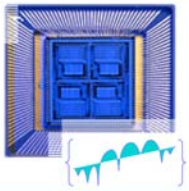
Processeurs de Traitement Numérique du Signal (DSP)



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

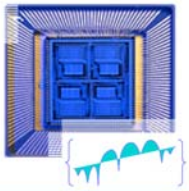
ENSSAT LANNION

UNIVERSITÉ DE RENNES 1

Merci à Daniel Ménard pour son support

IRISA

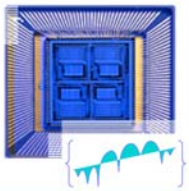
Processeurs de Traitement Numérique du Signal (DSP)



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

IRISA

I. Introduction



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

IRISA

Exemples d'applications

- Performances
- Faible coût
- Faible énergie

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

IRISA

Tâches élémentaires

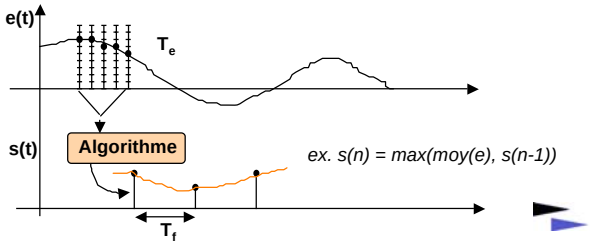
- Compression de signal (parole, audio, vidéo)
- Filtrage
- Modulation et démodulation
- Détection et correction d'erreurs
- Contrôle
- Traitements audio (e.g. réduction bruit, égalisation, annulation d'écho, conversion de fréquence)
- Reconnaissance vocale
- Synthèse de signaux

IRISA

Algorithmes de TdSI[1]

[1]Traitement du signal et de l'image

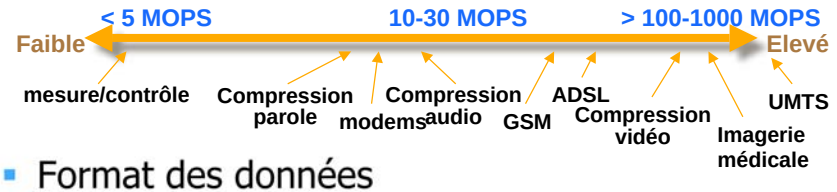
- Signaux numériques
  - Temps et amplitude discrets
  - Flot de données
  - Scalaires, vectorielles, matricielles, ...
- Traitement temps réel
  - Temps d'exécution T_{ex} guidé par flots de données
  - Période d'échantillonnage T_e période des sorties $T_f > T_{ex}$



IRISA

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 s ang	16 - 18 bits

IRISA

Fonctions typiques de TdSI

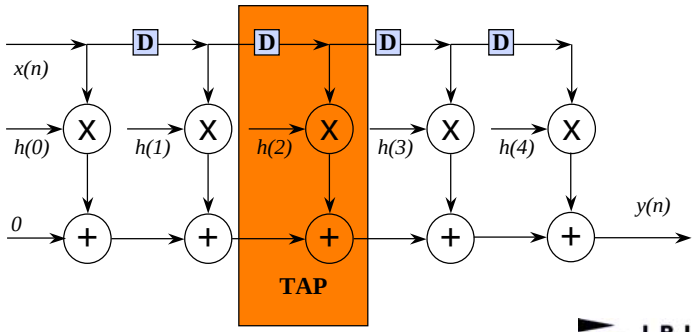
- Convolution, filtrage (RIF, RII), corrélation, DCT
  - y = y + x.h : MAC (multiplication-accumulation)
- Adaptation (LMS)
  - $y_n = y_{n-1} + x.h$: MAD (multiplication-addition)
- FFT, multiplication complexe
  - xr=xr.wr-xi.wi; xi=xr.wi+xi.wr
- Viterbi
  - a1 = x1 + x2; a2 = y1 + y2;
  - y = (a1>a2) ? a1 : a2 : ACS (addition-comparaison-sélection)
- Estimation de mouvement
  - sad += $|x_{i,j} - y_{i+u,j+v}|$: SAD (sum-of-absolute-difference)

IRISA

Exemple *Fil Rouge*

- Filtre Numérique RIF sur N points

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

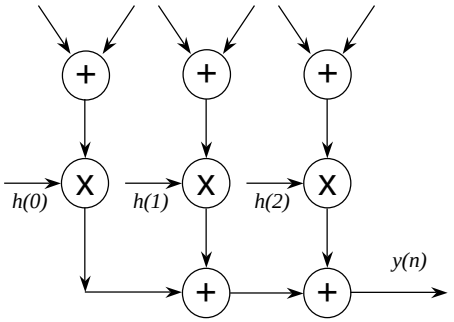


IRISA

Fonctions typiques de TdSI

- Filtre Numérique RIF symétrique sur N points

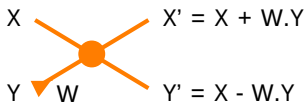
$$y(n) = \sum_{i=0}^{N/2} h(i)[x(n-i) + x(n-N+1+i)]$$



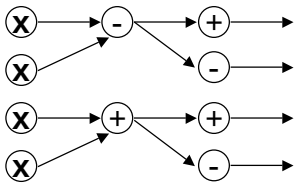
IRISA

Fonctions typiques de TdSI

- FFT Fast Fourier Transform
  - Butterfly (DIF)



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



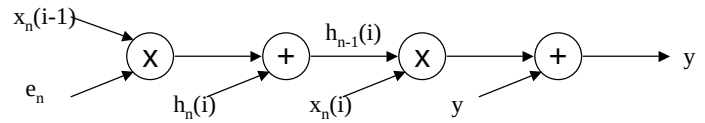
IRISA

Fonctions typiques de TdSI

- Filtrage adaptatif LMS

$$y_n = \sum_{i=0}^{N-1} \underline{h_n}(i).\underline{x_n}(i)$$
$$e_n = d_n - y_n$$
$$\underline{h_{n+1}}(i) = \underline{h_n}(i) + \mu.e_n.\underline{x_n}(i) \quad \forall i \in \{0 \ldots N-1\}$$

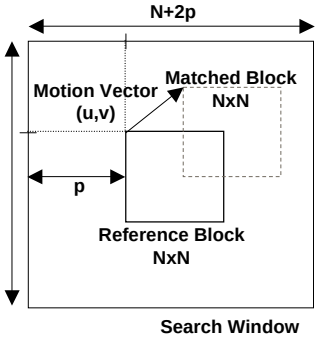
- DLMS



IRISA

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;
  }
}
```


IRISA

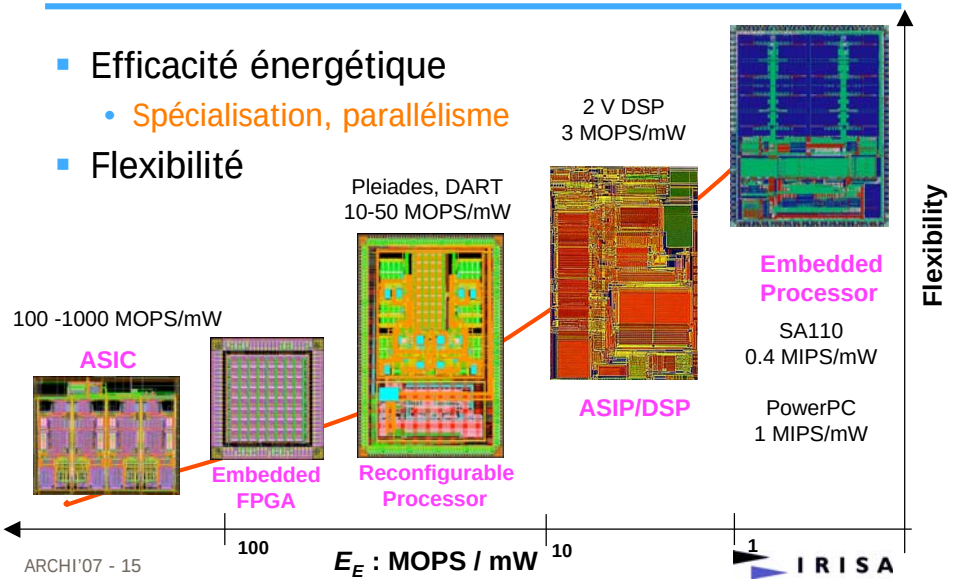
Caractéristiques algorithmiques

- 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)

IRISA

Solutions architecturales

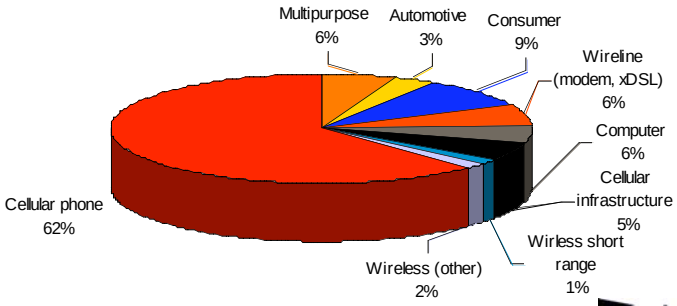
- Efficacité énergétique
  - Spécialisation, parallélisme
- Flexibilité



IRISA

Marché des DSP (1)

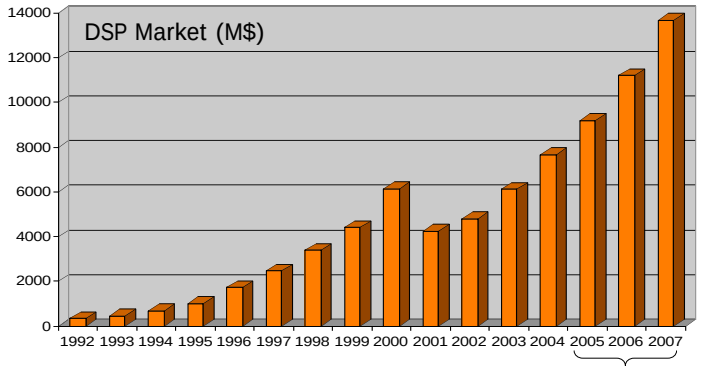
- Sans fils domine : 75% des revenus en 2004
- Systèmes grand public : DVD, TV et radio numérique, ...
- Contrôle de moteurs, ...



IRISA

Marché des DSP (2)

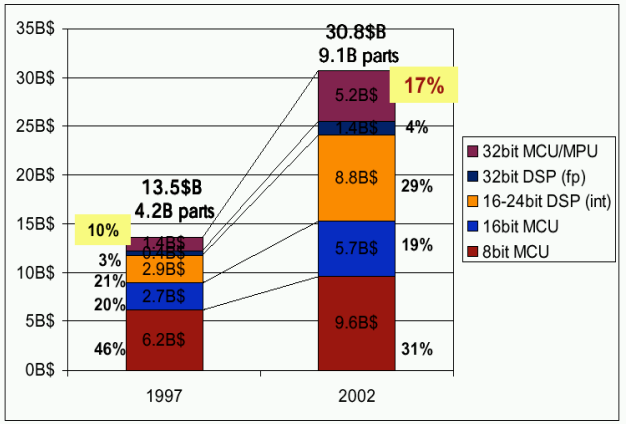
- En haut de la liste des plus fortes croissances du marché de l'industrie des semi-conducteurs



IRISA

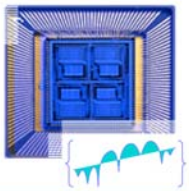
Marché des DSP (3)

- Place dans le marché des processeurs embarqués



IRISA

II. Architecture MAC/Harvard

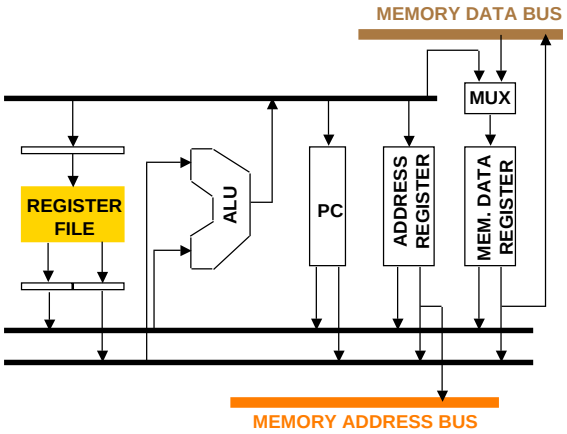


1. Genèse des DSP : MAC/Harvard
2. Modélisation des générations 1 et 2
   Panorama des DSP
3. Performances

IRISA

Architecture Von Neumann

- E.g. processeurs RISC



Code example:
multiply & accumulate
*r2 = r2 + #imm * r1*

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

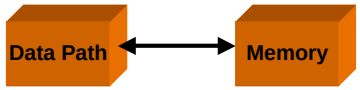
⇒ 3 instructions, plus de 3 cycles d'horloge

IRISA

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

IRISA

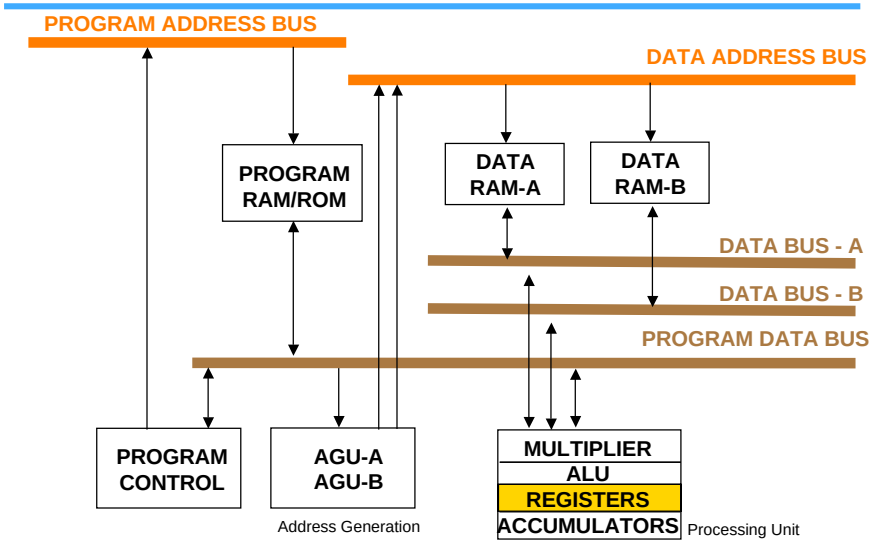
Généralités sur les DSPs

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 d'adressage complexes

Le tout dans un environnement temps réel

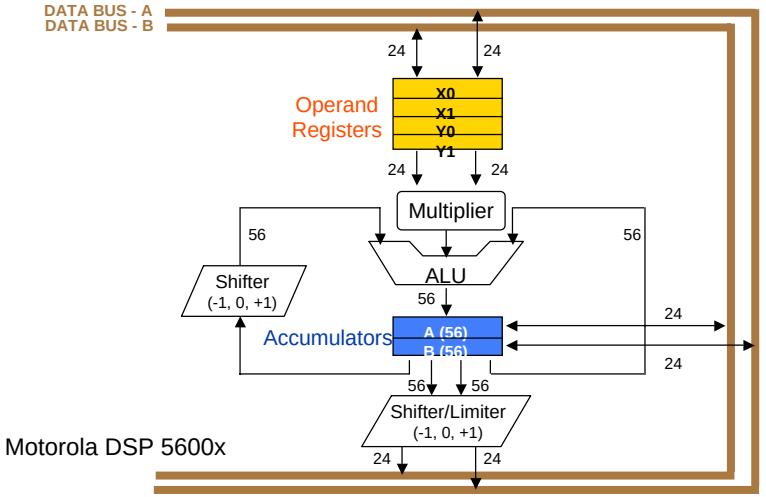
IRISA

Architecture Harvard (1)



IRISA

Unité de traitement

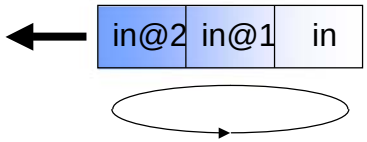


Motorola DSP 5600x

IRISA

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) ...



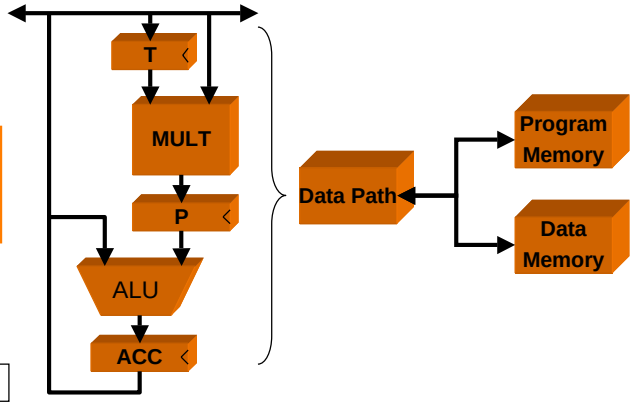
IRISA

FIR sur DSP conventionnel

- Instructions complexes s'exécutant en 1 cycle

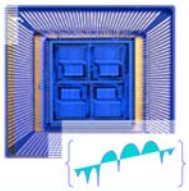
```
T=DM(*r1+);
P=T*PM(*r0+)
|| T=DM(*r1+);
loop:
  acc=acc+P
  || P=T*PM(*r0+)
  || T=DM(*r1+)
  jnz loop;
acc=acc+P
|| P=T*PM(*r0+);
acc=acc+P
```

Exécution en N+ε cycles



IRISA

II. Architecture MAC/Harvard



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

IRISA

Processeur : modélisation

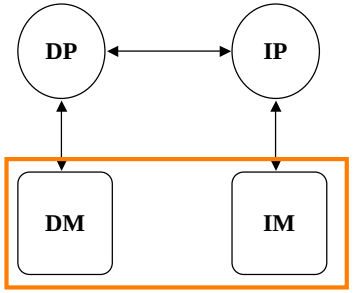
- **Unité de contrôle** (IP : *Instruction Processor)*
  - Unité fonctionnelle (UF) qui interprète les instructions et les passe à l'unité de traitement (DP)
- **Unité de traitement** (DP : *Data Processor)*
  - UF qui modifie ou transforme les données
- **Unité de mémorisation**
  - IM : *Instruction 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

[Flynn72] [Skillicorn88]

IRISA

Architecture Harvard de base

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

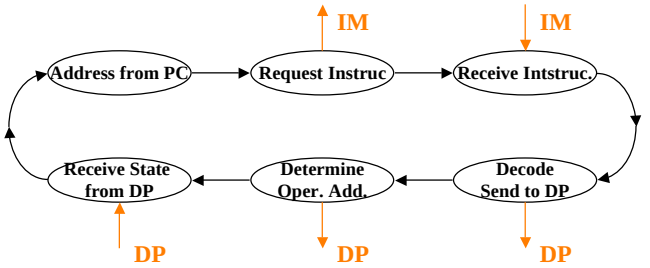


e.g. TMS320C10

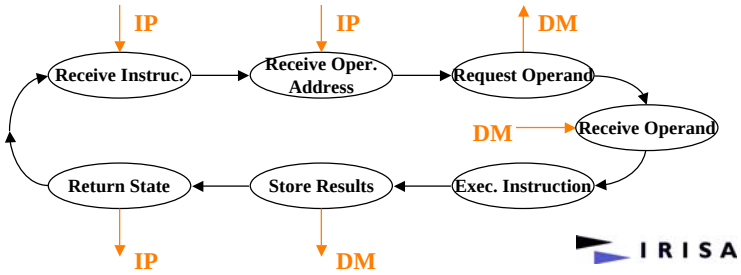
Classification de E. Lee [Lee89]

Processeur : modélisation

- IP

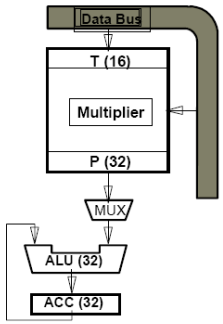


- DP



Architecture Harvard de base

- TMS320C10 (1982)
  - 2N+4 cycles
  - 2N+4 instructions!



Compute Y = AX1 + BX2 + CX3 + DX4

