Processeurs de traitement du signal (PTS)

Support de cours EII2

Octobre 2002

Daniel Menard

Daniel.Menard@enssat.fr

PLAN

I. Traitement Numérique du Signal
   1. Chaîne de traitement
   2. Caractéristiques du TNS

II. Généralités sur les DSP
   1. Solutions architecturales
   2. Marché des DSP
   3. Caractéristiques des DSP
   4. Les différentes générations

PLAN

III. Architecture des DSP conventionnels

1. Unité de traitement
2. Unité de mémorisation
3. Unité de contrôle
4. Unité de communication

IV. Panorama

1. Vue générale
2. TMS320C5x
3. TMS320C54x
4. TMS320C3x

PLAN

V. Implantation des algorithmes sur DSP
   1. Outils de développement
   2. Outils de développement pour C50 et C30
   3. Compilateurs C

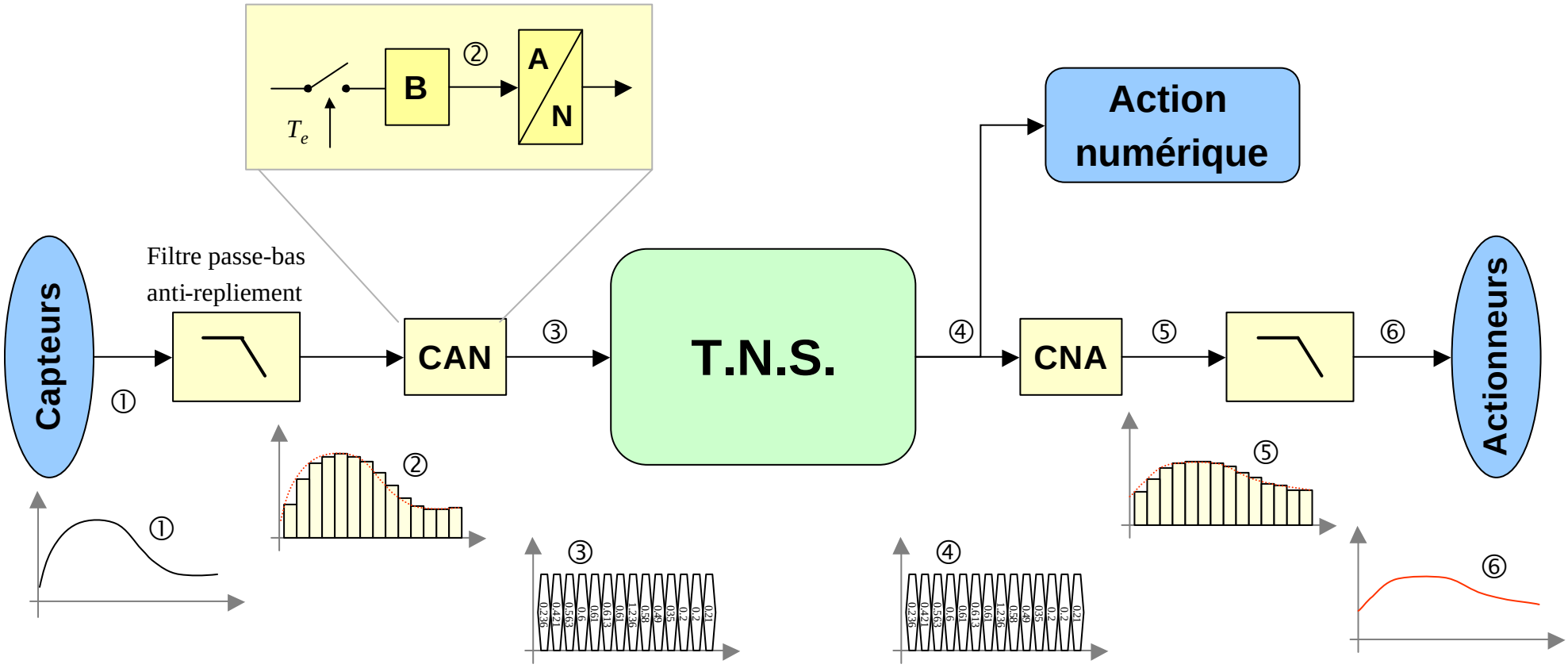
VI. Traitement en virgule fixe
   1. Arithmétique virgule fixe
   2. Détermination du domaine de définition
   3. Codage des données

VII. Conclusion

I. Traitement Numérique du Signal

1. Chaîne de traitement
2. Caractéristiques du TNS

Chaîne de traitement



Traitement numérique

- Avantages:
  - Stabilité des performances : pas de dérives en température ou dans le temps
  - Précision : garantie par le nombre de bits et elle est déterminée à priori
  - Flexibilité : changements aisés des paramètres des algorithmes
  - Intégration : système VLSI
  - Production : reproductibilité, pas de réglages

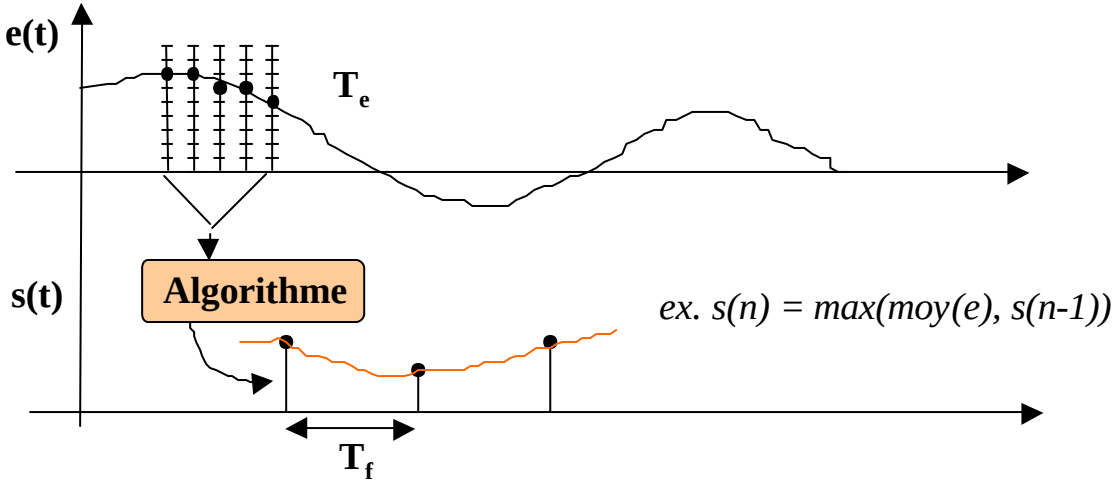
- Inconvénients:
  - Coût : élevé pour des applications relativement simples
  - Vitesse de traitement proportionnelle à la bande passante du signal
  - Complexité : réalisation matérielle et logicielle

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

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



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:

Notes:

II. Généralités sur les DSP

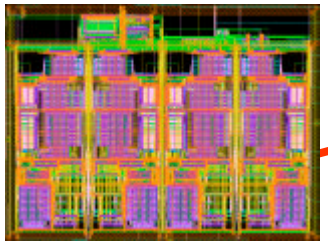
1. Solutions architecturales
2. Marché des DSP
3. Caractéristiques des DSP
4. Les différentes générations

Solutions architecturales

- Compromis performance - flexibilité

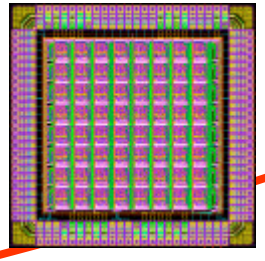
Efficacité énergétique :

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

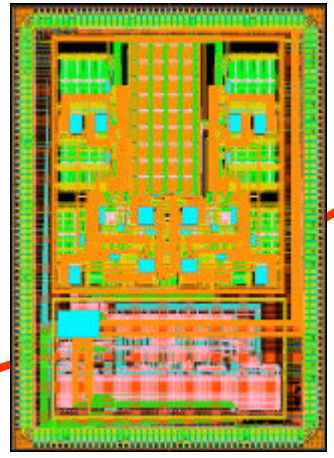


100-1000 MOPS/mW

ASIC



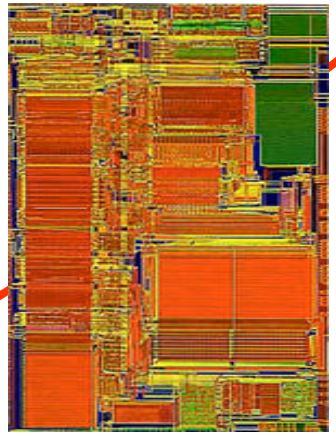
Embedded FPGA



Pleiades
10-50 MOPS/mW

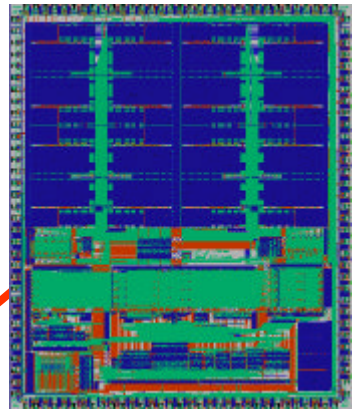
Reconfigurable Processor

ASIP



2 V DSP
3 MOPS/mW

DSP



Embedded Processor

SA110
0.4 MIPS/mW

Alpha
0.007 MIPS/mW

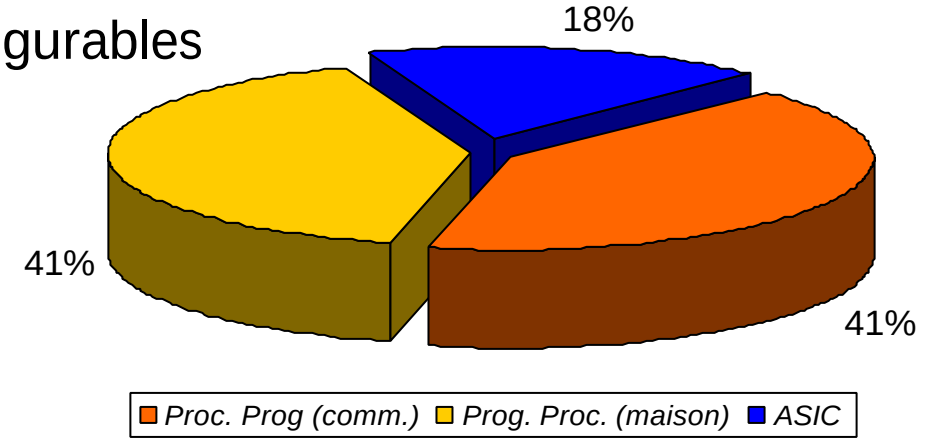
Flexibility

100 — 10 — 1

E_E : Efficiency : MIPS / Watt

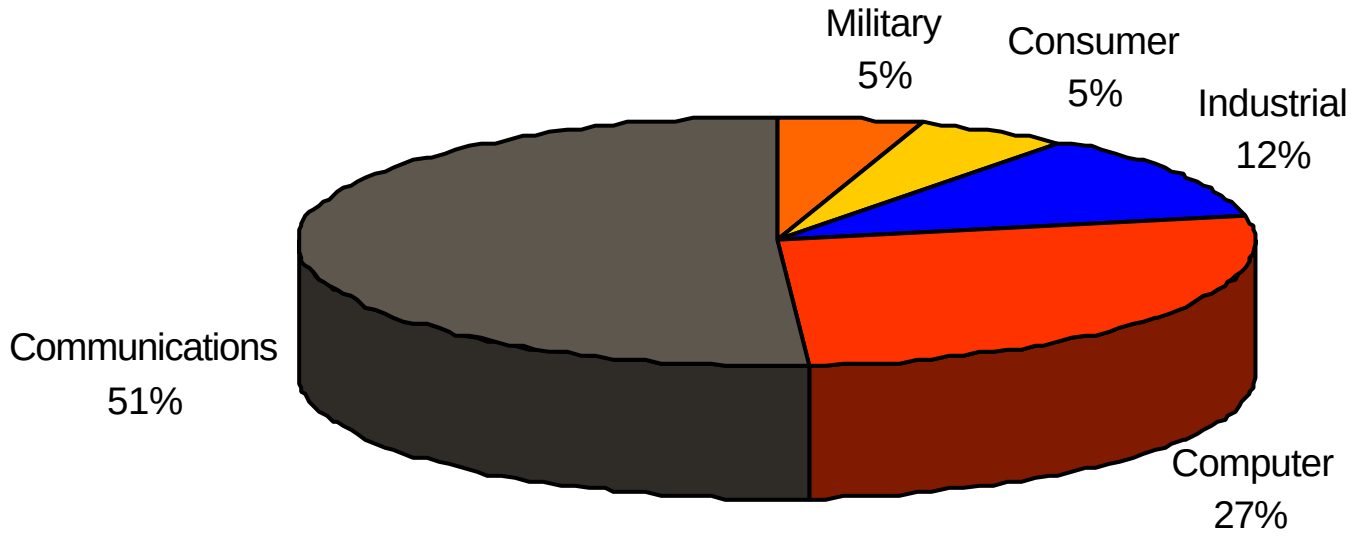
Solutions architecturales

- 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



Le marché des processeurs DSP

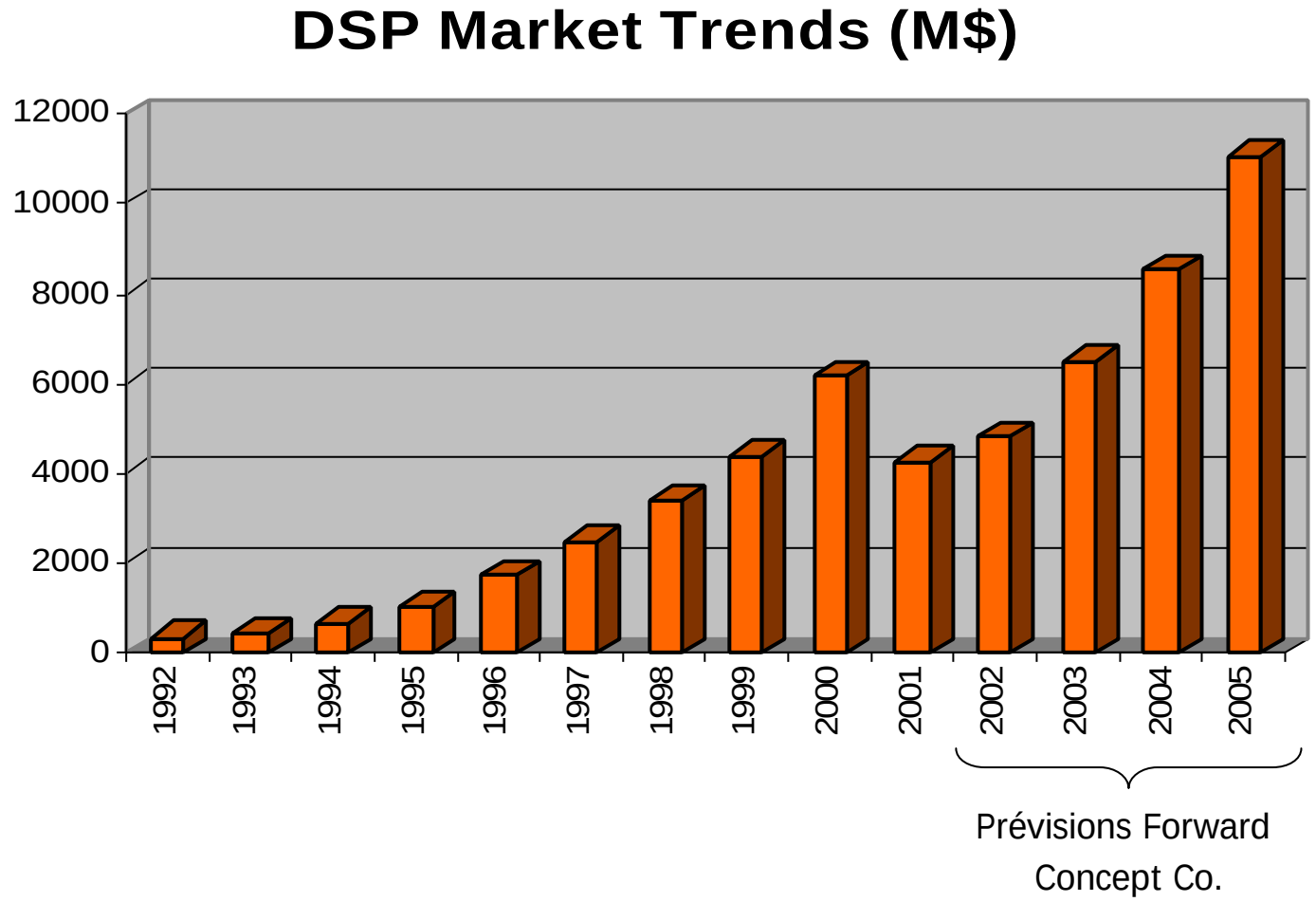
- Les communications dominent
- Ordinateurs sont seconds (e.g. asservissement des moteurs pour les disques durs, compression d'image, reconnaissance de la parole, conversion texte-vers-parole)



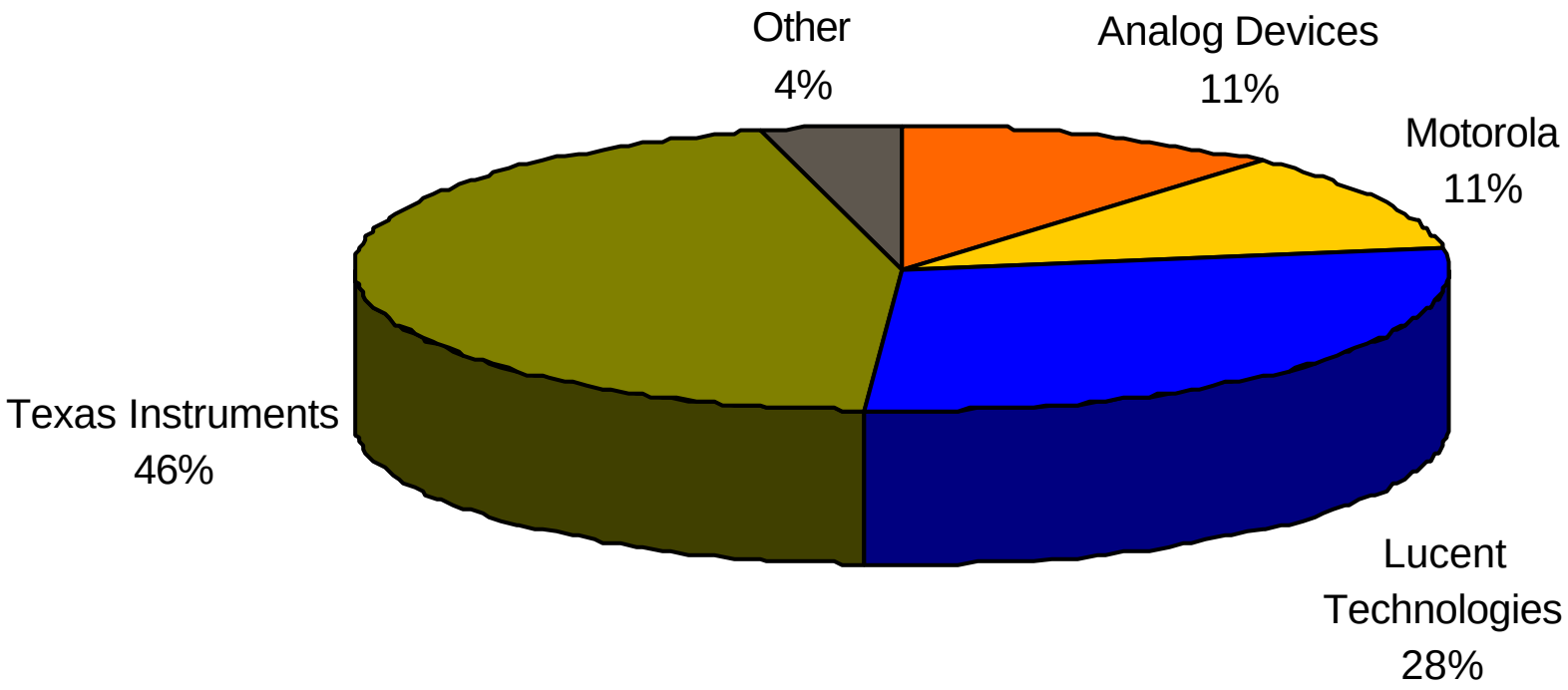
[ICE97] *B. McCleanICE, "Status 1997: A Report on the Integrated Circuit Industry", Integrated Circuit Engineering Corporation (ICE), Scottsdale, 1997*

Le marché des processeurs DSP

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



Le marché des processeurs DSP

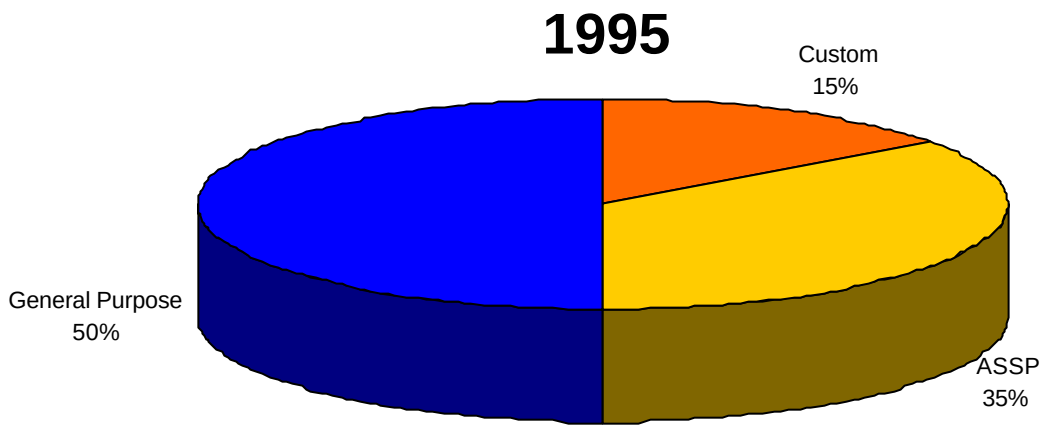


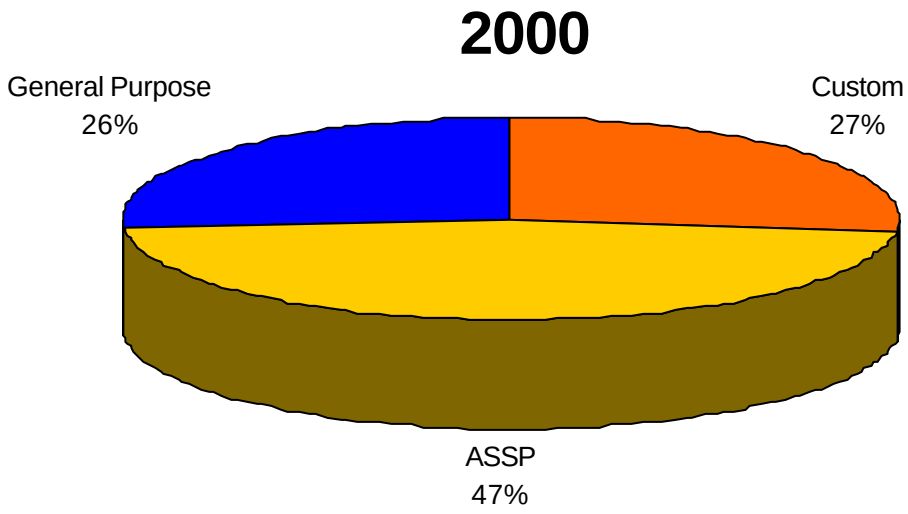
Worldwide Sales of Single-Chip DSPs in 1996

Tendances du marché des *DSP*

- *Application-specific standard products* (ASSPs) dominent

- Solutions à intégration de cœurs de DSP
  - e.g. DSP Group Pine and OAK DSP-cores for integration into Atmel's cell and gate array library





