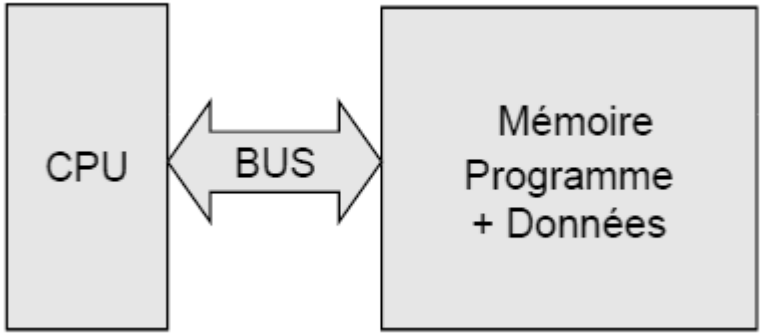
CHAPITRE 3

Architecture des DSP TMS320C6x

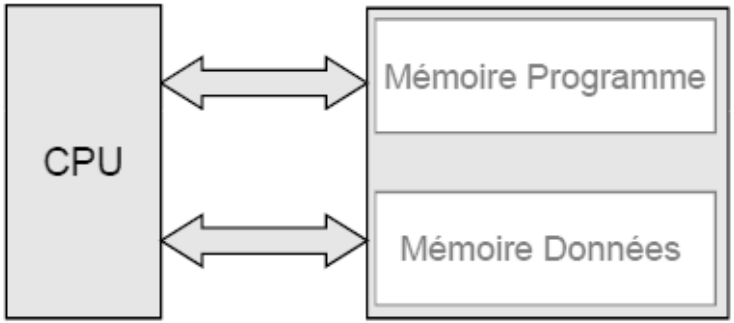
Introduction

- **Architecture Von Neuman**
  - Un seul BUS mémoire partagé
  - Programme non sécurisé



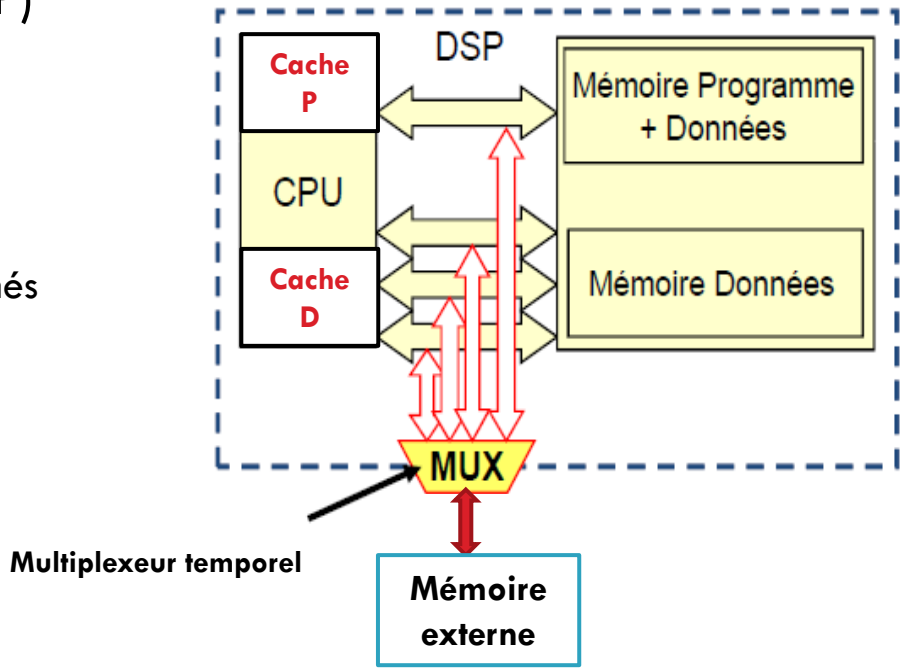
Introduction

- **Architecture Harvard**
  - Séparation programme et données
  - Accès mémoire non partagé



Introduction

- **Architecture Harvard modifiée** (DSP)
  - Mémoire. Programme + données
  - Mémoire données multi-accès
    - N BUS de données
    - Accès mémoire données simultanés
  - Mémoire externe
    - Extension mémoire par multiplexage temporel des BUS
  - Mémoire Cache
    - Cache données
    - Cache programme



Introduction

- **Besoin de plus de parallélisme:**

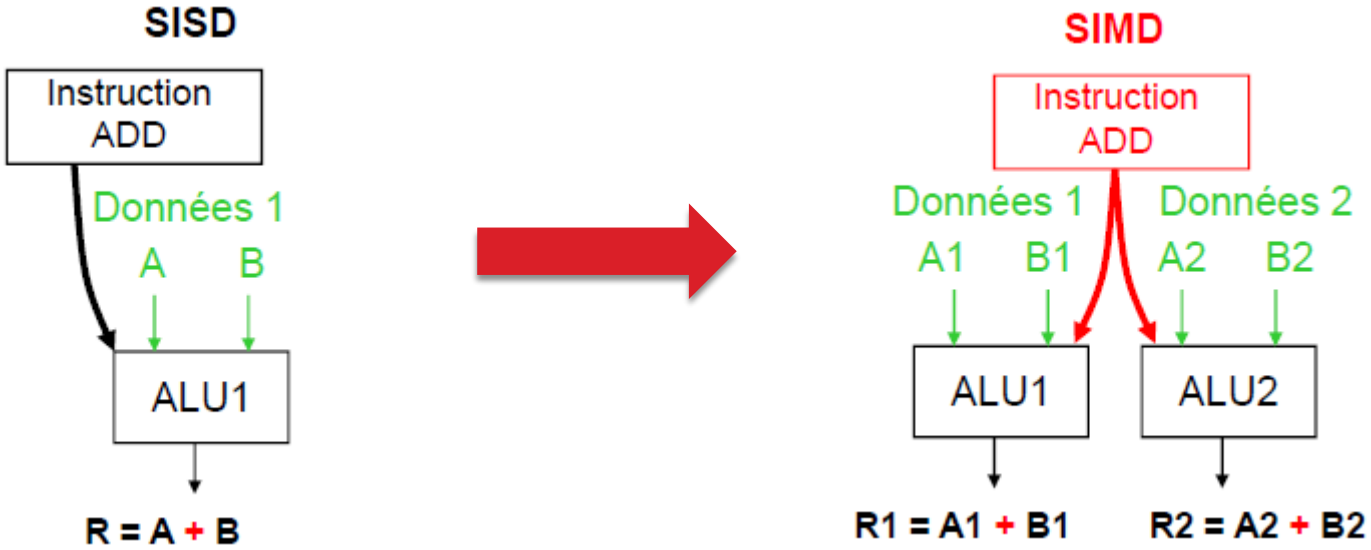
  - Limitations des architectures classiques:
    - Jeu d'instruction complexe: CISC (Complex Instruction Set Computer)
    - Parallélisme limité

  - Approche SIMD:
    - SIMD: Single Instruction on Multiple Data
    - La même instruction est appliquée simultanément sur plusieurs données

Introduction

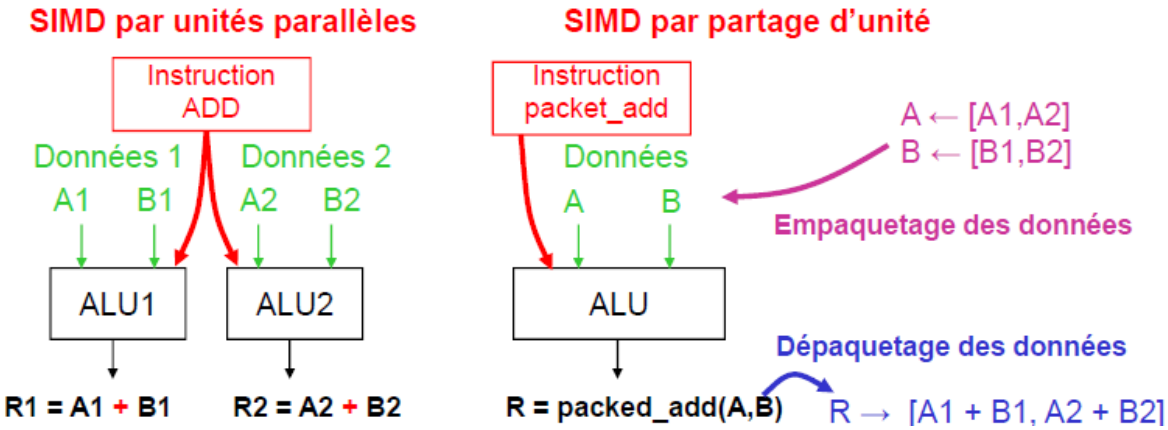
- **Vers plus de parallélisme**
  - **SISD: Single Instruction / Single Data**

SIMD: Single Instruction / Multiple Data



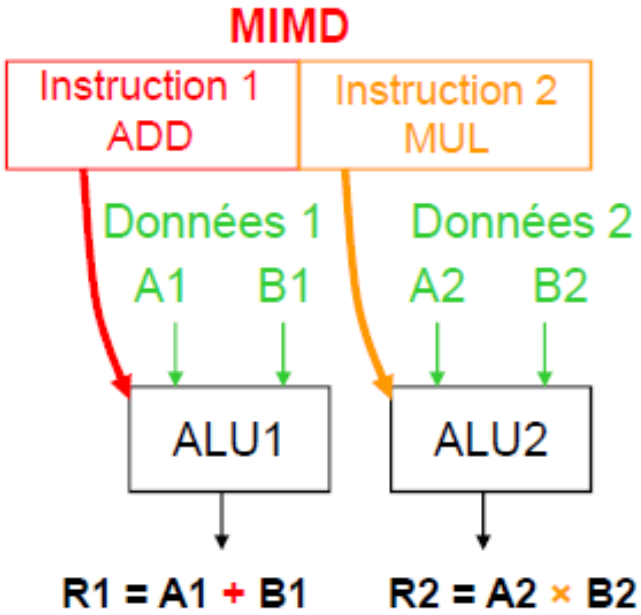
Introduction

- **Approche SIMD:**
  - Il existe deux types d'approches SIMD:
    - SIMD par unités parallèles
    - SIMD par partage d'unités



Introduction

- **Approche MIMD**
  - Multiple Instruction / Multiple Data
  - Introduction du mot d'instruction très large **VLIW**
  - VLIW: Very Long Instruction Word
  - Avantages:
    - **Parallélisme ++**
    - Unités indépendantes
    - Chaque instruction peut pointer vers une unité différente
  - Inconvénients:
    - **Bande passante mémoire importante**
    - Code machine plus grand



