1) l’horloge à 100 MHz, période ( = 10 ns)

pour 2 s, il faudra :

soit , ce qui demandera 28 bits pour représenter ce signal.

Pour des raisons de fonctionnement de la carte indépendant de ma volonté, le compteur sera par la suite codé sur 29 bit.

3) La condition pour que le compteur soit remis à zéro, en dehors du reset, c’est quand « end\_counter » passera au 1 logique.

4) les signaux d’entrées seront :

* clock (clk)
* resetn

Le signal de sortie :

* end\_counter

Signal interne :

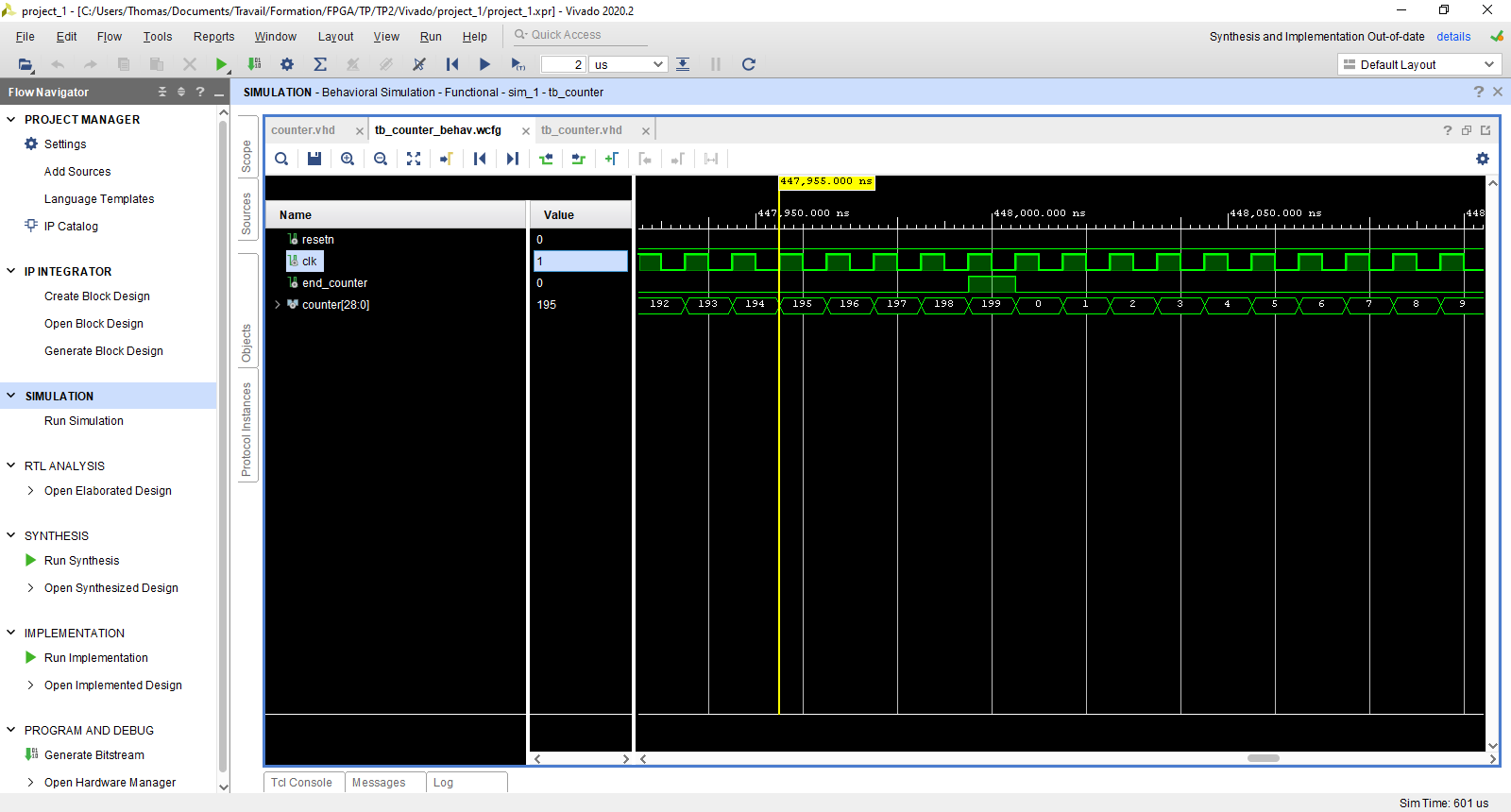
* counter
* s\_end\_counter : signal en interne, pour pouvoir utiliser end\_counter comme une entrée, vérifier son état.

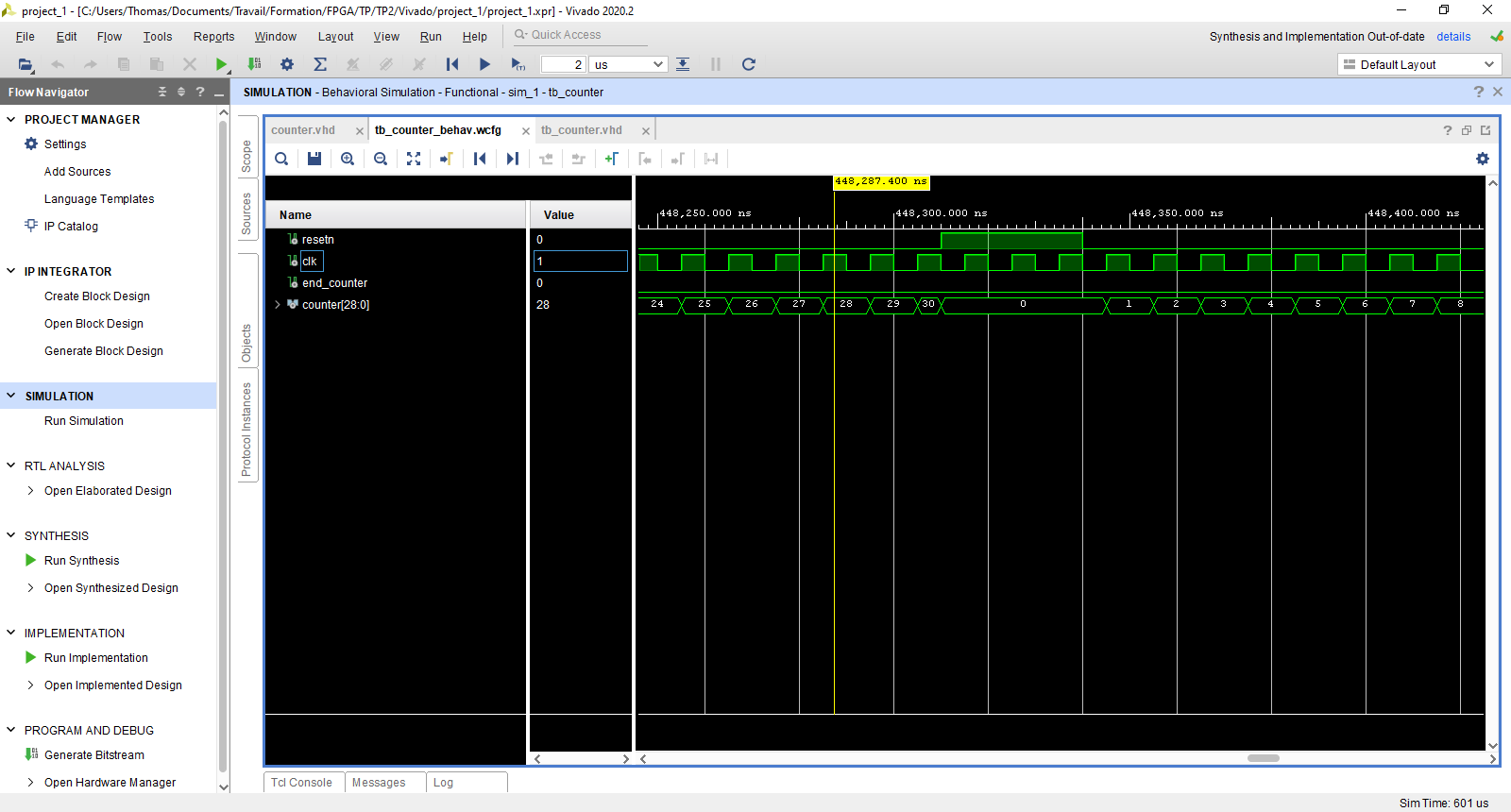
Figure 1: schéma RTL du compteur

7)

Sur la simulation, nous faisons compter jusqu’à 200 périodes, soit de 0 à 199. Nous voulons voir alors à 199 la sortie end\_counter passer à 1, puis alors le compteur recommencer à 0 (cf Figure 2).

