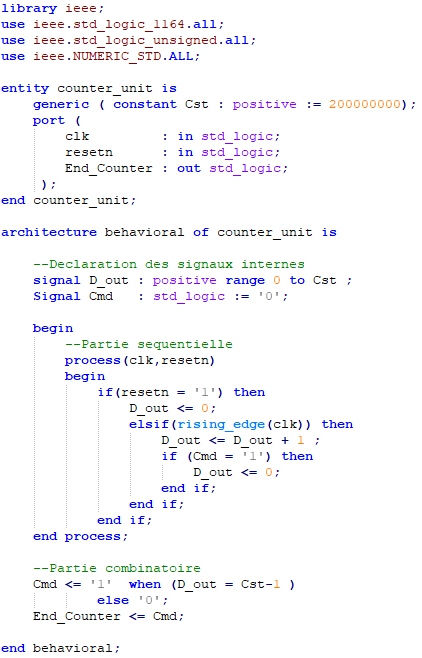
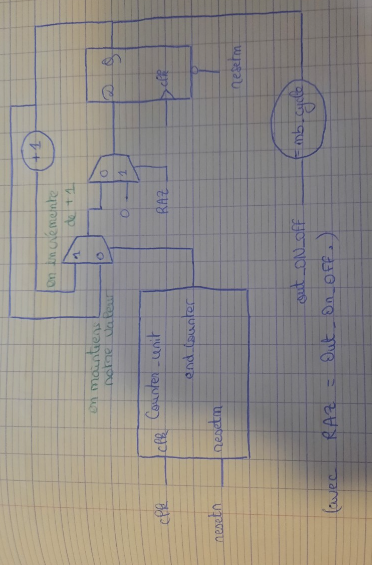
TP03 - Machine d’états (Kamal KHERCHOUCH)

1. **Dans un fichier *.vhd*, créez un module *Counter\_unit* à partir du compteur du TP1. Le module prendra en entrée un signal d’horloge et de resetn, et donnera en sortie le signal *end\_counter*. Utilisez un paramètre *generic()* pour définir le nombre de coup d’horloge à compter.**

Le module Counter\_Unit est decrit dans le fichier Counter\_Unit.vhd.



1. **En schéma RTL, créez un compteur du signal *end\_counter*. Ce compteur doit permettre de déterminer le nombre de cycles allumé/éteint qui ont été effectués par la LED. Le compteur doit pouvoir être remis à 0, maintenir sa valeur actuelle ou s’incrémenter.**

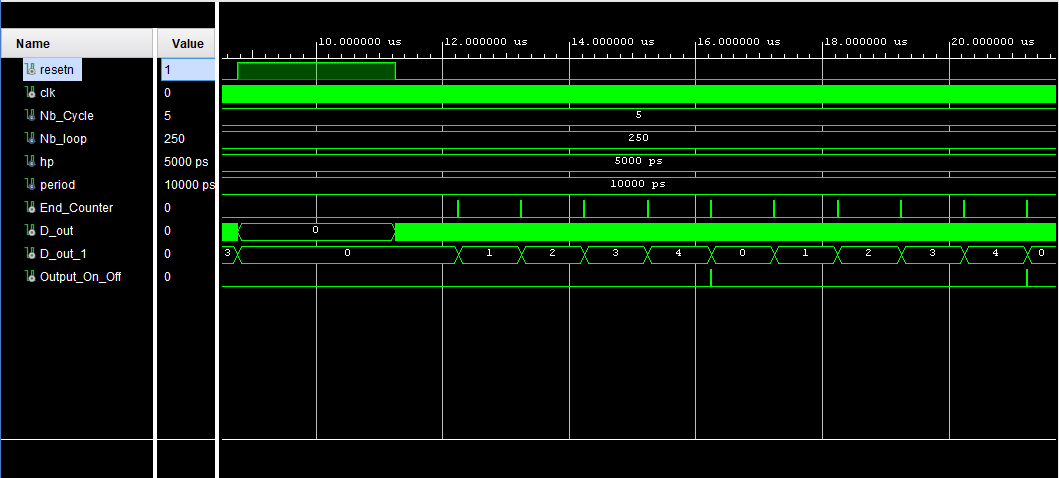


1. **Ecrivez un code VHDL décrivant ce compteur de cycle, vous utiliserez le module *Counter\_unit*.**

