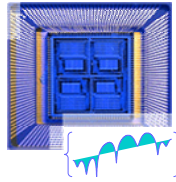


Processeurs de Traitement Numérique du Signal (DSP)



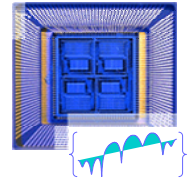
Olivier Sentieys
IRISA/ENSSAT Lannion
Université de Rennes I
sentieys@irisa.fr
<http://www.irisa.fr/R2D2>



Merci à Daniel Ménard pour son support



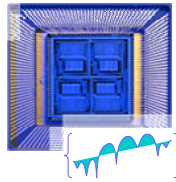
Processeurs de Traitement Numérique du Signal (DSP)



- I. Introduction
- II. Architectures MAC/Harvard
- III. Evolutions des DSP
- IV. Flot de développement



I. Introduction



1. Contexte applicatif
2. Caractéristiques algorithmiques
3. Solutions architecturales
4. Marché des DSP



Exemples d'applications

- Performances
- Faible coût
- Faible énergie

- **Téléphonie cellulaire**
- **Communications sans-fil**
- **Contrôle de moteur**
- **Modems**
- Voix sur Internet
- Audio grand public
- Navigation
- Vidéoconférence
- Jouets, consoles vidéo
- Synthèse musicale, effets
- Médical
- Communications satellite
- Analyse sismique
- Sécurité
- Reconnaissance vocale
- Sonar, radar
- Photo et caméra numériques
- Débruitage, écho
- Anticollision
- ...
- **et pleins d'autres à venir**



Tâches élémentaires

- Compression de signal (parole, audio, vidéo)
- Filtrage
- Modulation et démodulation
- Détection et correction d'erreurs
- Contrôle
- Traitements audio (e.g. réduction bruit, égalisation, annulation d'écho, conversion de fréquence)
- Reconnaissance vocale
- Synthèse de signaux

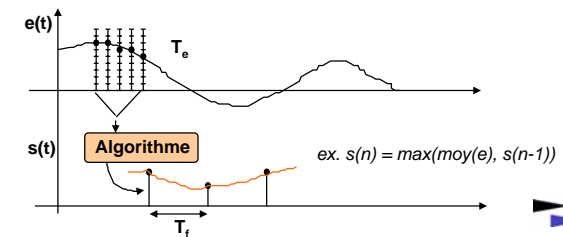
ARCHI'05 - 5



Algorithmes de TdSI¹

¹Traitement du signal et de l'image

- Signaux numériques
 - Temps et amplitude discrets
 - Flot de données
 - Scalaires, vectorielles, matricielles, ...
- Traitement temps réel
 - Temps d'exécution T_{ex} guidé par flots de données
 - Période d'échantillonnage T_e période des sorties $T_f > T_{ex}$

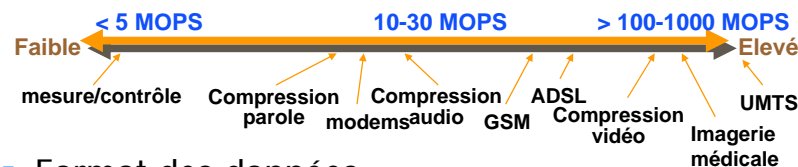


ARCHI'05 - 6



Diversités

- Complexité des algorithmes de TdSI



- Format des données

Application	Taille des données
PWM, mesures, contrôle convertisseurs sigma-delta	1 à 4 - 18 à 22 bits
radio HF/VHF, radar	6 - 14 bits
sonar	10 - 12 bits
parole	8 - 14 bits
audio	16 - 20 bits
imagerie / vidéo	8 - 36 bits (par pixel)
analyseurs de sang	16 - 18 bits

ARCHI'05 - 7



Caractéristiques algorithmiques

- Charge de calcul importante
 - Nids de boucles
 - Multiplications-accumulations (convolution)
 - Multiplications-additions (FFT, DCT, adaptation, distances, ...)
- Précision des calculs
 - Virgule fixe ou flottante
 - Compromis coût - précision des calculs
- Bande passante mémoire
- Calculs d'adressage complexes
 - Gestion signal, accès image, bit-reverse (FFT), ...
- Boucles de traitement courtes
 - Les instructions peuvent être placées en interne
 - Pas besoin de grande taille de cache (données ou instructions)

ARCHI'05 - 8



Fonctions typiques de TdSI

- Convolution, filtrage
 - $y = y + x.h$: MAC (multiplication-accumulation)
- Adaptation (LMS)
 - $y_n = y_{n-1} + x.h$: 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 (addition-comparaison-sélection)
- Estimation de mouvement
 - $sad += |x_{ij} - y_{i+u,j+v}|$: SAD (sum-of-absolute-difference)

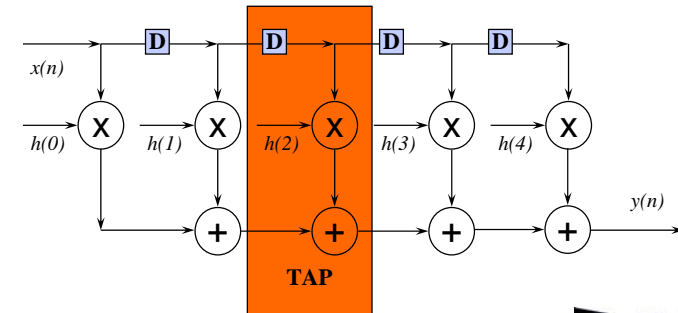
ARCHI'05 - 9



Exemple *Fil Rouge*

- Filtre Numérique RIF sur N points

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



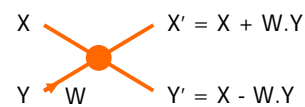
ARCHI'05 - 10



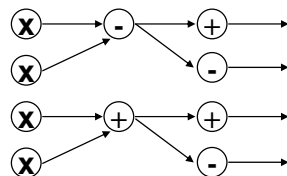
Fonctions typiques de TdSI

- FFT Fast Fourier Transform

- Butterfly (DIF)



$$\begin{cases} T_r = W_r.Y_r - W_i.Y_i \\ T_i = W_i.Y_r + W_r.Y_i \\ X'_r = X_r + T_r \\ X'_i = X_i + T_i \\ Y'_r = X_r - T_r \\ Y'_i = X_i + T_i \end{cases}$$



ARCHI'05 - 11



Fonctions typiques de TdSI

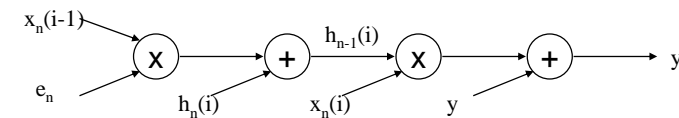
- Filtrage adaptatif LMS

$$y_n = \sum_{i=0}^{N-1} \underline{h}_n(i).x_n(i)$$

$$e_n = d_n - y_n$$

$$\underline{h}_{n+1}(i) = \underline{h}_n(i) + \mu.e_n.x_n(i) \quad \forall i \in \{0 \dots N-1\}$$

- DLMS

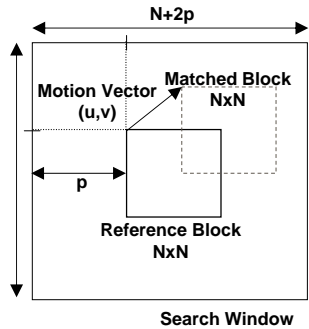


ARCHI'05 - 12



Fonctions typiques de TdSI

- Estimation de mouvement
 - Codage vidéo MPEGx, H26x



```

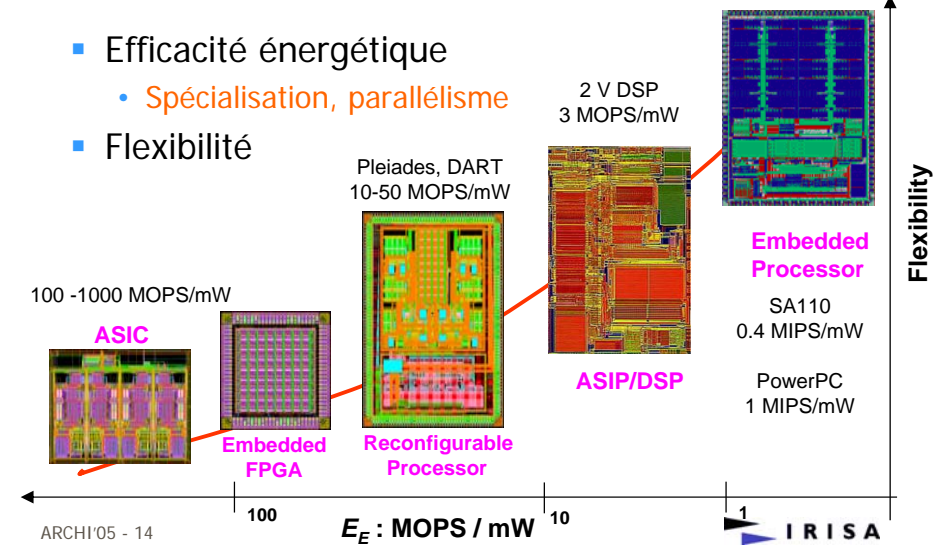
sadmin = MAXINT; mvx=0; mvy=0;
for (u=-p; u<=p; u++)
  for (v=-p; v<=p; v++) {
    sad = 0;
    for (i=0; i<N; i++) {
      for (j=0; j<N; j++) {
        sad = sad + ABS[BR(i,j)-FR(i+u,j+v)]
        /* if (sad>=sadmin) break; */
      }
    }
    if (sad<sadmin) {
      sadmin = sad; mvx = u; mvy = v;
    }
  }
    
```



ARCHI'05 - 13

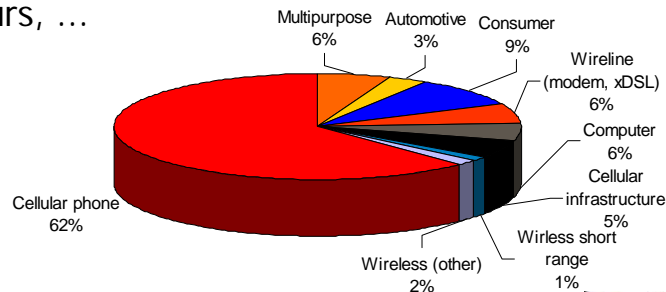
Solutions architecturales

- Efficacité énergétique
 - Spécialisation, parallélisme
- Flexibilité



Marché des DSP (1)

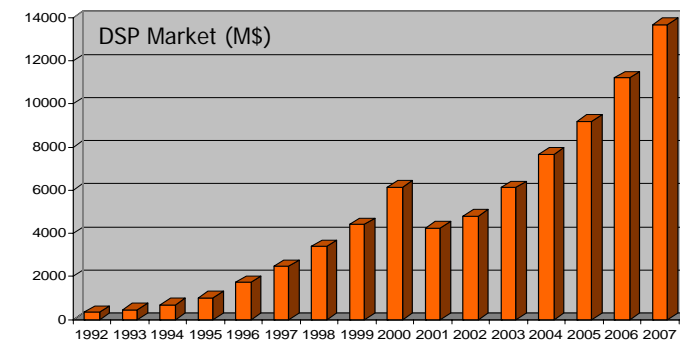
- Sans fils domine : 75% des revenus en 2004
- Systèmes grand public : DVD, TV et radio numérique, ...
- Asservissement des moteurs pour les disques durs, ...



ARCHI'05 - 15

Marché des DSP (2)

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



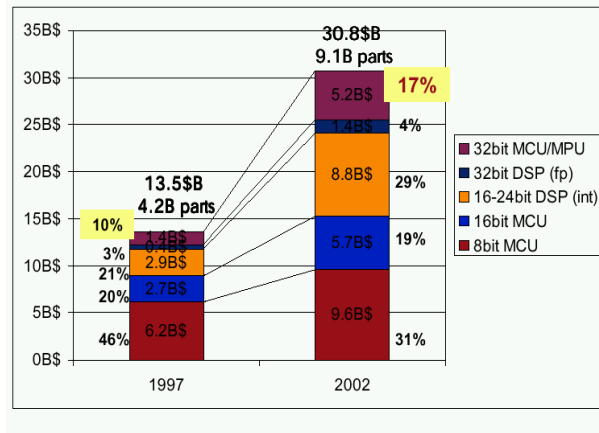
Prévisions Forward Concept Co.



ARCHI'05 - 16

Marché des DSP (3)

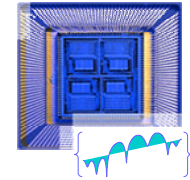
- Place dans le marché des processeurs embarqués



ARCHI'05 - 17



II. Architecture MAC/Harvard

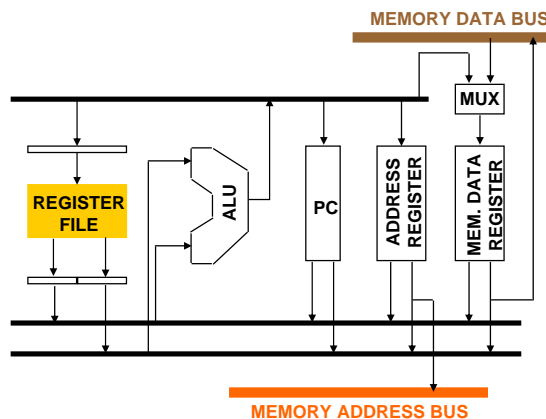


- Genèse des DSP : MAC/Harvard
- Modélisation des générations 1 et 2
Panorama des DSP
- Performances



Architecture Von Neumann

- E.g. processeurs RISC



Code exemple:
multiply & accumulate
 $r2 = r2 + \#imm * r1$

```
mov #imm,r3
mul r1,r3
add r3,r2
```

