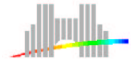


DSP : des processeurs dédiés au traitement numérique du signal

Arnaud Tisserand
INRIA LIP Arénaire

Séminaire LIP
mai 2003

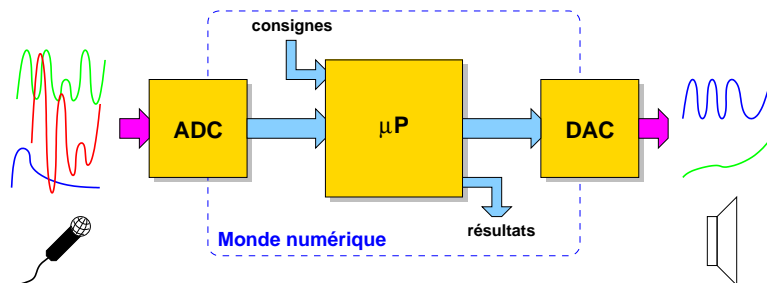


- 1 Besoins et contraintes en traitement numérique du signal
 - applications typiques
 - algorithmes typiques
- 2 Architecture des DSP
 - unités de calcul et types de données
 - structure mémoire
 - contrôle spécifique aux DSP
- 3 Exemples de DSP
 - processeurs actuels
 - évolution
 - comparaisons avec les processeurs généralistes

A. Tisserand – mai 2003 – séminaire LIP – Processeurs DSP

2/47

Structure générale des applications de TNS



TNS = traitement numérique du signal

ADC = Analog to Digital Converter

DAC = Digital to Analog Converter

Pourquoi faire du traitement numérique du signal ?

Les principaux avantages du calcul numérique sur le calcul analogique s'appliquent aussi au traitement du signal. On a par exemple :

- Grande résistance aux bruits
 - ▶ variations des tensions d'alimentation
 - ▶ variations de température
 - ▶ interférences électromagnétiques (EMI)
- Indépendance par rapport aux tolérances de fabrication
- Précision arbitraire
- Stabilité dans le temps
- Stockage des données sans dégradation
- Duplication des valeurs sans altération
- Programmation flexible
- Développement rapide

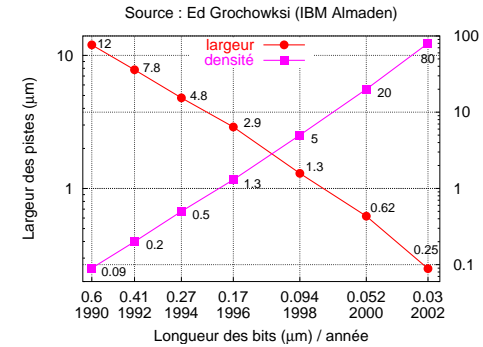
Applications et algorithmes classiques

- Radars et sonars
 - (Télé)communications :
 - ▶ modems
 - ▶ réseaux
 - ▶ téléphones cellulaires
 - Disques durs
 - Appareils audio
 - ▶ lecteurs CD, DVD, MP3. . .
 - ▶ prothèses auditives
 - ▶ synthétiseurs
 - ▶ reconnaissance de la parole
 - Vidéo
 - Automobile
 - Robotique
 - . . .
- Filtrage
 - Transformées
 - Codage/décodage
 - Compression/décompression
 - Cryptage/décryptage
 - Contrôle
 - Reconnaissance de la parole
 - Synthèse de signaux
 - Élimination d'écho
 - Estimation spectrale
 - . . .

Application type : les disques durs

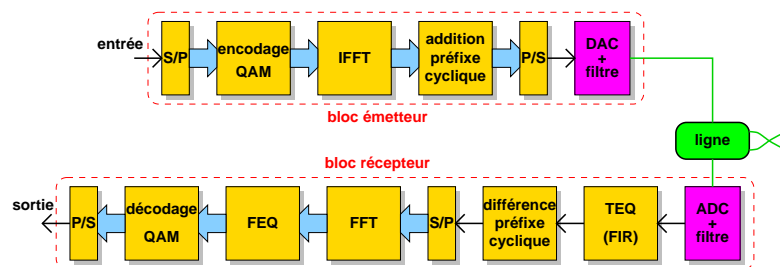
Contrôle de la position du bras supportant les têtes de lecture/écriture.

- 6 mois de développement pour 18 mois de cycle de vie
- vitesse de rotation de 5 400 à 15 000 tr/min.
- précision :



Application type : MODEM ADSL

ADSL = *Asymmetrical Digital Subscriber Line*



Besoins en calcul de 100 à 150 MIPS.