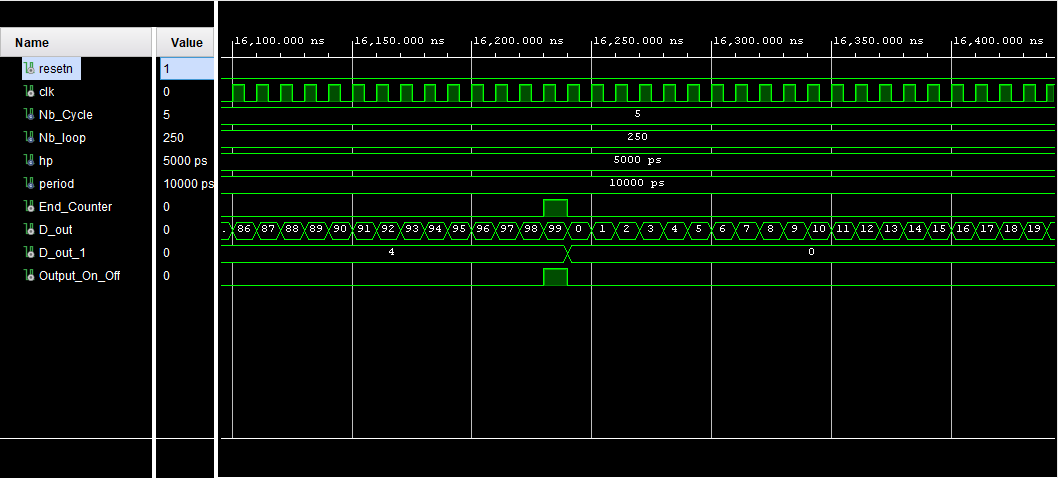
Une copie du code VHDL (à cette étape) est placée dans le dossier sous le nom « tp\_fsm - Question 3.vhd »

Il en est / sera de même pour son test bench associé « tb\_tp\_fsm - Question 3.vhd »

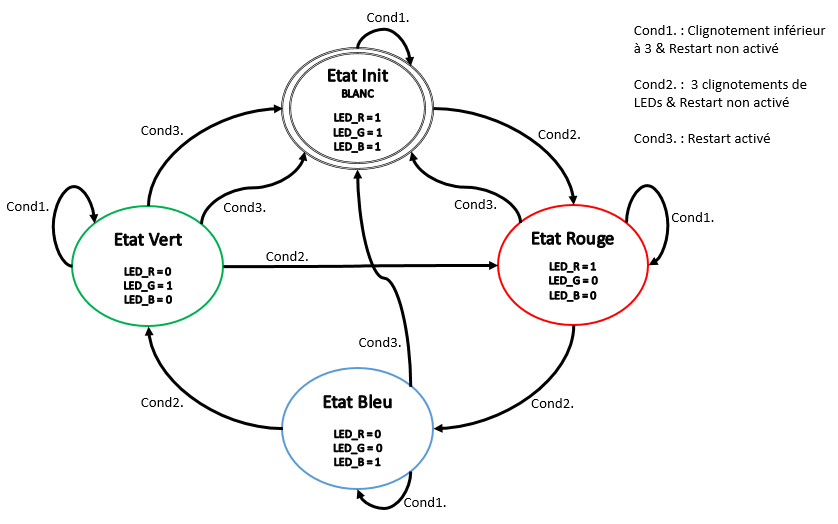
1. **Tester votre architecture avec un testbench.**



Ici, on valide bien le fonctionnement du reset. Lorsque le reset est activé, le compteur s’arrête de compter. Et lorsque le compteur redémarre, on attends bien 5 cycle de comptage du signal End\_Counter avant d’activer le signal de sortie Output\_On\_Off (ce dernier se fait lors de la dernière période du dernier cycle demandé). Voir image ci-dessous.



1. **Créez en RTL une machine à états (FSM) permettant de faire clignoter une LED RGB en rouge puis bleu et enfin en vert avant de recommencer le cycle (rouge, bleu, vert, …). Dans chaque état la LED devra clignoter 3 fois. De plus, si le bouton restart est appuyé, on retourne dans l’état initial quel que soit l’état dans lequel on se situe. L’état initial est l’état dans lequel on se situe au démarrage, on passe à l’état rouge après 3 clignotements de la LED en blanc (rouge, vert et bleu actifs en même temps).**



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

Les signaux d’entrée sont :

* Clk : in std\_logic ; 🡪 Horloge de notre système
* Resetn : in std\_logic ; 🡪 Reset de notre système
* Restart : in std\_logic ; 🡪 RAZ de la machine d’état

