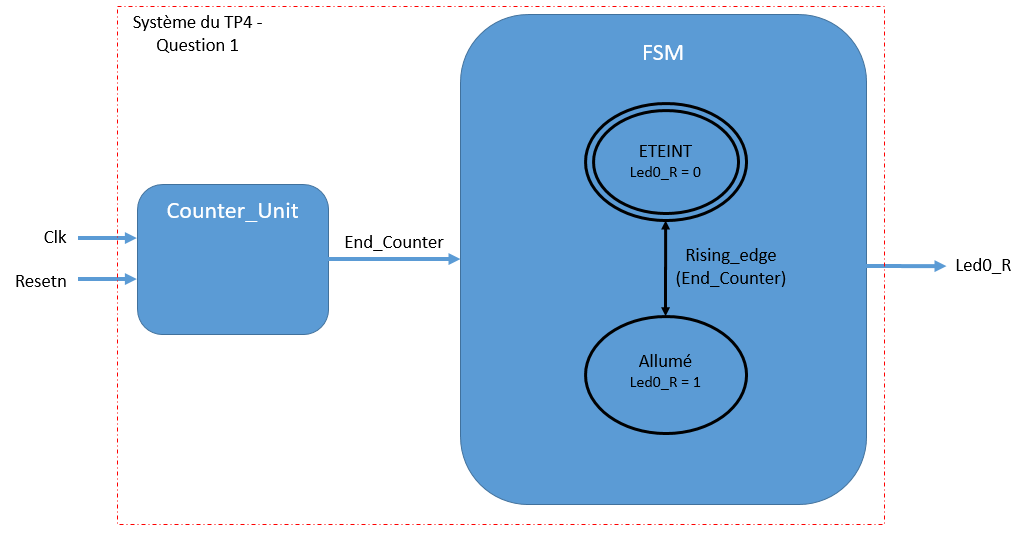
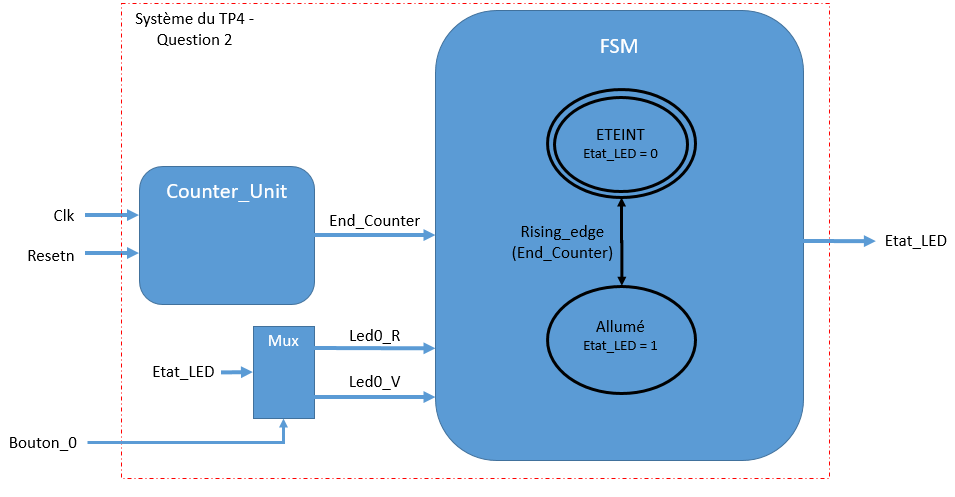
TP04 - Pilotage de LED et mémoire (Kamal KHERCHOUCH)

1. **Créez une architecture RTL permettant de faire clignoter une LED (par exemple la led0\_r) en utilisant le module *Counter\_unit* du TP2 et une machine à état.**