```
ZAC          ACC=0
LT   X1      T=X1
MPY  A       P=AX1
LTA  X2      ACC=AX1;T=X2
MPY  B       P=BX2
LTA  X3      ACC=AX1+BX2;T=X3
MPY  C       P=CX3
LTA  X4      ACC=AX1+BX2+CX3;T=X4
MPY  D       P=DX4
APAC         ACC=AX1+BX2+Cx3+DX4
SACH Y1      STORE 32-BIT RESULT
SACH Y2      AT LOCATIONS Y1, Y2
```

[Verbauwhede00]

Optimisations des performances

Comment améliorer l'architecture Harvard ?

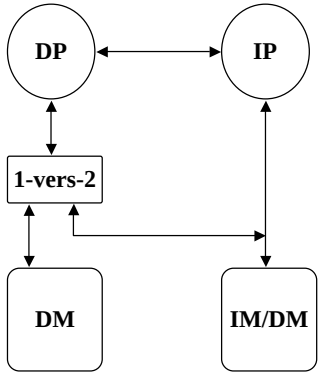
- Transformations du diagramme d'états
  - réarrangement des états
  - exécution en parallèle d'états (e.g. Store Results et Return State)
- Pipeline de l'exécution des états
  - mise en parallèle d'états appartenant à des instructions successives
- Augmentation du nombre de FUs et/ou de DMs
  - plusieurs DPs peuvent exécuter des instructions en parallèle sur des données indépendantes
  - FUs spécialisées pour la génération d'adresses

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

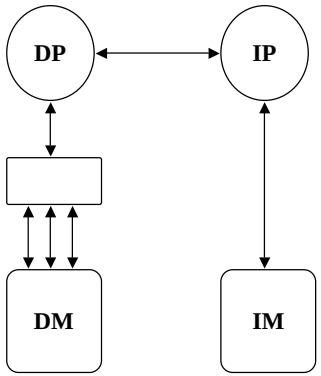
- Autorisation de mémorisation de données dans l'IM
- En un cycle : *fetch* deux opérandes de la mémoire, exécute MAC, écriture du résultat en I/O ou mémoire



e.g. AT&T DSP32 et DSP32C

Modification 2

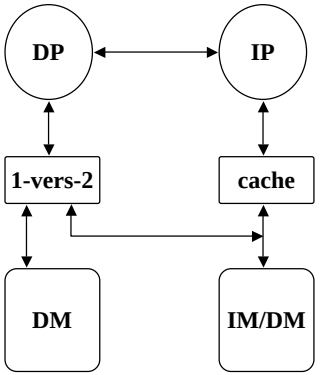
- DM est une mémoire multi-ports, plusieurs accès aux données par cycle
- Utilisable pour des mémoires internes



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

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

FIR sur DSP conventionnel

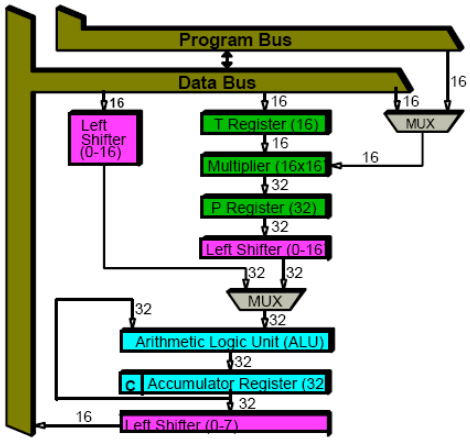
- TMS320C2x (1986)

```
ZAC || LTD;
LTD || MPY;

