Rapport : Logique

CLEMENT Andy

I/Introduction

Dans notre projet de convertisseur analogique/numérique, la logique va traiter les différentes données reçues par les éléments du circuit (R2R, Sample & Hold, ect) et piloter tous les éléments pour que tous fonctionnent de façon synchronisée. Pour cela, l'architecte a défini plusieurs étapes que l'on doit réaliser. Tout d'abord, pour chaque bit du R2R, on doit effectuer les tâches suivantes :

- Forcer le bit de test à 1 (**MSB**)
- Remplacer la valeur du MSB par Vcomp (Sample & Hold)
- Mémoriser le résultat de la comparaison (0 ou 1)

Avant chaque cycle, tous les bits devront être initialisés à 0 et la valeur du Sample sauvegardée pour assurer une tension stable durant toute la conversion.

II/Compteur

On aura besoin de traiter 9 éléments à tour de rôle (8 pour chaque bit, 1 pour le Sample & Hold). Pour cela, nous aurons besoin d'un **LPM_counter** codé sur 4 bits avec un set à 1 pour compter de 1 à 10 (On exclut le 0 car le temps d'horloge n'est pas le même à cet état.). Pour bien décoder tous les états, on utilisera un **LPM_decode**. À chaque fois que l'on passe à 10 (state10), le compteur repart de 1.

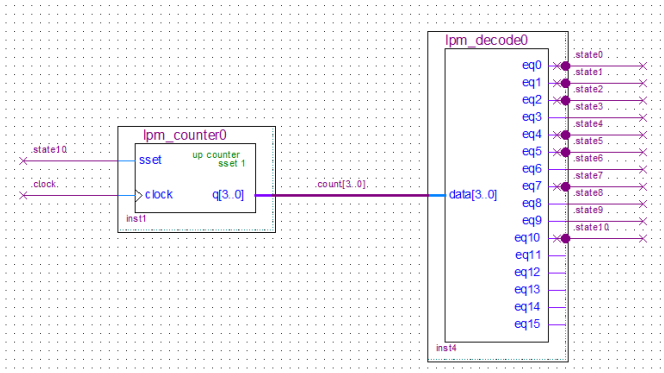

Figure 1: Schéma logique du compteur et du lpm_decode

On obtient alors ce chronogramme en simulation Quartus.

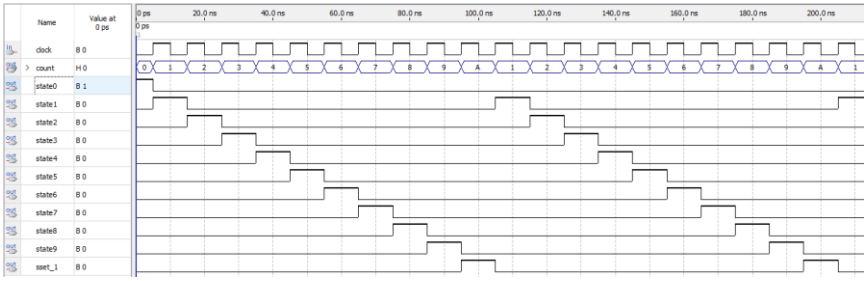

Figure 2: Chronogramme du compteur

On remarque qu'à l'état 0 le temps d'horloge est différent et que par la suite le compteur retourne bien à 1 après l'état 10.

III/Mémorisation

Pour mémoriser la valeur Vcomp sur chaque bit, on utilise la méthode du **Data gating**. Les composants utilisés sont alors :

- Un Multiplexeur (**21MUX**) : sélectionne la valeur de retour de la bascule ou la valeur Vcomp en fonction de la valeur de state.
- Une Bascule D (**LPM_FF**) : mémorise l'état 1 ou 0 du bit grâce à un rebouclage.
- Une Porte Or : Transmets la valeur enregistrée au reste du circuit.

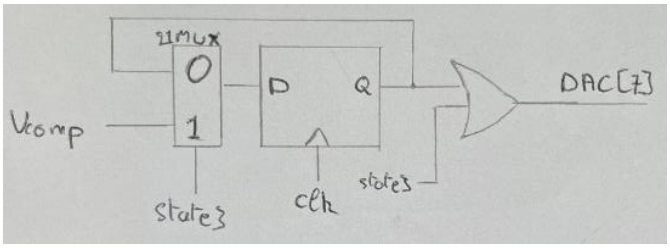

Figure 3: Schéma logique du Data gating

Avec cette disposition si state3 = 1 alors on enregistre la valeur Vcomp, sinon on mémorise. On a alors ce dispositif relié à chaque bit du convertisseur.

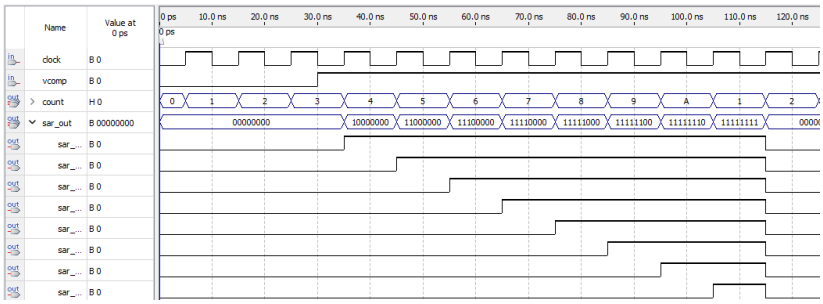

Figure 4: Chronogramme obtenu après le data gating

En regardant le chronogramme, on se rend compte que lorsque Vcomp est à l'état haut, les bits de sorties (**sar_out[7..0]**) sont enregistrés à l'état haut. Une fois arrivé à A le compteur se remet à 1 et les sorties sar_out[7..0] sont réinitialisées 1 coup d'horloge plus tard.