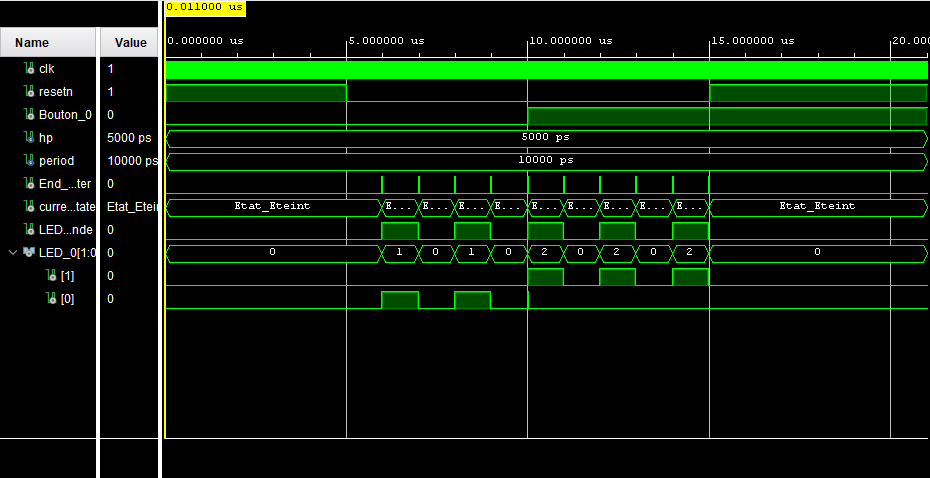
1. **Modifiez votre architecture pour piloter une LED rouge et une LED verte. Lorsque le bouton\_0 est appuyé, la LED verte est allumée, sinon la LED rouge est allumée.**



1. **Rédigez le code VHDL de votre architecture.**

Voir code.

1. **Réalisez une simulation en rédigeant un testbench. Que se passe-t-il si le bouton est pressé pendant plus d’un cycle d’horloge ?**



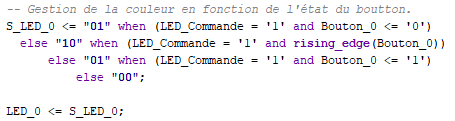
* Lorsque le resten est activé, le système ne fonctionne pas.
* Lorsque le resetn est désactivé, le compteur fait son travail.
* Lorsque le bouton\_0 est à 0 logique, seul le signal LED\_0[0] clignote en fonction de End\_Counter
* De même lorsque le bouton\_0 est un 1 logique avec LED\_0[1].

Si le bouton est pressé pendant plusieurs cycles d’horloge alors, le LED verte devrait se mettre à clignoter.

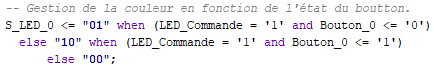
1. **Que faudrait-il faire pour que la LED ne clignote en vert qu’une seule fois même si le bouton est maintenu ?**

Il faut ajouter une condition que pour détecter le front montant sur le bouton 0. Cela permettra d’activer la LED verte une seule fois lors de la détection de l’appui, afin de revenir au clignotement de la LED rouge après coup.

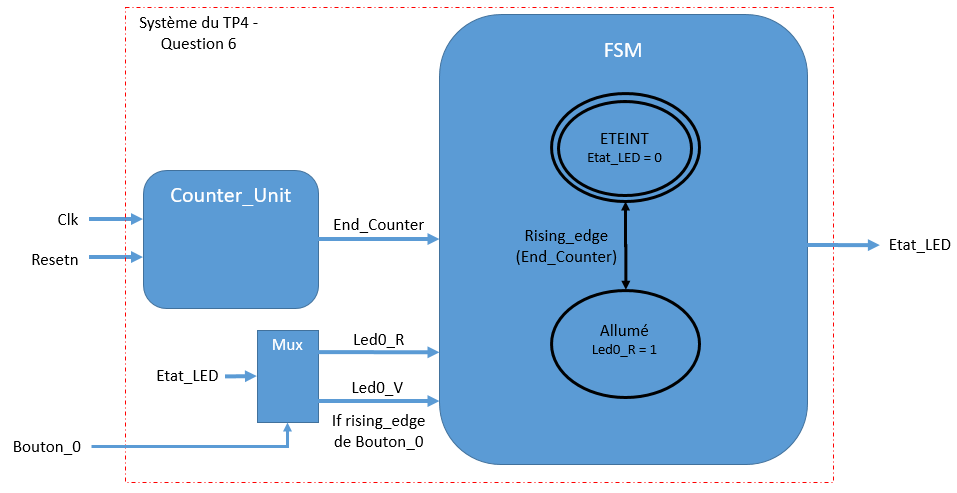
La partie combinatoire a donc était remplacé par :



