56000

```
        DO    #16,END
        ...   Loop body
End
```

C50

```
        SPLK  #16
        RPTB  End-1
        ...   Loop body
End
```


Notes :

Structures de contrôle

- Instructions de branchement
  - Branchement multi-cycles: ajout de NOP entre BR et 1ère instruction
  - Branchement retardé : déplacement de l'instruction de branchement BR

- Instructions à prédicat
  - Instructions conditionnelles du type :

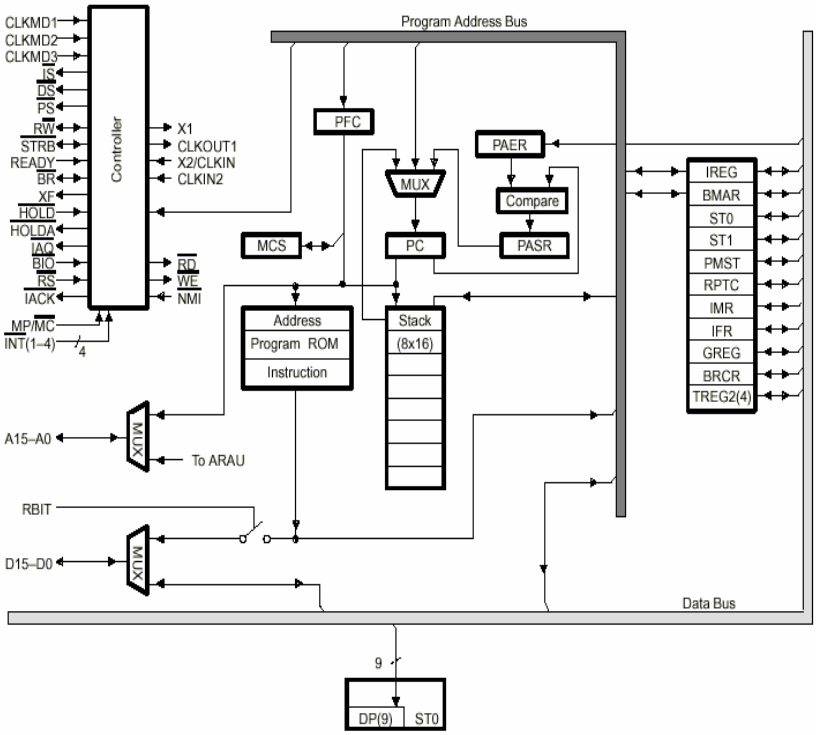
```
If              INF R1,R5,10
Then   [R1]     ADD R3,R2,3
Else   [!R1]    ADD R3,R2,3
```

- Instructions de mode
  - Définir un mode de fonctionnement spécifique

    exemple : contrôle d'un registre à décalage, d'une unité de saturation, ...

Notes :

Unité de contrôle du C50



- PC sur 16 bits
- pile de 8 * 16 bits
- PFC,IR: pour le pipeline

- Registres de status ST0,ST1,PMST

- Gestion des boucles
  - RPTC: repeat instruction
  - BRCR: repeat bloc

- Interruptions :
  - IMR: masque interruptions
  - IFR: flags interruptions

- BMAR: bloc move
- GREG: mémoire globale

Notes :

Notes

Notes :

I. Processeurs de traitement du signal

Notes :

Périphériques

- Périphériques intégrés:
  - ports séries
  - ports parallèles
  - timers
  - DMA
  - générateur de wait state
  - host port
  - PLL

- Connexion aisée aux CAN et CNA :
  - les CAN/CNA ne sont généralement pas intégrés dans les DSP afin de pouvoir choisir un CAN/CNA en adéquation avec l'application

Notes :

I. Processeurs de traitement du signal

Notes :

DSP conventionnels

- 16-24 bits en virgule fixe, accumulation 40-60 bits
- 32 bits en virgule flottante
- 16 à 32 bits d'instructions
- Une instruction par cycle, jeu d'instructions complexe
- Architecture non orthogonale, fortement contrainte
- Mémoire à accès multiples "on-chip"
- Unités dédiées de gestion des adresses
- Matériel dédié pour la gestion des boucles

100 millions de produits intègrent ces DSP

Notes :

Panorama des DSP

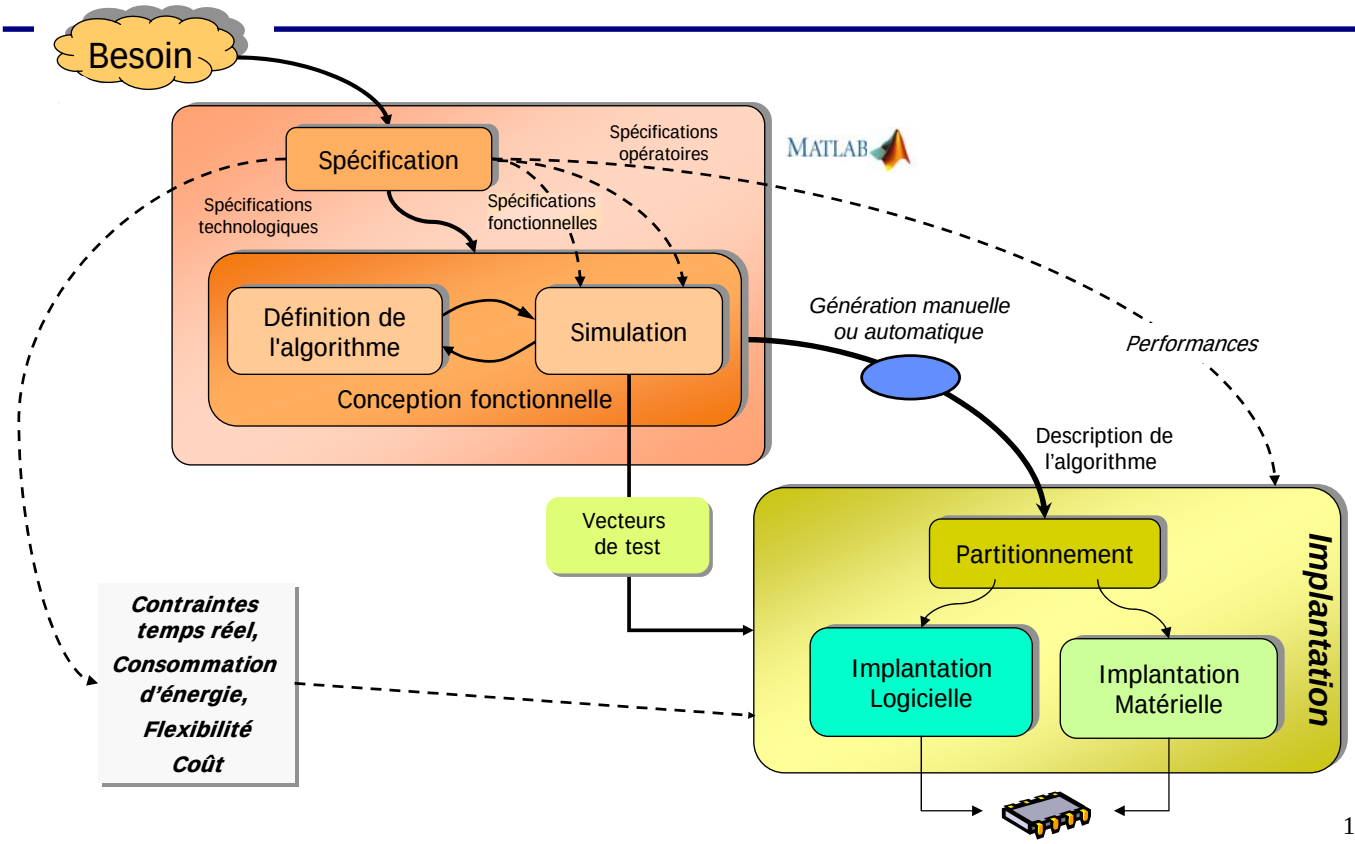
- Texas Instrument
  - C2000 : virgule fixe 16 bits : contrôle moteur
  - C5000 : virgule fixe 16 bits : faible consommation
    OMAP : C55x + µP ARM
  - C6000 : virgule fixe 16 bits (C67 flottant) : hautes performances
- Analog Device
  - ADSP21xx : virgule fixe 16 bits
  - SHARC : virgule flottante 32 bits
- Motorola
  - DSP560xx, DSP563xx : virgule fixe 24 bits : audionumérique
  - DSP566xx, DSP568xx : virgule fixe 16 bits
  - StareCore : virgule fixe 16 bits, hautes performances

Notes :

I. Processeurs de traitement du signal

Notes :

Cycle de développement (1)