⇒ 3 instructions, plus de 3 cycles d'horloge

ARCHI'05 - 19



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,y0,a
add a,b
mov y0,*r2
inc r0
inc r1
inc r2
dec ctr
tst ctr
jnz loop
```



Exécution en N.(15 à 20) cycles

ARCHI'05 - 20

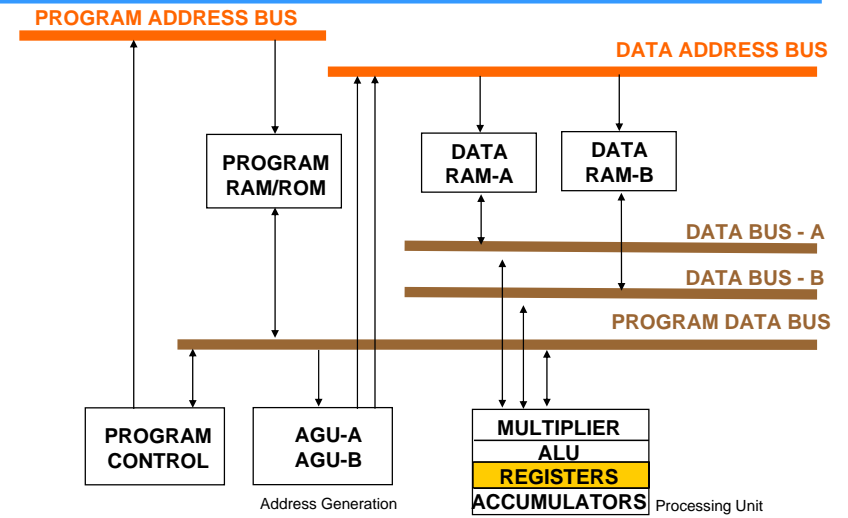


Généralités sur les DSPs

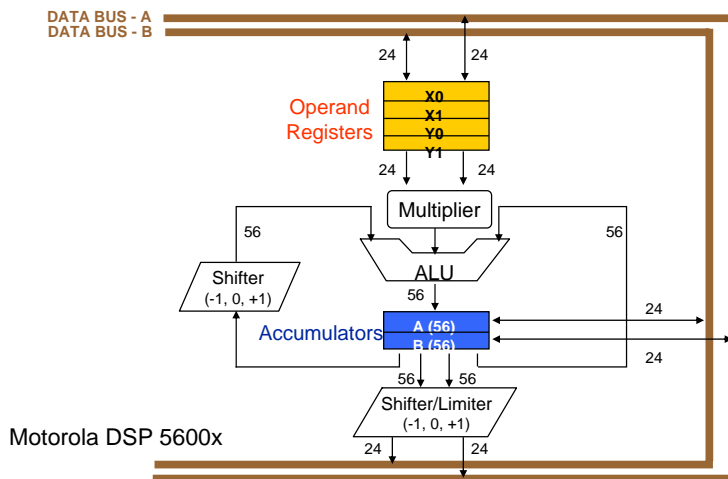
Propriétés du traitement numérique du signal	Conséquences sur les architectures
Calculs intensifs et répétitifs	<ul style="list-style-type: none"> ■ Fonctionnement pipeline ■ Architecture Harvard ■ Structures de contrôle évoluées
Primitives simples	<ul style="list-style-type: none"> ■ Unités de traitement spécialisées câblées ■ Gestion d'adressage complexes

Le tout dans un environnement temps réel

Architecture Harvard (1)

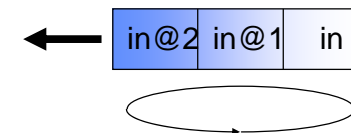


Unité de traitement



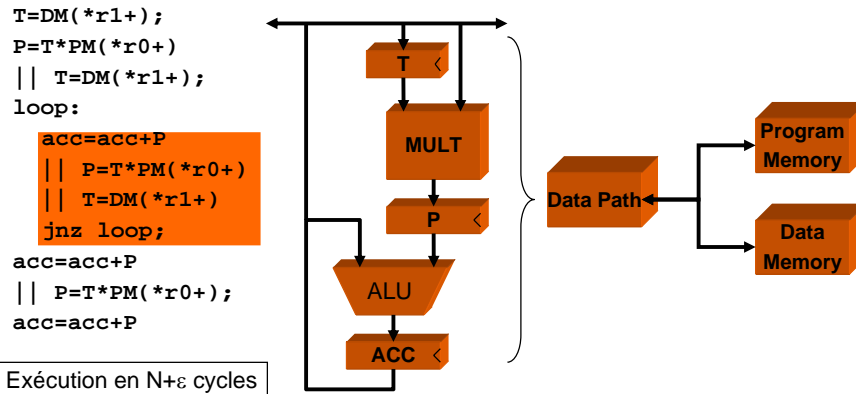
Architecture Harvard (2)

- 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) ...

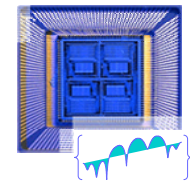


FIR sur DSP conventionnel

- Instructions complexes s'exécutant en 1 cycle



ARCHI'05 - 25



II. Architecture MAC/Harvard

- Genèse des DSP : MAC/Harvard
- Modélisation des générations 1 et 2
Panorama des DSP
- Performances



Processeur : modélisation

- Unité de contrôle** (IP : *Instruction Processor*)
 - Unité fonctionnelle (UF) qui interprète les instructions et les passe à l'unité de traitement (DP)
- 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

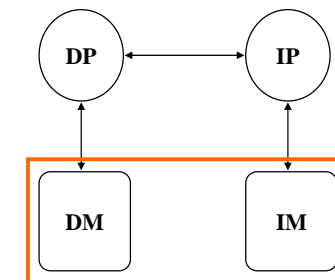
ARCHI'05 - 27

[Flynn72] [Skillicorn88]



Architecture Harvard de base

- IP : *Instruction Processor*, DP : *Data Processor*
- IM : *Instruction Memory*, DM : *Data Memory*



e.g. TMS320C10

ARCHI'05 - 28

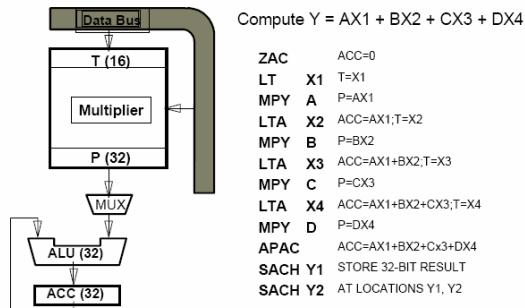
Classification de E. Lee [Lee89]



Architecture Harvard de base

- TMS320C10 (1982)

- 2N+4 cycles
- 2N+4 instructions!



[Verbauwhede00]

ARCHI'05 - 29



Optimisations des performances

Comment améliorer l'architecture Harvard ?

- Transformations du diagramme d'états
 - réarrangement des états pour augmenter le temps de cycle
 - exécution en parallèle d'états (e.g. Store Results et Return State)
- Pipeline de l'exécution des états
 - mise en parallèle d'états appartenant à des instructions successives
- Augmentation du nombre de FU
 - plusieurs DP peuvent exécuter des instructions en parallèle sur des données indépendantes

ARCHI'05 - 30



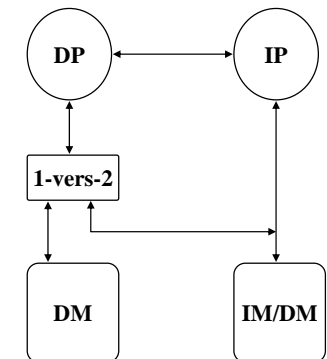
Interconnexions entre FUs

- Les interconnexions suivantes sont valables pour toutes les FUs précédentes (DP, IP, IM, DM).

- 1 - vers - 1
 - une FU est connectée à une autre FU
- n - vers - n
 - une FUi d'un ensemble de FUs est connectée à une autre FUi
- 1 - vers - n
 - une FU est connectée à n autres FUs d'un ensemble de FUs
- n - par - n
 - toute FUi d'un ensemble est connectée à chaque autre FUj

Modification 1

- Autorisation de mémorisation de données dans l'IM
- En un cycle : *fetch* deux opérandes de la mémoire, exécute MAC, écriture du résultat en I/O ou mémoire



e.g. AT&T DSP32 et DSP32C

ARCHI'05 - 32

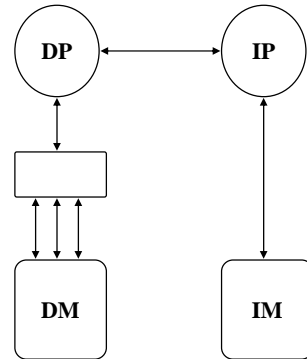


ARCHI'05 - 31



Modification 2

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



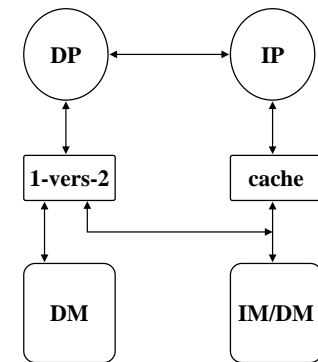
e.g. Fujitsu MB86232 (3 ports en mémoire interne)

ARCHI'05 - 33



Modification 3

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



TMS320C25 : cache 1 instruction (boucles)
 DSP16 : cache 15 instructions
 ADSP-2100 : cache 16 instructions

ARCHI'05 - 34



FIR sur DSP conventionnel

- TMS320C2x (1986)

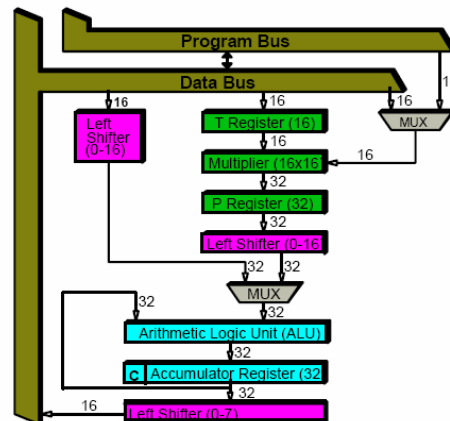
ZAC || LTD;
 LTD || MPY;

RPTK N-1

MACD

APAC || MPY;
 APAC;

Exécution en N cycles

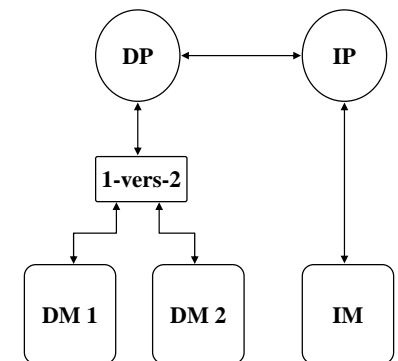


ARCHI'05 - 35



Deuxième génération (Modification 4)

- Deux mémoires données DM séparées
- En un cycle : *fetch* deux opérandes et une instruction si le cycle mémoire est le même que le cycle d'instruction de base

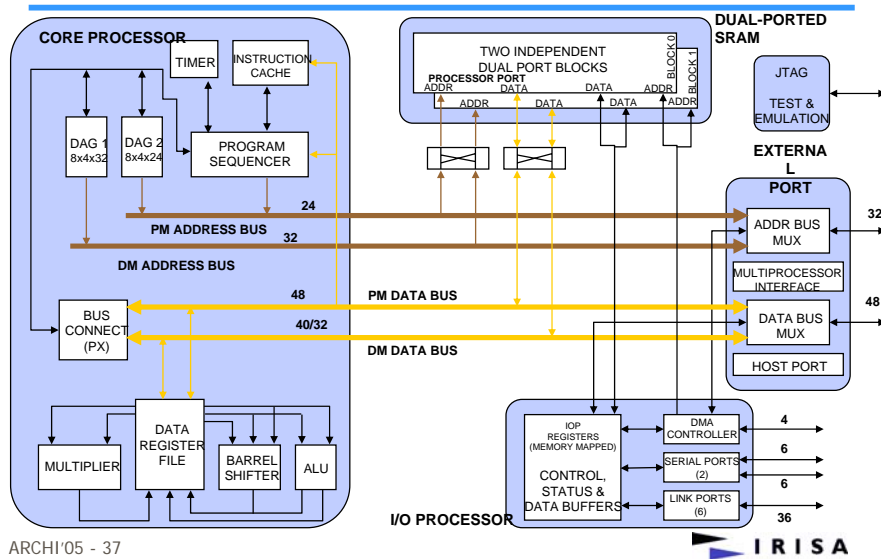


Motorola DSP 56001 et 96002
 TMS320C30, C40, C50
 ADSP-2100

ARCHI'05 - 36

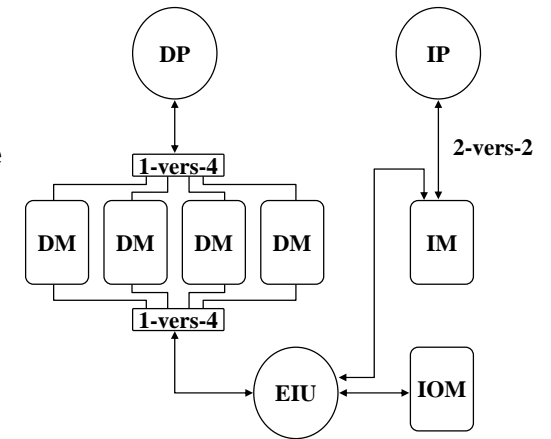


Architecture Analog Devices SHARC



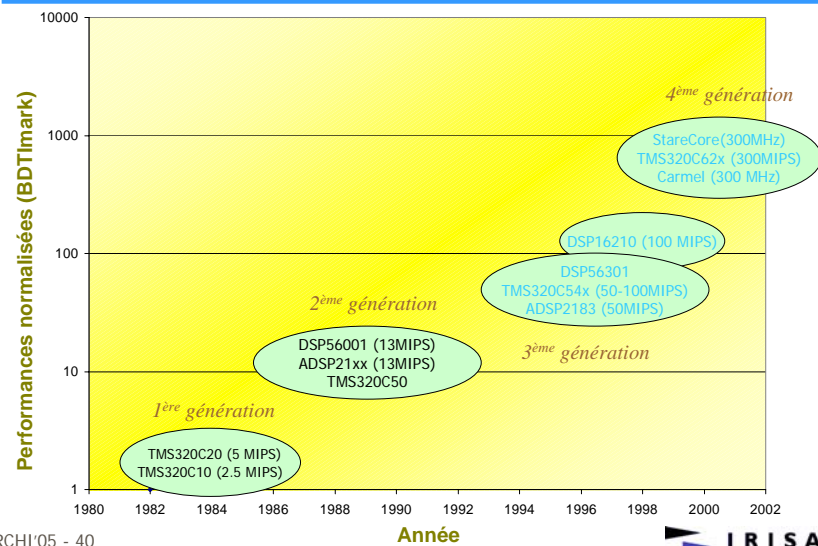
Modification 5

- Bancs mémoire multiples
- Instruction multi-opérandes en parallèle avec accès I/O
- Problème de partition des données sur un grand nombre de bancs (FFT)



Hitachi DSPi (6 bancs mémoire)

Généralisations



Caractéristiques complémentaires

- Localisation des données
 - Modèle registre-mémoire, Load-Store
- Codage des instructions complexes
 - Stationnarité temporelle ou par les données
- Format des instructions
 - Type d'encodage, compromis consommation/efficacité
- Structures de contrôle
 - Boucles matérielles, branchement
- Modes d'adressage

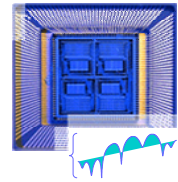
Résumé (1)

- Calcul →
- Précision
- Bande passante mémoire
- FUs spécialisées pour les fonctions classiques de TdSI en parallèle, MAC simple cycles
- Bits de garde, saturation, arrondis, multiplication 16-24 bits, addition 32-40 bits
