

TP3 : machine à état

2)

Création en schéma RTL d'un compteur sur end_counter.

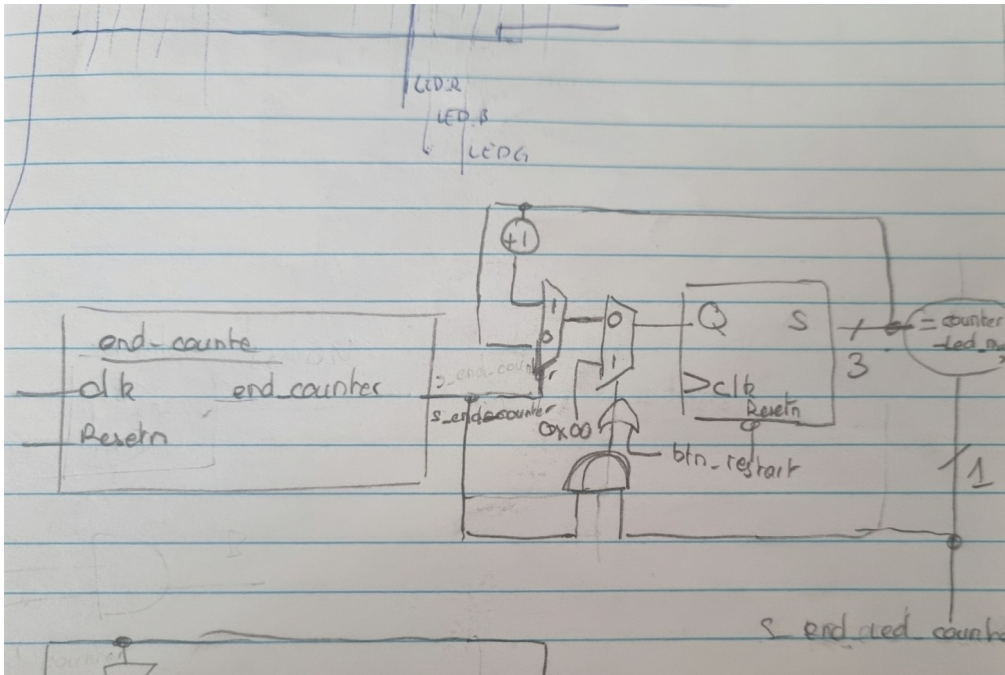


Figure 1: schéma RTL pour le compteur du signal end_counter

En entrée, nous aurons :

- clk
- resetn.

En signal interne :

- s_end_counter pour counter unit
- s_end_led_counter pour signaler la fin du compteur
- s_counter : sera le compteur de basé sur end_counter.

En sortie :

- end_led_counter, qui reprendra s_end_led_counter.

4)

Nous générons le testbench, avec un counter_unit sur 10 cycles d'horloge, et le compteur de cycle sur 6 coups d'horloge. Nous testerons également le resetn.

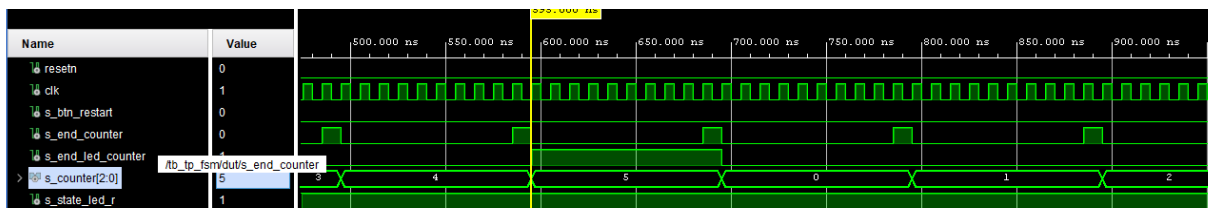


Figure 2: incrémentation du compteur et remise à zéro

Nous pouvons voir sur la figure précédente (Figure 2) le compteur qui s'incrémente sur chaque signal s_end_counter, et qui repart à zéro quand nous avons compté le 6me cycle en entier. Sur la figure ci-dessus, nous voyons bien le compteur incrémenté pour chaque passage de s_end_counter à 1.

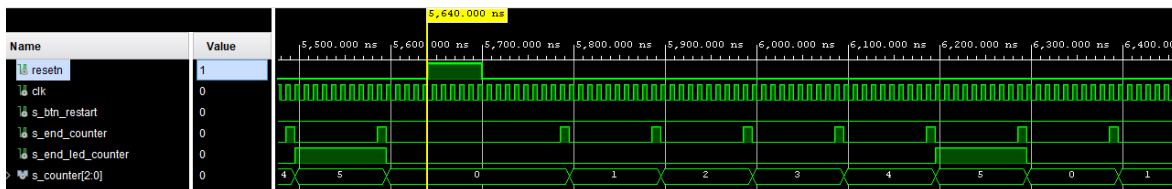


Figure 3: reset du compteur de cycle

Enfin sur la figure ci-dessus(Figure 3), nous confirmons le comportement à un reset, pour un départ du compteur.