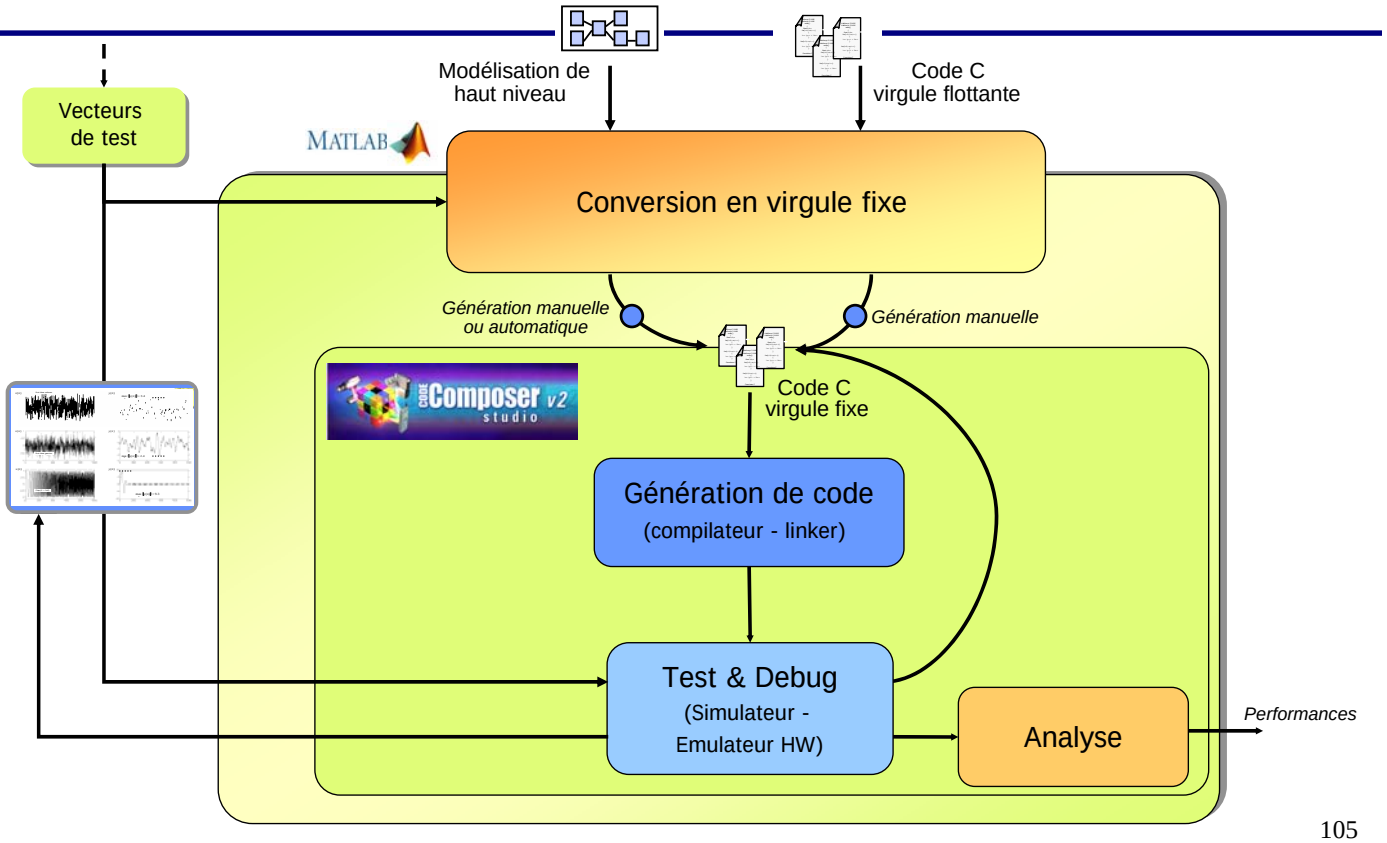


Notes :

Notes

Notes :

Cycle de développement (2)



Notes :

Cycle de développement (3)

- Conception fonctionnelle
  - Spécifications du flot de données du TS,
  - Simulation, validation
  - Prototypage rapide

    Matlab/Simulink (Mathworks), SPW (CoWare), Cossap (Synopsys), Ptolemy (Université de berkley)

- Développement du code
  - Implantation matérielle : code VHDL
  - Implantation logicielle : code C

Notes :

Cycle de développement (4)

- Implantation matérielle
  - Synthèse du circuit
  - Simulation VHDL et validation
  - Analyse des performances
  - Test

- Implantation logicielle
  - Génération de code (compilateur C/ass, linker)
  - Test & Debug : validation du code
  - Analyse : mesure des performances

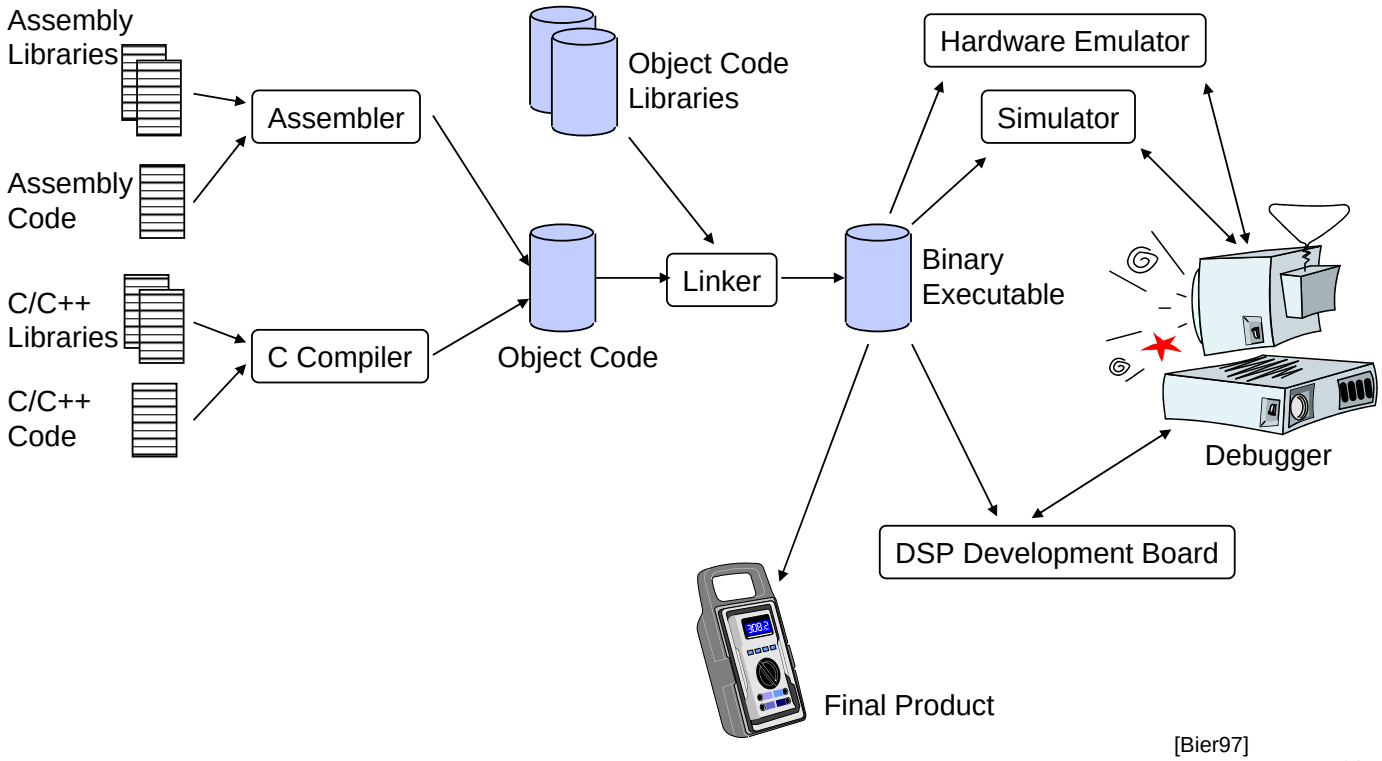
Notes :

Cycle de développement (5)

- Mise au point du logiciel - Debugging
  - Simulateur au niveau instruction
    - Mise au point avant que le matériel soit disponible
  - Emulation du Hardware
    - On-chip debugging/emulation : IEEE 1179.1 JTAG standard
  - Carte de développement
    - Vérification temps réel de l'algorithme
    - Système avec des faibles volumes de production

Notes :

Développement logiciel



[Bier97]

Notes :

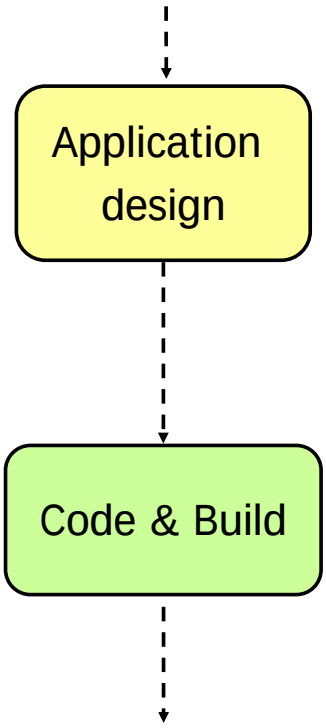
The degree to which ease of system development is a concern depends on the application. For example, users of a DSP-based rapid prototyping or simulation acceleration system in a research environment will probably require tool.s that make system development as simple as possible. On the other hand, a company developing a next-generation digital cellular telephone may be willing to suffer with poor development tools and an arduous development environment if the DSP chip selected shaves $5 off the cost of the end product. (Of course, this same company might reach a different conclusion if the poor development environment results in a three month delay in getting their product to market!)

