4. Classification des DSP

Il est impossible d'effectuer une classification « définitive » des DSP, car chaque constructeur met sur le marché tous les ans un nouveau composant qui surclasse les anciens ou les concurrents par la puissance de calcul, la rapidité (gestion du pipeline et fréquence d'Horloge), le nombre de registres, de Timers, de ports série…

4.1. Virgule fixe ou flottante

Un point essentiel des DSP est la représentation des nombres (les données) qu'ils peuvent manipuler. Il est possible de distinguer deux familles :

- **Les DSP à virgule fixe** : les données sont représentées comme étant des nombres fractionnaires à virgule fixe, (exemple -1.0 à +1.0), ou comme des entiers classiques. La représentation de ces nombres fractionnaires s'appuie la méthode du « complément à deux ». L'avantage de cette représentation (qui n'est qu'une convention des informaticiens) est de permettre facilement l'addition binaire de nombres aussi bien positifs que négatifs.

- **Les DSP à virgule flottante** : les données sont représentées en utilisant une mantisse et un exposant. La représentation de ces nombres s'effectue selon la formule suivante :

$$n = mantisse \times 2^{exposant}$$

Généralement, la mantisse est un nombre fractionnaire (-1.0 à +1.0), et l'exposant est un entier indiquant la place de la virgule en base 2 (c'est le même mécanise qu'en base 10).

4.1.1. Les DSP à virgules flottantes

Les DSP à virgule flottante sont plus souples et plus faciles à programmer que les DSP à virgule fixe. Un DSP comme le TMS320C30 manipule des nombres formés avec une mantisse de 24 bits et un exposant de 8 bits (taille de la donnée en mémoire : 32 bits). Les valeurs intermédiaires des calculs sont mémorisées dans des registres avec un format de 32 bits de mantisse et un exposant de 8 bits (taille du registre : 32 + 8 bits supplémentaires).

La dynamique disponible est très grande, elle va de $\mathbf{-1 \times 2^{128}}$ à $\boldsymbol{(1-2^{-23}) \times 2^{127}}$, toutefois la résolution reste limitée à 24 bits au mieux. Outre les nombres fractionnaires, ce DSP sait également manipuler les entiers avec une précision de 32 bits.

La très grande dynamique proposée par les DSP à virgule flottante permet virtuellement de ne pas se soucier des limites des résultats calculés lors de la conception d'un programme. Cet avantage a cependant un prix, à savoir qu'un système basé sur un DSP à virgule flottante a un coût de fabrication supérieur par rapport à un système basé sur DSP à virgule fixe. La puce d'un DSP à

virgule flottante nécessite à la fois une surface de silicium plus importante (cœur plus complexe), et un nombre de broches supérieur, car la mémoire externe est elle aussi au format 32 bits. Le système revient donc plus cher (exemple : 2 x 32 broches ne serait ce que pour les bus de données externes avec une architecture Harvard de base). Un DSP à virgule flottante est plutôt adapté (sans être impératif) à des applications dans lesquelles :

- les coefficients varient dans le temps (exemple : les filtres adaptatifs),

- le signal et les coefficients ont besoin d'une grande dynamique,

- la structure mémoire est importante (exemple : traitement d'image),

- la précision est recherchée sur toute une gamme dynamique importante (exemple : traitements audiophoniques de qualité professionnelle).

De part leurs facilités de programmation, ils peuvent également se justifier dans des projets ou le temps et la facilité de développement sont des facteurs importants. On les trouve également dans des produits de faible volume de production, pour lesquels le prix du DSP n'est pas significatif.

4.1.2. Les DSP à virgules fixes

Un DSP à virgule fixe est un peu plus compliqué à programmer qu'un DSP à virgule flottante.

Représentation des **nombres entiers** codés sur 4 bits en complément à 2 est donnée comme suit :

Alors la représentation des **nombres fractionnaires** codés sur 4 bits en complément à 2 est donnée comme suit :

Dans un DSP à virgule fixe typique comme le TMS320C25, les nombre sont codés sur 16 bits (rappel : des entiers classiques ou des fractionnaires).

Toutefois, sur ce DSP, les calculs sont effectués avec des accumulateurs de 32 bits. Lorsque les résultats doivent être stockés en mémoire, les 16 bits les moins significatifs sont perdus. Ceci permet de limiter les erreurs d'arrondis cumulatives. Il est toujours possible de stocker séparément en mémoire les 16 bits faibles puis les 16 bits fort s'il n'y a plus de registres libres lors d'une étape de calcul. Cette particularité n'est pas toujours disponible sur tous les DSP.

Dans ce cas, les calculs requérant absolument une double précision sont réalisés en chaînant deux à deux des instructions spéciales manipulant des données 16 bits en simple précision, au détriment du temps d'exécution.