RPTK N-1
MACD

APAC || MPY;
APAC;
```

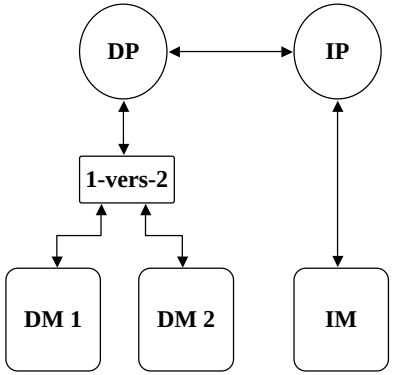
Exécution en N cycles



IRISA

Deuxième génération (Modification 4)

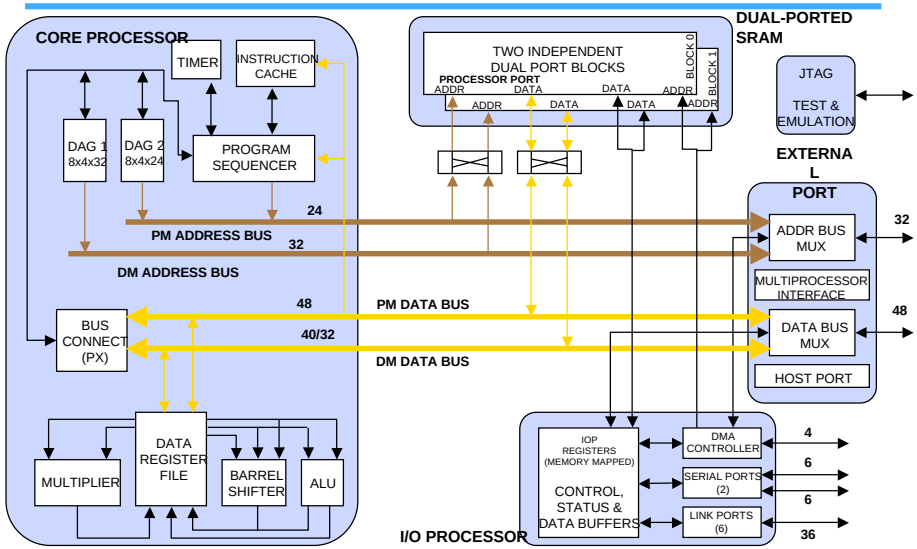
- Deux mémoires données DM séparées
- En un cycle : *fetch* deux opérandes et une instruction si le cycle mémoire est le même que le cycle d'instruction de base



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

IRISA

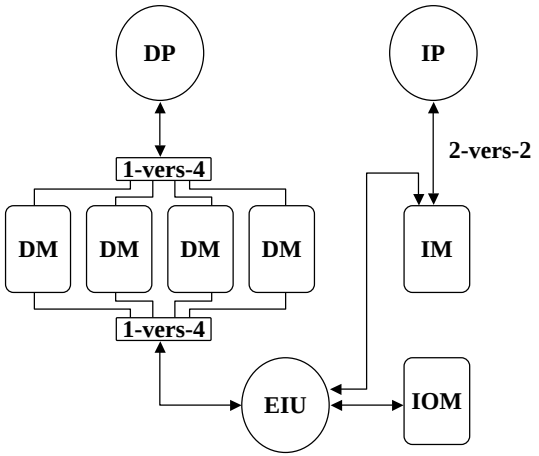
Architecture Analog Devices SHARC



IRISA

Modification 5

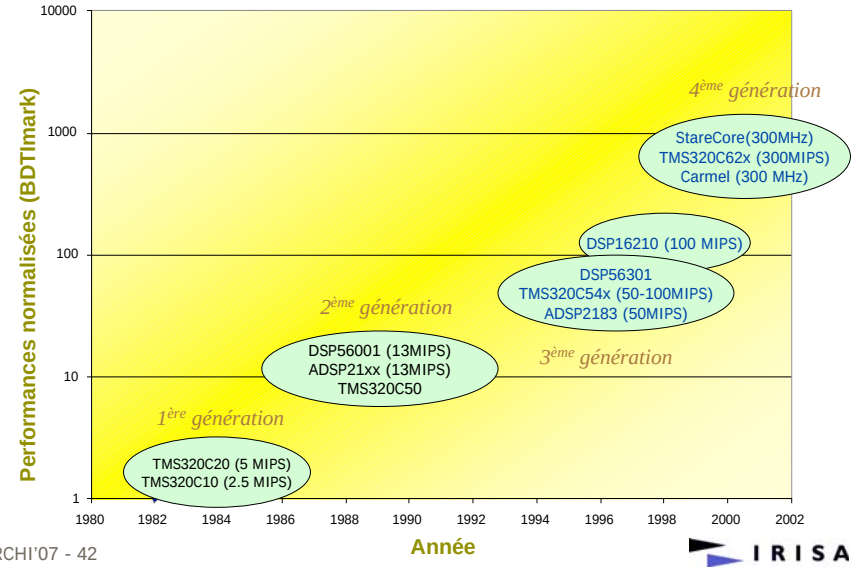
- Bancs mémoire multiples
- Instruction multi-opérandes en parallèle avec accès I/O
- Problème de partition des données sur un grand nombre de bancs (FFT)



Hitachi DSPi (6 bancs mémoire)

IRISA

Générations



IRISA

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

[cf. Annexes]

IRISA

Résumé (1)

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

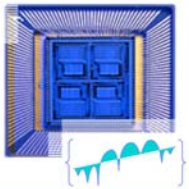
IRISA

Résumé (2)

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

IRISA

II. Architecture MAC/Harvard



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

IRISA

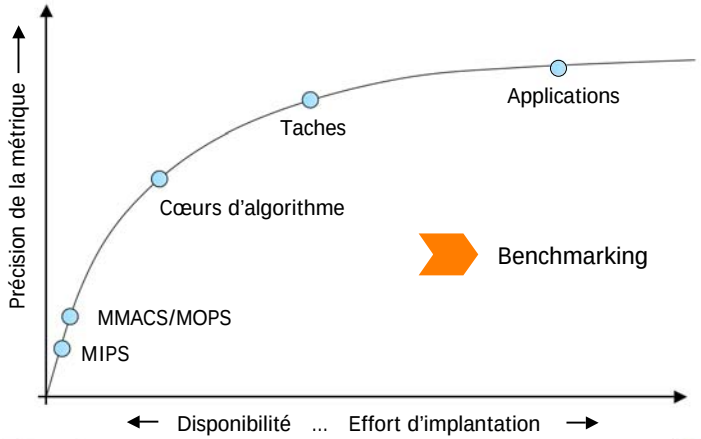
Puissance de calcul

- 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

IRISA

Métriques de performance

- Comparaison précision - effort d'implantation



IRISA

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

IRISA

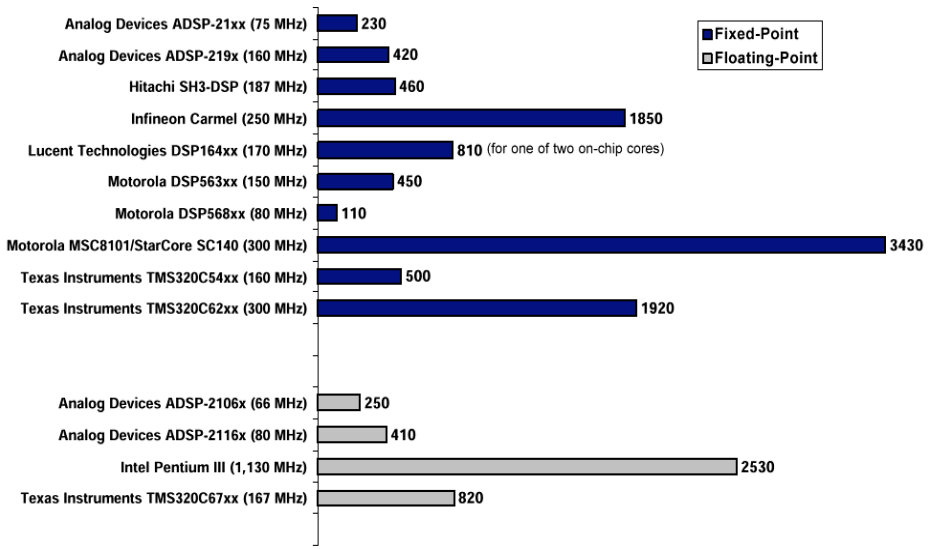
BDTImark2000

http://www.bdti.com

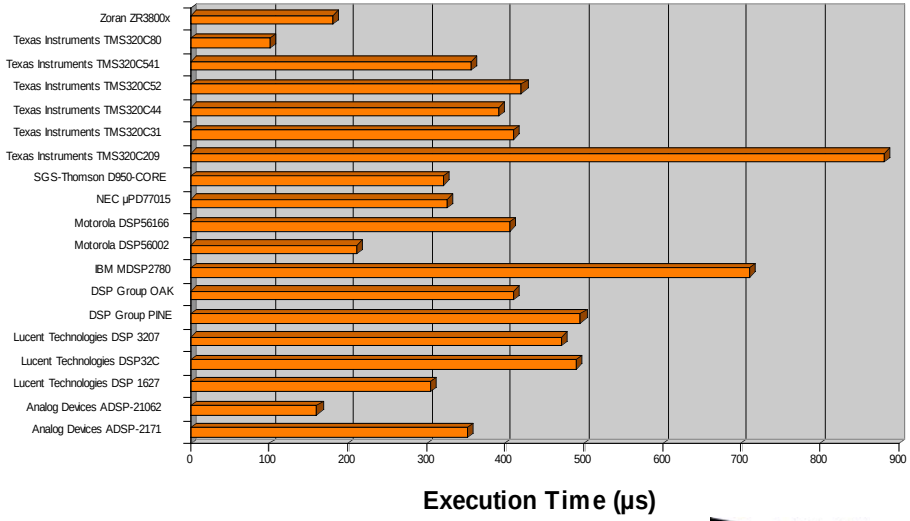
- Basé sur des cœurs d'algorithme de traitement du signal
  - Filtre FIR réel, traitement par blocs
  - Filtre FIR réel, traitement par échantillons
  - Filtre FIR complexe, traitement par blocs
  - Filtre adaptif LMS
  - Filtre IIR, 2 cellules d'ordre 2 cascadées
  - Produit de vecteurs
  - Addition de vecteurs
  - Recherche de la valeur maximale d'un vecteur
  - Décodeur de Viterbi
  - FFT 256 points
  - Manipulation de bits
- Code optimisé manuellement

IRISA

BDTIMARK



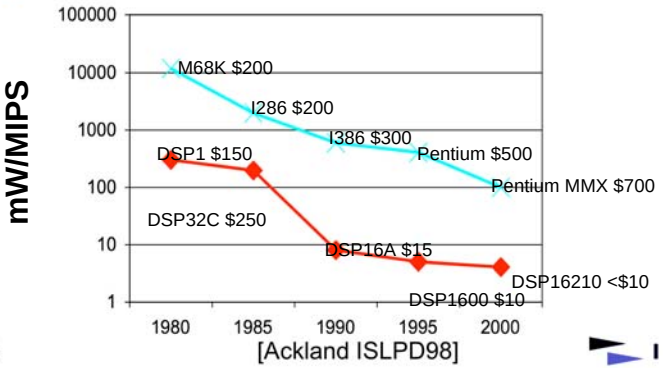
Temps d'exécution (FFT)



IRISA

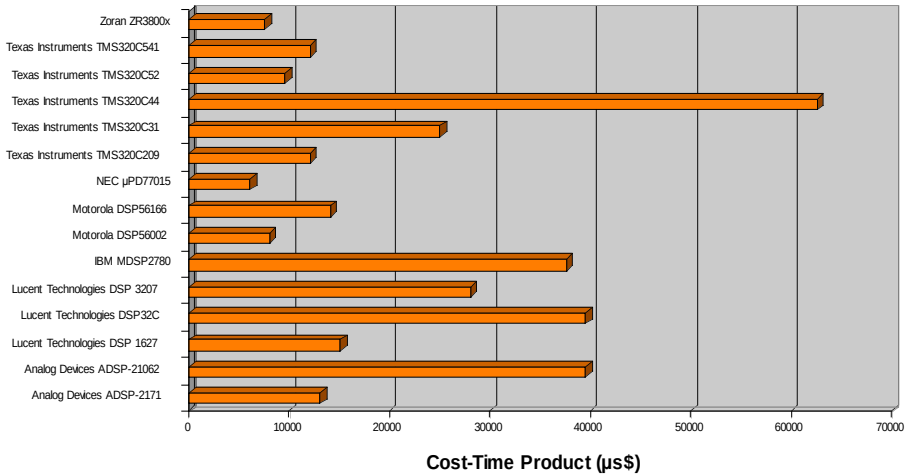
Coût

- 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



IRISA

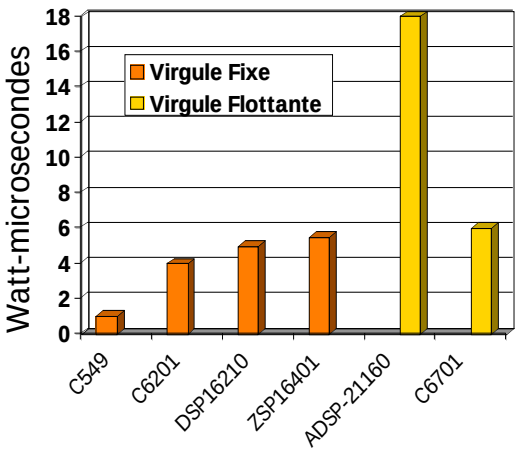
Coût - performance (FFT)



IRISA

Énergie

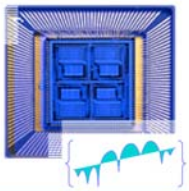
- Filtrage numérique RIF



- 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

IRISA

III. Évolution des DSP



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

IRISA

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

IRISA

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

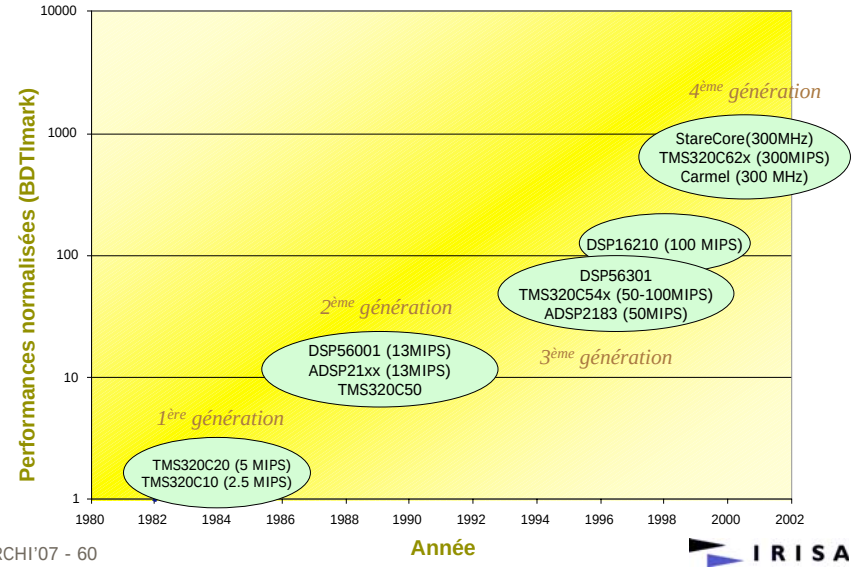
IRISA

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

IRISA

Générations



IRISA

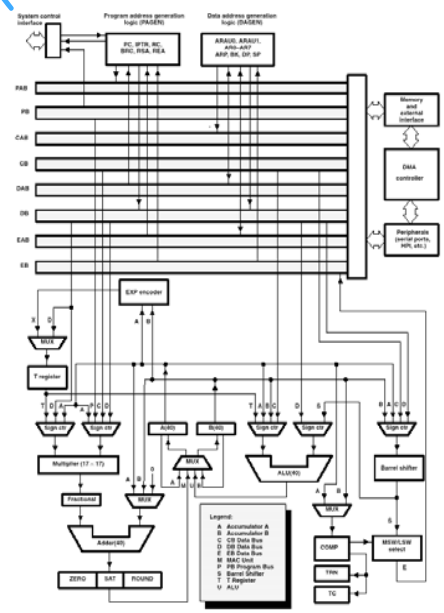
DSP conventionnels améliorés

- 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

IRISA

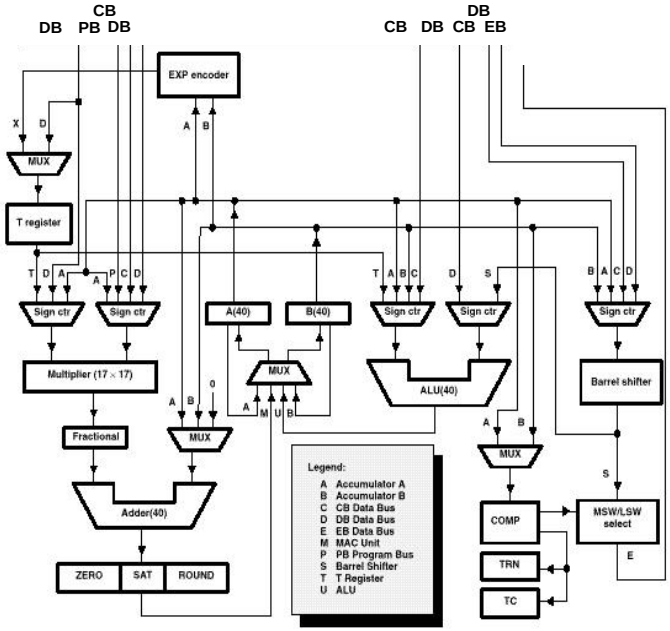
Architecture C54x

- C54x
  - 40-100-160 MIPS
  - 1000-2000-3000 MIPS/W
  - 60% of cellular handsets
  - $5 for C5402 100MIPS - $75
  - Dual cores (C5420)



C54x

- 17x17b mult.
- 40b ALU
- 40b adder
- ACS unit
- Viterbi,LMS
- FIRS, FFT

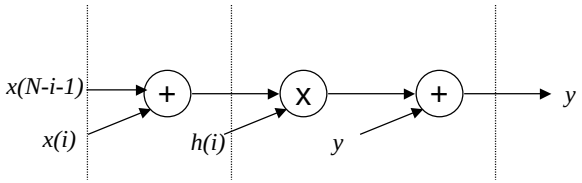


IRISA

C54x

- FIR symétrique

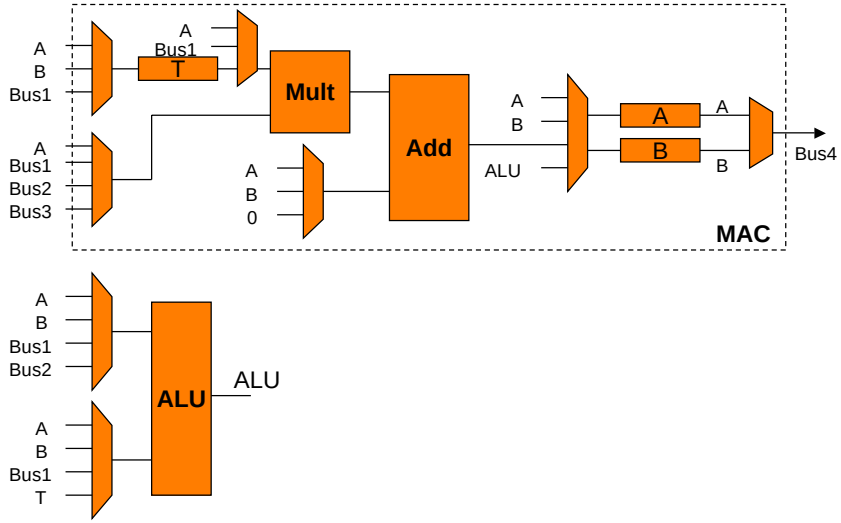
$$y(n) = \sum_{i=0}^{\frac{N}{2}-1} a_i \left[x(n-i) + x(n-N+i+1)\right]$$



DFG du coeur de la boucle

IRISA

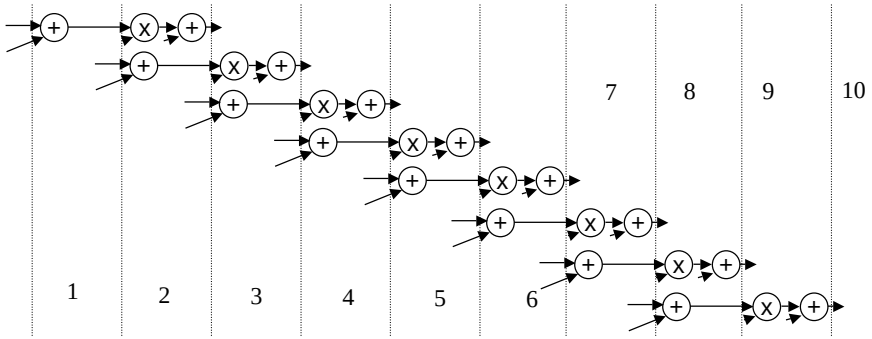
C54x



IRISA

C54x

- e.g. N=20



IRISA

C54x

- Ressources du C54x

T												
A												
B												
Mul/Add												
ALU	+	+	+	+	+	+	+	+	+	+		
Bus 1	x(0)											
Bus 2	x(15)											
Bus 3												
Bus 4												

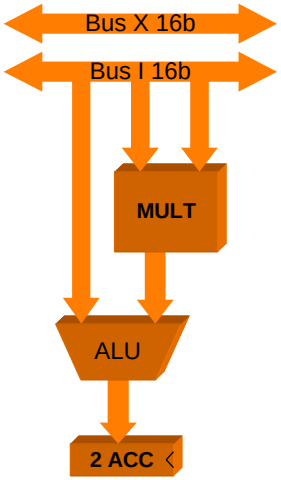
IRISA

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



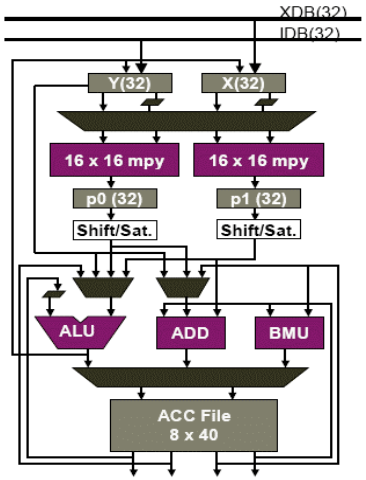
IRISA

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++
```

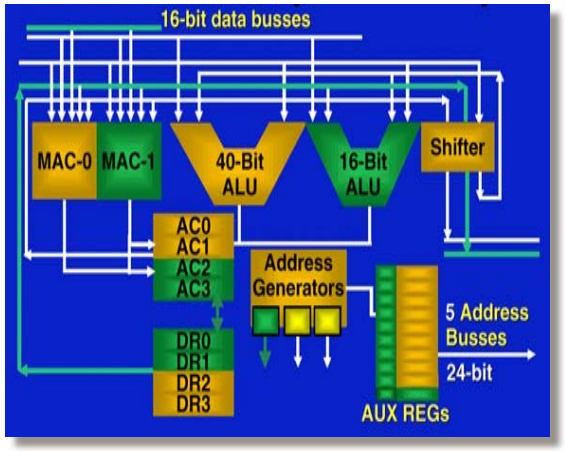
Exécution en N/2 cycles



IRISA

Architecture C55x

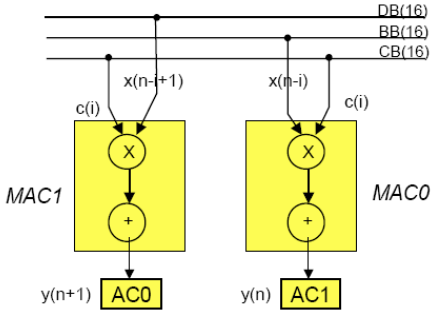
- C55x
  - Jusqu'à 20 MIPS/mW
  - C54x
    - Consommation -85%
    - Performances 5x
- C5510
  - 160 MHz, 320 MIPS
  - 80mW
  - 4 MIPS/mW
  - 160 KW SRAM
- C5502
  - 400 MIPS
  - 160 mW
  - 2.5 MIPS/mW



IRISA

C55x

- FIR sur C55x dual-MAC

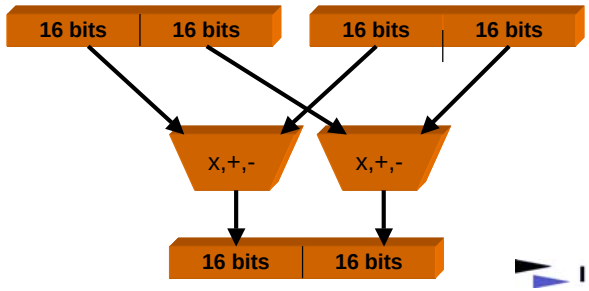



```
rptblocal sample_loop-1
    mov   *AR0+,*AR1          ; Put new sample to signal buffer x[n]
    mov   *AR0+,*AR3          ; Put next new sample to location x[n+1]
    mpy   *AR1+,*CDP+,AC0     ; The first operation
::  mpy   *AR3+,*CDP+,AC1
||  rpt   CSR
    mac   *AR1+,*CDP+,AC0     ; The rest MAC iterations
::  mac   *AR3+,*CDP+,AC1
    macr  *AR1,*CDP+,AC0
::  macr  *AR3,*CDP+,AC1      ; The last MAC operation
    mov   pair(hi(AC0)),dbl(*AR2+); Store two output data
sample_loop
```