Architecture des DSP C6x

Introduction

- Il existe différentes familles de DSP TMS dont la famille DSP TI TMS320C6x ou C6x
  - C6x = C62x / C64x/ C67x ….
- Possèdent toutes un jeu d'instruction réduit RISC (Reduced Instruction Set Computer)
- Domaines d'applications des DSP C6x (hautes performances) :
  - 3G
  - Modem xDSL
  - Traitement numérique des images

Architecture du DSP C6x

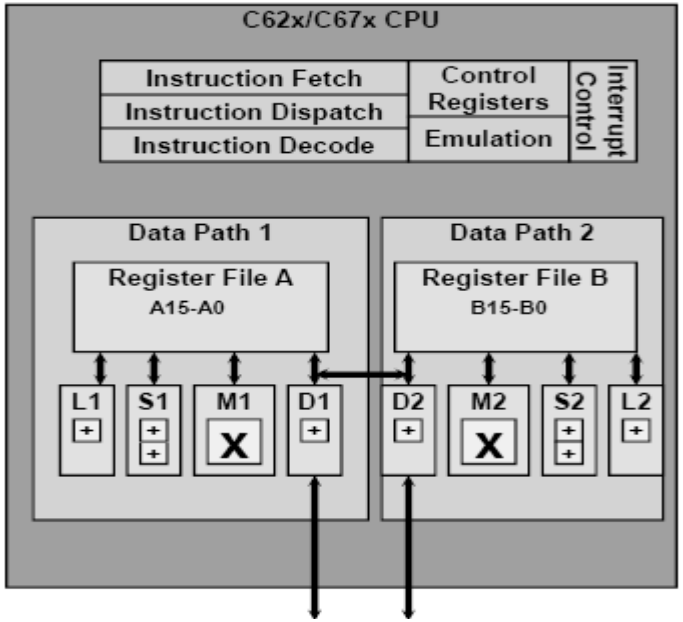
Famille C6000 ou C6x

- Les DSP C6x ont tous une architecture de base identique: **VelociTI**
- Ils ont un jeu d'instruction de base identique où certains sont compatibles en programmation
  - **C62x ← → C67x**
- **C64x** ont des bus de données **64bits**
- **C674x** est une fusion entre le **C67x** et **le C64x**

DSP	Arithmétique
C62x C64x C64x+	**Fixe**
C67x C67x+	**Flottante**
C674x C66x	**Fixe/Flottante**

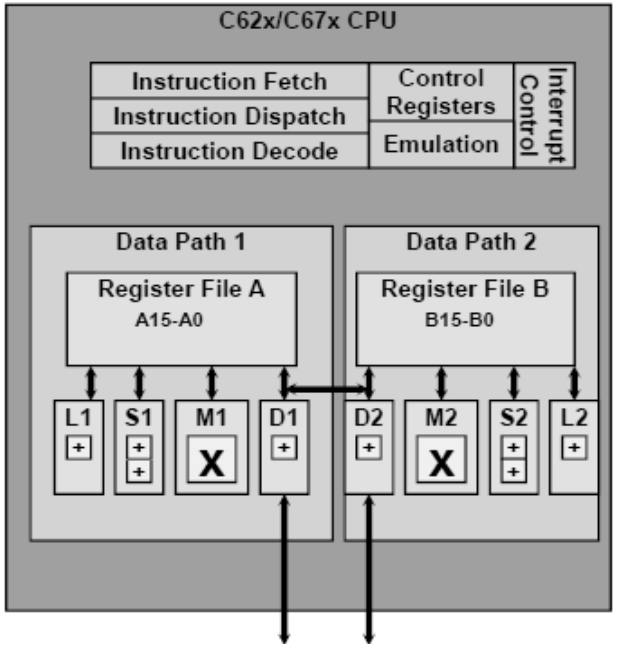
Architecture des DSP C6x

- **Architecture VLIW avancée (VelociTI):**
  - 2 chemins de données:
    - Data Path 1 (DP1)
    - Data Path 2 (DP2)
  - Chaque chemin comprend 4 unités fonctionnelles
    - L : ALU
    - S : décalage et ALU
    - M: multiplication
    - D: mouvement données de/vers la mémoire et ALU



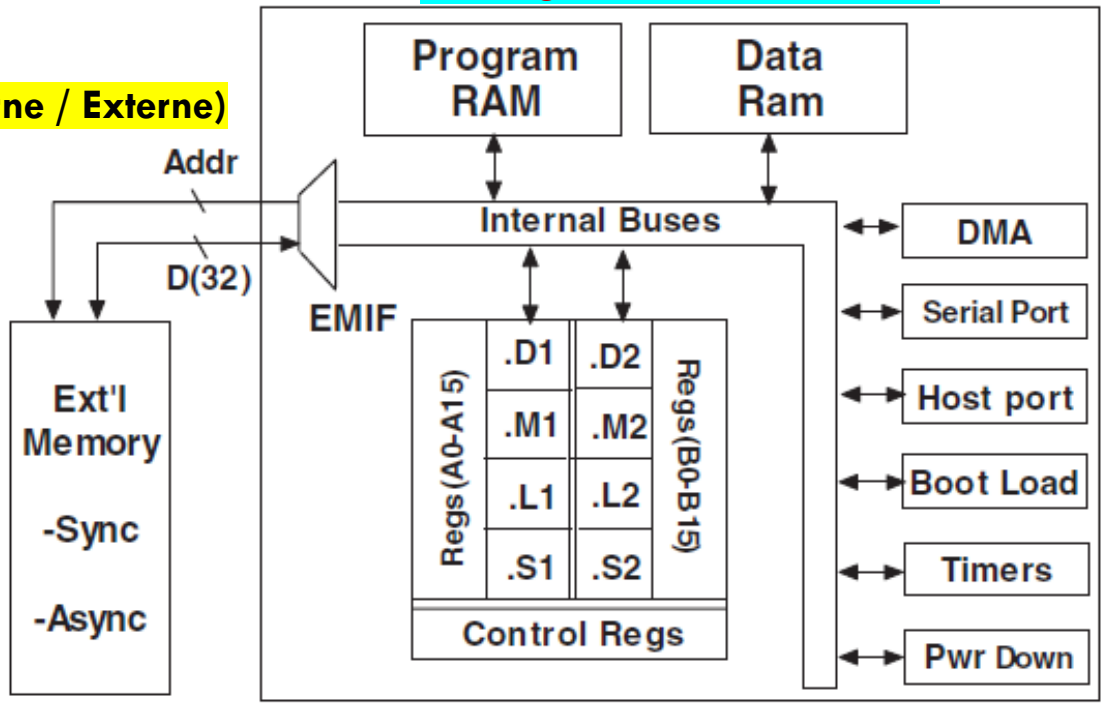
Architecture des DSP C6x

- **Architecture VLIW avancée (VelociTI):**
  - **8 instructions / cycle**
    - Un mot d'instruction (Word) = 8 x 32bits = 256bits
  - **MAC = LOAD (D) + MUL (M) + ADD (L)**
  - **2 blocs de registres A et B**
    - Chaque bloc comprend 16 registres 32bits
  - **Plusieurs registres de contrôle**



Architecture des DSP C6x

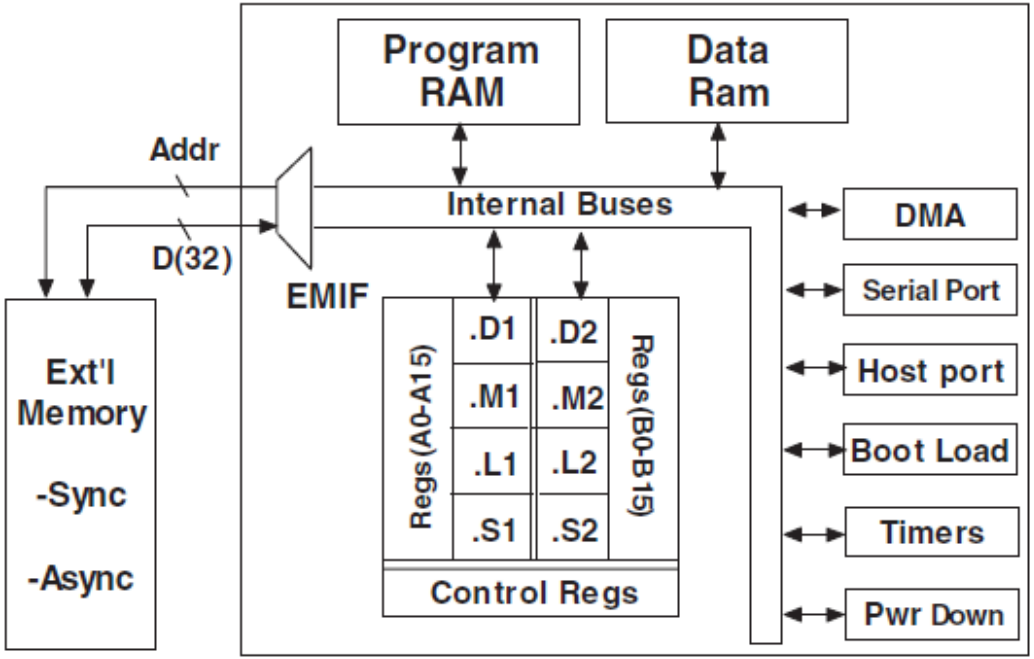
- **Diagramme générique du C6x**
  - **Bus Programme & Bus Données**
  - **Bus mémoire (Interne / Externe)**