- Harvard, bancs et bus adresses/données multiples

Résumé (2)

- Accès aux données →
- Localité temporelle d'exécution
- Flots de données (streaming)
- Temps réel
- Modes d'adressage complexes
- Boucles matérielles (zero-overhead), caches d'instructions spécifiques
- Pas de cache de données, DMA puissants, mémoires SRAM internes
- Gestion des interruptions

II. Architecture MAC/Harvard



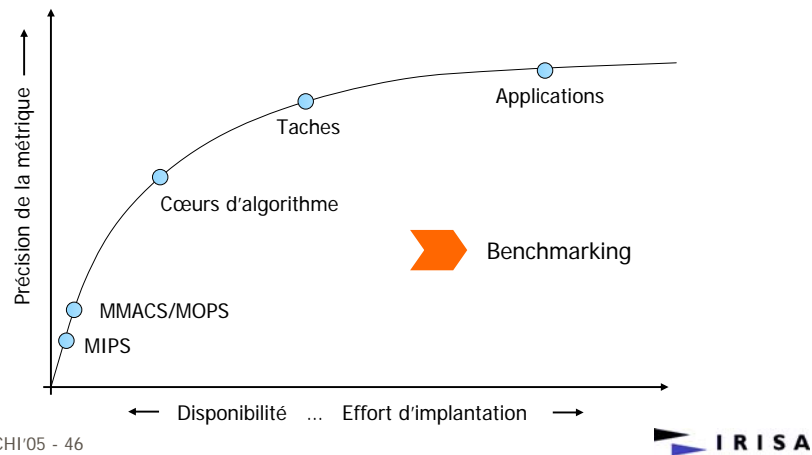
1. Genèse des DSP : MAC/Harvard
2. Modélisation des générations 1 et 2
Panorama des DSP
3. Performances

Puissance de calcul

- MIPS (Millions of instructions per second)
 - Mesure facile mais...
 - ...peu représentatif des performances réelles
 - VLIW, génération des adresse
- MOPS (Millions of operations per second)
- MMACS (Millions of MAC per second)
 - Prise en compte de l'aspect traitement du signal
 - Les autres opérations ne sont pas prises en compte
- MOPS/W : Efficacité énergétique

Métriques de performance

■ Comparaison précision - effort d'implantation



Contenu d'un benchmark

■ Différents niveaux

- Applications complète
 - Système de vidéo-conférence, émetteur/récepteur radio-mobile, player MP3 ...
- Taches d'une application
 - Codeur ou décodeur audio ou vidéo, codeur de parole, décodeur de Viterbi
- Cœurs d'algorithme
 - FIR, IIR, FFT, Control, ..
- Opérations
 - ADD, Mult/MAC, Décalage, Transfert mémoire

ARCHI'05 - 47



BDTImark2000

<http://www.bdti.com>

■ Basé sur des cœurs d'algorithme de traitement du signal

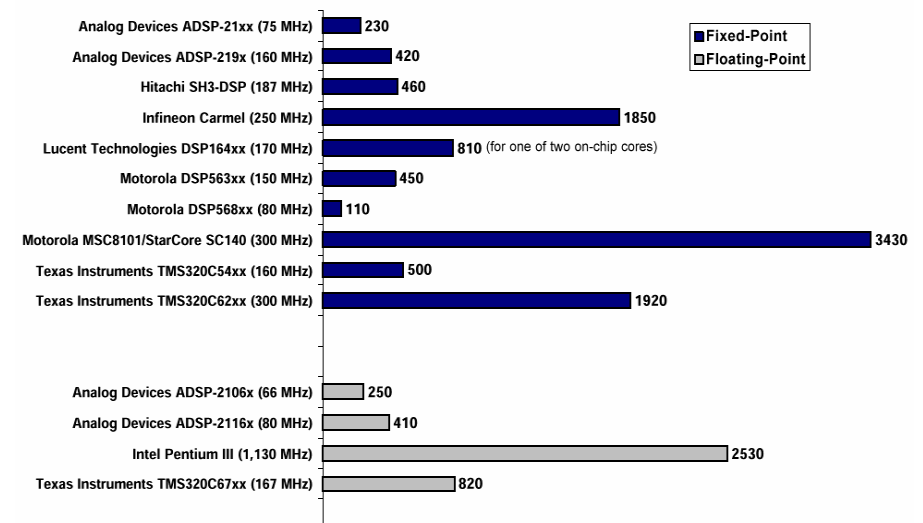
- Filtre FIR réel, traitement par blocs
- Filtre FIR réel, traitement par échantillons
- Filtre FIR complexe, traitement par blocs
- Filtre adaptif LMS
- Filtre IIR, 2 cellules d'ordre 2 cascadiées
- Produit de vecteurs
- Addition de vecteurs
- Recherche de la valeur maximale d'un vecteur
- Décodeur de Viterbi
- FFT 256 points
- Manipulation de bits

■ Code optimisé manuellement

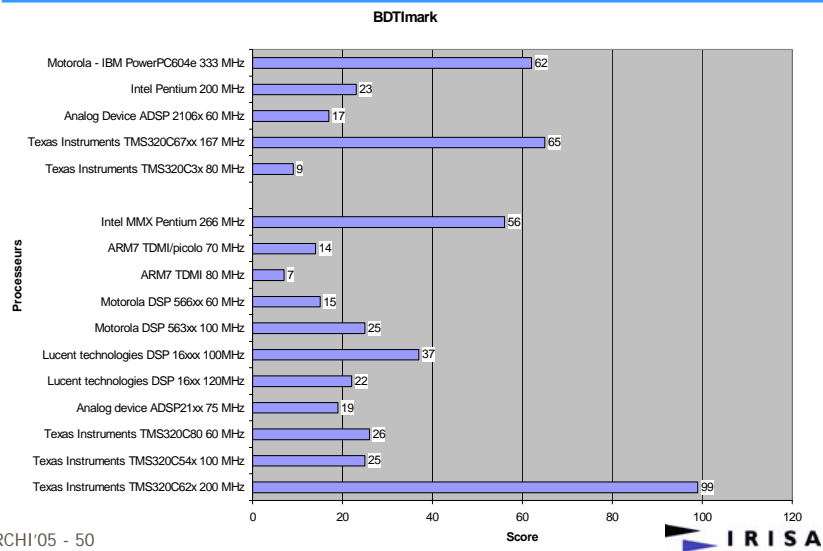
ARCHI'05 - 48



BDTIMARK

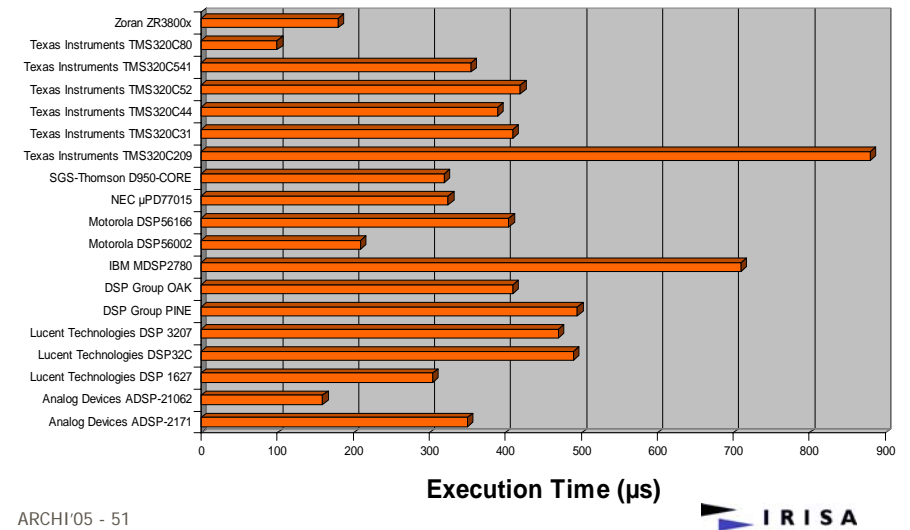


BDTIMARK



ARCHI'05 - 50

Temps d'exécution (FFT)

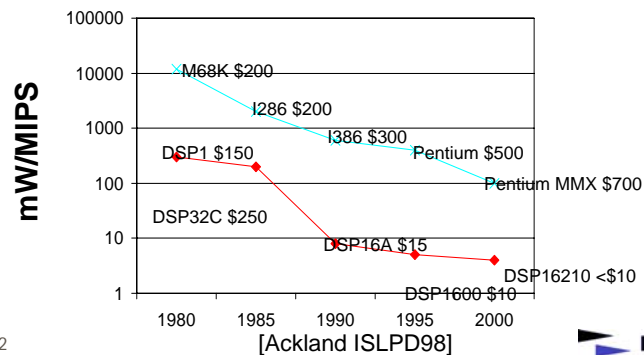


ARCHI'05 - 51

Coût

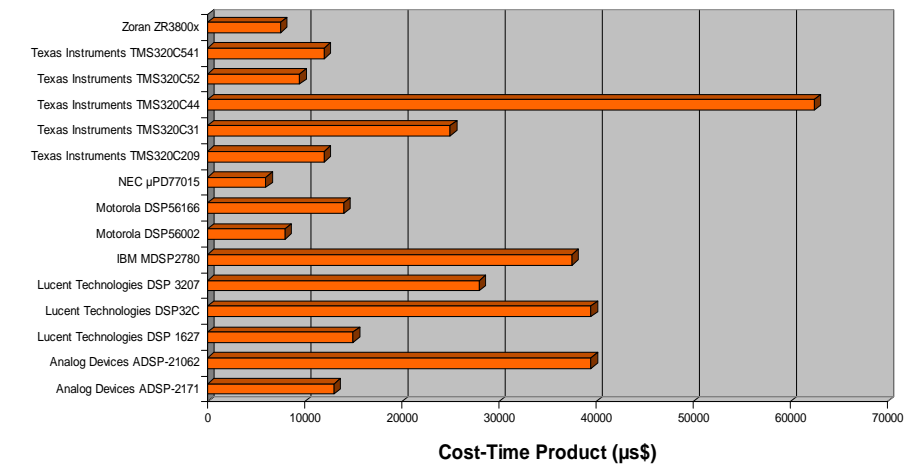
■ Important pour une production à fort volume

- Faible coût
 - Les DSPs sont efficaces en MIPS/mW et en MIPS/mm²
 - Cela se paye sur la flexibilité et la programmabilité



ARCHI'05 - 52

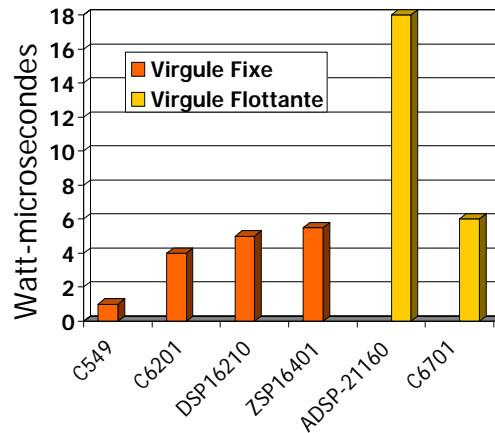
Coût - performance (FFT)



ARCHI'05 - 53

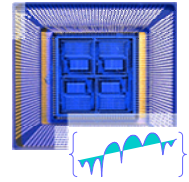
Énergie

▪ Filtrage numérique RIF



- C549
 - 100 MHz, 2.5V
- C6201
 - 200 MHz, 1.8V
- DSP16210
 - 100 MHz, 3.3V
- ZSP16401
 - 200 MHz, 2.5V
- ADSP-21160
 - 100 MHz, 2.5V
- C6701
 - 167 MHz, 1.8V

ARCHI'05 - 54



III. Évolution des DSP

1. DSP conventionnels améliorés
2. Capacités SIMD
3. DSP hybride MCU
4. VLIW
5. Superscalaire
6. Comparaison de performances



Évolution des DSP

- Améliorer les performances au delà de l'augmentation liée à la vitesse d'horloge ?
- Augmenter le parallélisme
 - Augmentation du nombre d'opérations exécutées dans chaque instruction
 - Augmentation du nombre d'instructions exécutées par cycle d'horloge
- Ajouter des unités spécialisées
- Améliorer le contrôle

ARCHI'05 - 56



Plus d'opérations par instruction

- Augmenter le nombre d'opérations qui peuvent être exécutées dans chaque instruction
 - Ajouter des unités d'exécution
 - multiplieur, additionneur, ...
 - jeu d'instruction à enrichir
 - taille de l'instruction à augmenter
 - bus de transfert mémoire à augmenter
 - Augmenter les capacités SIMD (ou SWP)
- Dans le même ordre d'idées
 - Utiliser des unités fonctionnelles spécialisées
 - Utiliser un coprocesseur
 - Architectures hybrides DSP/MCU

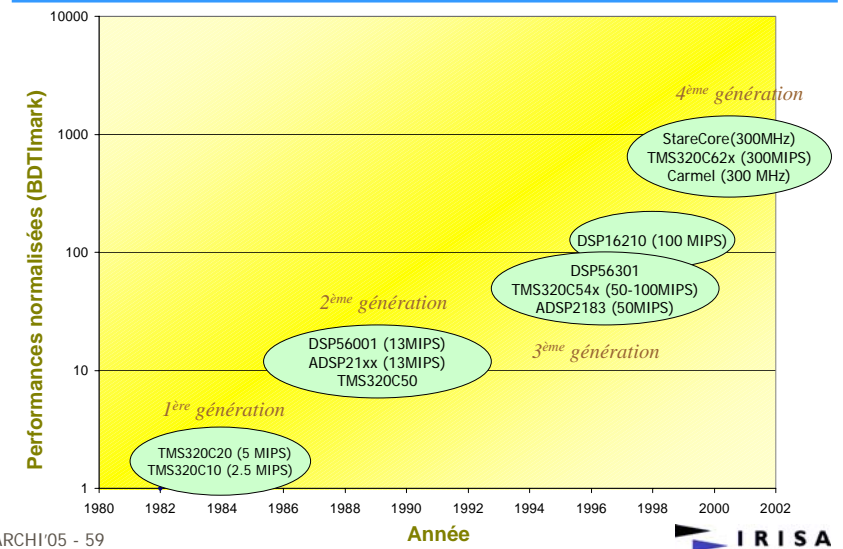
ARCHI'05 - 57



Plus d'instructions par cycle

- L'objectif est ici de profiter du parallélisme au niveau instruction (ILP) d'une application
- Deux techniques sont bien connues :
 - VLIW : empaquetage de plusieurs instructions de type RISC dans une seule "super-instruction"
 - Superscalaire : exécution parallèle de plusieurs instructions sélectionnées dynamiquement par le processeur

Généralisations

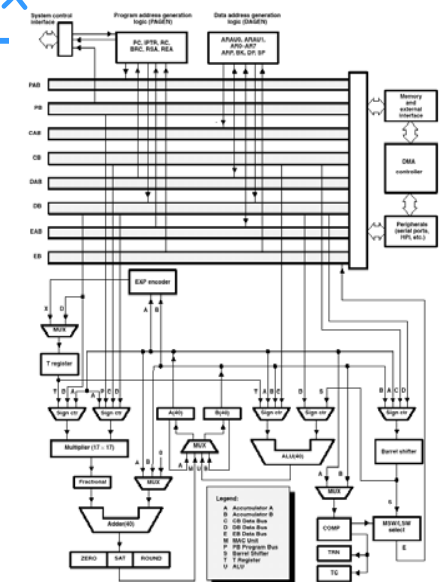


DSP conventionnels améliorés

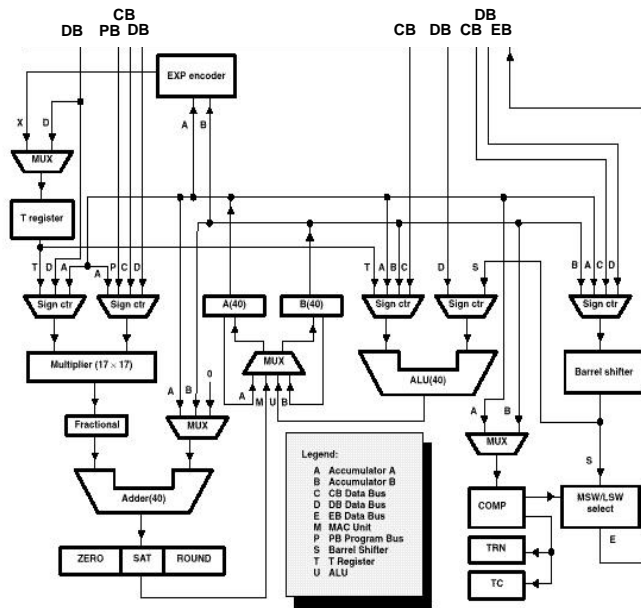
- Troisième génération (1995)
- Ajout d'unités spécialisés ou utilisation de coprocesseurs
 - TMS320C54x
- Architectures multi-MAC
 - Lucent DSP 16xxx
- Multiprocesseurs sur une puce
 - Principalement pour image/vidéo
 - TMS320C80, MC68356

Architecture C54x

- C54x
 - 40-100-160 MIPS
 - 1000-2000-3000 MIPS/W
 - 17x17b multiplieur, 40b ALU, 40b adder, ACS unit
 - Viterbi, LMS, **FIRS**, FFT
 - 60% of cellular handsets
 - \$5 for C5402 100MIPS - \$75
 - Dual cores (C5420)



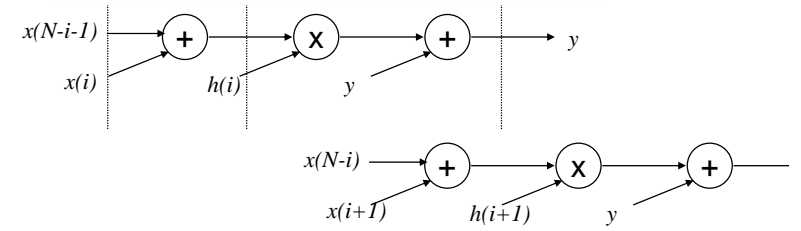
C54x



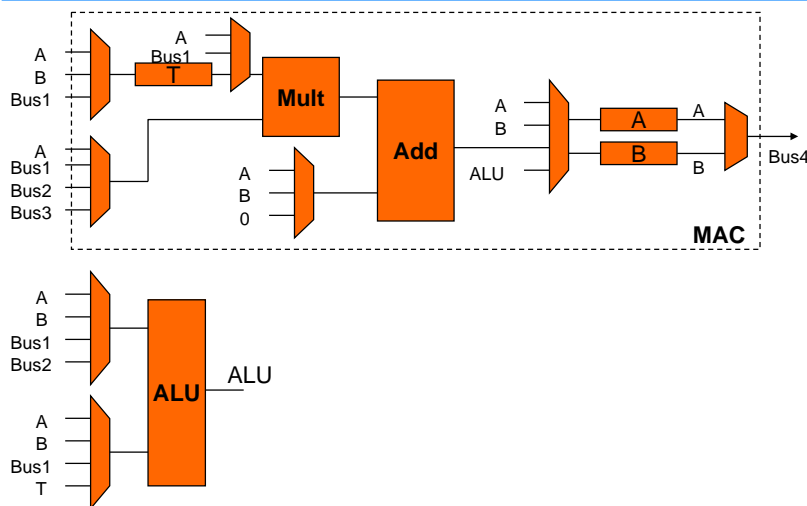
C54x

- FIR symétrique

$$y(n) = \sum_{i=0}^{\frac{N}{2}-1} a_i [x(n-i) + x(n-N+i+1)]$$



C54x



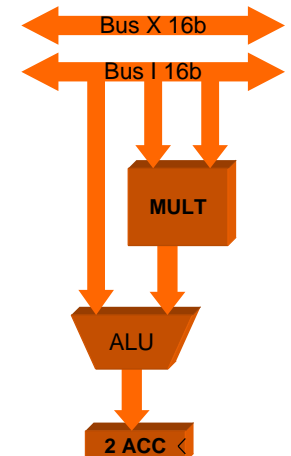
Architectures multi-MAC

- FIR sur DSP conventionnel

- e.g. Lucent DSP 16xx

