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 ?**

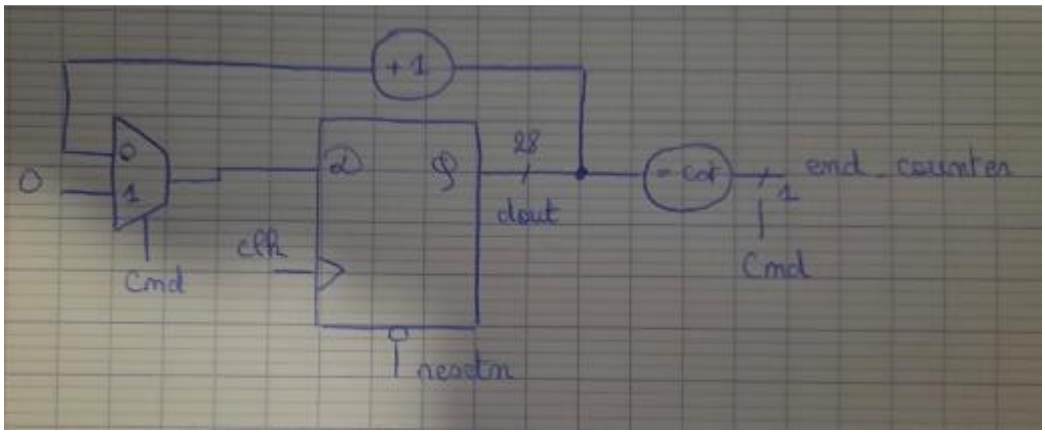
On a F = 100 Mhz, on a donc une période à 10 ns.
Pour attendre 2 secondes il nous faut donc 200 000 000 périodes. Ce qui correspond à un minimum de 28 Bits.

$$\mathrm{f} = \frac{1}{100.10^6} = 10\ ns$$

$$nb\ de\ période = \frac{2\ \text{secondes}}{\text{période}} = \frac{2.10^9}{10} = 200.10^6$$

$$nb\ de\ bits > \log2\ (\text{Nb de période}) > 27.58 = 28\ \text{Bits}$$

2. **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.**



3. **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.

4. **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

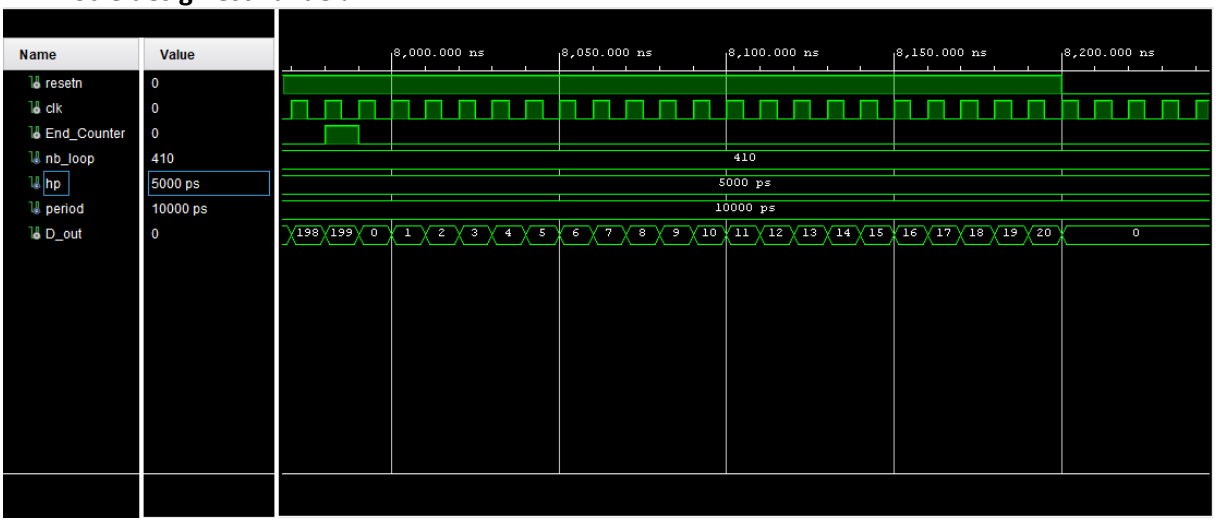
5. **Ecrivez à présent le compteur en VHDL en suivant le schéma RTL, faites attention de bien faire correspondre les noms des signaux de votre code VHDL avec ceux de votre schéma RTL.**

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

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

Voir le fichier tb_coubter.vhd

7. **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.