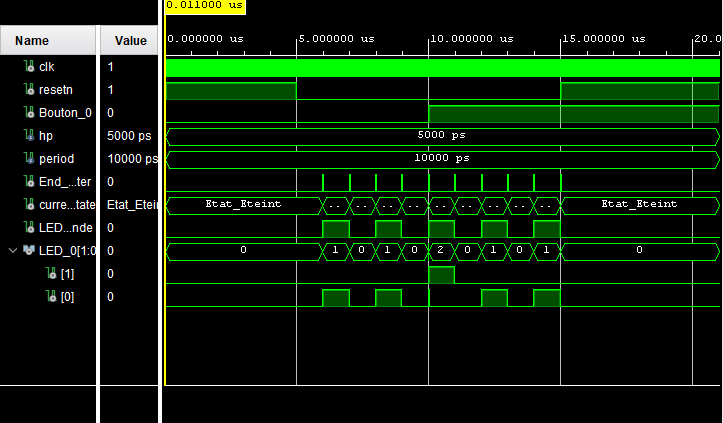
Au lieu de :



1. **Ajoutez cette solution à votre architecture RTL.**

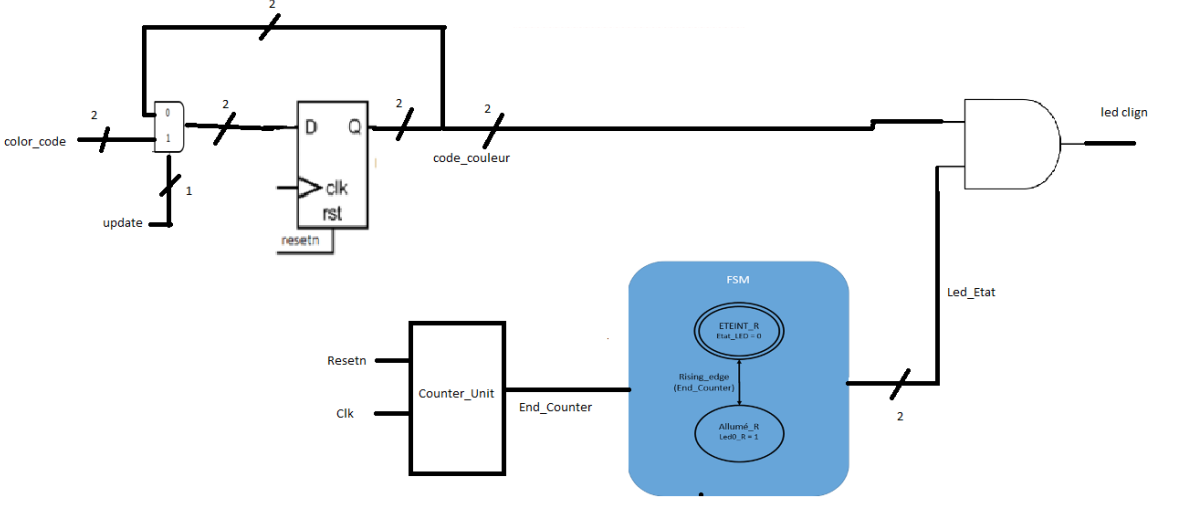


1. **Mettez à jour votre code VHDL et vérifiez votre résultat à la simulation.**

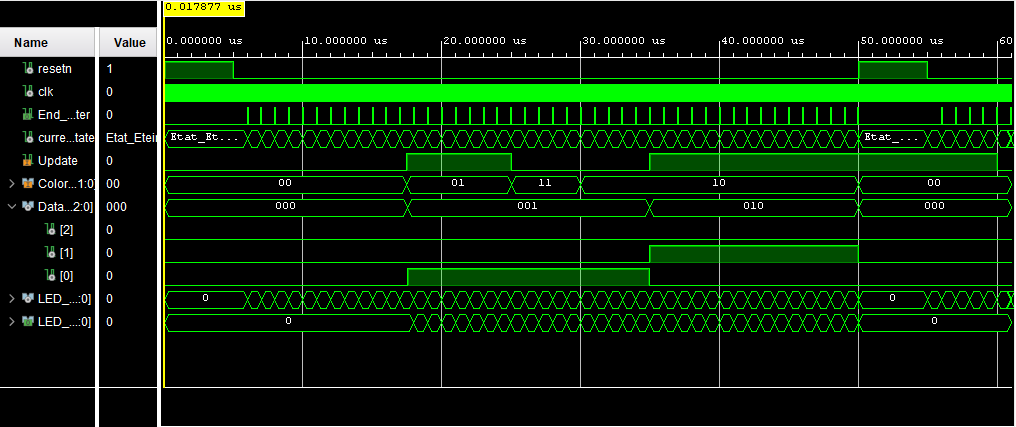


On voit bien que lors de la détection du front sur bouton 0, la LED verte clignote une fois, et la LED rouge reprend le relais.