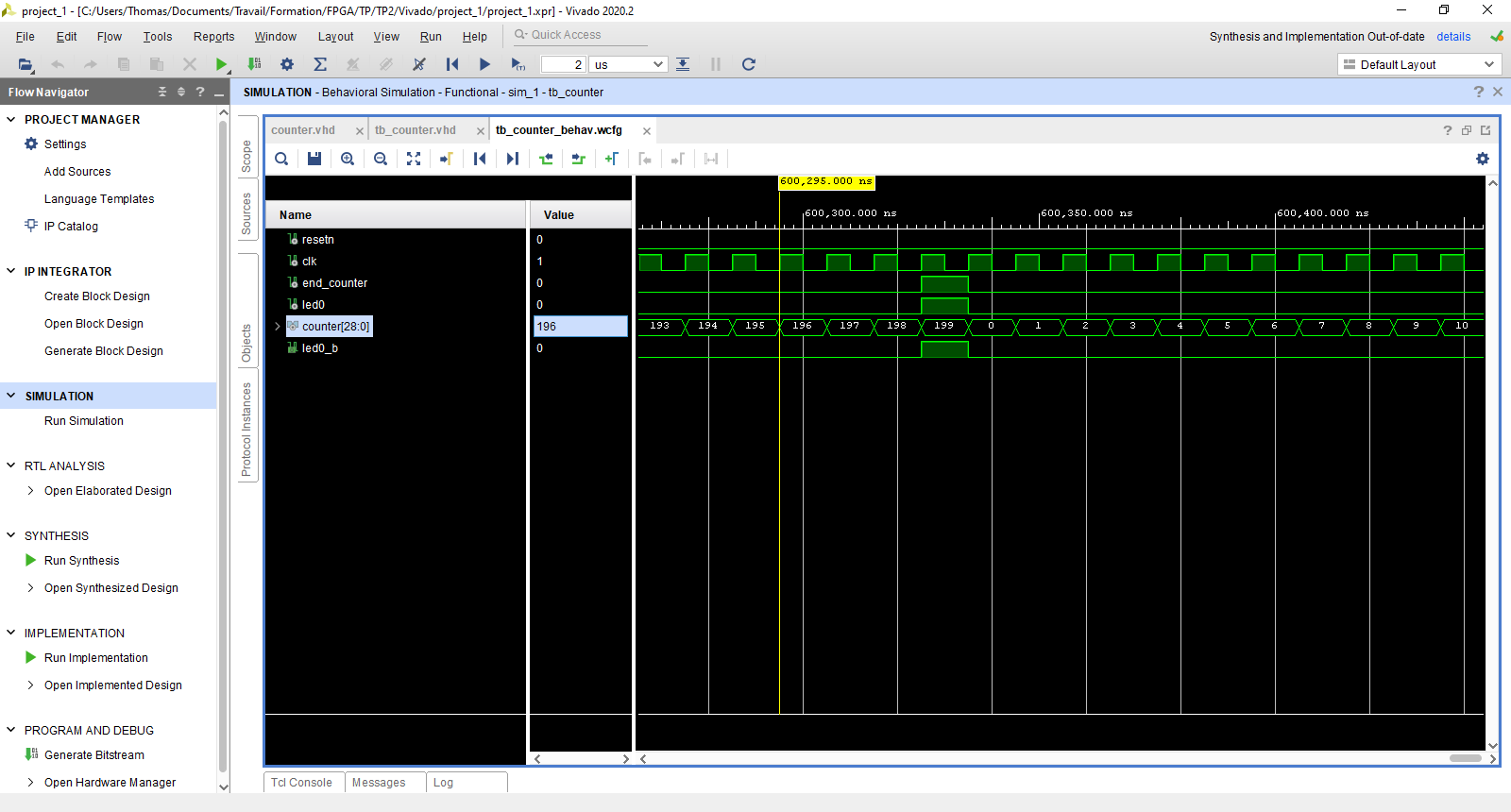
Nous voulons voir aussi le compteur recommencer à zéro lors d’un resetn passé à 1 (cf Figure 3).

Figure 2: passage de end\_counter à zéro, end\_couter = 1

Figure 3: reset du compteur

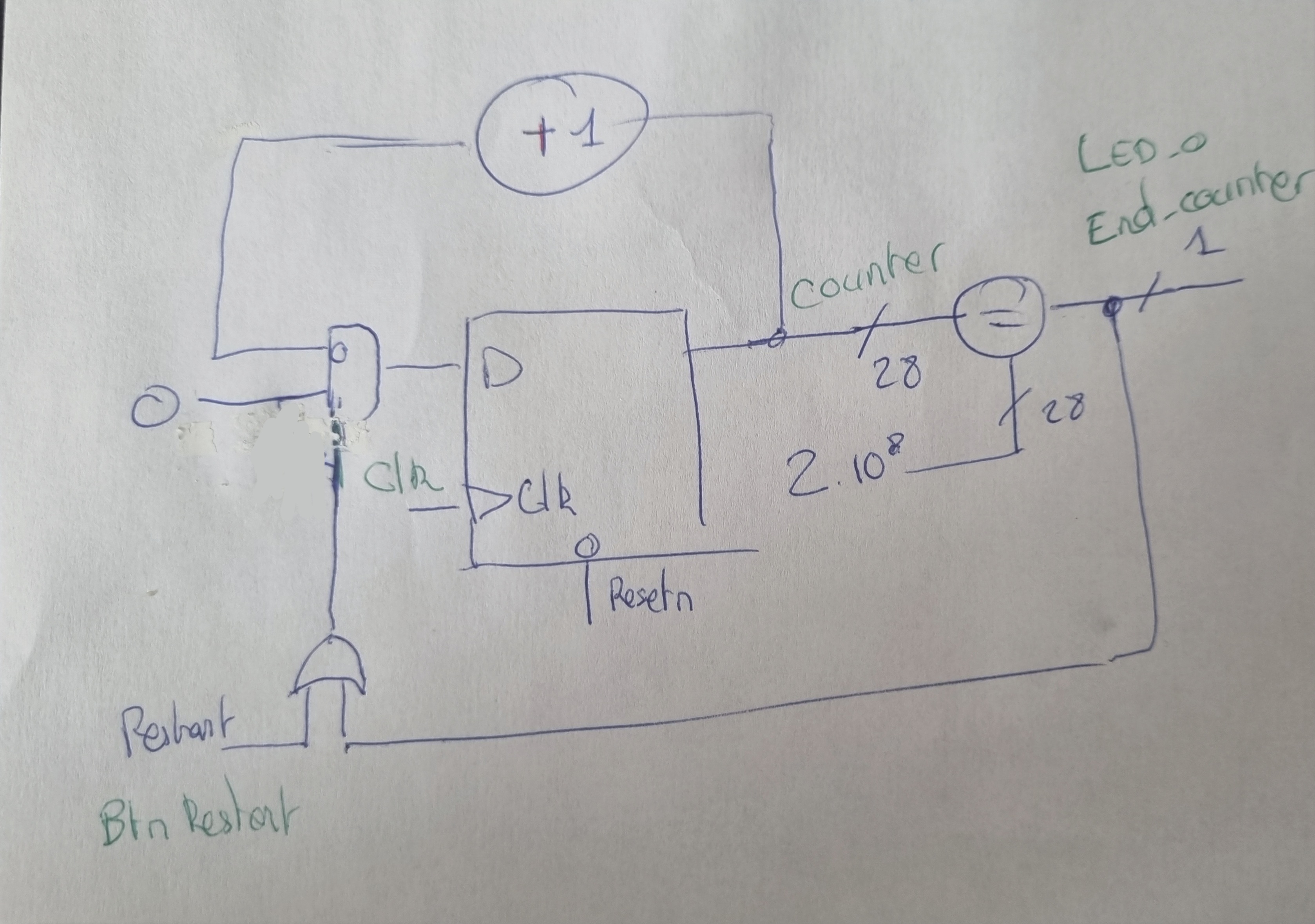
8)

Pour associer la LED avec le signal de fin de compteur, il suffit de rajouter une sortie LED, et l’associer avec le signal end\_counter. Nous vérifions ce bon fonctionnement dans la Figure 4.