La précision des calculs est un point critique des DSP à virgule fixe, car le concepteur de programmes doit rester vigilant à chaque étape d'un calcul. Il doit

rechercher la plus grande dynamique possible (c.à.d. exploiter au mieux la gamme des nombres disponibles), pour conserver une bonne précision des calculs, tout en évitant autant que faire ce peu les débordements du ou des accumulateurs. Les bits supplémentaires des accumulateurs (les bits de garde) prévus à cet effet permettent de réduire cette contrainte.

Les programmeurs contournent les limites des DSP à virgule fixe en déterminant à l'avance, et avec soins, la précision et la dynamique nécessaire (par méthode analytique ou avec des outils de simulation) pour réaliser leurs projets. Il est également possible d'effectuer des opérations en virgule flottante dans un DSP à virgule fixe par le biais de routines logicielles adéquates. Cette approche est néanmoins pénalisante en temps d'exécution, même sur un DSP à virgule fixe très rapide.

En termes de rapidité, les DSP à virgule fixe se placent d'ordinaire devant leurs homologues à virgule flottante, ce qui constitue un critère de choix important. Les DSP à virgule fixe sont les plus utilisés, car ils sont moins chers que les DSP à virgule flottantes. On les trouve dans tous les produits de grande diffusion ou le coût est un facteur important. Il peut cependant exister des exceptions, certains DSP à virgule fixe se présentant comme des microcontrôleurs perfectionnés plus chers qu'un DSP à virgule flottante de base.

Dans ce qui suit on revient à la représentation des nombres réels avec de s exercices:

Représentation de la partie décimale d'un nombre

Partons tout de suite sur un exemple : comment représenter 5,1875 en binaire ?

Il nous faut déjà représenter 5, ça, pas de problème : 101

Comment représenter le ",1875" ?

on multiplie 0,1875 par 2 : 0,1875 x 2 = 0,375. On obtient 0,375 que l'on écrira 0 + 0,375

on multiplie 0,375 par 2 : 0,375 x 2 = 0,75. On obtient 0,75 que l'on écrira 0 + 0,75

on multiplie 0,75 par 2 : 0,75 x 2 = 1,5. On obtient 1,5 que l'on écrira 1 + 0,5 (quand le résultat de la multiplication par 2 est supérieur à 1, on garde uniquement la partie décimale)

on multiplie 0,5 par 2 : 0,5 x 2 = 1,0. On obtient 1,0 que l'on écrira 1 + 0,0 (la partie décimale est à 0, on arrête le processus)

On obtient une succession de "a + 0,b" ("0 + 0,375", "0 + 0,75", "1 + 0,5" et "1 + 0,0"). Il suffit maintenant de "prendre" tous les "a" (dans l'ordre de leur obtention) afin d'obtenir la partie décimale de notre nombre : 0011

Nous avons $(101,0011)_2$ qui est la représentation binaire de $(5,1875)_{10}$

À faire vous-même EX1

Trouvez la représentation binaire de $(4,125)_{10}$

Il est possible de retrouver une représentation décimale en base 10 à partir d'une représentation en binaire.

Partons de $(100,0101)_2$

Pas de problème pour la partie entière, nous obtenons "4". Pour la partie décimale nous devons écrire : $0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0,3125$. Nous avons donc $(4,3125)_{10}$

À faire vous-même EX 2

Trouvez la représentation décimale de $(100,001)_2$

À faire vous-même EX3

Trouvez la représentation binaire de $(0,1)_{10}$

Que remarquez-vous ?

Dans l'exemple ci-dessus, nous remarquons que le processus de "conversion" ne s'arrête pas, nous obtenons : "0,0001100110011...", le schéma "0011" se répète à "l'infini". Cette caractéristique est très importante, nous aurons l'occasion de revenir là-dessus plus tard.

En base dix, il est possible d'écrire les très grands nombres et les très petits nombres grâce aux "puissances de dix" (exemples "$6,02.10^{23}$" ou "$6,67.10^{-11}$"). Il est possible de faire exactement la même chose avec une représentation binaire, puisque nous sommes en base 2, nous utiliserons des "puissances de deux" à la place des "puissances dix" (exemple "$101,1101.2^{10}$").

Pour passer d'une écriture sans "puissance de deux" à une écriture avec "puissance de deux", il suffit de décaler la virgule : "$1101,1001 = 1,1011001.2^{11}$" pour passer de "1101,1001" à "1,1011001" nous avons décalé la virgule de 3 rangs vers la gauche d'où le "2^{11}" (attention de ne pas oublier que nous travaillons en base 2 le "11" correspond bien à un décalage de 3 rangs de la virgule).