QAM = *Quadrature Amplitude Modulation*

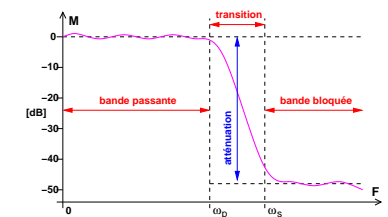
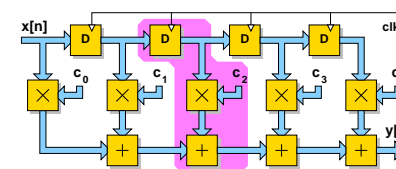
TEQ = *Time domain Equalizer*

FEQ = *Frequency domain Equalizer*

Algorithme type : les filtres FIR

Pour un filtre à réponse impulsionnelle finie (*Finite Impulse Response*) de taille N (nombre de coefficients), on doit effectuer un calcul du type :

$$y[n] = \sum_{i=0}^{N-1} c_i \times x[n-i]$$

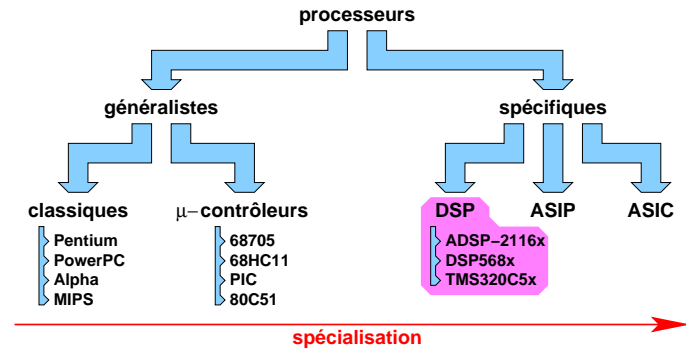


Contraintes : nb. de coefficients (10 → 300), précision (16 → 24 bits, un peu plus en interne), vitesse de fonctionnement. . .

Besoins en TNS

- Calculs rapides :
 - ▶ multiplications accumulations rapides
 - ▶ bande passante mémoire importante
 - ▶ support matériel pour accélérer le contrôle des algorithmes
- Contraintes temps réel :
 - ▶ entrées/sorties à débit fixe
 - ▶ prédiction des temps de réponse
- Contraintes des systèmes embarqués :
 - ▶ basse consommation d'énergie
 - ▶ taille des programmes
 - ▶ utilisation mono-circuit
- Production de masse
 - ▶ faible coût
 - ▶ rapidité de développement

Tentative de classification des processeurs



- DSP = *Digital Signal Processor*¹
- ASIP = *Application Specific Instruction set Processor*
- ASIC = *Application Specific Integrated Circuit*

¹Suivant le contexte DSP signifie aussi traitement numérique du signal : *Digital Signal Processing*.

Quel processeur utiliser pour le TNS

- μ-contrôleurs : pas assez performants
- ASIC : beaucoup trop coûteux, mise en œuvre complexe et longue
- ASIP : beaucoup trop coûteux, définition complexe
- processeurs généralistes : pas temps réel, trop coûteux, consomment trop d'énergie, difficilement embarquables

Solution :

Faire des processeurs **spécifiques** pour le traitement numérique du signal.

⇒ DSP

Utilisation des différents types de processeurs

Source : D. Tennenhouse (DoR Intel) RTSS'99.

type de processeur		unités vendues × 10 ⁶
embarqué	4 bits	2 000
	8 bits	4 700
	16 bits	700
	32 bits	400
DSP		600
généralistes classiques		150

Types de données

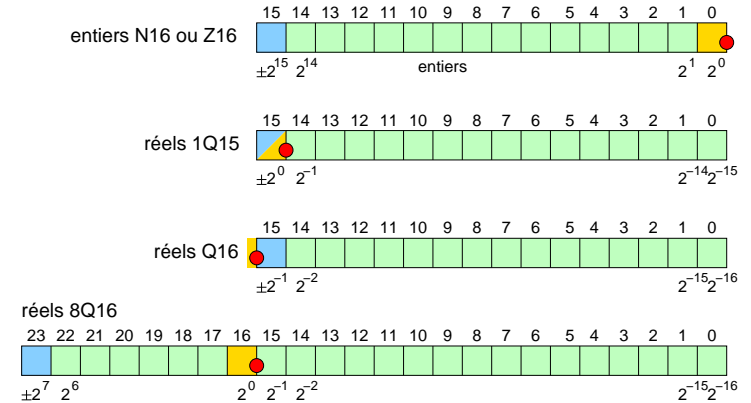
Les entrées/sorties représentent souvent des grandeurs physiques. On a donc besoin de manipuler des réels avec une précision limitée, typiquement entre 10 et 24 bits.