```
DO N TIME:
  acc=acc+P
  // P=X*Y
  // Y=*r0++
  // X=*r1++
```

Exécution en N cycles



Architectures multi-MAC

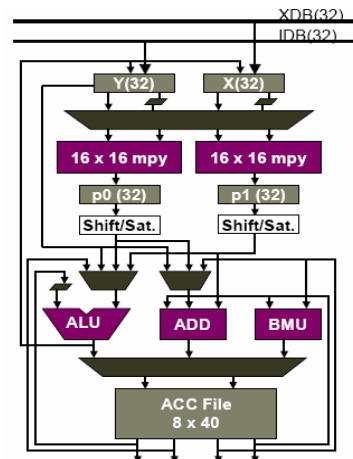
FIR sur DSP dual-MAC

- e.g. Lucent DSP 16xxx
 - 400 MIPS, 380mW
 - \$5 - \$80

```

DO N/2 TIME:
  acc=acc+P0+P1
  // P0=X0*Y0
  // P1=X1*Y1
  // Y=*r0++
  // X=*r1++
    
```

Exécution en N/2 cycles



Architecture C55x

C55x

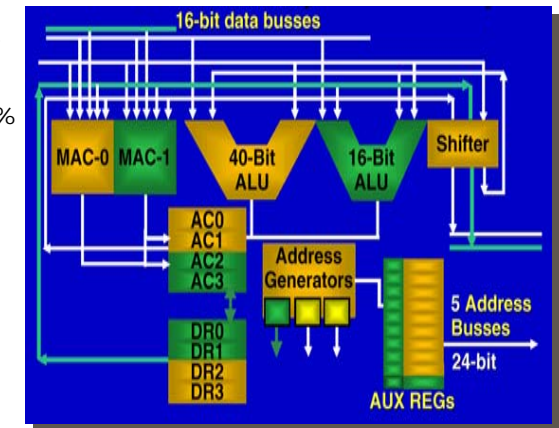
- Jusqu'à 20 MIPS/mW
- C54x
 - Consommation -85%
 - Performances 5x

C5510

- 160 MHz, 320 MIPS
- 80mW
- 4 MIPS/mW
- 160 KW SRAM

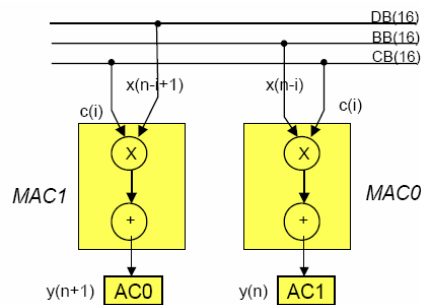
C5502

- 400 MIPS
- 160 mW
- 2.5 MIPS/mW



Architecture C55x

FIR symétrique

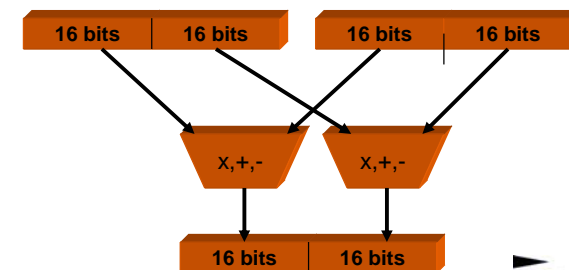


Capacités SIMD (ou SWP)

- Opérations parallèles sur différentes largeurs de chemins de données (16 bit, 8 bit, ...)

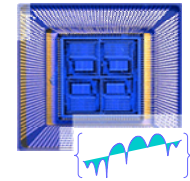
 - Split unités d'exécution
 - Unités d'exécution multiples

- Exemples
 - Lucent DSP16xxx, ADI ADSP-2116x, ADI TigerSHARC, TI C64x



Avantages et inconvénients

Méthode	Avantages	Inconvénients
Augmenter le nombre d'UE	Augmentation limitée de la consommation, du coût et de la densité de code Compatibilité maintenue	Augmentation importante de la complexité Difficulté à programmer et à compiler Perspectives limitées
Augmenter les capacités SIMD	Gain en performance très important pour des traitements par blocs Modifications architecturales limitées	Nécessité d'avoir un parallélisme au niveau des données important Consommation mémoire importante
Matériel dédié	Gain en performance important	Nécessite une bonne connaissance de l'application



III. Évolution des DSP

1. DSP conventionnels améliorés
2. Capacités SIMD
3. DSP hybride MCU
4. VLIW
5. Superscalaire
6. Comparaison de performances

Architecture hybride DSP/MCU

- Associer des fonctionnalités MCU...
 - gestion efficace des ITs
 - manipulation de bits
 - exécution conditionnelle
- à des fonctionnalités DSP,
 - calcul intensif
- pour obtenir un code performant et efficace
 - limiter la taille du code et donc du circuit

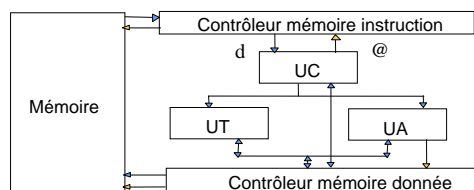
Architecture hybride DSP/MCU

- Méthodes de couplage

Méthode	Avantages	Inconvénients
Multiprocesseur	2 jeux d'instructions ≠ Les 2 cœurs travaillent en // Pas de conflits de ressources	Duplication de ressources Deux outils de développement
Coprocasseur	Les 2 cœurs travaillent en //	Modèle de programmation plus complexe Transferts de données
Extension	Modèle de programmation plus simple	Contraintes imposées par l'architecture initiale
Solution Hybride	Architecture plus "propre"	Développement plus risqué

Architecture Hybride DSP/MCU

- ST100, ST122, ST140 (STMicroelectronics)
 - 32 bits MCU / 16 bits DSP
 - Fonctionnalités DSP
 - MAC, architecture Harvard, modes d'adressages complexes
 - et microcontrôleur
 - architecture load/store, large espace d'adressage, code compact



ARCHI'05 - 75



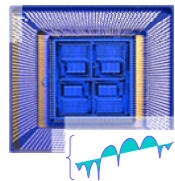
ST 140

- Compatible ST100
 - Compatible ST122 (dual-MAC)
 - Flexible 16-bit/32-bit Load/Store architecture
 - 4-MAC/4-ALU
 - Two data clusters
 - Double data memory bandwidth
- ST140-DSP Core in 0.13µm process
 - Frequency up to 600 MHz
 - Intensive Processing 2.4 GMAC/s
 - Large Data Bandwidth 9.6 Gbytes/s
 - Power Consumption down to 0.180 mW/MHz
 - 100 mW, 24 MMAC/mW

ARCHI'05 - 76



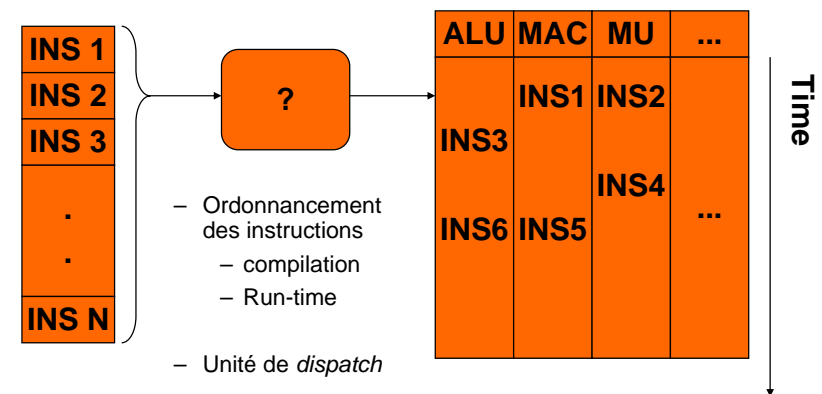
III. Évolution des DSP



1. DSP conventionnels améliorés
2. Capacités SIMD
3. DSP hybride MCU
4. VLIW
5. Superscalaire
6. Comparaison de performances



Parallélisme niveau instructions



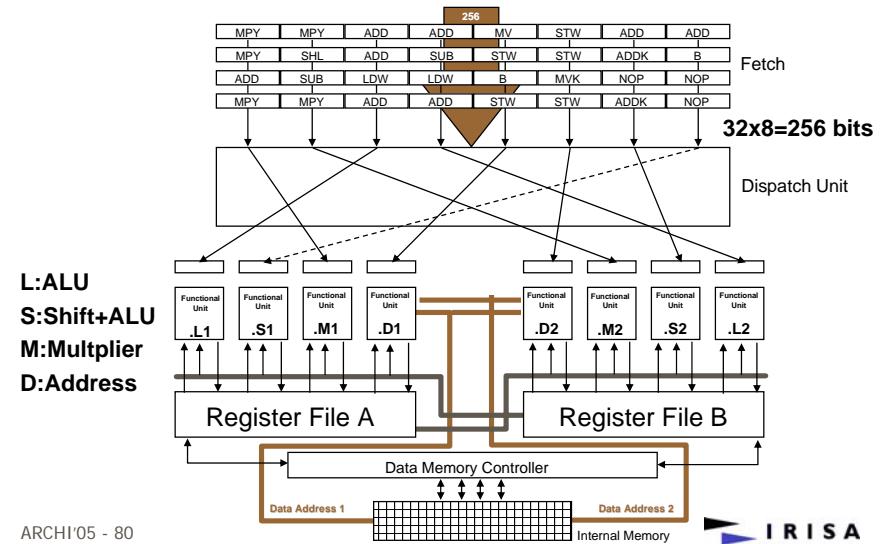
ARCHI'05 - 78



Very Long Instruction Word

- **Caractéristiques**
 - Plusieurs instructions par cycle, empaquetées dans une "super-instruction" large
 - Architecture plus régulière, plus orthogonale, plus proche du RISC
 - Jeu de registres uniforme, plus large
- **Exemples**
 - TI TMS320 C62x et C64x
 - ADI TigerSHARC ADS-TS20x
 - Freescale (Motorola) MSC71xx et MSC81xx
 - StarCore SC1400 Agere/Motorola (DSP core)

VLIW : C62xx



Texas Instruments TMS 320C6x

- VLIW CPU with eight functional units (RISC-like code)
- 2400 MIPS @ 300MHz
- 1M-bit on-chip memory
- 32-bit ext. mem. interface
- 2 enhanced-buffered serial ports
- 16-bit host access port (Host processor access to on-chip data memory)
- Flexible PLL clock generator (x ext. clock rate for 2 or 4)
- Price: \$9-\$102
- CPU
 - Two sets of functional units including:
 - Two multipliers
 - Six arithmetic logic units (ALUs)
 - 32 registers with 32-bit word-length each
 - Data-addressing units .D1 and .D2 exclusively responsible for data transfers between memory and the register files
 - 8-/16-/32-bit data support
 - 40-bit arithmetic options
 - Saturation and normalisation
 - Bit-field manipulation and instruction: extract, set, clear, bit counting

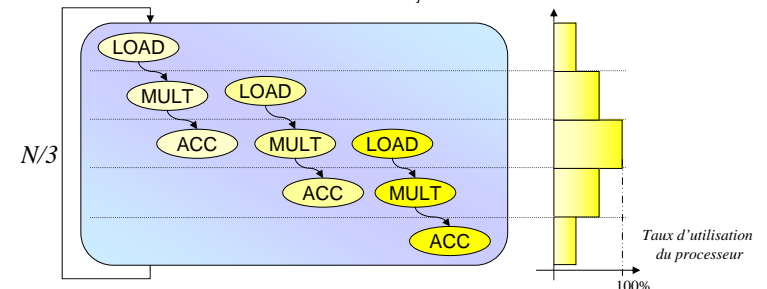
Exploitation du parallélisme

- **Déroulage des boucles : augmente l'ILP**