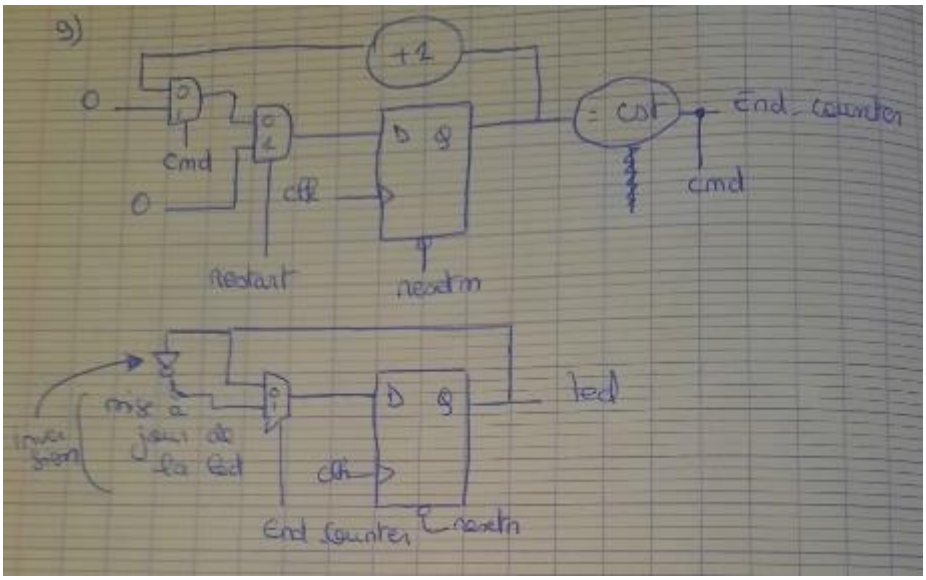
8. **Associez une LED avec le signal de teste d'arrêt du compteur. Pour cela, il faudra ajouter une sortie et la relier à une broche d'une LED dans le fichier de contrainte (*.xdc*). La LED sera alors allumée pendant seulement un coup d'horloge.**

On à modifier le fichier de contrainte comme suit :

```
# RGB LEDs
set_property -dict {PACKAGE_PIN L15 IOSTANDARD LVCMOS33} [get_ports End_Counter]
#set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { LED_Output }]; #IO_L16P_T2_35 Sch=led0_g
#set_property -dict { PACKAGE_PIN N15   IOSTANDARD LVCMOS33 } [get_ports { led0_r }]; #IO_L21P_T3_DQS_AD14P_35 Sch=led0_r
#set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { led1_b }]; #IO_0_35 Sch=led1_b
set_property -dict {PACKAGE_PIN L14 IOSTANDARD LVCMOS33} [get_ports LED_Output]
#set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports { led1_r }]; #IO_L23N_T3_35 Sch=led1_r

# Buttons
set_property -dict {PACKAGE_PIN D20 IOSTANDARD LVCMOS33} [get_ports restart]
set_property -dict {PACKAGE_PIN D19 IOSTANDARD LVCMOS33} [get_ports resetn]
```