Caractéristiques des DSP

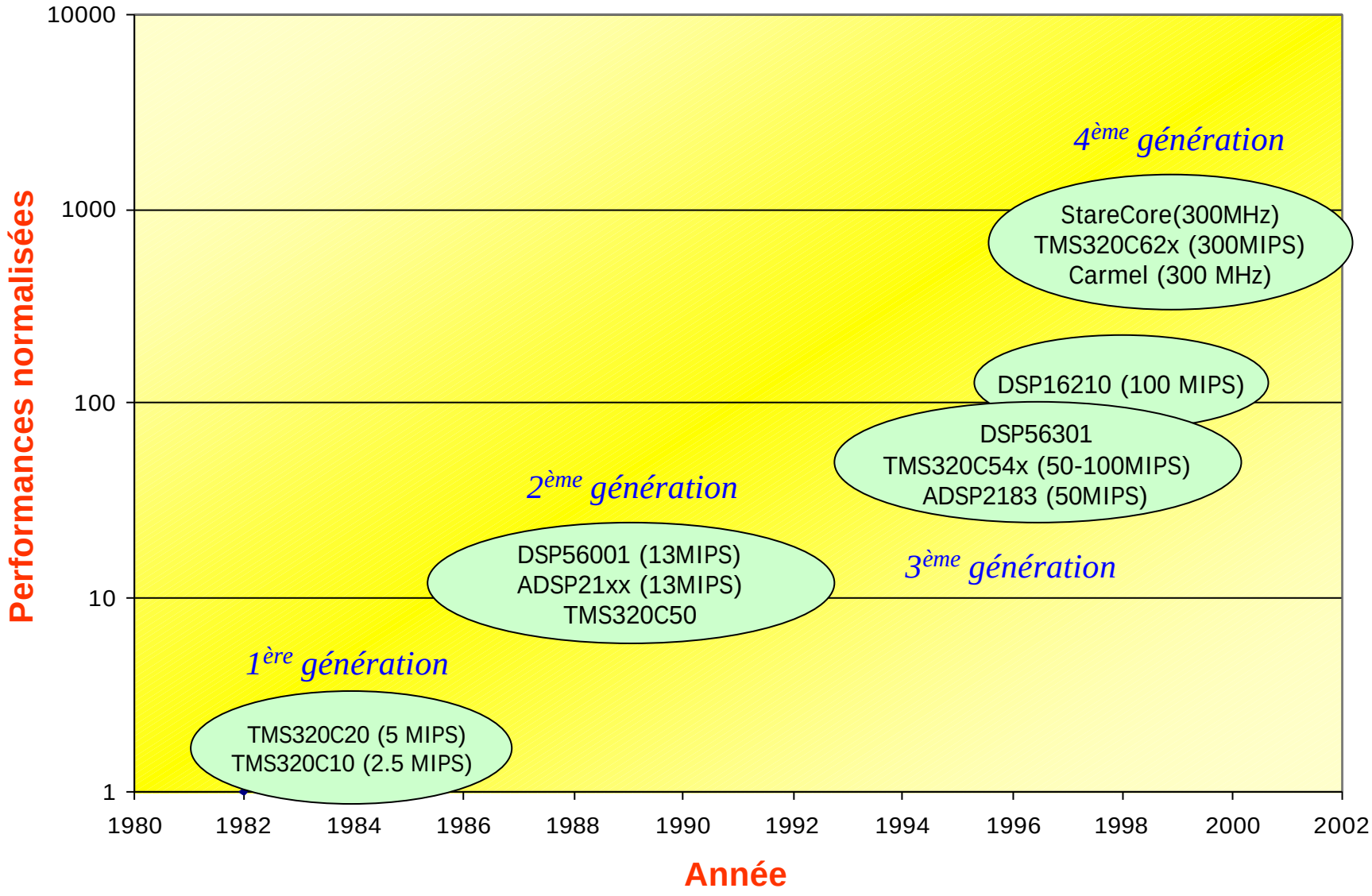
Propriétés du traitement numérique du signal	Conséquences sur les architectures
Calculs intensifs et répétitifs	♦ Fonctionnement pipeline ♦ Architecture Harvard ♦ Structures de contrôle évoluées
Primitives simples	♦ Unités de traitement spécialisées câblées ♦ Gestion complexe de l'adressage

Le tout dans un environnement temps réel

Caractéristiques des DSP

- Applications embarquées et « grand public »
  - Très faible coût
    - le coût est proportionnel à la surface du circuit
  - Faible consommation
    - $Pmoy = \alpha . C . V_{dd}^{2} . F$
      - 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

Les différentes générations



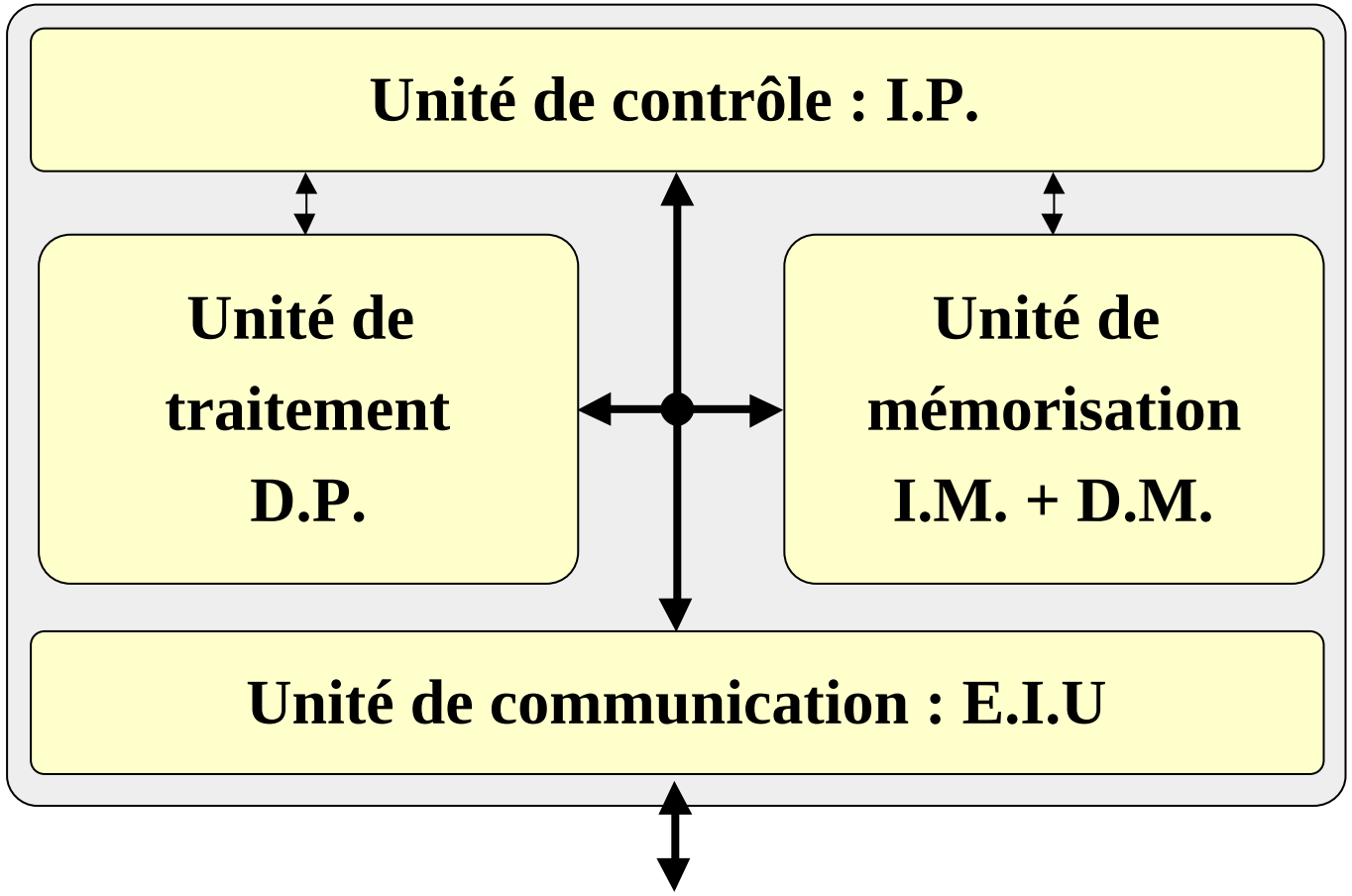
Notes:

Notes:

III. Architecture des DSP conventionnels

1. Unité de traitement
2. Unité de mémorisation
3. Unité de contrôle
4. Unité de communication

Modélisation d'un processeur



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

Diagramme d'état d'un IP

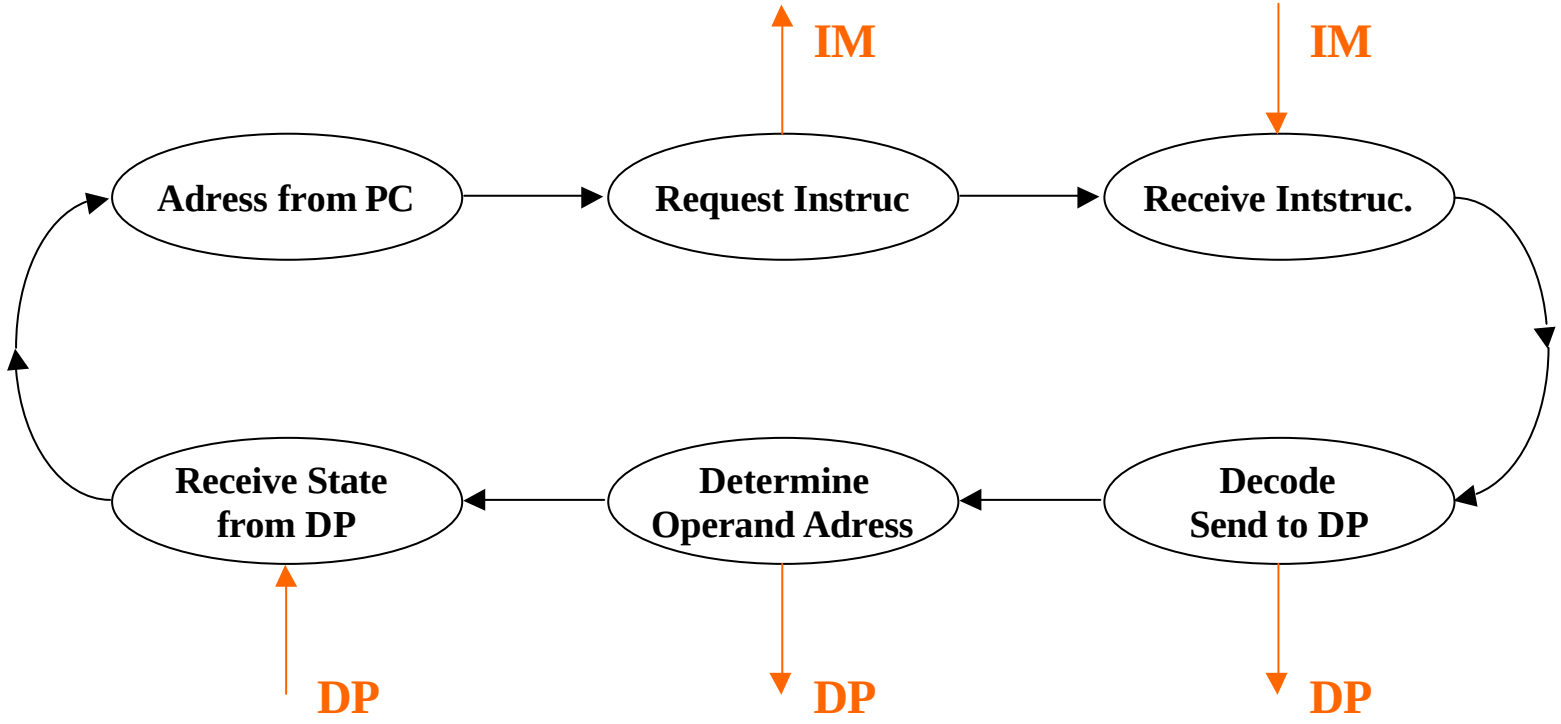
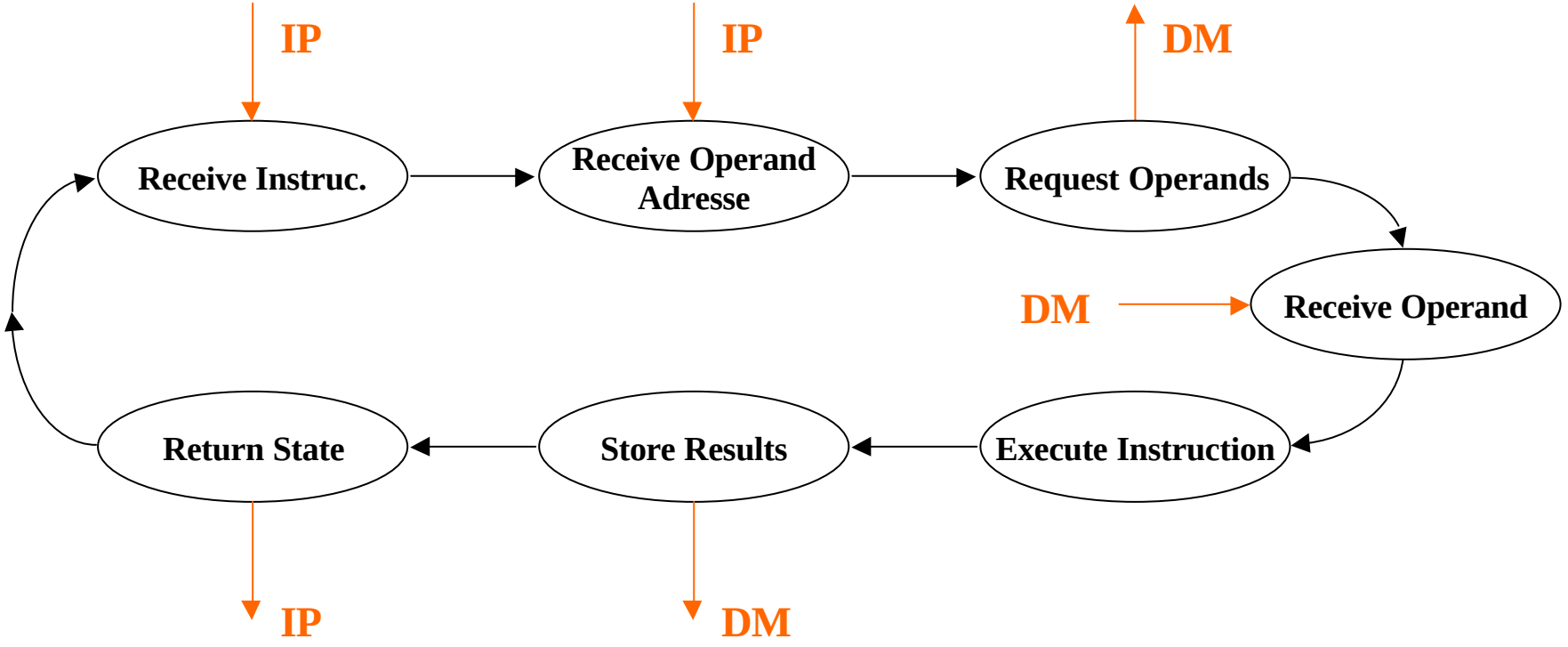


Diagramme d'état d'un DP



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
- 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

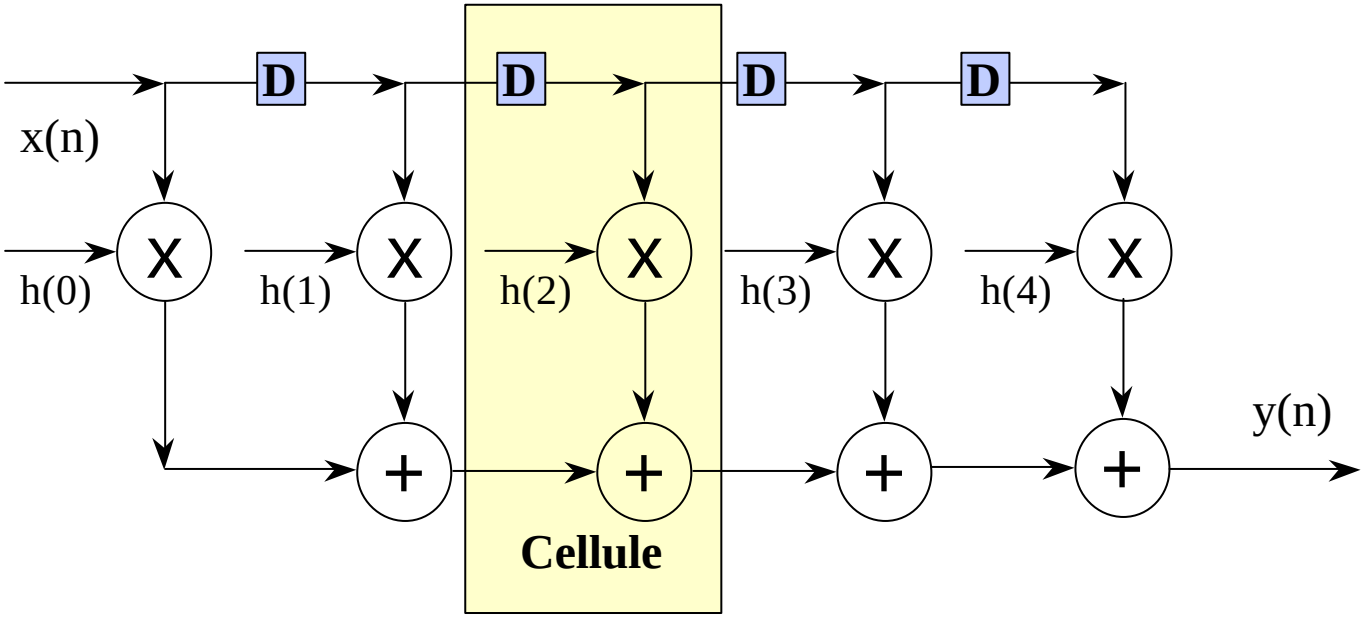
III. Architecture des DSP conventionnels

1. Unité de traitement

Exemple *Fil Rouge*

- Filtre numérique FIR sur N points

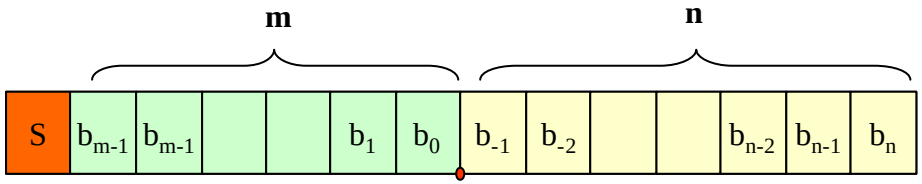
$$y(n) = \sum_{i=0}^{N-1} h(i).x(n-i) = x(n) * h(n)$$



⇨ Objectif : traitement d'une cellule par cycle

Représentation des données

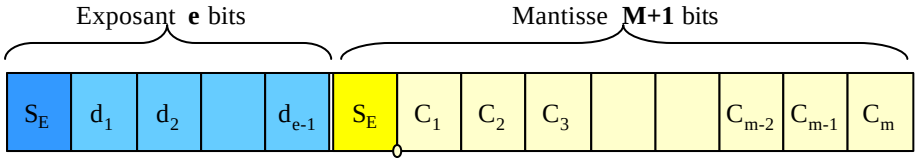
- Virgule fixe
  - Représentation : signe - partie entière - partie fractionnaire



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

  - Le format d'une donnée ne varie pas au cours du temps

- Virgule flottante
  - Représentation : exposant - mantisse



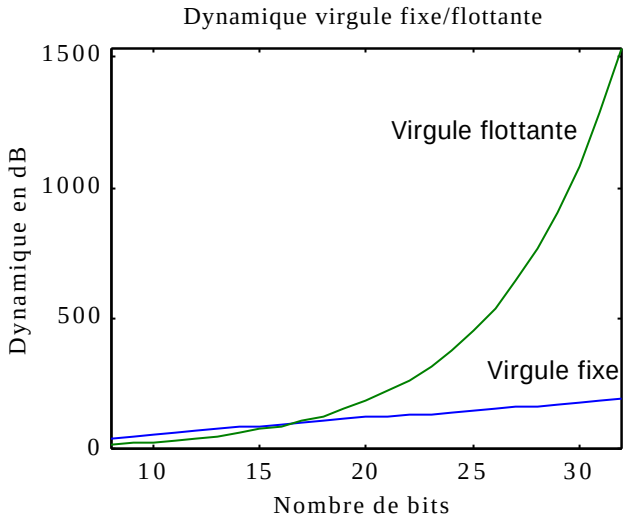
$$x = 2^u (-1)^{S_E} \left(\frac{1}{2} + \sum_{i=1}^{M} C_i 2^{i-1} \right)$$

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

Comparaison fixe - flottant

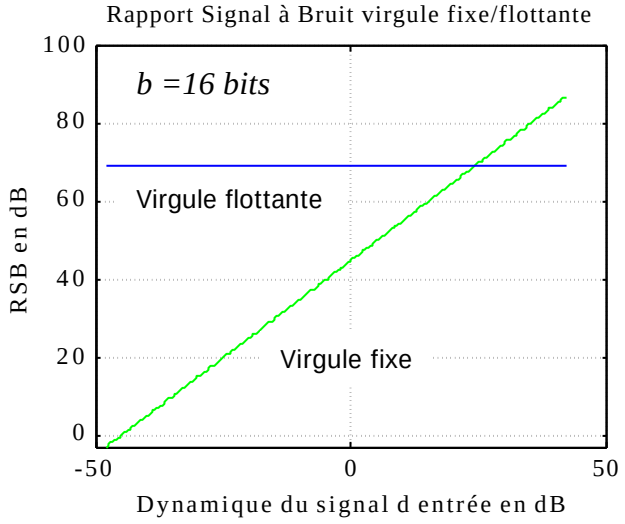
- Dynamique

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



- Rapport Signal à Bruit de Quantification

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



DSP 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

C50 50 ns: 233 FF (100 Pièces)
25 ns: 282 FF (100 Pièces)

DSP 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

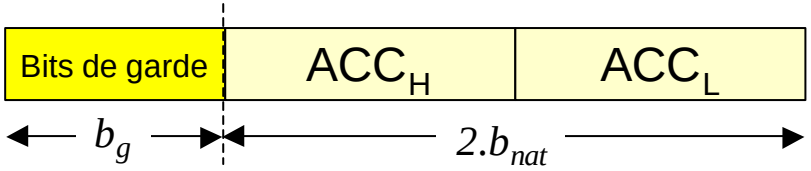
C40 50 ns: 1282 FF (100 Pièces)
33 ns: 1410 FF (100 Pièces)

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

Éléments de l'UT

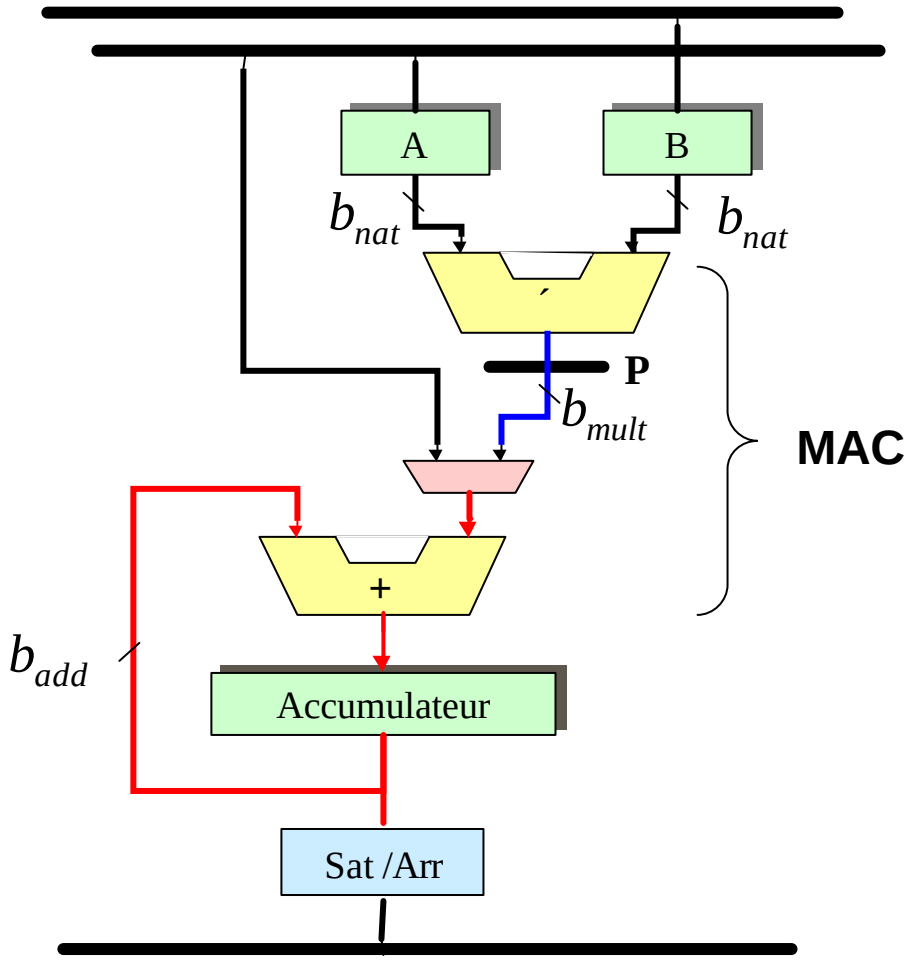
- Unités de saturation ou d'arrondi
- Unités spécifiques
  - Unité de manipulation de bit, Viterbi, ...
- Unités de stockage de l'UT
  - Registres opérandes
    - Stockage des opérandes sources ou des résultats intermédiaires
  - Registres d'accumulation
    - Stockage du résultat de l 'additionneur
    - 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 $b_{add} = b_{mult} + b_g$



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	$2\ b_{nat}$ $2\ b_{nat} + b_g$