Toutefois, pour éviter certains problèmes de précision et de faux dépassement de capacité lors de longues séquences de calcul, on peut aussi avoir des types de données intermédiaires plus grands (ex : accu. 40 bits).

Codages classiques :

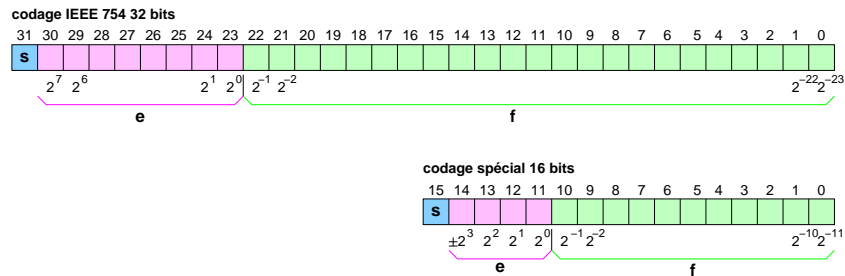
- Codage en **virgule fixe** (la grande majorité des DSP) :
 - ▶ mots de 16 ou 24 bits (très souvent)
 - ▶ mots de 20 ou 32 bits (rarement)
- Codage en **virgule flottante** :
 - ▶ mots de 32 bits (plus ou moins compatible IEEE 754)
 - ▶ mots de 16, 24 ou 32 bits (avec des codages particuliers)

Codages en virgule fixe



Valeurs signées en complément à 2 (parfois en signe/magnitude).

Codages en virgule flottante



$$x = (-1)^s \times (1.f) \times 2^e$$

La mantisse est composée du 1 implicite et de la fraction f . Suivant les DSP, il y a des codages différents de l'exposant (avec ou sans biais) et des valeurs particulières (0, $\pm\infty$, NaN).

Codage flottant IEEE 754 simple précision (32 bits)

caractéristique	valeur
largeur des mots	32 bits
taille mantisse	24 bits (23 fraction + 1 implicite)
taille exposant	8 bits
intervalle mantisse	$[1, 2 - 2^{-23}]$
biais exposant	127
nb. ordinaire	$e + biais \in [1, 254], x = (-1)^s \times 1.f \times 2^{e-biais}$
zéro (± 0)	$e + biais = 0, f = 0$
Not a Number (NaN)	$e + biais = 255, f \neq 0$
Infinis ($\pm\infty$)	$e + biais = 255, f = 0$
nb dénormalisé	$e + biais = 0, f \neq 0, x = (-1)^s \times 0.f \times 2^{-126}$
min	$\approx 1.2 \times 10^{-38}$
max	$\approx 3.4 \times 10^{38}$

Les nombres dénormalisés ne sont pas supportés dans les DSP.

Virgule fixe ou virgule flottante ?

Quelques considérations de choix :

- Précision : souvent un faux problème
 - ▶ 23 bits de mantisse du flottant proche de 24 bits en virgule fixe
- Dynamique des nombres représentables.
- Complexité du code du fait des opérations de recadrage en virgule fixe.
- Performances :
 - ▶ vitesse : DSP virgule fixe sont plus rapides
 - ▶ taille : DSP virgule fixe sont plus petits
 - ▶ énergie : DSP virgule fixe consomment moins

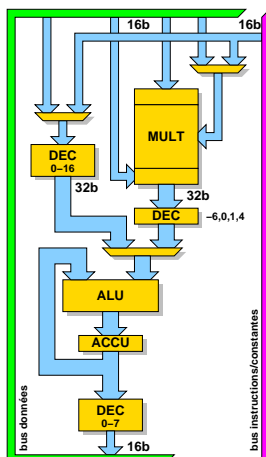
État actuel : 95% des DSP en virgule fixe et sur 16 bits.

Problème : le développement à haut niveau des programmes se fait essentiellement avec des flottants (Matlab), comment alors passer automatiquement à un code en virgule fixe ?