5)

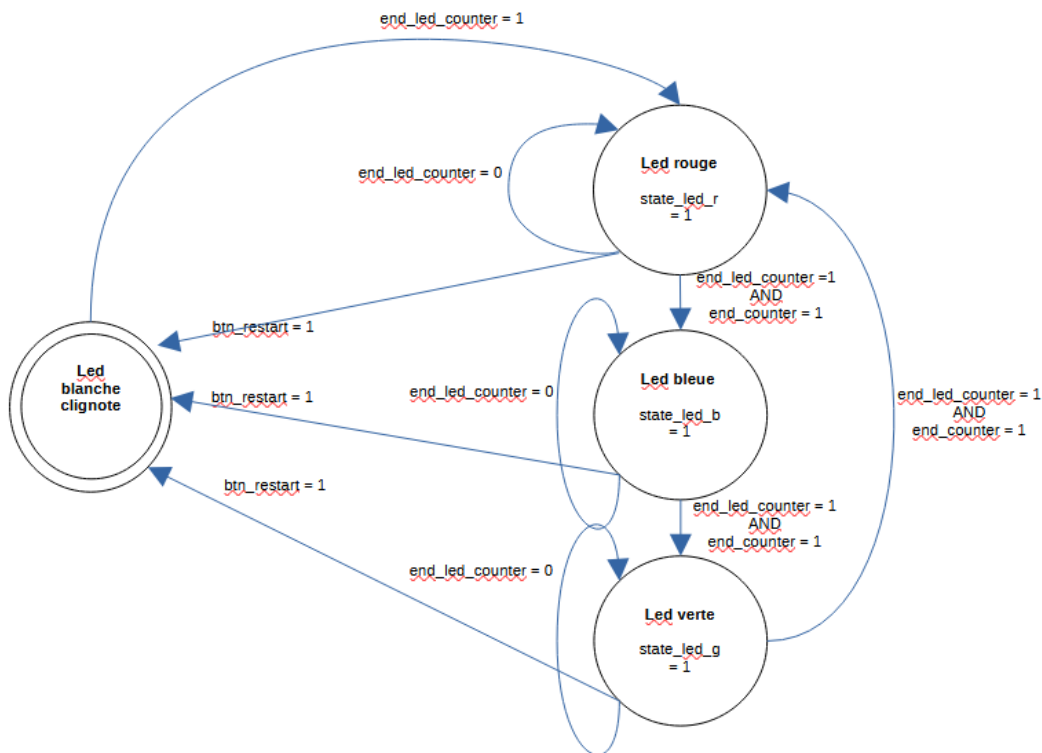


Figure 4: machine à état pour clignotement de led

Dans cette machine, nous avons en signal d'entrée : *btn_restart*, *end_led_counter* et *end_counter*, soit respectivement l'appuie du bouton restart, la fin du compteur de cycle de led allumée/éteinte et le signal de fin du counter_unit.

En sortie, nous aurons *state_led_r*, *state_led_b* et *state_led_g*, 1 quand la led doit être allumée, ou 0 sinon (ex : Pour led blanche clignote, *state_led_r* = *state_led_b* = *state_led_g* = 1) .

allumage des LED :

Pour clignoter les LEDs, nous utiliserons un signal *s_led_on*, qui coupler à *s_state_led_r*, *s_state_led_b* et *s_state_led_g*, permettra de faire clignoter les leds lorsqu'il sera à 1. Le schéma RTL est donné ci-dessous.