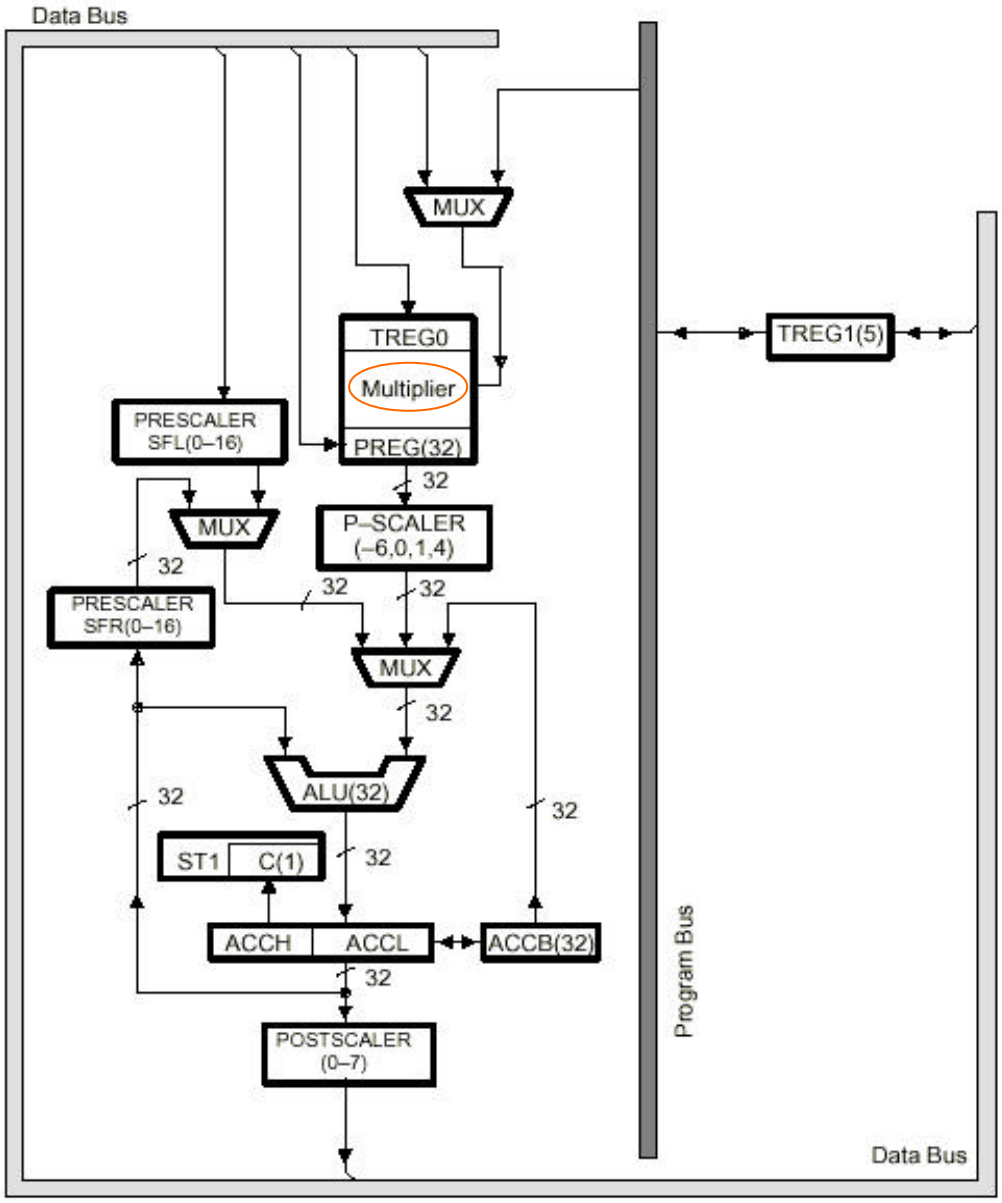


Interconnexions registres - opérateurs spécialisées

⇨ Structure hétérogène

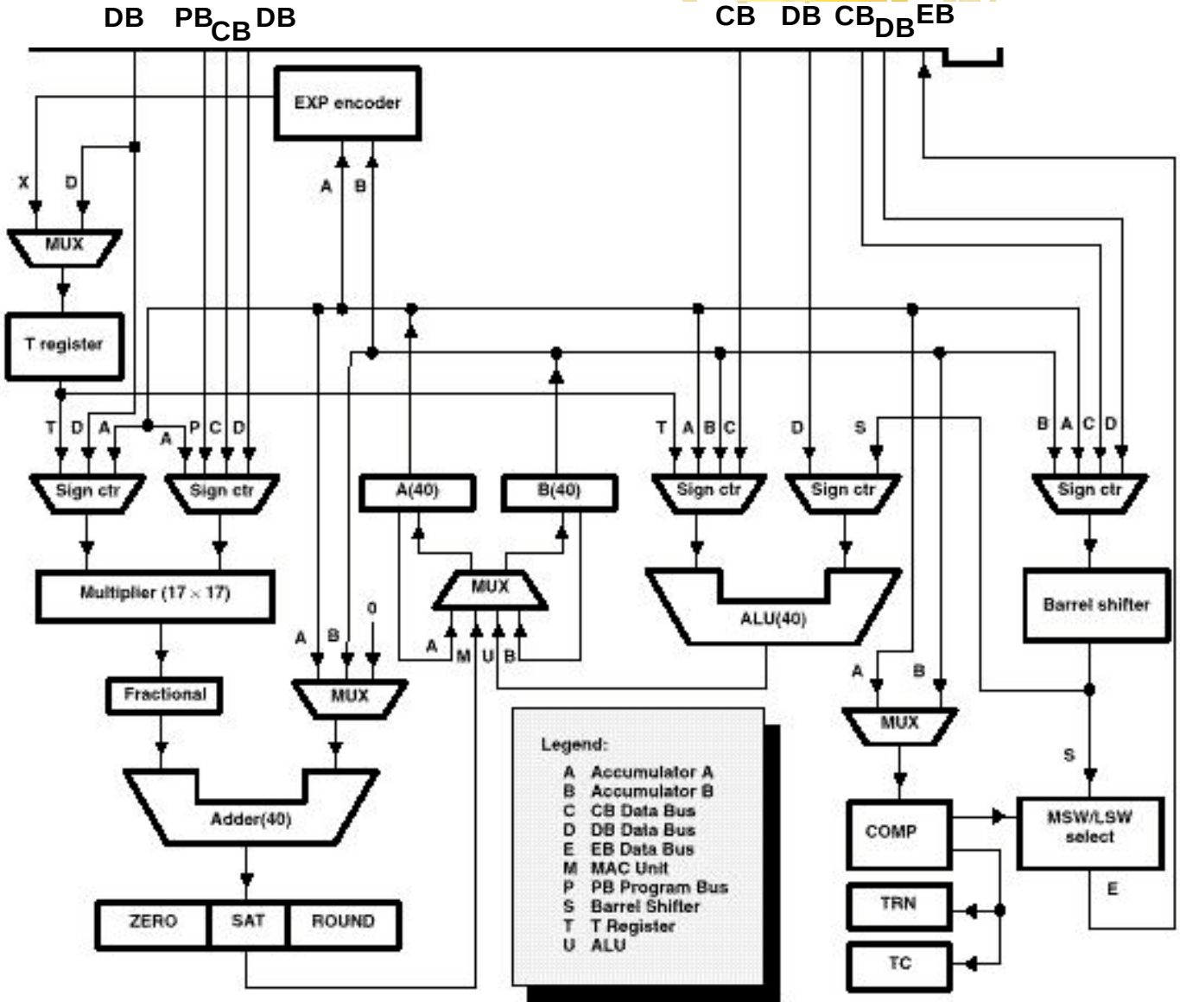
⇨ Bonnes performances en terme de consommation et de surface

Exemple : TMS320C50



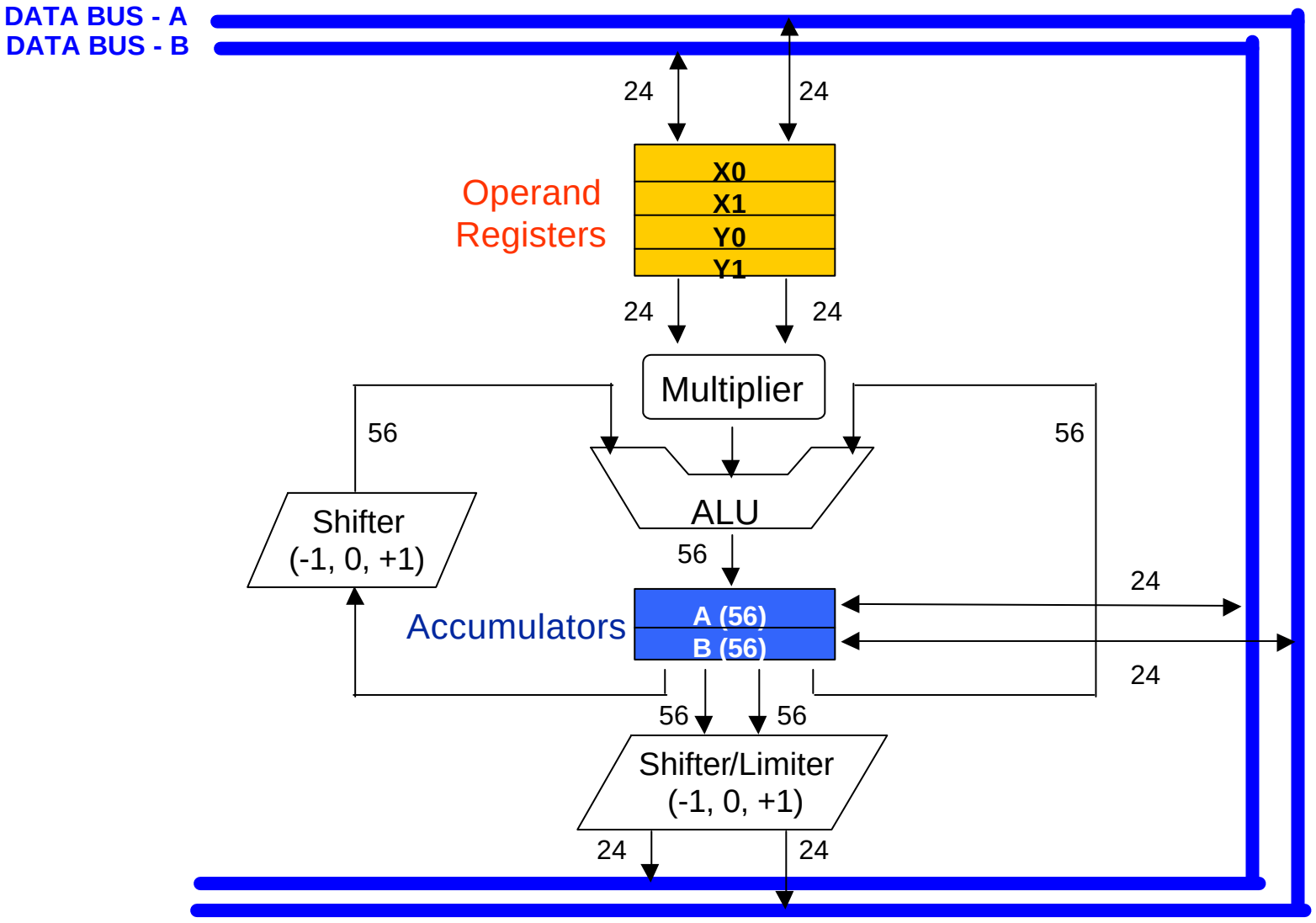
- 1 multiplieur 16*16 bits
  - opérande source 1 : TREG0
  - opérande source 2 : mémoire
  - opérande destination : PREG
- 1 ALU (32 bits)
- 1 registre d'accumulation 32 bits
- Plusieurs registres à décalage spécialisés
- 1 unité logique PLU supplémentaire pour les calculs sur les bits

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

Exemple : Motorola DSP 5600x



Notes:

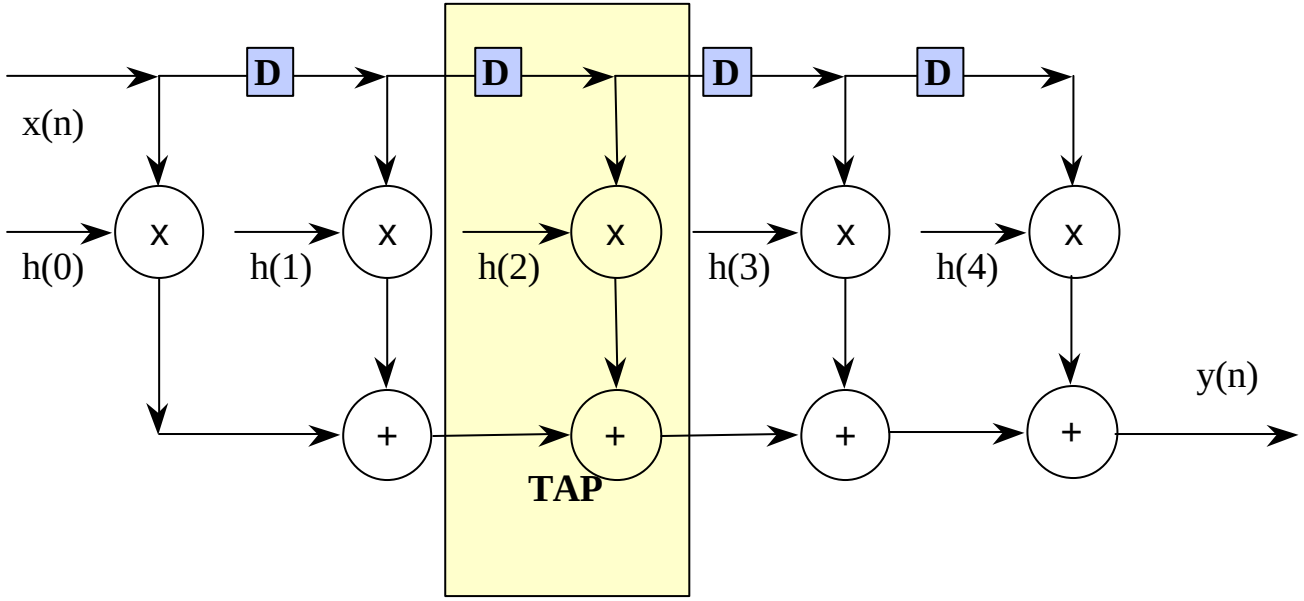
Notes:

III. Architecture des DSP conventionnels

2. Unité de mémorisation

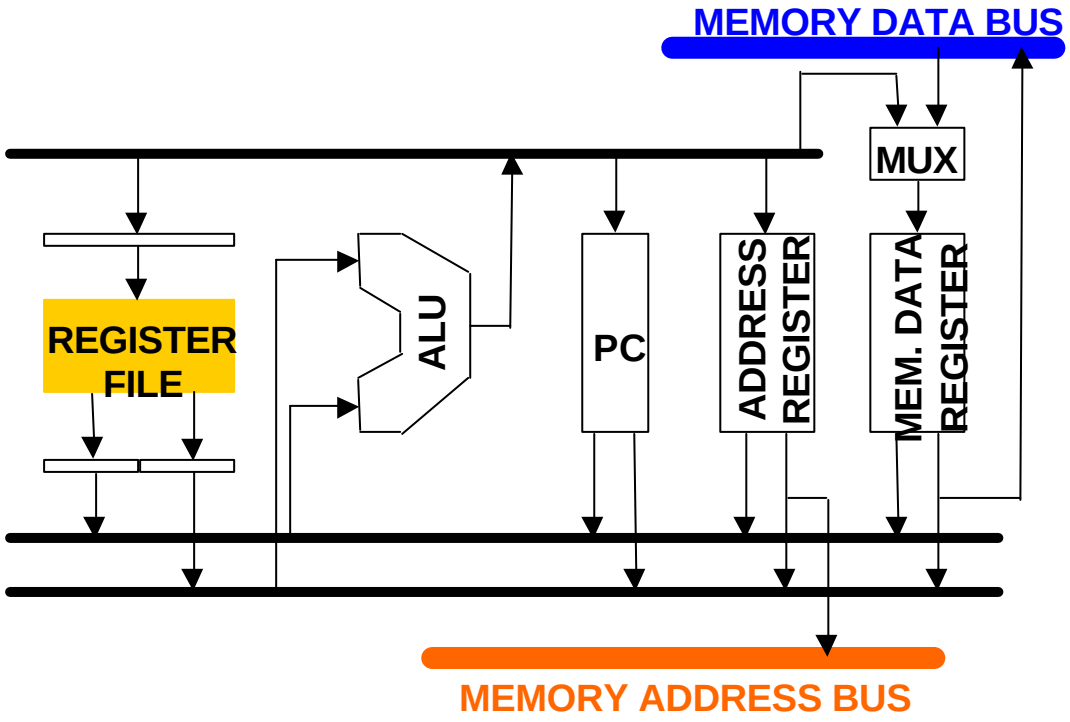
Exemple FIR *Fil Rouge*

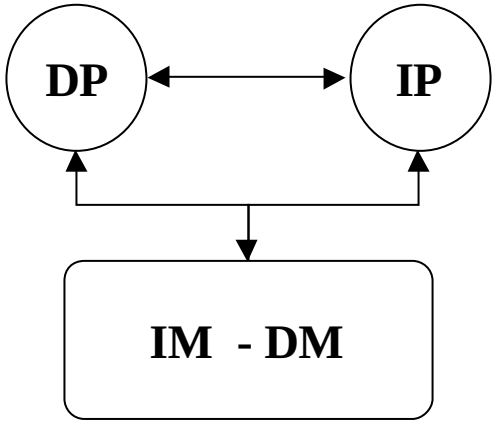
- Les différents accès à la mémoire:
  - Recherche de l 'instruction
  - Lecture de la donnée x_{n-k}
  - Lecture du coefficient h_k
  - Vieillissement des données $x_{n-k-1} = x_{n-k}$



Architecture Von Neumann

- Les processeurs RISC





FIR sur machine Von Neumann

- Problèmes :
  - Bande passante avec la mémoire
  - Code pour le contrôle et la gestion de l'adressage
  - Multiplication lente

```
loop:
   mov *r0,x0
   mov *r1,x1
   mpy x0,x1,a
   add a,b
   mov x1,*r2
   inc r0
   inc r1
   inc r2
   dec ctr
   tst ctr
   jnz loop
```

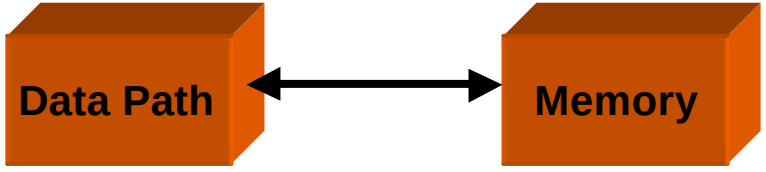
lecture des opérandes sources

opération MAC

vieillissement de l'échantillon

gestion des pointeurs d'adresse

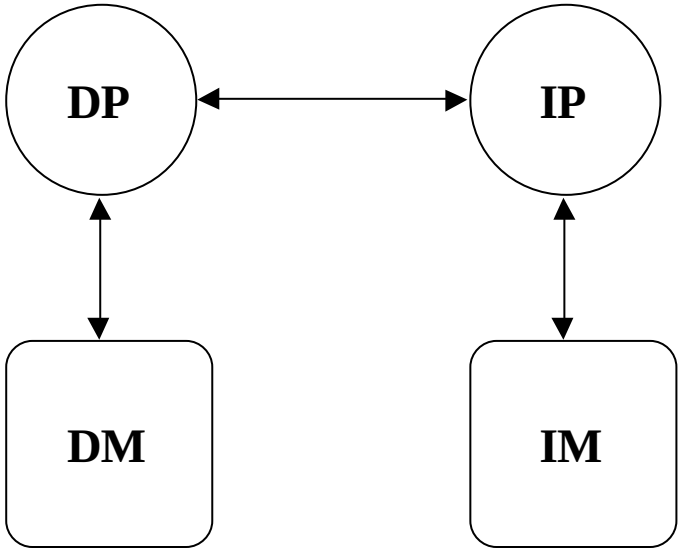
gestion de la boucle



Exécution en 15 à 20 cycles

Architecture Harvard de base

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



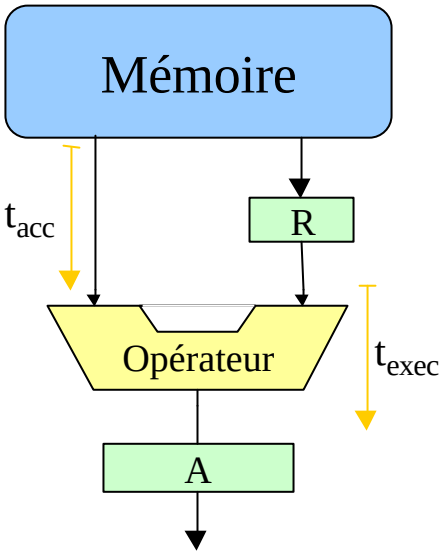
Ex: TMS320C10

Classification de E. Lee (1989)

Localisation des opérandes

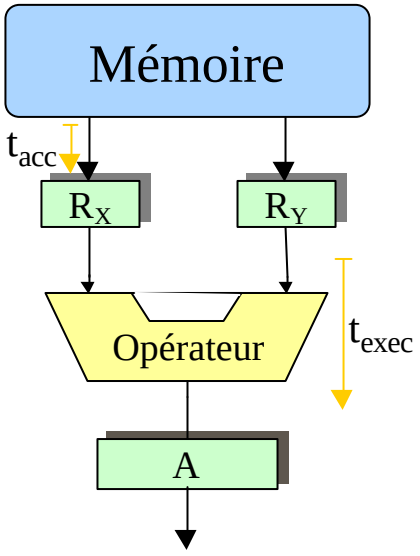
- Modèle registre-mémoire
  - Opérandes situées en mémoire et dans les registres
    - Temps d ’exécution de l ’instruction

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



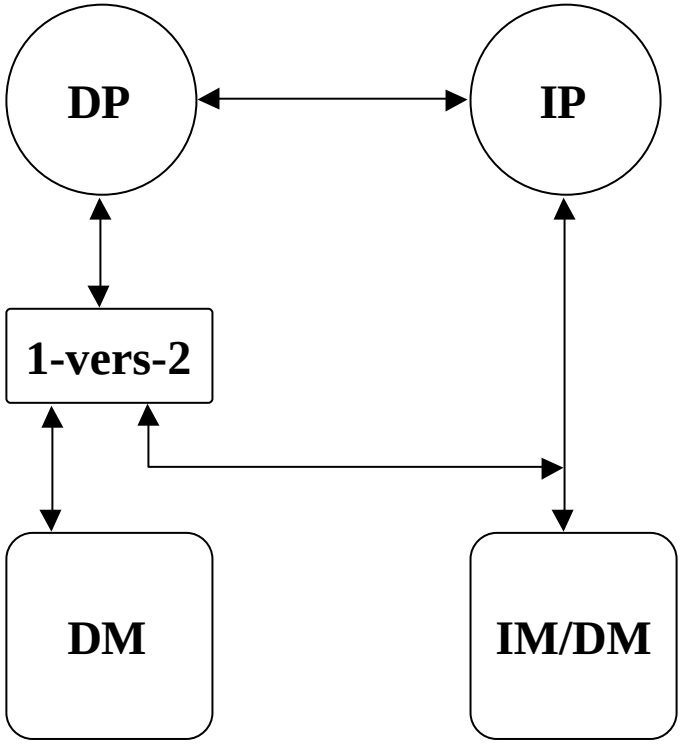
- Modèle « Load-Store »
  - Opérandes situées uniquement dans des registres
    - Temps d ’exécution de l ’instruction

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



Modification 1

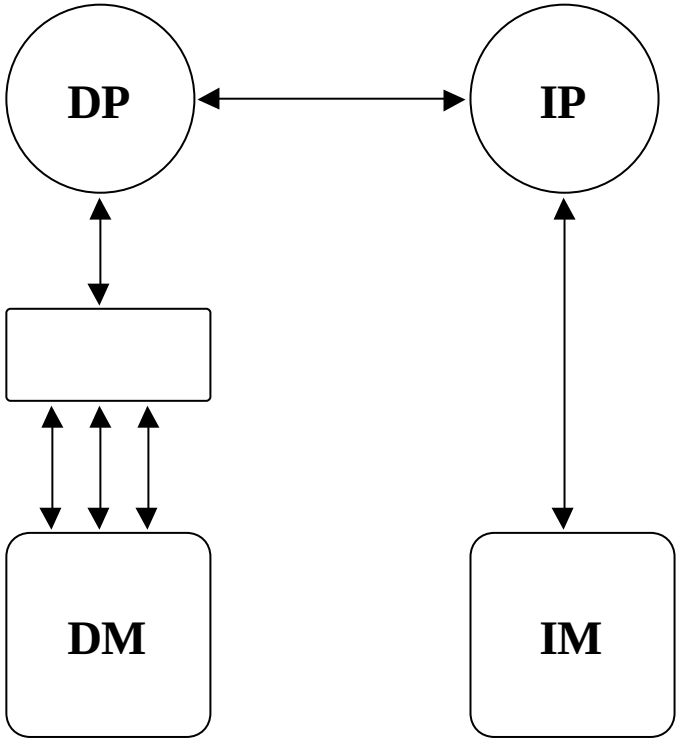
- Autorisation de mémorisation de données dans l'IM
- En un cycle si 2 accès mémoire par cycle ($t_{acc} = 1/2.t_{inst}$) :
  - *fetch* de l'instruction
  - *fetch* deux opérandes de la mémoire
  - exécution d'un MAC
  - écriture du résultat en I/O ou mémoire