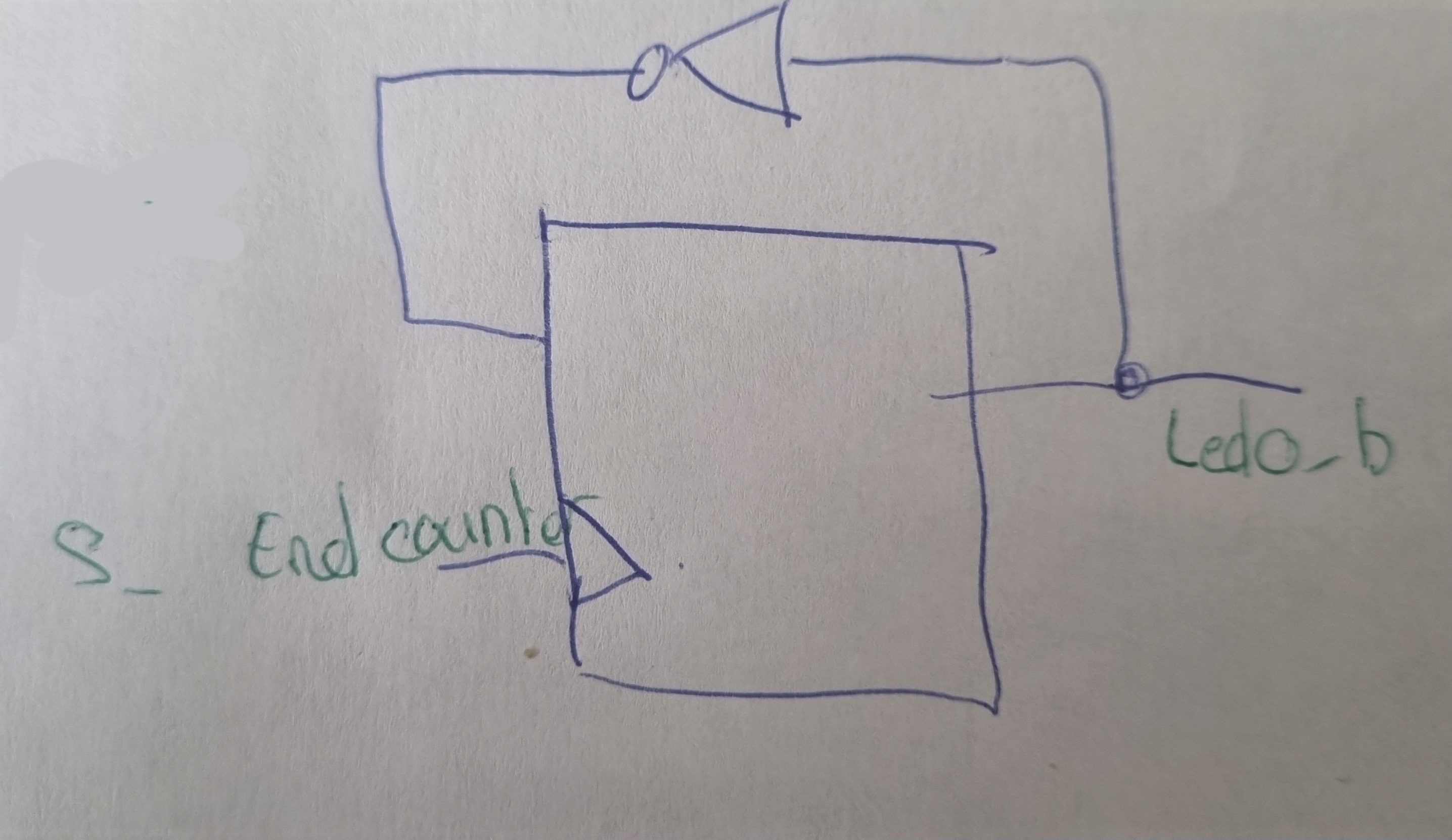
Figure 4: led associée au signal de fin de compteur

9)

Rajout d’un porte OR pour pouvoir réaliser la remise à zéro avec un bouton :

Figure 5: compteur avec un bouton de remise à zéro

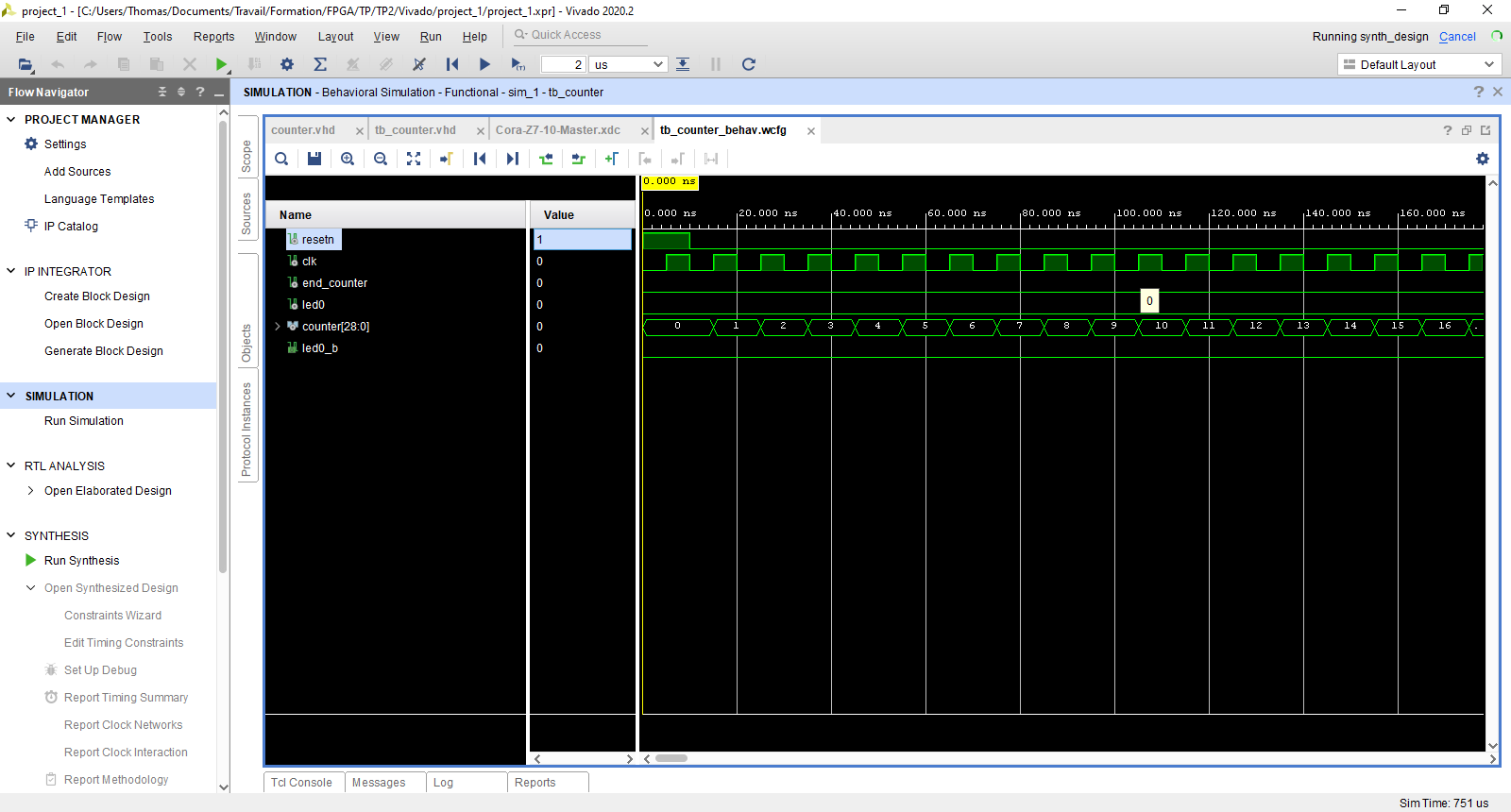
Pour pouvoir réaliser le changement d’état de la led, nous réalisons une bascule flip flop