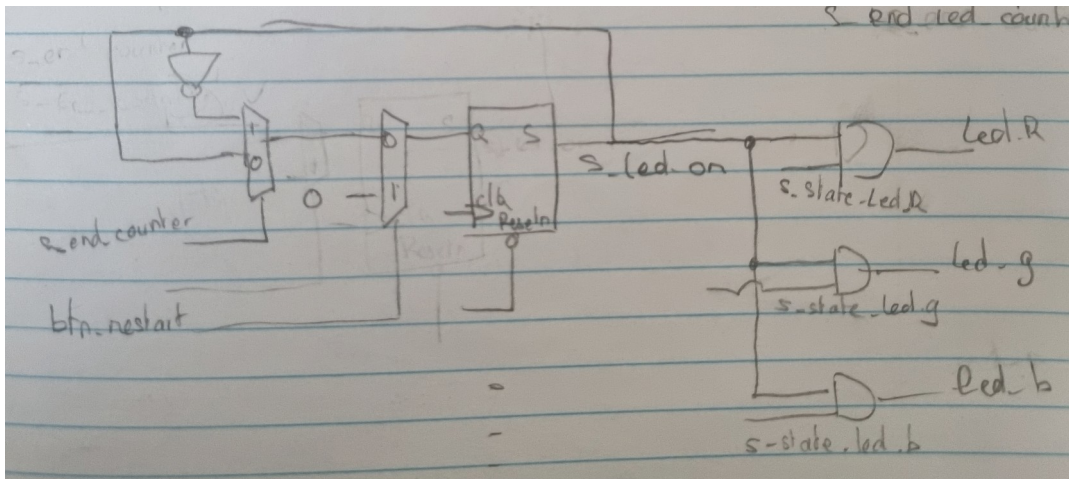


Figure 5: RTL design pour le clignotement des LEDs.

8)

Le testbench permettra de tester :

- le passage de l'état initial au état suivant : la boucle de clignotement de led
- test du bouton restart, avec le compteur déjà incrémenté. Avec une led allumée sur une 10 période de 10ns (100 Mhz), qui doit donc clignoter 3 fois, soit 600 ns ($6 \times 10 \times \text{periode} = \text{temps d'un état}$). nous attendons plusieurs cycle : ici 6, donc une attente de 360 période, soit 3,6 μ s. Nous voulons vérifier le restart sur un état entamé, donc rajoutons à ce temps 10 périodes (100 ns).
- Observation du reset. Nous attendons 3 cycles ($3 \times 6 \times \text{périodes}$) avant de lancer l'appuie sur le reset.

Nous obtenons les courbes suivantes.

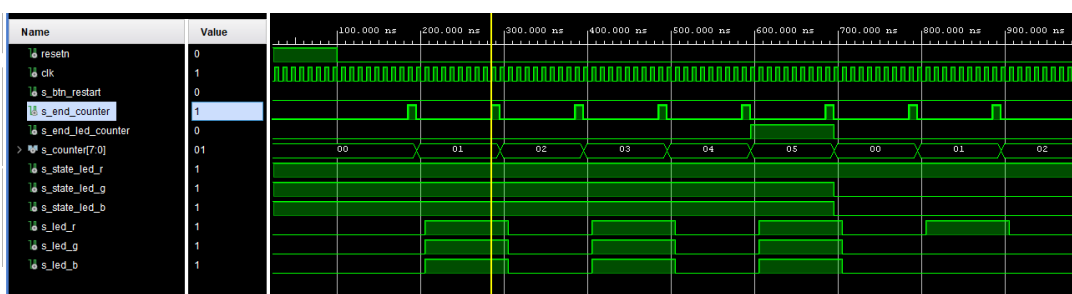


Figure 6: clignotement des leds à l'état initial

Sur la courbe ci-dessus (Figure 6), nous pouvons voir qu'au début, à l'état initial, les 3 leds clignotent en même temps, puis après trois fois les trois leds allumées et éteintes, nous entrons dans le cycle rouge, où seulement celle-ci sera allumée.

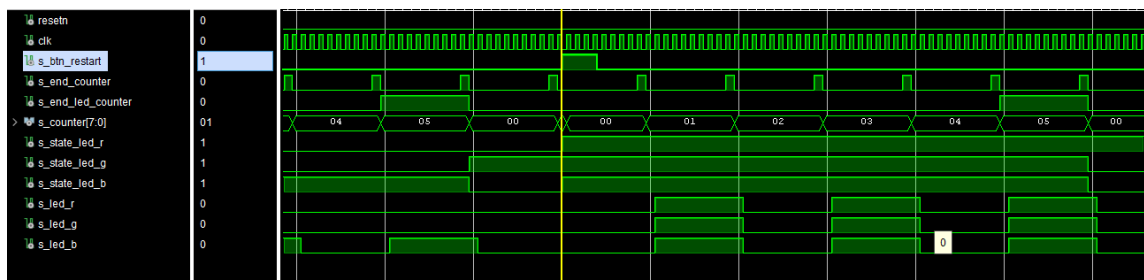


Figure 7: retour à l'état initial avec l'appuie de restart

sur la courbe ci-dessus(Figure 7), nous pouvons voir l'état vert qui passe à l'état bleu, puis lors de l'appuie du bouton restart (s_btn_restart), nous passons dans le cycle initial, avec le clignotement des leds.

