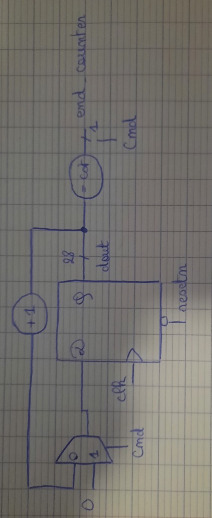
TP02 - Compteurs (Kamal KHERCHOUCH)

1. **L’horloge du système est fixée à 100MHz. Combien de période faut-il compter pour attendre 2 secondes ? Combien de bits faut-il au minimum pour représenter cette valeur ?**

On a F = 100 Mhz, on a donc une période à 10 ns.

Pour attendre 2 secondes il nous faut donc 200 000 000 périodes. Ce qui correspond à un minimum de 28 Bits.

1. **Dessinez le schéma RTL de ce compteur. Si le compteur atteint la valeur calculée précédemment, un signal *end\_counter* passe à 1, sinon *end\_counter* vaut 0. N’oubliez pas de mettre sur chaque signal son nombre de bits. Commencez par réaliser une boucle d’incrémentation : +1 à chaque coup d’horloge.**



1. **Ajoutez une condition pour que le compteur soit remis à 0 lorsqu’il a atteint la valeur souhaitée.**

Lorsque End\_Counter est à 1, Cmd passe à 1 et à l’aide du MUX on réinitialise le compteur.

1. **Listez les signaux d’entrée, de sortie et les signaux internes de votre architecture.**

Les signaux en entrée sont :

Clk : in std\_logic

Resetn : in std\_logic

Les signaux en sortie sont :

End\_Counter : Out std\_logic

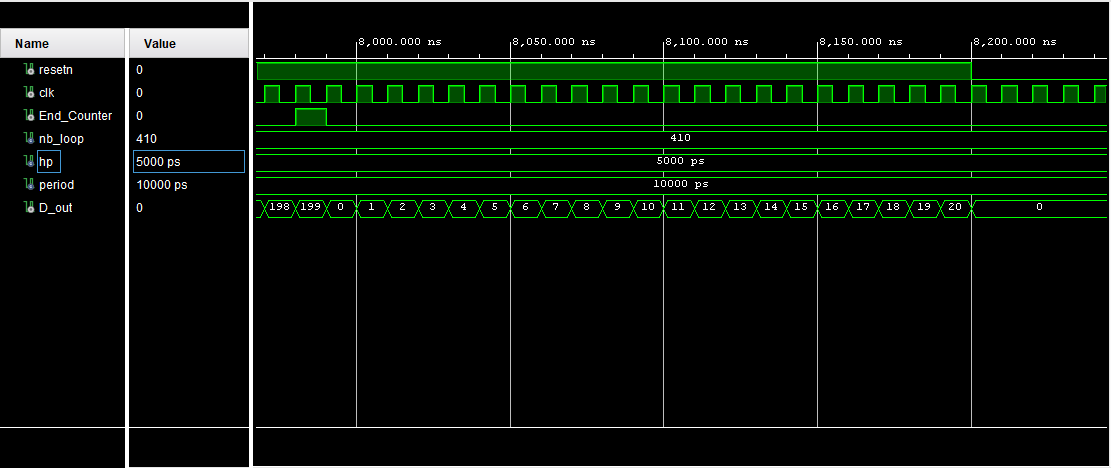
1. **Ecrivez à présent le compteur en VHDL en suivant le schéma RTL, faites attention de bien faire correspondre les noms des signaux de votre code VHDL avec ceux de votre schéma RTL.**

Voir fichier Counter.vhd (code complet du TP, donc prend en compte les questions suivantes)

1. **Ecrivez un fichier de testbench pour tester votre design.**

Voir le fichier tb\_coubter.vhd

1. **Lancez une simulation. Que devez-vous observez sur votre chronogramme pour vérifier que votre design est valide ?**