Les signaux de sortie sont :

* Output\_On\_Off : out std\_logic ; 🡪 Sortie du compteur de notre Syst.
* LED\_OUT : out std\_logic\_vector (2 downto 0) 🡪 LEDs RGB pour la carte CORA

Les signaux internes sont :

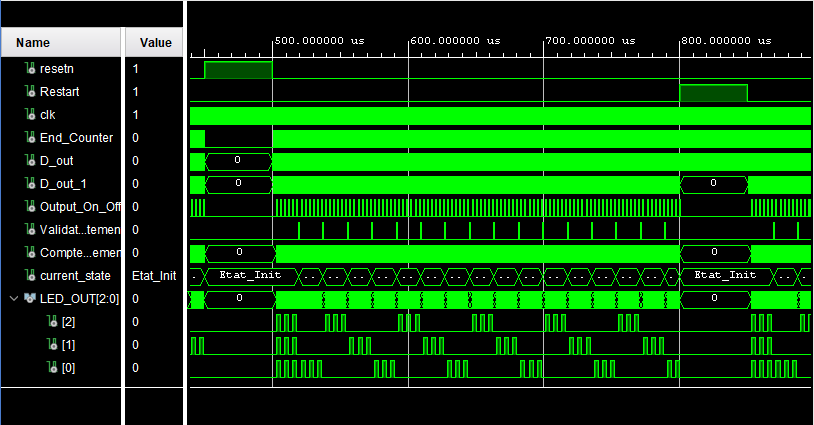
* S\_LED\_OUT : std\_logic\_vector (2 downto 0) ; 🡪 Signal interne pour LED\_OUT
* Compteur\_clignotement : positive :=0 ; 🡪 Comptage du Nb de clignotement
* Validation\_clignotement : std\_logic ; 🡪 Permet la validation du chgt d’état

**7. Ajoutez à votre code VHDL les éléments que vous venez de créer.**

Cf. Code complet du projet.

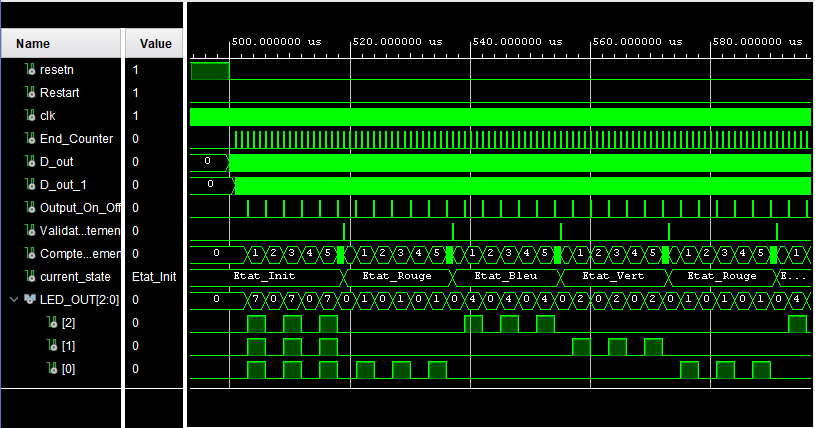
**8. Ecrivez un testbench pour tester votre architecture. Vérifiez à la simulation que vous obtenez le résultat attendu.**

Validation des signaux resetn et Restart : lorsque l’un des deux signaux est activé, tous les signaux de comptage retours à 0 et la machine d’état revient à l’état initial avec les LEDs en OFF.



Validation de la machine d’état : Ici, on voit bien qu’on suit le chemin de notre machine à savoir

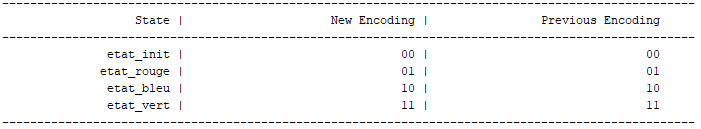
Etat\_Init 🡪 Etat\_Rouge 🡪 Etat\_Bleu 🡪 Etat\_Vert puis on revient à Etat\_Rouge car aucun restart n’est activé.



