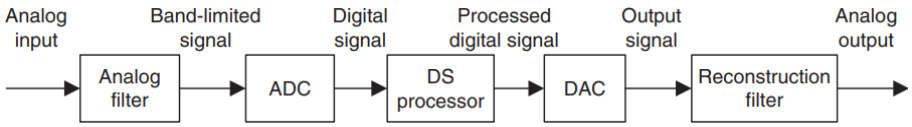
ARCHITECTURE D'UN DIGITAL SIGNAL PROCESSING (DSP)

1 INTROCUTION GENERALE

La technologie du traitement numérique du signal et ses progrès ont eu un impact considérable sur notre société moderne partout dans le monde. Sans DSP (pour Digital Signal Processing), nous n'aurions pas d'audio ou de vidéo numérique et internet; enregistrement numérique; lecteurs CD, DVD et MP3; Caméras digitales; téléphones numériques et cellulaires; satellite et télévision numérique; ou des réseaux avec ou sans fil. Les instruments médicaux seraient moins efficaces ou incapables de fournir des informations utiles pour des diagnostics précis s'il n'y avait pas d'analyseurs d'électrocardiographie numérique (ECG) ou de systèmes de radiographie numérique et d'imagerie médicale. Sans DSP, les scientifiques, les ingénieurs et les technologues n'auraient aucun outil puissant pour analyser et visualiser les données et effectuer leur conception, etc.

Le concept de DSP est illustré par le schéma fonctionnel simplifié de la figure ci-dessous, qui se compose d'un filtre analogique d'anti-repliement, d'une unité de conversion analogique-numérique (ADC), d'un processeur de signal numérique (DS Processor), d'une conversion numérique-analogique (DAC) et un filtre de reconstruction (anti-image).



L'objectif de ce cours est de vous présenter les concepts du traitement numérique du signal. Pour ce faire, la première étape consiste à expliquer qu'est-ce qu'un signal et comment le traiter. Dans la suite de ce chapitre, nous présentons brièvement les caractéristiques d'un processeur de traitement numérique du signal connu aussi, sous le nom DSP pour "Digital signal Processor".

Différents types de signaux

Dans un système électronique, il existe divers capteurs (tels que des microphones, des caméras et des capteurs de pression ...) qui transforment une quantité physique sous forme électronique. Si une sortie de capteur change avec le temps, nous l'appelons un ***signal***. Plus généralement, on peut définir un signal comme une donnée qui évolue par rapport à une variable dépendante. Un signal peut être traité par un système soit pour en obtenir des informations, soit pour le modifier.

Signaux à temps continu

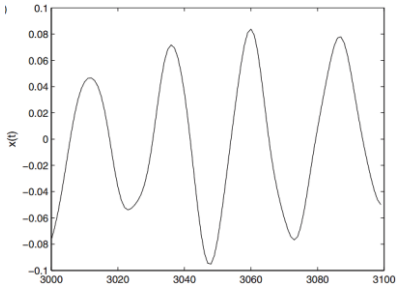
Si un signal acquis est sous forme analogique, nous l'appelons un signal à temps continu. Ce signal peut être traité ultérieurement par un système à temps continu composé des dispositifs analogiques.

Signaux à temps discret

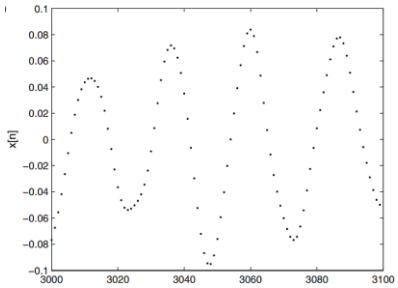
Bien que le traitement d'un signal à temps continu avec un système à temps continu puisse sembler raisonnable, ce n'est pas le cas pour la plupart des applications. Une autre méthode consiste à échantillonner le signal. Cela correspond à un signal à temps discret.