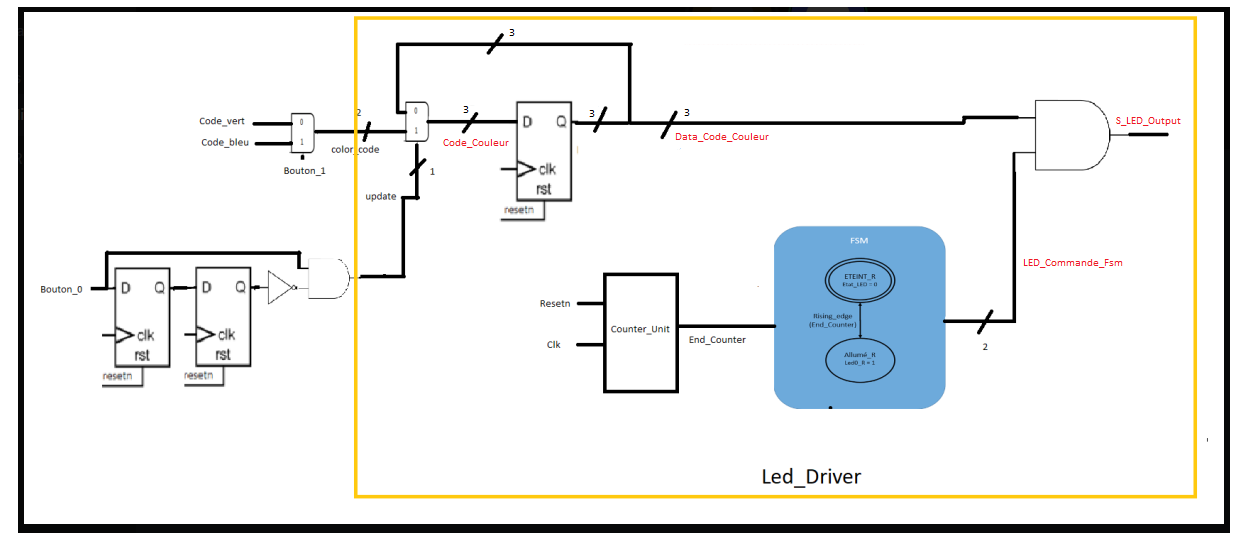
1. **Créez un module de pilotage d’une LED RGB en RTL. Ce dernier doit permettre de faire clignoter une LED RGB connectée en sortie d’une couleur définie par un code couleur donné en entrée. Le changement de couleur de la LED RGB n’a lieu que si un signal *update* est reçu.**