- **Bus DMA**
- **Bus Périphériques**

Architecture du DSP C6x

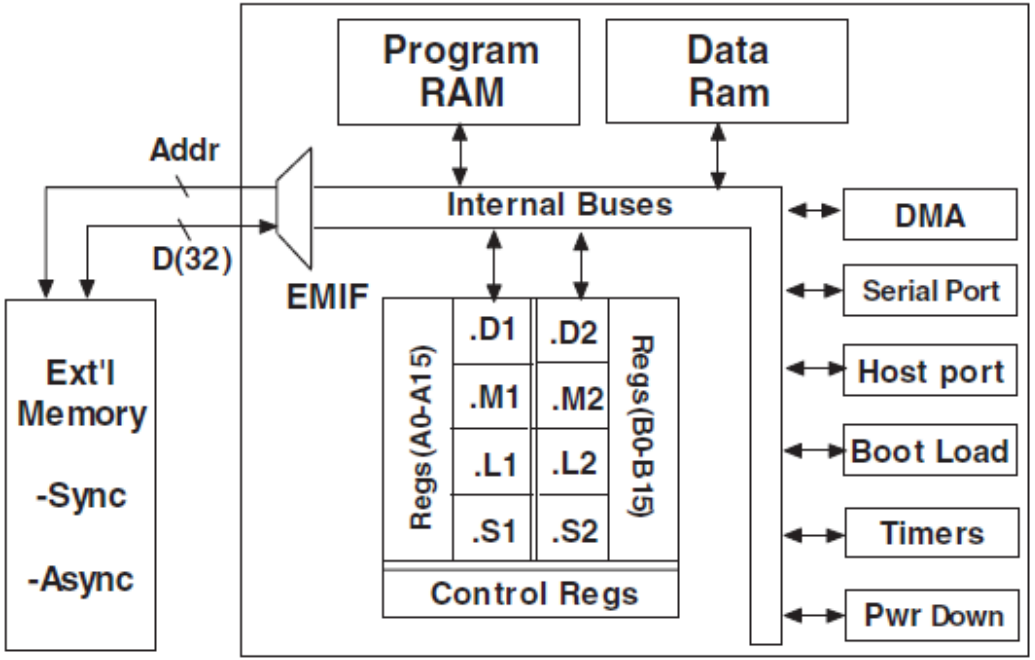
- **Périphériques communs**
  - **EMIF (External Memory Interface)**
    - Interface d'accès mémoire externe
  - **DMA (Direct Memory Access)**
    - Mouvement des données mémoire sans intervention du CPU
  - **Enhanced DMA (DMA améliorée)**
    - Identique au DMA avec plus de canaux mémoires programmables



Architecture du DSP C6x

- **Périphériques communs**

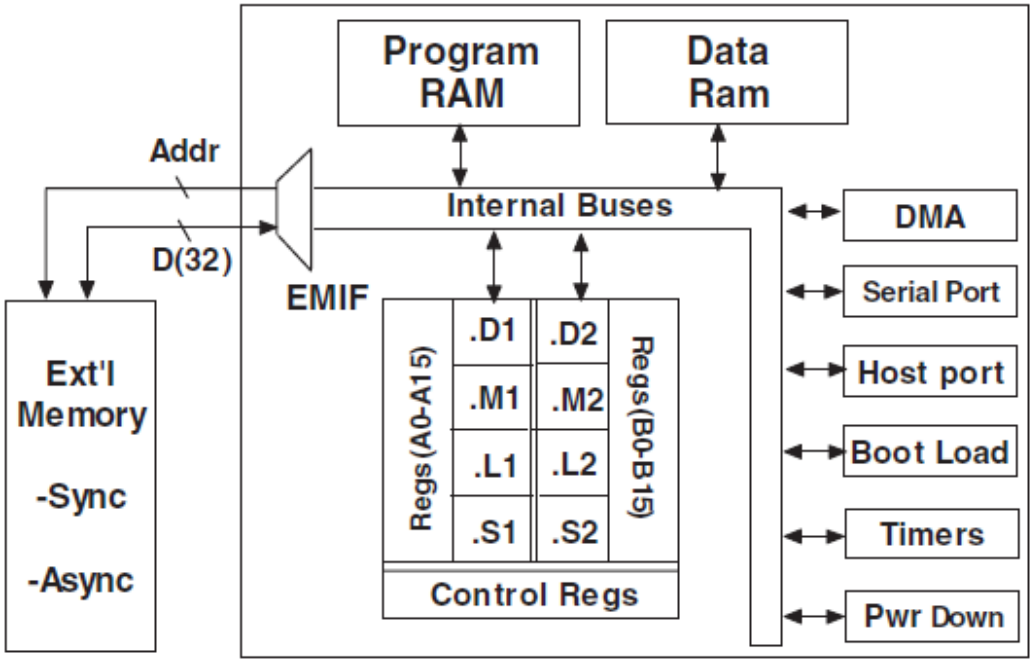
  - **Timers 32bits**
    - Compteur / Timer / Interruption
  - **Power Down Modes**
    - Gestion de la consommation
  - **Boot Load:**
    - Gestion du démarrage DSP via HP ou un block mémoire externe



Architecture du DSP C6x

- **Périphériques communs**

  - **Port série synchrone**
    - Multichannel Buffered Serial Port (**McBSP**)
  - **Port parallèle**
    - Host Port Interface (HPI)



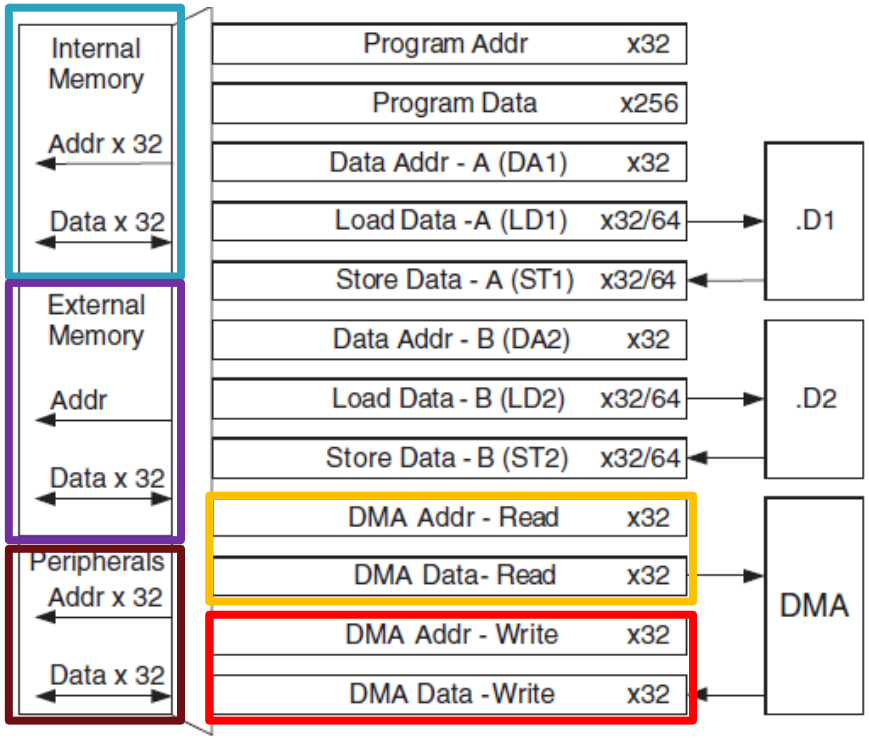
Architecture des DSP C6x

- **BUS communs**

Mémoire interne 32bits

Mémoire externe 32bits

Périphériques 32bits



BUS Lecture DMA

- **Addr:** 32bits
- **Données:** 32bits

BUS Ecriture DMA

- **Addr:** 32bits
- **Données:** 32bits

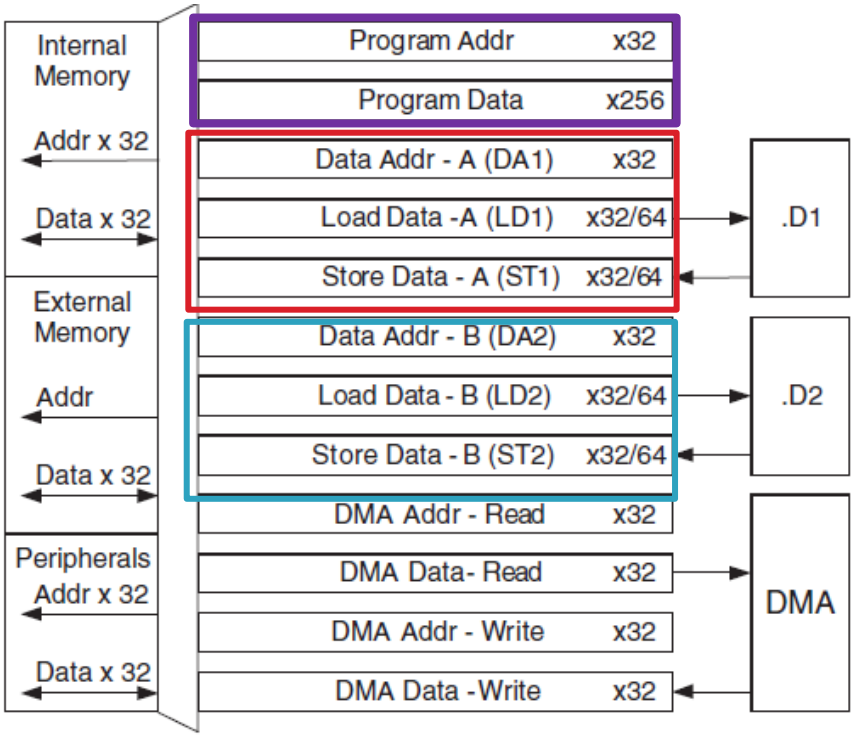
Architecture du DSP C6x

- **BUS communs**

Programme

- **Addr:** 32bits
- **Données: 256 bits**

(8 instructions x 32bits)



Données 1:

- **Addr DA1:** 32bits
- **Load LD1:** 32 bits
- **Store ST1:** 32 bits

Données 2:

- **Addr DA2:** 32bits
- **Load LD2:** 32 bits (64bits C64x)
- **Store ST2:** 32 bits (64bits C67x)