AT&T DSP32 et DSP32C

Modification 2

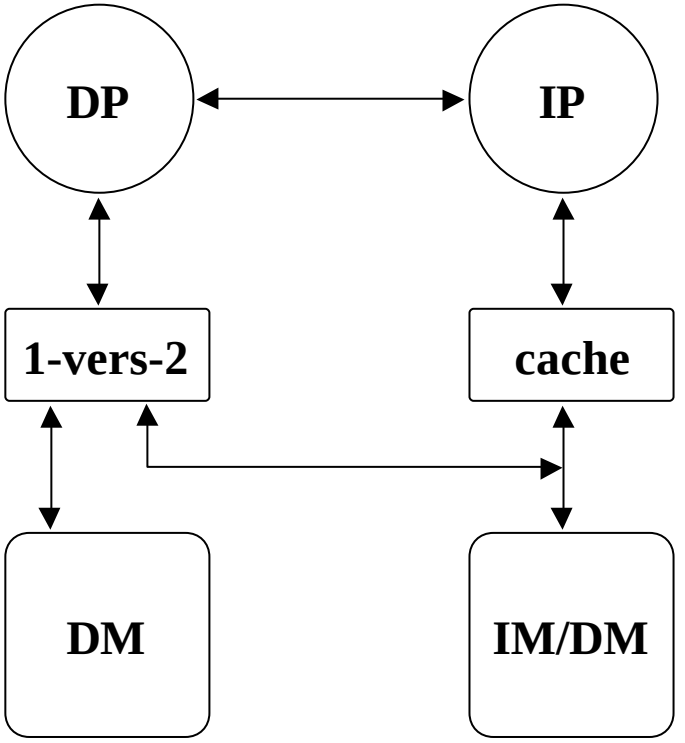
- DM est une mémoire multi-ports, plusieurs accès aux données par cycle
- Utilisable pour des mémoires internes au CI



Fujitsu MB86232 (3 ports en mémoire interne)

Modification 3

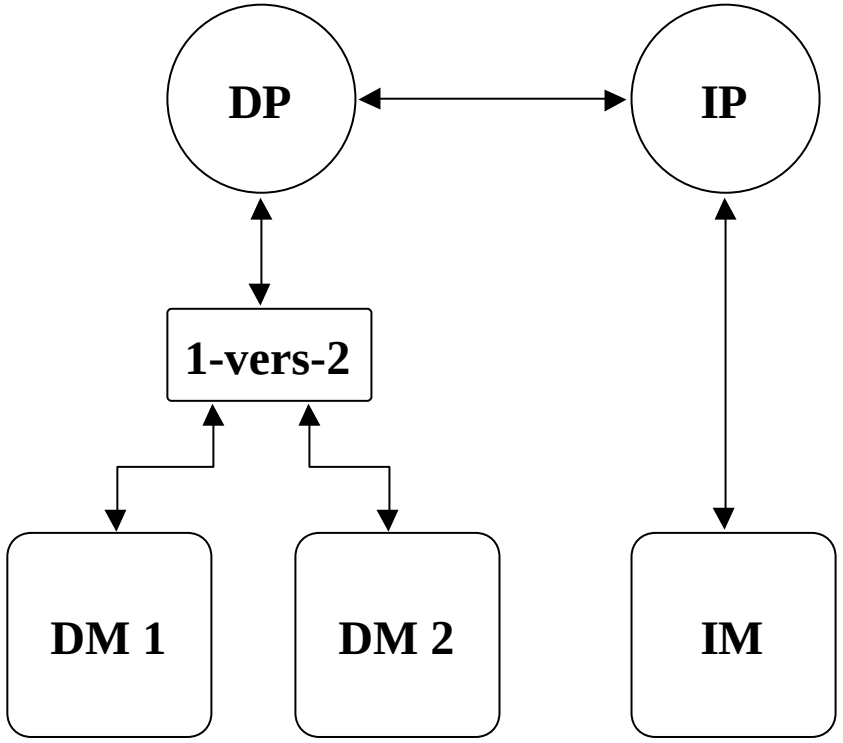
- Cache pour charger les instructions fréquentes
- Évite les conflits d'accès données et instructions de la modification 1



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

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:

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

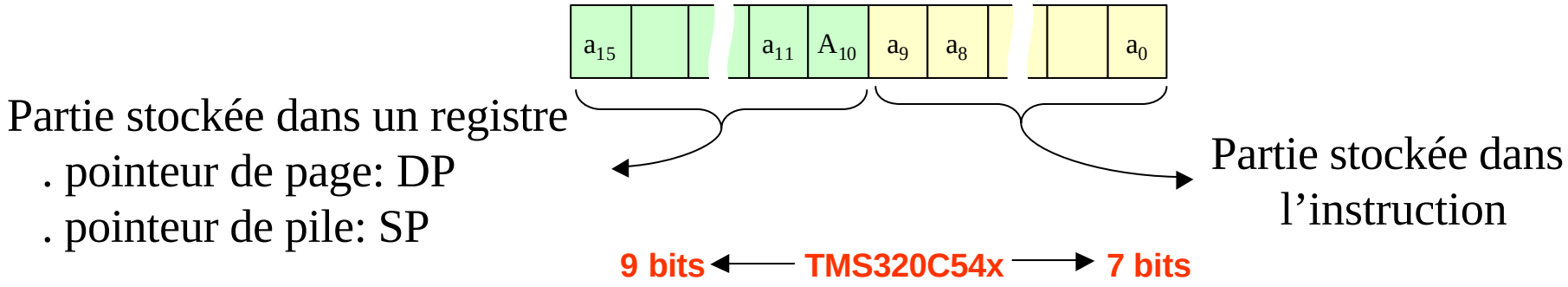
Modes d'adressage

- 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 :



Modes d'adressage

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

LD *AR1, A — addr = AR1 (A ⇦ *AR1)

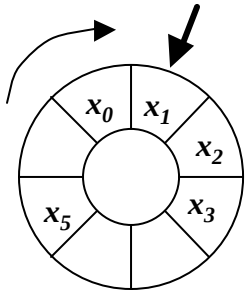
  - Possibilités de post modifications :
    - linéaire : AR:= AR ± 1 — LD *AR1+, A

addr = AR1
AR1 = AR1 + 1

    - indexé : AR:= AR ± MR — LD *AR1+0, A

addr = AR1
AR1 = AR1 + AR0

      - MR: registre d'index



    - modulo : $(AR:= AR \pm 1)_N$ — LD *AR1+% ,A

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

(BK) specifies the size of the circular buffer.

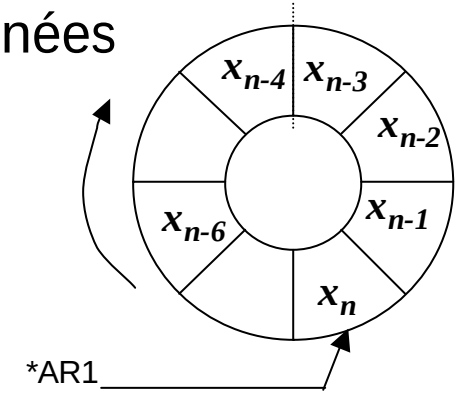
    - bit-reverse : FFT — LD *AR1+0B ,A

addr = AR1
AR1 = bitrev(AR1 + AR0)

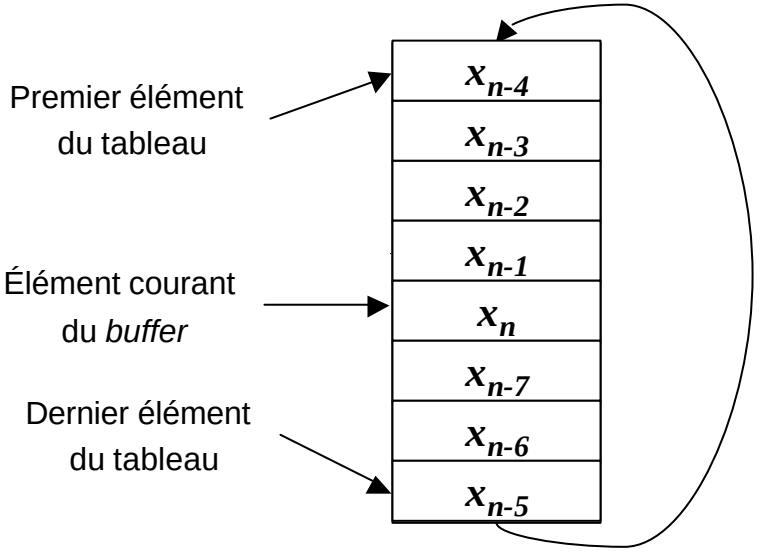
After access, AR0 is added to ARx with reverse carry (rc) propagation.

Buffer circulaire et *bit-reverse*

- *Buffer* circulaire :
  - Vieillissement automatique des données



N=8

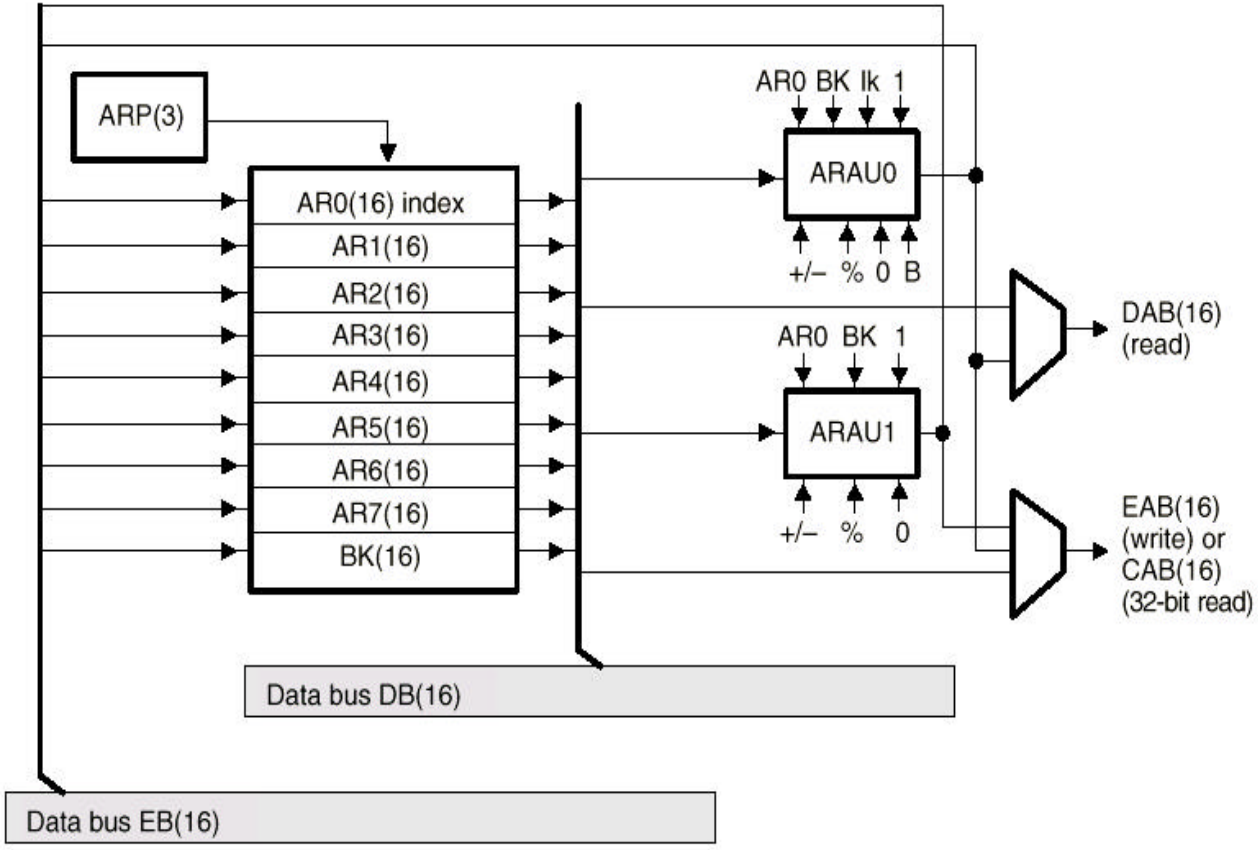


- Bit-Reverse :
  - Adressage des données en entrée ou en sortie de la FFT
  - Attention à l 'adresse de début du tableau : xxxx0000

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

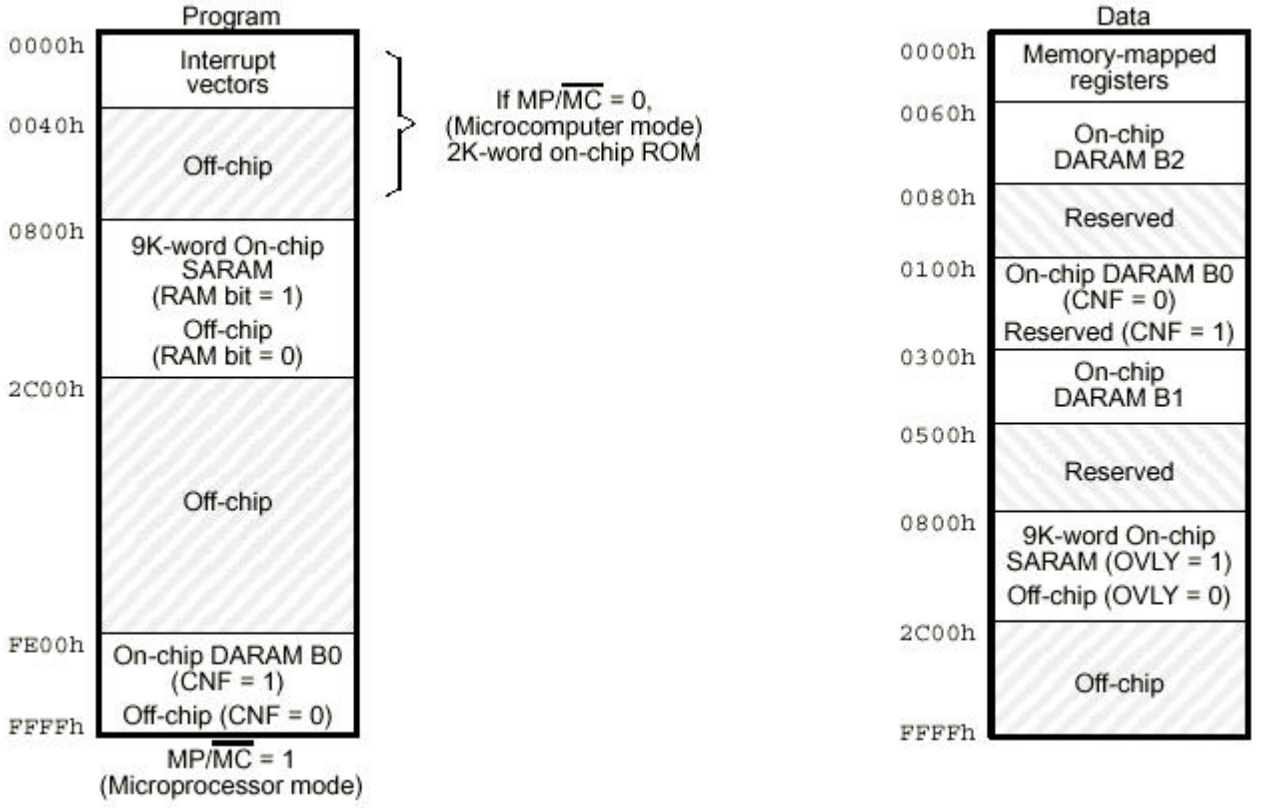
Unité de gestion des adresses

- 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

Espace mémoire du C50



- 64K * 8 bits d'espace programme, donnée et IO
- des registres mappés en mémoire
- mémoires on-chip ou externes suivant la configuration choisie

Espace mémoire du C50

- Mémoire programme:
  - Suivant la broche MP/MC :
    - mode micro-processeur: pas de *on-chip ROM*
    - mode micro-contrôleur: 2k de *on-chip ROM*
- Mémoire donnée/programme:
  - Le bloc B0 de 512 mots peut être mappé en espace donnée ou en espace programme
  - Le bloc SARAM de 9K peut être mappé en espace donnée ou en espace programme
- Mémoire donnée
  - bloc B2 32 mots et bloc B1 512 mots de DARAM donnée

Espace I/O du C50

- 64k de ports parallèle I/O 16 bits
- Accessibles par les bus adresse/donnée plus le signal IS
- Accédés par les instructions spécifiques IN et OUT
  - IN DAT7,0FFFEh: écrit dans la case mémoire donnée DAT7 la valeur du port 65534
  - OUT DAT7,0FFFFh: écrit sur le port 65535 la valeur contenue dans la case mémoire donnée DAT7
- 16 de ces ports (50h à 5Fh) sont des MMR (accessibles par toutes les instructions et pas seulement IN et OUT)
- Utilisés pour connecter les périphériques indispensables (convertisseurs D/A ou A/D, codecs ...)

Notes:

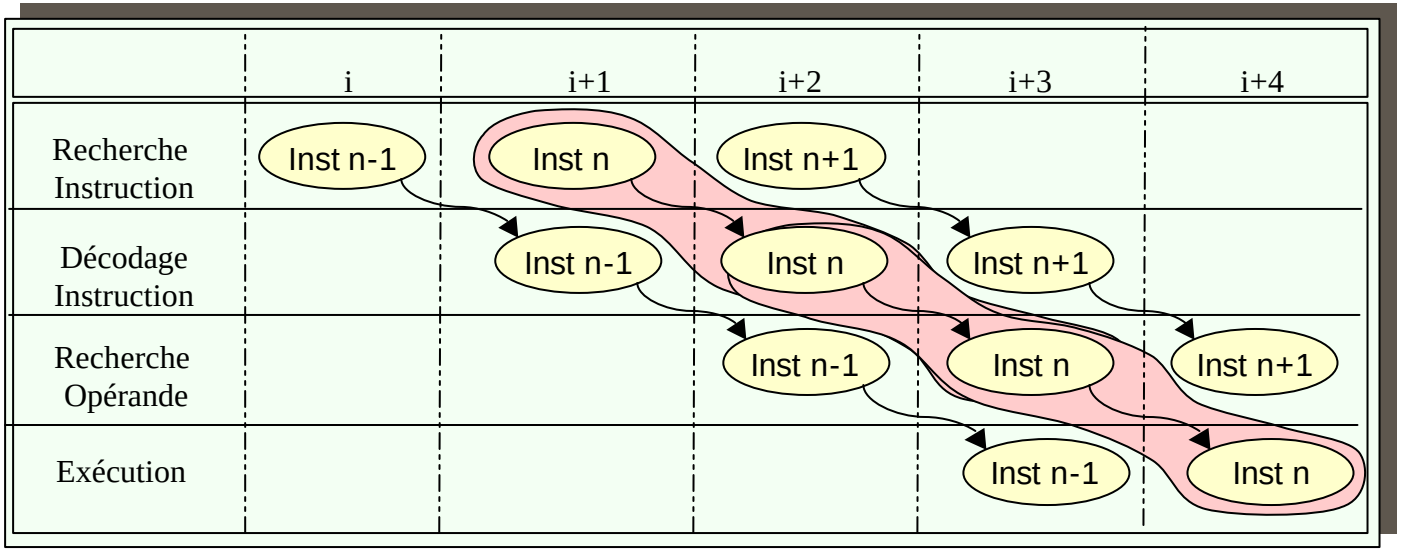
Notes:

III. Architecture des DSP conventionnels

3. Unité de contrôle

Pipeline

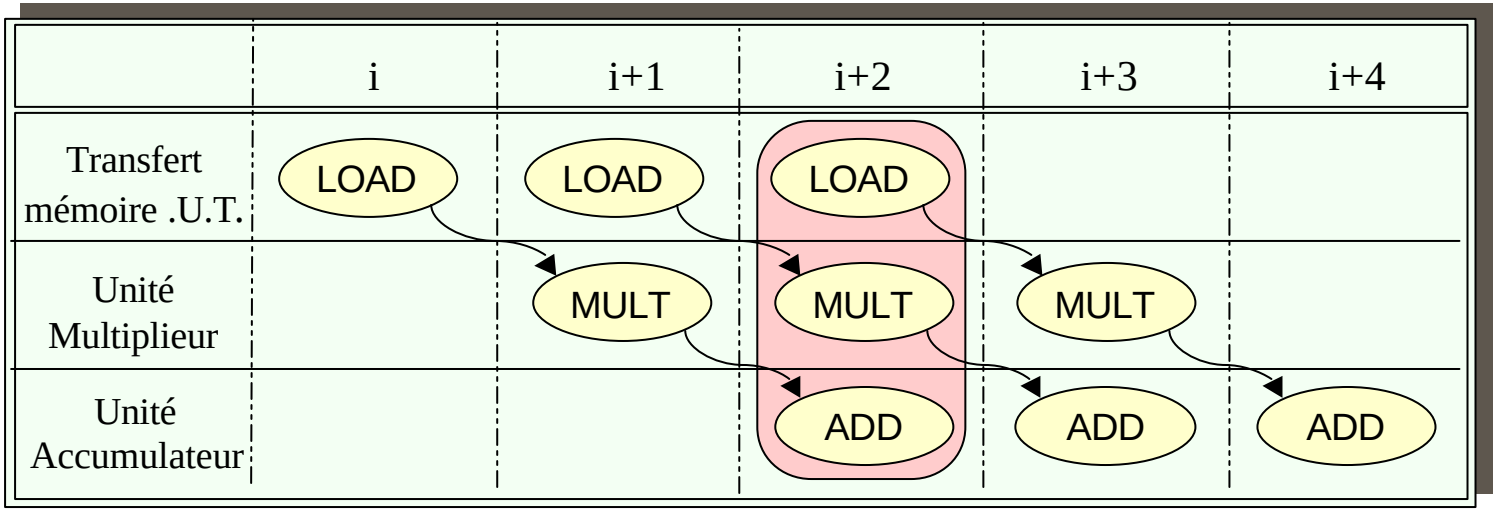
- Exécution de l'instruction en plusieurs phases :
  - 1. Recherche de l'instruction
  - 2. Décodage de l'instruction
  - 3. Recherche des opérandes
  - 4. Exécution



Codage des instructions

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

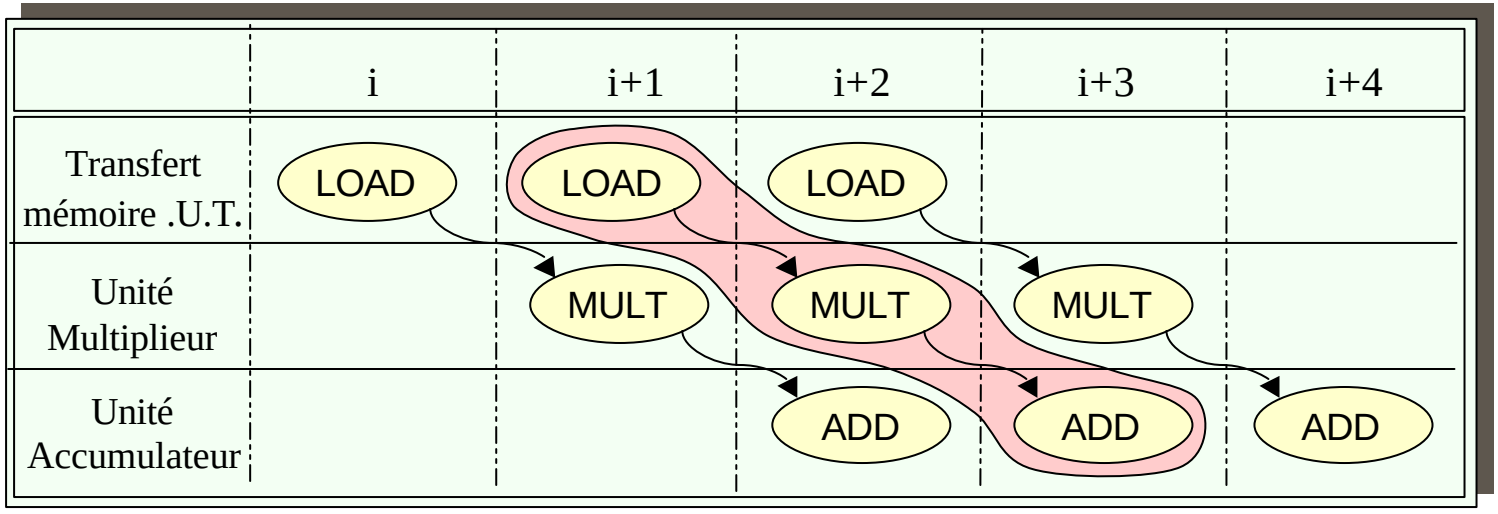
```
MULT R1,R2,T   ADD T,A,A   LD R1,AR1   LD R2,AR2
```