Signaux numériques

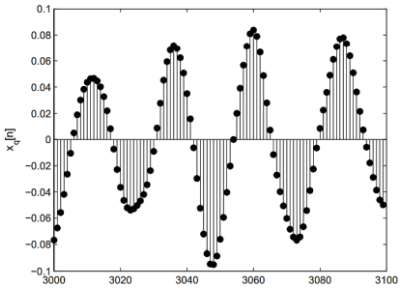
Dans un système numérique (tel qu'un microcontrôleur), un signal à temps discret peut être représenté sous la forme d'un tableau, où chaque échantillon du signal est une entrée du tableau. Ce tableau ne peut stocker que des valeurs avec une plage et une résolution limitée. Chaque entrée du tableau ne peut être représentée que par un certain nombre de bits. Par conséquent, l'amplitude du signal à temps discret doit être quantifiée. Le signal résultant est appelé signal numérique.



Signal à temps continu



Signal à temps discret



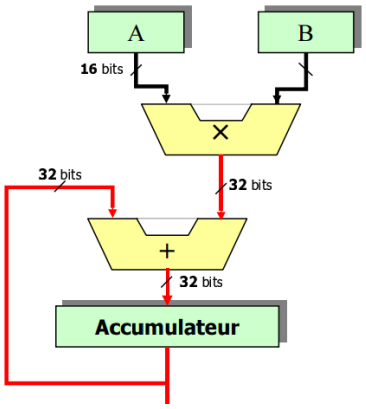
Signal numérique

2 Digital Signal Processor

Contrairement aux microprocesseurs et aux microcontrôleurs, les processeurs de traitement numérique du signal sont des microprocesseurs spécialement conçus pour gérer les tâches de traitement numérique du signal (la transformée de Fourier rapide, le filtrage, la convolution et la corrélation ...). Ces dispositifs ont connu une croissance considérable au cours de ces dernières décennies, trouvant une utilisation partout, partant des téléphones cellulaires aux instruments scientifiques avancés. En fait, les ingénieurs en matériel utilisent l'acronyme DSP pour désigner "Digital Signal Processor", tout comme les développeurs d'algorithmes utilisent le même acronyme pour "Digital Signal Processing".

Pour accélérer la vitesse d'exécution du traitement du signal numérique, les processeurs DS sont conçus sur la base de l'architecture Harvard, qui sépare les bus de la mémoire programme et de données. L'accès simultané au programme et aux données permet de paralléliser les opérations, ainsi, trois lectures et une écriture en mémoire peuvent être effectuées en un cycle machine. Cela permet le stockage du résultat parallèlement à une opération de calcul.

Outre l'unité arithmétique et logique (UAL), le DSP dispose d'une unité MAC. Il s'agit d'un matériel dédié et l'instruction correspondante est généralement appelée opération MAC (Multiply And Accumulate). Comme le montre la figure ci-dessous, dans un MAC matériel typique, le multiplicateur a une paire de registres d'entrée, chacun contenant l'entrée 16 bits du multiplicateur. Le résultat de la multiplication est accumulé dans une unité d'accumulation de 32 bits. Le registre du résultat contient les données en double précision de l'accumulateur.



3 VIRGULE FIXE ET VIRGULE FLOTTANTE

Afin de traiter les données du monde réel, nous devons sélectionner un DSP approprié. Le fait qu'un DSP utilise une méthode à virgule fixe ou flottante dépend de la façon dont la CPU du processeur effectue l'arithmétique. Un DSP à virgule fixe représente les données au format entier complément à 2 et manipule les données à l'aide de l'arithmétique des entiers, tandis qu'un processeur à virgule flottante représente les nombres à l'aide d'une mantisse (partie fractionnaire) et d'un exposant. Etant donné que le DSP à virgule fixe fonctionne en utilisant le format entier, qui ne représente qu'une plage dynamique très étroite du nombre entier, un problème tel qu'un débordement lors de manipulation de données peut se produire. Par conséquent, nous devons consacrer beaucoup plus d'efforts de codage pour faire face à un tel problème. Tandis que le DSP à virgule flottante offrent une plage dynamique de données plus large, de sorte que le codage devient beaucoup plus facile. Cependant, le DSP à virgule flottante contient plus d'unités matérielles pour gérer l'arithmétique des nombres entiers et l'arithmétique à virgule flottante, il est donc plus coûteux et plus lent que les processeurs à virgule fixe en termes de cycles d'instructions.