Figure 6: bascule n-flip-flop pour le changement d'état de LED à la fin d'un compteur

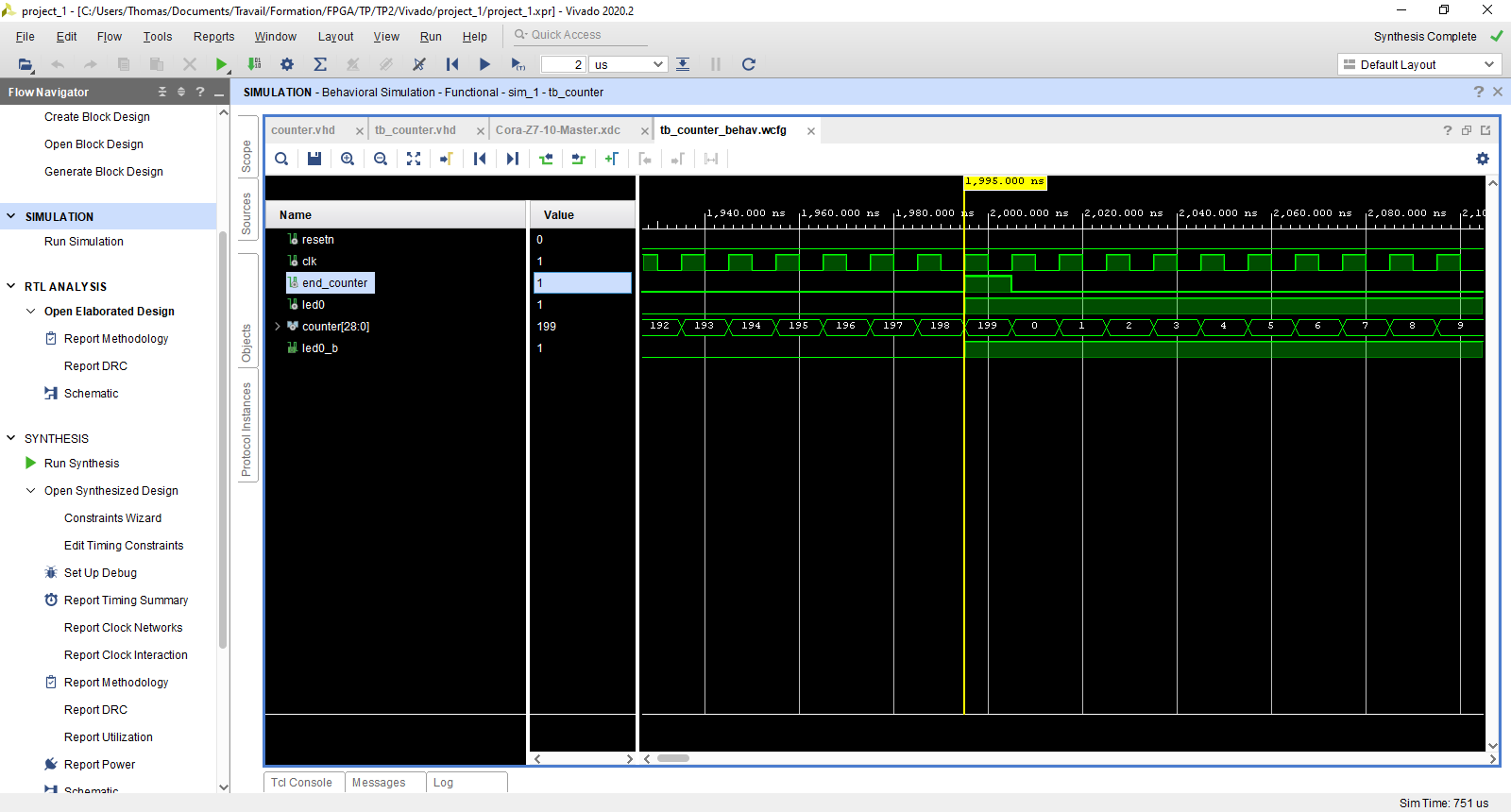
12)

les signaux que nous voulons vérifier :

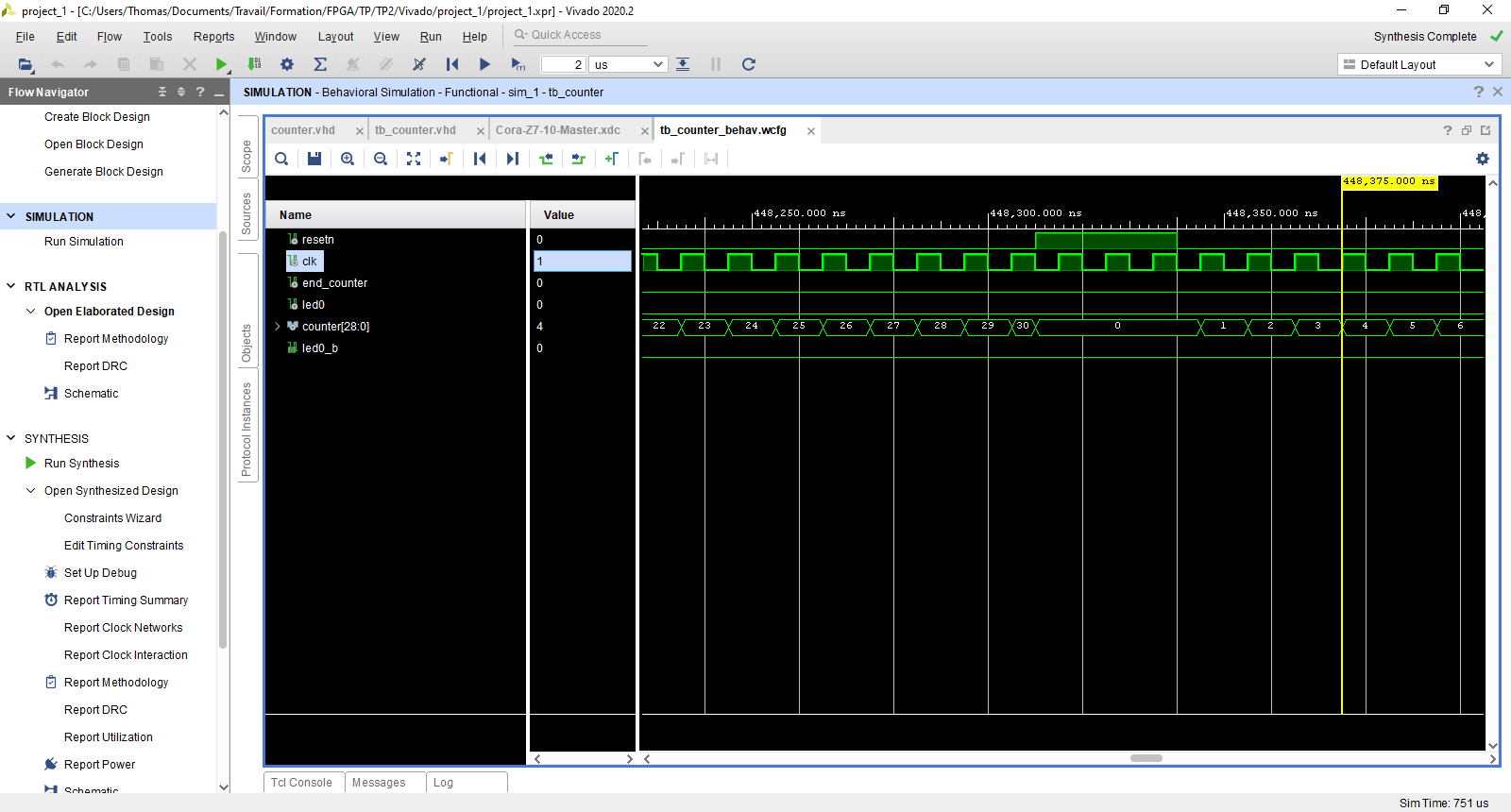
* l’horloge
* le compteur, qui se remet à zéro une fois à la valeur max (ici nous avons pris 199)
* end\_counter : qui passe à 1 quand le compteur repart à zéro
* Led0 la variation de la led, quand compteur max , avec inversion d’état
* bouton Restart : qui relance bien à zéro le compteur
* resetn : qui relance bien le compteur à zéro