Codage des instructions

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

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



Format des instructions

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

⇙ Compromis ⇘

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

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
```

```
SPLK    #16
RPTB    End-1
```

C50

Structures de contrôle

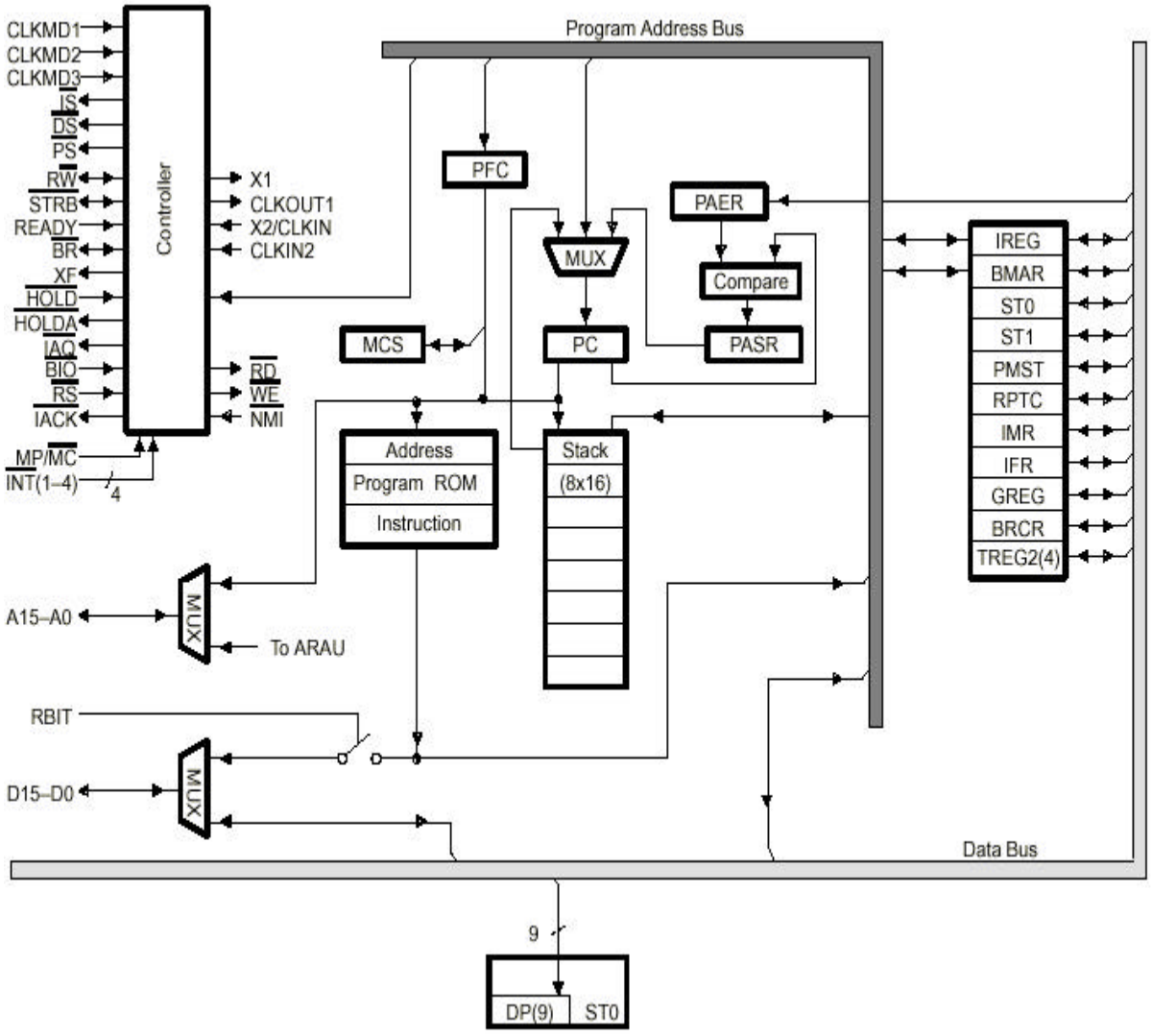
- Instructions de branchement
  - Branchement multi-cycles: ajout de NOP entre BR et 1ère instruction
  - Branchement retardé : déplacement de l'instruction de branchement BR

- 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, ...

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

III. Architecture des DSP conventionnels

4. Unité de communication

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

IV Panorama

1. Vue générale
2. TMS320C5x
3. TMS320C54x
4. TMS320C3x

DSP conventionnels

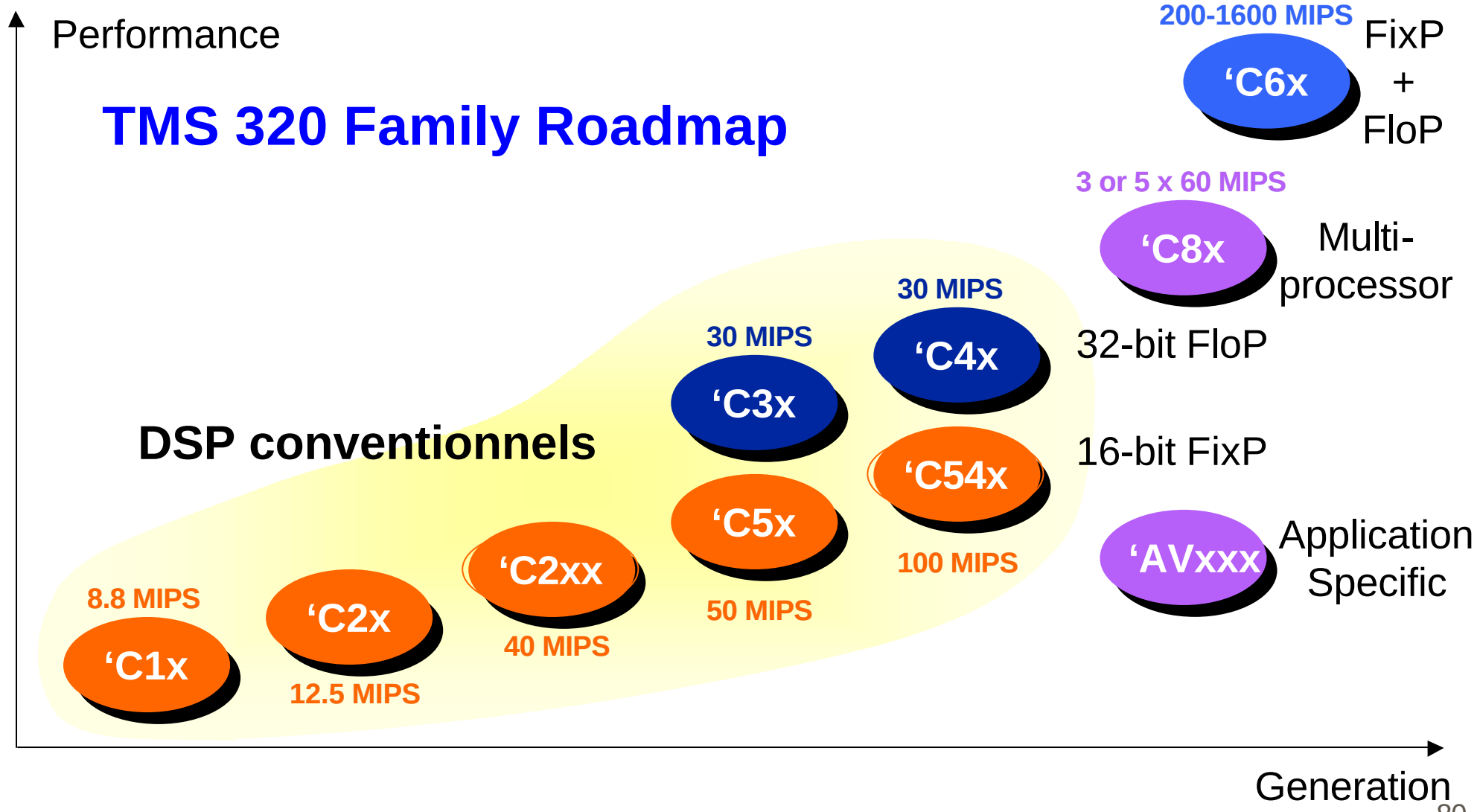
- 16-24 bits en virgule fixe, accumulation 40-60 bits
- 32 bits en virgule flottante
- 16 à 32 bits d'instructions
- Une instruction par cycle, jeu d'instructions complexe
- Architecture non orthogonale, fortement contrainte
- Mémoire à accès multiples "on-chip"
- Unités dédiées de gestion des adresses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

100 millions de produits intègrent ces DSP

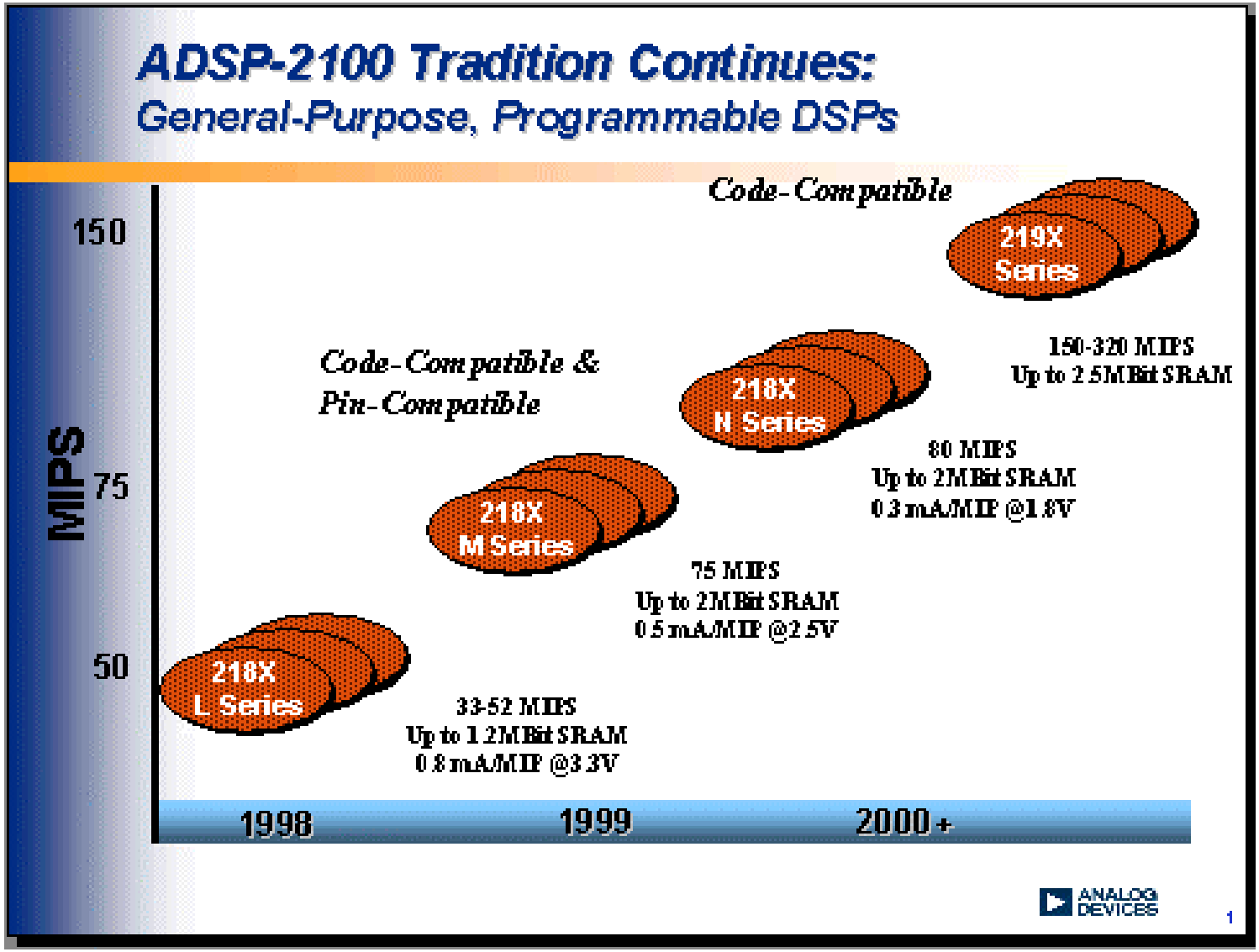
Panorama des DSP

- Texas Instrument
  - C2000 : virgule fixe 16 bits : contrôle moteur
  - C5000 : virgule fixe 16 bits : faible consommation
  - 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

Texas Instruments



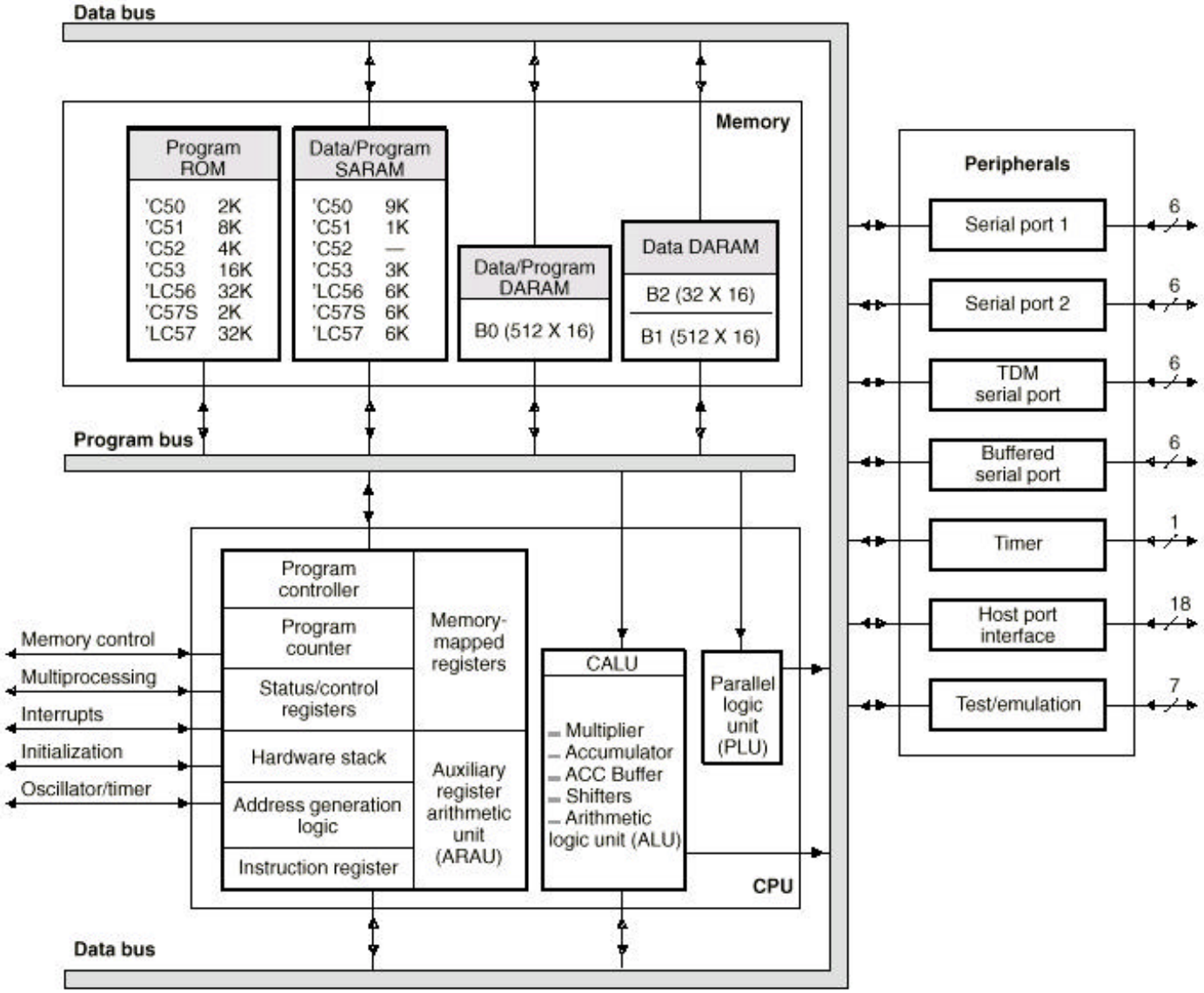
Analog Device



IV Panorama

2. TMS320C5x

TMS320C5x

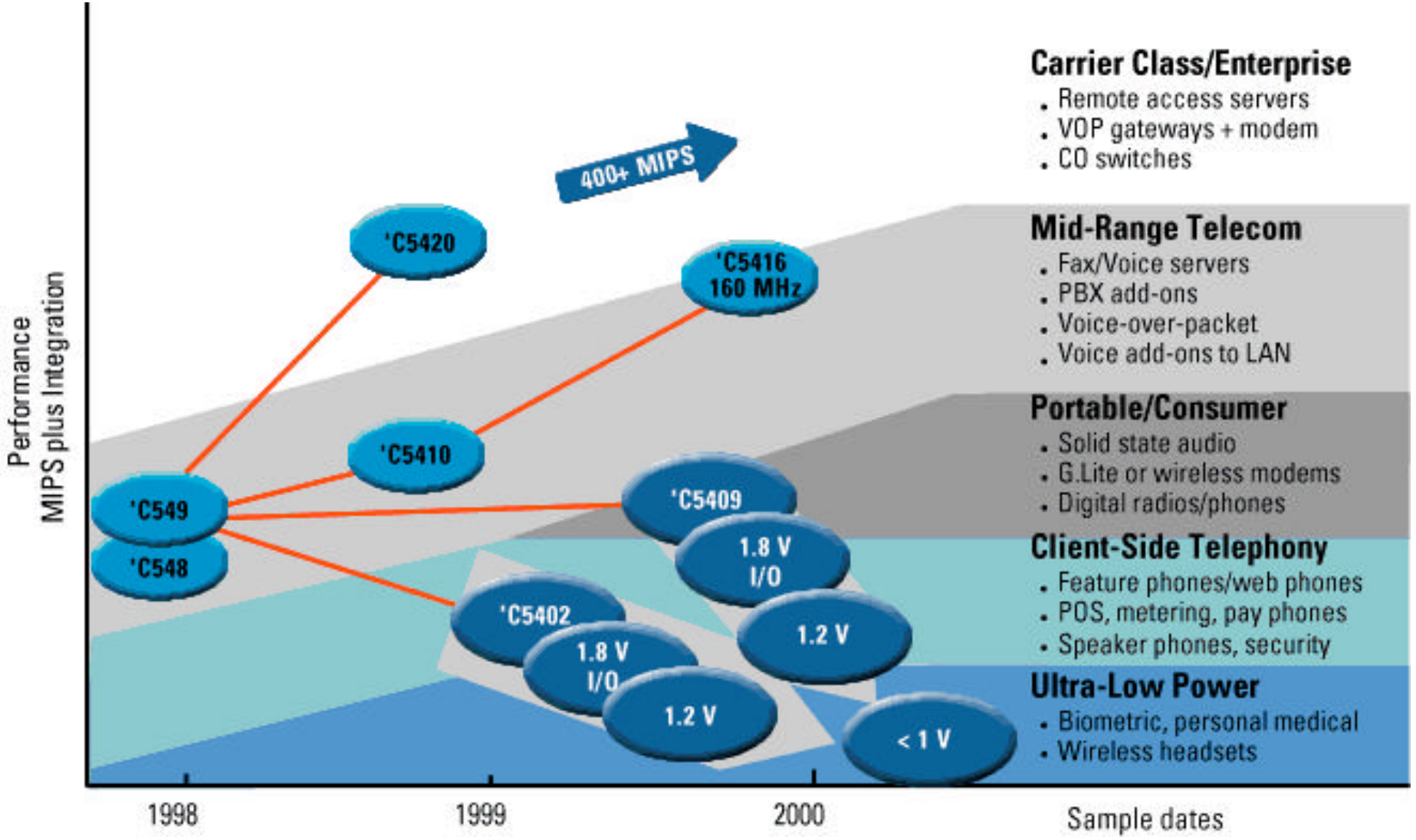

Data bus
Memory
Program ROM
'C50 2K
'C51 8K
'C52 4K
'C53 16K
'LC56 32K
'C57S 2K
'LC57 32K
Data/Program SARAM
'C50 9K
'C51 1K
'C52 —
'C53 3K
'LC56 6K
'C57S 6K
'LC57 6K
Data/Program DARAM
B0 (512 X 16)
Data DARAM
B2 (32 X 16)
B1 (512 X 16)
Program bus
Memory control
Multiprocessing
Interrupts
Initialization
Oscillator/timer
Program controller
Program counter
Status/control registers
Hardware stack
Address generation logic
Instruction register
Memory-mapped registers
Auxiliary register arithmetic unit (ARAU)
CALU
Multiplier
Accumulator
ACC Buffer
Shifters
Arithmetic logic unit (ALU)
Parallel logic unit (PLU)
CPU
Data bus
Peripherals
Serial port 1
Serial port 2
TDM serial port
Buffered serial port
Timer
Host port interface
Test/emulation
6
6
6
6
1
18
7

IV Panorama

3. TMS320C54x

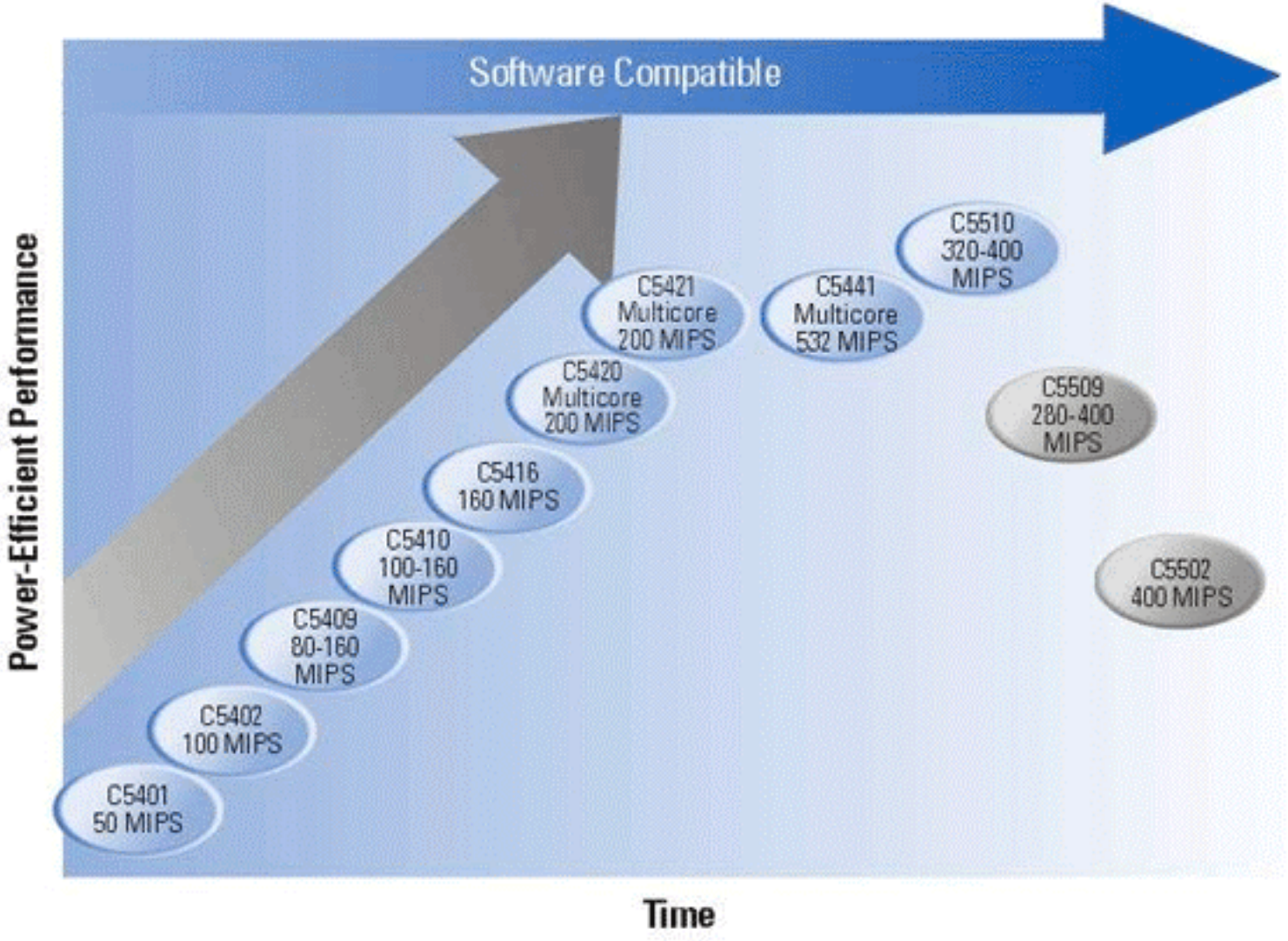
Famille C5x

- Evolution

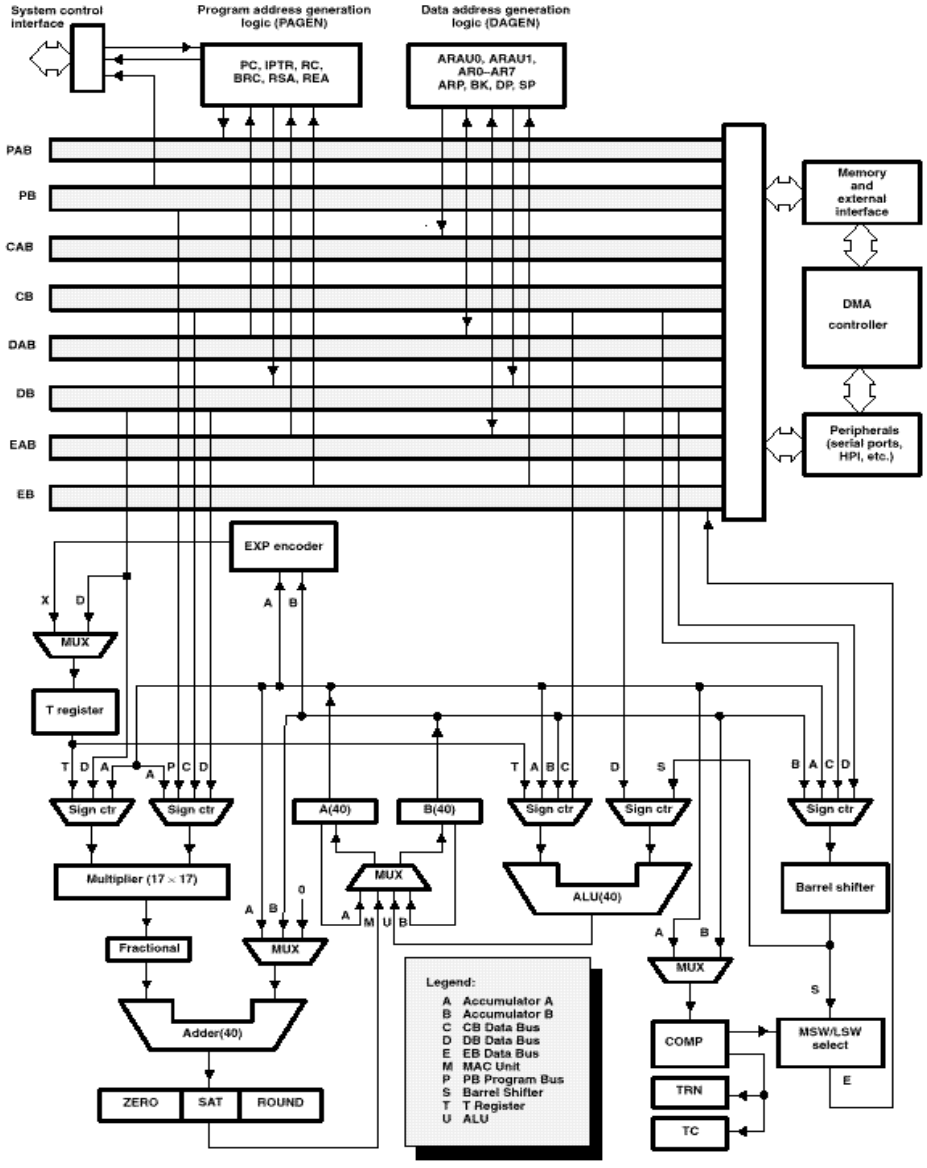


Famille C5000

- Evolution

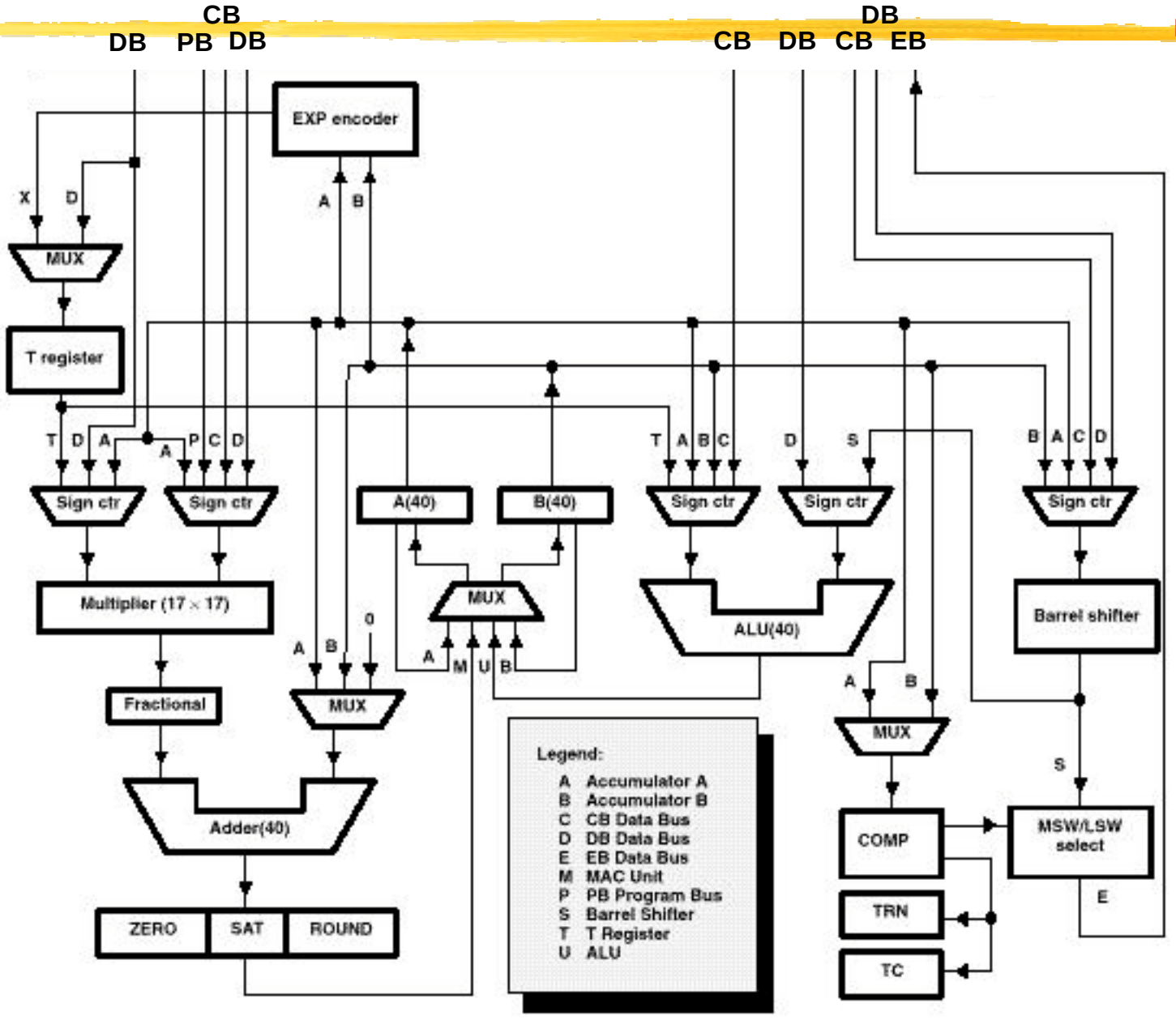


Architecture du C54x



- C54x
  - 40-100-160 MIPS
  - 1000-2000-3000 MIPS/W
  - 17x17b multiplier
  - 40b ALU
  - 40b adder, ACS unit
  - 60% of cellular handsets
  - $5 for C5402 100MIPS - $75