Unités de calcul typiques des DSP

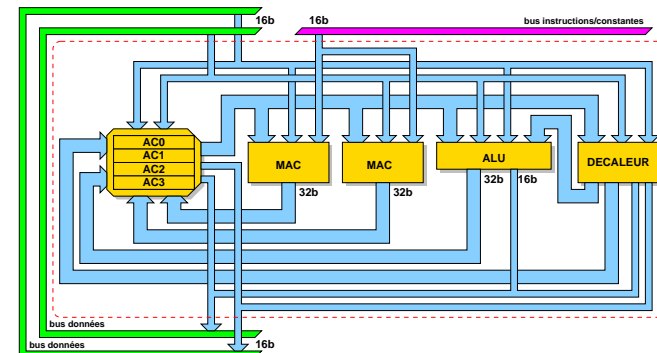
- arithmétique et logique **ALU**
 - ▶ opérations classiques
 - ▶ saturation, min, max
- multiplication–addition avec ou sans accumulation interne **MAC**
 - ▶ avec multiplieur–additionneur dédié
 - ▶ avec multiplieur et additionneur séparés (+ décaleurs)
- décalage
- spécifiques :
 - ▶ manipulation de bits
 - ▶ comptage de bits
 - ▶ support matériel pour calcul partiel de \div et $\sqrt{\quad}$
 - ▶ conversion entiers \Leftrightarrow flottants
 - ▶ :

Unité de calcul du TMS320C24x



- 1 accumulateur 33 bits
- 1 multiplieur $16 \times 16 \rightarrow 32$
- 1 ALU 32 bits
- différents décaleurs

Unité de calcul du TMS320C55x



- 4 accumulateurs 40 bits
- 2 MAC $17 \times 17 \pm 40 \rightarrow 40$ (avec ou sans saturation à 32/40 bits)
- 1 ALU 40 bits (SIMD 2×16 bits)
- 1 décaleur ($\ll 31$ à $\gg 32$)

Unités de manipulation de bits *BMU*

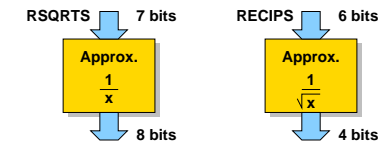
Ces unités sont spécialisées pour faire des :

- décalages / rotations
- insertions / extractions de portions de mots
- permutations
- tests à 0 ou 1 de certains bits
- mises à 0 ou 1 de certains bits
- inversions de certains bits
- inversions de l'ordre des bits
- comptages des bits à 1 ou 0 d'un mot
- comptages des bits à 1 ou 0 des poids forts d'un mot
- opérations logiques avec masques

Support matériel pour des fonctions plus complexes

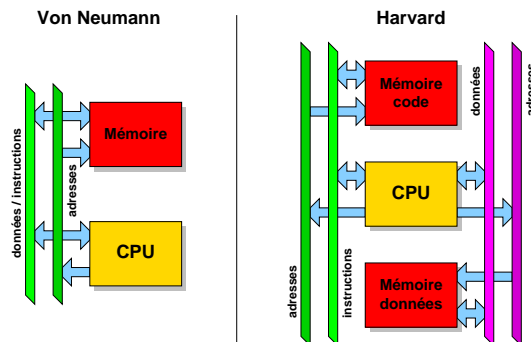
Dans certains DSP, on trouve des petites tables (style ROM) pour stocker des approximations initiales pour effectuer des divisions, des racines carrées (et racines carrées inverses) à l'aide de la méthode de Newton.

Exemple : instructions RECIPS et RSQRTS dans famille ADSP-21000.



On trouve aussi, dans certains DSP, un support matériel (et dans le jeu d'instructions) pour exécuter une division avec un algorithme non-restaurant en base 2. Un bit du quotient est produit à chaque cycle (test, addition-soustraction, conversion $\{-1, 1\} \rightarrow \{0, 1\}$).

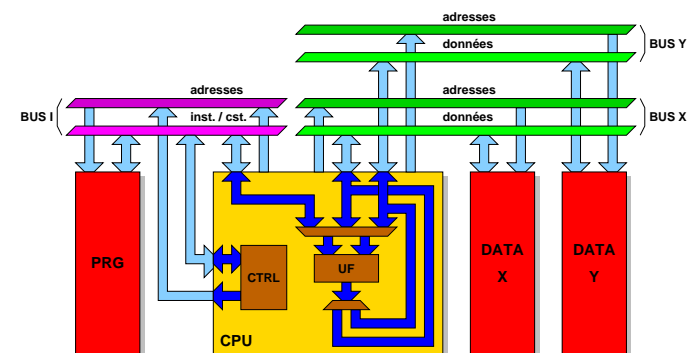
Les 2 grands types de structures mémoire



Principales différences :