3.1 Représentation en virgule fixe

Dans un calculateur, un entier est représenté par un nombre binaire de longueur fixe. Un nombre binaire de longueur n est représenté par $n-uplet(x_{n-1}, x_{n-2}, \cdots\cdots, x_1, x_0)$. Chaque digit (ou bit) x_i peut prendre les valeurs 0 ou 1.

La séquence ci-dessous de n digits représente la valeur entière X:

$$X = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots\cdots +, x_1 2 + x_0 = \sum_{i=0}^{n-1} x_i 2^i$$

$r = 2$: base binaire.

Les entiers sont représentés en complément à 2, le digit x_{n-1} représente le bit de signe (0 : positif, 1 : négatif).

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

Dans ce cas le domaine de définition du codage : $D = [-2^{n-1}, 2^{n-1} - ulp]$

Le pas de quantification du codage "unit in the last position" $ulp = 2^0 = 1$

Les réels codés en virgule fixe sont généralement représentés en complément à deux. La séquence comporte une partie entière est une partie fractionnaire. Cela se fait en partitionnant les n digits en deux ensembles : k digits pour la partie entière et m digits pour la partie fractionnaire, vérifiant $n = k + m$.

$$(\underbrace{x_{n-1}2^{k-1} + x_{n-2}2^{k-2} + \cdots\cdots +, x_1 2 + x_0}_{partie\ entière} \circ \underbrace{x_{-1}2^{-1} + x_{-2}2^{-2} + \cdots\cdots +, x_{-m}2^{-m}}_{partie\ fractionnaire})$$

Soit

$$X = -x_{n-1}2^{n-1} + \sum_{i=-m}^{n-2} x_i 2^i$$

x_{n-1} représente le bit de signe.

$D = [-2^{n-1}, 2^{n-1} - ulp]$ avec $ulp = 2^{-m}$

En complément à 2, un nombre positif est représenté de la même manière pour les nombres signés ; alors qu'un nombre négatif $-Y$ est représenté par $R - Y$ avec $R = r^k$.

Le complément à 2 de $(R - Y) = R - (R - Y) = Y$.

Pour une opération de soustraction :

$$X - Y = X + (R - Y) = R - (Y - X)$$

Si $Y > X$; $-(Y - X)$ est représenté en complément à 2

Si $X > Y$; $(X - Y) = X + (R - Y) = R + (X - Y)$, le terme supplémentaire R doit être éliminé.

On définit le digit $\bar{x}_i$ le complément simple de x_i :

$$\bar{x}_i = (r - 1) - x_i$$

Notons $\bar{X}$ le $n - uplet(\bar{x}_{k-1}, \bar{x}_{k-2}, \cdots\cdots, \bar{x}_{-m})$, obtenue après la complémentation de chaque digit de la séquence correspondant à X. Faisons la somme de $\bar{X}$ et X. Nous obtenons toujours $\bar{x}_i + x_i = (r - 1)$, indépendamment de la valeur exacte de x_i. Ajoutons alors ulp à la somme de $\bar{X}$ et X.

X	x_{k-1}	x_{k-2}	$\cdots$	x_{-m}	
$+\overline{X}$	$\bar{x}_{k-1}$	$\bar{x}_{k-2}$	$\cdots$	$\bar{x}_{-m}$	
	$(r-1)$	$(r-1)$	$\cdots$	$(r-1)$	
$+ulp$				1	
1	0	0	$\cdots$	0	$= r^k$

On peut écrire donc :

$$X + \bar{X} + ulp = r^k$$

Le résultat est stocké dans un registre de longueur $n = k + m$. Le bit MSB est éliminé et le résultat final est bien 0.