Figure 7: initialisation du compteur

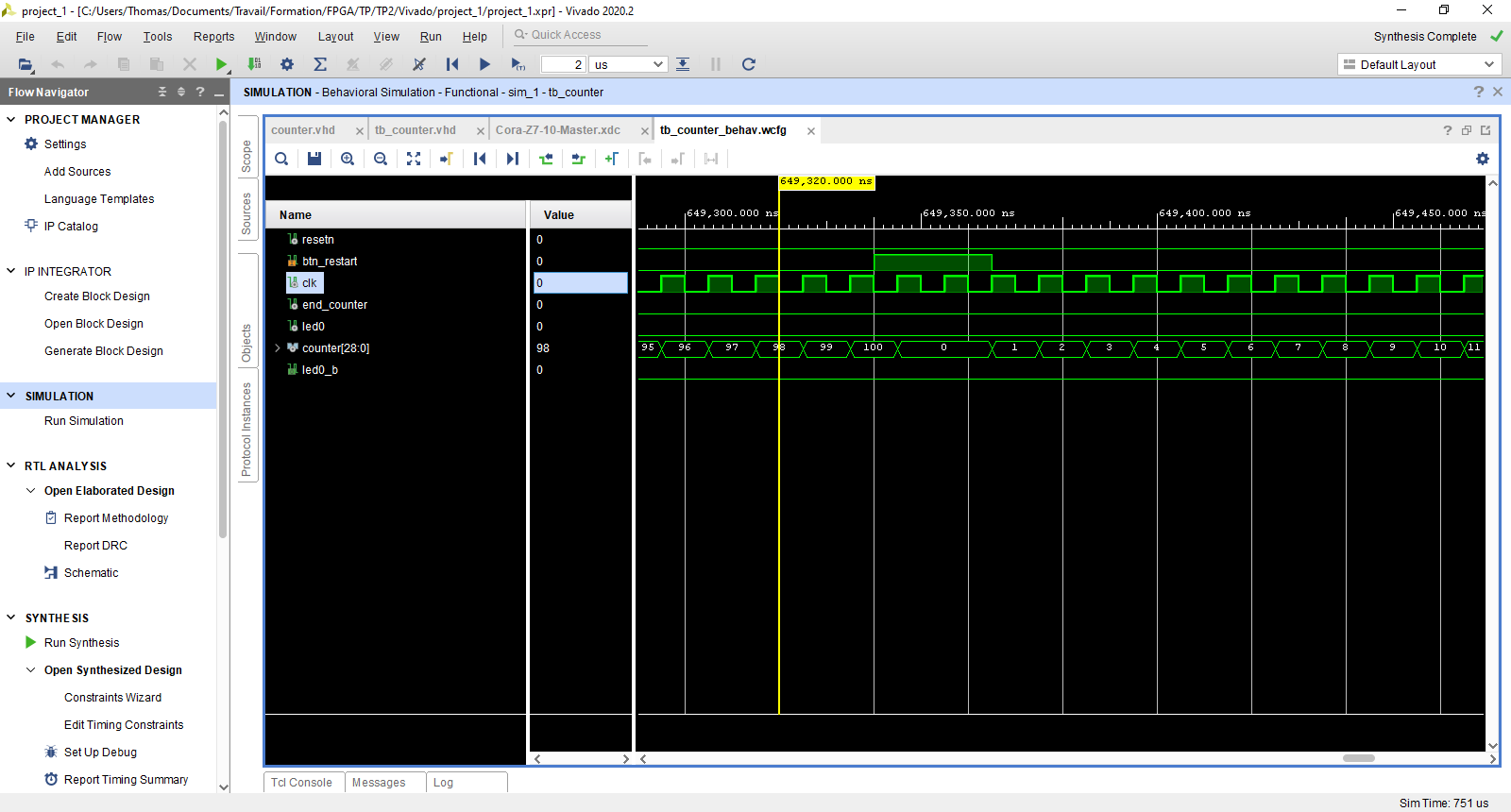
Nous constatons sur la figure ci-dessus que le compteur commence bien à zéro, et s’incrémente sur chaque front montant de l’horloge.

Figure 8: compteur maximum, led remise à zéro

Sur la figure ci-dessus, quand le compteur arrive au max, il repart bien à zéro, end\_counter passe à 1 pour indiquer la fin du compteur, et la led (led0\_b) change d’état.