- Bande passante (accès instruction et donnée simultanés ou non)
- Nombre de mémoires (coût, surface)
- Contrôle plus ou moins complexe

Structure mémoire de DSP à hautes performances



- 1 mémoire + bus pour les instructions (et les constantes en immédiat)
- 2 mémoires + bus pour les données

Modes d'adressage standards

Codage dans l'instruction de la méthode de calcul de la position d'une donnée (dans un registre ou en mémoire). Exemples de modes d'adressage classiques (DSP et μ -processeurs généralistes) :

adressage	donnée	exemple
registre	dans un registre	MOV R1, R3
immédiat	encodée dans l'instruction	MOV R1, 17
direct	dans mémoire à l'adresse spécifiée dans l'instruction	MOV R1, M(1234)
indirect	dans mémoire à l'adresse spécifiée par un registre	MOV R1, M(R3)
relatif	indirect + décalage spécifié dans l'instruction	MOV R1, M(R3+4)
indexé	indirect dont l'adresse est la somme de 2 registres	MOV R1, M(R3+R8)

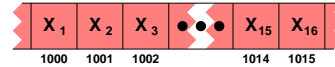
Le support matériel de « bons » modes d'adressage est important pour garantir de bonnes performances (moins d'instructions) mais il coûte en complexité (décodage plus complexe, surface de circuit plus importante).

instruction typique : OP DEST, SRC1, SRC2

Adressages à modification implicite des registres d'adresse

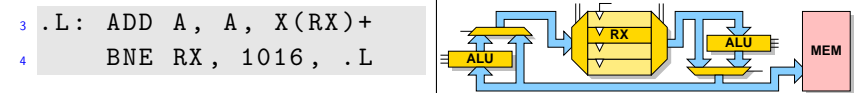
Dans une boucle, on passe du temps à mettre à jour les indices de boucle et les pointeurs (parcours de tableaux) :

<pre> 1 a ← 0 2 for i from 1 to 16 do 3 a ← a + X_i </pre>	<pre> 1 CLR A // accu ← 0 2 LD RX, 1000 3 .L: ADD A, A, X(RX) 4 INC RX 5 BNE RX, 1016, .L </pre>
---	---



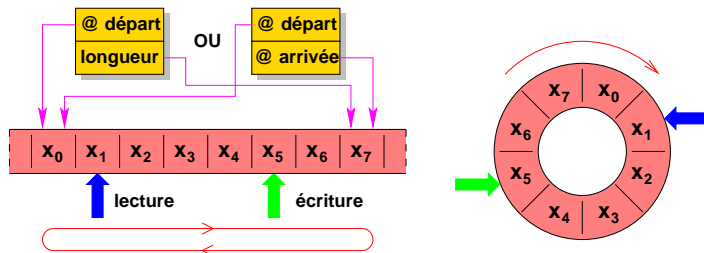
Pour accélérer ces traitements, on utilise les extensions d'adressage suivantes :

- avec pré-incrément ou post-incrément
- avec pré-décrément ou post-décrément



Adressage circulaire ou modulo

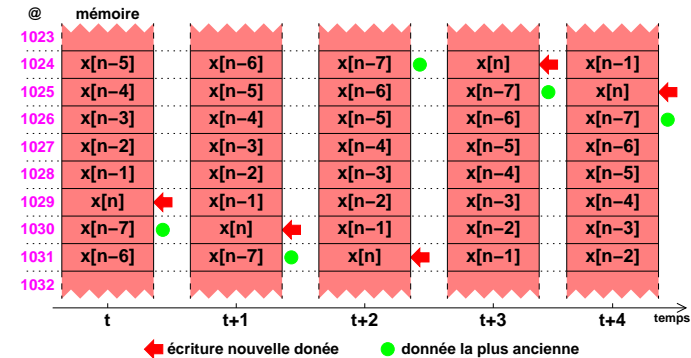
But : ne conserver que les n dernières valeurs des entrées, ou d'un calcul précédent, comme dans une FIFO sans gestion explicite de la mémoire.



- Version (départ, longueur)² : TMS320C3x ou TMS320C4x
- Version (départ, arrivée) : TMS320C5x ou DSP16xx

²Restrictions possible sur l'adresse de départ (bord de zone).