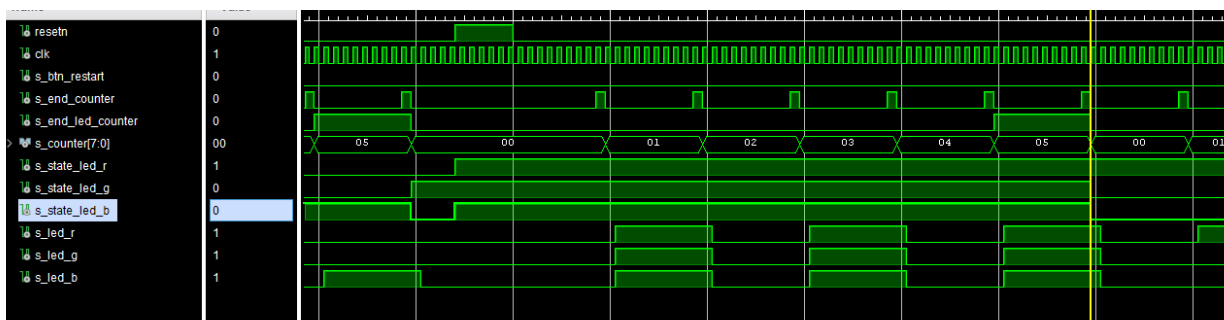


Figure 8: retour à l'état initial par appuie sur bouton reset

Sur la Figure 8, première ligne, activation du compteur. Nous avons ici le compteur qui maintient son zéro (en fait il recommence bien à zéro), et les trois LEDs qui clignotent en même temps.

9)

Detailed RTL Component Info :

+---Adders :

2 Input 3 Bit Adders := 1

+---Registers :

3 Bit Registers := 1

1 Bit Registers := 1

+---Muxes :

2 Input 3 Bit Muxes := 1

4 Input 2 Bit Muxes := 1

2 Input 2 Bit Muxes := 7

2 Input 1 Bit Muxes := 2

4 Input 1 Bit Muxes := 3

Nous avons un compteur sur 3 bits, pour compter le nombre de cycle allumé-éteint, avec 2 entrées : clk et btn_restart.

2 registres : 1 à 3 bits pour le compteur de cycle de led, et un autre pour allumer-éteindre les LED.

Multiplexeurs :

- Nous retrouvons les 2 multiplexeurs pour le compteur d'état de led :
 - 1 multiplexeur avec une entrée de 3 bits, qui correspond sur le schéma au multiplexeur qui permet d'incrémenter ou non le compteurs.
 - 1 multiplexeur avec 2 entrées à 1 bit qui correspondra au choix du retour du signal à zéro ou 1.
- Pour commander l'allumage des led, 2 multiplexeurs ont été utilisés, regroupés en 1 seul car utilise le même signal de commande : 'btn_restart'. C'est le multiplexeur à 4 entrées sur 1 bit.
- Pour la machine à état, nous trouvons, qui commande « next_state » :
 - Le 2^{me} multiplexeur à 2 entrées sur 1 bit.
 - 1 multiplexeur avec 4 entrées à 2 bits
 - 5 multiplexeurs à 2 entrées sur 2 bits.
- Pour la commande d'état des LED (machine à état) : les 3 multiplexeurs avec 4 entrées sur un bit. Ceux-ci correspondent à la commande des états rouges, vert et bleue : state_led_r, state_led_g, state_led_b, commandés par current_state (cf Figure 9). (selon le current_state, gestion des états des leds).

Il manque 2 multiplexeurs sur les 7 à 2 entrées sur 2 bits, qui n'apparaissent pas dans VIVADO, dont je ne peux remonter la source.

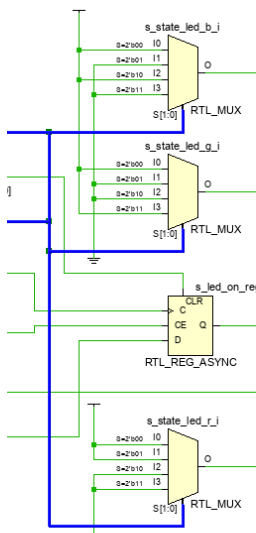


Figure 9:
multiplexeurs de
commande d'état
des leds

19	+	+	+	+
10		Cell	Count	
11	+	+	+	+
12	1	BUFG		1
13	2	CARRY4		7
14	3	LUT2		31
15	4	LUT3		4
16	5	LUT4		4
17	6	LUT5		4
18	7	LUT6		3
19	8	FDCE		34
.0	9	IBUF		3
.1	10	OBUF		3
.2	+	+	+	+

Nous pouvons voir ici que nous avons :

- 3 buffer d'entrée : clk, resetn, bouton_restart
- 3 buffers de sortie : led_r, led_g, led_b
- 34 registres à reset asynchrone : 28 pour le compteur de cycle d'horloge, 3 pour le compteur de cycle (3 bit), et 2 pour la machine à état et enfin 1 pour faire clignoter les LED.

9)