That said, items to consider when choosing a DSP are software tools (assemblers, linkers, simulators, debuggers, compilers, code libraries, and real-time operating systems) hardware tools (development boards and emulators), and higher-level tools (such as block-diagrain-based code-generation environments).

A fundamental question to ask when choosing a DSP is how the chip will be programmed. Typically, developers choose either assembly language, a high-level language - such as C or Ada - or a combination of both. Surprisingly, a large portion of DSP programming is still done in assembly language. Because DSP applications often have voracious number-crunching requirements, programmers are often unable to use compilers which generate assembly code that executes slowly. Rather, programmers are often forced to endure long sessions hand oplimizing assembly code to lower execution time and code size to acceptable levels. This is especially true in consumer applications where cost constraints prohibit upgrading to a higher-performance DSP processor or adding a second processor.

Users choosing high-level language compilers often find that the compilers work better for floating-point DSPs than for fixed-point ones, for several reasons. First, most high-level languages do not have native support for fixed-point arithmetic. Second, floating-point processors tend to feature more regular, less restrictive instruction sets than smaller, fixed-point processors, and are thus better compiler targets. Third, as mentioned, floating-point processors typically support larger memory spaces than fixed-point processors, and are thus better able to accomodate compiler-generated code, which tends to be larger than hand crafted assembly code. ../..

Code Composer Studio

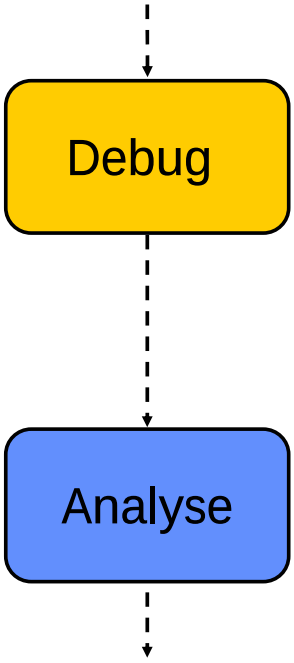


. Choix de la cible
. Configuration de l'OS DSP/BIOS
. Librairies :
  - DSPLib
  - TMS320 DSP Algorithm Standard

. IDE
. Générateur de code
  . Compilateur C
  . Compilateur assembleur
  . Linker
. Gestionnaire de projet

Notes :

Code Composer Studio



Simulateur
Émulateur HW

- **Exécution pas à pas**
- **Analyse de variables**
- **Déclenchement sur événements**

- **. Visualisation de signaux**
- **. *Profiler* (mesure du temps d'exécution du code)**
- **. Analyse temps réel (visualisation des tâches du DSP/BIOS)**

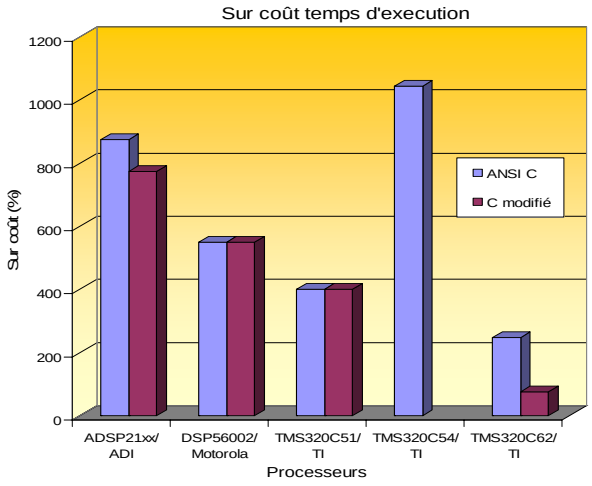
Notes :

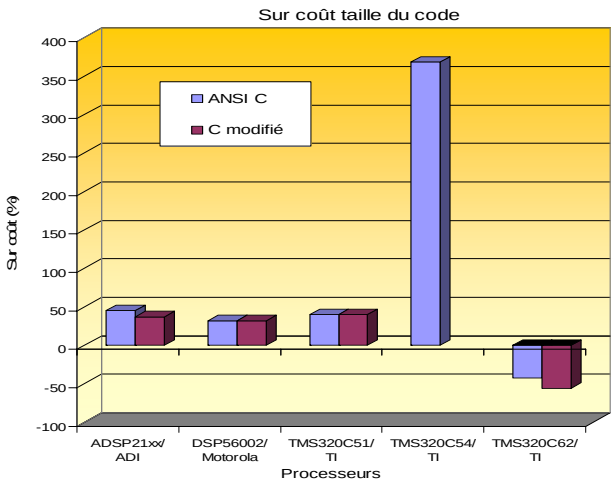
Notes

Notes :

Inefficacité des compilateurs C

- Analyse du surcoût associé au compilateur C : DSPStone





[Ropers99]

  - Meilleurs performances pour les DSP plus généraux et les DSP virgule flottante / architecture homogène

Notes :

Les raisons de l'inefficacité

- Inadéquation du langage C pour décrire des applications TNS
  - Absence de support pour l'arithmétique virgule fixe
    - Extensions du langage C
    - Langage orienté signal
- Inefficacité des techniques classiques de développement des compilateurs (développement du DSP puis du compilateur)
- Inefficacité des techniques classiques de compilation qui ont été développées pour les architectures homogènes
  - Les architectures des DSP conventionnels sont hétérogènes
- Absence de support pour les spécificités des DSP
  - Registres d'état, modes d'adressage (modulo, bit inversé)…

Notes :

Développement en C ou en assembleur

- Développement en assembleur :
  - Le code est optimisé
  - Le code n'est pas portable - demande du savoir faire
- Développement en C :
  - Plus abordable rapidement
  - Beaucoup moins performant et pas aussi simple ...
  - Conseils :

    ne pas déclarer ses variables en *float* (chaque opération fera appel à des routines de la bibliothèque de calcul en flottant)

    déclarer les variables en *int* et les recadrages sont gérés par des décalages :
    - nécessité de connaître les conventions utilisées par le compilateur
      - ex: renvoi d 'une donnée en double précision en mémoire

Notes :

Conclusion

- L ’architecture des DSP conventionnels a été optimisée afin d’implanter efficacement les applications de TNS
  - Conséquences :
    - Le coût et la consommation d’énergie sont très faibles
    - Les performances obtenues sont bonnes mais elles sont insuffisantes pour les nouvelles applications :
      - Téléphonie de 3ème génération ...
        - Nécessité d’utiliser des architectures nouvelles
    - Les compilateurs de langage de haut niveau sont peu efficaces
      - Les parties critiques de l’application doivent être codées à la main en assembleur

Notes :

Évolution des architectures

- Augmentation du nombre d'opérations exécutées par instruction
  - Architectures conventionnelles améliorées : Multi-MAC
  - Capacités SIMD
- Augmentation du nombre d'instructions exécutées par cycle
  - VLIW
  - Superscalaires
- Architectures clusterisées
- DSPs hybrides MCUs

Notes :

II. Arithmétique virgule fixe

1. Présentation de l'arithmétique virgule fixe

2. Processus de codage en virgule fixe

2.1. Détermination de la dynamique

2.2. Détermination de la position de la virgule

2.3. Détermination de la largeur des données

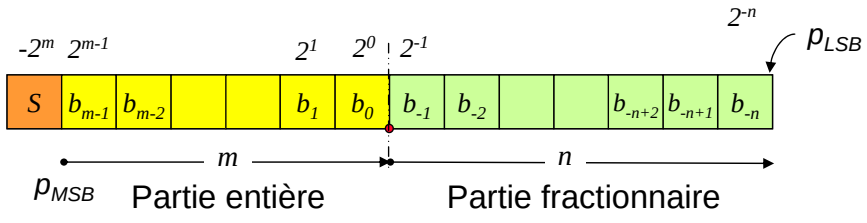
3. Exemple d'un filtre IIR cascadé

Notes :

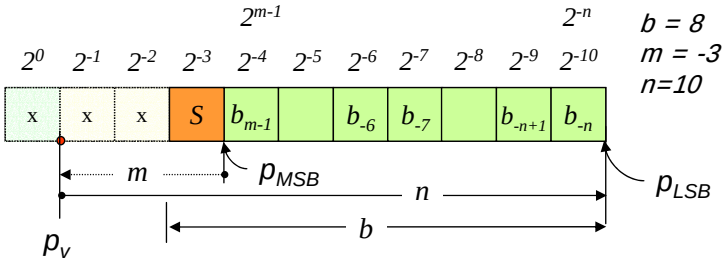
Codage en virgule fixe complément à 2

- 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)

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

Notes :

Notes

Notes :

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
1111.01	-0.75
1000.01	-7.75
1000.00	-8

-0.5 = -8+7. 5

-7.75 = -8+0.25

Notes :