IRISA

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



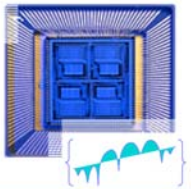
IRISA

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

IRISA

III. Évolution des DSP



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

IRISA

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

IRISA

Architecture hybride DSP/MCU

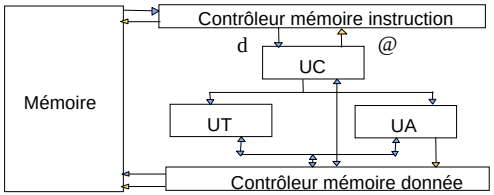
- Méthodes de couplage

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
Coprocesseur	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

IRISA

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



IRISA

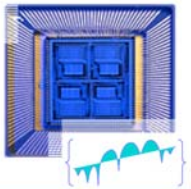
ST 140

- 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

IRISA

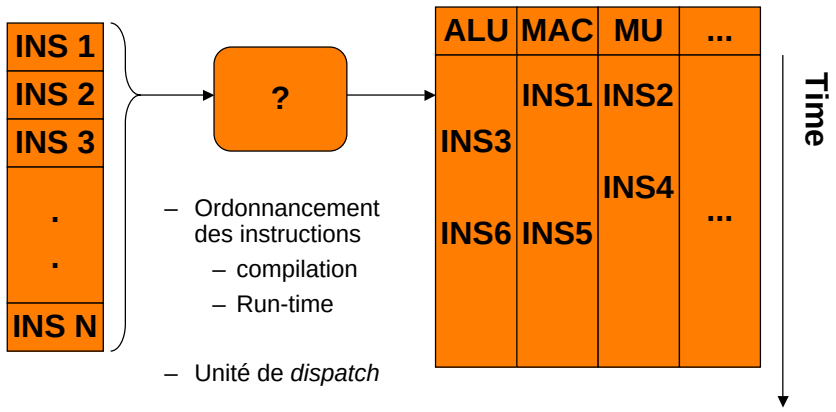
III. Évolution des DSP



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

IRISA

Parallélisme niveau instructions



– Ordonnancement des instructions
  – compilation
  – Run-time
– Unité de *dispatch*

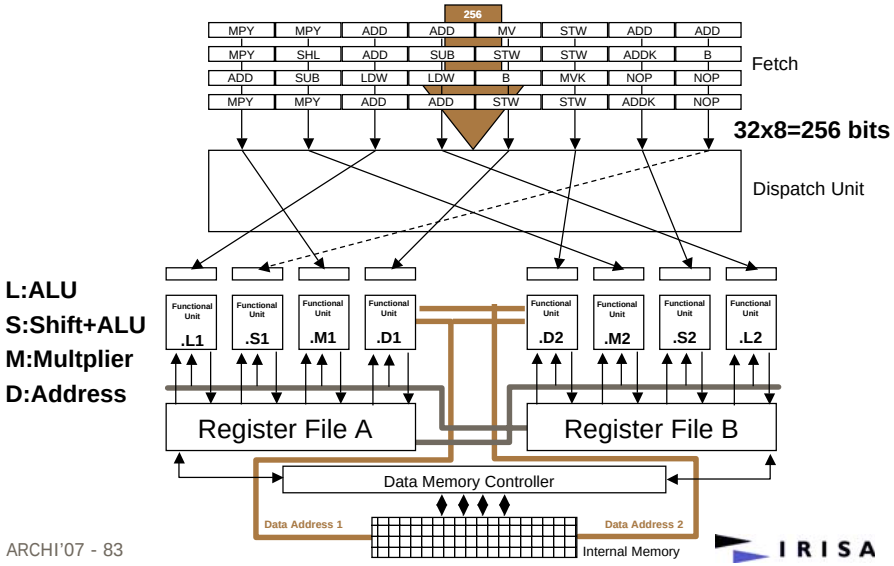
IRISA

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)

IRISA

VLIW : C62xx



IRISA

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

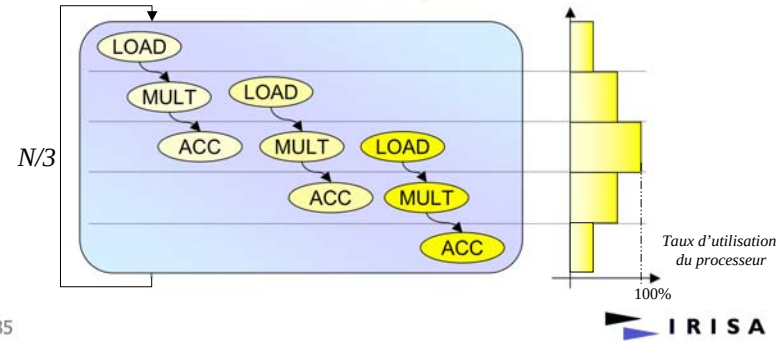
IRISA

Exploitation du parallélisme

- Déroulage des boucles : augmente l'ILP

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

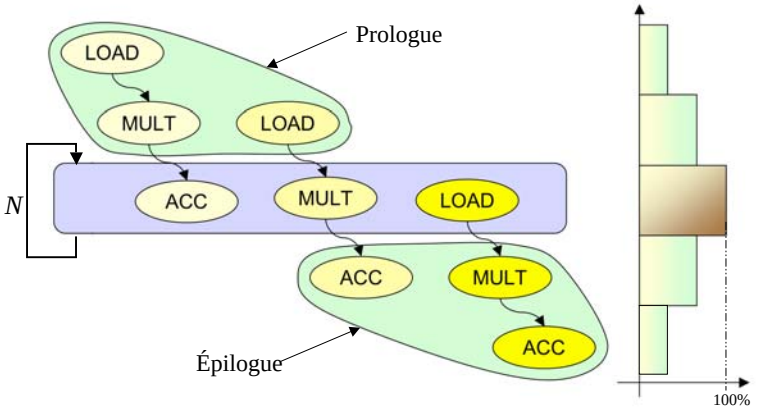
```
For(i=0;i<N;i+=3)
{
   ACC=ACC + x[ i] .h[ i]
   ACC=ACC + x[ i+1] .h[ i+1]
   ACC=ACC + x[ i+2] .h[ i+2]
}
```



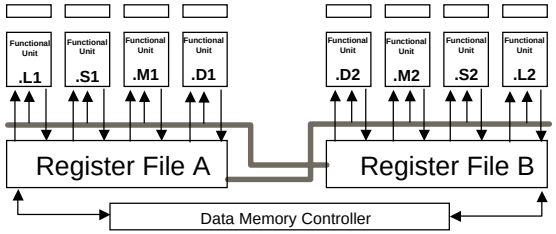
IRISA

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	-	-	-	-

IRISA

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       ; Intialize the loop counter (B0) to 8
2            MVKL .S1  0, A5       ; Intialize 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
```


IRISA

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

IRISA

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

IRISA

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

IRISA

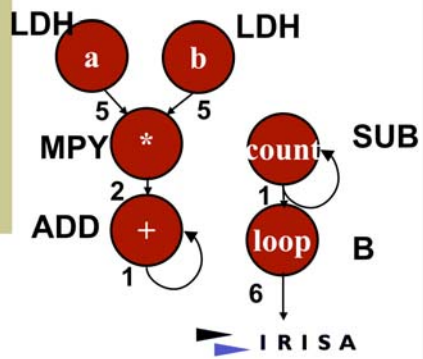
Remplacer les NOP Not Optimized Properly

```
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



IRISA

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

IRISA

Déroulage de boucles

- Déroulage + 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						
16			ADD						

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

IRISA

Pipeline logiciel

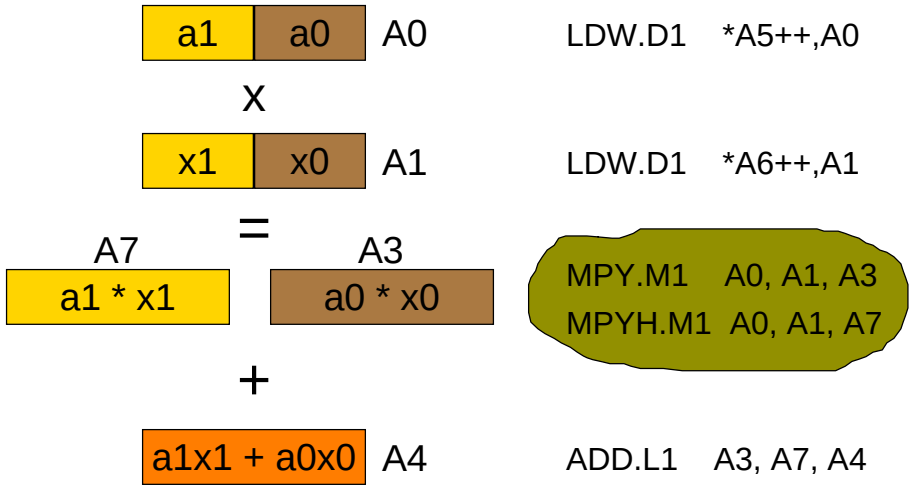
- Réintroduction de la boucle sur le motif répété

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

IRISA

Accès mémoire multiples



IRISA

Accès mémoire multiples

- Lecture de 4 données + 2 MAC en parallèle

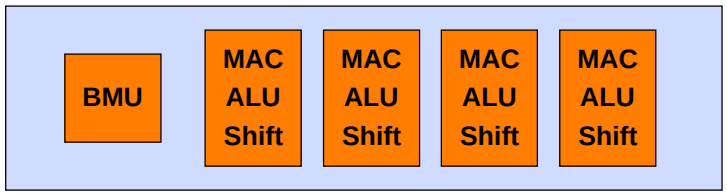
Cycle	.D1	.D2	.L1	.L2	.M1	.M2	.S1	.S2	NOP
1	LDW	LDW							
2	LDW	LDW							
3	LDW	LDW							
4	LDW	LDW							
5	LDW	LDW							
6	LDW	LDW			MPY	MPYH			
7	LDW	LDW			MPY	MPYH			
8	LDW	LDW	ADD	ADD	MPY	MPYH			
9			ADD	ADD	MPY	MPYH			
10			ADD	ADD	MPY	MPYH			
11			ADD	ADD	MPY	MPYH			
12			ADD	ADD	MPY	MPYH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

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

IRISA

StarCore SC1400 core

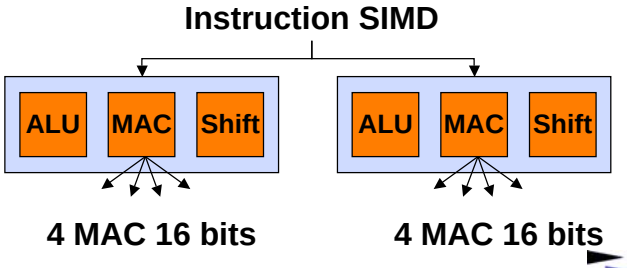
- Processeur VLIW 16 bits développé en commun par Agere (Lucent) et Freescale (Motorola)
- 300 MHz, faible tension
- Optimisé pour faible consommation
  - Meilleure densité de code (16 bits) que C6x
  - Pipeline plus simple (5 étages contre 11)



IRISA

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)