Architecture du DSP C6x

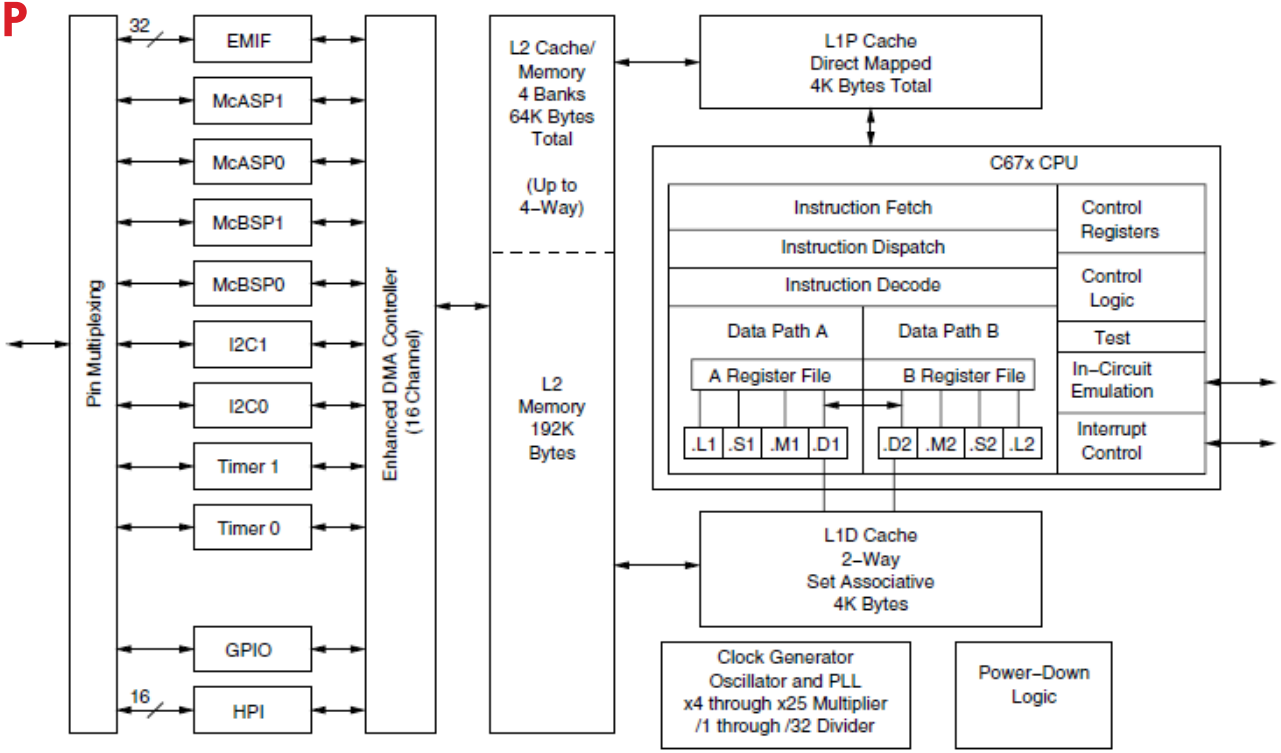
- **Architecture interne DSP C6713**

2 ALU à virgule fixe
- D1 , D2

4 ALU à virgule fixe/flot.
- L1, L2 / S1, S2

2 MUL à virgule fixe/flot.
- M1, M2

Compatible IEEE 754



Architecture du DSP C6x

- **Architecture interne DSP C6713**

Fréquence Horloge:

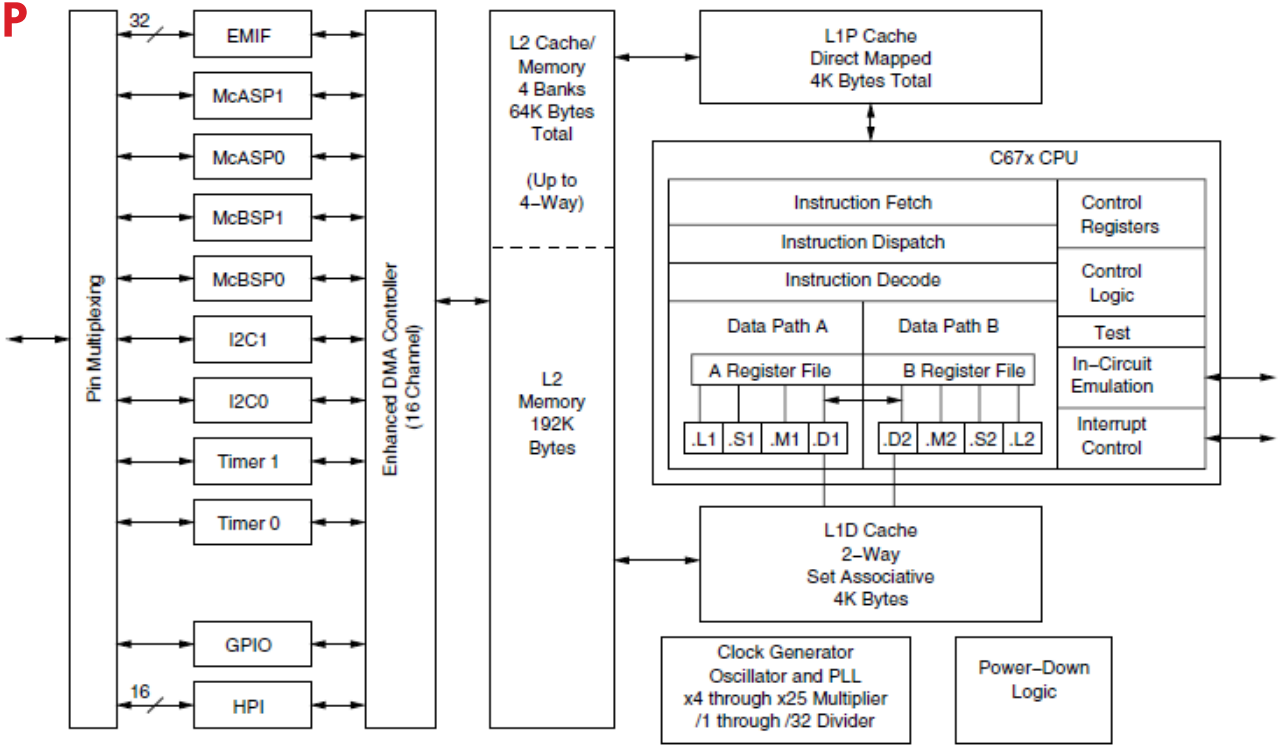
- **300 Mhz = 2400 MIPS**
- **1800 MFLOPS**

Mémoire interne cache

- **264Kb**

Mémoire externe

- **1Gb**



Mémoire du DSP C6x

- Les DSP C67x ont une architecture mémoire cache à 2 niveaux

- Mémoires cache niveau 1 (L1 ou Level 1)
  - L1P cache programme 4 KB
  - L1D cache données 4 KB

Mode cache non actif par défaut

- Mémoires cache niveau 2 (L2 ou Level 2)
  - L2 mémoire partagé programme/données de 256 KB composée de:
    - **64K mémoire cache (divisée en 4 blocks de 16K)**
    - **192Kb mémoire SRAM (mémoire RAM statique)**

Mémoire du DSP C6x

Espace mémoire

- Un espace adressable de $2^{32bits} = 4Gb$: 0x0000 0000 → 0xFFFF FFFF
- Espace mémoire **interne allant de 0x0000 0000 → 0x0003 FFFF (256Kb)**
  - Espace mémoire cache interne L2
- Espace mémoire **mixte allant de 0x0004 0000 → 0x7FFF FFFF**
  - Espace registres périphériques + plusieurs espaces réservés
  - Registres Timer, ports McBSP, HPI, GPIO …etc
    - GPIO ou General Purpose I/O : E/S à usage général

Mémoire du DSP C6x

Espace mémoire

- Espace mémoire **externe: 0x8000 0000 → 0xBFFF FFFF (1Gb)**
  - Espace mémoire externe (EMIF) divisé sur 4 blocks de 256Mb
  - Blocks: CE0, CE1, CE2 et CE3
- Espace mémoire **réservé: 0xC000 0000 → 0xFFFF FFFF (1Gb)**
- Exemple:
  - **Cartographie mémoire du DSP TMS320C6713 (source: datasheet TI)**

Cartographie mémoire C6713

MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	192K	0000 0000 – 0002 FFFF
Internal RAM/Cache	64K	0003 0000 – 0003 FFFF
Reserved --fonct. interne	24M – 256K	0004 0000 – 017F FFFF
External Memory Interface (EMIF) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	128K	0184 0000 – 0185 FFFF
Reserved	128K	0186 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	512	019C 0000 – 019C 01FF
Device Configuration Registers	4	019C 0200 – 019C 0203
Reserved	256K – 516	019C 0204 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
Reserved	768K	01A4 0000 – 01AF FFFF
GPIO Registers	16K	01B0 0000 – 01B0 3FFF

Interne

Périphériques + Réservés

Périphériques + Réservés	Reserved	240K	01B0 4000 – 01B3 FFFF
	I2C0 Registers	16K	01B4 0000 – 01B4 3FFF
	I2C1 Registers	16K	01B4 4000 – 01B4 7FFF
	Reserved	16K	01B4 8000 – 01B4 BFFF
	McASP0 Registers	16K	01B4 C000 – 01B4 FFFF
	McASP1 Registers	16K	01B5 0000 – 01B5 3FFF
	Reserved	160K	01B5 4000 – 01B7 BFFF
	PLL Registers	8K	01B7 C000 – 01B7 DFFF
	Reserved	264K	01B7 E000 – 01BB FFFF
	Emulation Registers	256K	01BC 0000 – 01BF FFFF
	Reserved	4M	01C0 0000 – 01FF FFFF
	QDMA Registers	52	0200 0000 – 0200 0033
	Reserved	16M – 52	0200 0034 – 02FF FFFF
	Reserved	720M	0300 0000 – 2FFF FFFF
	McBSP0 Data Port	64M	3000 0000 – 33FF FFFF
	McBSP1 Data Port	64M	3400 0000 – 37FF FFFF
	Reserved	64M	3800 0000 – 3BFF FFFF
	McASP0 Data Port	1M	3C00 0000 – 3C0F FFFF
	McASP1 Data Port	1M	3C10 0000 – 3C1F FFFF
	Reserved	1G + 62M	3C20 0000 – 7FFF FFFF
Externe	EMIF CE0†	256M	8000 0000 – 8FFF FFFF
	EMIF CE1†	256M	9000 0000 – 9FFF FFFF
	EMIF CE2†	256M	A000 0000 – AFFF FFFF
	EMIF CE3†	256M	B000 0000 – BFFF FFFF
Réservé	Reserved	1G	C000 0000 – FFFF FFFF

Unités fonctionnelles

- Unités L :
  - .L1
  - .L2

Virgule fixe	Virgule flottante
Arithmétique Comparaison 32/40bits	Arithmétique
Logique 32 bits	Conversion INT → SP INT → DP DP → SP

Unités fonctionnelles

- Unités S :
  - .S1
  - .S2

- Mouvement de/vers les registres contrôle possible seulement sur .S2

Virgule fixe	Virgule flottante
Décalages 32/40 bits	Valeur Réciproque / Absolue / Racine carrée
Branchements / Génération de Constantes	Conversion SP → DP
Arithmétique 32bits	Comparaison/ Addition/Soustraction au format SP et DP

Unités fonctionnelles

- Unités M :
  - .M1
  - .M2

Virgule fixe	Virgule flottante
Multiplication 16x16bits 32x32bits	Multiplication au format SP et DP