Règles de l'arithmétique virgule fixe

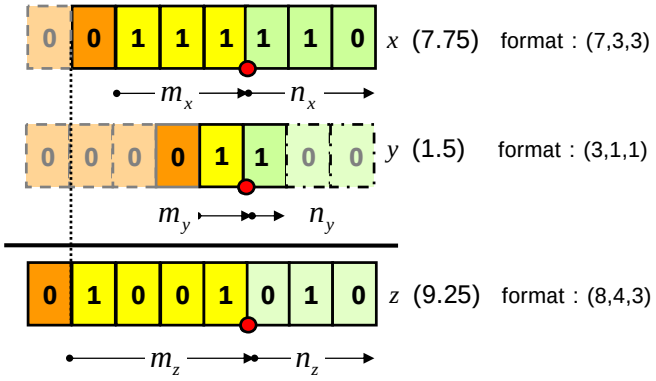
- Addition: $z=x+y$
  - *Règle* : le format des opérandes x et y doit être identique
  - *Étapes* :

    Choix d'un format commun $(b_{ADD},m_{ADD},n_{ADD})$

    Alignement de la virgule

    Extension des bits des opérandes *a* et *b*

$$m_{ADD} = \max(m_x, m_y)$$
$$n_{ADD} = \max(n_x, n_y)$$
$$b_{ADD} = m_{ADD} + n_{ADD} + 1$$



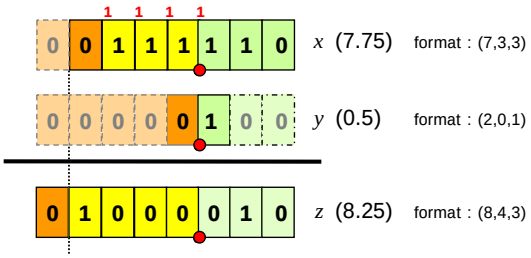
$$m_z = \begin{cases} m_{ADD}+1 & \textbf{si } x+y \notin D_{ADD} \\ m_{ADD} & \textbf{si } x+y \in D_{ADD} \end{cases}$$
$$n_z = n_{ADD}$$
$$b_z = m_{ADD} + n_{ADD} + 1$$

Notes :

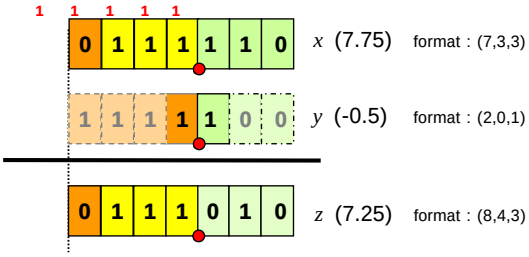
Exemple de calcul

- Addition: $z=x+y$
  - Débordement possible si $sign(x) = sign(y) = s_{xy}$

    Débordement présent si $sign(z) \neq s_{xy}$



Nécessité d'un bit supplémentaire pour coder le résultat



S 'il n 'y a pas de débordement la dernière retenue peut être ignorée

Notes :

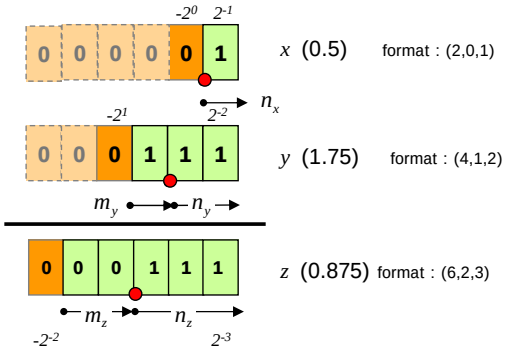
Règles de l'arithmétique virgule fixe

- Multiplication: $r = a \times b$
  - *Règle* : la représentation de *a* et *b* doit être identique
  - *Étapes* :

    Extension des bits de signe des opérandes *a* et *b*

    Doublement du bit de signe du résultat

    - Le bit redondant peut être intégré à la partie entière du résultat



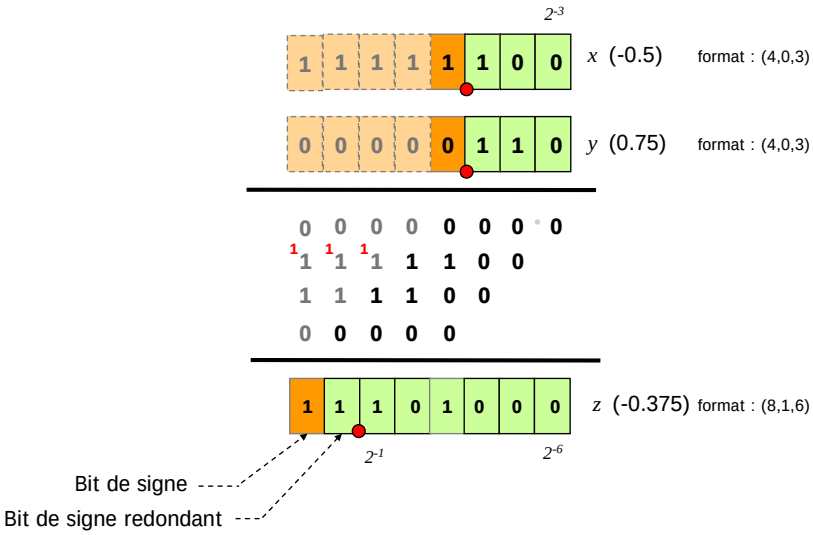
$$n_z = n_x + n_y$$
$$m_z = m_x + m_y + 1$$
$$b_z = b_x + b_y$$

Notes :

Exemple de calcul

- Multiplication: $r = a \times b$

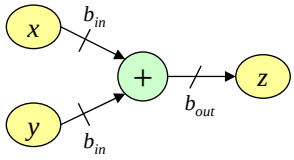
Extension du signe des opérandes



Notes :

Résumé des règles

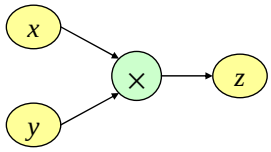
- Addition
  - Choix du format commun :



si $b_{out} = b_{in}$ **alors** $m_{ADD} = \max(m_x, m_y, m_z)$

si $b_{out} > b_{in}$ **alors** $m_{ADD} = \max(m_x, m_y)$

- Multiplication

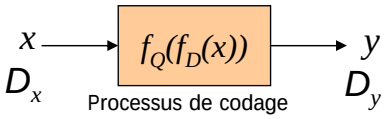


$$n_z = n_x + n_y$$
$$m_z = m_x + m_y + 1$$
$$b_z = b_x + b_y$$

Doublement du bit de signe

Notes :

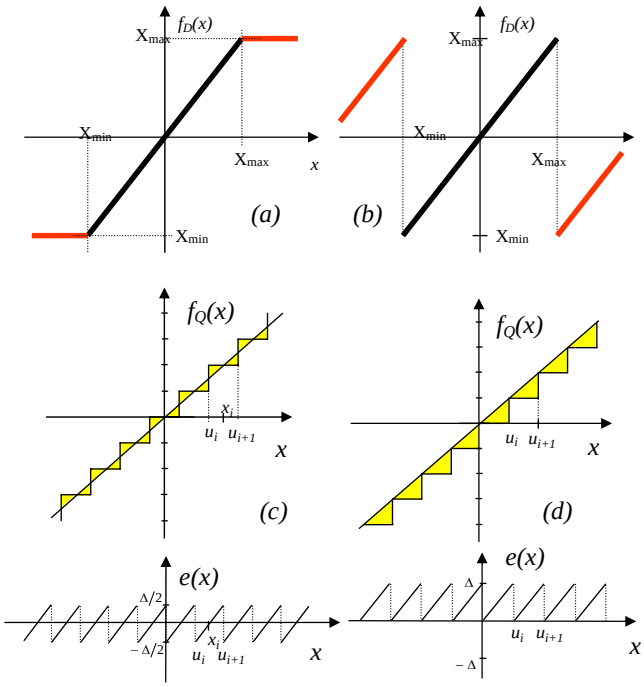
Processus de codage



- Loi de dépassement $f_D(x)$
  - Saturation *(a)*
  - Modulaire *(b)*

- Loi de quantification $f_Q(x)$
  - Arrondi *(c)*
  - Troncature *(d)*

    Erreur de quantification
    - $e(x)= x-y$



Notes :

Notes

Notes :