IV/Transmission du signal

On rappelle que pour envoyer les données nous ne disposons que d'un seul fil, c'est pourquoi chaque bit doit être envoyé 1 par 1 (à chaque coup d'horloge) pour être réceptionné correctement. Pour envoyer la valeur des bits qui viennent d'être récupérés par le dispositif de Data Gating, on mémorise les états de chaque bit dans une autre **LPM_FF**, ceci permettra de libérer les bascules précédentes pour qu'elles puissent à nouveau enregistrer les états futurs. Les bits sortent sur la sortie **adc_out[7..0].**

Une fois enregistré, on doit transmettre les données en respectant le protocole **RS232** en générant un bit de début de séquence de transmission (état haut) un bit de start (état bas) la transmission sur 8 bits et enfin le bit de stop (état haut). La transmission se fait alors sur 10 coups d'horloge. Pour cela on utilise un multiplexeur **161MUX** qui sélectionne à chaque coup d'horloge le bit de données à envoyer.

En sortie du multiplexeur, on ajoute une **DFF** pour absorber les délais dû à la logique en condition réelle. En ajoutant cette bascule on assure que tous les bits soient envoyées au coup d'horloge souhaité. On a finalement le schéma suivant :

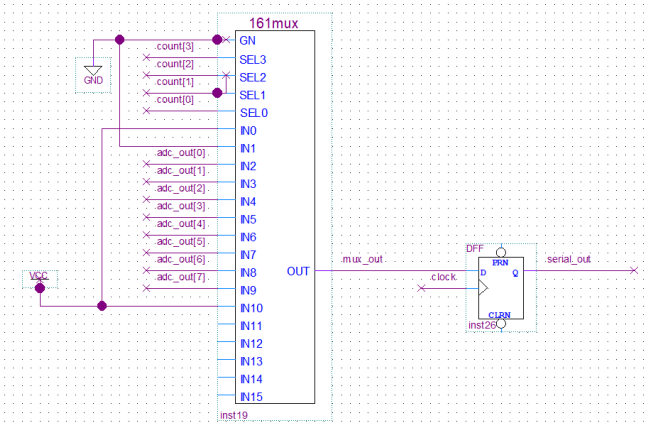

Figure 5: schéma logique du bloc de transmission

Cette configuration impose que l'envoi des données se fasse un coup d'horloge plus tard que la fin de la conversion. On remarque que la sortie **serial_out** est décalée d'un coup d'horloge par rapport à **mux_out**, à cause du **DFF**, on a alors la transmission qui commence en même temps que le Sample.

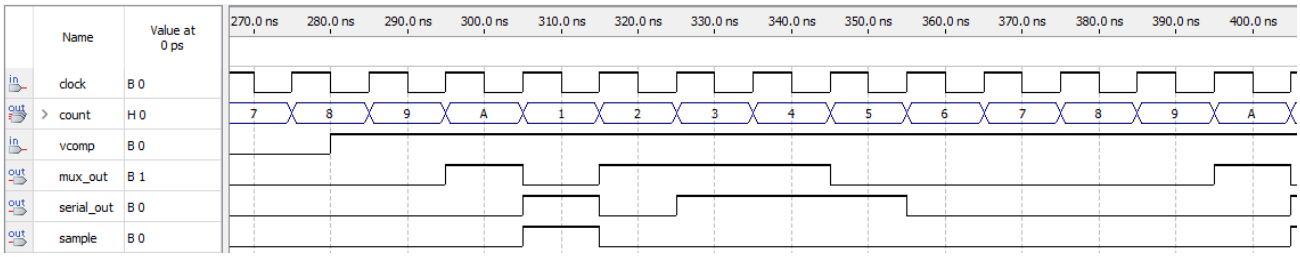

*Figure 6: Chronogramme de la transmission avec **mux_out** et **serial_out***

La logique fonctionnant bien, on programme la carte en techno **MAX7000S** et injectons la logique dans la carte. En testant le convertisseur dans la réalité, on a le comportement espéré.

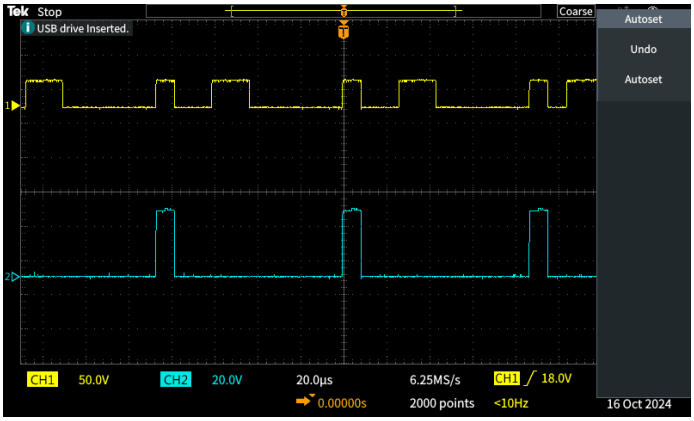

Figure 7: Affichage sur oscilloscope du signal obtenue

Le signal en jaune est **serial_out** et le signal en bleu est le signal venant du Sample. On remarque que le comportement des 2 signaux correspond à la simulation.