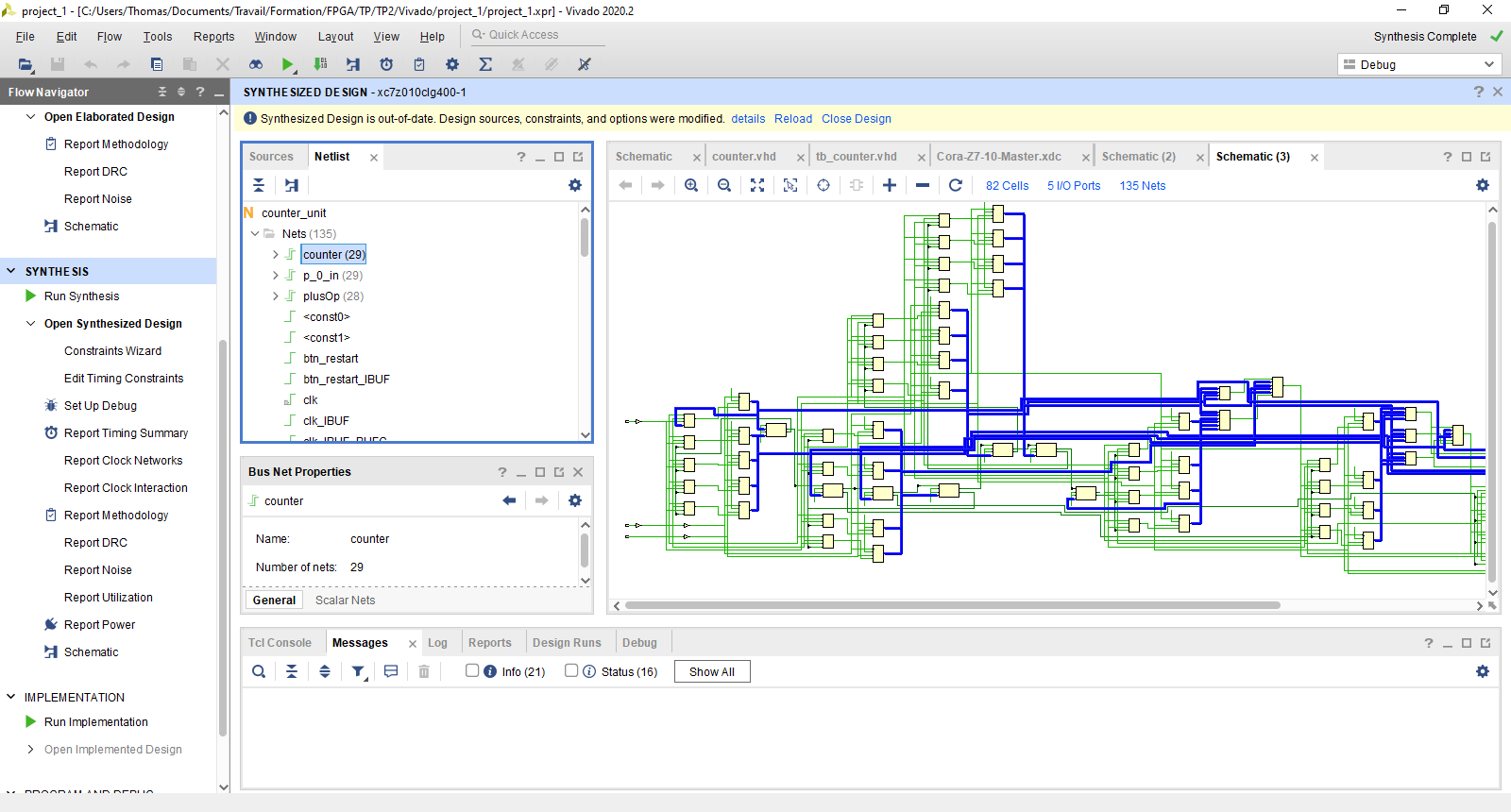
Figure 9: reset du compteur

Sur la figure ci-dessus, nous constatons bien que lorsque le reset est activé, le compteur repart à zéro, et end\_counter ne change pas d’état, ainsi que la led, ce qui correspond au fonctionnement attendu (led change d’état et end\_counter = 1 seulement quand on attend la valeur maximale)

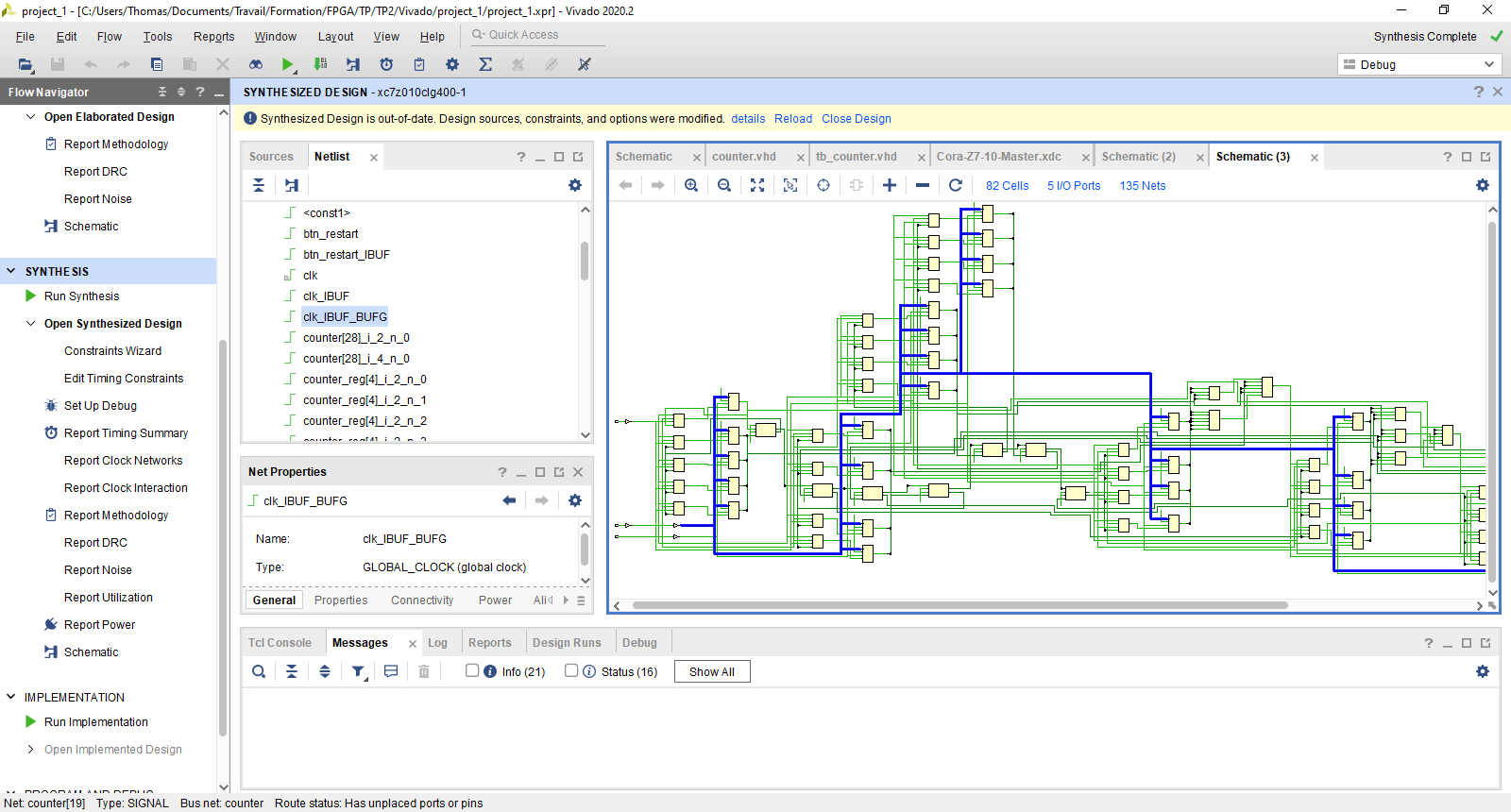
Figure 10: bouton restart pressé

Sur la figure ci-dessus, nous avons avec le bouton restart (bn\_restart) bien une remise à zéro comme avec resetn.

