TP n° 8 - Les DSP

Les DSP (Digital Signal Processors) sont, comme leur nom l'indique, des processeurs dédiés au traitement des signaux numériques.

I. Spécificités des DSP

Les particularités de ces puces, par rapport aux processeurs généraux (Intel, Cyrix, AMD...) sont, en général :

- Une architecture Harvard, séparant bus programme et bus données
- Des instructions adaptées aux opérations classiques de traitement de signaux : multiplication & accumulation en un seul cycle d'instruction (MAC), transferts de données par DMA (Direct Access Memory) pour la plupart
- Un adressage circulaire, permettant de ne pas faire des tests de modulo sur les pointeurs de tampons
- Un mode d'adressage par inversion de bits, servant à réorganiser les échantillons de sortie de Transformées de Fourier Rapides (TFR ou FFT).
- De la mémoire interne minimisant les temps d'accès RAM
- Des banques de mémoire, permettant de scinder par exemple partie réelle et partie imaginaire
- Une architecture à pipe-line permettant de paralléliser le maximum d'opérations élémentaires (préchargement d'instruction, chargement des données, exécution d'opérations arithmétique/flottante, stockage des résultats)
- Des entrées sorties dignes d'un microcontrôleur
- Parfois même des convertisseurs analogiques-numériques numériques analogiques intégrés !

Ces différents atouts ne se retrouvent pas sur tous les DSP. Les DSP bas de gamme ne disposant, en général que d'une architecture Harvard, d'instructions adaptées et de mode d'adressage à inversion de bits.

II. Caractéristiques principales des DSP

Ce qui distingue deux DSP différents sont, pour les caractéristiques principales :

- Son type : virgule fixe / virgule flottante. Je vous propose d'en apprécier la différence dans le prochain paragraphe.
- Sa vitesse, exprimée en MIPS (Mega Instructions par Seconde), qui n'est pas obligatoirement représentative des performances du DSP. En effet, le nombre de cycles d'horloge par instruction peut varier d'un DSP à l'autre. De plus, cela dépend aussi de la complexité des instructions. Les instructions peuvent être basiques (lecture/écriture d'une donnée dans un registre) ou plus étoffées (chargement d'une donnée, élévation au carré, addition d'un registre avec l'accumulateur) compte-tenu de l'architecture même du DSP.
- Sa quantité de mémoire interne (DRAM/RAM/ROM/Flash...)
- Ses entrées/sorties (ports série, ports parallèles) et leurs vitesses respectives
- Son architecture interne, avec la présence ou non de canaux DMA. L'architecture jouera en effet sur le degré de parallélisme, la gestion des ruptures de pipe-line, la facilité du multi-processing...

Le choix d'un DSP pour une application particulière sera conditionné par le coût relatif du DSP par rapport au BOM (Bill of Material) de l'application, et de la complexité des opérations à effectuer. Il est clair qu'un DSP introduit dans un appareil grand public est plus souvent à virgule fixe. Si les coûts de développement d'une application en virgule fixe sont beaucoup plus importants qu'un design en virgule flottante, les volumes viendront amortir cette dépense... Il faut l'espérer.

Pour ce qui à trait à la vitesse, on peut se poser la question en ces termes:
"Quel est le flux de données maximum à traiter ? Puis-je déplacer le problème de vitesse en problème de mémoire ?"
En effet, pour certaines applications, on n'aura pas besoin d'une grande vitesse, à condition d'avoir de la mémoire. Les algorithmes seront plus simples. Il suffit de prendre l'exemple suivant pour se rendre compte.
Pour implémenter un algorithme de traitement de signal adaptatif, nous pouvons utiliser un algorithme simple stable du type LMS qui ne nécessite pas de grandes ressources en terme de calcul. En revanche, un nombre élevé de coefficients sera nécessaire pour parvenir à converger rapidement. En utilisant un algorithme plus complexe du type RMS, le nombre de coefficients du filtre sera beaucoup moins élevé, mais les quantités de calculs par coefficients seront plus importantes.
Le travail du concepteur sera alors dans le choix d'un compromis, et éventuellement, d'astuces de programmation - voire même en assembleur - pour parvenir à ses fins en réduisants les coûts, en fonction de la stratégie générale de son entreprise.

III. Virgule fixe / Virgule flottante

Deux familles principales de DSP se partagent le marché. Les premiers sont les DSP à virgule fixe, qui nécessitent une attention accrue du développeur lors de la programmation. Les seconds sont les DSP à virgule flottante, aisés d'utilisation mais plus onéreux.

Aperçu des contraintes de programmation d'un DSP fixe

Dans un DSP à virgule fixe, c'est au programmeur de connaître à tout instant l'état de ses opérateurs. Supposons que nous voulions faire le produit de deux nombres a et b.
Les questions que devra se poser le programmeur sont :

1. Quelles sont les dynamiques réelles des nombres a et b ?
Réponse : $-3.5 < a < 3.5$ et $0 < b < 100$

2. Quelle est la dynamique du résultat $P = ab$?
Réponse : $-350 < P < 350$

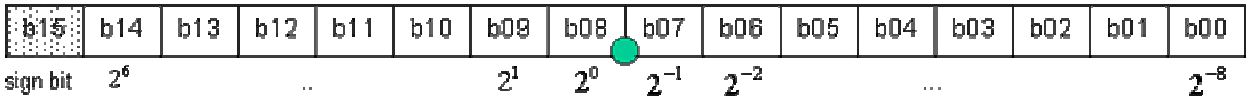
3. Quelle est la dynamique de mon DSP ?
Réponse : 16 bits

4. Quelle est la précision que je désire ?
Réponse : Le maximum à condition de coder mes nombres sur 16 bits.

5. Alors ? Quelles solutions ?
Réponse :

Et bien, tout d'abord, "a" peut-être codé en format Q2.13, b peut-être codé en format Q7.8.