```

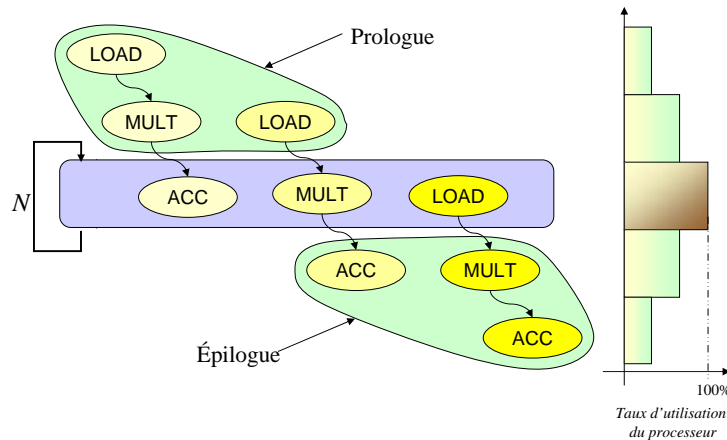
For(i=0;i<N;i++)
{
  ACC=ACC + x[i].h[i]
}

For(i=0;i<N;i+=3)
{
  ACC=ACC + x[i].h[i]
  ACC=ACC + x[i+1].h[i+1]
  ACC=ACC + x[i+2].h[i+2]
}
    
```

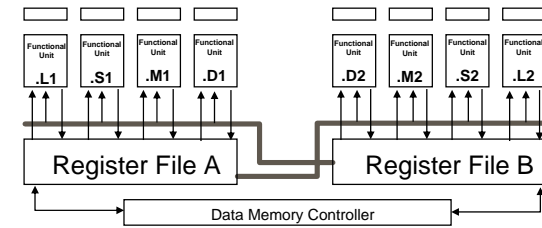


Exploitation du parallélisme

- Pipeline logiciel : maximiser l'IPC
 - Optimisation du code assembleur



C6x



Instruction	Description	Nombre de cycles	Unités .M	Unités .L	Unités .S	Unités .D
LDH	Charge une donnée 16 bits venant de la mémoire dans un registre	5	-	-	-	.D1 .D2
LDW	Charge deux données 16 bits consécutives venant de la mémoire	5	-	-	-	.D1 .D2
MPY	Multiplication entre 2 registres, résultat dans un troisième	2	.M1 .M2	-	-	-
ADD	Addition	1	-	.L1 .L2	.S1 .S2	.D1 .D2
SUB	Soustraction	1	-	.L1 .L2	.S1 .S2	.D1 .D2
B	Branchement	6	-	-	.S1 .S2	-
NOP	aucune opération	1	-	-	-	-

Exploitation du parallélisme

- Filtre FIR sur DSP VLIW C6x

```

x[0] = input;           // Update most recent sample
acc = 0;                // Zero accumulator
for (i=0; i<8; i++)    // 8 taps
{
    prod = h[i] * x[i]; // perform Q.15 multiplication
    acc = acc + prod;   // Update 32-bit accumulator
}
output = (short) (acc >> 15); // Cast output to 16 bits
    
```

```

1 Start    MVKL .S2 8, B0      ; Initialize the loop counter (B0) to 8
2          MVKL .S1 0, A5     ; Initialize the accumulator (A5) to 0
3 Loop    LDH .D1 *A8++,A2    ; Load x(i) in A2
4          LDH .D1 *A9++,A3    ; Load h(i) in A3
5          NOP 4              ; LDH has a latency of 5 cycles
6          MPY .M1 A2,A3,A4    ; Multiply x(i) and h(i) in A4
7          NOP                ; MPY has a latency of 2
8          ADD .L1 A4,A5,A5    ; Add A4 to the accumulator A5
9 [B0]    SUB .L2 B0,1,B0     ; Sub 1 to the counter B0
10 [B0]    B .S1 loop        ; Branch to loop if B0 <> 0
    
```

Méthodes d'optimisation

- Depuis le code "RISC"
- Instructions en parallèle
- Enlever les NOP
- Déroulage de boucle
- Pipeline logiciel
- Lecture mémoire de paquets de données
 - 2 LDH → 1 LDW
 - 2 LDW → 1 LDDW

Code non optimisé

```

loop:
  ldh.d1    *A8++,A2
  ldh.d1    *A9++,A3
  nop       4
  mpy.m1    A2,A3,A4
  nop
  add.l1    A4,A6,A6
  sub.l2    B0,1,B0
[b0] b.s1   loop
  nop       5
    
```

- Code RISC
- 40 itérations
- 16*40 = 640 cycles

Instructions parallèles

```

loop:
  ldh.d1    *A8++,A2
  || ldh.d2  *B9++,B3
  nop       4
  mpy.m1x   A2,B3,A4
  nop
  add.l1    A4,A6,A6
  sub.l2    B0,1,B0
[b0] b.s1   loop
  nop       5
    
```

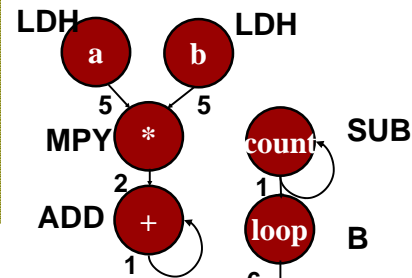
- Parallélisme ?
- 40 itérations
- 15*40 = 600 cycles

Remplacer les NOP Not Optimized Properly

```

loop:
  ldh.d1    *A8++,A2
  || ldh.d2  *B9++,B3
  nop       4
  mpy.m1x   A2,B3,A4
  nop
  add.l1    A4,A6,A6
  sub.l2    B0,1,B0
[b0] b.s1   loop
  nop       5
    
```

- Instructions à la place des NOP
- Dépendances inter-instructions



Remplacer les NOP

```

loop:
  ldh.d1    *A8++,A2
  || ldh.d2  *B9++,B3
  sub.l2    B0,1,B0
[b0] b.s1   loop
  nop       2
  mpy.m1x   A2,B3,A4
  nop
  add.l1    A4,A6,A6
    
```

- Instructions à la place des NOP
- 40 itérations
- 8*40 = 320 cycles

Déroutage de boucles

- Déroutage + ré-ordonnancement des instructions

Cycle	.D1	.D2	.L1	.L2	.M1	.M2	.S1	.S2	NOP
1	LDH	LDH							
2	LDH	LDH							
3	LDH	LDH							
4	LDH	LDH							
5	LDH	LDH							
6	LDH	LDH			MPY				
7	LDH	LDH			MPY				
8	LDH	LDH	ADD		MPY				
9	LDH	LDH	ADD		MPY				
10			ADD		MPY				
11			ADD		MPY				
12			ADD		MPY				
13			ADD		MPY				
14			ADD		MPY				
15			ADD		MPY				
16			ADD						

- N=40 itérations
- 7 + (N-7) + 7 = 47 cycles
- 47 instructions VLIW!

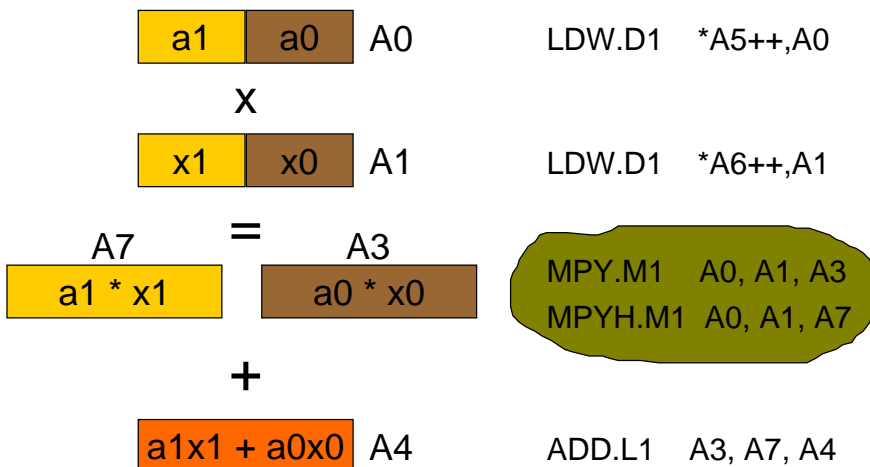
Pipeline logiciel

- Réintroduction de la boucle sur le motif répété

Cycle	.D1	.D2	.L1	.L2	.M1	.M2	.S1	.S2	NOP
1	LDH	LDH							
2	LDH	LDH			SUB				
3	LDH	LDH			SUB		B		
4	LDH	LDH			SUB		B		
5	LDH	LDH			SUB		B		
6	LDH	LDH			SUB	MPY	B		
7	LDH	LDH			SUB	MPY	B		
8	LDH	LDH	ADD	SUB	MPY		B		
9			ADD		MPY				
10			ADD		MPY				
11			ADD		MPY				
12			ADD		MPY				
13			ADD		MPY				
14			ADD						
15			ADD						
16									

- 47 cycles
- 15 instructions

Accès mémoire multiples



Accès mémoire multiples

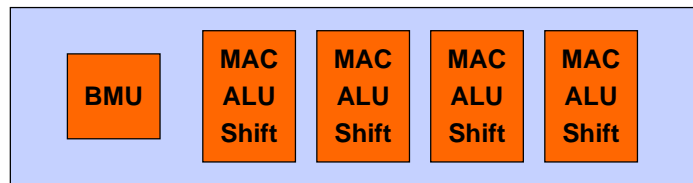
- Lecture de 4 données + 2 MAC en parallèle

Cycle	.D1	.D2	.L1	.L2	.M1	.M2	.S1	.S2	NOP
1	LDW	LDW							
2	LDW	LDW							
3	LDW	LDW							
4	LDW	LDW							
5	LDW	LDW							
6	LDW	LDW			MPY	MPYH			
7	LDW	LDW			MPY	MPYH			
8	LDW	LDW	ADD	ADD	MPY	MPYH			
9			ADD	ADD	MPY	MPYH			
10			ADD	ADD	MPY	MPYH			
11			ADD	ADD	MPY	MPYH			
12			ADD	ADD	MPY	MPYH			
13			ADD	ADD	MPY	MPYH			
14			ADD	ADD					
15			ADD	ADD					
16			ADD						

- N=40 itérations
- 7 + (N/2-7) + 8 = 28 cycles

StarCore SC1400 core

- Processeur VLIW 16 bits développé en commun par Agere (Lucent) et Freescale (Motorola)
- 300 MHz, faible tension
- Optimisé pour faible consommation
 - Meilleure densité de code (16 bits) que C6x
 - Pipeline plus simple (5 étages contre 11)

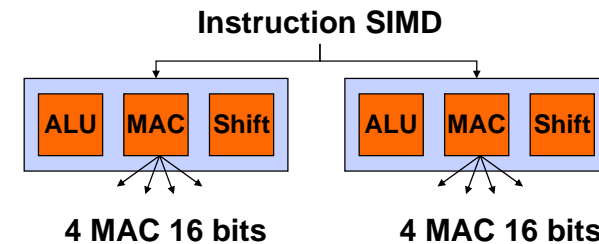


ARCHI'05 - 95



VLIW combiné au SIMD

- ADI TigerSHARC
 - Combine le VLIW au SIMD afin d'atteindre un parallélisme massif
 - SIMD hiérarchique
 - Le TigerSHARC peut exécuter 8 multiplications 16x16 en virgule fixe par cycle (4x le C62xx)



ARCHI'05 - 96



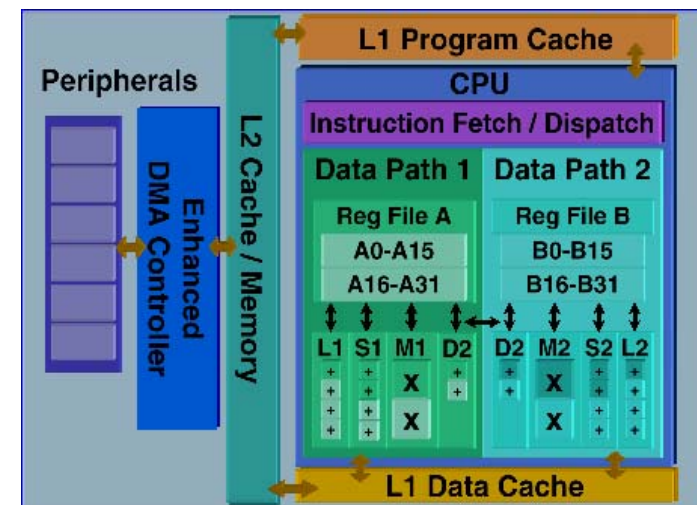
VLIW combiné au SIMD

- C64x
 - Jusqu'à 1.1 GHz, ~9 GOPS
 - Six ALUs (32/40-Bit)
 - une 32/40-Bit, deux 16-Bit, ou quatre 8-Bit opérations arithmétiques par cycle et par ALU
 - Deux multiplieurs, quatre 16x16-Bit ou huit 8x8-Bit multiplications par cycle
 - Coprocesseurs VCP (Viterbi) et TCP (Turbo)
- 'C6411: 300 MHz, \$39, 1.0 V, 250mW, 2400 MIPS, 1200 MMACS

ARCHI'05 - 97



C64x



ARCHI'05 - 98



Superscalaire

- Techniques dérivées des processeurs généraux hautes-performances
 - Prédiction de branchement
 - Cache dynamique
- Plusieurs (2-4) instructions par cycle
- Jeu d'instructions de type RISC
- Parallélisme important
- Exemple
 - LSI Logic LSI40x (ZSP400 core)
 - 4-way, 260MHz

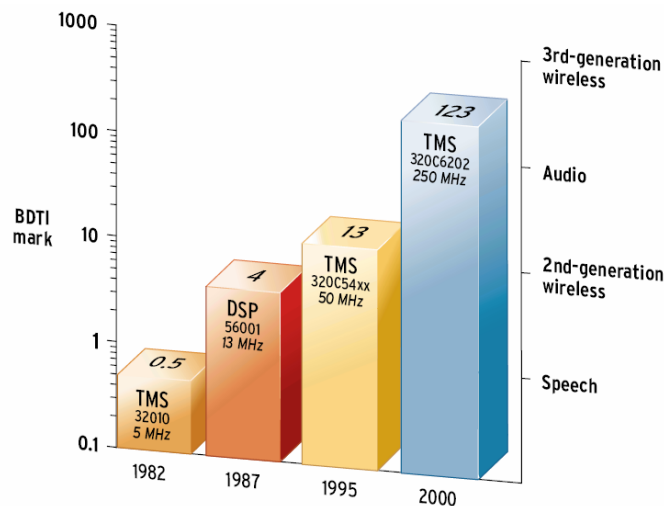
```