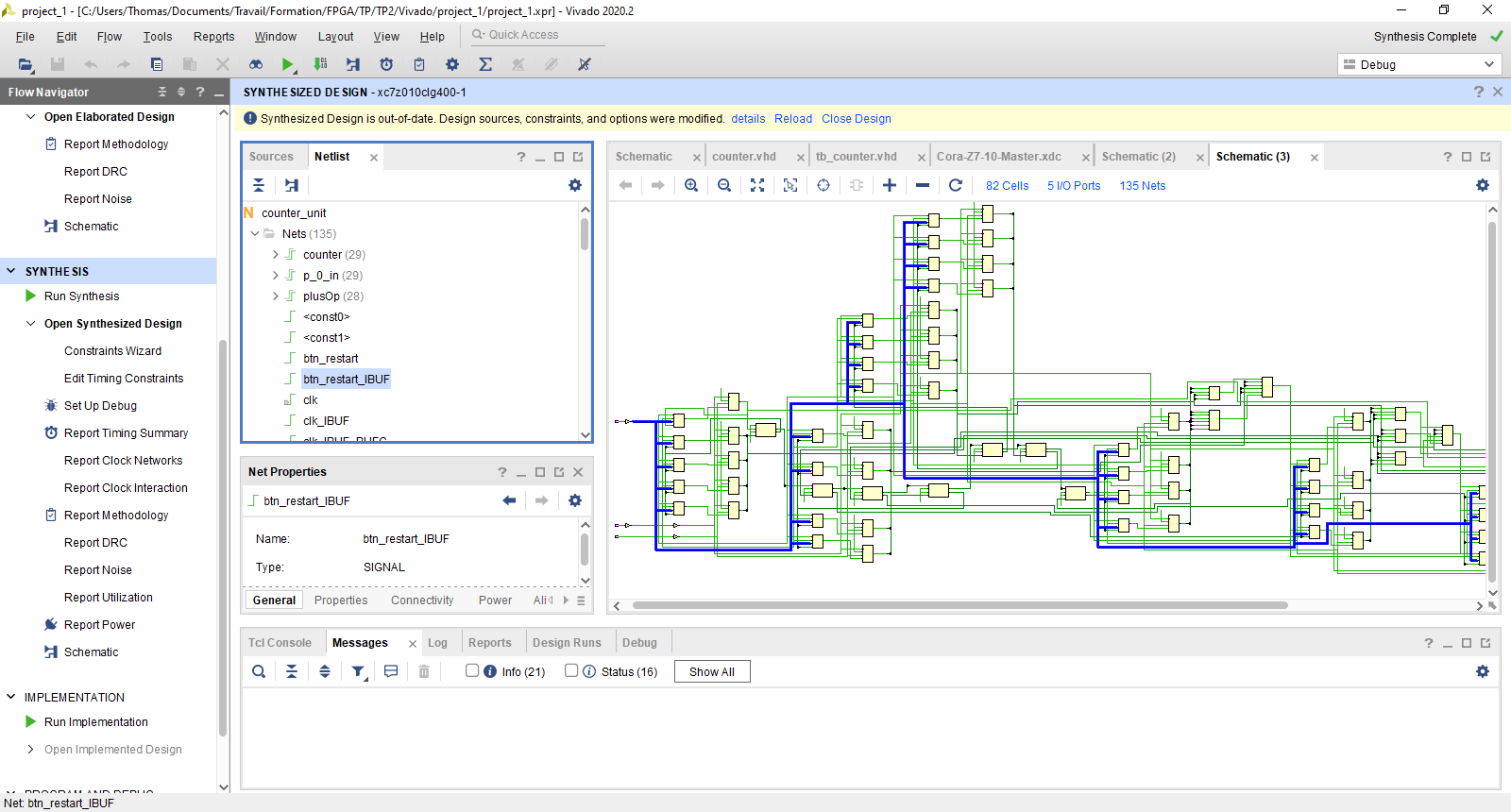
13)

Figure 11: registre 'counter'

Nous avons sur Figure 11 les 29 bascules que compose le compteur de 29 bit dans le cas présent.

Figure 12: signal d'horloge : clk

le signal d’horloge que nous pouvons voir sur la Figure 12, qui est bien relié aux bascules du compteur.

Figure 13: bouton restart

sur la Figure 13, nous voyons le bouton restart relié aux multiplexeurs du compteur

14)

Nous pouvons voir dans le composent un adder avec 2 input de 28 bit qui correspondra au compteur.

2 registres : un de 1 bit, un autre de 28 bit : pour le compteur, puis le passage de end\_counter pour commander la LED

1 multiplexeur, qui permettra de soit compter, soit remettre à zéro le compteur.

---------------------------------------------------------------------------------

Detailed RTL Component Info :

+---Adders :

2 Input 28 Bit Adders := 1

+---Registers :

28 Bit Registers := 1

1 Bit Registers := 1

+---Muxes :

2 Input 28 Bit Muxes := 2

---------------------------------------------------------------------------------

Nous avons également :

|7 |FDCE | 29|

|8 |IBUF | 3|

|9 |OBUF | 2|

soit :

- 29 registres à reset asynchrone (28 pour le compteur, 1 pour le changement de l’état de la led)

- 3 input buffer : 3 entrées : clk, resetn et btn\_restart

- 2 output buffer : end\_counter et led.

16)

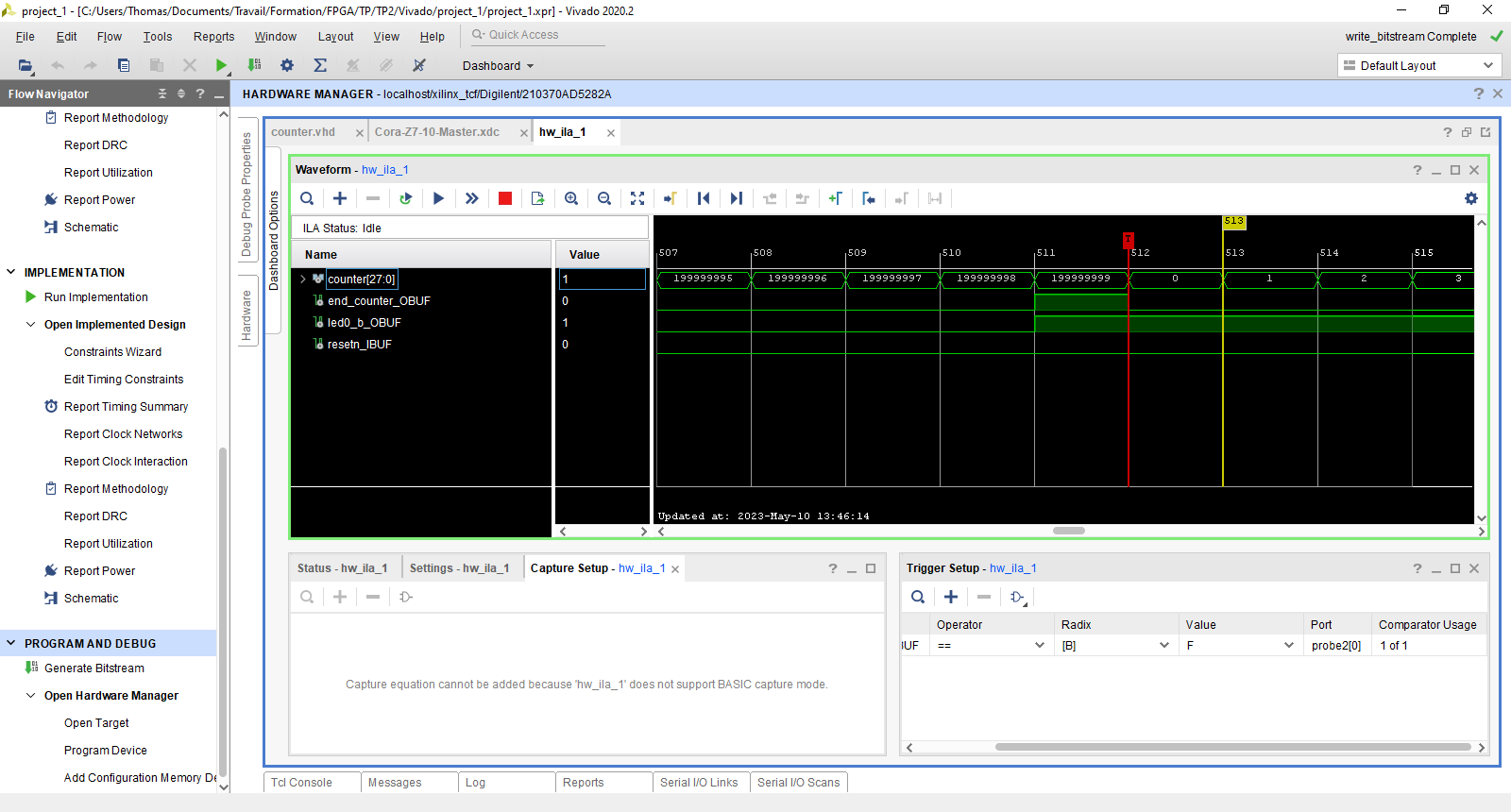
pour le rapport de timing, nous avons TNS (negative slack) et WHS (hold stack) = 0, donc pas de violation de set up.

