TP N°4 : IMPLEMENTATION D'UN FILTRE FIR SUR UN DSP A POINT FLOTTANT (TMS320C6711)

I- BUT

- Test des outils logiciels et matériels avec le Code Composer Studio.
- Se familiariser avec des applications sur la carte TMS320C6711 DSK.
- Implémentation d'un filtre numérique sur un DSP.

II- Rappels théoriques

Les filtres numériques présentent par rapport aux filtres analogiques les avantages-inconvénients suivants

Avantages :

- Les filtres numériques sont insensibles aux conditions extérieures (chaleur, humidité, etc.)
- Certains filtres numériques sont impossibles à réaliser de manière analogique (exemple : les filtres RIF).
- Les filtres numériques ne sont pas sensibles aux non idéalités d'un 10ème amplificateur opérationnel. Ainsi, un filtre RII du 10 ordre est tout à fait envisageable (attention quand même au bruit de calcul !).
- La problématique du bruit change d'aspect : dans le filtrage numérique on parle de 'bruit de quantification' et de 'bruit de calcul'. Le premier est lié au nombre de bits employés pour la quantification (8 bits, 16 bits, etc.). Le second est négligeable si l'unité de calcul est de type 'floating point'. De toute manière, le bruit numérique est localisé : on sait d'où il vient et il reste stable.

Inconvénients :

- Les filtres numériques nécessitent un filtrage analogique anti-repliement à l'échantillonnage et à la restitution.
- Les performances d'un filtre sont directement proportionnelles à la puissance de l'unité de calcul (processeur ou DSP).
- Beaucoup de problèmes peuvent apparaître si l'unité de calcul est de type 'fixed point'. Les paramètres d'un filtre nécessitent parfois une double précision pour être opérationnels, ce qui ralentit les performances.

On trouve essentiellement deux grandes familles de filtres numériques :

A. Filtres RIF : Ces filtres non récursifs n'ont pas de contre-réaction.

Avantages :

- ✓ Toujours stables.

- ✓ Phase linéaire si symétrie des coefficients ⇒ pas de distorsion de phase.
- ✓ Possibilité de réaliser toutes sortes de filtres en dessinant simplement des gabarits de réponse

en amplitude H(f) et en calculant la transformée de Fourier inverse h(t).

Inconvénients :

- Beaucoup de calculs par rapport à un RII équivalent au niveau des performances.
- Le retard du filtre (de groupe ou de phase) peut être important si le nombre de 'taps' est élevé.

B. Filtres RII : Ces filtres récursifs ont une contre-réaction.

Avantage :

- Beaucoup moins de calculs par rapport à un RIF équivalent au niveau des performances.

Inconvénients :

- Il faut vérifier la stabilité.
- Phase non linéaire ⇒ distorsion de phase.

C- Equation aux différences d'un filtre à réponse impulsionnelle finie

La sortie est une combinaison d'un ensemble fini d'éléments d'entrées :

$$y(n)= \sum_{k=0}^{N-1} h(k)x(n-k) \qquad |h(n)| < \infty,$$

Où, h(n) est la réponse impulsionnelle du filtre et N et sa longueur.

Le gabarit réel d'un filtre FIR est le suivant,

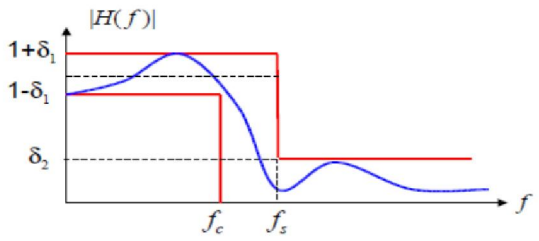

Figure 2.1. : Gabarit réel de la réponse fréquentielle d'un filtre FIR.

Le filtre est caractérisé par :

- La bande passante BP.
- La bande atténuée (ou coupée).
- La largeur ΔF= fs - fc de la zone de transition.
- L'amplitude des oscillations en bande passante δ_1.
- L'amplitude des ondulations en bande atténuée δ_2.

Pour calculer les coefficients de la réponse impulsionnelle, on a recours à deux méthodes : la première est celle du calcul manuel, qui demande beaucoup de temps, et que nous ne traitons pas dans ce travail, tandis que la deuxième méthode est celle du calcul à l'aide de l'outil 'fdatool' de Matlab.