LOOP // FIR on LSI40x
LDDU R4, R14, 2
LDDU R8, R15, 2
MAC2.A R4,R8
AGNO LOOP
    
```

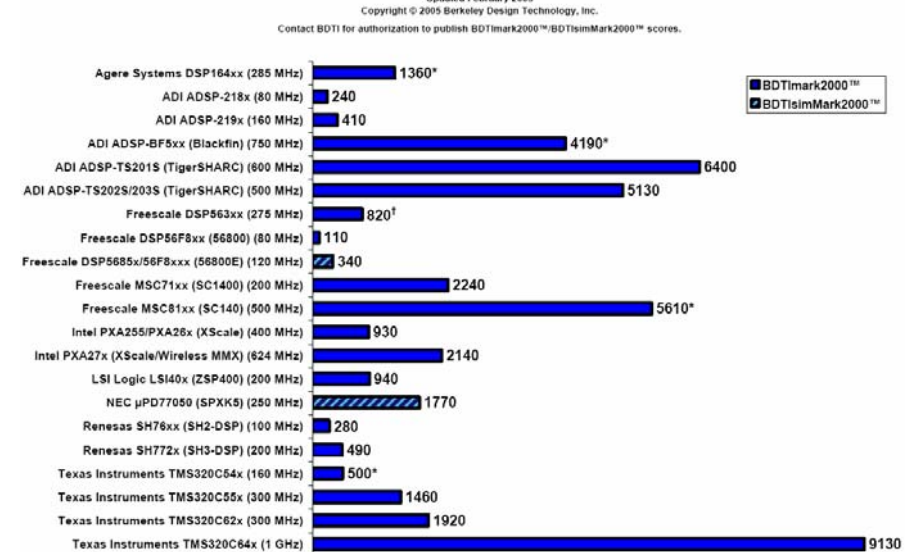
Avantages et inconvénients

Méthode	Avantages	Inconvénients
VLIW	Grand bon dans les performances Architectures plus orthogonales ⇒ meilleures cibles pour les compilateurs	Bande passante vers la mémoire importante Forte consommation Séquençement délicat Augmentation de la taille du code importante
Superscalaire	Grand bon dans les performances Architectures plus orthogonales ⇒ meilleures cibles pour les compilateurs Pas de problèmes de séquençement	Bande passante vers la mémoire importante Plus forte consommation Temps d'exécution difficilement prédictible

Génération : bilan



BDTImark2000™/BDTIsimMark2000™ Scores for Fixed-Point Packaged Processors



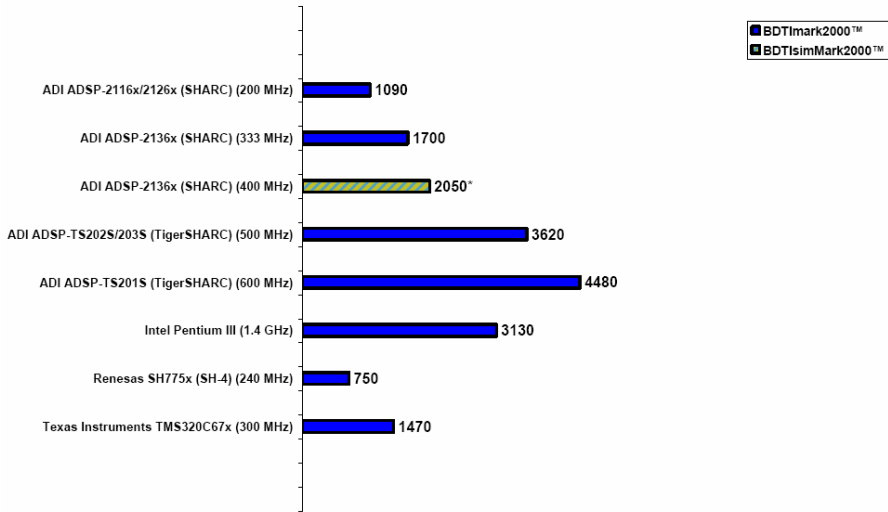
* For one core
† Benchmarked with 24-bit fixed-point data; all other processors benchmarked with 16-bit fixed-point data
‡ Projected

BDTIsimMark2000™ scores may be based on projected clock speeds. For information, visit: www.bdti.com/benchmark.html

BDTImark2000™/BDTIsimMark2000™ Scores for Floating-Point Packaged Processors

Updated November 2004
Copyright © 2004 Berkeley Design Technology, Inc.

Contact BDTI for authorization to publish BDTImark2000™/BDTIsimMark2000™ scores.



*Projected
All processors benchmarked with 32-bit floating-point data

BDTImark2000™ scores may be based on projected clock speeds. For information, visit: www.BDTI.com/benchmarks.html

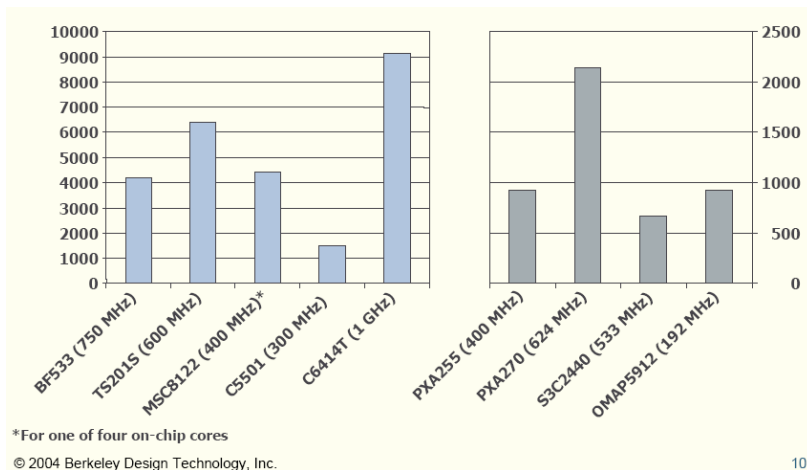


Processeurs récents (2005)

- DSP
 - Analog Devices BF53x (Blackfin)
 - Analog Devices TS20x (TigerSHARC)
 - Freescale MC1xx/81xx
 - Texas Instruments C55x
 - Texas Instruments C64x
- Processeurs applicatifs
 - Intel PXA255/26x (XScale)
 - Intel PXA27x (XScale + Wireless MMX)
 - Samsung S3C24xx (ARM9)
 - Texas Instruments OMAP591x (ARM9 + C55x)



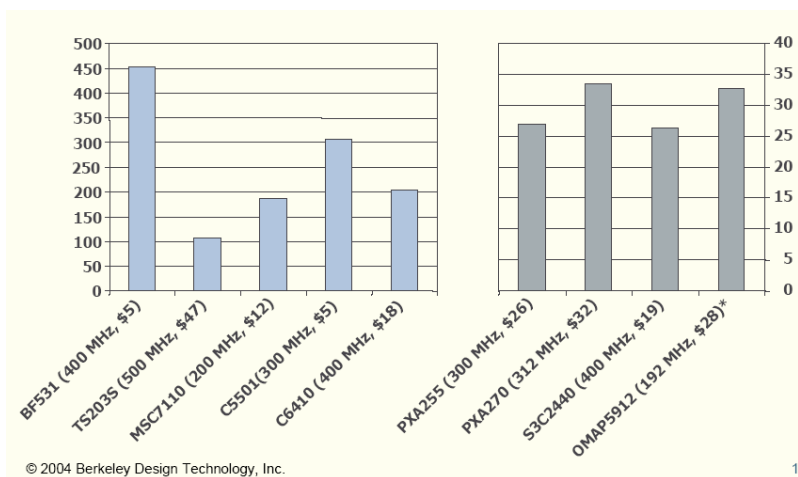
Vitesse (BDTImark2000)



*For one of four on-chip cores

© 2004 Berkeley Design Technology, Inc.

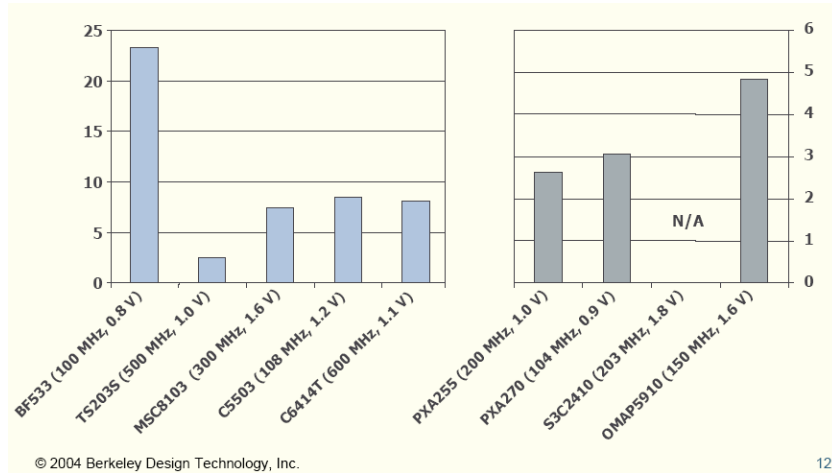
Coût - performance (mark/\$)



© 2004 Berkeley Design Technology, Inc.



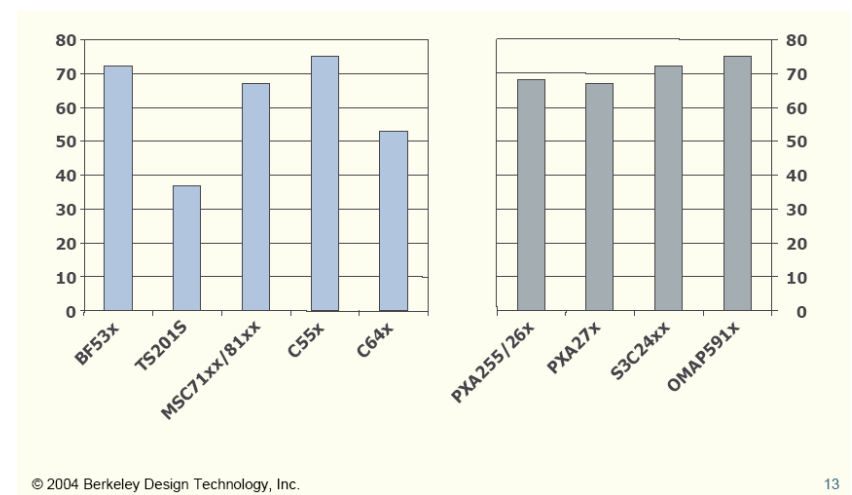
Efficacité énergétique (mark/mW)



ARCHI'05 - 108



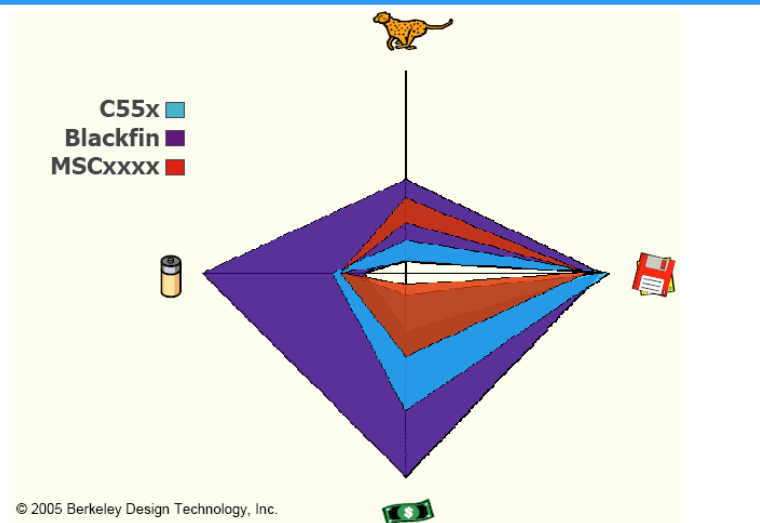
Efficacité mémoire (memmark)



ARCHI'05 - 109



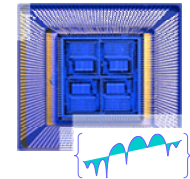
BDTIMark



ARCHI'05 - 110



III. Flot de développement

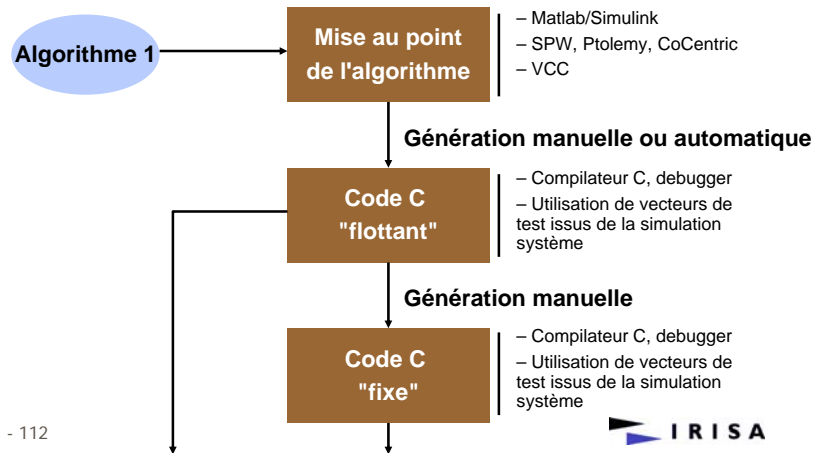


1. Flot général de développement
2. Compilation pour DSP
3. Arithmétique virgule fixe



Flot de développement général

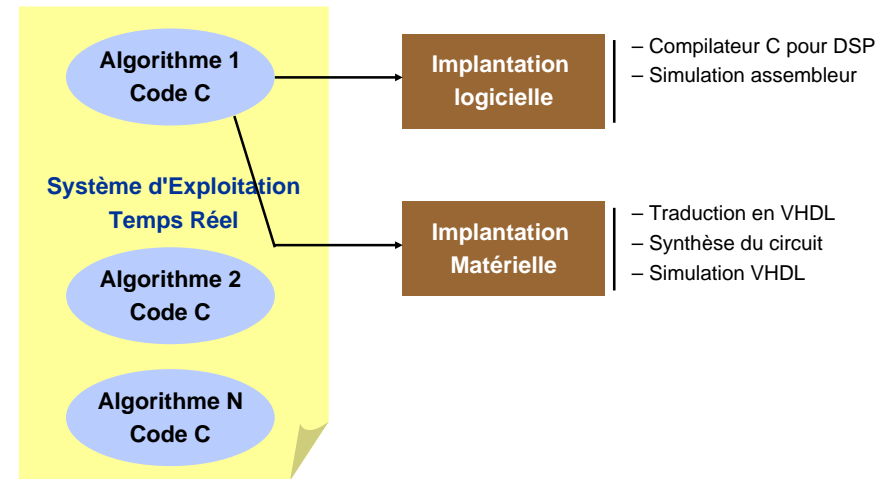
- Développement (actuel) d'applications (signal)



ARCHI'05 - 112



Flot général (suite)

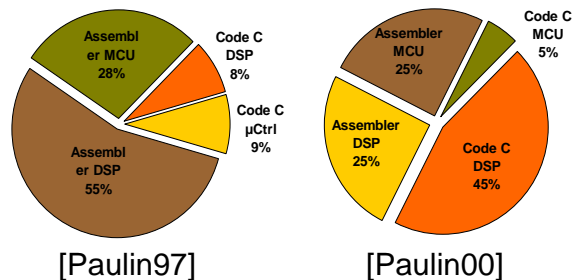


ARCHI'05 - 113



Compilation

- Compilation C vs optimisation assembleur
- Conception de code enfoui dans l'industrie
 - e.g. STMicroelectronics

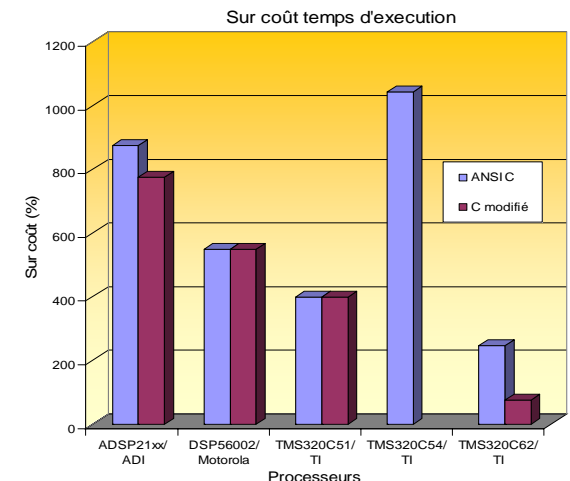


ARCHI'05 - 115



Inefficacité des compilateurs C

- Surcoût associé au compilateur
- Facteur 4 à 10 (virgule fixe)
- Facteur 2 à 4 (flottants)
- Orthogonalité
 - VLIW

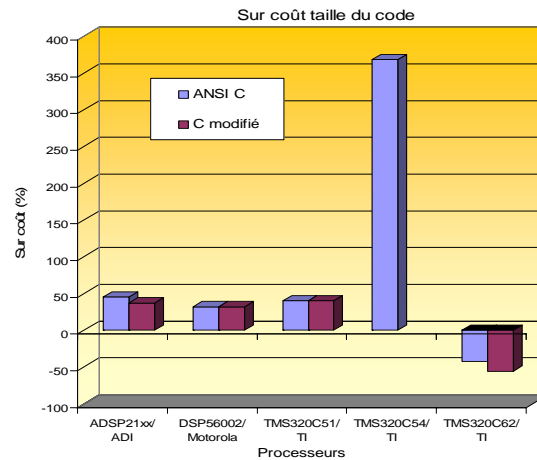


ARCHI'05 - 116



Inefficacité des compilateurs C

- Surcoût associé au compilateur
 - Taille du code



ARCHI'05 - 117



Raisons de l'inefficacité

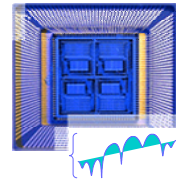
- Absence de support de l'arithmétique virgule fixe
- Inefficacité des techniques classiques de développement des compilateurs
 - développement de l'architecture puis du compilateur
 - compilateurs recyclables
- Inefficacité des techniques classiques de compilation
 - architecture non orthogonale, unités spécialisées, instructions complexes
- Absence de support des spécificités des DSP
 - registres d'état, modes d'adressage (modulo, bit inversé)...

➤ **Extensions du langage C : e.g. Embedded C**

ARCHI'05 - 118



III. Flot de développement



- Flot général de développement
- Compilation pour DSP
- Arithmétique virgule fixe



Formats de codage

Virgule fixe

- Représentation : signe - partie entière - partie fractionnaire

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

- $b_x = m_x + n_x + 1$
- 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 $u = (-1)^{S_E} \sum_{i=1}^{e-1} d_i 2^i$

ARCHI'05 - 120



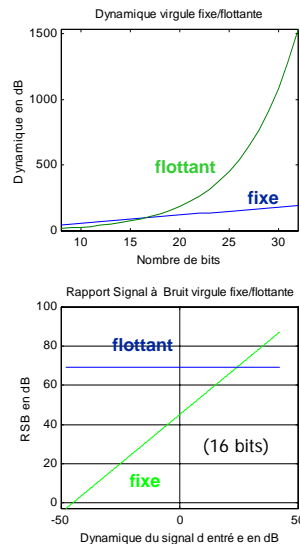
Comparaison fixe - flottant

- Dynamique

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

- Rapport signal à bruit de quantification (SQNR)

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

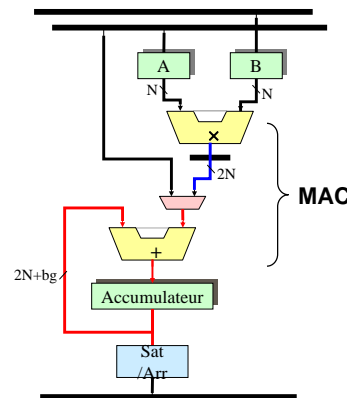


Comparaison fixe - flottant

- Virgule fixe (>90% des ventes des DSP)