L’horloge est bien à 100 Mhz :

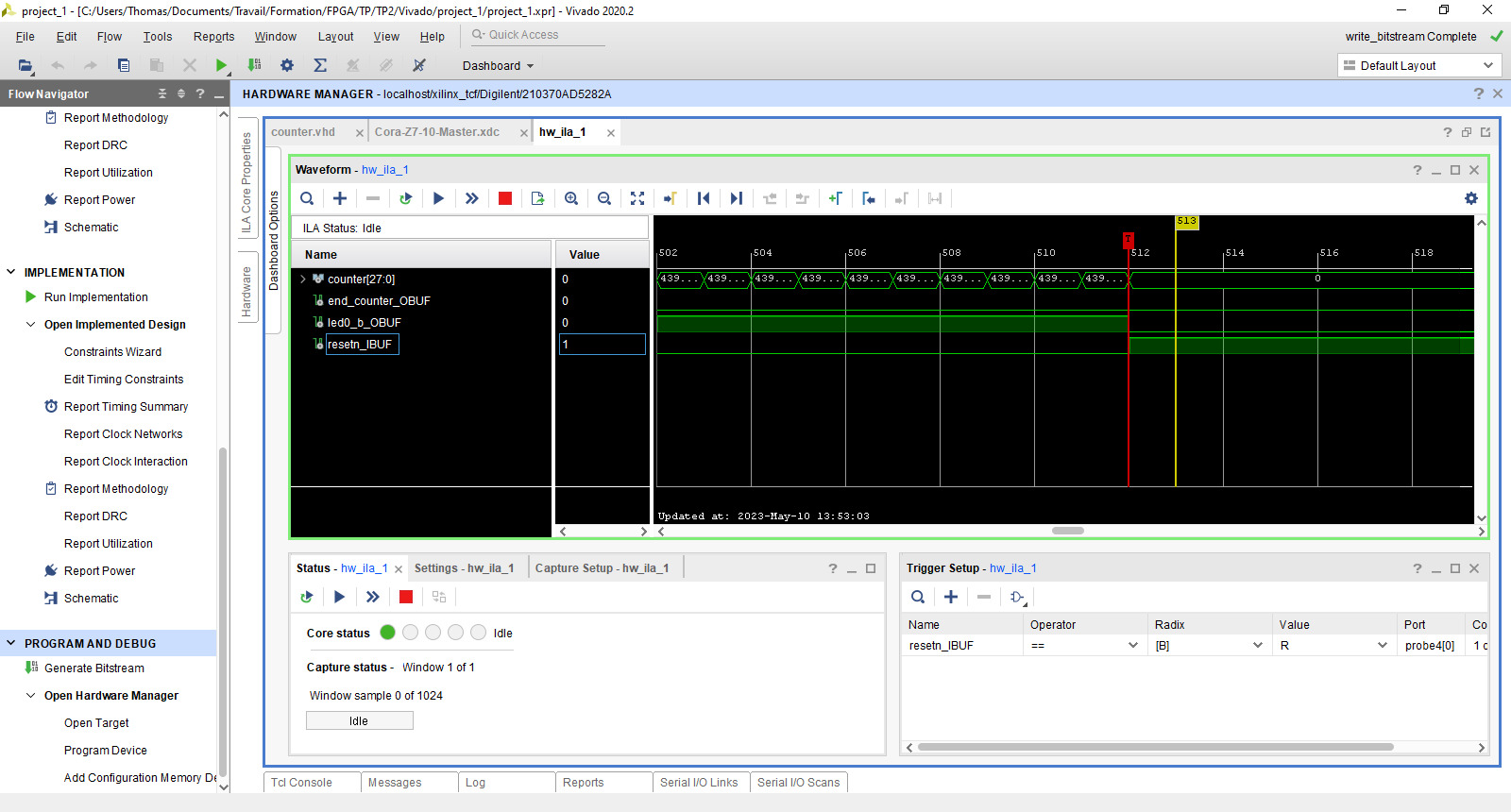
*sys\_clk\_pin {0.000 4.000} 10.000ns 100.000 Mhz*

chemin critique : *Data Path Delay: 4.855ns (logic 0.964ns (19.857%) route 3.891ns (80.143%))*

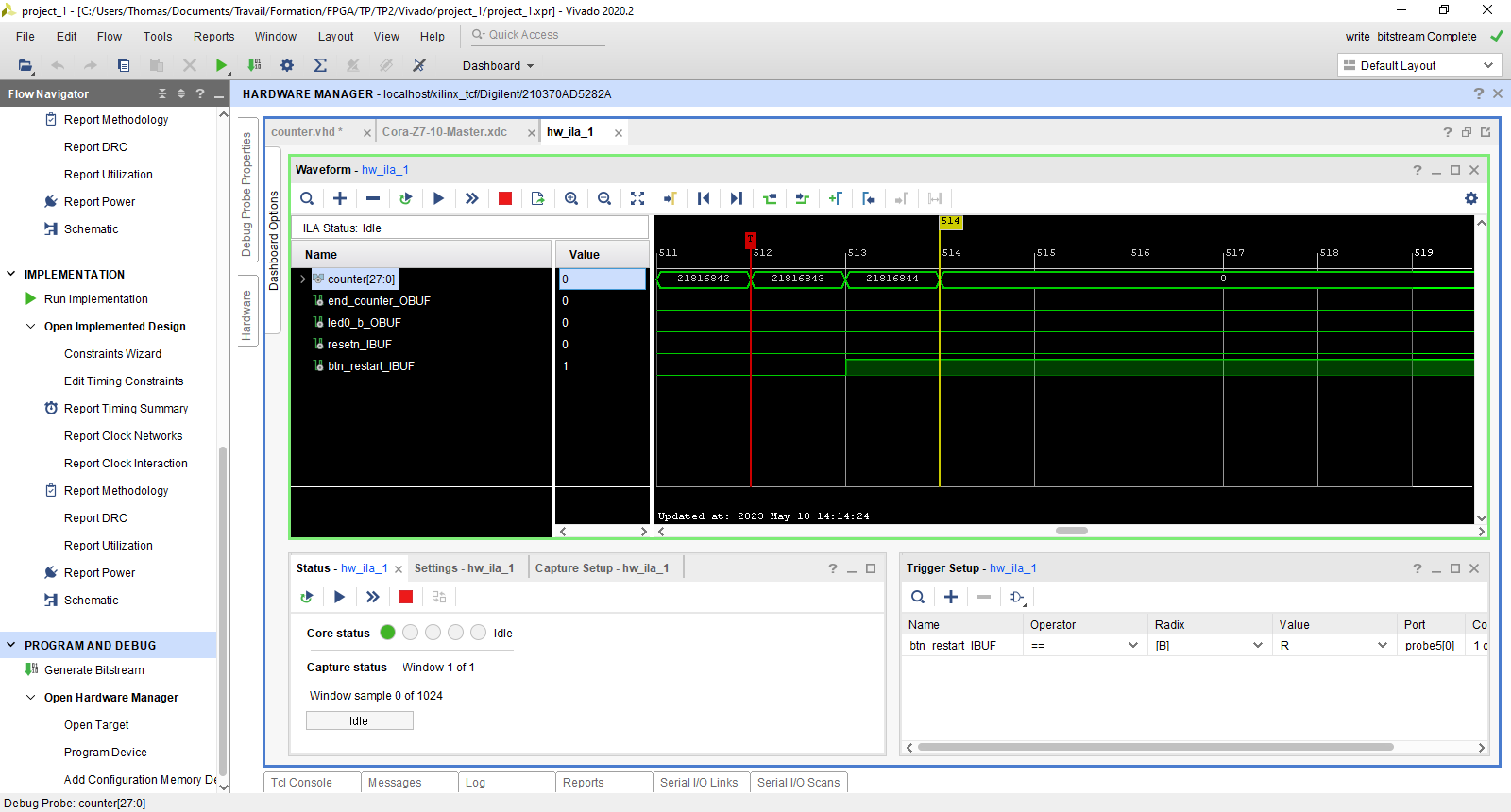
*17)*

Figure 14: end\_counter et changement d'état de led

Conformément au testbench, nous avons bien à la fin du compteur un signal end\_counter = 1 et la led qui change d’état.

Figure 15: reset

Une remise à zéro lors de l’appuie sur le reset.

Figure 16: appuie du bouton restart

lors

Lors du restart, nous avons bien le compteur qui repart à zéro, indépendament de la led.