III- Préparation

Taper dans la fenêtre de commande Matlab 'fdatool'. Cette commande permet d'ouvrir la fenêtre suivante,

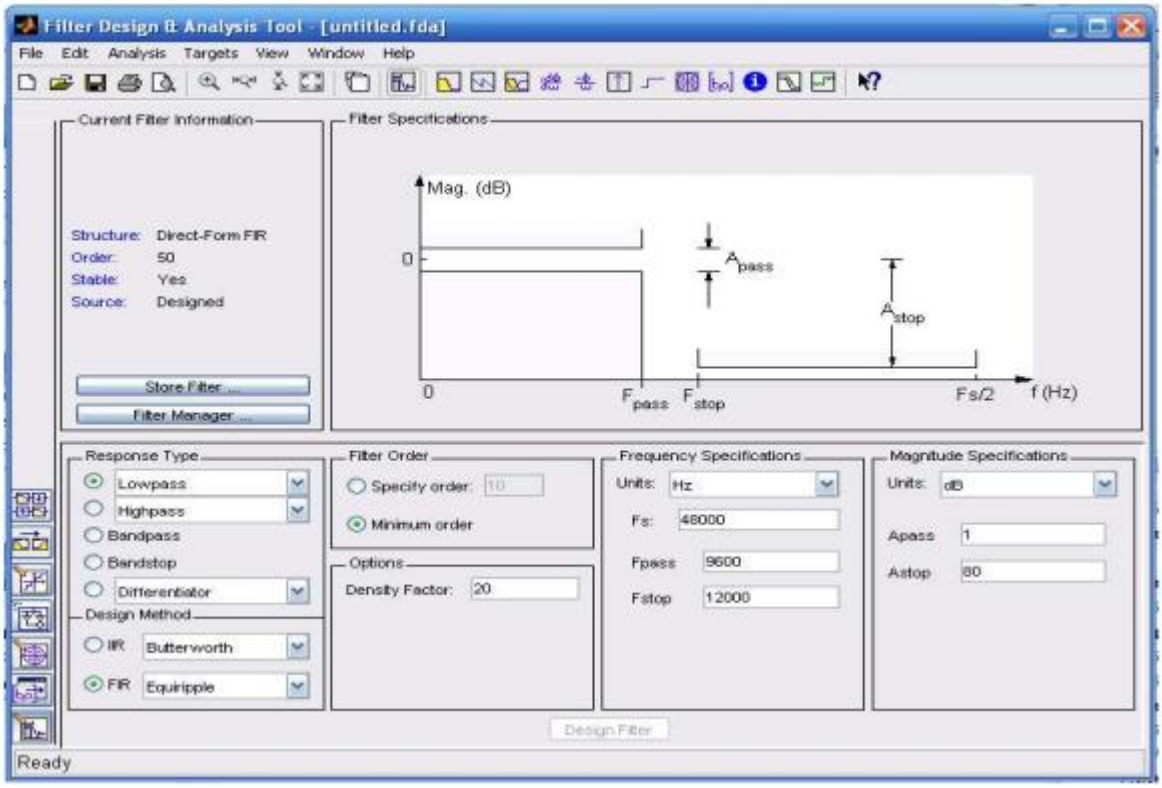

Figure 2.2. Fenêtre 'fdatool' de MATLAB.

A ce niveau, nous précisons le type, les caractéristiques (citées ci-dessus), la méthode de conception du filtre,… Sélectionner ensuite « Design Filter », puis (Target Generate C Header…) pour générer les coefficients de la réponse impulsionnelle dans un fichier en-tête.

Implémentation d'un filtre FIR passe-bas

Dans ce travail nous voulions implémenter un filtre FIR passe-bas de fréquence de coupure **F_c=1500 Hz**.

Algorithme

Pour implémenter ce filtre FIR sur le DSP, il faut réserver une partie de la mémoire pour stocker les N coefficients de la réponse impulsionnelle, une autre partie sera réservée pour les échantillons de l'entrée. Cette

dernière devra mettre à jour à chaque réception d'une nouvelle donnée x(n). En effet, chaque nouvel échantillon doit écraser l'avant dernier en gardant la même taille du tableau comme le montre le schéma suivant :

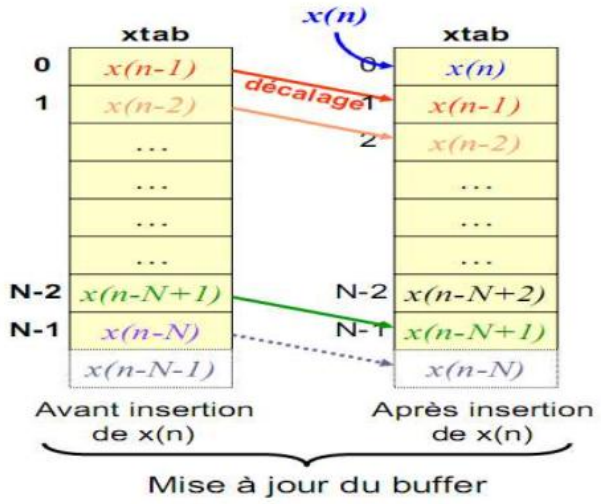

Figure 2.3. Mémoire de donnée pour un filtre FIR.