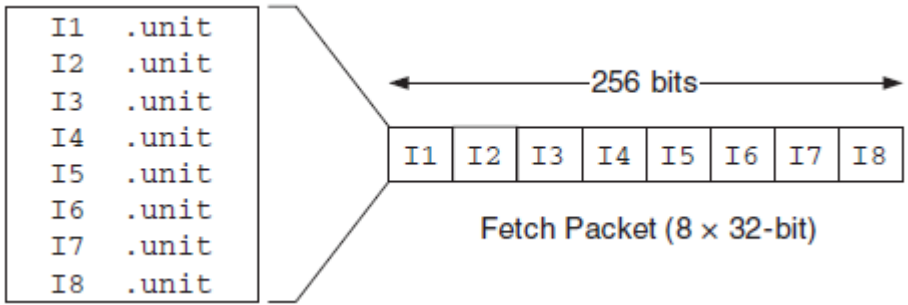
Unités fonctionnelles

- Unités D :
  - .D1
  - .D2

Virgule fixe	Virgule flottante
Génération d'adresses 32bits (linéaire ou circulaire)	Chargement/Stockage données avec un offset constant de 5bits
Chargement / Stockage données avec un offset constant de 5bits (15bits sur l'unité .D2)	
Arithmétique 32bits	

Paquets Fetch / Exécution

- **Paquet Fetch FP / Paquet Exécution EP:**
  - Fetch Paquet (FP) en anglais
  - Un FP a une taille de 256bits
  - Un FP contient 8 instructions de 32bits



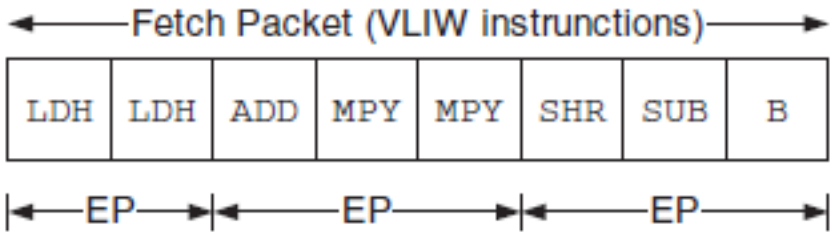
Paquets Fetch / Exécution

- **Paquet Fetch FP / Paquet Exécution EP:**
  - Les instructions du FP peuvent être exécutées en parallèle (||) ou en série
  - Les instructions exécutées en parallèle (||) constitue un paquet d'exécution appelé EP (Execution Paquet en anglais)
  - Un FP peut avoir:
    - 1 EP au minimum
    - 8 EP au maximum
  - Chaque instruction dans un EP **doit** utiliser une unité fonctionnelle différente

Paquets Fetch / Exécution

- **Exemple 1:**
  - Ce **FP** contient **3 EP**
  - Les instructions dans un **EP** utilisent des unités fonctionnelles non identiques

```
   LDH   .D1
|| LDH   .D2
   ADD   .L1
|| MPY   .M1
|| MPY   .M2
   SHR   .S1
|| SUB   .L2
|| B     .S2
```



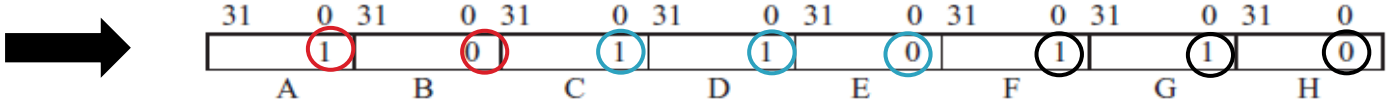
Paquets Fetch / Exécution

- **Paquet Fetch FP / Paquet Exécution EP:**
  - Le bit LSB d'une instruction appelé BIT **P** (||) permet de trouver le nombre de EP sur le FP
    - Si LSB = 1 : L'instruction suivante appartient au même EP
    - Si LSB = 0 : L'instruction suivante appartient à un autre EP
  - Le BIT **P** de la dernière instruction du FP est toujours 0 car un EP ne peut pas dépasser 8 instructions

```
    Instruction A
||  Instruction B

    Instruction C
||  Instruction D
||  Instruction E

    Instruction F
||  Instruction G
||  Instruction H
```



Architecture PIPELINE

- **Définition générale du Pipeline :**

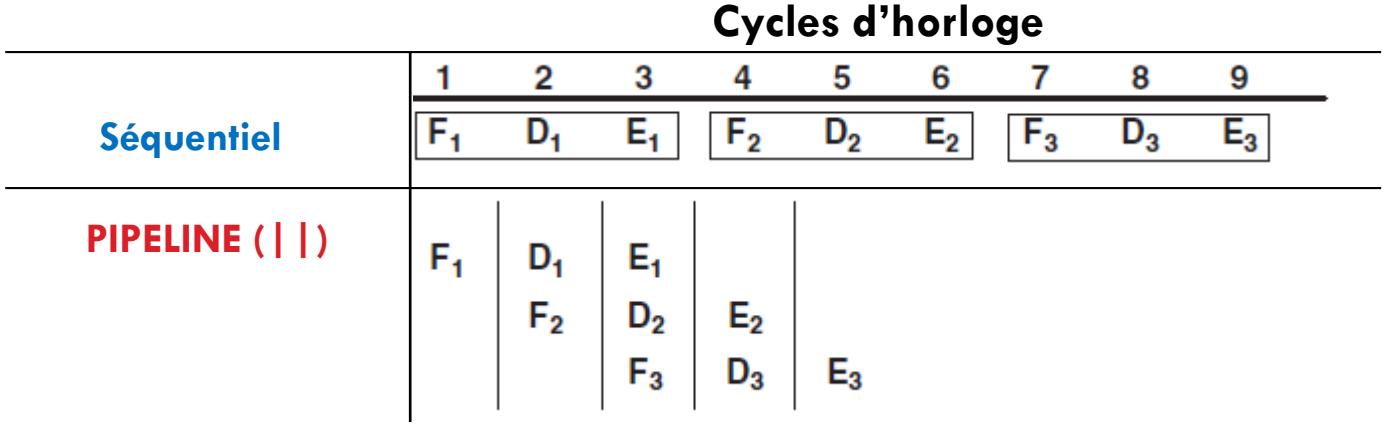
  - Dans un CPU, l'exécution d'une instruction passe par 3 étapes de base :
    - F: Fetch (ou recherche et chargement en anglais)
    - D: Décodage
    - E: Exécution

  - En fonction du CPU, ces étapes peuvent être effectuées :
    - En séquence
    - En parallèle (||)

  - L'exécution des instructions en || correspond à une architecture appelée architecture **PIPELINE**

Architecture pipeline

- **Exemple:** Exécution de 3 instructions en séquence ensuite en PIPELINE (| |)
  - Nombre de cycles:
    - Séquence = 9
    - PIPELINE = 5 (un gain de 4 cycles)



Architecture pipeline

- **Pipeline hardware du DSP C6x**

  - Le DSP C6x possède une architecture pipeline optimisée où chaque étage pipeline est décomposée en plusieurs « sous-étages » ou phases

  - **Objectif:**
    - Augmentation du parallélisme par effet pipeline

Architecture pipeline

Phases de l'étage Fetch :

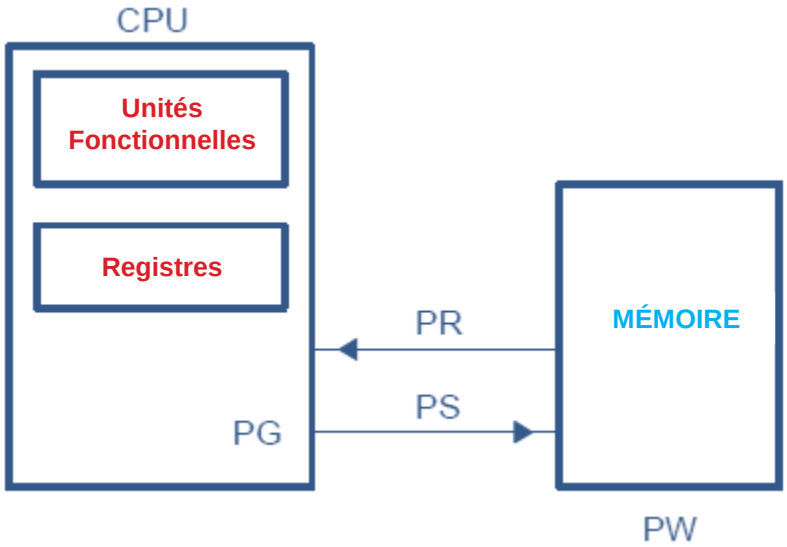
- Inclut 4 phases où chaque phase requiert un cycle

A. **Phase PG: Program address Generate**
   - **Génération** sur le CPU de l'adresse du **paquet FP**

B. **Phase PS: Program address Send**
   - **Envoi** de l'adresse du **paquet FP** vers la mémoire

C. **Phase PW: Program address ready Wait**
   - **Attente** de la fin de la **lecture** mémoire

D. **Phase PR: Program fetch packet Receive**
   - **Réception** du **paquet FP** au niveau du CPU

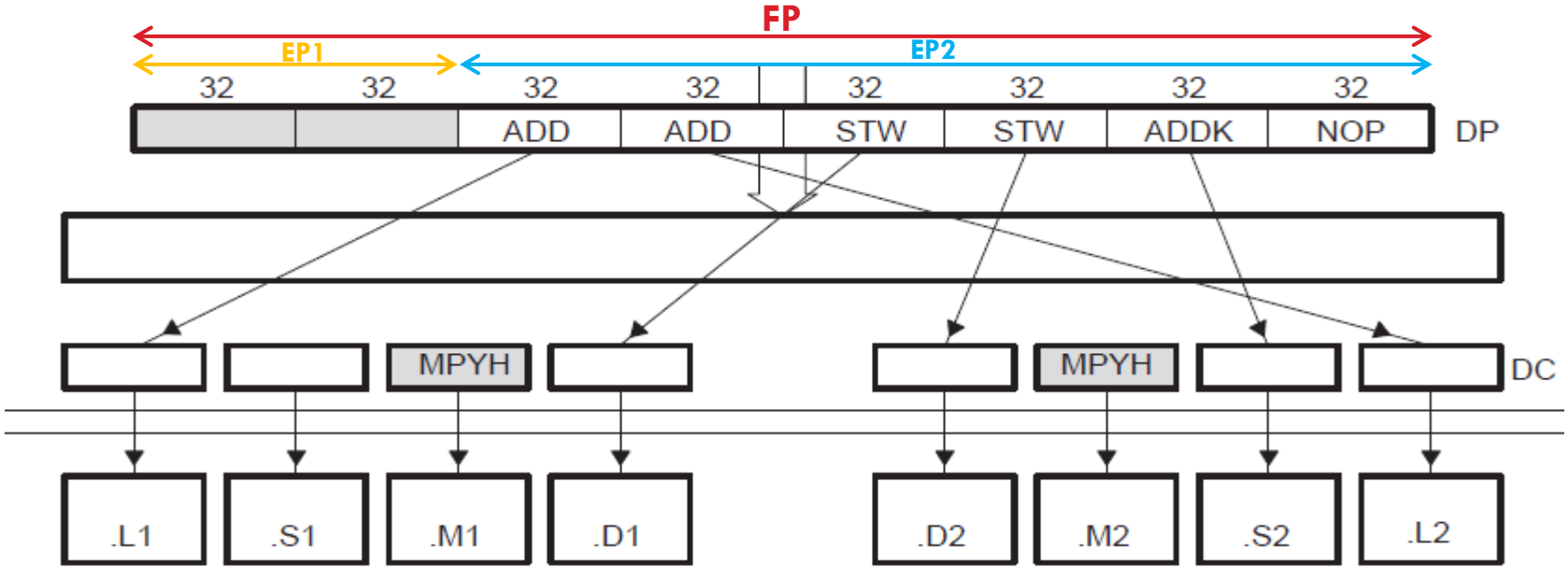


Architecture pipeline

- **Phases de l'étage Décodage**
  - Inclut 2 phases et nécessite un cycle / phase:
    - Phase DP: Dispatch
      - Envoi des instructions de chaque paquet EP dans le paquet FP vers leurs unités
    - Phase DC: Decode
      - Décodage des instructions (registres sources / destinations …) avant la phase de l'exécution