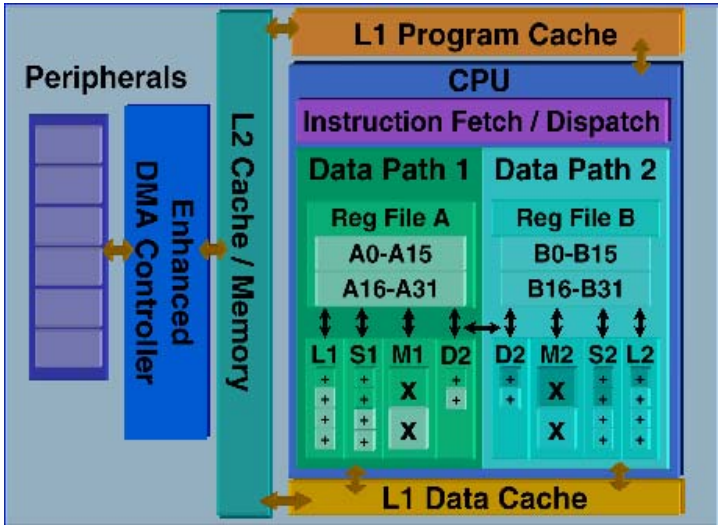
IRISA

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

IRISA

C64x



IRISA

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
   AGN0    LOOP
```

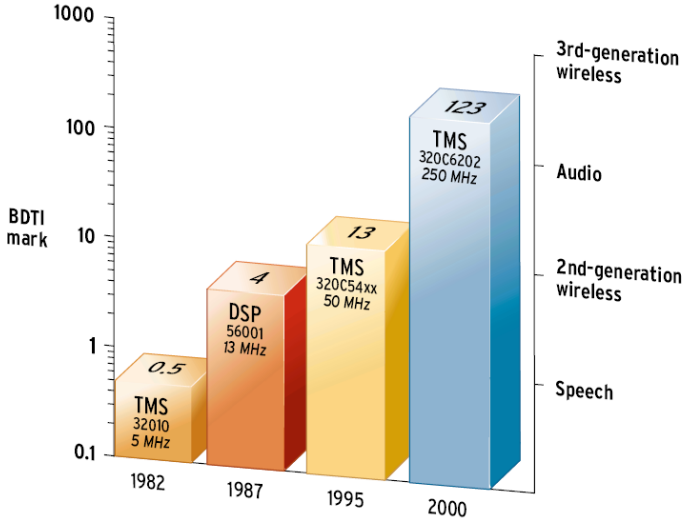

IRISA

Avantages et inconvénients

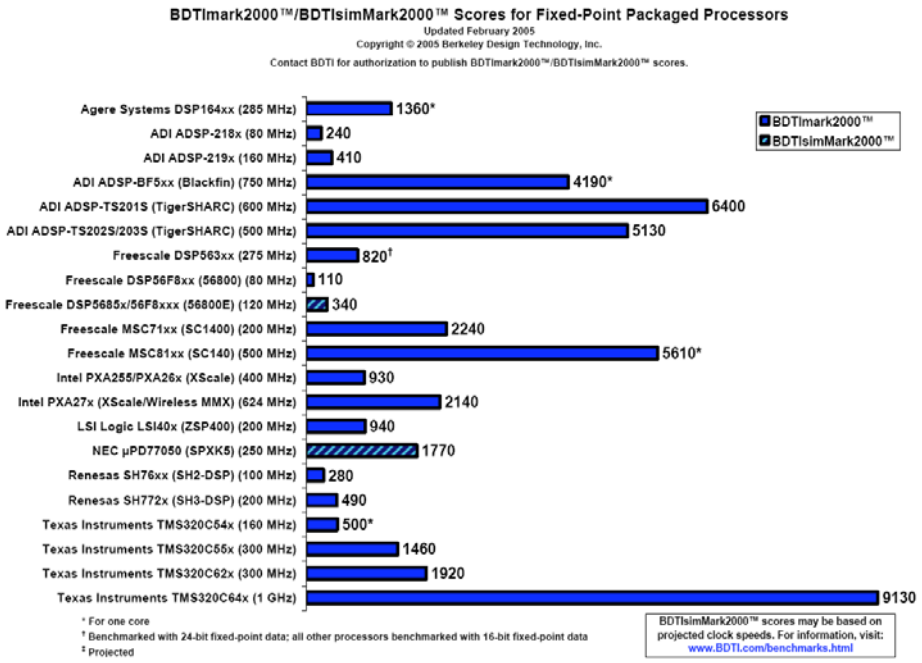
Méthode	Avantages	Inconvénients
VLIW	Grand bon dans les performances Architectures plus orthogonales ⇒ meilleures cibles pour les compilateurs	Bande passante vers la mémoire importante Forte consommation Séquencement délicat Augmentation de la taille du code importante
Superscalaire	Grand bon dans les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

IRISA

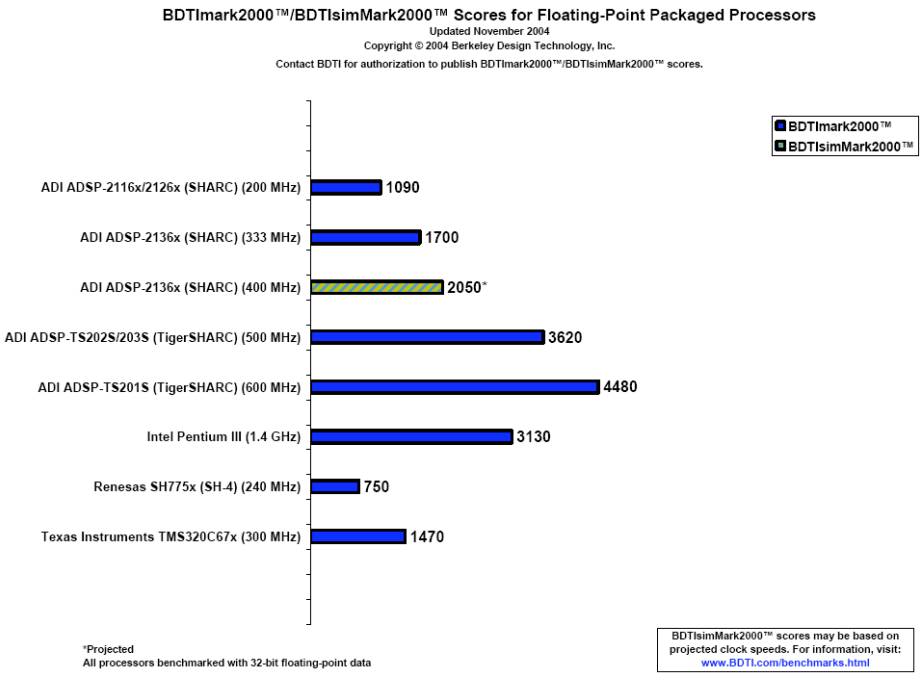
Générations : bilan



IRISA



IRISA

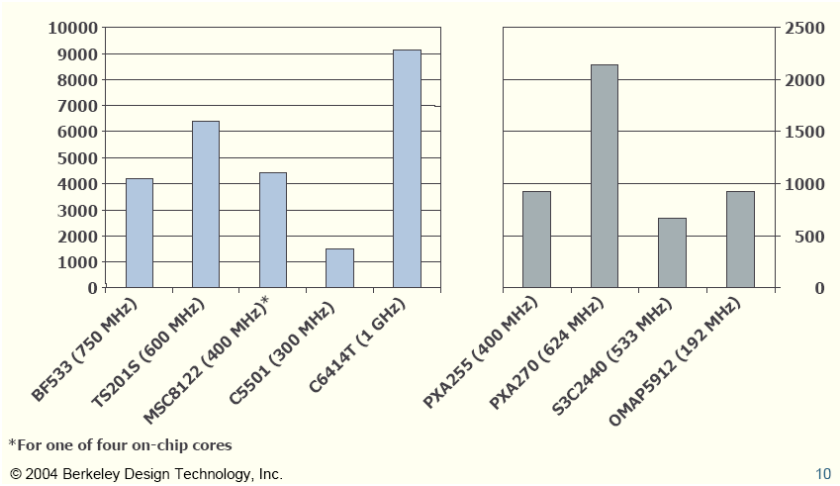


IRISA

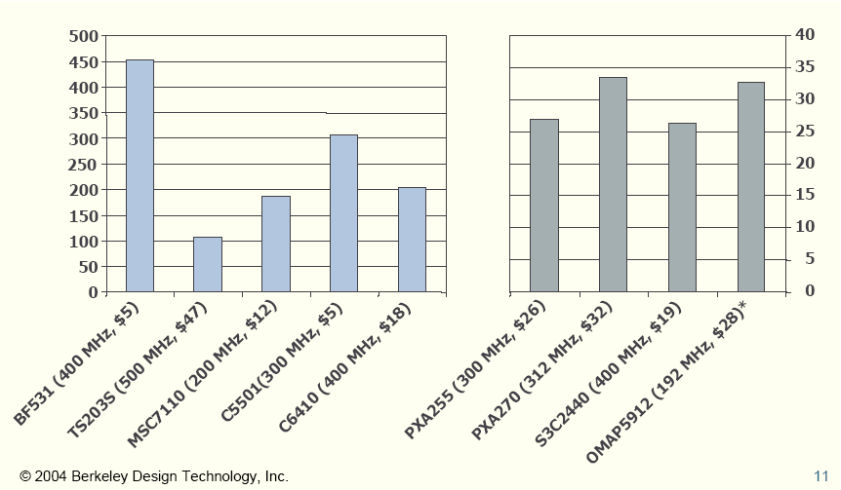
Processeurs récents (2006)

- 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 OMAP 591x (ARM9 + C55x)

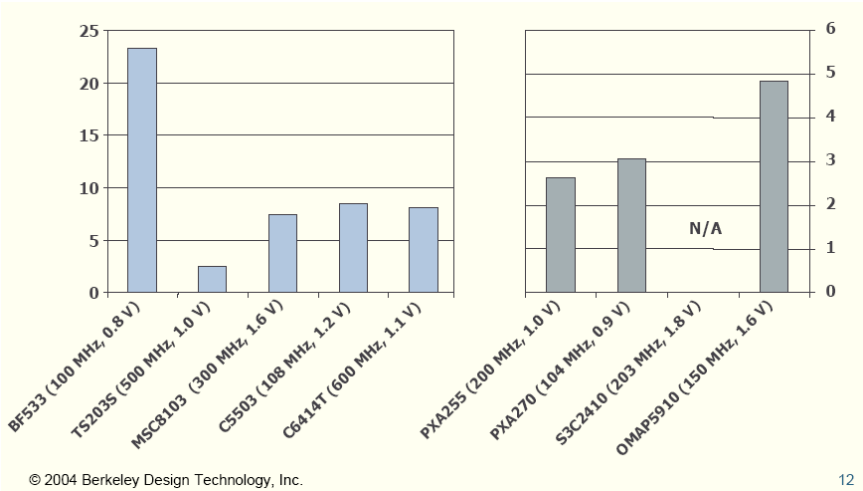
Vitesse (BDTImark2000)



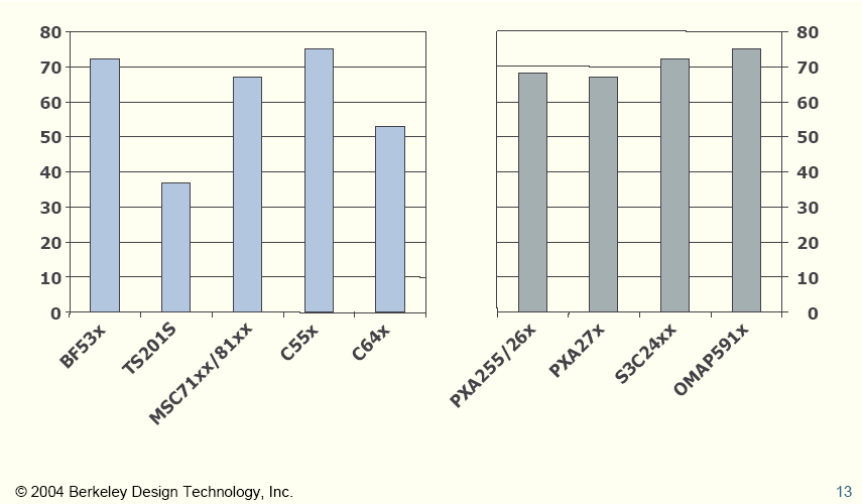
Coût – performance (mark/$)



Efficacité énergétique (mark/mW)

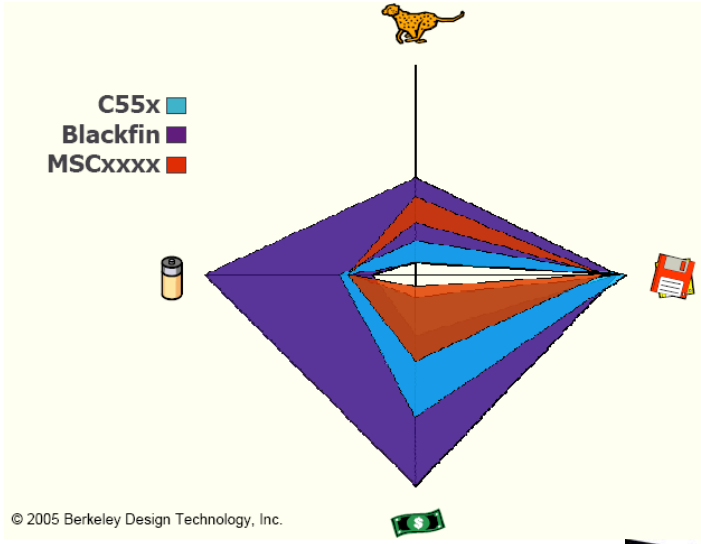


Efficacité mémoire (memmark)



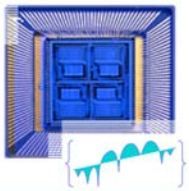
IRISA

BDTIMark



IRISA

III. Flot de développement

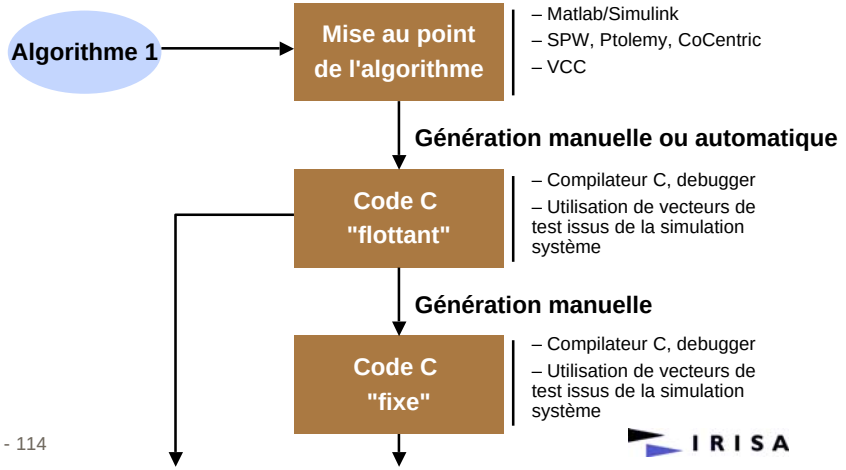


1. Flot général de développement
2. Compilation pour DSP
3. Arithmétique virgule fixe

IRISA

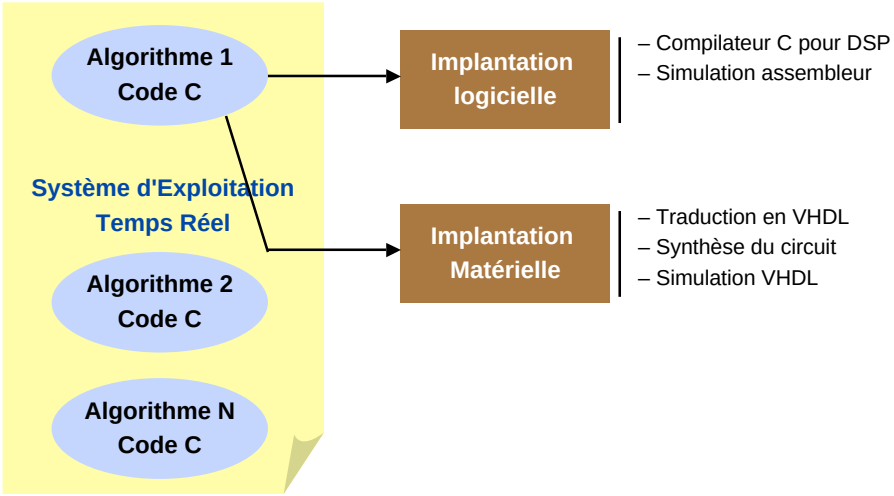
Flot de développement général

- Développement (actuel) d'applications (signal)



IRISA

Flot général (suite)



IRISA

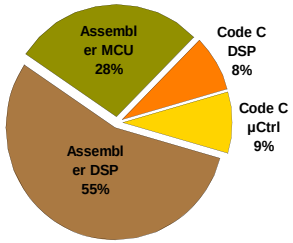
Outils de développement

- Spécifications flots de données du TS, prototypage rapide, simulation, validation
  - Matlab/Simulink, Cadence SPW, Synopsys Cossap, Ptolemy
- Outils de profiling
  - gprof, ...
- Compilateurs depuis des langages de *haut-niveau*
  - DSP en virgule fixe ou flottante
- Optimisations de l'assembleur
- Critères de sélection des outils et du langage
  - Compilateurs : taille et vitesse du code
  - Simulateurs : vitesse

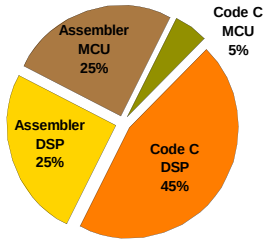
IRISA

Compilation

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



[Paulin97]

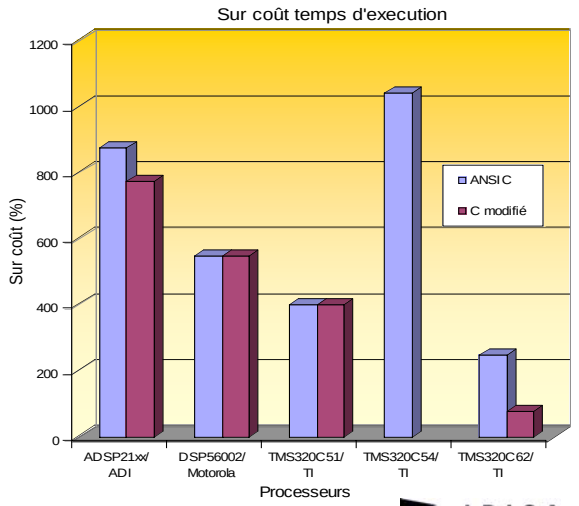


[Paulin00]

IRISA

Inefficacité des compilateurs C

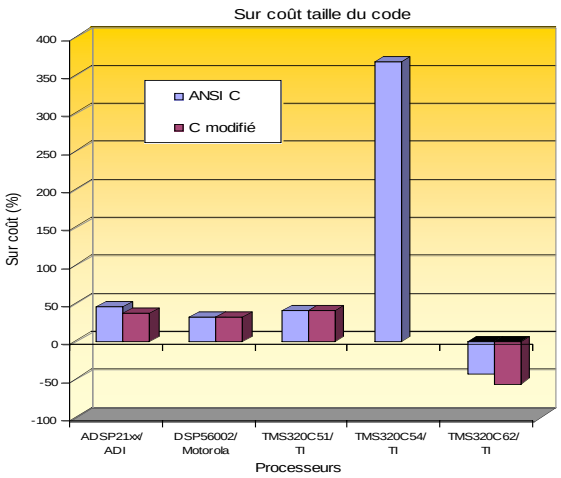
- Surcoût associé au compilateur
- Facteur 4 à 10 (virgule fixe)
- Facteur 2 à 4 (flottants)
- Orthogonalité
  - VLIW



IRISA

Inefficacité des compilateurs C

- Surcoût associé au compilateur
  - Taille du code



IRISA

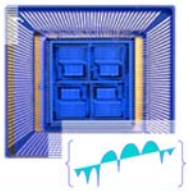
Raisons de l'inefficacité

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

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

IRISA

III. Flot de développement

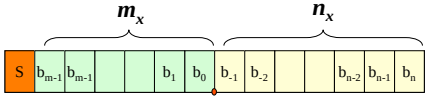


1. Flot général de développement
2. Compilation pour DSP
3. Arithmétique virgule fixe

IRISA

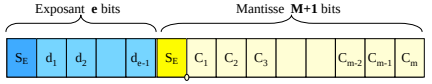
Formats de codage

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



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

  - o $b_x = m_x + n_x + 1$
  - o le format d'une donnée ne varie pas au cours du temps
- Virgule flottante
  - Représentation : exposant - mantisse



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

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

IRISA

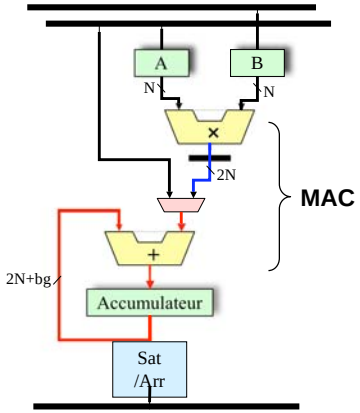
Comparaison fixe - flottant

- 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

IRISA

Eléments de l'UT

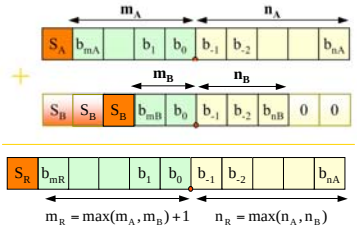
- Multiplieur câblé
- Additionneur (1 Op/cycle)
- U.A.L.
- Registres à décalage
  - ⇨ recadrage des données
- Gestion des débordements
- Bits de garde
- Registres d'accumulation
  - ⇨ stockage des données en double précision



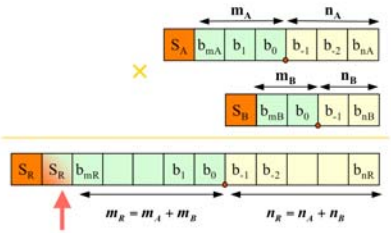
IRISA

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



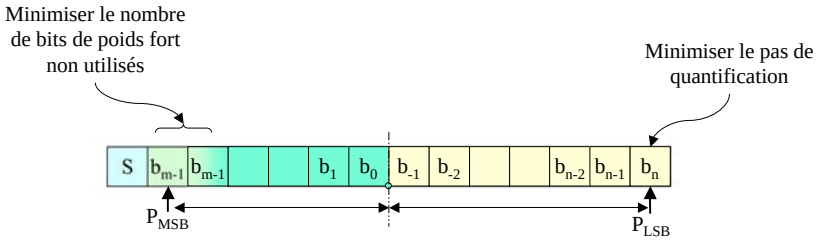
- Multiplication: a × b
  - Représentation identique
  - Doublement du bit de signe



IRISA

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



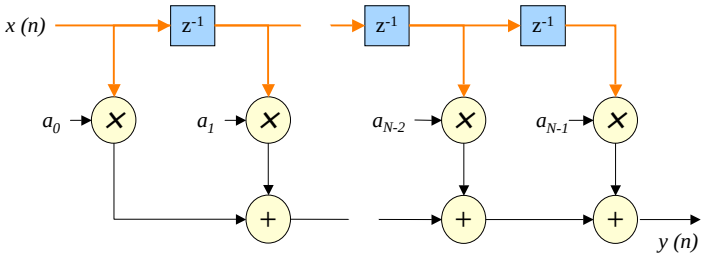
IRISA

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

Fil rouge : filtre FIR

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



Fil rouge : filtre FIR