9. **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.**



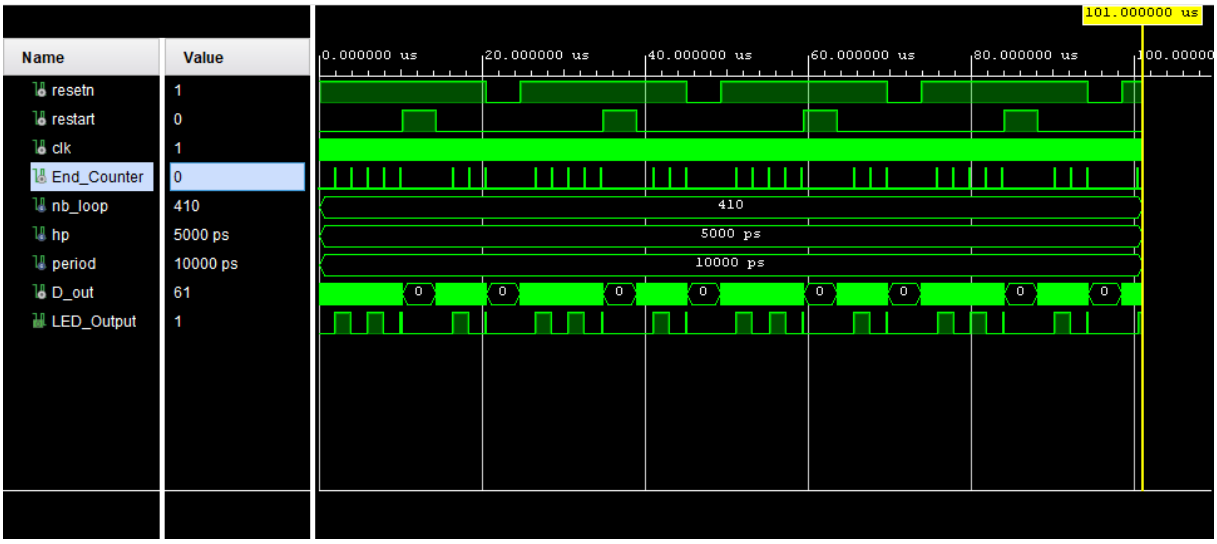
10. **Faites les mises à jour nécessaires sur le code VHDL pour correspondre au nouveau schéma. Le signal *restart* sera une entrée du design.**

Voir fichier Counter.vhd

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

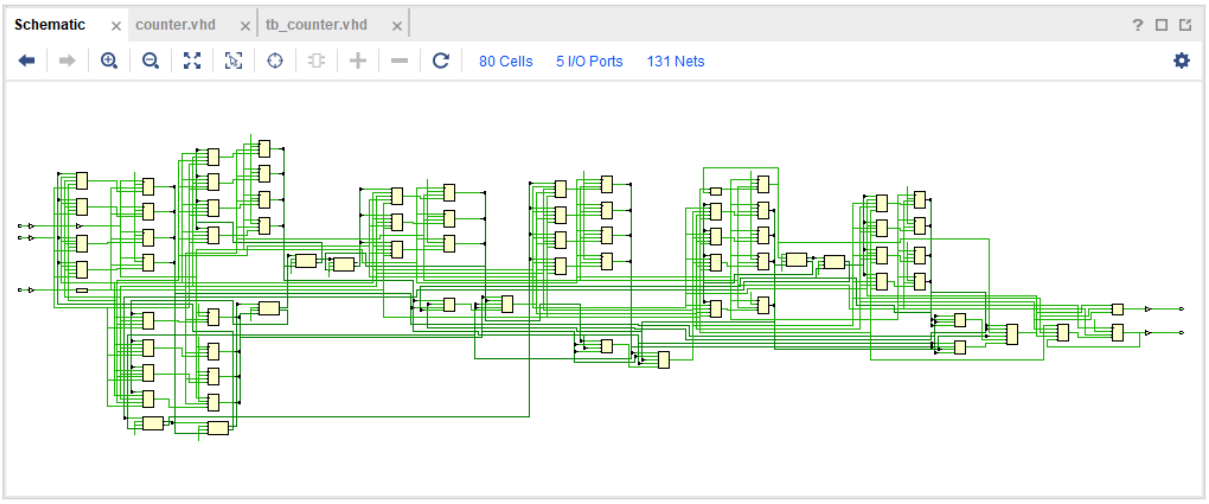
Présent dans la capture de la question 8.