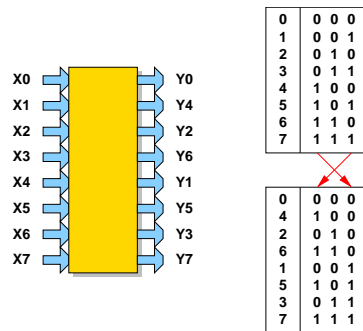
Exemple d'adressage circulaire



ADD R1, R1, M(RX) accède toujours à X[n]
 ADD R1, R1, M(RX-1) accède toujours à X[n-1]
 ⋮

Adressage avec inversion de l'ordre des bits

Pour faire des calculs comme la FFT, on peut avoir besoin de manipuler directement certains bits des adresses.



Dans certains cas, *bit-reverse* signifie prendre le complémentaire bit à bit.

Exemple de taux d'utilisation des modes d'adressage

Cas du TMS320C54x de Texas Instruments sur un ensemble de test de 54 routines de TNS optimisées en assembleur³ :

mode adressage	taux d'utilisation
immédiat	30%
indirect avec post-incrément	19%
indirect	18%
direct	12%
relatif	11%
indirect avec post-décrément	6%
autres	4%

³Source : Hennessy & Patterson, *Computer Architecture : A Quantitative Approach*.

Les caches dans les DSP

L'accès à une mémoire externe, ou des mémoires internes de grande taille, coûte beaucoup en temps et en énergie.

Approches utilisées dans les DSP :

- intégrer des mémoires caches
 - ▶ *repeat buffer*
 - ▶ cache d'instructions
 - ▶ cache de données
- éviter les mouvements de données en mémoire
 - ▶ modes d'adressage spécifiques (circulaire)
 - ▶ optimisations de code particulières

Un petit « cache » d'instructions : le *Repeat Buffer*

Idee : conserver une instruction, ou un petit bloc d'instructions, dans un tampon pour l'exécuter plusieurs fois de suite.

<pre> 1 a ← 0 2 for i from 1 to 64 do 3 a ← a + X_i × Y_i </pre>	<pre> 1 LD R1, 1024 // adr. X_i 2 LD R2, 2048 // adr. Y_i 3 RPTZ A, 63 4 MAC A, X(R1)+, Y(R2)+ </pre>
---	---

Intérêt :

- libère le bus instruction (pour des transferts de constantes)
- délai d'accès plus faible (les instructions sont déjà là)
- diminue la consommation d'énergie (transferts moins gourmands)

Limites : taille très faible (1 à 16 instructions), chargement **manuel** (instruction spéciale), ne peuvent pas contenir des instructions de contrôle.

Caches d'instructions

Les caches instructions des DSP sont un peu différents de ceux que l'on trouve dans les processeurs généralistes :

- petites tailles (32 instructions à 2 Kinstructions)
- découpés en secteurs indépendants (de 1 à 8)
- **configurables** par l'utilisateur :
 - ▶ secteur avec verrou individuel
 - ▶ possibilité d'arrêter la fonction cache
 - ▶ purge possible des secteurs ou du cache complet

Ces caractéristiques particulières sont indispensables pour garantir de bonnes performances pour le temps réel.

Exemples de caractéristiques mémoires et caches

DSP		type	horloge MHz	caches			interne	
famille	proc.			L1P Ko	L1D Ko	L2 Ko	RAM Ko	ROM Ko
TMS320C6xx VLIW 8 × 32b inst.	C62x	fix.	150 – 300	0 ou 4	0 ou 4	0 ou 64	64 – 512	0
	C64x	fix.	300 – 720	16	16	256 – 1024	0	0
	C67x	flo.	100 – 200	16	16	256 – 1024	0	0
TMS320C5xx IPC = 2	C54x	fix.	40 – 160	0	0	0	16 – 512	0 – 256
	C55x	fix.	144 – 200	0, 16, 24	0	0	32 – 320	32 – 64
TMS320C2xx CPI = 1 à 2	C24x	fix.	20 – 40	0	0	0	1 – 5	0 – 32
	C28x	fix.	150	0	0	0	36	128 – 256
DSP56xx	563x	fix.	80 – 240	0 ou 3	0	0	9 – 100	0
	568x	fix.	80 – 120	0	0	0	4 – 128	0 – 2
ADSP21xx SIMD/SISD	211x	flo.	80 – 100	< 1	0	0	128 – 512	0
	210x	flo.	40 – 60	< 1	0	0	64 – 512	0
TigerSHARC	TS101	2	300	0	0	0	768	0

Exécution conditionnelle explicite

L'exécution d'une instruction, ou d'un petit bloc d'instructions, peut être rendue conditionnelle en ajoutant une instruction spéciale juste avant. Dans le cas où la condition est vraie, les instructions sont exécutées, sinon elles sont remplacées en interne par des NOP.