```
float x[N] = {0.123, -0.569,...} /* Signal d'entrée du filtre */
float h[N] = {-0.0295, 0.0364,.., -0.0295 }; /* Coefficients du filtre */
int main() {
  float x[N], y[M], acc;
  int i,j;

  for(i=0; i<N; i++){x[i] = 0;}    /* Initialisation variables internes */

  for(j=0;  j<M; j++) {            /* Filtrage du vecteur d'entrée input */
        x[0] = Input[j];
        acc = x[0]*h[0] ;
        for(i=N-1; i>0; i--)
        {
          acc = acc + x[i]*h[i]; /* Calcul d'une cellule du filtre */
          x[i] = x[i-1];         /* Vieillissement du signal */
        }
        y[j] = acc;
   }
}
```


Dynamique des données

- Normes utilisées en TS
- Norme L1

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

➢ Méthode garantissant l'absence de débordements

- Norme Chebychev

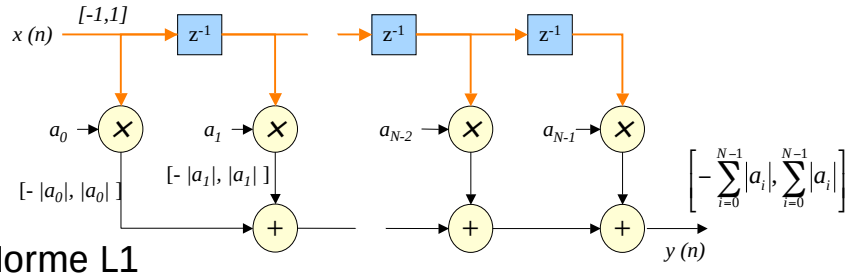
$$z_{\max 2} = \max_{n,\omega} \left(|x(n)|\right) \max\left(|H(\omega)|\right)$$

➢ Méthode garantissant l'absence de débordements pour un signal d'entrée à bande étroite e.g. *x(n)=cos(wt)*

- Arithmétique d'intervalle (non récursifs)

Filtre FIR

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



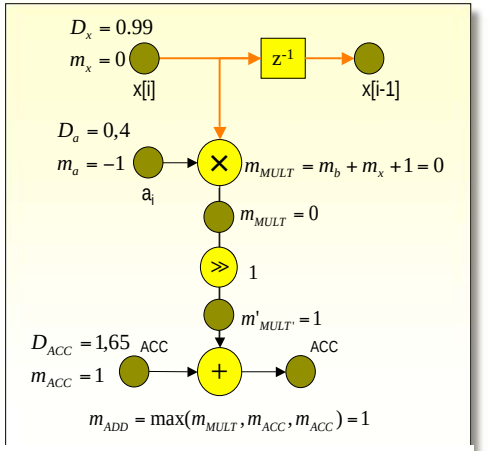
- Norme L1

$$\max_n\left(|y(n)|\right) = \max_n\left(|x(n)|\right)\sum_{m=-\infty}^{\infty}|h(m)| = \sum_{m=0}^{N-1}|a_i| = 1.65$$

IRISA

Filtre FIR

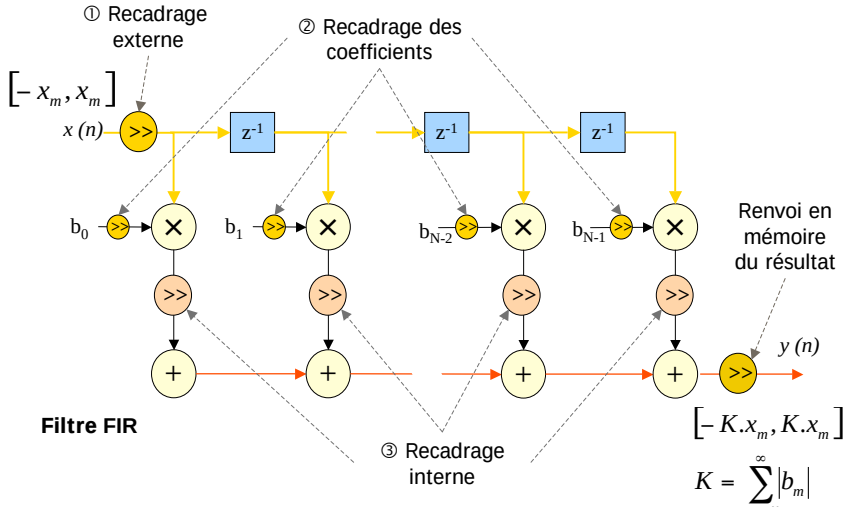
- Détermination de la position de la virgule pour les données
- Insertion des opérations de recadrage pour aligner la virgule



Graphe flot de données d'une cellule du FIR

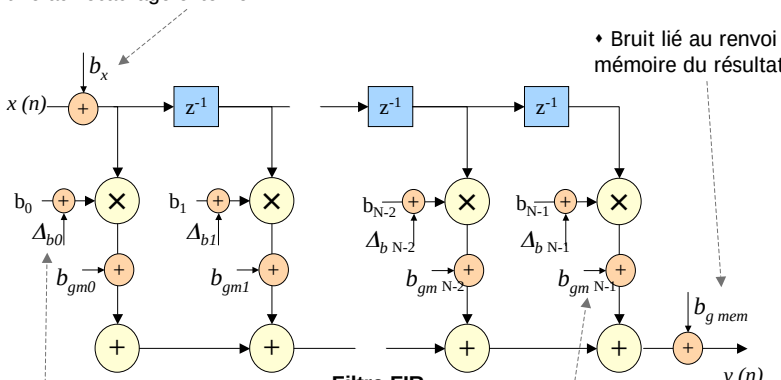
IRISA

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

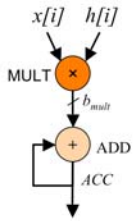


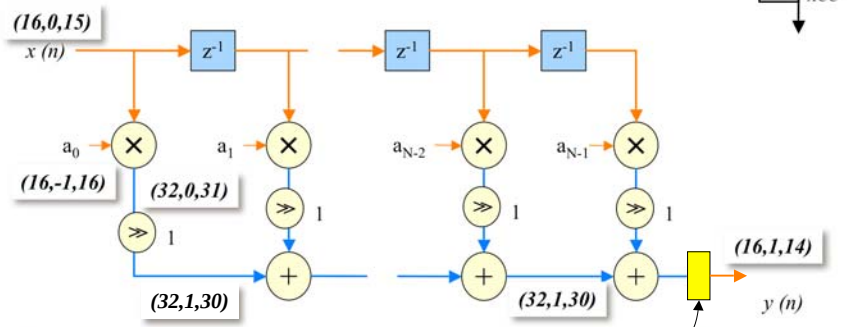
- Bruit lié au renvoi en mémoire du résultat
- Biais lié au codage des coefficients ②
- Bruit lié au recadrage interne ③

IRISA

Filtre FIR

- Architecture du processeur cible
  - Données en mémoire sur 16 bits
  - Multiplication 16 bits × 16 bits ⇨ 32 bits
  - Addition 32 bits + 32 bits ⇨ 32 bits





IRISA

Filtre FIR : code C virgule fixe

```
int x[M] = {-809, -6180, -1570, ...}        /* Signal x, codage (16,0,15)*/
int h[N] = {-1933, 2386, 3631,};            /* Coefficients (16,-1,16)  */

int main() {
 int x[N]; y[M];
 long  acc;
 int i,j;

 for(i=0; i<N; i++){x[i] = 0;}              /* Initialisation des variables internes du filtre */

 for(j=0;  j<M; j++) {                      /* Filtrage du vecteur d'entree input */
   x[0] = Input[j];
   acc = (long)(x[0]*h[0])>>1;

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

Le signal d'entrée et les coefficients sont spécifiés au niveau du code C en entiers sur 16 bits (absence de type virgule fixe en C) :
- l'entier représentant x (16,0,15) est obtenu en multipliant x par 2^{15}
- l'entier représentant h (16,-1,16) est obtenu en multipliant h par 2^{16}

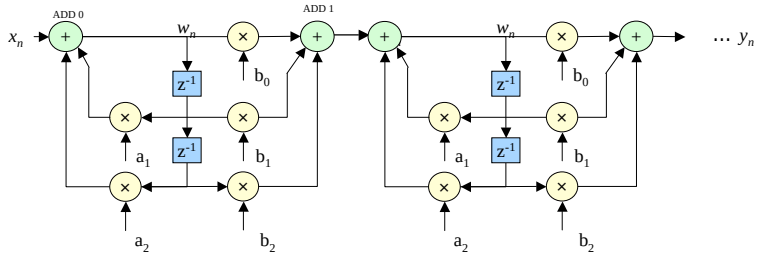
Recadrage de la sortie de la multiplication : changement de format : (32,0,31) ⇨ (32,1,30)

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

IRISA

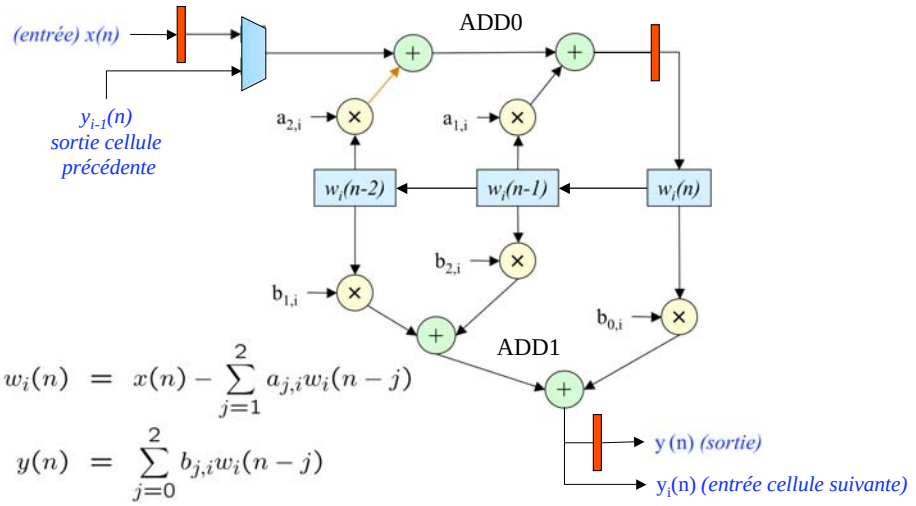
Filtre récursif (IIR)

- Cascade de cellules d'ordre 2



IRISA

Filtre IIR d'ordre 2 *(cellule i)*



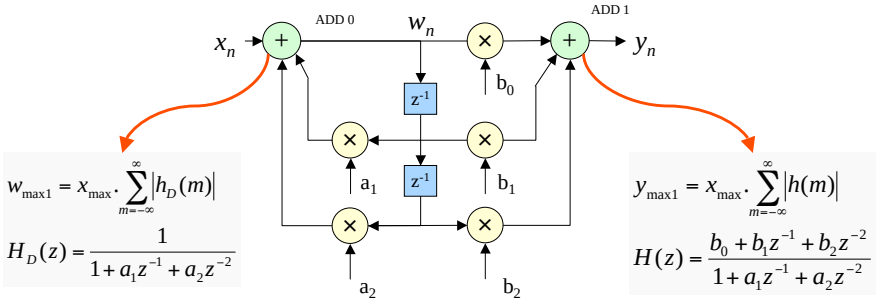
$$w_i(n) = x(n) - \sum_{j=1}^{2} a_{j,i} w_i(n-j)$$

$$y(n) = \sum_{j=0}^{2} b_{j,i} w_i(n-j)$$

IRISA

Dynamique des données (IIR)

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



IRISA

Dynamique des données (IIR)

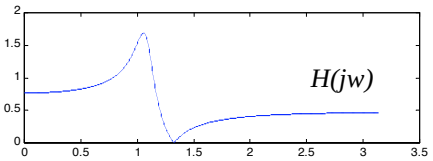
- Exemple

$x \in [-1,1]$

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

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

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



$H_D(jw)$

Codage des coefficients

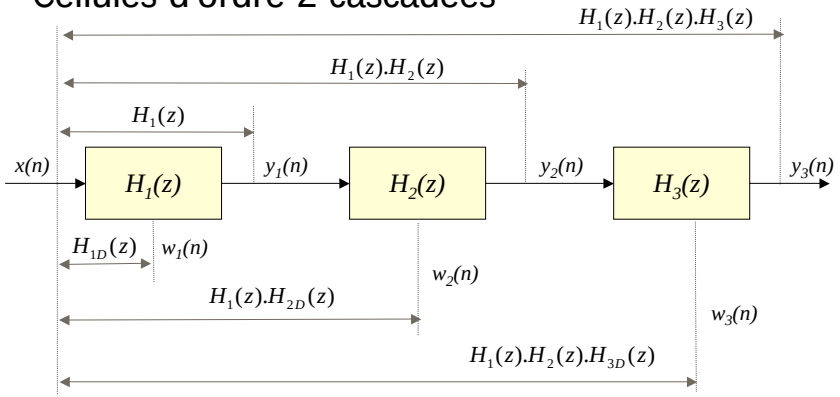
```
a1 = -27951
a2 =  27581
b2 =  16384
b0 =  16384
b1 = -7956
```