Si l'on désire décaler la virgule vers la gauche, il va être nécessaire d'utiliser des "puissances de deux négatives" "$0,0110 = 1,10.2^{-10}$", nous décalons la virgule de 2 rangs vers la droite, d'où le "-10"

Représentation des flottants dans un ordinateur

La norme IEEE 754 est la norme la plus employée pour la représentation des nombres à virgule flottante dans le domaine informatique. La première version de cette norme date de 1985.

Nous allons étudier deux formats associés à cette norme : le format dit "simple précision" et le format dit "double précision". Le format "simple précision" utilise

32 bits pour écrire un nombre flottant alors que le format "double précision" utilise 64 bits. Dans la suite nous travaillerons principalement sur le format 32 bits.

Que cela soit en simple précision ou en double précision, la norme IEEE754 utilise :

1 bit de signe (1 si le nombre est négatif et 0 si le nombre est positif) des bits consacrés à l'exposant (8 bits pour la simple précision et 11 bits pour la double précision) des bits consacrés à la mantisse (23 bits pour la simple précision et 52 bits pour la double précision)

Nous pouvons vérifier que l'on a bien 1 + 8 + 23 = 32 bits pour la simple précision et 1 + 11 + 52 = 64 bits pour la double précision.

Pour écrire un nombre flottant en respectant la norme IEEE754, il est nécessaire de commencer par écrire le nombre sous la forme $1,XXXXX.2^{e}$(avec e l'exposant), il faut obligatoirement qu'il y ait un seul chiffre à gauche de la virgule et il faut que ce chiffre soit un "1". Par exemple le nombre "1010,11001" devra être écrit "$1,01011001.2^{11}$". Autre exemple, "0,00001001" devra être écrit "$1,001.2^{-101}$".

La partie "XXXXXX" de "$1,XXXXX.2^{e}$" constitue la mantisse (dans notre exemple "1010,11001" la mantisse est "01011001"). Comme la mantisse comporte 23 bits en simple précision, il faudra compléter avec le nombre de zéro nécessaire afin d'atteindre les 23 bits (si nous avons "01011001", il faudra ajouter 23 - 8 = 15 zéros à droite, ce qui donnera en fin de compte "01011001000000000000000")

Notre première intuition serait de dire que la partie "exposant" correspond simplement au "e" de "$1,XXXXX.2^{e}$" (dans notre exemple "1010,11001", nous aurions "11"). En faite, c'est un peu plus compliqué que cela. En effet, comment représenter les exposants négatifs ? Aucun bit pour le signe de l'exposant n'a été prévu dans le norme IEEE754, une autre solution a été choisie :

Pour le format simple précision, 8 bits sont consacrés à l'exposant, il est donc possible de représenter 256 valeurs, nous allons pouvoir représenter des exposants compris entre $(-126)_{10}$ et $(+127)_{10}$ (les valeurs -127 et +128 sont des valeurs réservées, nous n'aborderons pas ce sujet ici). Pour avoir des valeurs uniquement positives, il va falloir procéder à un décalage : ajouter systématiquement 127 à la valeur de l'exposant. Prenons tout de suite un exemple (dans la suite, afin de simplifier les choses nous commencerons par écrire les exposants en base 10 avant de les passer en base 2 une fois le décalage effectué) :

Repartons de "1010,11001" qui nous donne $1,01011001.2^{3}$, effectuons le décalage en ajoutant 127 à 3 : "$1,01011001.2^{130}$", soit en passant l'exposant en base 2 : "$1,01011001.2^{10000010}$". Ce qui nous donne donc pour "1010,11001" une mantisse "01011001000000000000000" (en ajoutant les zéros nécessaires à droite pour avoir 23 bits) et un exposant "10000010" (même si ce n'est pas le cas ici, il peut être nécessaire d'ajouter des zéros pour arriver à 8 bits, ATTENTION, ces zéros devront être rajoutés à gauche).

À noter que pour le format double précision le décalage est de 1023 (il faut systématiquement ajouter 1023 à l'exposant afin d'obtenir uniquement des valeurs positives)

Nous sommes maintenant prêts à écrire notre premier nombre au format simple précision :

Soit le nombre "-10,125" en base 10 représentons-le au format simple précision :

nous avons $(10)_{10} = (1010)_2$ et $(0,125)_{10} = (0,001)_2$ soit $(10,125)_{10} = (1010,001)_2$

Décalons la virgule : $1010,001 = 1,010001.2^3$, soit avec le décalage de l'exposant $1,010001.2^{130}$, en écrivant l'exposant en base 2, nous obtenons $1,010001.2^{10000010}$

Nous avons donc : notre bit de signe = 1 (nombre négatif), nos 8 bits d'exposant = 10000010 et nos 23 bits de mantisse = 01000100000000000000000

Soit en "collant" tous les "morceaux" : 11000001001000100000000000000000

Cette écriture étant un peu pénible, il est possible d'écrire ce nombre en hexadécimal : C1220000