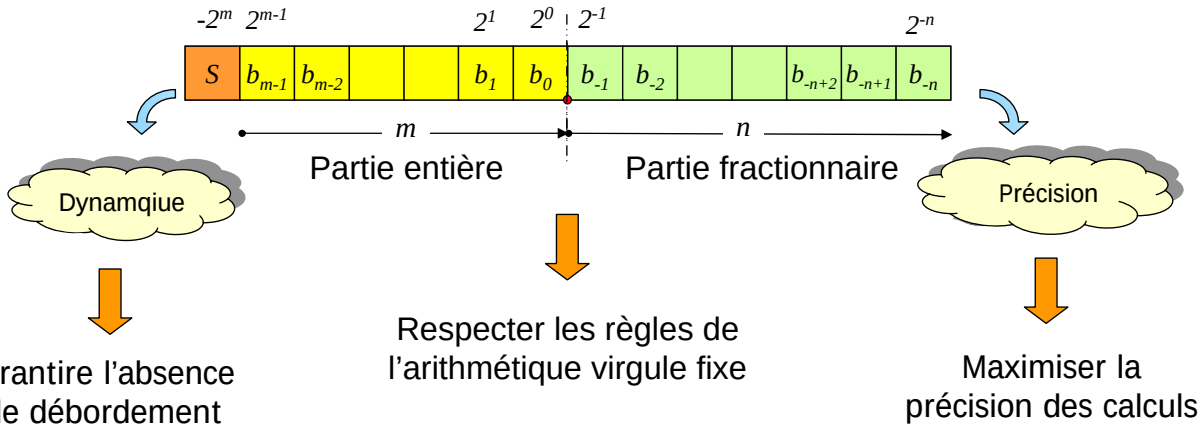
II. Arithmétique virgule fixe

Notes :

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



Garantire l'absence de débordement

Respecter les règles de l'arithmétique virgule fixe

Maximiser la précision des calculs

RSBQ (Rapport Signal à Bruit de Quantification)

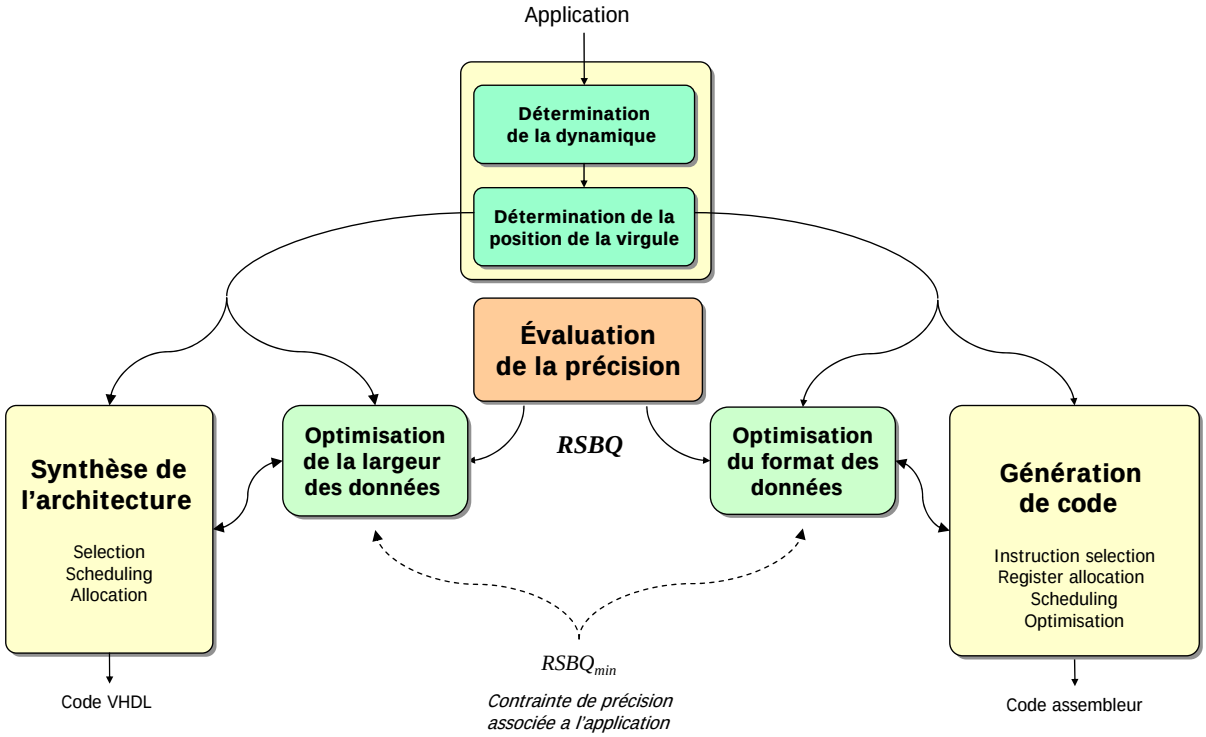
Notes :

Les différentes étapes du codage

- Détermination du domaine de définition : $D(x_i)$
  - Valeur minimale et maximale des données
- Détermination de la position de la virgule m_{xi}
  - Nombre de bits minimal pour représenter la partie entière permettant d'éviter les débordements
- Déterminer et optimiser la largeur des données b_{xi}
  - Utiliser au mieux l'architecture
- Évaluation de la précision de l'algorithme :
  - Calcul du Rapport Signal à Bruit de Quantification (cf. TNS partie 2)

Notes :

Méthodologie de codage



Implantation matérielle : FPGA, ASIC

Implantation logicielle : DSP, µC

Notes :

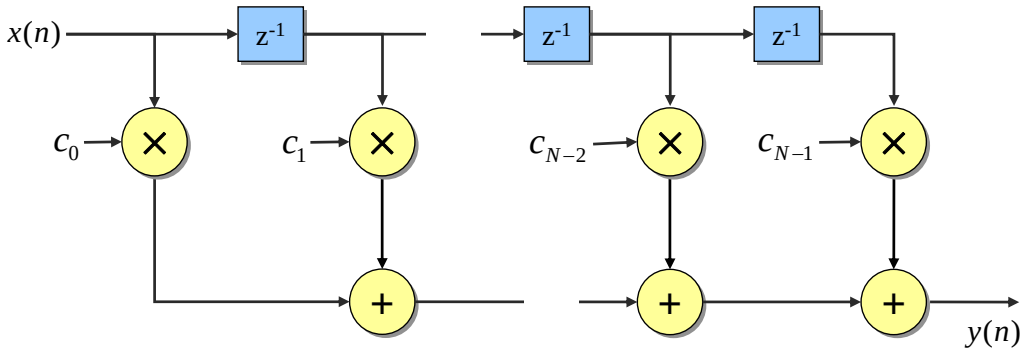
Fil rouge : filtre FIR

```
float x[N] = {0.123, -0.569,...} /* Définition du signal d 'entrée du filtre */
float h[N] = {0.2, 0. ,.., 0.2 }; /* Définition des coefficients du filtre*/

int main()
{
 float x[N], y[M], acc;
  int i,j;
  for(i=0; i<N; i++){x[i] = 0;}          /* Initialisation des variables internes du filtre */

  for(j=0;  j<M; j++)                    /* Filtrage du vecteur d'entrée input */
  {
        x[0] = Input[j];
        acc = x[0]*h[0] ;

        for(i=N-1; i>0; i--)
        {
           acc = acc + x[i]*h[i];        /* Calcul d'une cellule du filtre */
           x[i] = x[i-1];                /* Vieillissement des variables internes du filtre */
        }
        y[j] = acc;
   }
}
```


Notes :

Fil rouge : filtre FIR

- Graphe Flot de Signal du filtre FIR



Notes :

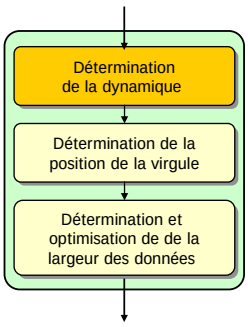
II. Arithmétique virgule fixe

Notes :

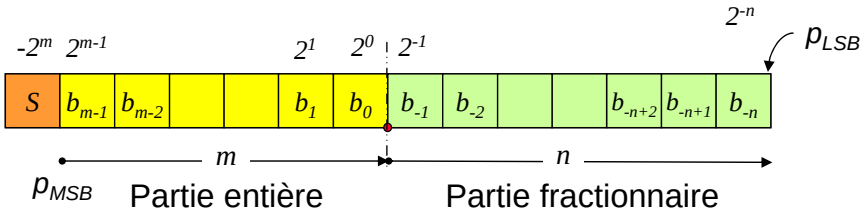
Notes

Notes :

Détermination de la dynamique



Format des données: *(b,m,n)*



But: *déterminer le nombre de bits minimal pour représenter la partie entière* (m_{xi})

$$m_{x_i} = \lceil \log_2 (\max (|x_i|)) \rceil$$

Notes :

Méthodes pour déterminer $D(x_i)$

- Méthodes statistiques :
  - Simulation de l'algorithme en virgule flottante
  - Détermination de la dynamique à partir des paramètres statistiques du signal

  ⇨ Estimation précise mais dépendante du signal d'entrée

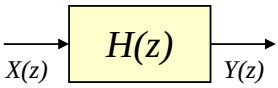
- Méthodes analytiques :
  - Détermination de l'expression de la dynamique
  - Utilisation de la norme L_1

  ⇨ Estimation conservatrice mais l'absence de débordement est garantie

Notes :

Méthodes analytiques

- Normes dans le cas des systèmes linéaires



– Norme L1:

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

➢ Méthode garantissant l'absence de débordement

– Norme Chebychev

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

➢ Méthode garantissant l'absence de débordement pour un signal d'entrée à bande étroite ($x(n)=\cos(wt)$)

– Norme L2

$$y_{\max 3} = \max_n \left(|x(n)| \right) . \sqrt{\sum_{m=-\infty}^{\infty} |h(m)|^2}$$

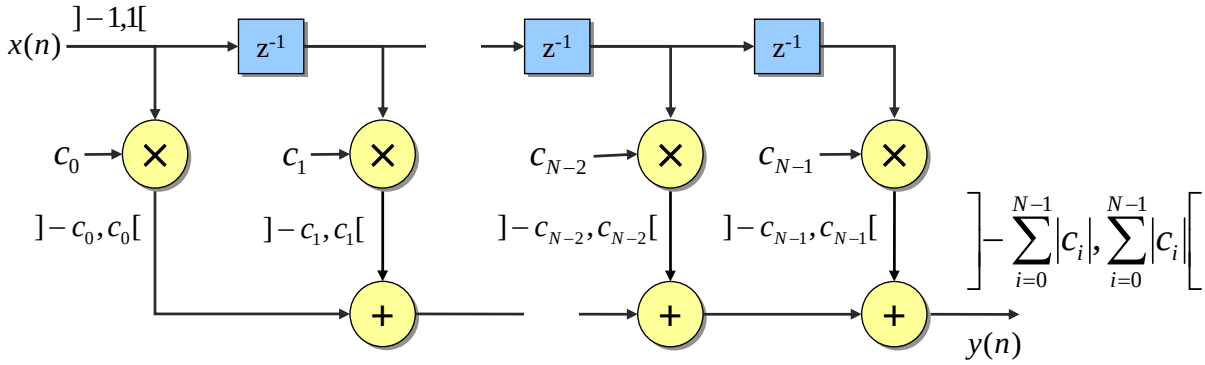
➢ Méthode limitant la probabilité de débordement

$$y_{\max 3} \leq y_{\max 2} \leq y_{\max 1}$$

Notes :

Filtre FIR

- Propagation de la dynamique des entrées au sein du GFS représentant l'application



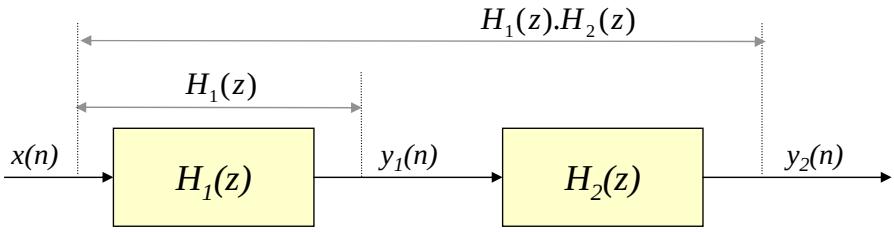
- Normes L1

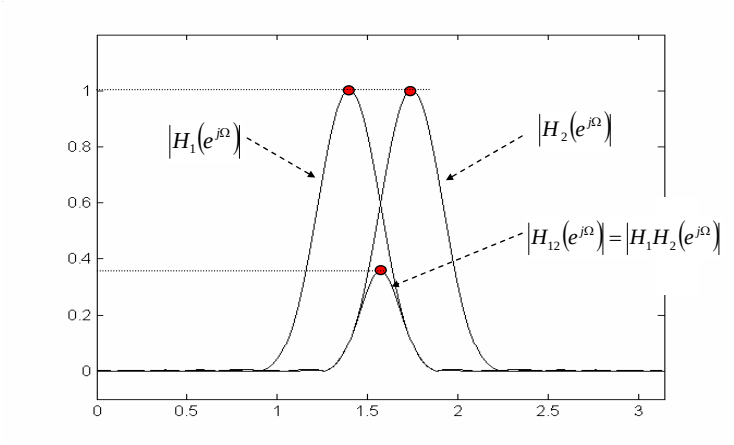
$$\max_n\left(\left|y(n)\right|\right)=\max_n\left(\left|x(n)\right|\right).\sum_{m=-\infty}^{\infty}\left|h(m)\right|=\sum_{m=0}^{N-1}\left|c_i\right|=10.65$$

Notes :

Dynamique dans un système cascadé

- Dynamique de la sortie y_2



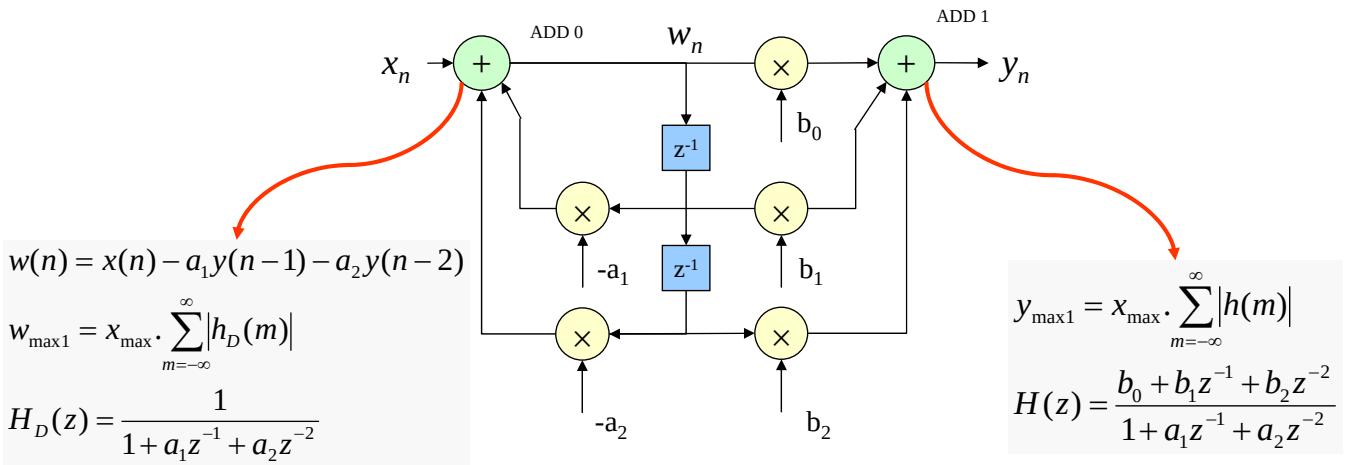


Notes :

Dynamique dans un filtre IIR

- Sources de débordement : additionneurs
  - Filtre IIR forme directe II: 2 sources ADD0 et ADD1

Forme directe II



$$w(n) = x(n) - a_1 y(n-1) - a_2 y(n-2)$$

$$w_{max1} = x_{max} \cdot \sum_{m=-\infty}^{\infty} |h_D(m)|$$

$$H_D(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

$$y_{max1} = x_{max} \cdot \sum_{m=-\infty}^{\infty} |h(m)|$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Notes :

Exemple : filtre IIR directe II

- Filtre H(z)

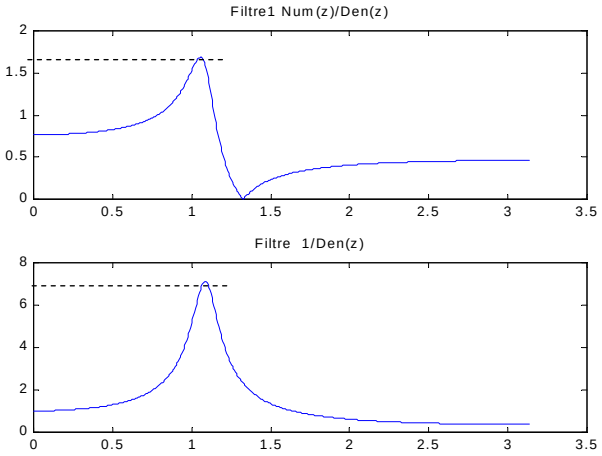
$$H(z) = \frac{0.5 - 0.2428z^{-1} + 0.5}{1 - 0.85z^{-1} + 0.8417z^{-2}}$$

- Norme L_1

$$x_{\max} \sum_{m=-\infty}^{\infty} |h(m)| = 2.5645 \longrightarrow m_y = 2$$

$$x_{\max} \sum_{m=-\infty}^{\infty} |h_D(m)| = 9 \longrightarrow m_w = 4$$

- Norme Chebychev



Fonction de transfert des filtres H(z) et $H_D(z)$

Notes :

Notes

Notes :

Exemple : filtre IIR directe II

- Comparaison des différentes méthodes

Méthodes	$G_{1.max}$	N_1	$G_{2.max}$	N_2
Méthodes analytiques				
Norme L_1	2,56	2	9	4
Norme Chebychev	1.687	1	7,13	3
Méthodes statistiques				
Chirp	1,66	1	7.12	3
Bruit blanc gaussien	0.74	0	2.34	2
Bruit blanc uniforme	1.4	1	4.87	3

$$G_{1\max} = \frac{y_{\max}}{x_{\max}} \qquad N_1 = \lceil \log_2(G_{1\max}) \rceil$$