Qi.f signifie : coder la partie entière du nombre avec i bits, coder la partie fractionnaire avec f bits. i+f=15, il reste ainsi 1 bit pour coder le signe.



L'exemple ci-dessus représente un codage Q7.8 pouvant coder des nombres compris entre -128 et +127.99609375.

Pour retrouver ce résultat, il suffit d'appliquer la formule suivante :

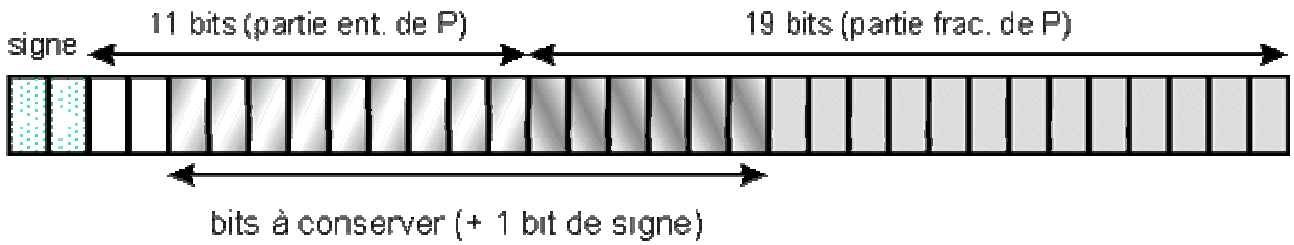
$$N = -b_{15} \cdot 2^{i} + \sum_{k=0}^{f-1} b_k \cdot 2^{k-f} + \sum_{k'=0}^{i-1} b_{k'+f} \cdot 2^{k'}$$

avec i=7 et f=8.

3 peut être codé avec 2 bits, il reste donc 13 bits pour coder la partie fractionnaire de a. 100 peut être codé sur 7 bits, laissant ainsi 8 bits pour la partie fractionnaire de b.

Une multiplication d'un nombre codé en Qi.f par un nombre codé en Qi'.f' donne un résultat dont le codage est Qi+i'.f+f' avec...2 bits de signe.

L'accumulateur d'un DSP sur D bits a une longueur minimale de 2D bits. Ainsi, dans notre exemple, le résultat de la multiplication sera en Q11.19.A présent, il est judicieux de se référer à la condition 4 : à savoir, avoir le maximum de précision tout en restant sur 16 bits.350 doit être codé au minimum sur 9 bits. Il reste ainsi 6 bits pour coder la partie fractionnaire du résultat.



Au bout du compte, la précision que l'on pourra obtenir, en stockant le résultat sur 16 bits, sera :

$$-512 < P < +511.984375.$$

DSP à virgule flottante

Sur un DSP à virgule flottante, les calculs sont transparents pour l'utilisateur. Les temps de développement sont réduits de façon conséquente. Les nombres sont codés sous la forme :

Signe Exposant Mantisse, généralement sur 32 à 40 bits.

Ainsi, la dynamique est bien meilleure ! Mais les opérations effectuées par les circuits (hardware) sont beaucoup plus complexes qu'avec un DSP à virgule fixe. Ceci implique que les DSP à virgule flottante sont vendus à des prix bien supérieurs aux fixes...

Les projets impliquant des DSP à virgule flottante nécessitent soit de grandes capacités de calcul, soit une grande précision, pour des volumes de production peu importants, ou des prix de ventes élevés. Les principaux consommateurs de DSP à virgule flottante sont les militaires et les grands systèmes de télécommunications.

IV. Conseils de programmation

Voici quelques conseils de programmation de DSP :

Les DSP sont la plupart du temps impliqués dans des applications temps réel. Les programmes et les tâches doivent ainsi souvent être optimisés pour réduire les charges CPU...et, au besoin (marketing) ajouter le maximum de fonctionnalités.

Ainsi, le programmeur cherchera à réduire les instructions dans son programme, utilisant le minimum de périodes d'horloge.

Pour cela il évitera :

- Les branchements, qui nécessitent souvent des vidages de pipe-line, prenant x périodes d'horloge (x étant la taille du pipe de 3 à 7)
- Les tests...pour les mêmes raisons
- De faire confiance aveuglément au compilateur C qu'il peut utiliser (règle d'or pour les calculs)

Il veillera à :

- Soigneusement définir le mapping mémoire. En effet, il faut mettre les variables utilisées le plus souvent dans la mémoire interne, à zéro "wait state". Les accès à la mémoire externe peuvent être très pénalisants suivant la technologie RAM.
- Bien dimensionner la taille de sa pile de façon à ne pas subir de débordement (des fois, le programme plante, et on se demande pourquoi !).

Eviter les tests et les branchements est certes difficile, mais il existe souvent des astuces permettant de ruser d'une façon habile. Par exemple, sur le Motorola 563xxx, l'instruction MOVE peut être conditionnée par le registre d'état, sans nécessiter de temps de cycle superflu.

Il reste que chaque DSP a sa propre philosophie, qu'il faut appréhender avec un peu de temps.