Le module LED Driver est créé (LedDriver.vhd) et testé via une simulation



1. **Ajouter la logique nécessaire pour piloter les entrées/sorties de votre module.**



Entrée :

* Clk
* Resetn
* Bouton\_Color\_Code (Bouton 1)
* Buton\_Update (Bouton 0)

Sortie :

* LED\_Output

1. **Ecrivez le code VHDL correspondant à votre architecture.**

Un nouveau projet a été créé sous le nom PilotageLED\_Avec\_Module\_LedDriver.vhd

Ce dernier utilise le module LedDriver qu’on vient de créer précédemment qui lui-même utilise le module Counter\_Unit validé lors du TP2.

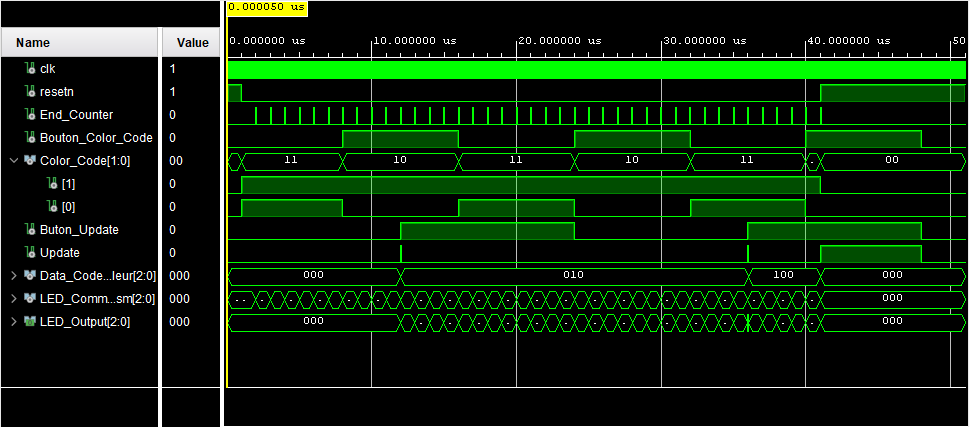
1. **Ecrivez un testbench pour tester votre design.**

Le testbench associé est à consulté via le fichier tb\_PilotageLED\_Avec\_Module\_LedDriver.vhd

Un prend en compte différents processus :

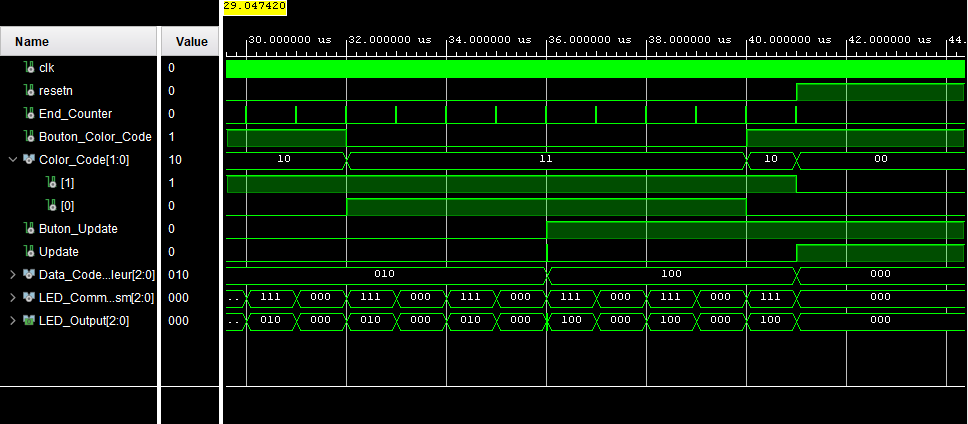
* Gestion de l’horloge : clk
* Gestion du reset : resetn
* Gestion du bouton 1 : Bouton\_Color\_Code
* Gestion du bouton 0 : Buton\_Update