En général, stocker le résultat d'une opération arithmétique dans un registre de longueur fixe, est équivalent à prendre le reste après division par r^k.

En réarrangeant l'écriture :

$$r^k - X = \bar{X} + ulp = R - X = \bar{X} + ulp$$

Le complément à 2 de $R - X = R - (R - X) = X$

Exemple :

Exécutons l'opération $X + (-Y)$ avec $Y > X$. Prenons $X = 2$ et $Y = 7$.

	0	0	1	0	2
+	1	0	0	1	−7
	1	0	1	1	−5

Le résultat correct est donné en complément à 2.

Prenons le cas $X > Y$. Prenons $X = 7$ et $Y = 2$.

	0	1	1	1	7
+	1	1	1	0	−2
1	0	1	0	1	5

Uniquement les quatre bits de poids faible sont retenus, le bit MSB correspondant à R est ignoré.

Il est possible d'étendre un nombre binaire codé en complément à 2 en ajoutant à gauche le bit de signe et à droite des 0.

$$\cdots, x_{n-1}, x_{n-1}, \{x_{n-1}, \cdots \cdots, x_0\}, 0, 0 \cdots$$

3.1.1 Addition et soustraction

En complément à 2, tous les digits, inclus le bit de signe participent à l'opération s'addition ou de soustraction. Le débordement est géré par le biais des indicateurs OV (OVerflow) et le bit de retenue C (Carry-out).

$$OV = Carry_{out} \oplus Carry_{in}$$

$Carry_{in}$: retenue du bit de signe.

En général, si X et Y sont de signe opposé, il n'y a pas de débordement (overflow) et la retenue sera ignorée. Exemple :

	0	0	1	0	1	5
+	1	0	1	1	0	−10
	1	1	0	1	1	−5

	0	1	0	1	0	10
+	1	1	0	1	1	−5
1	0	0	1	0	1	5

Si X et Y sont de même signe, il peut y avoir un débordement. Dans ce cas, il faut étendre le résultat sur $n+1$ bits. La valeur de C représente le nouveau bit de signe. Exemple :

	1	1	0	0	1	−7
+	1	0	1	1	0	−10
1	0	1	1	1	1	15

	0	0	1	1	1	7
+	0	1	0	1	0	10
	1	0	0	0	1	−15

3.1.2 Multiplication

Soit le multiplicande et le multiplicateur dénoter par A et X, ayant respectivement les séquences des bits suivantes :

$$A = a_{n-1}\, a_{n-2} \cdots a_1\, a_0 \quad \text{et} \quad X = x_{n-1}\, x_{n-2} \cdots x_1\, x_0$$

Où a_{n-1} et x_{n-1} sont les bits de signe.

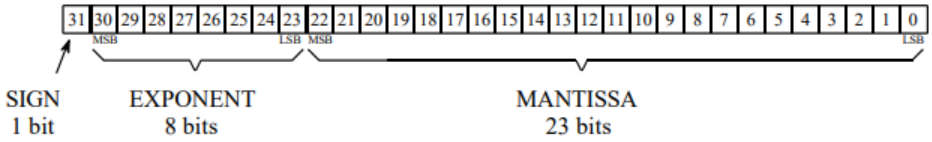
$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

$$A \cdot X = A \cdot \sum_{i=0}^{n-2} x_i 2^i - A x_{n-1} 2^{n-1}$$

3.2 Représentation en virgule flottante

Pour augmenter la plage dynamique de représentation des nombres, un format à virgule flottante, similaire à la notation scientifique est utilisé. Bien que plusieurs formats similaires soient utilisés, le plus courant est ANSI/IEEE Std. 754-1985. Cette norme définit le format des nombres 32 bits appelés simple précision, ainsi que des nombres 64 bits appelés double précision. Comme le montre la figure suivante, les 32 bits utilisés

en simple précision sont divisés en trois groupes distincts : les bits 0 à 22 forment la mantisse, les bits 23 à 30 forment l'exposant et le bit 31 est le bit de signe.