12. **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.

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



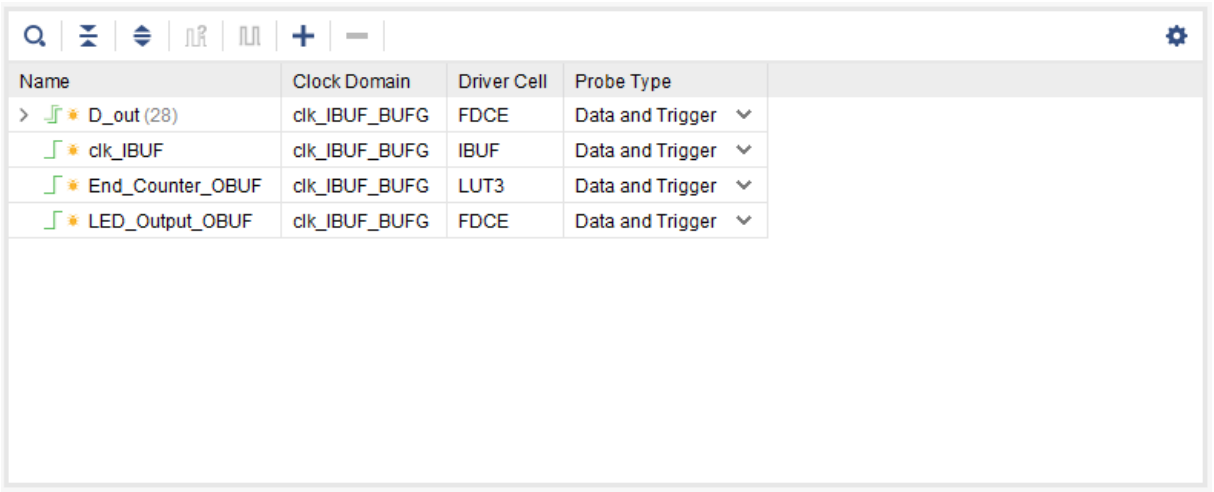
On a un registre par bit, donc 28 registres, A gauche nous retrouvons nos entrées et à droite nos sorties, on retrouve également quelques LUT pour la partie combinatoire.

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

```
73  ---------------------------------------------------------------------------------
74  Start RTL Component Statistics
75  ---------------------------------------------------------------------------------
76  Detailed RTL Component Info :
77  +---Adders :
78         2 Input   28 Bit       Adders := 1
79  +---Registers :
80                    28 Bit    Registers := 1
81                     1 Bit    Registers := 1
82  +---Muxes :
83         2 Input   28 Bit        Muxes := 2
84         2 Input    1 Bit        Muxes := 1
85  ---------------------------------------------------------------------------------
86  Finished RTL Component Statistics
87  ---------------------------------------------------------------------------------
```

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.

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



Name	Clock Domain	Driver Cell	Probe Type
D_out (28)	clk_IBUF_BUFG	FDCE	Data and Trigger
clk_IBUF	clk_IBUF_BUFG	IBUF	Data and Trigger
End_Counter_OBUF	clk_IBUF_BUFG	LUT3	Data and Trigger
LED_Output_OBUF	clk_IBUF_BUFG	FDCE	Data and Trigger

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

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1,612 ns	Worst Hold Slack (WHS):	0,014 ns	Worst Pulse Width Slack (WPWS):	2,750 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	3740	Total Number of Endpoints:	3724	Total Number of Endpoints:	2080