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

IRISA

Dynamique des données (IIR)

- Cellules d'ordre 2 cascadées



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

IRISA

Position de la virgule (IIR)

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

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

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

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

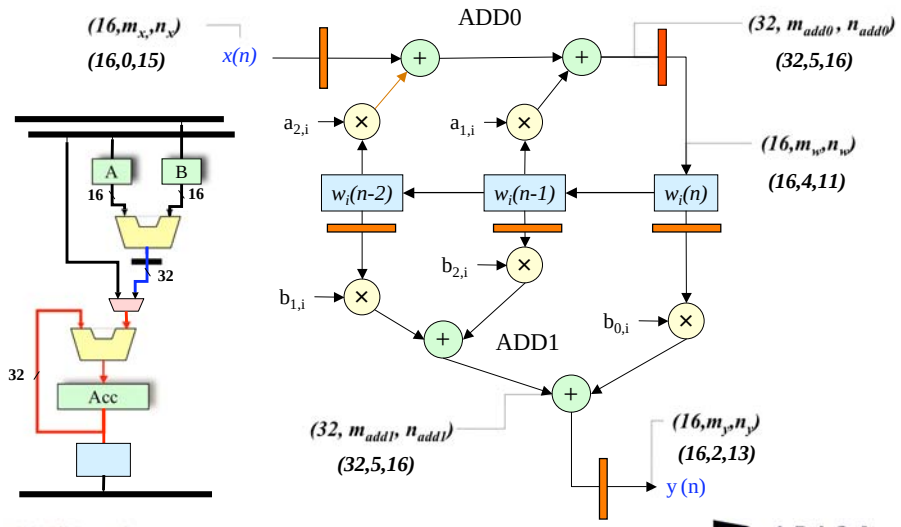
$$m_{b_{ij}} = \lceil \log_2 (\max_j (|b_{ij}|)) \rceil = 0$$

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

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

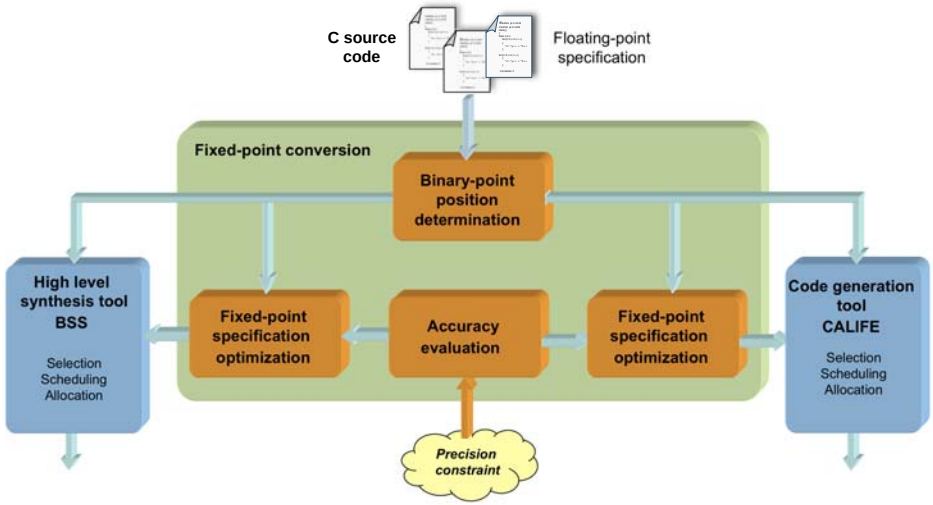
IRISA

Codage des données (IIR)



IRISA

Conversion flottant-fixe (IRISA)



[Ménard02, Ménard04]

IRISA

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

IRISA

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...

IRISA

Perspectives

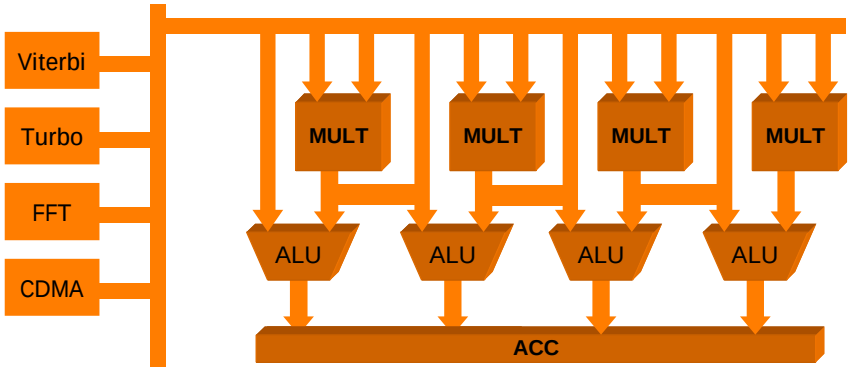
- Quelle sera l'architecture pour le traitement du signal et de l'image de demain (ou d'après demain) ?

IRISA

Piste 1 : DSP conventionnel

- DSP conventionnel multi-MAC
- Unités spécialisés

Programmation ?



IRISA

Piste 1 : DSP conventionnel

- DSP spécialisés
  - Texas Instruments OMAP 591x
    - Mobile 2-3G
    - ARM9 + C55x
  - Digital Media System-on-Chip (DMSoC): DaVinci
    - Codec vidéo
    - ARM926EJ-S Core
      - 202.5-MHz at 1.05 V or 256-MHz at 1.2 V
    - C64x+TM
      - 405-MHz at 1.05 V or 513-MHz at 1.2 V
    - Blocs dédiés au codage vidéo
    - Interface imager/DAC/HD

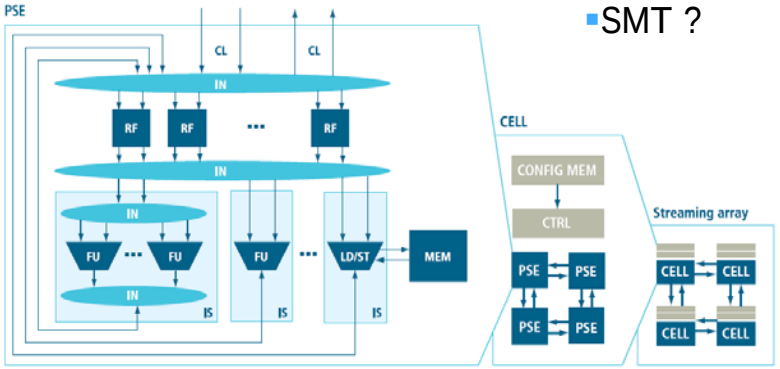
IRISA

Piste 2 : ULIW

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

Compilation ?

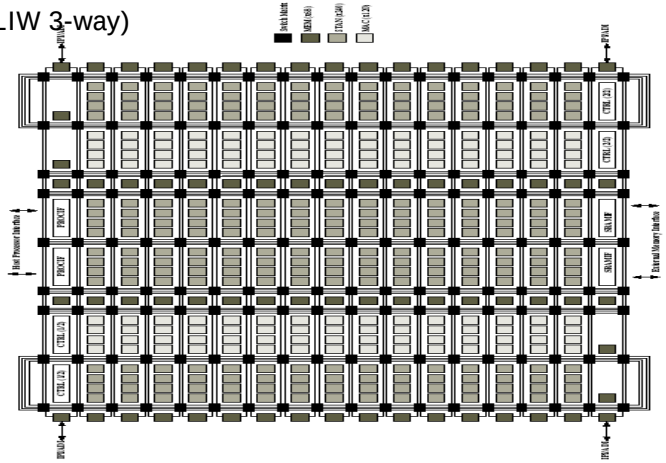
- SMT ?



IRISA

Piste 3 : multiprocesseur

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

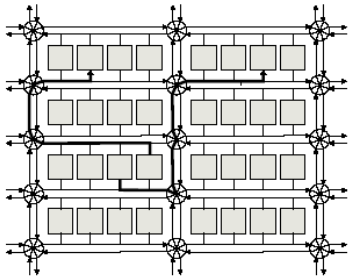


Device Characteristics	Parameter	Unit
LIW RISC processors	308	per device
Co-processors	14	per device
Processor clock	160	MHz
Peak processing capacity	197.1	GIPS
Total on-chip data bandwidth	3.3	Tbit/s
Peak 16-bit MACs	38.4	G-MACs
Internal bus speed	5.12	Gbit/sec
On-chip RAM	1003	KBytes
External SRAM (SDRAM)	8 (128)	MBytes
SRAM/SDRAM access speed	40/80/160	MHz
DMA channels	4 * 2.24	Gbit/sec
IPI (16-bit inter-device channels)	4 * 2.56	Gbit/sec
ADI (16-bit I/O channels)	8 * 2.4	Gbit/sec

IRISA

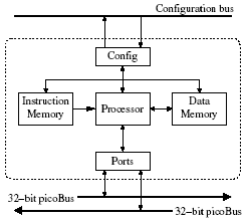
PicoArray

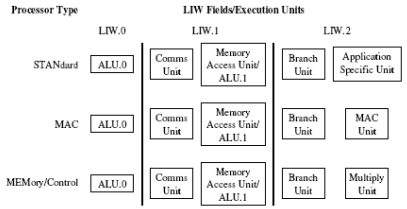
- Static Interconnect



- VHDL: structure
- C/Assembly: PE

- PE: 16-bit 3-way VLIW

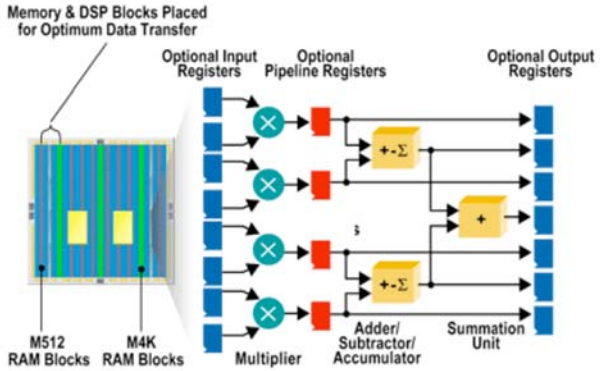




IRISA

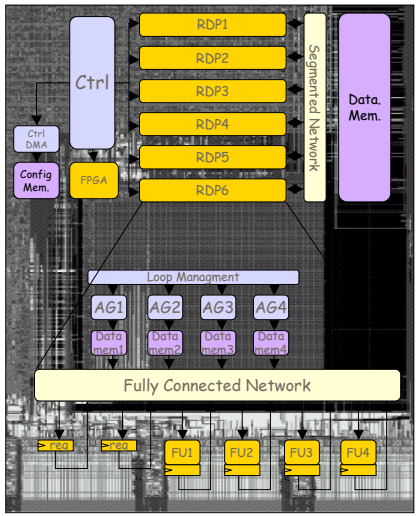
Piste 4 : FPGA

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



IRISA

Piste 5 : processeur reconfigurable

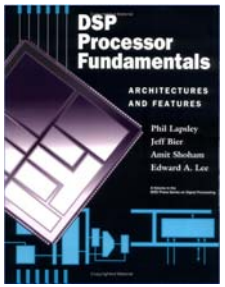


- Architecture DART
  - 3G/UMTS Mobile Terminal
  - 802.11a (Channel Est.)
  - IRISA/R2D2
  - STMicroelectronics
  - CEA LIST/LETI
- Performances
  - 5-10 GOPS/cluster@0.13m
  - 200MHz, 11mm^2
  - 300 mW @ 5 GOPS
  - 16 MOPS/mW @ 5 GOPS
  - Circuit en juin 2005

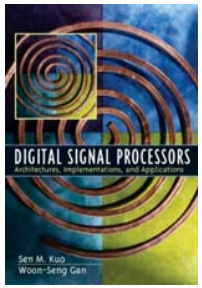
IRISA

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



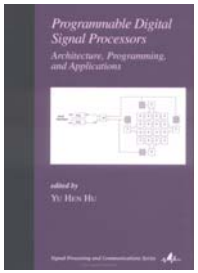
- Digital Signal Processors : Architectures, Implementations, and Applications by Sen M. Kuo, Woon-Seng S. Gan, Prentice Hall, 2004



IRISA

Bibliographie

- Programmable Digital Signal Processors by Yu Hen Hu (Editor), Marcel Dekker, 2001



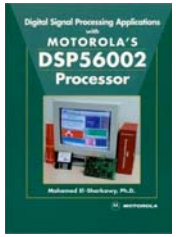
- VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis by Vijay Madisetti, Butterworth-Heinemann, 1995

- Méthodes et architectures pour le TSI en temps réel, de Didier Demigny, Hermès, 2002

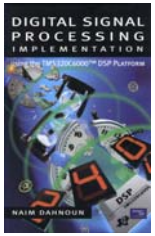
IRISA

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

IRISA

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, april 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.