1. **Vérifier votre résultat à la simulation.**



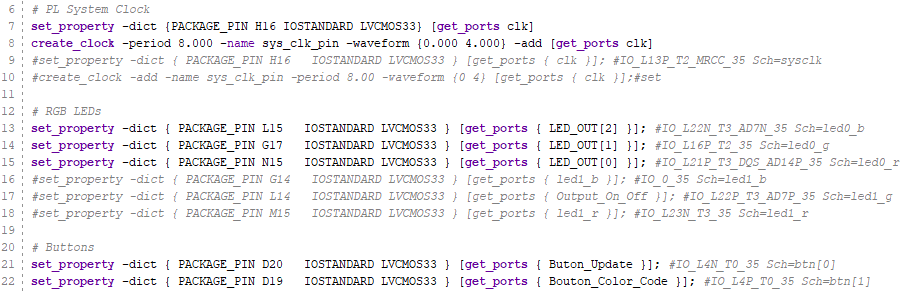
Le système est validé :

* L’activation du reset désactive le comptage du signal End\_Counter, la gestion du signal Color\_Code et force notre sortie à 0 (LEDs OFF)
* Ici on voit bien que lorsque le Bouton\_Color\_Code est à 0 alors on selectionne Color\_code = « 11 » ce qui correspond à la couleur bleu. Sinon on sélectionne la couleur vert avec Color\_code = « 10 »
* On voit également que lorsqu’on appui sur le Buton\_Update le front montant est détecté par le signal Update et que le signal Data\_Code\_Couleur prend la couleur sélectionné par Code\_Couleur. Lorsque le Buton\_Update est relâché, le signal Data\_Code\_Couleur conserve son ancienne valeur.
* Le clignotement des LEDs se fait correctement voir image ci-dessous.



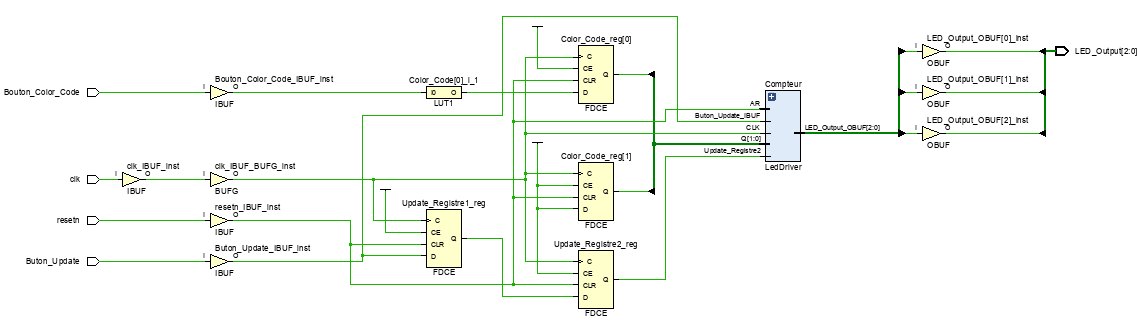
1. **Réalisez une synthèse et étudiez le rapport de synthèse, les ressources utilisées doivent correspondre à votre schéma RTL.**