Architecture de l ’UT du C54x

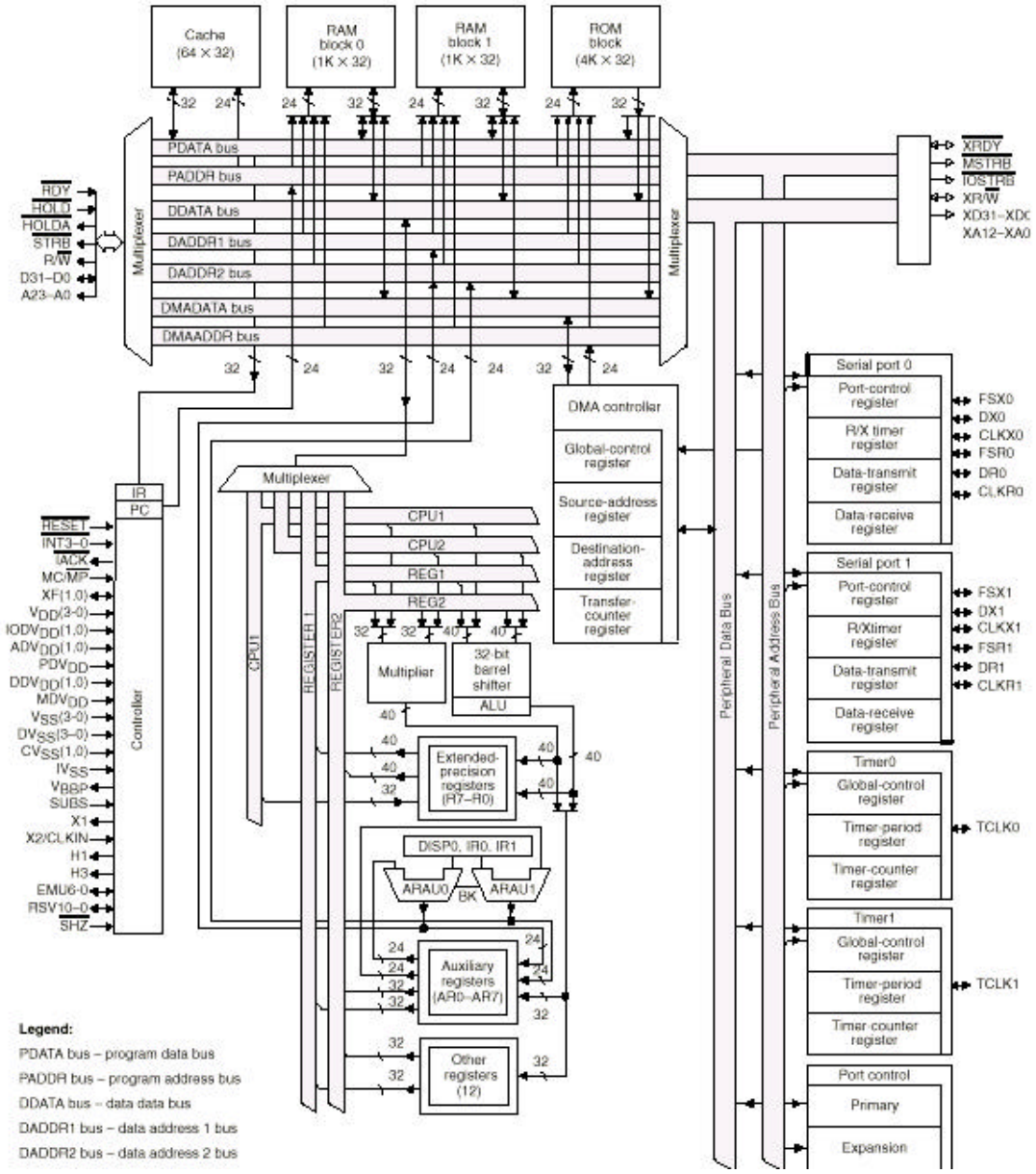


IV Panorama

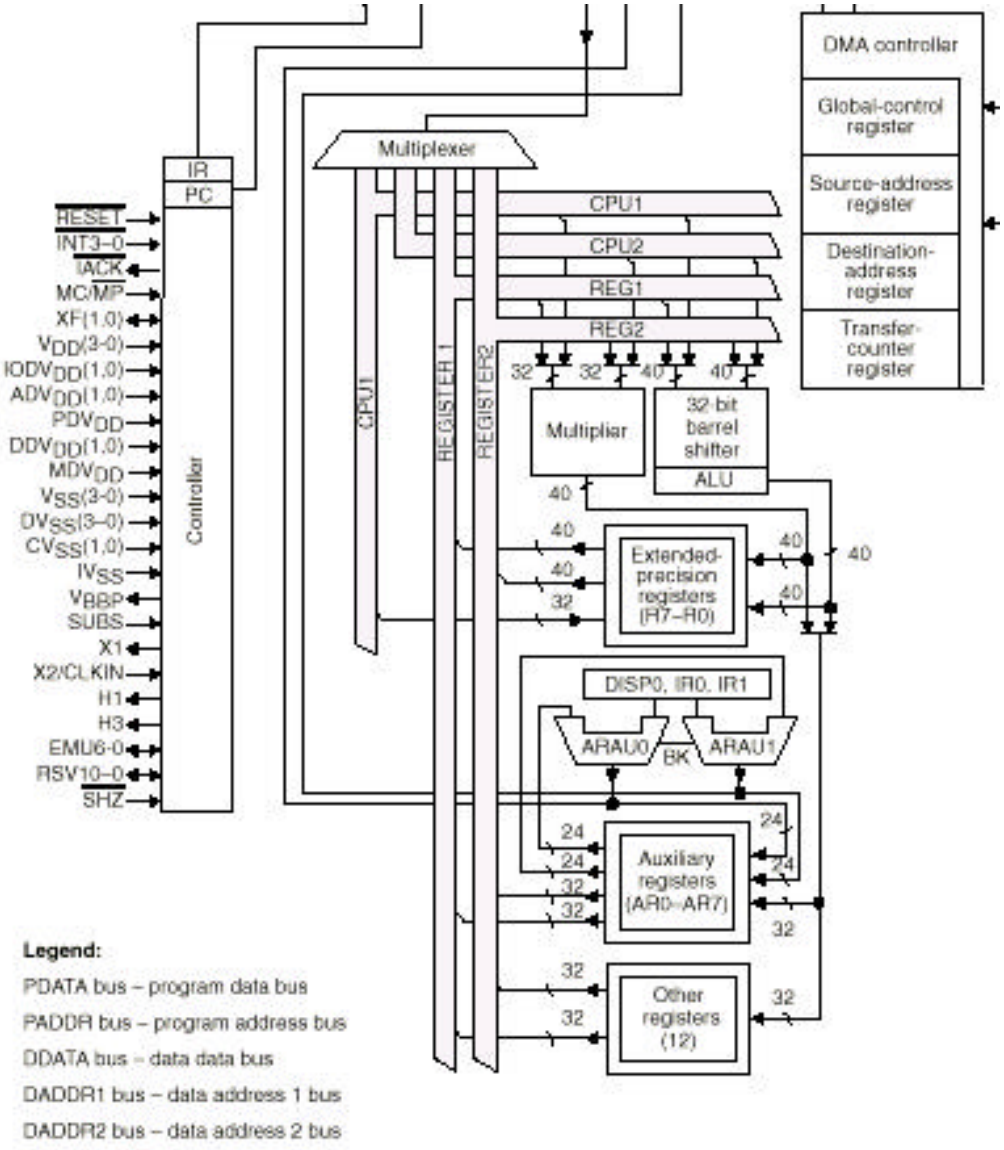
4. TMS320C3x

Architecture du TMS 320C3x

http://archi.enssat.fr



Architecture du TMS 320C3x



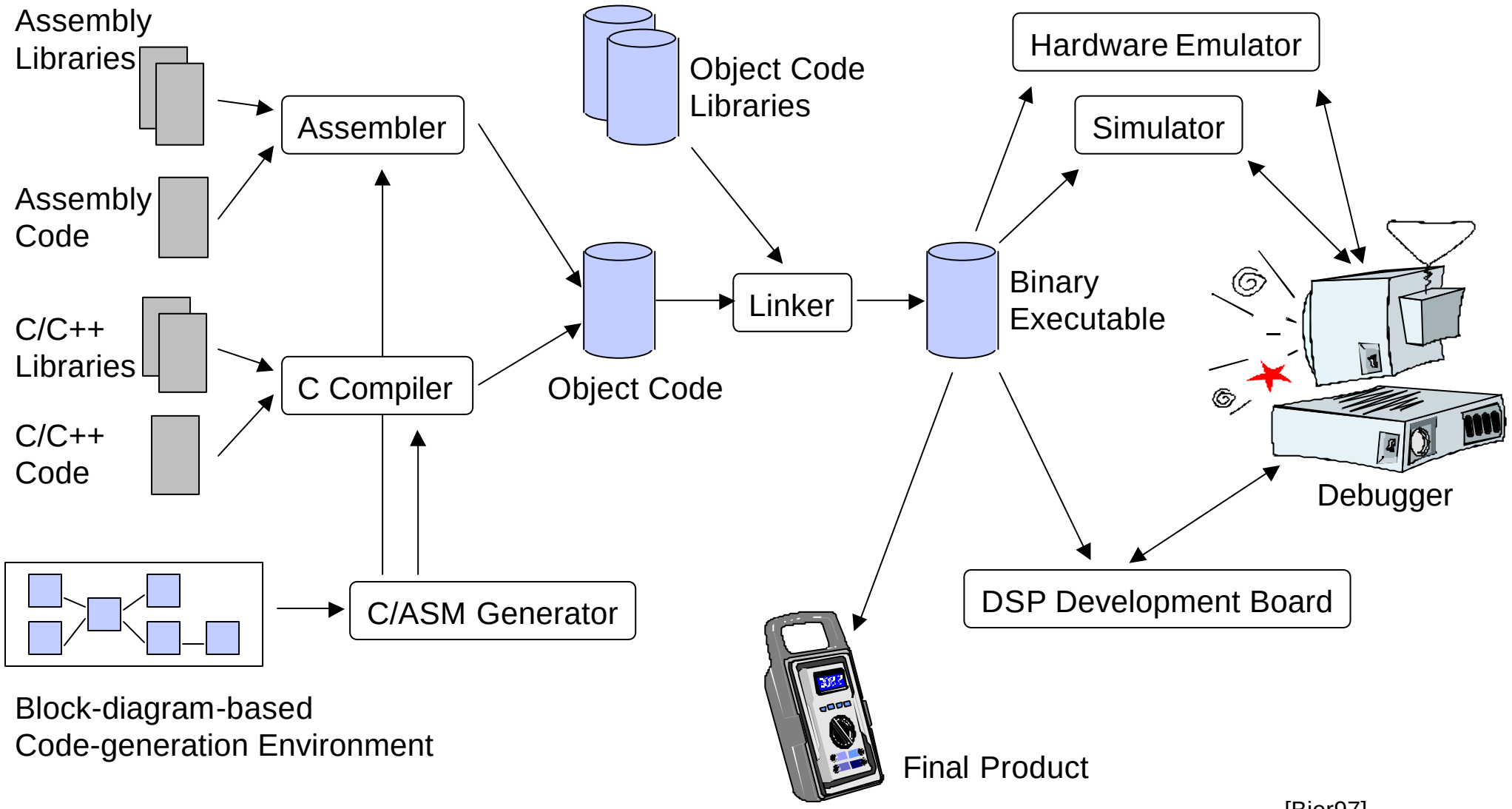
Notes:

Notes:

V Implantation des algorithmes sur DSP

1. Outils de développement
2. Outils de développement pour C50 et C30
3. Compilateurs C

Outils de développement



[Bier97]

Outils de développement

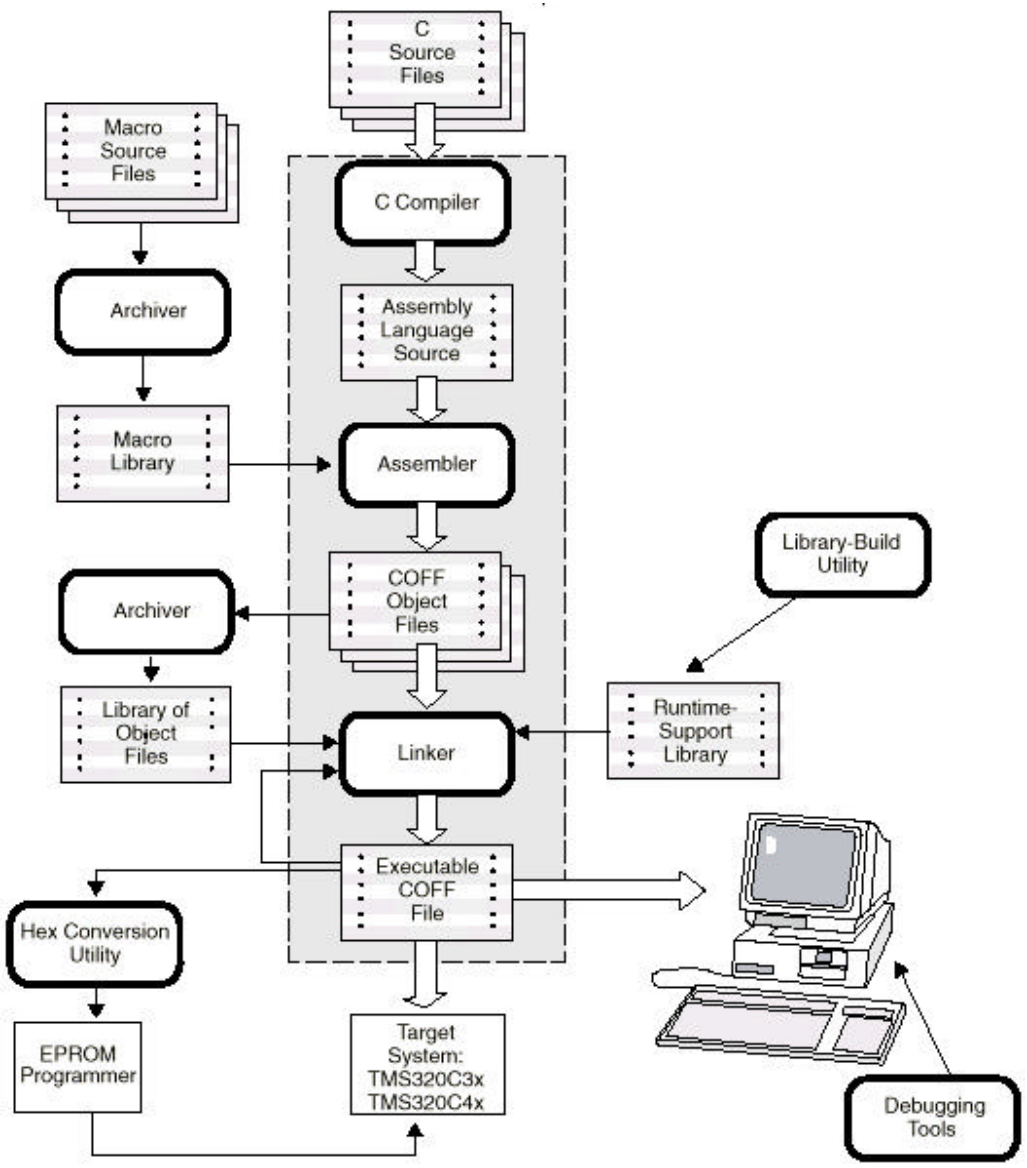
- Développement haut niveau
  - Spécifications du flot de données du TS,
  - Simulation, validation
  - Prototypage rapide
  - Matlab/Simulink, Cadence SPW, Synopsys Cossap, Ptolemy
- Compilateurs de langage de *haut-niveau*
  - DSP en virgule fixe ou flottante
- Outils de profiling - optimisations de l'assembleur

Développement de l'application

- Mise au point - 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

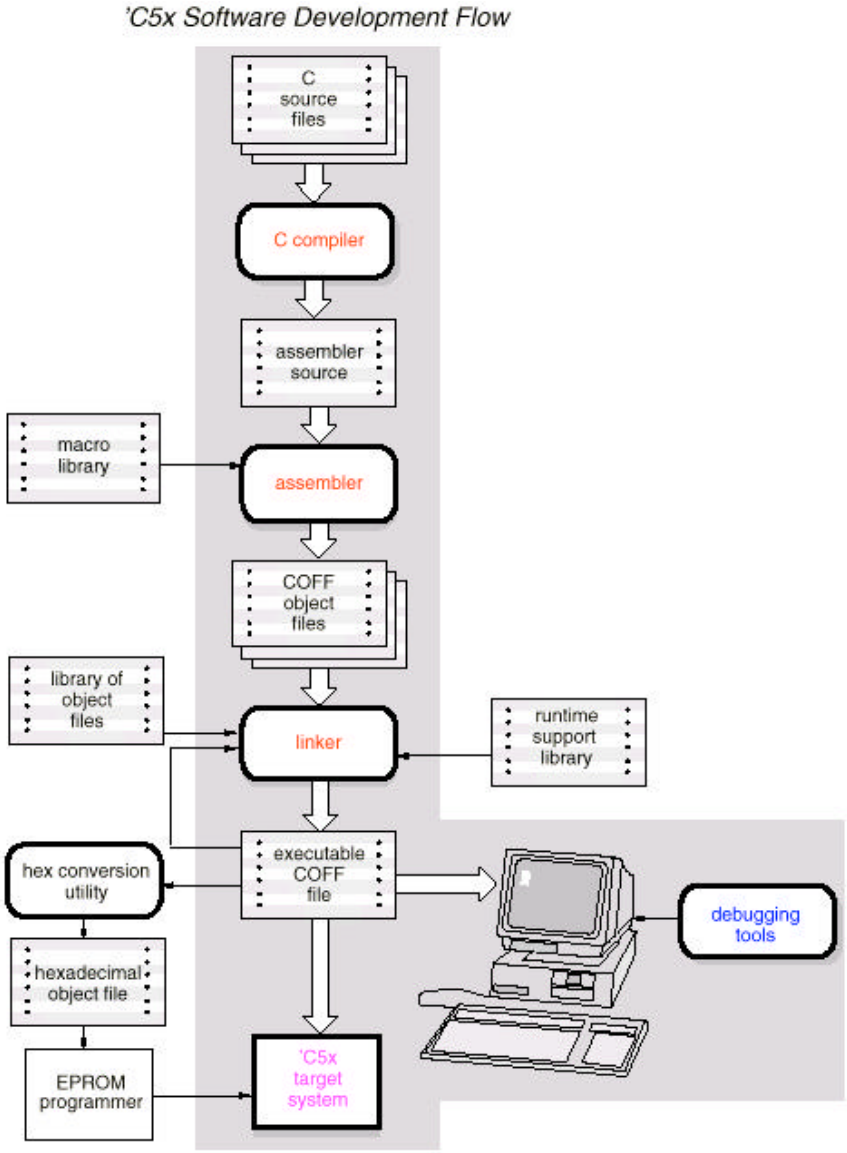
Chaîne de développement TI pour C30

- Compilateur C
- Librairies
  - rtslib
- Émulateur
- Cartes d'évaluation
  - EVM C30
  - DSK C30



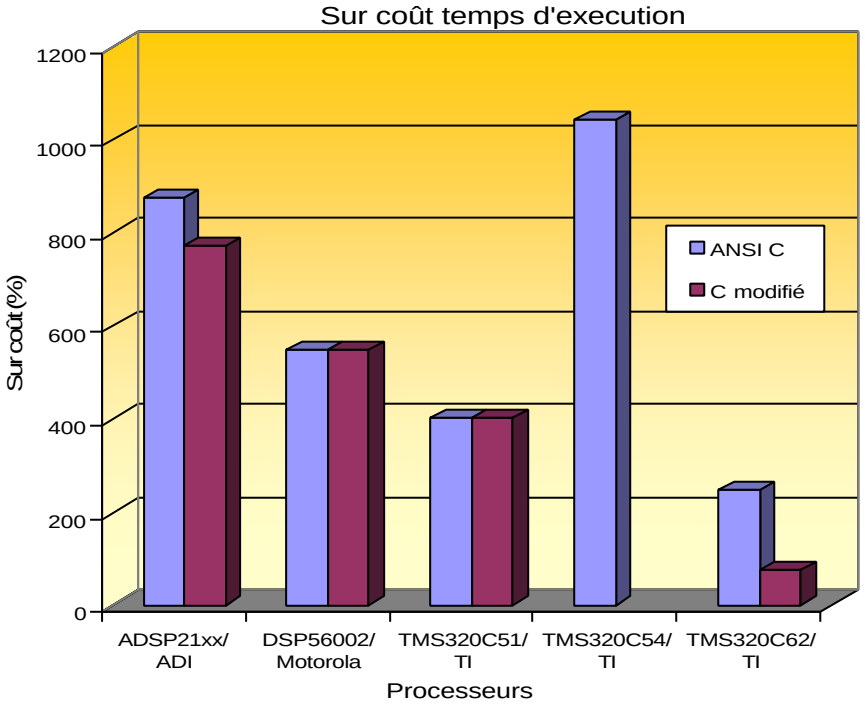
Chaîne de développement TI pour C50

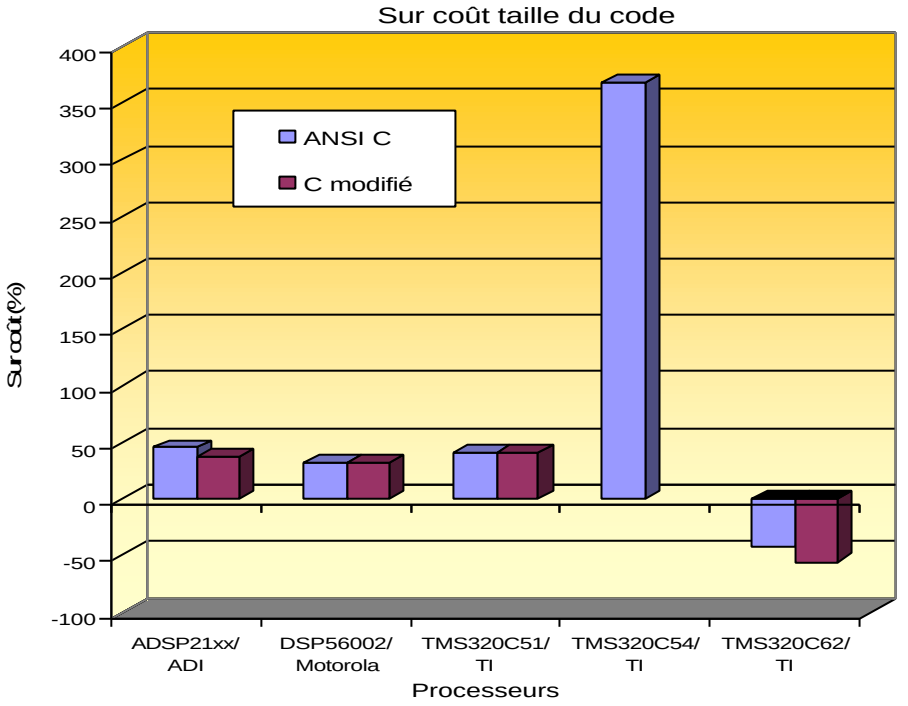
- Compilateur C
- Librairies
  - rtslib
- Émulateur
- Cartes d'évaluation
  - EVM C50
  - DSK C50



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

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)…

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

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

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

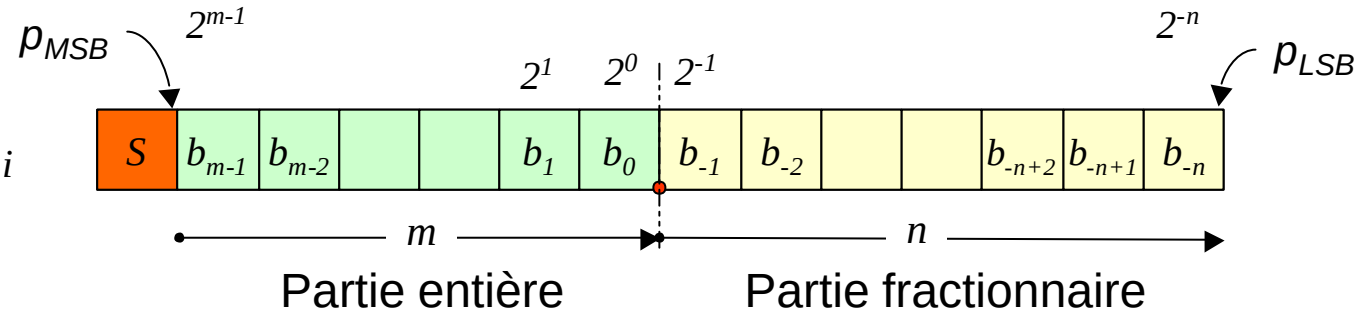
VI. Traitement en Virgule Fixe

1. Arithmétique virgule fixe
2. Détermination du domaine de définition
3. Codage des données

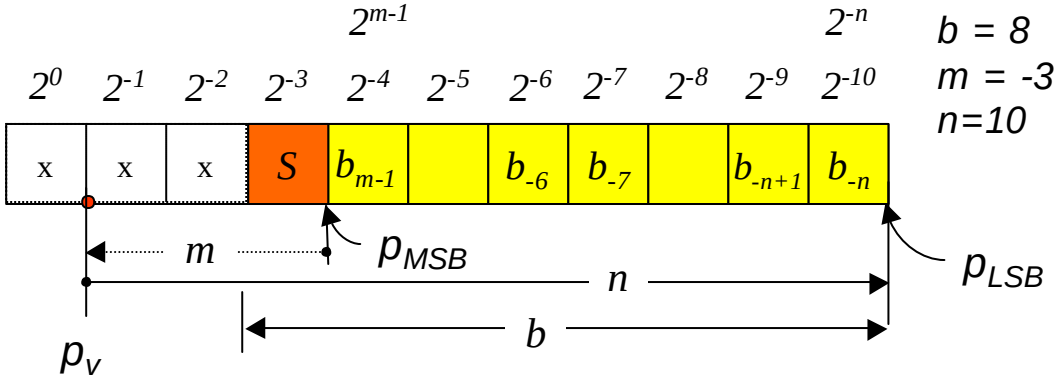
Codage en virgule fixe complément à 2

- **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 = 8
m = -3
n=10

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

ex: 0.09472 (8,-3,10)
***0**1100001*
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
1111.01	-0.75
1000.01	-7.75
1000.00	-8

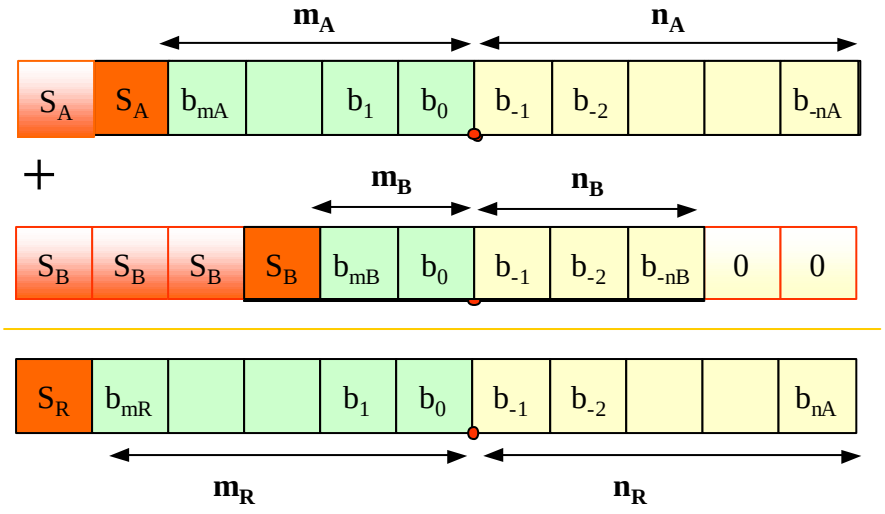
-0.5 = -8+7. 5

-7.75 = -8+0.25

Règles de l'arithmétique virgule fixe

- Addition: $r=a+b$
  - *Règle* : le format des opérandes *a* et *b* doit être identique
  - *Étapes* :
    - Choix d'un format commun (b_c,m_c,n_c)
    - Alignement de la virgule
    - Extension des bits des opérandes *a* et *b*

$$
\begin{aligned}
m_c &= \max(m_A, m_B) \\
n_c &= \max(n_A, n_B) \\
b_c &= m_C + n_C + 1
\end{aligned}
$$



$$
m_R = \begin{cases} m_C + 1 & \text{si } a+b \notin D_C \\ m_C & \text{si } a+b \in D_C \end{cases}
$$

$$
\begin{aligned}
n_R &= n_C \\
b_R &= m_R + n_R + 1
\end{aligned}
$$

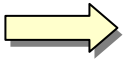
Exemple de calcul

- Addition: $r=a+b$
  - débordement possible si $sign(a) = sign(b) = s_{ab}$
    - débordement présent si $sign(r) \neq s_{ab}$