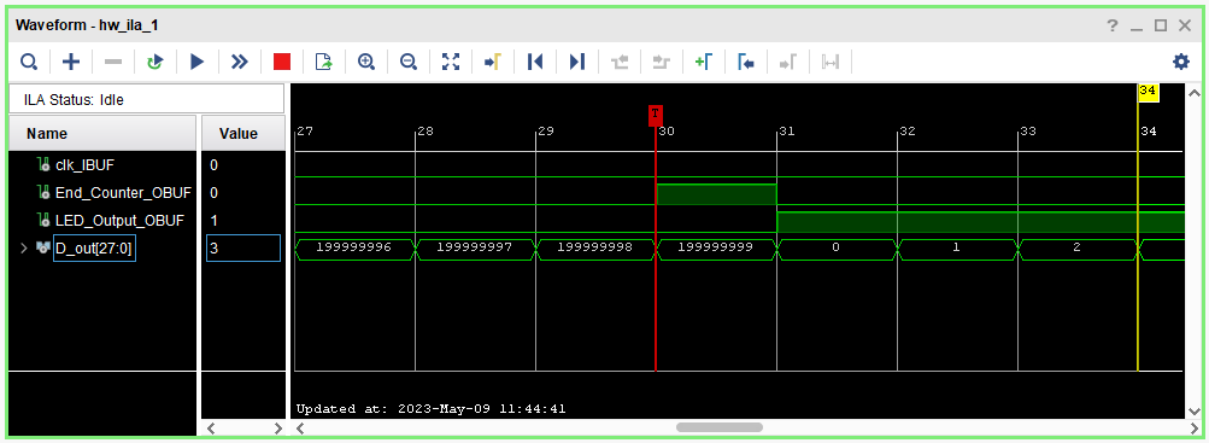
All user specified timing constraints are met.

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

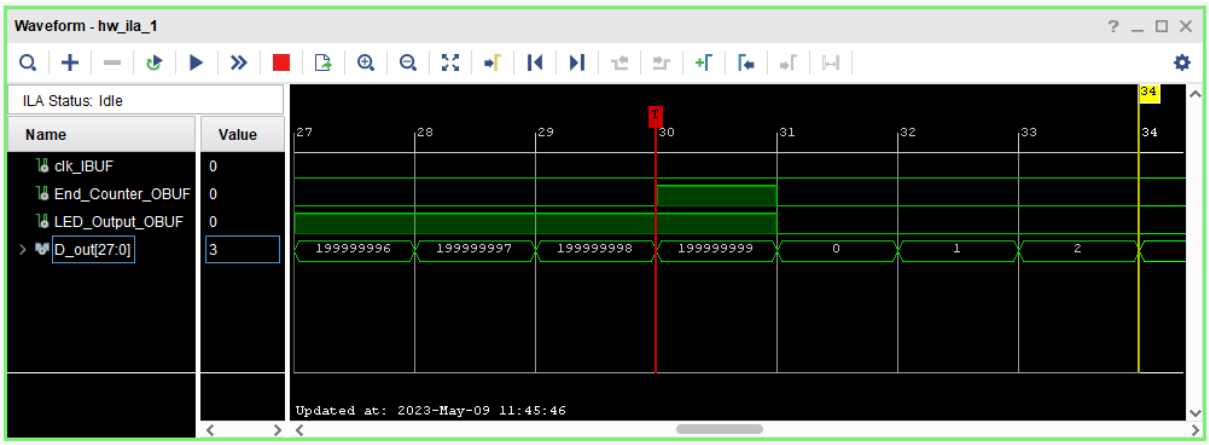
Le chemin critique est :

```
197 | Max Delay Paths
198 | --------------------------------------------------------------------------------------
199 | Slack (MET) :             26.456ns  (required time - arrival time)
200 |   Source:                 dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/state_reg[0]/C
201 |                             (rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bs
202 |   Destination:            dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/state_temp_reg[1]/D
203 |                             (rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bs
204 |   Path Group:             dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK
205 |   Path Type:              Setup (Max at Slow Process Corner)
206 |   Requirement:            33.000ns  (dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/T
207 |   Data Path Delay:        6.513ns  (logic 1.931ns (29.647%)  route 4.582ns (70.353%))
```

17. 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.