Architecture pipeline

- **Phases de l'étage Décodage**
  - Exemple: Un **FP** avec **2 EP**



Architecture pipeline

Phases de l'étage Exécution:

- La majorité des instructions C6x nécessitent une seule phase d'exécution (E1) pour compléter leur exécution
- Certaines instructions présentent un délai d'exécution d'où l'existence de nombreuses phases d'exécution
- Le nombre de phases : **#Phases = Délai + 1**
  - Pipeline C6x en virgule fixe: #Phases maximum = 6 (E1 … E6)
  - Pipeline C6x en virgule flottante: #Phases maximum = 10 (E1 … E10)
- Le temps d'occupation de l'unité fonctionnelle par une instruction est appelé **latence**

Architecture pipeline

Phases de l'étage Exécution:

- Les latences des instructions C6x en virgule fixe sont toujours égales à 1 cycle

Instruction C6x	Délai	Latence	#Phases
NOP	0	1	1
Instruction à 1 cycle	0	1	1
Multiplication (MPY)	1	1	2
Chargement (LDx)	4	1	5
Branchement (B)	5	1	6
Comparaison DP (C67x)	1	2	2
Conversion INT → DP (C67x)	4	1	5
ADDDP/SUBDP (C67x)	6	2	7
MPYDP (C67x)	9	4	10

Instructions à virgule flottante

Architecture pipeline

- **Phases de l'étage Exécution:**
  - Exemple pipeline C6x à virgule fixe (1 EP par FP)

Clock Cycle											
1	2	3	4	5	6	7	8	9	10	11	12
PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5	E6
	PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5
		PG	PS	PW	PR	DP	DC	E1	E2	E3	E4
			PG	PS	PW	PR	DP	DC	E1	E2	E3
				PG	PS	PW	PR	DP	DC	E1	E2
					PG	PS	PW	PR	DP	DC	E1
						PG	PS	PW	PR	DP	DC

FP

Effet Pipeline

Registres C6x

Registres à usage général

- Le C6x possède un registre file sur chaque chemin DP
- Chaque registre file comprend 16 registres 32 bits
  - A0-A15
  - B0-B15
- Les registres peuvent être utilisés pour:
  - Données
  - Pointeurs d'adresses
  - Tests de conditions

Registres C6x

Registres à usage général

- Ces registres sont directement connectées aux port des unités fonctionnelles

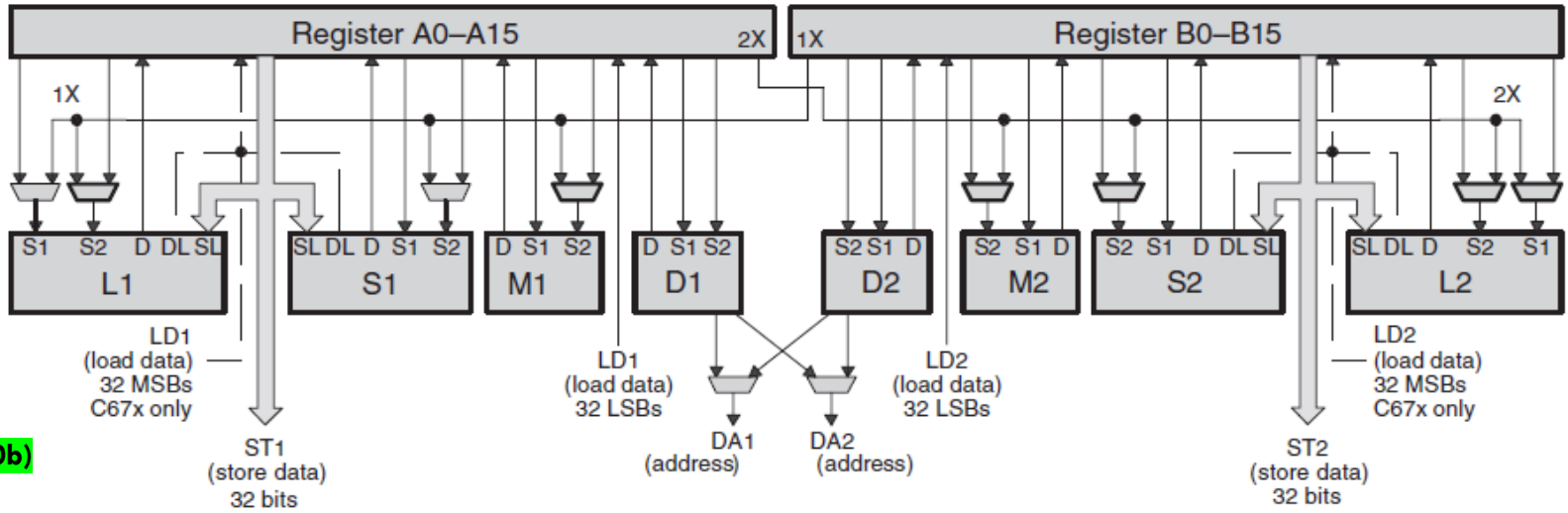
S1 =source 1 (32b)

S2 =source 2 (32b)

D =destination (32b)

SL = source long (40b)

DL=destination long (40b)



Registres C6x

Registres à usage général

- Les registres C62x/C67x peuvent traiter des données en:
  - « Packed » 16 bits (SHORT)
  - 32bits (INT ou SP)
  - 40bits (LONG)
  - 64bits (Double LONG ou DP)

- Quelques emplois particuliers des registres:
  - A0-A1 et B0-B2 : Tests des conditions
  - A4-A7 et B4-B7 : Adressage circulaire

Registres C6x

Paires de registres 40bits/64bits:

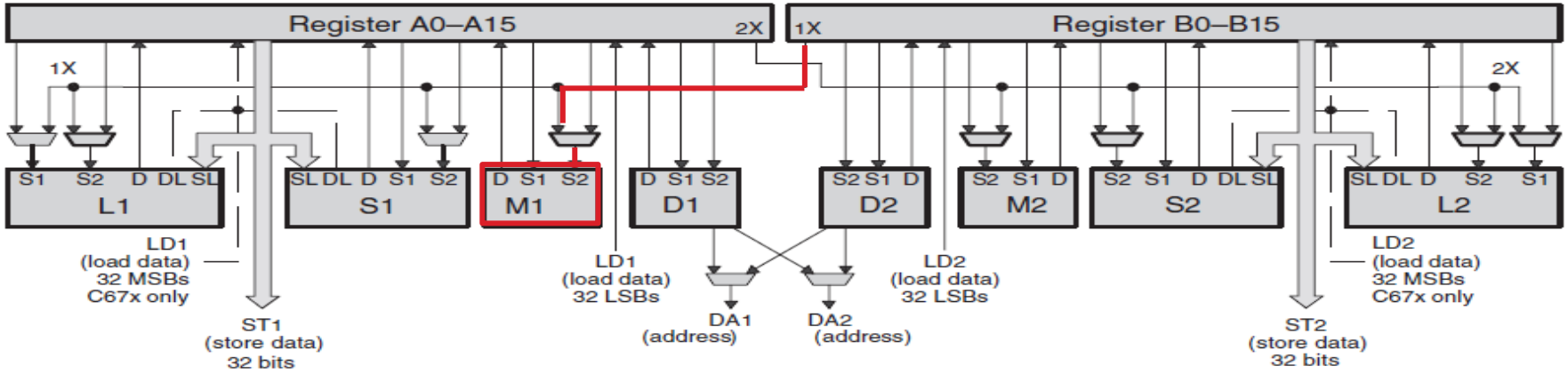
- Dans le cas 40 et 64bits, une paire de registres est utilisée:
  - Les 32bits LSB sont stockés dans un registre d'index pair
  - Les bits MSB (8bits ou 32bits) sont stockés dans le registre suivant qui a toujours un index impaire
  - Exemple:
    - **A1 : A0**
    - Le tableau à droite illustre toutes les combinaisons possibles

A	B
A1:A0	B1:B0
A3:A2	B3:B2
A5:A4	B5:B4
A7:A6	B7:B6
A9:A8	B9:B8
A11:A10	B11:B10
A13:A12	B13:B12
A15:A14	B15:B14

Registres C6x

Chemins croisés 1X et 2X

- Les unités L1 S1 et M1 peuvent lire les registres B à travers un chemin croisé appelé **1X**
- Les unités L2 S2 et M2 peuvent lire les registres A à travers un chemin croisé appelé **2X**
- Le nombre de lecture est limité à une lecture par chemin par cycle (2 instructions max en ||)
- Exemple : **MPY** .M1X A5 , B5 , **A5**



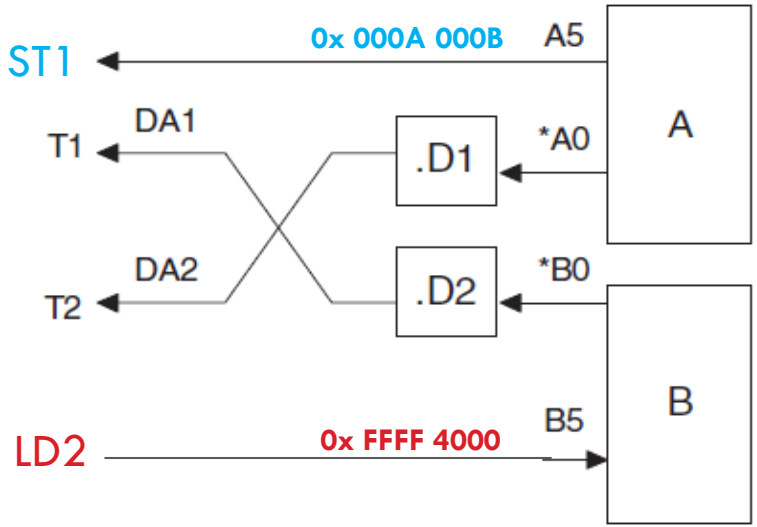
Registres C6x

Chemins croisés T1 et T2

- Il existe deux bus adresses données
  - DA1 appelé chemin croisé T1
  - DA2 appelé chemin croisé T2