```
  1 1 1
  0110  (6)
  1011  (-5)
  ----
  0001  (1)
```

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

```
  1 1 1
  0011  (3)
  0111  (7)
  ----
  1010  (-6)
```

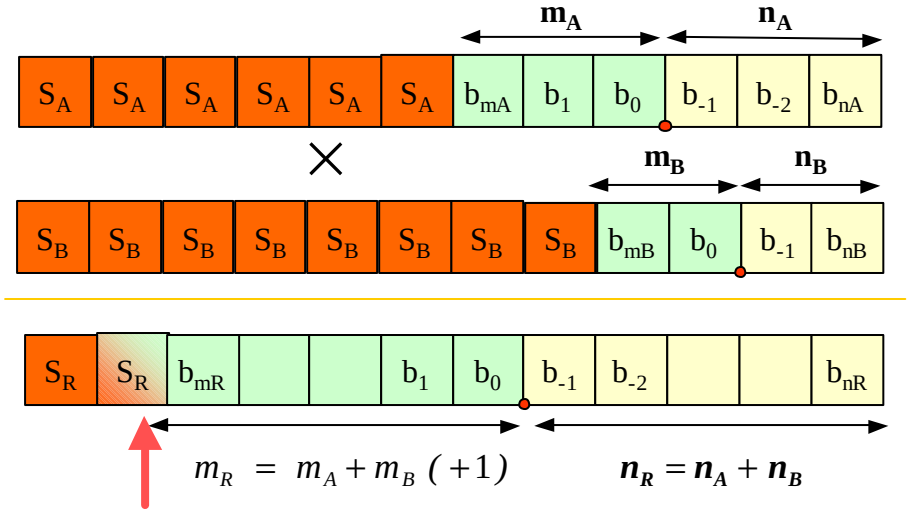


```
    1 1 1
  00011  (3)
  00111  (7)
  -----
  01010  (10)
```

Nécessité d 'un bit supplémentaire pour coder le résultat

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



$$n_R = n_A + n_B$$
$$m_R = m_A + m_B + 1$$
$$b_R = b_A + b_B$$

Exemple de calcul

- Multiplication: $r = a \times b$
  - *Extension du signe des opérandes*

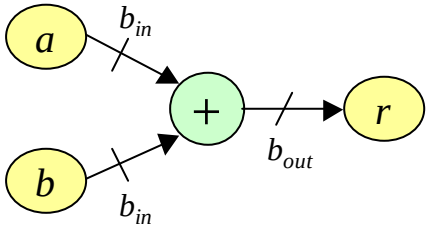
```
      1100  (-4)
      0110  ( 6)
-----------------
1111100.
111100. .
 1 1 1
-----------------
11101000
```

-128+64+32+8 = -24