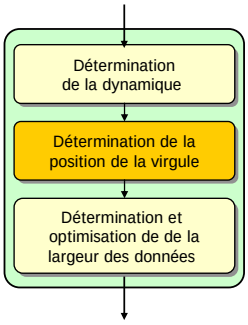
$$G_{2\max} = \frac{w_{\max}}{x_{\max}} \qquad N_2 = \lceil \log_2(G_{2\max}) \rceil$$

Notes :

II. Arithmétique virgule fixe

Notes :

Obtention de la position de la virgule



- Objectifs du codage en virgule fixe
  - Respect des règles de l'arithmétique virgule fixe
    - Alignement de la virgule des opérandes sources d'une addition ou d'une soustraction
    - Format commun pour les opérandes sources
  - Prévenir les débordements
    - Recadrage des données : adapter le format des données à la dynamique des données
    - Débordement possible lors d'une addition ou d'une soustraction
      - Nécessité d'introduire des bits supplémentaires

Notes :

Règles pour la position de la virgule

- Donnée x :

$$m_x = \left\lceil \log_2 \left(\max_n \left(|x(n)| \right) \right) \right\rceil$$

- Multiplication : format de la sortie

$$m_z = m_x + m_y$$

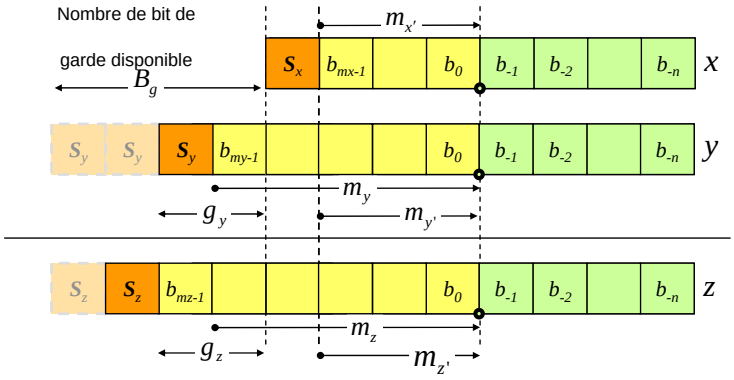
- Addition : choix d'un format commun

$$m_{ADD} = \max(m_x, m_y, m_z)$$

Notes :

Règles pour la position de la virgule

- Additionneur avec bit de garde
  - Changement de référentiel



m_y : position de la virgule de y par rapport au bit le plus significatif de y

$m_{y'}$: position de la virgule de y dans le référentiel commun

$$m_{y'} = m_y - g_y$$

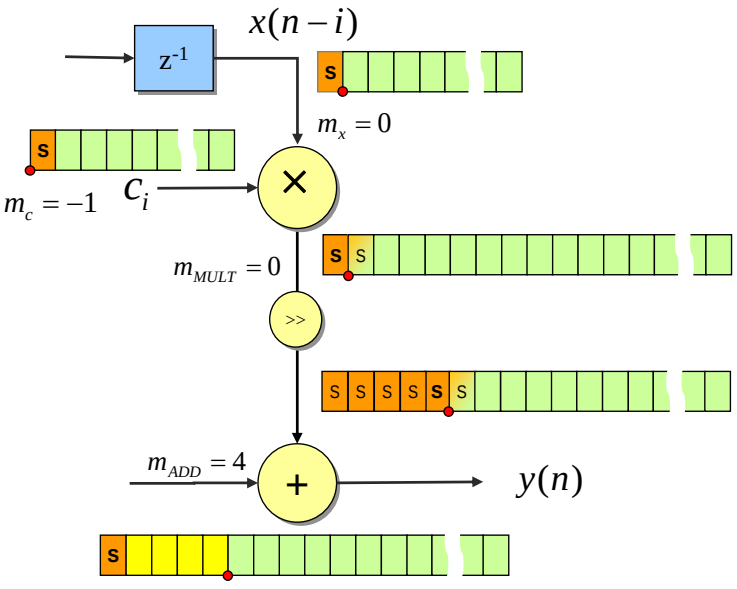
$$m_{z'} = m_z - g_z$$

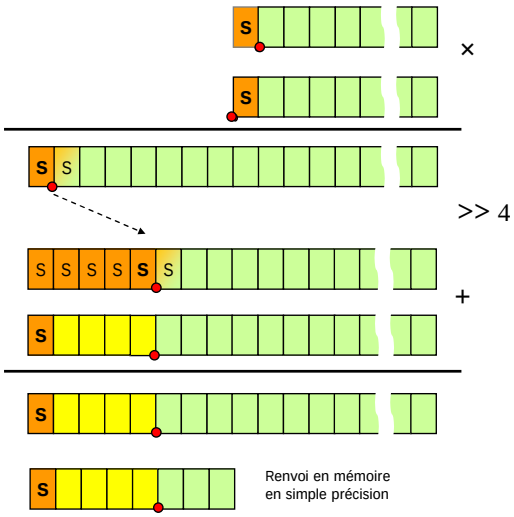
$$m_{ADD} = \max(m_x - g_x, m_y - g_y, m_z - B_g)$$

Notes :

Cas du filtre FIR

- Architecture sans bit de garde

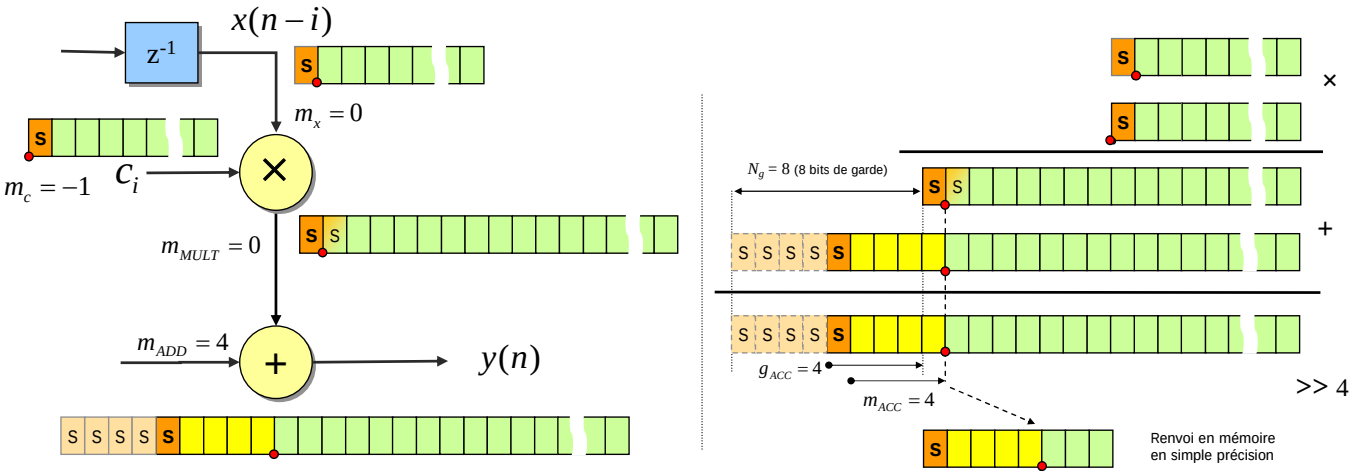




Notes :

Cas du filtre FIR

- Architecture avec 8 bits de garde



Notes :

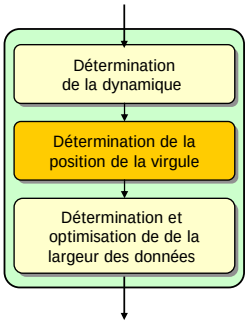
Notes

Notes :

II. Arithmétique virgule fixe

Notes :

Optimisation de la largeur

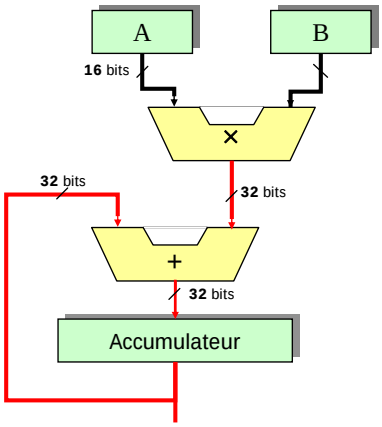


- Implantation logicielle
  - Architectures traditionnelles : opérateurs de largeur unique
  - Architectures évoluées : différents types de données sont supportés