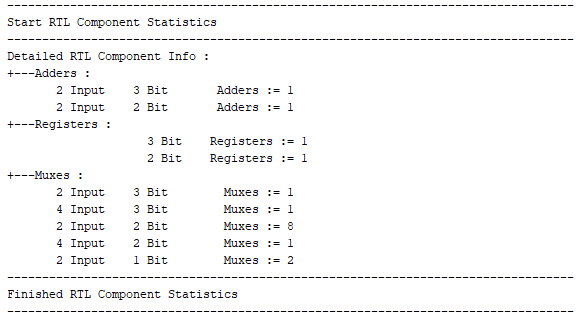
**9. Exécutez la synthèse et relevez les ressources utilisées (y compris la FSM). Sur la schématique, identifiez où se situe votre compteur de cycle.**

Dans le fichier : Synthesis\_report, on retrouve :

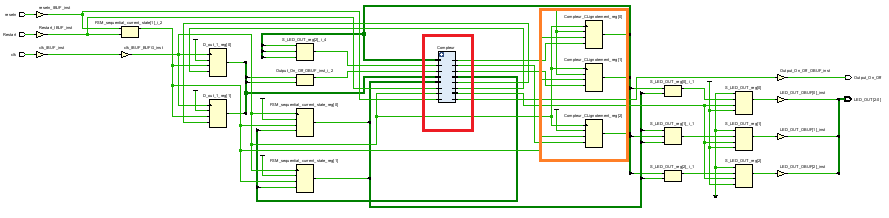
* La description de notre machine d’état



* Les composants qui permettent de constituer notre schéma RTL



Le schematic est le suivant :



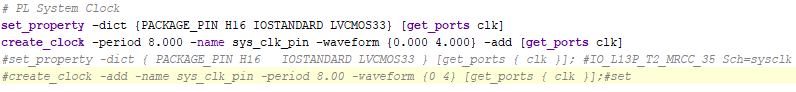
En rouge on repère notre Counter\_unit module et en orange le compteur qui nous permet de gérer le comptage du clignotement.

**10. Modifiez le fichier de contraintes pour connecter vos entrées / sorties du système avec les broches de la carte. Réglez l’horloge pour que sa fréquence soit à 100MHz.**

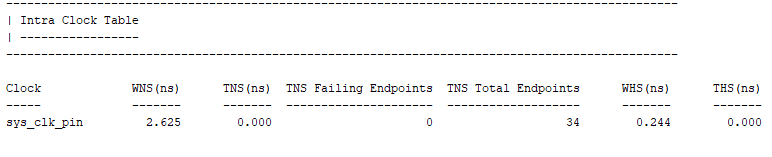
Dans le fichier de contrainte :

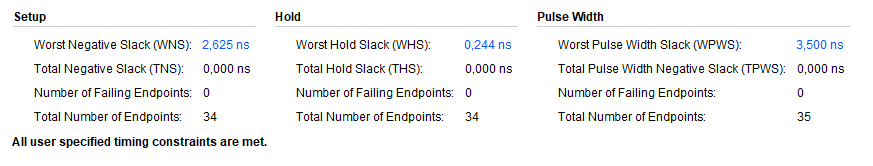
* btn[0] sera affecté au signal resetn
* btn[1] sera affecté au signal Restart
* led0\_b sera affecté au signal LED\_OUT[2]
* led0\_g sera affecté au signal LED\_OUT[1]
* led0\_r sera affecté au signal LED\_OUT[0]

Il faut également modifier le PL System Clock de la façon suivante :



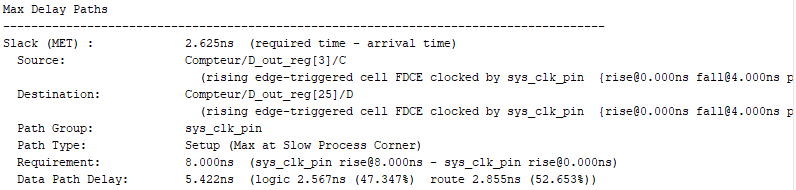
**11. Lancez l’implémentation puis étudiez le rapport de timing (vérifiez les violations de set up et de hold et identifiez le chemin critique).**





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

Le chemin critique est le suivant :



**12. Générez le bitstream pour vérifier le système sur carte.**

Dans un premier temps, un bitstream à était généré avec :

* generic ( constant Cst : positive := 100000000);

Le fichier bitstream est dispo au lien : TP\TP03\Bitstream\**tp\_fsm\_3secondes.bit**

Pour les besoins de la réalisation de la vidéo, on va régénérer un bitstream avec :

* generic ( constant Cst : positive := 100000000/3);

Le fichier bitstream est dispo au lien : TP\TP03\Bitstream\**tp\_fsm\_1seconde.bit**