```
       110  (-2)
       100  (-4)
-----------------
  1110 . .
  110 . . .
  10 . . . .
   1
-----------------
  001000  (8)
```

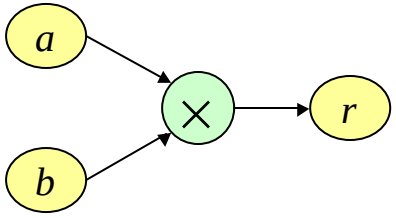
Résumé des règles

- Addition
  - Choix du format commun :



$$\text{si } b_{out} = b_{in} \text{ alors } m_c = \max(m_A, m_B, m_R)$$
$$\text{si } b_{out} > b_{in} \text{ alors } m_c = \max(m_A, m_B)$$

- Multiplication

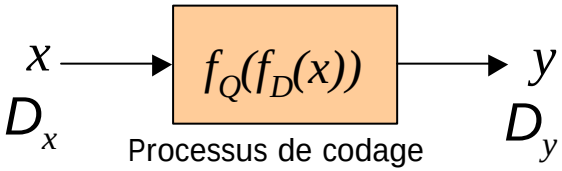


$$n_R = n_A + n_B$$
$$m_R = m_A + m_B + 1$$
$$b_R = b_A + b_B$$

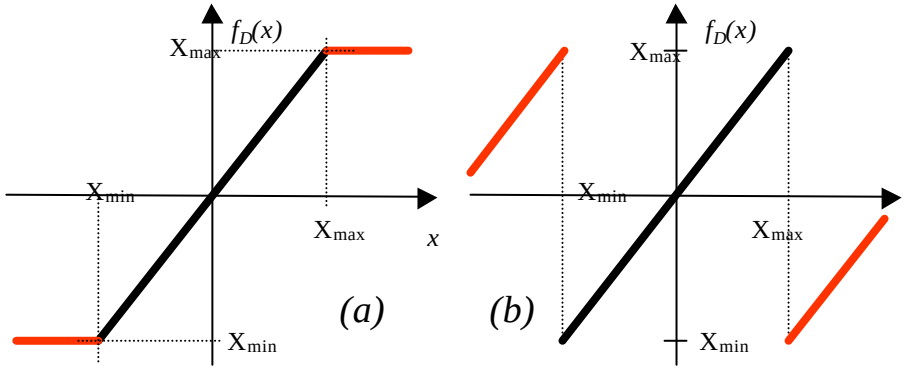
Doublement du bit de signe

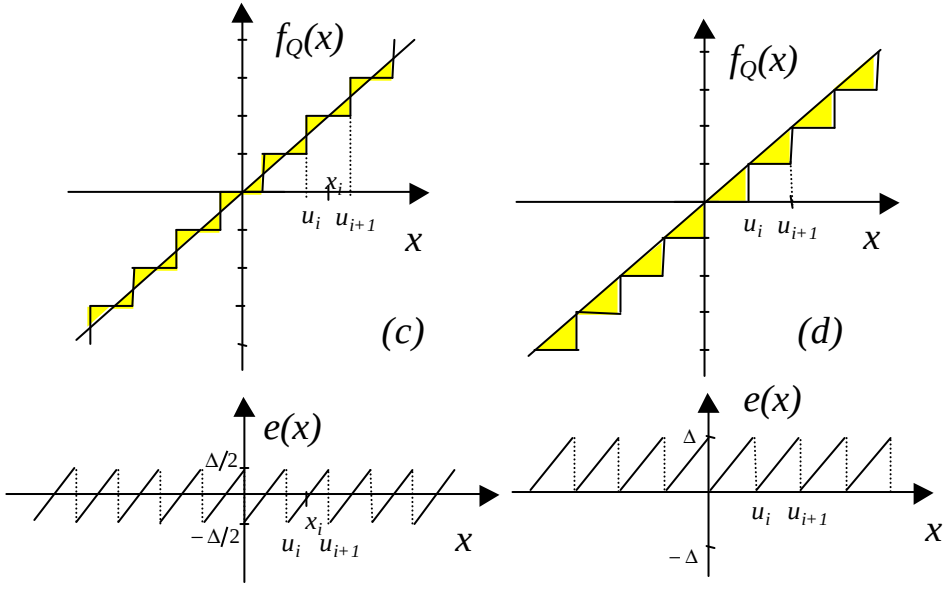
Notes:

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
      - $e(x) = x - y$





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
- Objectifs et contraintes du codage :
  - Respecter l'intégrité de l'algorithme
    - Respecter les règles de l'arithmétique virgule fixe
    - Garantir l'absence de débordement
  - Maximiser la précision (Rapport Signal à Bruit de Quantification : RSBQ)

Les différentes étapes du codage

- Détermination du domaine de définition : $D(x_i)$
  - Déterminer le nombre de bits minimal pour représenter la partie entière
- Codage et Optimisation des données :
  - Respecter l'intégrité de l'algorithme
  - Utiliser au mieux l'architecture
- Évaluation de la précision de l'algorithme :
  - Calcul du Rapport Signal à Bruit de Quantification (cf. TNS)

Notes:

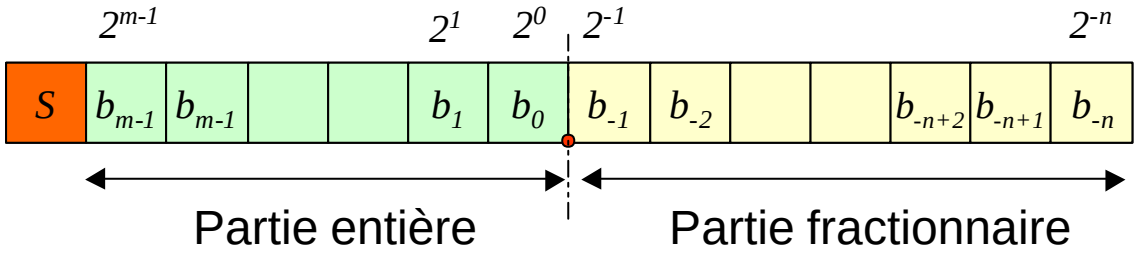
Notes:

VI. Traitement en Virgule Fixe

2. Détermination du domaine de définition

Détermination du domaine de définition

- Format des données: *(b,m,n)*



$b = m + n + 1$ bits

format: *(b,m,n)*

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

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

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

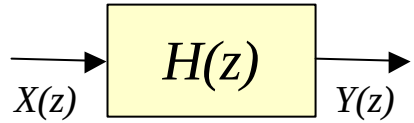
  ⇨ Estimation précise mais dépendante du signal d'entrée

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

  ⇨ Estimation conservatrice mais l'absence de débordement est garantie

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

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

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

  - Norme L2

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

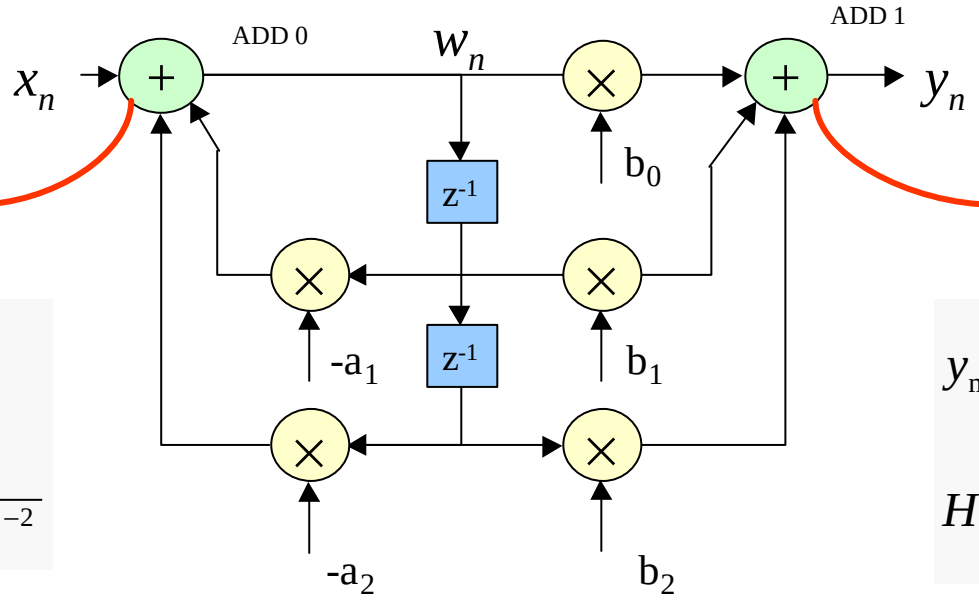
➢ Méthode limitant la probabilité de débordement

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

Dynamique des données dans un IIR

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

Forme directe II



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

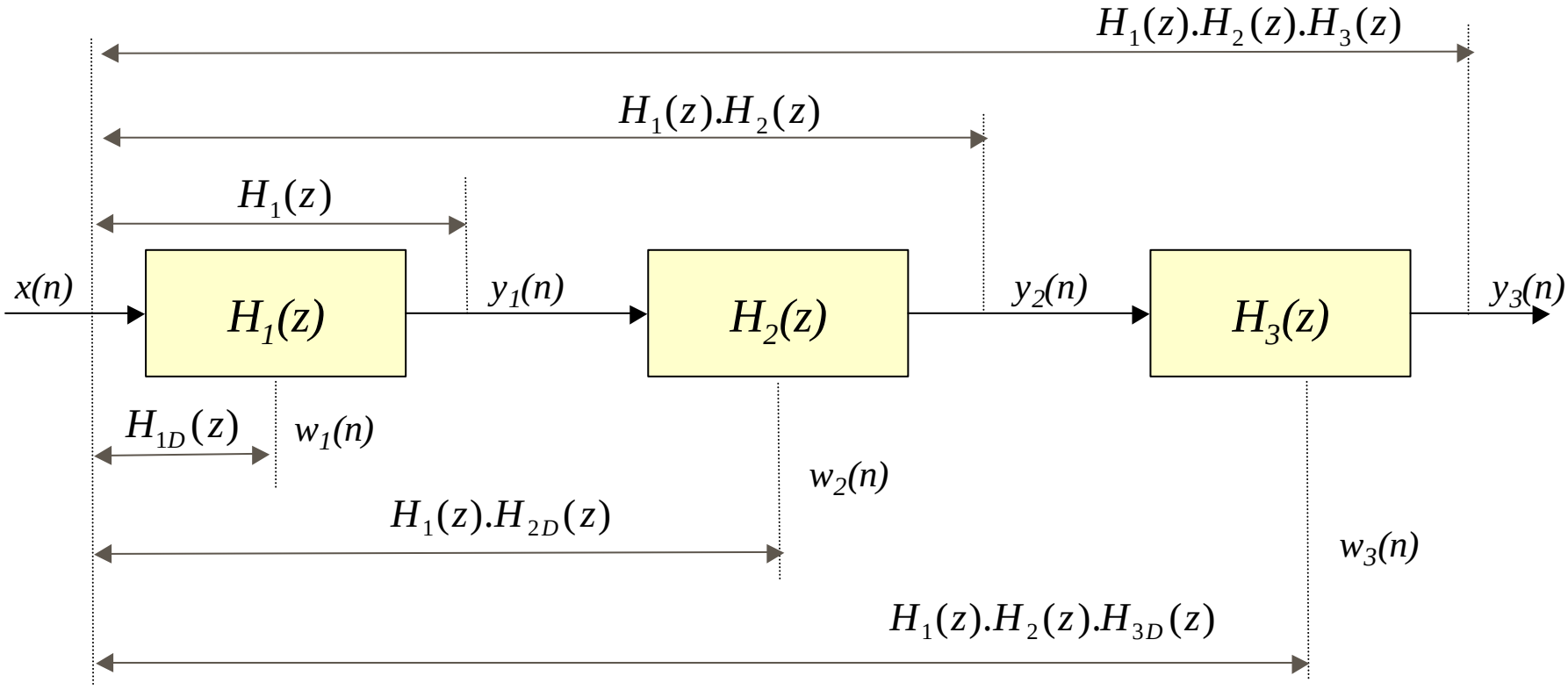
$$H_D(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

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

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

Dynamique des données dans un IIR

- Cellules d'ordre 2 cascadées



$$H_{iD}(z) = \frac{1}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}} \qquad H_i(z) = \frac{b_{0i} + b_{1i}z^{-1} + b_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}}$$

Exemple : filtre IIR directe II

- Filtre H(z)

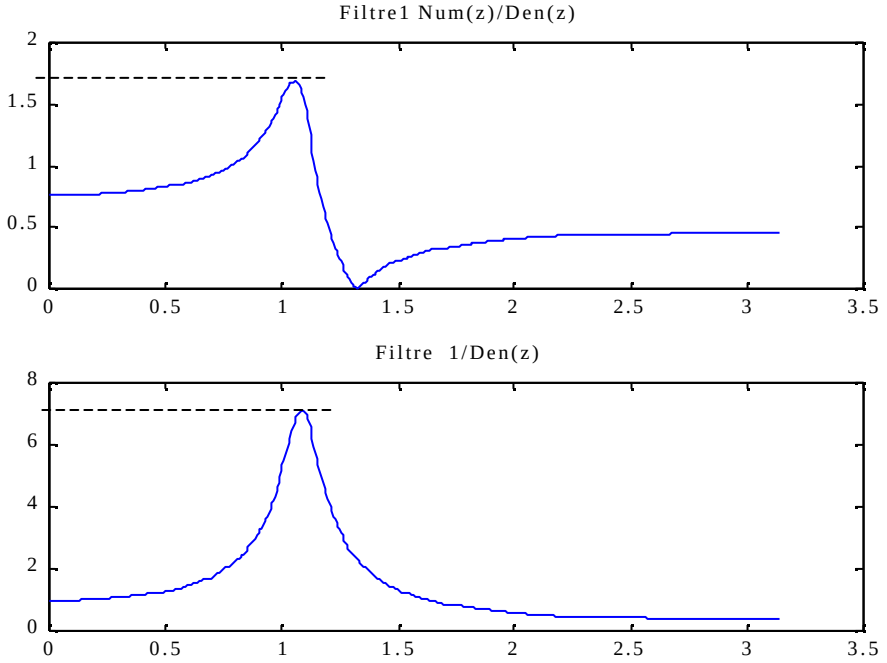
$$H(z)=\frac{0.5-0.2428z^{-1}+0.5}{1-0.85z^{-1}+0.8417z^{-2}}$$

- 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 $H_D(z)$

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:

Notes:

VI. Traitement en Virgule Fixe

3. Codage des données

Contraintes du codage en virgule fixe

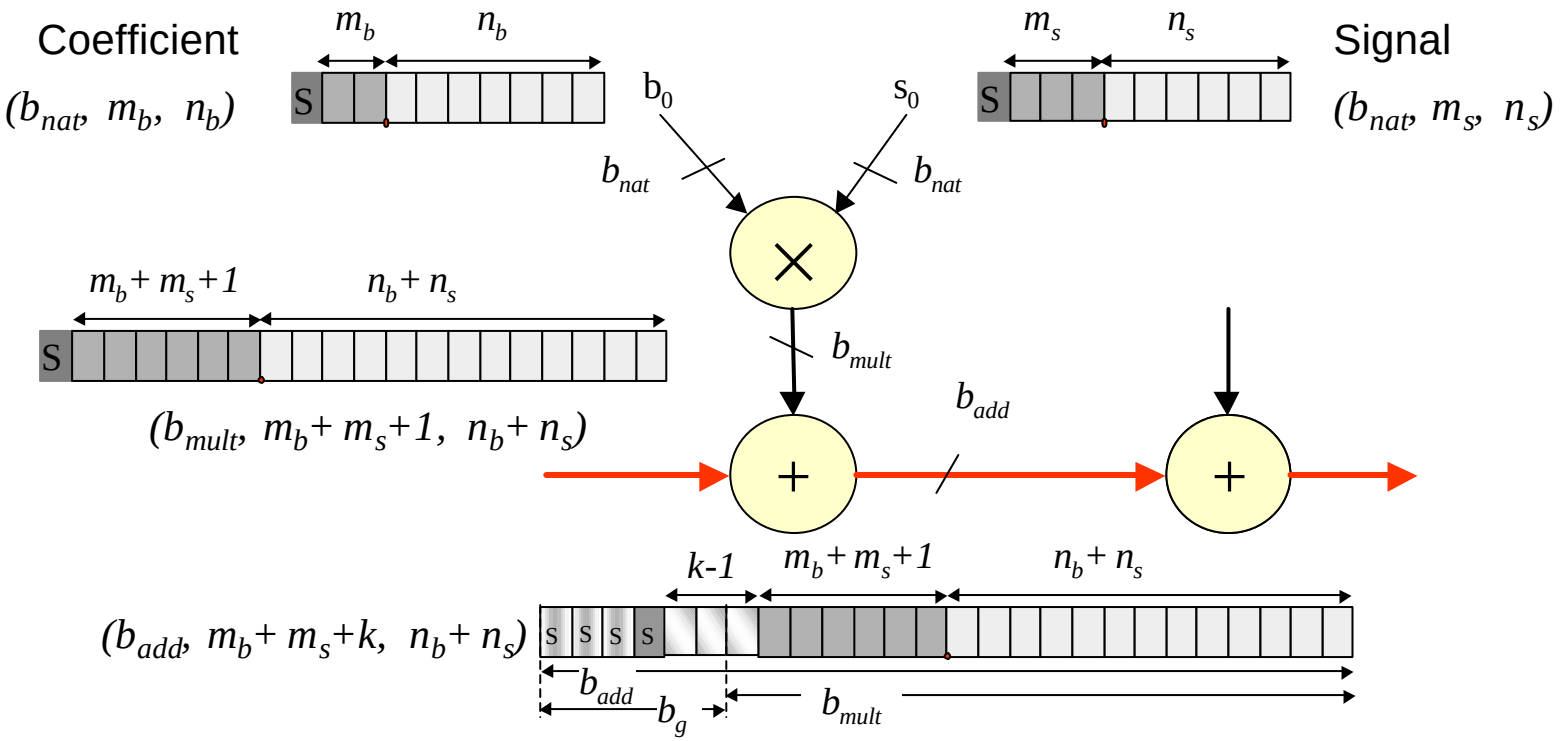
- Respect des règles de l'arithmétique virgule fixe
  - Alignement de la virgule des opérandes sources d'une addition ou d'une soustraction
    - Format commun pour les opérandes sources

- Prévenir les débordements
  - Débordement possible lors d'une addition ou d'une soustraction
    - Nécessité d 'introduire des bits supplémentaires :
      - Recadrage des données : adapter le format des données à la dynamique des données

Introduction des bits supplémentaires

- Utilisation des bits de garde de l'accumulateur (b_g bits) :
  - $b_{add} = b_{mult} + b_g$
  - Pas de débordement si le nombre de bits supplémentaires k respecte :

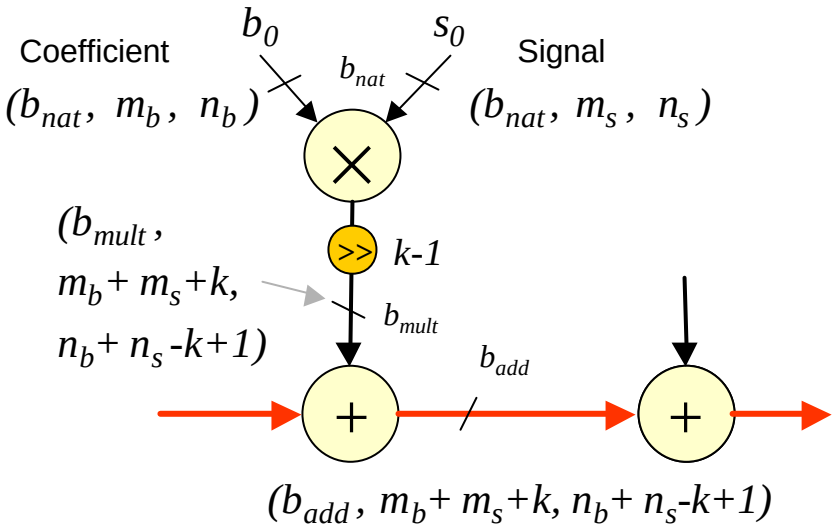
$$k \leq b_g$$

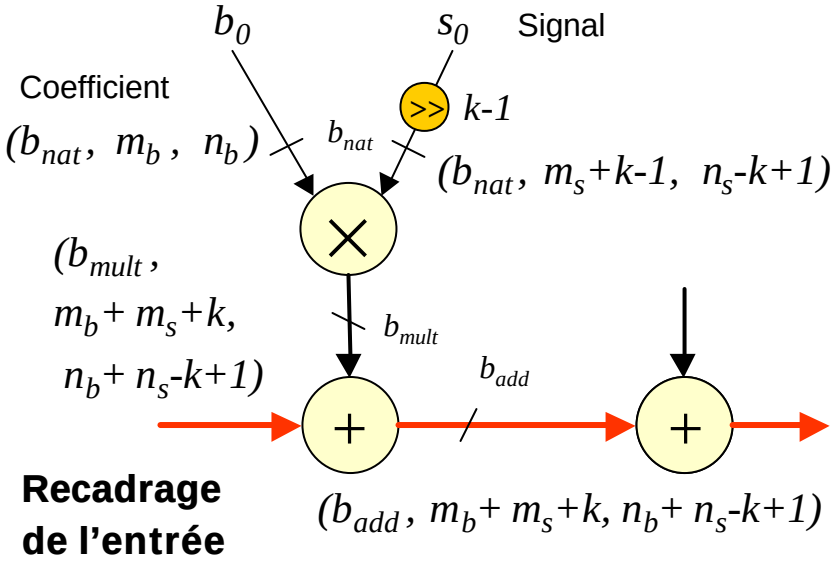


Introduction des bits supplémentaires

- Recadrage de la sortie du multiplieur $(b_{add} = b_{mult})$:
  - Recadrage interne

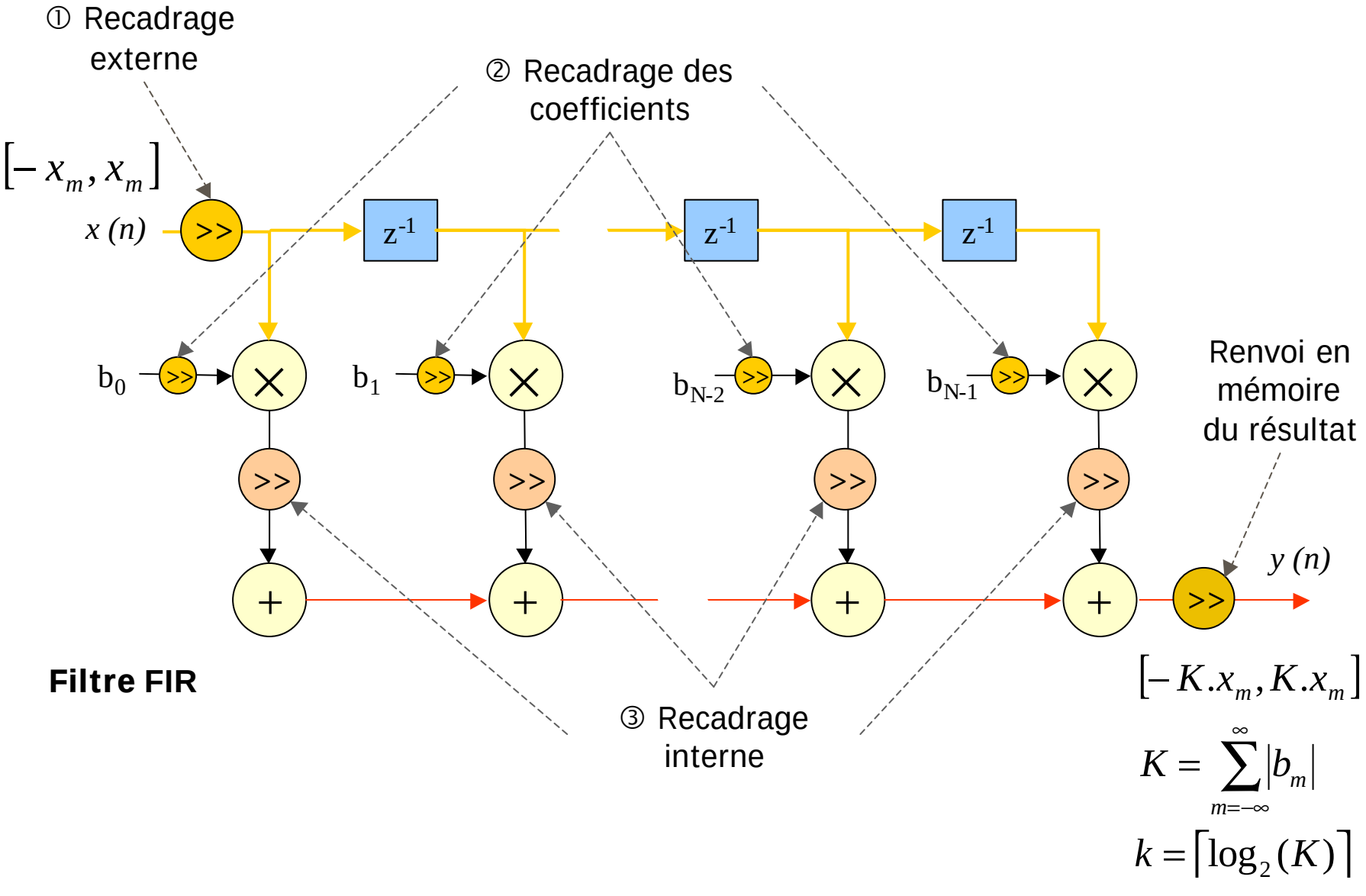
- Recadrage avant la multiplication $(b_{add} = b_{mult})$:
  - Recadrage de l'entrée
  - Recadrage des coefficients
    - réalisé lors du codage de ces coefficients





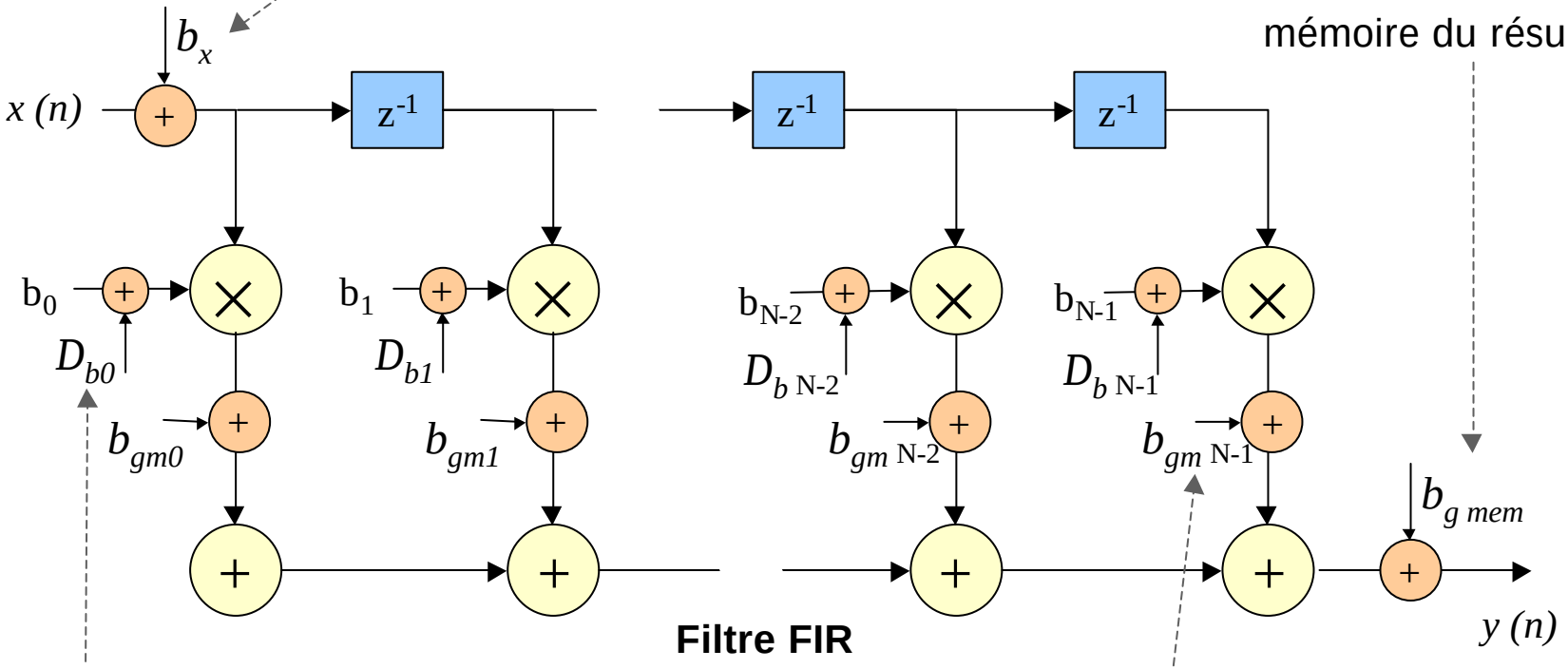
Recadrage de l'entrée

Recadrage des données dans un FIR



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

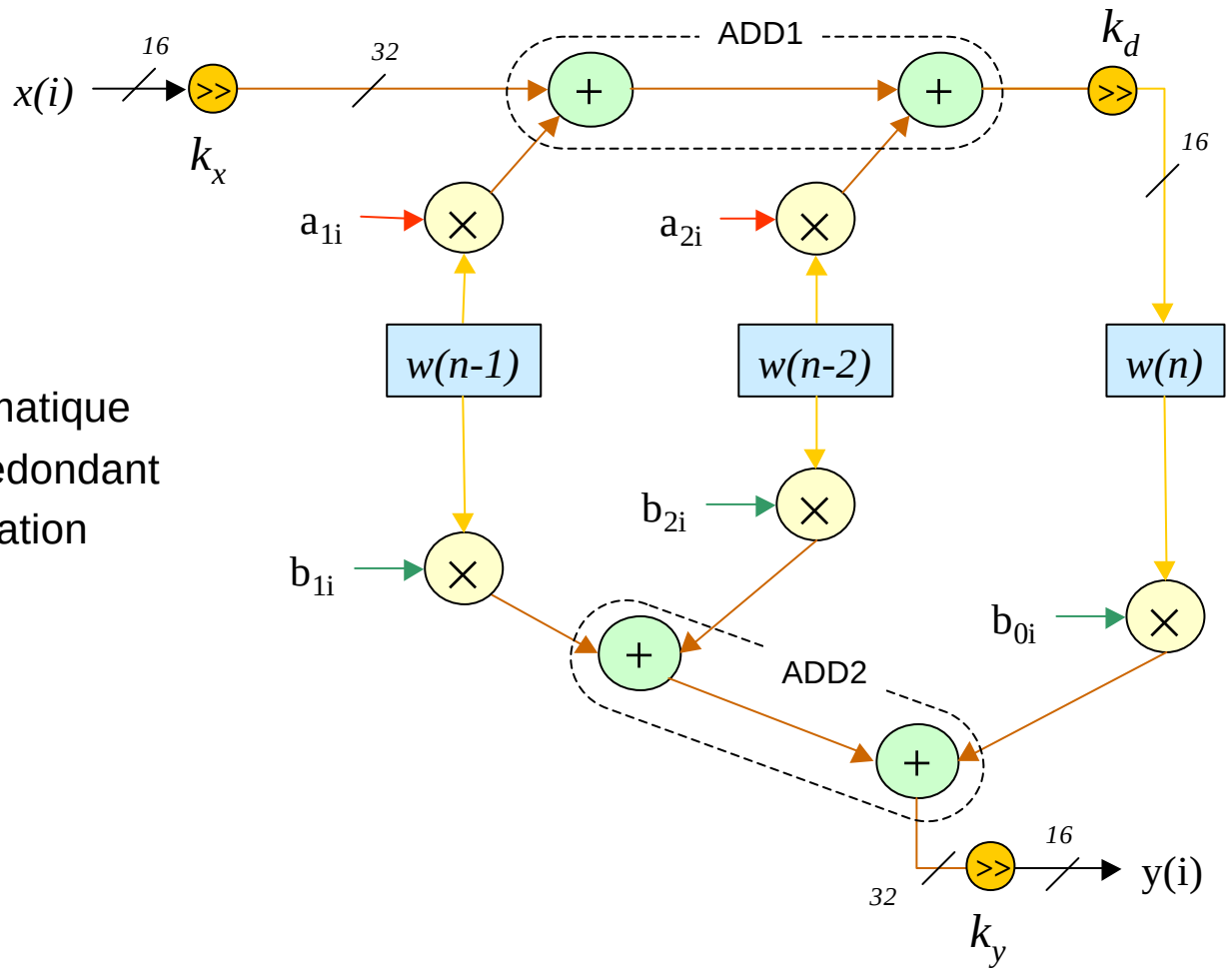
- Bruit lié au recadrage interne ③

Notes:

Notes:

Exemple de codage : filtre IIR

- Graphe flot du filtre IIR



Élimination automatique du bit de signe redondant après la multiplication

Exemple de codage : filtre IIR

- Système d'équations à résoudre
  - Garantir l'absence de débordement sur les données et les coefficients

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

$$m_w \geq \left\lceil \log_2 \left(\max_n \left(|w(n)| \right) \right) \right\rceil$$

$$m_y \geq \left\lceil \log_2 \left(\max_n \left(|y(n)| \right) \right) \right\rceil$$

$$m_{a_i} \geq \left\lceil \log_2 \left(\max_i \left(|a_i| \right) \right) \right\rceil$$

$$m_{b_i} \geq \left\lceil \log_2 \left(\max_i \left(|b_i| \right) \right) \right\rceil$$

Exemple filtre IIR

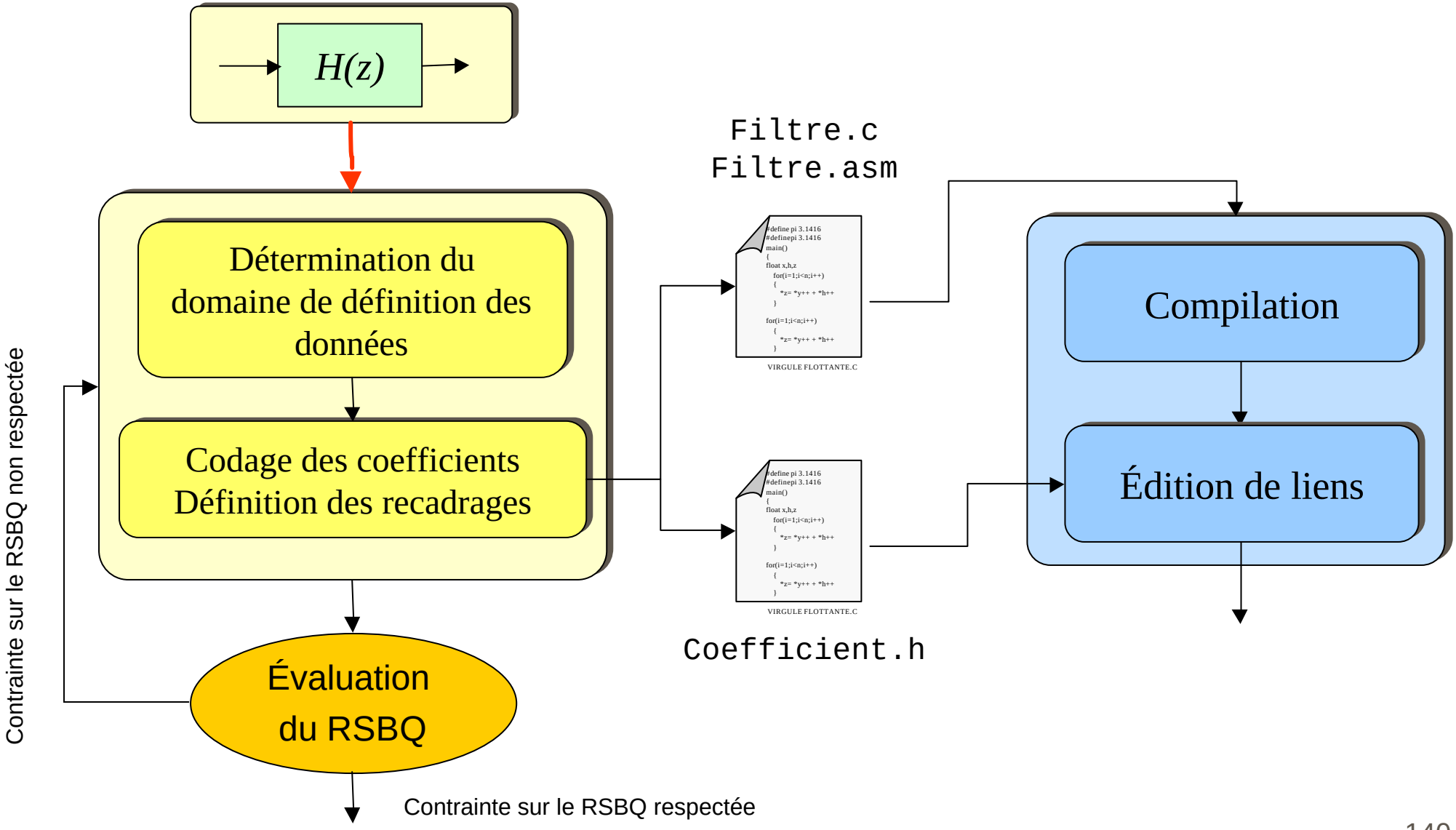
- Choix d ’un format commun pour les additionneurs

$$m_{ADD1} = \max\left(m_a + m_w, m_x, m_w\right)$$
$$m_{ADD2} = \max\left(m_b + m_w, m_y\right)$$

- Alignement de la virgule

$$m_a + m_w = m_{ADD1}$$
$$m_w + k_w = m_{ADD1}$$
$$m_x + k_x = m_{ADD1}$$
$$m_b + m_w = m_{ADD2}$$
$$m_y + k_y = m_{ADD2}$$

Méthode de codage des données



Conclusion

- Les différentes étapes :
  - Détermination des fonctions de transfert intermédiaires
  - Calcul de leur réponse impulsionnelle
  - Détermination du domaine de définition des données

  - Analyse du GFD (graphe flot de données) du programme
  - Détermination du système d'équations régissant le format des données
  - Résolution du système d'équations
  - Obtention des recadrages et du format des coefficients

Notes:

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.