 - Précision importante mais problèmes de débordement
 - nécessité de recadrer les données
 - Temps de développement plus long
 - Efficacité énergétique plus importante, plus rapide et moins cher
 - Consommation moins importante
 - Marché : applications grand public
 - C5x > \$5
- Virgule flottante
 - Temps de développement plus rapide, compilateurs plus efficaces
 - Plus grande portabilité, pas de recadrage
 - Pas de débordements : dynamique de 1500dB (32 bits)
 - Plus cher, consomme plus
 - C67x > \$30

Éléments de l'UT

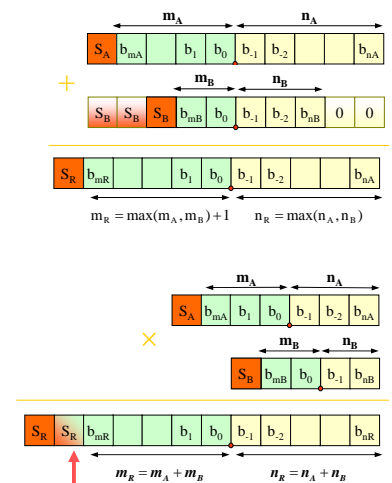
- Multiplieur câblé
 - Additionneur
 - U.A.L.
- (1 Op/cycle)



- Registres à décalage
 - ⇒ recadrage des données
- Gestion des débordements
- Bits de garde
- Registres d'accumulation
 - ⇒ stockage des données en double précision

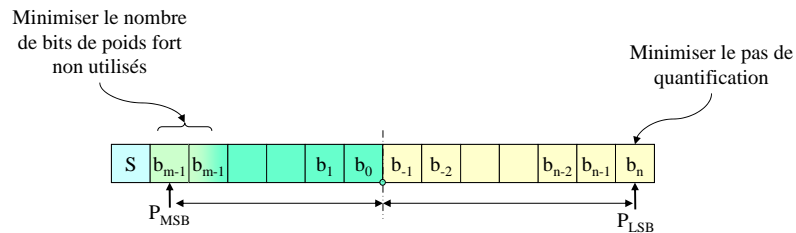
Règles de l'arithmétique virgule fixe

- Addition: $a + b$
 - Format de a et b identique
 - Choix d'un format commun
 - Alignement de la virgule
 - Extension de bits
- Multiplication: $a \times b$
 - Représentation identique
 - Doublement du bit de signe



Codage en virgule fixe : objectifs

- Objectifs pour le codage en virgule fixe
 - Garantir l'absence de débordements
 - Connaissance du domaine de définition des données
 - Maximiser la précision



ARCHI'05 - 125

Codage en virgule fixe : étapes

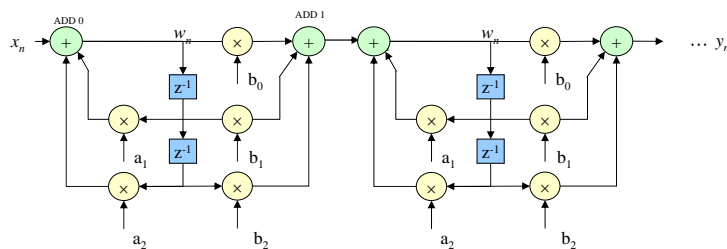
1. Détermination de la dynamique des données
 - Méthodes statistiques ou analytiques
2. Détermination de la position de la virgule
3. Détermination de la largeur des données
 - Opérateurs SIMD, précision multiple
4. Evaluation de la précision des calculs
 - Méthodes par simulation ou analytiques



ARCHI'05 - 126

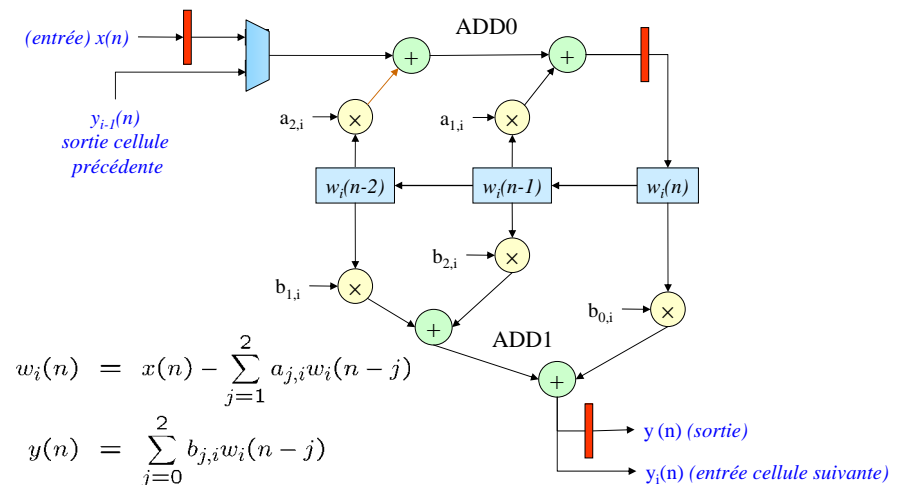
Filtre récursifs (IIR)

- Cascade de cellules d'ordre 2



ARCHI'05 - 127

Filtre IIR d'ordre 2 (cellule i)



ARCHI'05 - 128

Dynamique des données

- Normes utilisées en TS
- Norme L1

$$z_{\max 1} = \max_n (|x(n)|) \cdot \sum_{m=-\infty}^{\infty} |h(m)|$$

➤ Méthode garantissant l'absence de débordements

- Norme Chebychev

$$z_{\max 2} = \max_{n,\omega} (|x(n)|) \max (|H(\omega)|)$$

➤ Méthode garantissant l'absence de débordements pour un signal d'entrée à bande étroite e.g. $x(n) = \cos(\omega n)$

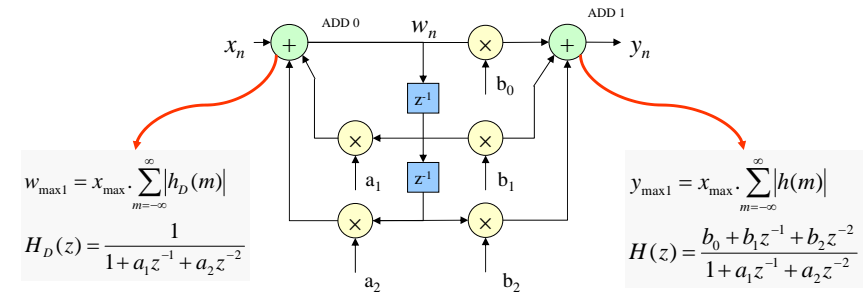
- Arithmétique d'intervalle (non récursifs)

ARCHI'05 - 129



Dynamique des données (IIR)

- Sources de débordements : additionneurs
 - IIR : 2 sources ADD0 et ADD1



ARCHI'05 - 130



Dynamique des données (IIR)

- Exemple

$x \in [-1, 1]$

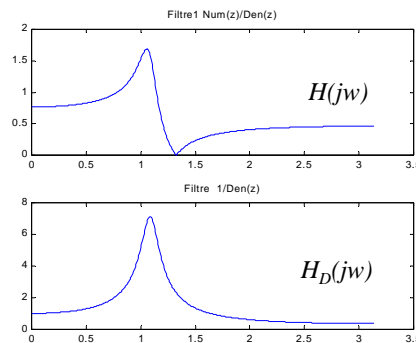
$$H(z) = \frac{0.5 - 0.2428z^{-1} + 0.5}{1 - 0.85z^{-1} + 0.8417z^{-2}}$$

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

$$x_{\max} \sum_{m=-\infty}^{\infty} |h_D(m)| = 9 \rightarrow m_w = 4 \rightarrow k_1 = 4$$

Codage des coefficients

a1 = -27951
 a2 = 27581
 b2 = 16384
 b0 = 16384
 b1 = -7956



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

ARCHI'05 - 131



Position de la virgule (IIR)

$$m_x = \lceil \log_2(\max_n (|x(n)|)) \rceil = 0$$

$$m_{w_i} = \lceil \log_2(\max_n (|w_i(n)|)) \rceil = 4$$

$$m_{y_i} = \lceil \log_2(\max_n (|y_i(n)|)) \rceil = 2$$

$$m_{a_{ij}} = \lceil \log_2(\max_j (|a_{ij}|)) \rceil = 0$$

$$m_{b_{ij}} = \lceil \log_2(\max_j (|b_{ij}|)) \rceil = 0$$

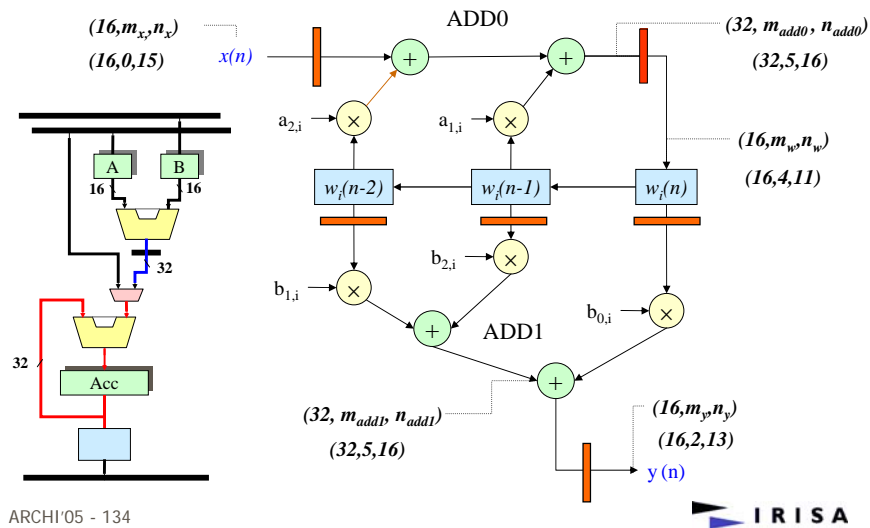
$$m_{ADD0} = \text{MAX}(m_x, m_{a_i} + m_{w_i} + 1, m_{w_i}) = 5$$

$$m_{ADD1} = \text{MAX}(m_{b_i} + m_{w_i} + 1, m_{y_i}) = 5$$

ARCHI'05 - 133

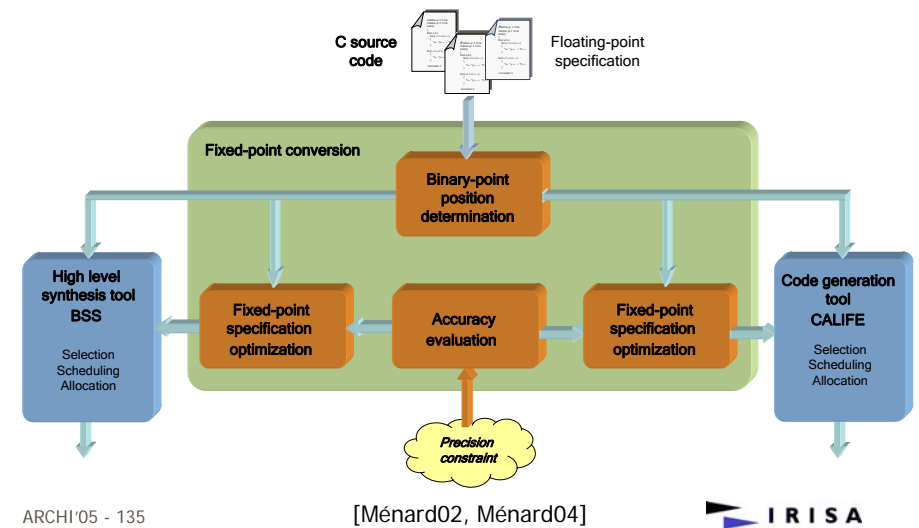


Codage des données (IIR)



ARCHI'05 - 134

Conversion flottant-fixe (IRISA)



ARCHI'05 - 135

[Ménard02, Ménard04]

Conclusions

- Processeurs spécialisés garantissent :
 - efficacité énergétique, rapport performances/coût
 - autres exemples probants
 - Processeurs multimédia
 - Network Processor NPU
- Les performances des processeurs DSPs ont augmenté de 150x sur 15 ans (40% par an)
- Les nouvelles architectures sont nombreuses
 - Mais les DSPs conventionnels dominent toujours le volume

ARCHI'05 - 136

Conclusions

- Les processeurs généraux ont maintenant des performances qui concurrencent les DSPs
 - Mais le prix ... et la consommation ne sont pas applicables aux applications embarquées grand public
- La facilité de compilation est un facteur important
 - *time-to-market...*
- Choisir un DSP requiert une analyse fine
 - qui dépend de l'application...

ARCHI'05 - 137

Perspectives

- Quelle sera l'architecture pour le traitement du signal et de l'image de demain (ou d'après demain) ?

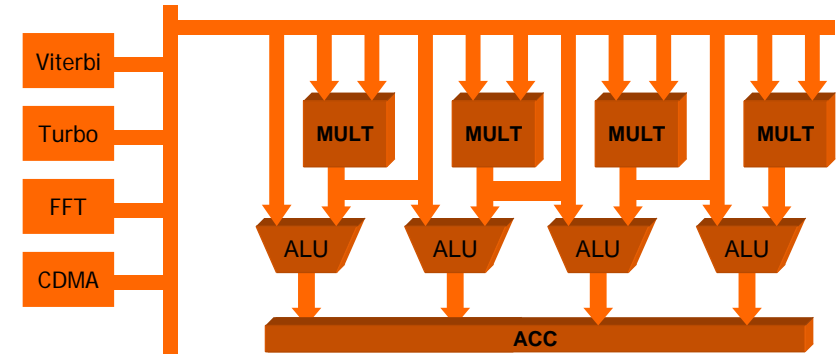


ARCHI'05 - 138



Piste 1 : DSP conventionnel

- DSP conventionnel multi-MAC