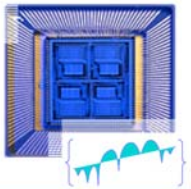
IRISA

Webographie

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

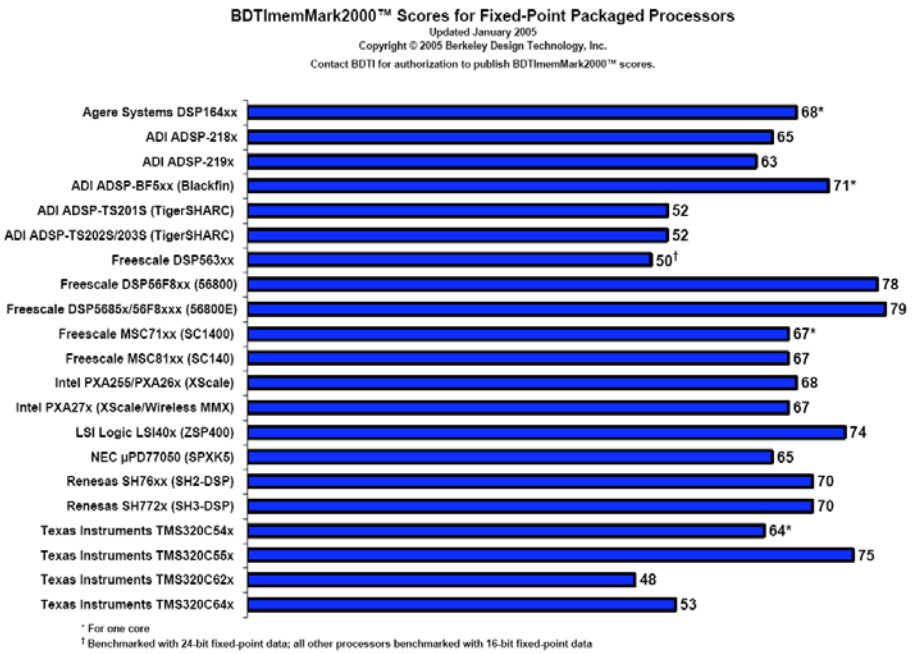
IRISA

Annexes

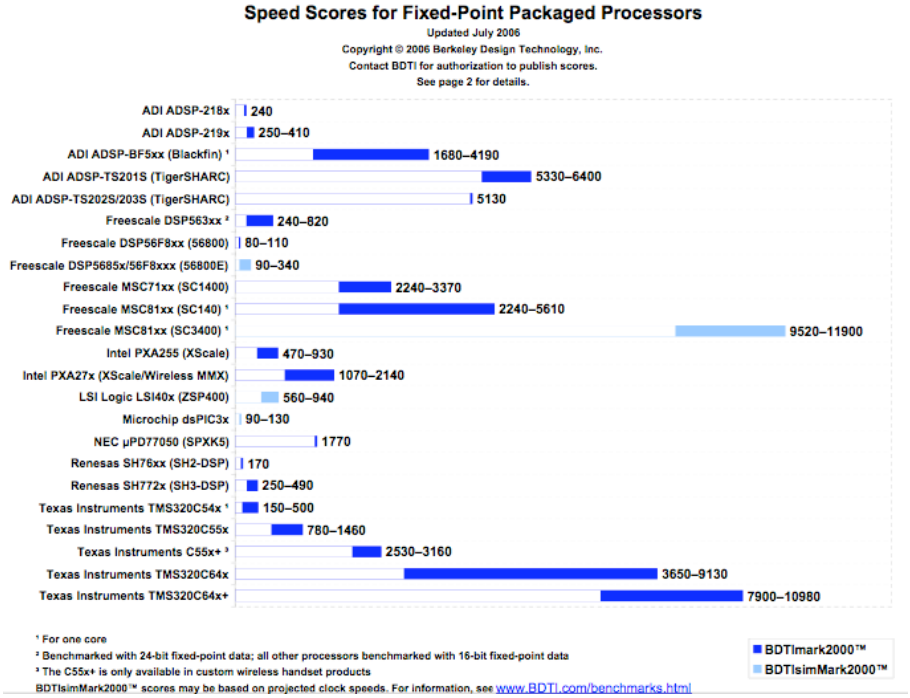


- BDTIMark
- Caractéristiques complémentaires
- Génération de code pour DSP

IRISA



IRISA

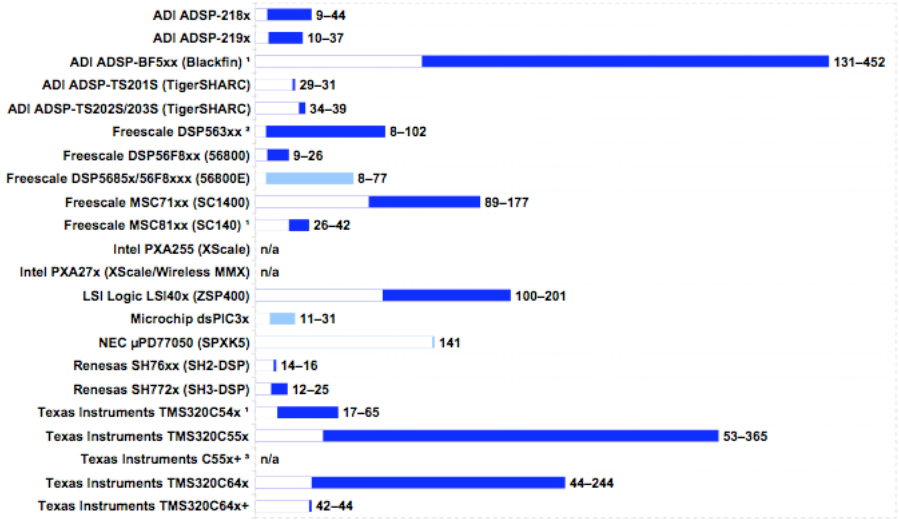


Speed per Dollar Ratios for Fixed-Point Packaged Processors

Updated May 2006

Contact BDTI for authorization to publish scores.

See page 2 for details.



[1] For one core

[2] Benchmarked with 24-bit fixed-point data; all other processors benchmarked with 16-bit fixed-point data

[3] The C55x+ is only available in custom wireless handset products

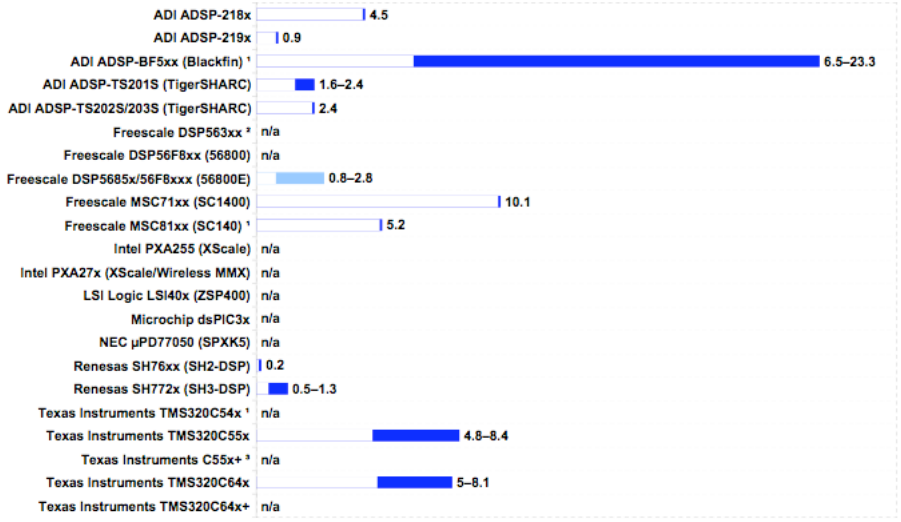
BDTIsimMark2000™ scores may be based on projected clock speeds. For information, see www.BDTI.com/benchmarks.html

Speed per Milliwatt Ratios for Fixed-Point Packaged Processors

Updated May 2006

Contact BDTI for authorization to publish scores.

See page 2 for details.



[1] For one core

[2] Benchmarked with 24-bit fixed-point data; all other processors benchmarked with 16-bit fixed-point data

[3] The C55x+ is only available in custom wireless handset products

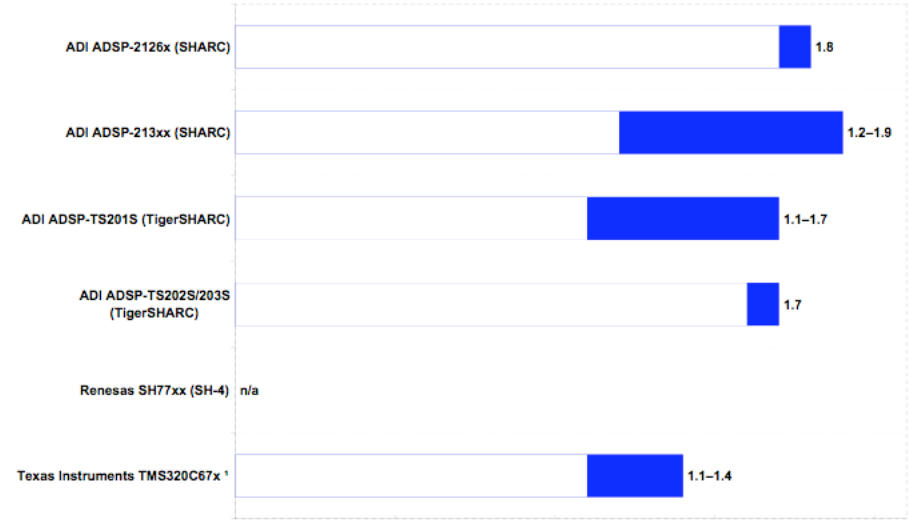
BDTIsimMark2000™ scores may be based on projected clock speeds. For information, see www.BDTI.com/benchmarks.html

Speed per Milliwatt Ratios for Floating-Point Packaged Processors

Updated May 2006

Contact BDTI for authorization to publish scores.

See page 2 for details.



[1] Score does not apply to TMS320C67x+ parts (e.g., the TMS320C672x)

All processors benchmarked with 32-bit floating-point data.

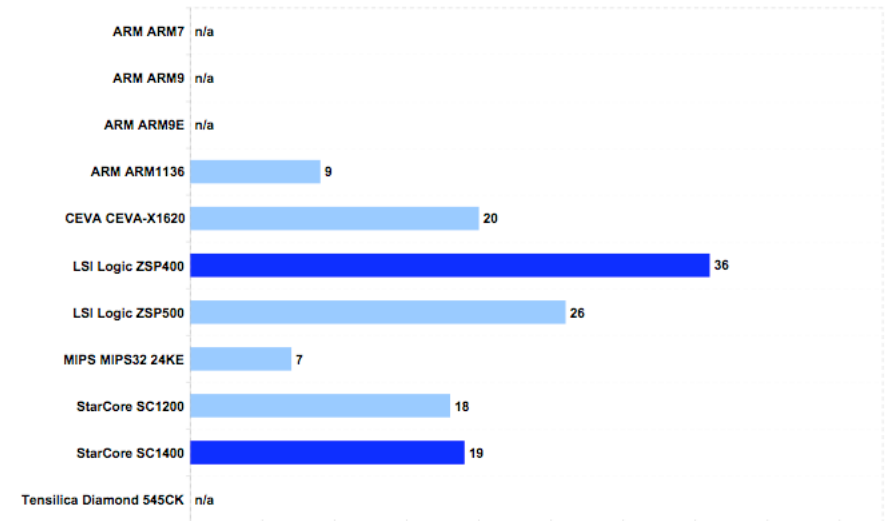
BDTIsimMark2000™ scores may be based on projected clock speeds. For information, see www.BDTI.com/benchmarks.html

Speed per Milliwatt Ratios for Fixed-Point Licensable Cores

Updated September 2006

Contact BDTI for authorization to publish scores.

See page 3 for details.



All scores use worst-case clock speeds for the TSMC CL013G process and ARM Artisan SAGE-X library

Vendors can choose different speed/area/power trade-offs; to understand the trade-offs, please view all BDTI metrics for each core.

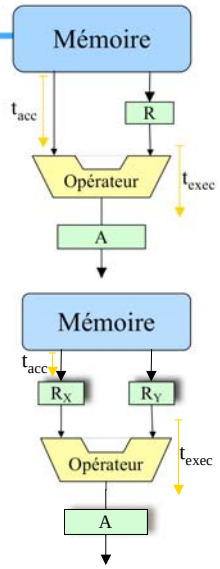
BDTIsimMark2000™ scores may be based on projected clock speeds. For information, see www.BDTI.com/benchmarks.html

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

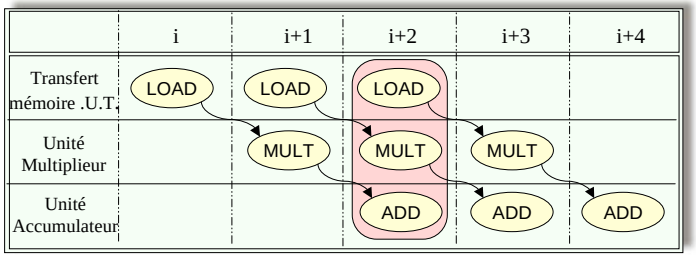


IRISA

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
```

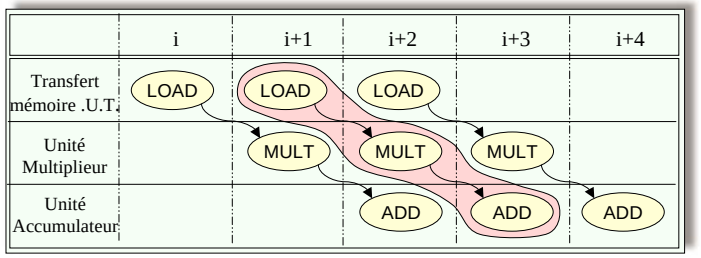


IRISA

Codage des instructions

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

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



IRISA

Structures de contrôle

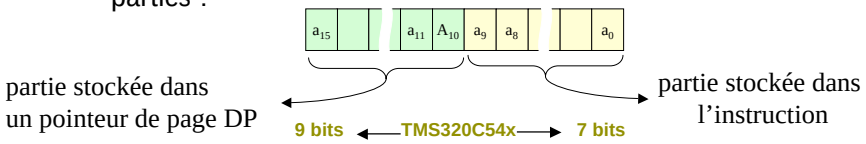
- Boucle matérielle
  - Optimiser le traitement des boucles de petite taille
    - Initialisation des paramètres de la boucle en 1 instruction
    - Pas d'instructions supplémentaires pour la gestion de la fin de la boucle
- Instructions de branchement
  - Branchement multi-cycles: ajout de NOP entre BR et 1ère instruction
  - Branchement retardé
- Instructions à prédicat
  - Instructions conditionnelles

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


IRISA

Modes d'adressage

- **Adressage immédiat :** la donnée est stockée directement dans l'instruction (Ex C54x: LD #248, A)
- **Adressage registre directe :** les données sont stockées dans des registres (Ex C54 : SUB A, B)
- **Adressage mémoire directe :** l'adresse de la donnée est stockée dans l'instruction
  - Pour limiter le nombre de bits stockés dans l'instruction un adressage paginé est utilisé. L'adresse est composée de deux parties :



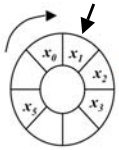
IRISA

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 *ARx+0, A — addr = AR1; AR1 = AR1 + AR0
      - MR: registre d'index
    - modulo : $(AR:= AR \pm 1)_N$ — LD *AR1+% ,A — addr = AR1; AR1 = circ(AR1 + 1)



(%) specifies the size of the circular buffer.

    - bit inversé : FFT — LD *ARx+0B ,A — addr = ARx; ARx = B(ARx + AR0)

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

IRISA

Génération de code pour DSP

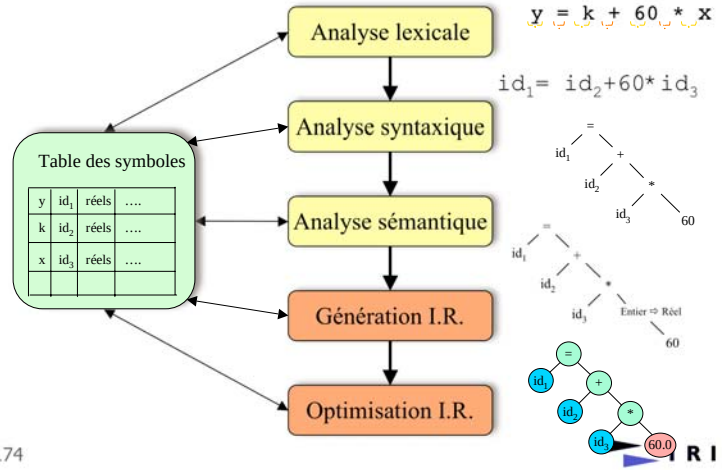
- Génération d'un code de qualité
  - rapide (Texec faible)
  - compacte (Tcode faible)
  - sûr de fonctionnement
- Intégration des spécificités des DSP
  - arithmétique virgule fixe
  - spécificités des unités de mémoire et de contrôle
- Flexibilité du compilateur
  - compilateur « reciblable »

[Leupers97,Marwedel99]

IRISA

Partie frontale

- But: transformation du code source en une représentation intermédiaire (I.R.)



[SUIF]

IRISA

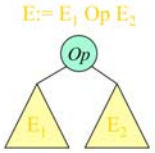
Partie finale

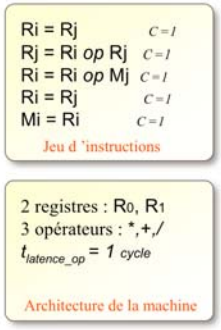
- But: génération d'un code cible à partir de la représentation intermédiaire
- Sélection d'instructions [SPAM, CHESS, CodeSyn, Calife]
  - o Reconnaissance de motifs [Leupers97, Leupers99]
  - o Couverture d'arbre ⇨ Programmation dynamique
- Allocation et assignation de registres
  - o allocation de registres ⇨ Coloration de graphe
  - o assignation de registres [Araujo95, Wess95]
- Ordonnancement
  - Ordonnancement par liste, ILP, de traces
- Optimisations dépendantes de l'architecture
  - o adresses mémoire [Liao96, Sudarsanam95]

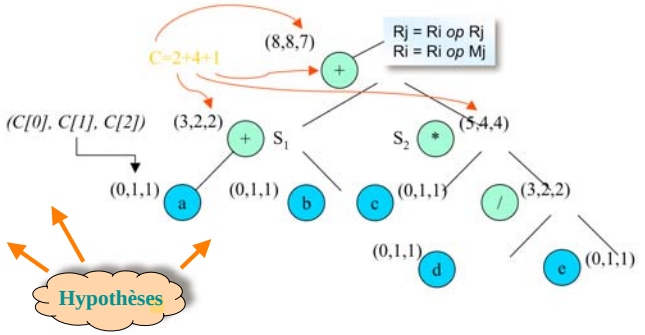
IRISA

Sélection de code

- Principe :
  - découpage du problème de production de code optimal pour E en sous-problèmes optimaux de production de code pour les sous-expressions E1 et E2







IRISA

Post-optimisations

- Post-optimisations
  - instructions de changement de mode
  - assignation des bancs de mémoire
  - l'allocation mémoire et la génération des adresses
- Génération des adresses : exploiter les U.G.A.
  - ↳ registres d'adresse : ARi
  - ↳ post-modifications (ARi++, ARi--)
  - ↳ registres d'offset : MRk (ARi= ARi ± MRk)

⇨ Optimiser le placement des données en mémoire afin d'utiliser efficacement les UGA

IRISA