

## TP6 : Récupération des coefficients du filtre (à faire avant le TP)

Ouvrir avec Matlab/simulink le fichier `filtre_cos_2019.slx`. Parcourir les différents blocs : observer que sur le générateur de Bernoulli, le sample time est de  $0.2 \mu\text{s}$  (soit un débit de 5 Mb/s), avec 2 bits par frame.

Dans le bloc **QPSK Baseband Modulator**, observer la constellation, et vérifier que I et Q valent  $\pm\sqrt{2}/2$ .

On rappelle la commande permettant d'obtenir les coefficients du filtre en cosinus surélevé : `rcosdesign( $\alpha$ , Span, OSR, Shape)`, où  $\alpha$  est le facteur de roll-off; **Span** le nombre de temps-symbole sur lequel s'étale la réponse impulsionnelle (forcément pair, typiquement 6 à 12); **OSR** le facteur de sur-échantillonnage; **Shape** une chaîne de caractères parmi 'normal' et 'sqrt'.


Ici, vérifier :

- que le taux de suréchantillonnage (nombre d'échantillons par symbole en sortie du filtre ou OSR : Over sampling Ratio) vaut 4. Quel est le sample rate en sortie du filtre ?
- que la réponse impulsionnelle s'étale sur 8 symboles, soit  $4 \times 4 = 16$  échantillons de chaque côté du maximum, et 33 en tout. On rappelle que pour un filtre FIR, les échantillons de la réponse impulsionnelle sont les coefficients du filtre.

Pour connaître les valeurs numériques des coefs, taper dans la fenêtre de commande Matlab : `rcTxFilt=rcosdesign( $\alpha$ , Span, OSR, Shape)`

On peut aussi les visualiser : `fvtool(rcTxFilt)`

et cliquer sur . Noter la valeur maximale approximative atteinte en valeur absolue.

Enfin, exécuter le modèle; observer le diagramme de l'oeil et le signal de sortie du filtre en cliquant sur  (dans la barre d'outils). Noter la valeur maximale atteinte par le signal (utiliser le diagramme de l'oeil pour cela).

En pratique, la sortie du filtre sera l'entrée d'un DAC 14 bits. C'est pourquoi elle doit être représentée sous forme d'un *entier positif, codé sur 14 bits*, donc compris entre 0 et  $2^{14} - 1 = 16383$ .

Lors de la description de notre filtre en VHDL, il nous faut, en sortie, passer de nombres à virgule tels que ceux utilisés par Matlab à des entiers sur 14 bits. Deux solutions :

1. on utilise pour la description VHDL du filtre des nombres à virgule fixe en utilisant un package spécifique. Puis, on transforme chaque échantillon de sortie du filtre en un entier sur 14 bits ;
2. on utilise pour la description VHDL du filtre des entiers. Les types et les nombres de bits doivent être choisis judicieusement.

La première solution est inapplicable ici car le package nécessaire n'est pas supporté par la version gratuite de Quartus; nous utiliserons donc la seconde.

Nous configurerons le bloc de mapping (l'équivalent du **QPSK Baseband Modulator** de Matlab) afin que les valeurs de  $I$  (échantillons d'entrée du filtre) soient des entiers : 1 ou -1, soit un gain de  $\sqrt{2}$  par rapport aux valeurs utilisées par défaut dans le bloc Simulink. La sortie du filtre est dans

ce cas comprise dans l'intervalle  $[-0.86, 0.86]$ . Pour obtenir un entier sur 14 bits en sortie (compris entre  $-2^{13}$  et  $2^{13} - 1$  car il s'agit d'entiers signés), on multipliera chaque coefficient par

$$\frac{2^{13} - 1}{0,86}$$

et on arrondira à l'entier inférieur. La commande Matlab à utiliser est :

```
filter_coefs=floor(8191/0.86*rcTxFilt)
```

Noter les valeurs obtenues (sur une feuille ou dans un fichier texte).

Nous n'aurons pas besoin ici d'accéder individuellement aux bits et donc nous utiliserons pour les échantillons et les coefficient le type `integer`, en restreignant les variables ou signaux aux intervalles adéquats.

En sortie du filtre, nous obtiendrons des entiers compris entre  $-2^{13}$  et  $2^{13} - 1$ . Le passage à un entier situé dans un intervalle compatible avec le DAC, donc compris entre 0 et  $2^{14} - 1$ , se fera simplement en ajoutant  $2^{13}$ .