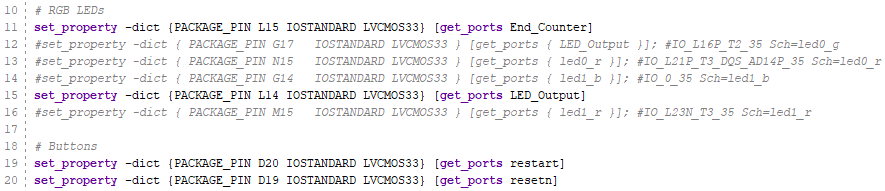


Pour les besoin du test, nous avons reduit la valeur de la constante à 200. Ici, on voit bien que lorsque D\_out est à 199 (car il commence à 0) on a le signal End\_counter qui passe à 1 le temps d’une période de l’horloge Clk.

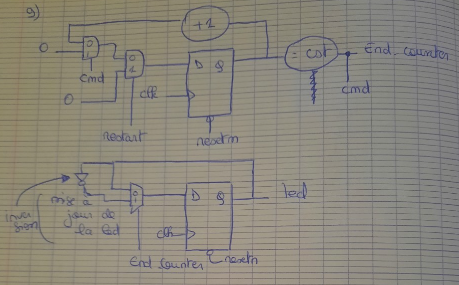
On valide également le fonctionnement du Reset car lorsque RST est à 0, le compteur D\_out passe automatiquement à 0.

1. **Associez une LED avec le signal de teste d’arrêt du compteur. Pour cela, il faudra ajouter une sortie et la relier à une broche d’une LED dans le fichier de contrainte (.*xdc*). La LED sera alors allumée pendant seulement un coup d’horloge.**

On à modifier le fichier de contrainte comme suit :



1. **Modifiez le schéma RTL du compteur pour ajouter une remise à 0 lorsqu’un signal *restart* est à 1. Ajoutez la logique nécessaire pour que la LED clignote telle que : allumée 2s, éteinte 2s.**



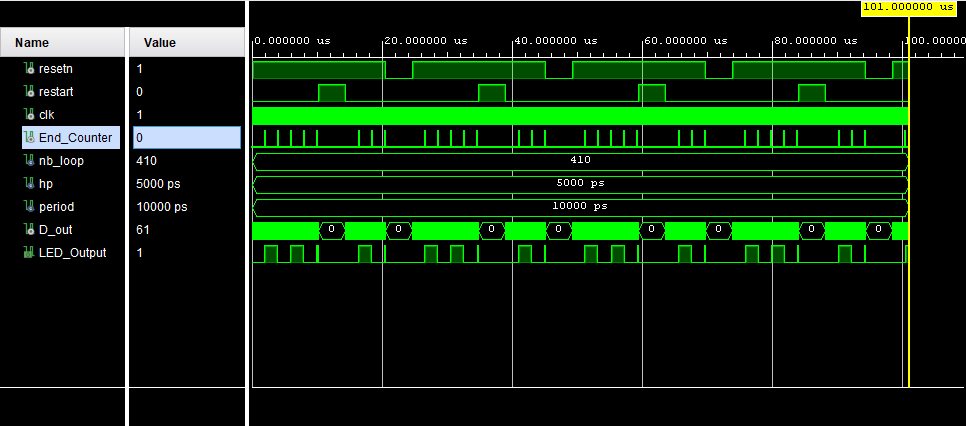
1. **Faites les mises à jour nécessaires sur le code VHDL pour correspondre au nouveau schéma. Le signal *restart* sera une entrée du design.**

Voir fichier Counter.vhd

1. **Associez la nouvelle entrée *restart* à un bouton.**

Présent dans la capture de la question 8.