Exemple d'utilisation :

<pre> 1 a ← a + X_i 2 if a < 0 then 3 b ← b + 10 4 end if 5 :</pre>	<pre> 1 ADD A, A, X(R1)+ 2 XC 1, ALT // A < 0 3 ADD B, B, 10 4 :</pre>
---	---

Intérêt :

- moins coûteux que de faire des sauts (en délai et en énergie)
- nombre de cycles parfaitement prévisible

Exécution conditionnelle explicite : conditions types

Les conditions utilisables pour l'exécution conditionnelle sont limitées. En général, on trouve des conditions basées sur les états de drapeaux internes, de certains signaux d'entrées/sorties et sur des tests assez simples sur les accumulateurs.

Exemples de conditions utilisables (partiellement cumulables) :

condition	opérande	condition	opérande
$ACCU = 0$	AEQ	$ACCU$ dépas.	AOV
$ACCU \neq 0$	ANEQ	$ACCU$ non dépas.	ANOV
$ACCU > 0$	AGT	drapeau de test à 1	TC
$ACCU \geq 0$	AGEQ	drapeau de test à 0	NTC
$ACCU < 0$	ALT	signal BIO à 1	BIO
$ACCU \leq 0$	ALEQ	signal BIO à 0	NBIO

DSP : processeurs à pipeline court

processeurs	nombre d'étages					
	total	fetch	decode	memory access	exec	write back
TMS320C3x	4	1	1	1	1	
TMS320C54x	6	2	1	2	1	
TMS320C55x	10 – 11	4	4		2 – 3	
TMS320C62x	7 – 12	4	2	1 – 6		
DSP563x	7	2	1	2	2	
DSP568x	3	1	1	1		
SHARC	3	1	1	1		
TigerSHARC	8	3		5		
ARM7	3	1	1	1		
ARM9	5	1	1	1	1	1
PowerPC 603	5	1	1	2		1
Pentium	5 – 7	1	2	0 – 1	1 – 2	1
Pentium MMX	6 – 8	2	2	0 – 1	1 – 2	1
Pentium II	12 + 2	≈ 5	≈ 2	≈ 2 – 3	≈ 2	≈ 2
Pentium 4	20	20				

Pipeline : exemple du TMS320C62x

Extrait de la documentation du C62x :

- 4 Program Fetch
 - ▶ Generate fetch address
 - ▶ Send address to memory
 - ▶ Wait for data ready
 - ▶ Read opcode
- 2 Decode
 - ▶ Route instructions to functional units (dispatch)
 - ▶ Instruction decoded at functional unit
- 1 – 6 Execute
 - ▶ Multiply (MPY/ SMPY) Delay → 1
 - ▶ Load (LDB/ H/ W) Delay → 4
 - ▶ Branch (B) Delay → 5

Soit un total de 7 à 12 étages.

Quelques DSP actuels

Fabricant	DSP	fréquence MHz	type	taille des mots	
				données	instr.
Analog Devices	ADSP-21xx	80	fixe	16	24
	ADSP-2116x	100	flot.	32/40	48
	ADSP-2153x	300	fixe	16	16/32
Texas Instruments	TMS320C24x	40	fixe	16	16/32
	TMS320C28x	150	fixe	32	16/32
	TMS320C54x	160	fixe	16	16
	TMS320C55x	200	fixe	16	8 – 48
	TMS320C62x	300	fixe	16	32
	TMS320C64x	600	fixe	8/16	32
	TMS320C67x	167	flot.	32	32
Motorola	DSP563xx	240	fixe	24	24
	DSP568xx	80	fixe	16	16
	DSP5685x	120	fixe	16	16

Source : www.bdti.com

DSP versus processeurs généralistes : consommation d'énergie

Fabricant	DSP	fréquence MHz	puissance W
Texas Instruments	TMS320C54x	144 – 200	0.065 – 0.160
	TMS320C55x	50 – 160	0.040 – 0.090
	TMS320C62x	150 – 30	0.9 – 2.1
	TMS320C64x	300 – 720	0.25 – 1.06
	TMS320C67x	100 – 225	0.5 – 1.4
Intel	Pentium M	266	5.3
	Pentium II M	400	8
	Pentium IV	2 800	68
Motorola/IBM	604e	350	8
	G3	466	5.5
	G4+	533	14
SPARC	Ultra I	200	30
	Ultra II	450	25
	Ultra III	1 200	53

Évolution "hautes performances" des DSP

Comment accélérer les programmes ?