Une mise à jour du fichier de contrainte



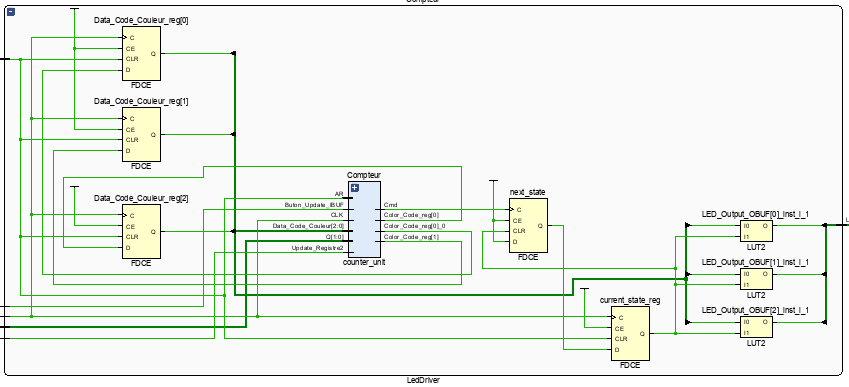


On obtient le schéma RTL suivant :

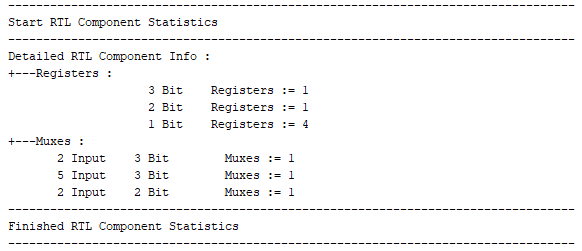


Sur le Schéma RTL suivant on voit bien que l’on a :

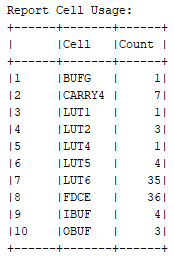
* 4 registres (2 pour la gestion de l’Update et 2 pour la gestion du color\_code)
* Le module LedDriver
* Nos entrées et sorties



Si on détaille notre module LedDriver on y retrouve bien Notre module Counter\_Unit, nos registres pour le code Couleur sur 3 bits et notre machine d’état à 2 états.



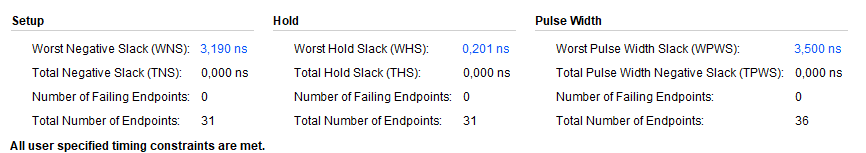
Le rapport de synthèse nous confirme qu’on utilise bien 3 registres et 3 multiplexeurs, ce qui est conforme au schéma RTM proposé.



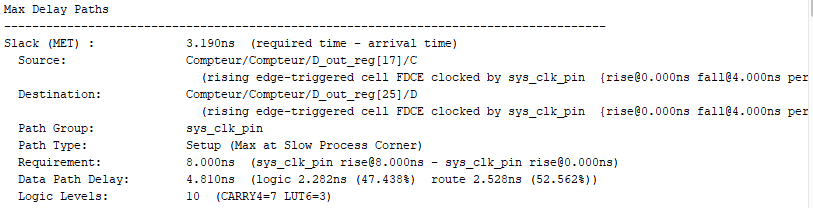
Ici, on voit bien qu’on a :

* 4 Entrées : IBUF
* 3 Sorties : OBUF
* 36 registres : FDCE (26 registres pour le module Counter\_Unit, 6 pour le module LedDriver (avec la prise en compte de la Fsm) et 4 pour le système)

1. **Effectuez le placement routage et étudiez les rapports.**



Notre système ne devrait pas rencontrer de métastabilité car on ne retrouve pas de stack (TNS et THS = 0ns)



On observe que chemin le plus long est celui du module compteur, ce qui ne me semble pas aberrent vu qu’il utilise beaucoup de registre.

1. **Générez le bitstream et vérifiez que vous avez le comportement attendu sur carte.**

Voir la Video