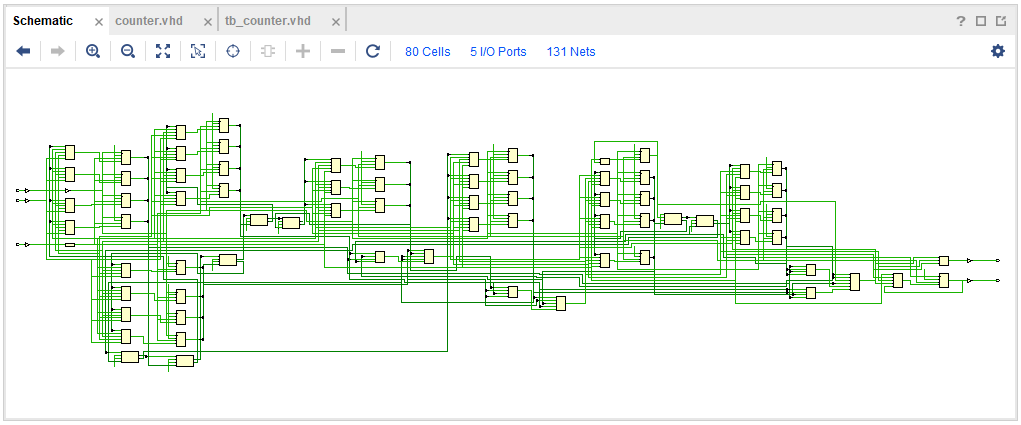
1. **Mettez à jour votre testbench puis vérifier votre design avec une simulation. Quels sont les signaux que vous devez observer ?**



Les signaux que l’on observe sont : Resetn, Restart, End\_counter et Led\_Out.

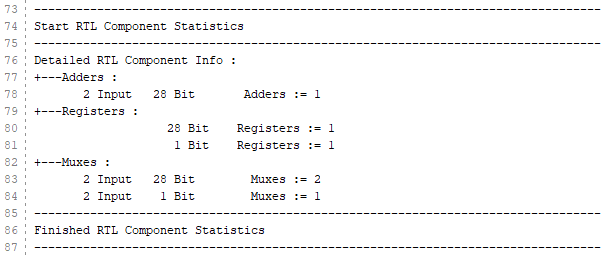
La simulation valide le fonctionnement du système.

1. **Exécutez la synthèse puis ouvrez la schématique. Identifiez sur la schématique les différents éléments de votre architecture RTL.**



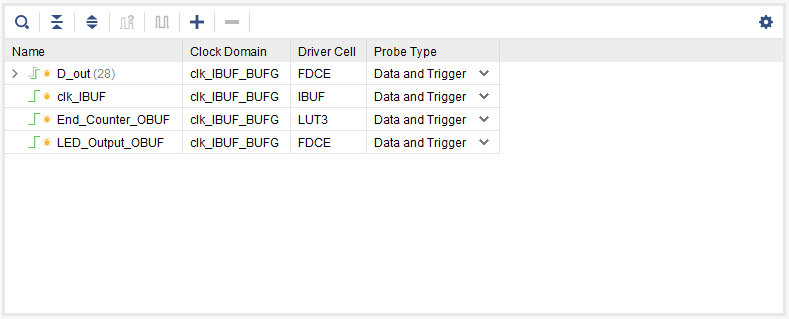
On a un registre par bit, donc 28 registres, A gauche nous retrouvons nos entrées et à droite nos sorties, on retrouve également quelques LUT pour la partie combinatoire.

1. **Ouvrez le rapport de synthèse et relevez les ressources utilisées. Comparez vos résultats avec les résultats attendus selon votre architecture RTL.**

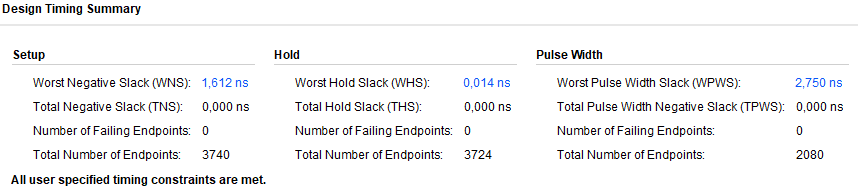


On a bien un registre de 28bits, un addeur qui va permettre de faire la comparaison des signaux (Cst et le compteur) et deux mux, le premier pour la gestion lié à Cmd (sur 28 bits) et le second sur un bit lié au restart.