- Exemple:
  - Chargement valeur 0x FFFF 4000
  - Stockage valeur 0x000A 000B

```
LDW .D1T2  *A0 , B5
|| STW .D2T1   A5 , *B0
```



Registres C6x

Registres de contrôle C6x

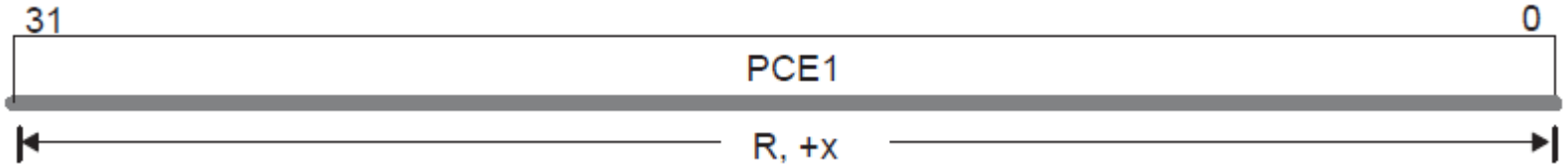
- Le C6x inclut un registre file de plusieurs registres de contrôle
- Ces registres sont accessibles en lecture/écriture seulement
- L'accès se fait uniquement via ***.S2*** en utilisant l'instruction **MVC** (un cycle E1 du pipeline)
- Registres communs aux C6x:
  - CSR: Control Status Register
  - PCE1: Program Counter E1
  - AMR : Address Modes Register

Registres C6x

Registres de contrôle C6x

PCE1: Compteur Programme phase E1

- Contient l'adresse du FP dans la phase E1 du pipeline



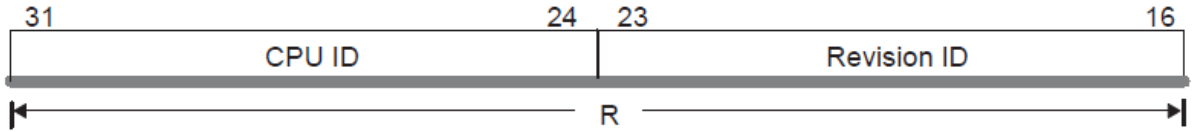
- PCE1: Program Counter phase E1
- R: Read (Lecture via **MVC**)
- +x: Valeur indéterminée après un RESET

Registres C6x

Registres de contrôle C6x

CSR: Registre Contrôle Status

- CSR 16bits MSBs



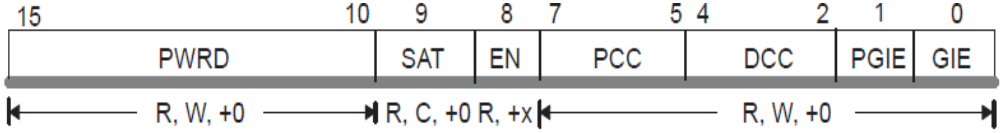
N° Bits	Nom	Fonction
31-24	CPU ID	Identifie le type du CPU
23-16	Rev. ID	Identifie la version de la révision matérielle

Registres C6x

Registres de contrôle C6x

CSR: Registre Contrôle Status

- CSR 16bits LSBs



- PWRD: Power Down
- SAT: Saturation
- EN: Endianess
- P(D)CC: Program(Data) Cache Control
- (P)GIE: (Previous)Global Interrupt Enable

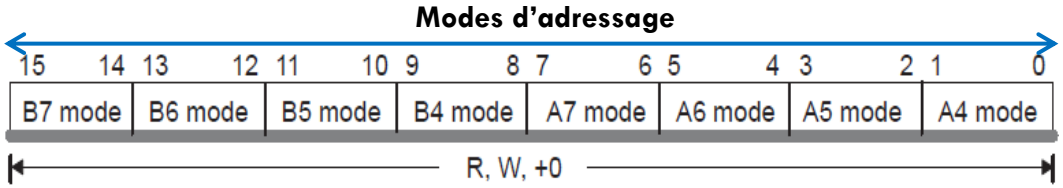
N° Bits	Nom	Fonction
15-10	PWRD	Contrôle de la gestion d'énergie
9	SAT	(1) si saturation utilisé
8	EN	Endianess (1): Big (0): Little
7-5	PCC	Contrôle cache Programme
4-2	DCC	Contrôle cache Données
0	GIE	(1): Activation globale des interruptions
1	PGIE	Maintient de l'interruption prise antérieurement

Registres C6x

Registres de contrôle C6x

AMR: Registre mode d'adressage

- AMR 16bits LSBs
- **BK** = Taille du block de l'adressage circulaire



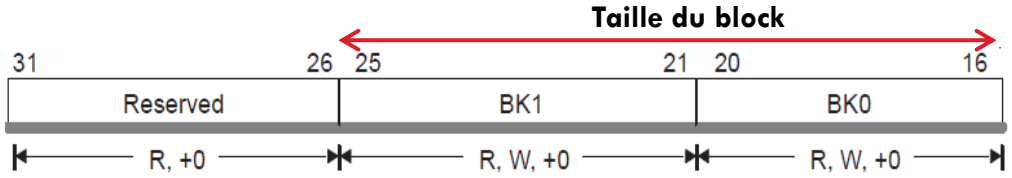
2 bits du Rx	Mode @
00	Linéaire
01	Circulaire Bk0
10	Circulaire Bk1
11	Réservé

Registres C6x

Registres de contrôle C6x

AMR: Registre mode d'adressage

- AMR 16bits MSBs
- **BK** $= 2^{N+1}$ où N = 5bits
- On peut choisir deux tailles de block: BK1 ou BK2



N	BK
00000	2 bits
00110	128 bits
01001	1k
11111	4G (linéaire)

Timers

- Le C6x dispose de TIMER 32 bits à usage général pour:
  - Mesure de la durée d'événements
  - Énumération des événements
  - Génération des impulsions
  - Génération d'interruptions
  - Synchronisation DMA
- Le TIMER peut avoir une source d'horloge interne ($Fosc/4$) ou externe et dispose de deux pins :
  - TOUT Timer Output: fonctionne en sortie horloge ou sortie à usage général
  - TINP Timer Input : fonctionne en entrée horloge ou entrée à usage général

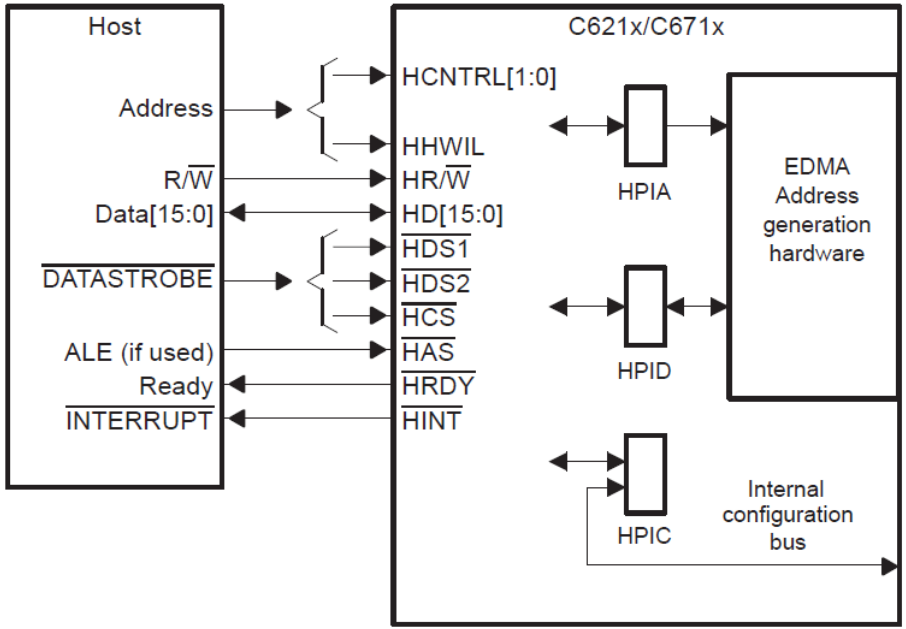
Port parallèle HPI

- **Host Port Interface (HPI)**
  - Le HPI est un port parallèle à travers qui un processeur maitre (PC) pourra accéder à l'espace mémoire DSP
  - Le Host pourra aussi accéder aux périphériques mappés en mémoire
  - L'accès à ces espaces mémoires est effectué à travers le contrôleur de la mémoire EDMA ou de la mémoire DMA

Port parallèle HPI

Interface

- **Signaux HPI:**
  - HD[15:0]: Bus données 16bits
  - HR/$\overline{W}$: Sélection Lecture/Ecriture
  - HCNTRL: Type d'accès Host
  - HHWIL: Entrée sur un demi-mot
  - $\overline{HCS}$, $\overline{HDS1}$ et $\overline{HDS2}$: Signalements
  - $\overline{HAS}$: Similaire au ALE
  - $\overline{HRDY}$: Accés Host prêt
  - $\overline{HINT}$: Sortie Interruption vers le Host



Port parallèle HPI

Registres

- **HPIC: HPI Control**
  - Contient les bits de configuration et d'initialisation de l'interface HPI
  - Mappé en mémoire : 0x 0188 0000
- **HPID: HPI Data**
  - Contient les data lu sur la mémoire après accès HPI
  - Contient les data écrit sur la mémoire après accès HPI
  - Non mappé sur l'espace mémoire
- **HPIA : HPI Address**
  - Contient l'adresse mémoire où l'accès en cours à eu lieu
  - Accessible seulement par le Host
  - Non mappé sur l'espace mémoire