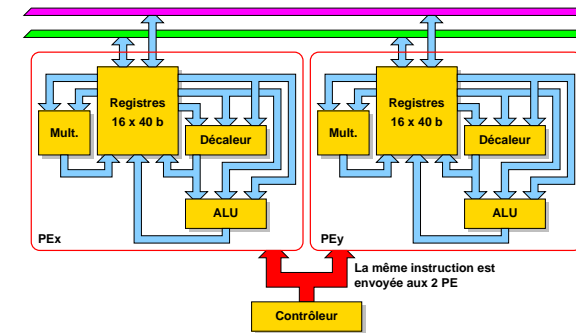
⇒ PARALLÉLISME

Solutions courantes :

- SIMD : DSP16xxx, ADSP-2116x
- Superscalaire : LSI401Z
- VLIW : TMS320 C6x
- approches combinées :
 - ▶ VLIW + SIMD : TigerSHARC

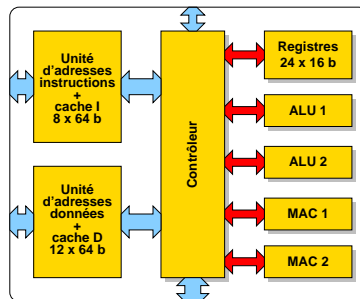
DSP SIMD : ADSP-21160 SHARC

SIMD = *Single Instruction, Multiple Data*



- Décodage des instructions simple
- Demande un parallélisme bien spécifique

DSP superscalaire : LSI401Z

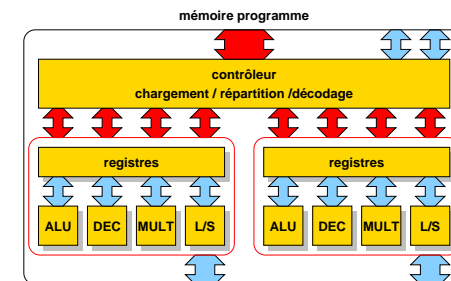


- issu du cœur synthétisable ZSP400 (sous licence)
- processeur 16 bits super-scalaire 4 voies
- exécution dans l'ordre (pipeline 5 étages)
- instructions 16 bits
- support données 16/32 bits
- charge 4 instructions par cycle
- exécute jusqu'à 4 instructions par cycle
- 2 ALU 16 bits
- 2 MAC $16 \times 16 + 32 \rightarrow 32$
- 16 registres généraux 16 bits (ou 8 de 32 bits)
- 4 niveaux de boucles matérielles (+bit reverse et adressage circulaire)

- Le processeur se charge de tout : détection, ordonnancement. . .
- Circuit de contrôle complexe (vitesse ↓, surface et conso. ↑)

DSP VLIW : TMS320C62x

VLIW = *Very Large Instruction Word*



- VLIW 8 voies (instructions 32 bits, soit bus-I de 256 bits)
- Parallélisme dans les instructions (compilateur ou assembleur)
- Contrôle plus simple que superscalaire

La R&D industrielle autour des DSP en France

Pour en savoir plus . . .



- Texas Instruments :
 - ▶ Villeneuve-Loubet → centre R&D
- Motorola :
 - ▶ Saclay → centre R&D
 - ▶ Toulouse → centre R&D + Fab.
 - ▶ Crolles 2 → centre R&D + Fab.
- ST Microelectronics :
 - ▶ Crolles → centre R&D + Fab.
 - ▶ Rousset → centre R&D + Fab.
- ARM :
 - ▶ Sophia Antipolis → centre R&D
- Atmel :
 - ▶ Rousset → centre R&D + Fab.
- Infineon :
 - ▶ Echirrolles → centre R&D
- ...

Digital Signal Processing

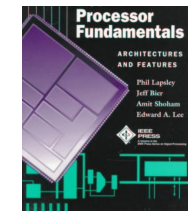
Principles, Algorithms, and Applications

John G. Proakis et Dimitris G. Manolakis

Troisième édition, 1996

Prentice-Hall

ISBN : 0-13-373762-4



DSP Processor Fundamentals

Architectures and Features

Phil Lapsley, Jeff Bier,
Amit Shoham et Edward A. Lee

1997, IEEE Press

ISBN : 0-7803-3405-1

Fin

Questions ?

Pour me contacter :

- arnaud.tisserand@ens-lyon.fr
- <http://www.ens-lyon.fr/~atissera/> (va changer)
- Laboratoire LIP. ENS Lyon. 46 allée d'Italie. F-69364 Lyon cedex 07.

Merci.