1. **Ouvrez le Set Up Debug. Placez des sondes sur les signaux à observer que vous avez défini à la question 12.**

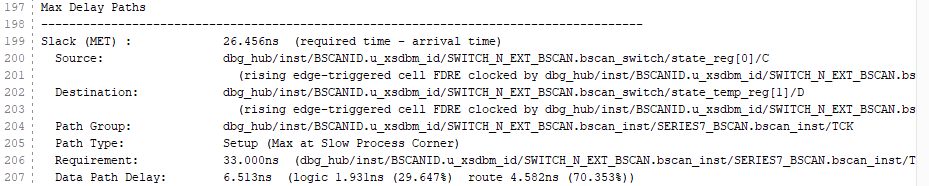


1. **Lancez l’implémentation puis étudiez le rapport de timing (vérifiez les violations de set up et de hold et identifiez le chemin critique).**

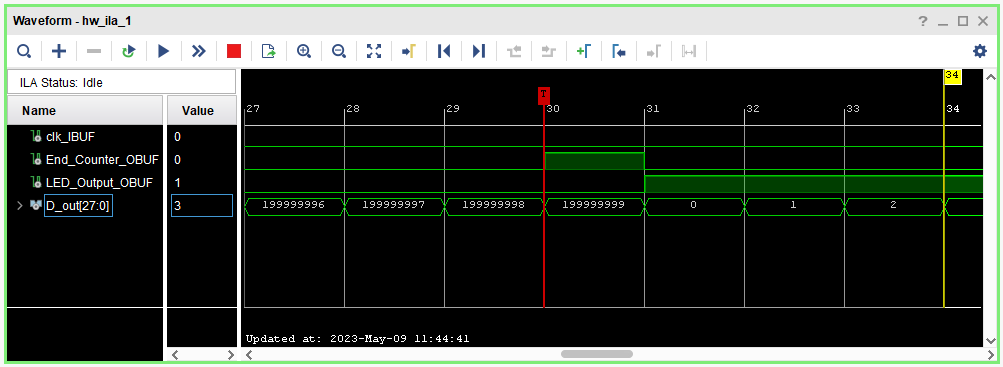


Le rapport de timming est conforme aux attentes pas de Slack (THS et TNS = 0ns) donc pas de métastabilité sur notre système.

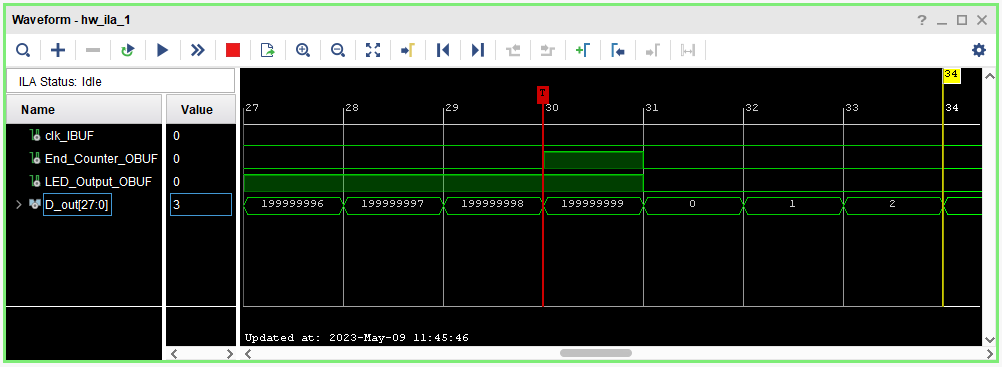
Le chemin critique est :



1. **Générez le bitstream pour observer le système sur carte. Relevez les résultats de la ILA.**



Les sondes ILAs valide le fonctionnement du système lors du passage de LED\_OUT à 1 lorsque le compteur (D\_out) atteint sa valeur max.



Les sondes ILAs valide le fonctionnement du système lors du passage de LED\_OUT à 0 lorsque le compteur (D\_out) atteint sa valeur max.