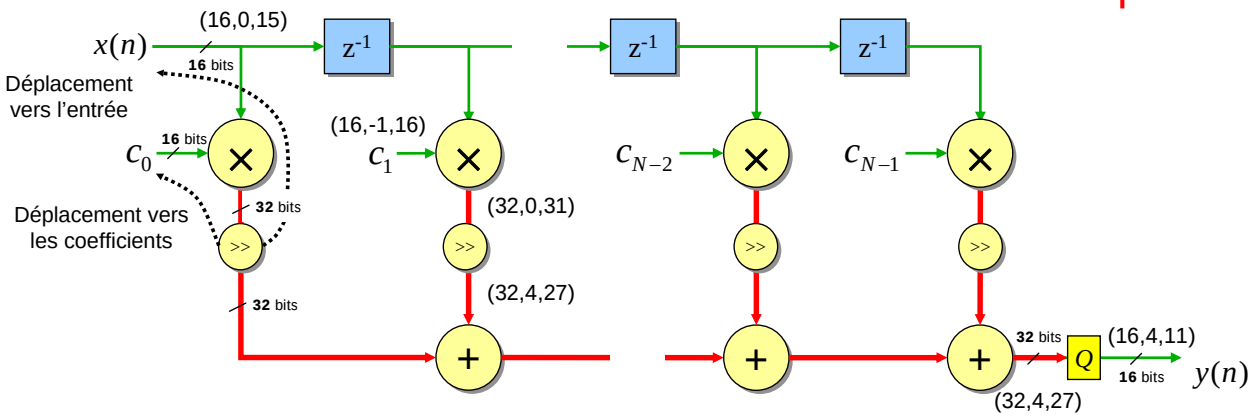
- Implantation matérielle
  - Objectif : optimiser la largeur des opérateurs

Notes :

Cas du filtre FIR

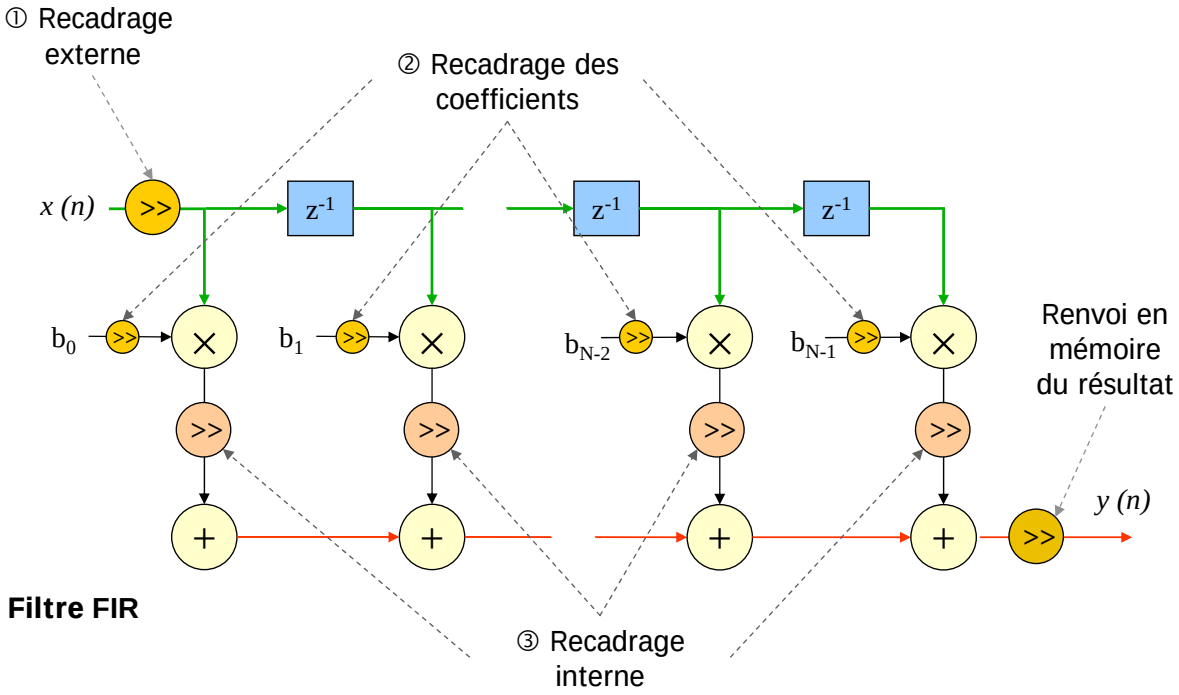


- Architecture double précision (sans bit de garde)



Notes :

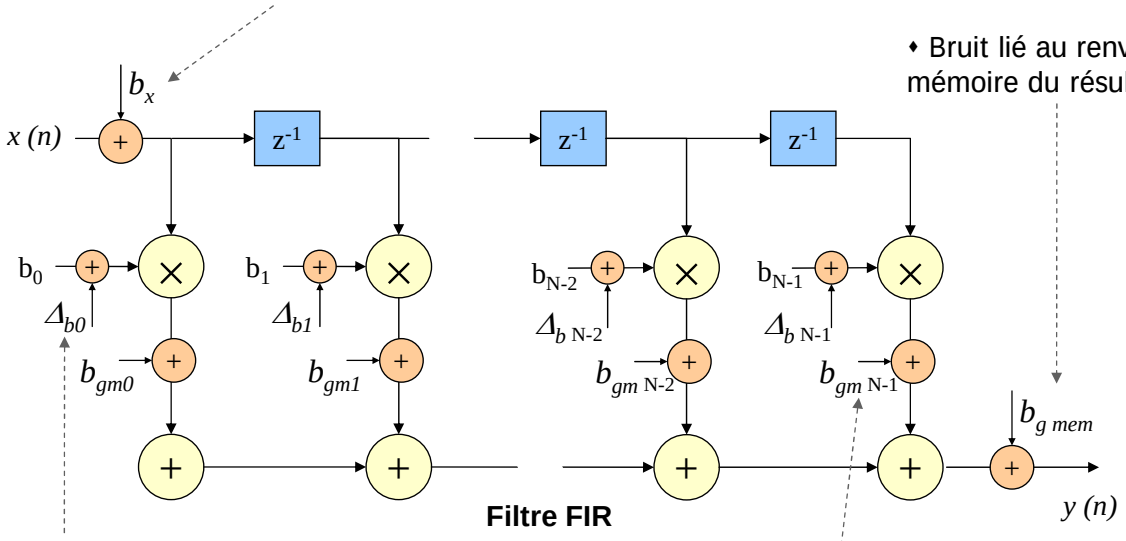
Recadrage des données dans un FIR



Notes :

Sources de bruit dans un FIR

- Bruit de quantification associé à l'entrée
- Bruit lié au recadrage externe ①



Filtre FIR

- Bruit lié au renvoi en mémoire du résultat

- Biais lié au codage des coefficients ②

- Bruit lié au recadrage interne ③

Notes :

Filtre FIR : code C virgule fixe

```
int Input[M] = {-809, -6180, -1570, ...}    /* Signal x, codage (16,0,15)*/
int c[N] = {13107, 14336, 15565, ..., 13107, 14336}; /* Coefficients (16,-1,16)  */
int main()
{
 int x[N]; y[M];
 long  acc;
 int i,j;
  for(i=0; i<N; i++){x[i] = 0;}      /* Initialisation des variables internes du filtre */
  for(j=0;  j<M; j++)         /* Filtrage du vecteur d'entree input */
  {
    x[0] = Input[j];
    acc = (long)(x[0]*c[0]) >> 4;

    for(i=N-1; i>0; i--)
    {
       acc = acc + ((long)(x[i]*c[i]) >> 4);        /* Calcul d'une cellule du filtre */
        x[i] = x[i-1];                              /* Vieillissement des variables internes */
    }
       y[j] = (int)(acc>>16);
  }
}
```

Le signal d'entrée et les coefficients sont spécifiés au niveau du code C en entiers sur 16 bits (absence de type virgule fixe en C) :

- l'entier représentant *Input* (16,0,15) est obtenu en multipliant *Input* par 2^{15}
- l'entier représentant c (16,-1,16) est obtenu en multipliant c par 2^{16}

Recadrage de la sortie de la multiplication : changement de format : (32,0,31) ⇨ (32,4,30)

Réduction de la largeur de la variable 32 bits ⇨ 16 bits
Récupération des 16 bits les plus significatifs de la donnée *(l 'opération de cast sur acc permet de récupérer les bits de poids faible uniquement)*

Notes :

II. Arithmétique virgule fixe

Notes :

Références

- O. Sentieys **Processeurs de traitement du signal (DSP) : état de l'art, applications, évolution et perspectives**, École thématique du CNRS, Seix (Ariège), 20-23 novembre 2000.
- P. Laspley, J. Bier, E. Lee, **DSP Processor Fundamentals**, IEEE Press, 1996, ISBN 0-7803-3405-1
- G. Baudoin, F. Virollau **Les DSP, Famille TMS320C54x, Développement d 'applications**, 2000, Dunod, ISBN 2-10-004646-2
- [ICE97] B. McClean ICE, "**Status 1997: A Report on the Integrated Circuit Industry**", Integrated Circuit Engineering Corporation (ICE), Scottsdale, 1997
- [Bier97] J. Bier, P. Lapsley & G. Blalock, "**Choosing a DSP Processor**", Berkeley Design Technology (BDT), 1997.
- [Lapsley96] P. Lapsley and G. Blalock, **"How to Estimate DSP Processor Performance**", IEEE Spectrum, July 1996.
- [Ropers99] A. Ropers and al., "**DSPstone : TI C54x**" Report IISPS AAchen University of Technology, 1999.

Notes :