 - Unités spécialisés
- Programmation ?**



ARCHI'05 - 139

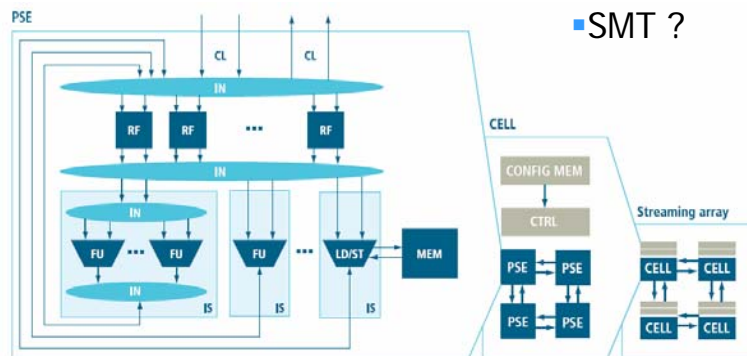


Piste 2 : ULIW

- Ultra Large Instruction Word
 - e.g. Silicon Hive AVISPA+

Compilation ?

- SMT ?

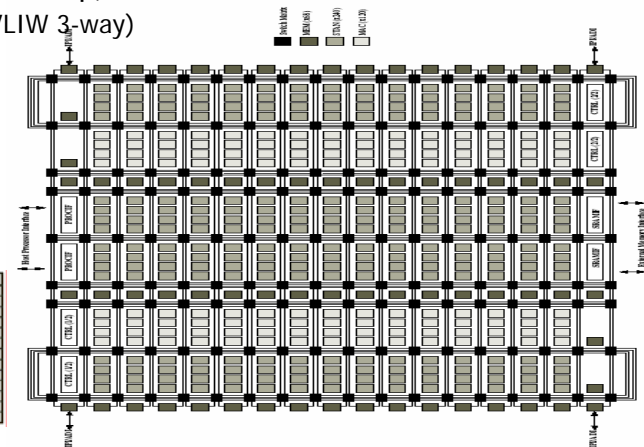


ARCHI'05 - 140



Piste 3 : multiprocesseur

- PicoArray (picoChip)
 - ~320 PE (VLIW 3-way)
 - 160 MHz
 - 190 GIPS



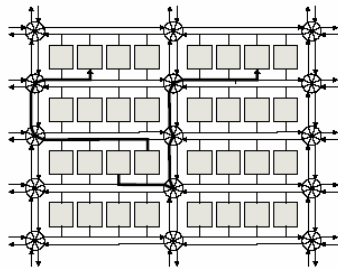
Device Characteristics	Parameter	Unit
Die size	308	mm ² per device
Die processors	14	per device
Processor clock	160	MHz
Peak processing capacity	190 GIPS	GIPS
Die on chip data bandwidth	3.3	Tb/s
Peak 3D MAC/s	38.4	G-MAC/s
Internal bus speed	5.15	Gbit/sec
Die chip RAM	1003	Kibytes
External RAM (DDR2)	8 (128)	Mibytes
External SDRAM access speed	4000/160	MHz
CDMA channels	4 * 2.4	Gbit/sec
RF 110.040 ultra-device channels	4 * 2.55	Gbit/sec
RF 110.040 TV channels	8 * 2.4	Gbit/sec

ARCHI'05 - 141



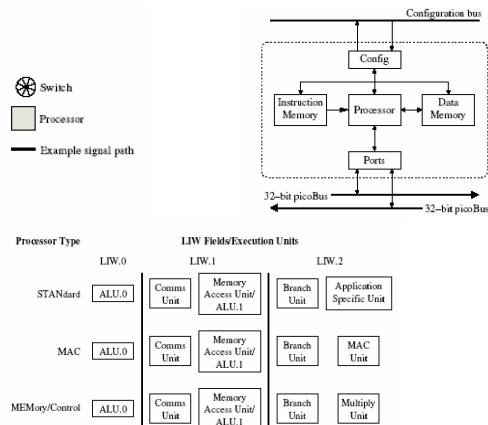
PicoArray

- Static Interconnect



- VHDL: structure
- C/Assembly: PE

- PE: 16-bit 3-way VLIW

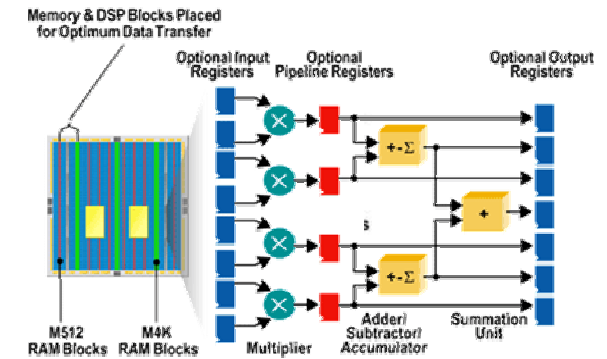


Processor Type	LIW Fields/Execution Units		
	LIW.0	LIW.1	LIW.2
STANDARD	ALU.0	Comms Unit Memory Access Unit/ ALU.1	Branch Unit Application Specific Unit
MAC	ALU.0	Comms Unit Memory Access Unit/ ALU.1	Branch Unit MAC Unit
MEMory/Control	ALU.0	Comms Unit Memory Access Unit/ ALU.1	Branch Unit Multiply Unit

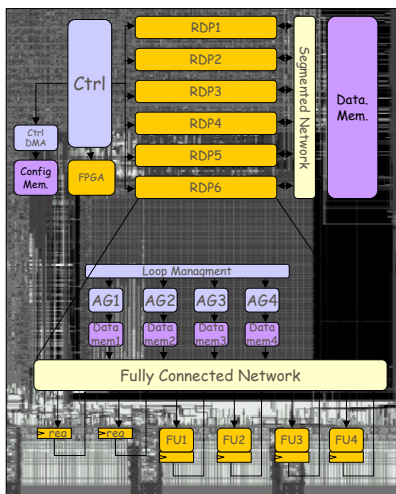


Piste 4 : FPGA

- e.g. Altera Stratix
 - Multiplieurs 9x9 – 18x18, 2 GMAC/s par bloc, 250 MHz



Piste 5 : processeur reconfigurable

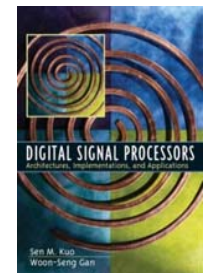
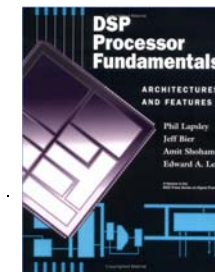


- Architecture DART
 - 3G/UMTS Mobile Terminal
 - 802.11a (Channel Est.)
- IRISA/R2D2
- STMicroelectronics
- CEA LIST/LETI
- Performances
 - 5-10 GOPS/cluster@0.13m
 - 200MHz, 11mm²
 - 300 mW @ 5 GOPS
 - 16 MOPS/mW @ 5 GOPS
 - Circuit en juin 2005



Bibliographie

- DSP Processor Fundamentals : Architectures and Features (IEEE Press Series on Signal Processing) by Phil Lapsley, Jeff Bier, Amit Shoham, Edward A. Lee, Wiley-IEEE Press, 1997
- Digital Signal Processors : Architectures, Implementations, and Applications by Sen M. Kuo, Woon-Seng S. Gan, Prentice Hall, 2004



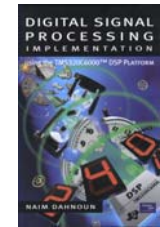
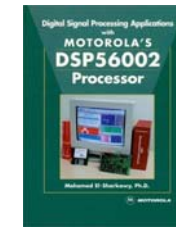
Bibliographie

- Programmable Digital Signal Processors
by Yu Hen Hu (Editor), Marcel Dekker, 2001
- VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis
by Vijay Madisetti,
Butterworth-Heinemann, 1995
- Méthodes et architectures pour le TSI en temps réel, de Didier Demigny, Hermès, 2002



Bibliographie

- Digital Signal Processing Applications With Motorola's DSP56002 Processor
by Mohammed El-Sharkawy,
Prentice Hall, 1996
- Digital Signal Processing Implementation Using the TMS320C6000 DSP Platform (With CD-ROM) by Naim Dahnoun, Prentice Hall, 2000
- DSP Applications Using C and the TMS320C6x DSK by Rulph Chassaing, Wiley-Interscience, 2002



Références

- [David00] R. David, Etat de l'art des cœurs de DSP, Rapport DEA, ENSSAT, 2000.
- [Sentieys00] O. Sentieys, Etat de l'art des DSP, École thématique du CNRS - Conception de systèmes enfouis, Seix (Ariège), 20-23 novembre 2000.
- [Sentieys01] O. Sentieys, DSP et processeur superscalaire : la convergence ?, Symposium en Architectures Nouvelles de Machines, Paris, 2001.
- [Ménard04] D. Ménard, Processeur de traitement du signal, ENSSAT/EII, 2004.
- [ICE97] B. McClean, "Status 1997: A Report on the Integrated Circuit Industry", Integrated Circuit Engineering Corporation (ICE), Scottsdale, 1997
- [Bhaskaran95] V. Bhaskaran & K. Konstantinides, "Image and Video Compression Standards - Algorithms and Architectures", Kluwer Academic Publishers, Boston, 1995.
- [Bier97] J. Bier, P. Lapsley & G. Blalock, "Choosing a DSP Processor", Berkeley Design Technology (BDT), 1997.
- [DeMan97] H. De Man and al., "Language Based Design of Digital Systems", AI Course, KU Leuven/IMEC, avril 1997.
- [Lapsley96] P. Lapsley and G. Blalock, "How to Estimate DSP Processor Performance", IEEE Spectrum, July 1996.
- [Pirsch97] P. Pirsch: "Video and Image Processing Architectures - Dedicated and Programmable Solutions", NFWO Symposium on Video Processing Architectures, January 1997.
- [Ackland98] B. Ackland and C. Nicol, "High Performance DSPs - What's Hot and What's Not?", IEEE Int. Symposium on Low Power Electronic Design ISLPED, 1998.
- [Ropers99] A. Ropers and al., "DSPstone : TI C54x" Report IISPS Aachen University of Technology, 1999.

Webographie

- <http://www.bdti.com>
- <http://dsprelated.com/>
- <http://www.ti-training.com/courses>
- <http://www.eg3.com/dsp/index.htm>