Port série McBSP

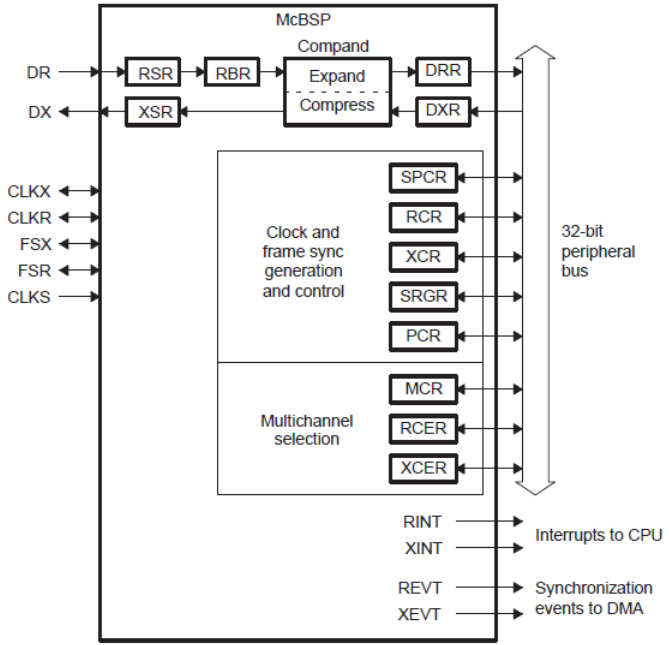
Multichannel Buffered Serial Port (McBSP)

- Le McBSP est un périphérique d'interface série qui a pour principales caractéristiques:
  - Communication Full Duplex
  - 2 Registres Buffer (1 pour l'émission et 1 pour la réception)
  - Transceiver multicanaux (128 canaux)
  - Traitement données en 8,12,16,20,24 et 32bits
  - Interface directe avec les standards industriels :
    - Codec audio AC97
    - Interface série SPI
    - Compander µ/A

Port série McBSP

Interface

- **Signaux:**
  - **D R/X:** Data réception/transmission
  - **CLK R/X:** Horloge réception/transmission
  - **CLK S:** Horloge externe
  - **FS R/X:** Synchronisation trame réception/transmission
  - **INT R/X:** Interruptions CPU en réception/transmission
  - **EVT R/X:** Interruptions EDMA en réception/transmission



Périphériques du C6x

E/S à usage général ou GPIO

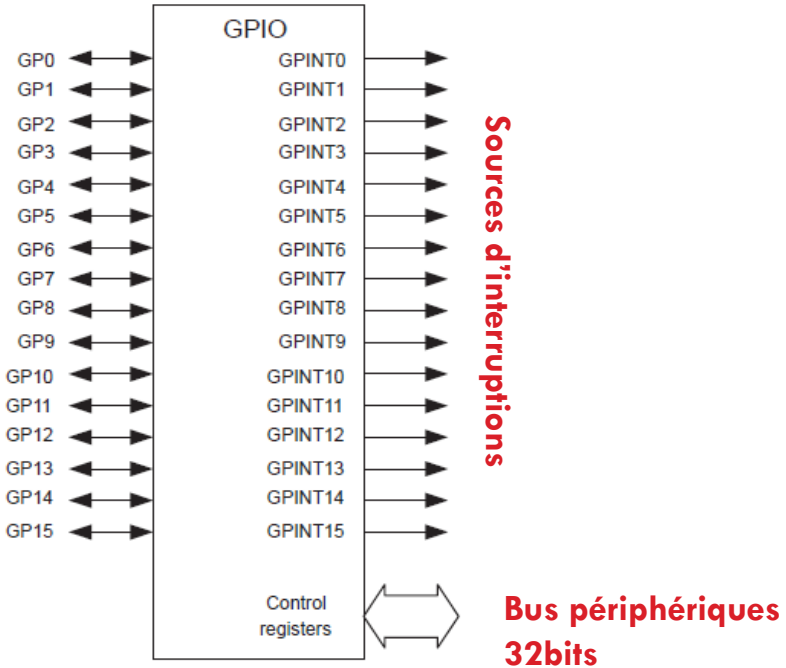
- Ce périphérique offre 16 pins à usage général GP0-GP15 qui peuvent être utilisés comme entrées ou comme sorties

- Le périphérique GPIO peuvent générer des interruptions externes pour le CPU ou de signaux de synchronisation pour la mémoire EDMA
  - GP0-GP15: sources de synchronisation mémoire EDMA
  - GP0 et GP4-GP7: sources d'interruptions externes

Périphériques du C6x

- **E/S à usage général ou GPIO**
  - **Interface GPIO**

GPIO: General-Purpose Input/Output



Interruptions

Types d'interruptions du C6x

- Le C6x peut gérer 3 types d'interruptions classées par ordre de priorité:
  - 01 interruption $\overline{\text{RESET}}$ (Priorité haute)
    - Dure 21 cycles
  - 01 interruption non masqués NMI
    - Evènements très critiques ex: défaillance imminente (perte alimentation)
  - 12 interruptions masqués INTm (INT4-INT15) (Priorité basse)
    - Evénements internes et/ou externes

Interruptions

Organisation des interruptions

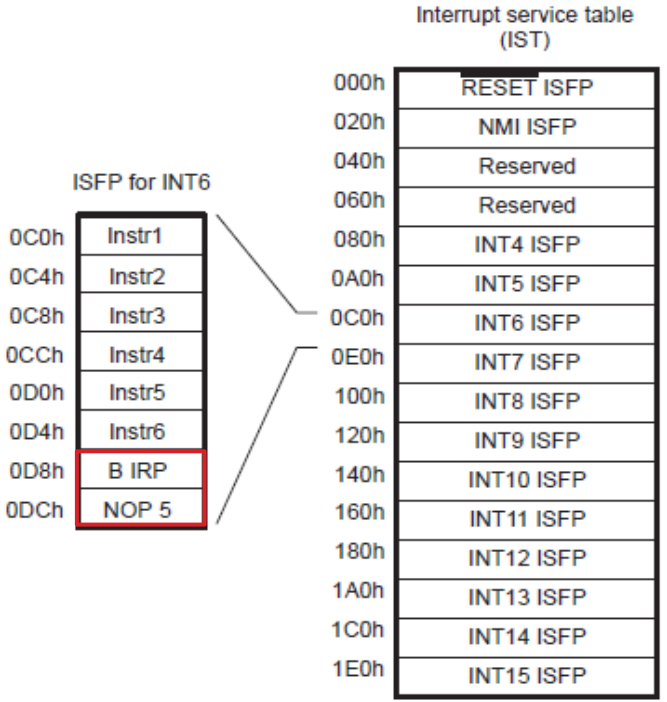
- Les services d'interruption sont sous forme de paquets FP appelé **ISFP (Interrupt Service FP)**

- Les adresses ISFP sont réunis dans une table mémoire programme **IST (Interrupt Service Table) :**

  - **0x000:** ISFP du $\overline{\text{RESET}}$
  - **0x020:** ISFP du NMI
  - **0x040-0x060:** Réservé
  - **0x080-0x1E0:** ISFP des interruptions masqués (INT4-INT15)

Les interruptions

- **Table IST**
  - **Registres Pointeurs : ISTP, NRP, IRP**
    - ISTP = 0x @IST = 0x000
    - NRP = 0x @EP (dernier EP avant NMI)
    - IRP = 0x @EP (dernier EP avant INTm)
    - Exemple avec une interruption masquée (INTm):
      - **m = 6 ➔ Interruption masquée INT6**

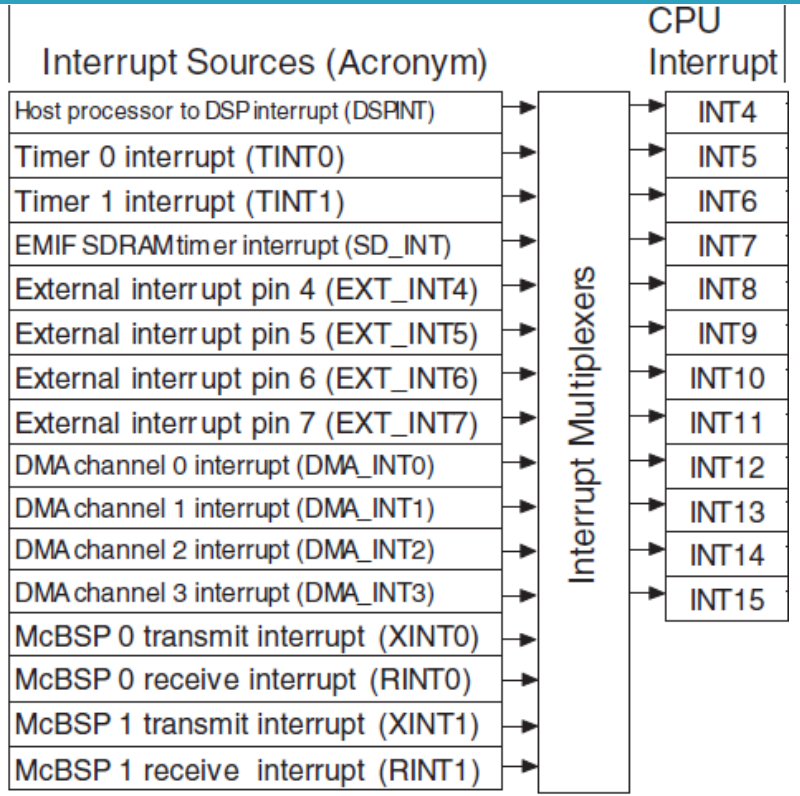


Interruptions

Les interruptions masquées

- Il existe plusieurs sources d'interruptions masquées :
  - Interruptions HPI (Connexion, Déconnexion, …)
  - Interruptions Timer (Overflow)
  - Interruptions EMIF (Lecture, Ecriture, …)
  - Interruptions DMA (Lecture, Ecriture, …)
  - Interruptions McBSP (Transmission, Réception..)

- Ces sources sont multiplexés en 12 sources d'interruptions (INT4 jusqu'à INT15)



Les interruptions

Registres de contrôle des interruptions

- La gestion de toutes ces interruptions passe par 8 registres qui sont classés en :
  - Registres Activation d'interruption
  - Registres Etat :
  - Registres Pointeurs

Les interruptions

- **Registres de gestion des interruptions**

	Reg	Nom	Fonction
Activation	CSR	Control Status Register	Activation globale des interruptions masqués
Activation	IER	Interrupt Enable Register	Activation individuelle des interruptions
Etats	IFR	Interrupt Flag Register	Etat actuelle des interruptions NMI et INTm
Etats	ISR	Interrupt Set Register	Changer manuellement l'état des interruptions INTm via le registre IFR
Etats	ICR	Interrupt Clear Register	
Pointeurs	ISTP	Interrupt Service Table Register	Pointe vers l'adresse du ISFP à exécuter à partir de la table IST
Pointeurs	NRP	NMI Return Pointer	Pointe vers l'adresse du EP qui été actif avant une interruption NMI
Pointeurs	IRP	Interrupt Return Pointer	Pointe vers l'adresse du EP qui été actif avant une interruption masquée INTm