Ces bits forment le nombre à virgule flottante, F, par la relation suivante :

$$F = (-1)^S \times M \times 2^{E-127}$$

Le terme $(-1)^S$ signifie que le bit de signe, S, est 0 pour un nombre positif et 1 pour un nombre négatif.

L'exposant E est un nombre compris entre 0 et 255. En soustrayant 127 de ce nombre, le terme en exposant peut courir de -127 à +128.

La mantisse M formée de 23 bits est la fraction binaire. Le premier bit est toujours 1, ce bit est caché, puisqu'il ne fait pas partie des 23 bits de la mantisse.

$$M = 1.f = 1.\underbrace{m_{22}\, m_{21} \cdots \cdots m_0}_{23\ bits} = 1 + m_{22}2^{-1} + m_{21}2^{-2} + \cdots \cdots m_0 2^{-23}$$

En dehors de 256 combinaisons du champ exposant, deux sont réservées pour des valeurs spéciales :

$E = 0$ est réservé pour Zéro (avec la partie fractionnaire $f = 0$) et dénormalisé (avec $f \neq 0$).

$E = 225$ est réservé pour $\pm\infty$ (avec $f = 0$) et "Not a Number" NaN (avec $f \neq 0$).

	$f = 0$	$f \neq 0$
$E = 0$	0	Denormalized
$E = 255$	$\pm\infty$	NaN

Pour les exposants restants (c-à-d $1 \leq E \leq 254$), la valeur du nombre à virgule flottante est donnée par :

$$F = (-1)^S \times 1.f \times 2^{E-127}$$

En utilisant ce schéma de codage, le plus grand nombre pouvant être représenté est $\pm(2 - 2^{-23}) \cdot 2^{127} = \pm 3{,}4 \cdot 10^{38}$. De même, le plus petit nombre pouvant être $\pm 1.0 \cdot 2^{-126} = \pm 1{,}2 \cdot 10^{-38}$.

3.2.1 Opération virgule flottante

Soit X et Y et Z des nombres code en virgule flottante représentés par (S_x, M_x, E_x), (S_y, M_y, E_y) et (S_z, M_z, E_z).

Addition/Soustraction

Soit $Z = X \pm Y$

$$Z = \begin{cases} ((-1)^{S_x} M_x \pm (-1)^{S_y} M_y 2^{-(E_x - E_x)}) 2^{E_x - 127} & si\ E_x \geq E_y \\ ((-1)^{S_x} M_x 2^{-(E_y - E_x)} \pm (-1)^{S_y} M_y) 2^{E_y - 127} & si\ E_x < E_y \end{cases}$$

L'opération d'addition ou de soustraction est résumée dans les étapes suivantes :

$Etape\ 1$: Calculer la différence d de deux opérandes $d = |E_x - E_y|$.

$Etape\ 2$: Décaler la mantisse du plus petit nombre de d position à droite.

$Etape\ 3$: additionner ou soustraire les mantisses et prendre pour le résultat l'exposant le plus grand.

$Etape\ 4$: Normaliser le résultat si nécessaire.

Multiplication

$$Z = X \cdot Y = \left((-1)^{S_x} M_x \cdot (-1)^{S_y} M_y\right) 2^{E_x + E_y - 127}$$

Ceci implique les étapes suivantes :

$Etape\ 1$: Si l'un des opérandes ou le deux égale à Zéro, retourner un résultat nul ; sinon :

$Etape\ 2$: Calculer le signe du résultat $S_z = S_x\ xor\ S_y$.

$Etape\ 3$: Calculer la mantisse $M_z = M_x \cdot M_y$

$Etape\ 4$: Calculer l'exposant du résultat : $E_z = E_x + E_y - 127$.

$Etape\ 5$: Normaliser le résultat.