IV- Matériels utilisés

- Un starter kit DSK320C6711
- Un GBF GFG8255A.
- Un oscilloscope numérique de marque GDS 840S ayant une bande passante de 250 MHz.
- Des sondes pour oscilloscope.
- Un PC
- Des câbles de connexion.
- Deux fiches jacks

V- Manipulation

a)

1- Créer un projet en CCS et nommer le comme étant « filtre_fir.pjt ».
2- Joindre au programme principal en langage C « filtre_fir.c », les fichiers suivants c6713dskinit.c, Vectors_intr.asm, c6711dsk.cmd, rts6700.lib.
3- Construire le projet en allant au Project et ensuite en cliquant sur Rebuild All).
4- Après la construction, cela génère le fichier exécutable 'filtre_fir.out' et finalement charger ce fichier exécutable sur le DSP (File Load).

b) La création d'un script MATLAB pour la génération d'une somme de deux signaux sinusoïdaux, la première sinusoïde a une fréquence de 1 kHz et la deuxième de 2 kHz (le signal qui sera filtré)

c) Relier l'entrée 'LINE IN' du DSK avec le PC à l'aide d'un câble JACK.

d) Relier la sortie de la carte à un oscilloscope pour visualiser le signal de sortie.

e) Lancer le programme.

Afin de s'assurer de la validité de la réponse du filtre implémenté sur le DSP, nous allons faire une comparaison entre sa réponse et celle obtenue théoriquement avec l'outil 'fdatool' de MATLAB.

- Attaquer la carte par un signal sinusoïdal à partir d'un générateur GBF dont la fréquence varie autour de la fréquence de coupure.

- Faire varier la fréquence du signal d'entrée et relever l'amplitude du signal de sortie selon le tableau ci-dessous

f (Hz)	400	700	1020	1120	1430	1480	1500	1520	1540	1560
V_e (volt)	2	2	2	2	2	2	2	2	2	2
V_s (Volt)										
V_s / V_e										

f (Hz)	1580	1590	1610	1620	1660	1680	1700	1720	1740	1760
V_e (volt)	2	2	2	2	2	2	2	2	2	2
V_s (Volt)										
V_s/ V_e										

f (Hz)	1780	1800	1860	1900	2000	2100	2200	2300	2400	2500
V_e (volt)	2	2	2	2	2	2	2	2	2	2
V_s (Volt)										
V_s/ V_e										

Tableau 2.1. La variation du rapport Vs/ Ve en fonction de la fréquence du GBF

Avec, f = Fréquence, Ve = La tension d'entrée et Vs= La tension de sortie

- Tracer la réponse réelle V_S / Ve du filtre implémenté sur le DSP.
- Comparer les résultats théoriques obtenus partir de MATLAB et ceux trouvés pratiquement.

- Quelle conclusion peut-on en déduire.

Programme principal en langage C du filtre FIR passe-bas de fréquence de coupure 1500Hz.

```
#include "dsk6711_aic23.h" // fichier support pour le codec et le DSK
#define Nbr_coeff 81
Uint32 fs=DSK6711_AIC23_FREQ_8KHZ ; // fréquence d'échantillonnage
int sorie = 0; //initialiser la sortie du filtre
short h[Nbr_coeff]=
{
-6, 13, 42, 52, 14, -40, -45, 18, 74, 35, -70, -100, 12, 137, 89, -105, -192, -14,
228, 188, -142, -338,-76,361, 361, -179, -579, -208, 569, 685, -209, -1040, 519, 992,
1496,
-229, -2492, -1827, 3237, 9832, 12872, 9832, 3237, -1827, -2492, -229, 1496, 992, -
519,
-1040, -209, 685, 569, -208, -579, -179, 361, 361, -76, -338, -142,188,228, 14, -192, -
105, 89, 137, 12, -100, -70, 35, 74, 18, -45, -40, 14, 52, 42, 13, -6};
short X_retard[Nbr_coeff]; //échantillons retardés
interrupt void c_int11() //ISR
{
short i;
X_retard [0]=input_sample(); //le nouvel échantillon d'entrée
sortie = 0; // initialiser la sortie du filtre
for (i = 0; i< N; i++)
sortie =sortie+ (h[i] * X_retard [i]); //sortie(n) += h(i)* x(n-i)
for (i = Nbr_coeff-1; i > 0; i--)
X_retard [i] = X_retard [i-1]; // déplacer les échantillons retardés
output_sample(sortie >> 15); //échantillon de sortie du filtre en format short
return;
}
void main()
{
comm_intr(); //init DSK, codec, McBSP
while(1); //infinite loop43
}
```

Script MATLAB pour la génération de la somme de deux signaux sinusoïdaux de fréquences 1kHz et 2kHz

```
F1=1000; % fréquence du premier signal
F2=2000; % fréquence du deuxième signal
fe=8000; % fréquence de l'échantillonnage
N= 2^16; % Nombre d'échantillons
temps=(0:N-1)/fe; %axe temporel en seconde
x1=sin(2*pi*F1*temps); %premier signal
x2=sin(2*pi*F2*temps); %deuxième signal
for u=1:100,
sound(x1+x2) ; %écouter le mélange des deux fréquences
end
```