Segment 2

# TP2 : Compteur de temporisation

## Objectif

L'objectif de cet TP est faire clignoter une LED en utilisant un compteur de temporisation. Un compteur de temporisation permet de compter le nombre de coup d'horloge nécessaire pour attendre un temps voulu. En connaissant la fréquence de l'horloge il est possible de déterminer combien de périodes d'horloge il faut compter pour attendre 3 secondes par exemple.

## Questions

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 ?

F=100MHz,

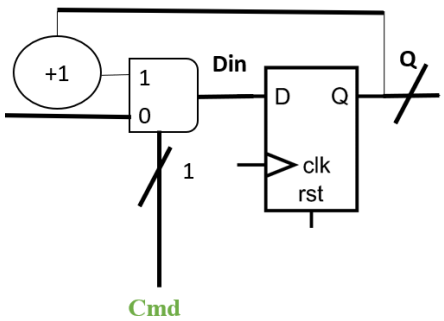
Soit T la période, T=1/F=10ns

Pour atteindre 2 secondes, il faut :

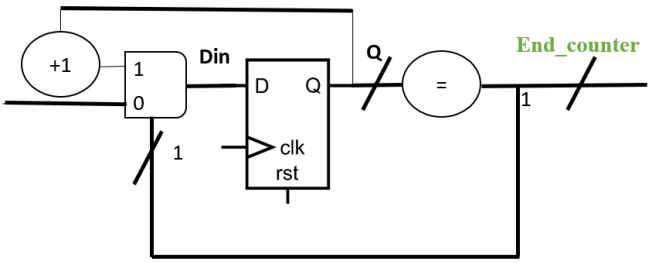
Nbp=2s/10ns= $2*10^8$

Le nombre de période est donc $2*10^8$ et le bits minimum pour représenter cette valeur est de 28.

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




4. Listez les signaux d'entrée, de sortie et les signaux internes de votre architecture.

Entrée
clk : in std_logic;
resetn
Sortie
end_counter

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.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity counter_unit is
  port (
        clk             : in std_logic;
    resetn          : in std_logic;
    -- restart      : in std_logic;
    end_counter     : out std_logic
  );
end counter_unit;

architecture behavioral of counter_unit is
```

```
        --Declaration des signaux internes
   constant cte : positive := 20;
   signal Q : std_logic_vector(27 downto 0);
  signal end_count : std_logic;
begin
                --Partie sequentielle
                process(clk,resetn)
                begin
                        if(resetn = '1') then
                        Q       <=(others=>'0');
                        elsif(rising_edge(clk)) then
                        if(end_count= '1') then
                        -- if(end_count= '1'or restart='1') then
                         Q       <=(others=>'0');
                         else
                           Q <= Q+ std_logic_vector(to_signed(1,28));
                        end if;
                        end if;
                end process;
                --Partie combinatoire
          end_count <= '1' when Q =  std_logic_vector( to_unsigned(cte-1, 28) )
            else '0';
   end_counter <= end_count;
end behavioral;
```

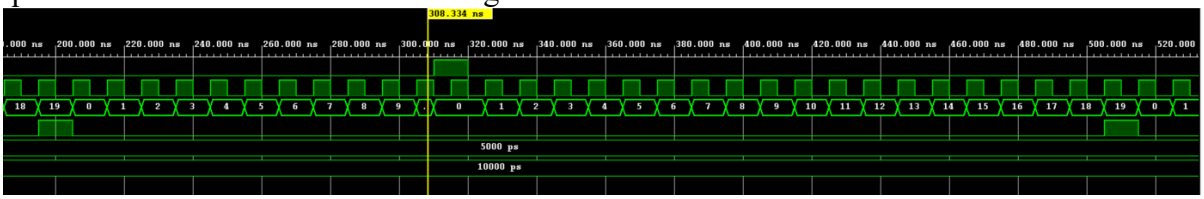
6. Ecrivez un fichier de testbench pour tester votre design.

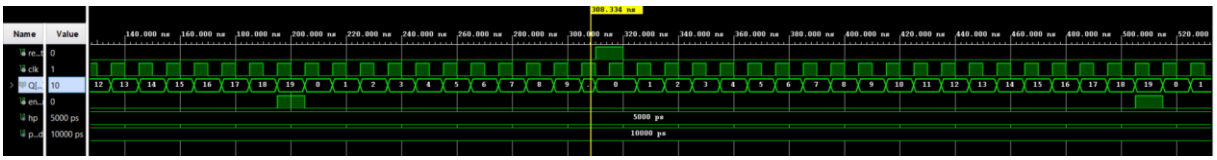
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity tb_counter is
end tb_counter;
```

```
architecture behavioral of tb_counter is

   signal resetn      :  std_logic := '0';
   signal clk         :  std_logic := '0';
 -- signal restart     : std_logic := '0';
   signal end_counter: std_logic;
   -- Les constantes suivantes permette de definir la frequence de l'horloge
   constant hp : time := 5 ns;       --demi periode de 5ns
   constant period : time := 2*hp;  --periode de 10ns, soit une frequence de 100Hz

   --Declaration de l'entite a tester
  component counter_unit
      port (
         clk          : in std_logic;
         resetn       : in std_logic;
       --  restart     : in std_logic;
         end_counter   : out std_logic
       );
 end component;



   begin

   --Affectation des signaux du testbench avec ceux de l'entite a tester
   uut: counter_unit

      port map (
         clk => clk,
         resetn=>resetn,
       --  restart=>restart,
         end_counter => end_counter
      );

   --Simulation du signal d'horloge en continue
   process
   begin
      wait for hp;
      clk <= not clk;
   end process;

   process
   begin

     -- TESTS A EFFECTUER
     resetn <= '1';
```

```
        wait for 10ns;
        resetn <= '0';
    --
    wait for 300 ns;

       end process;

    end behavioral;
```

7. Lancez une simulation. Que devez-vous observez sur votre chronogramme pour vérifier que votre design est valide ?





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.

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.

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

use ieee.numeric_std.all;


entity tb_counter is

end tb_counter;


architecture behavioral of tb_counter is
```

```
 signal resetn      :  std_logic := '0';
 signal clk         :  std_logic := '0';
signal restart     : std_logic := '0';
 signal end_counter: std_logic;
 -- Les constantes suivantes permette de definir la frequence de l'horloge
 constant hp : time := 5 ns;       --demi periode de 5ns
 constant period : time := 2*hp;  --periode de 10ns, soit une frequence de 100Hz

 --Declaration de l'entite a tester
component counter_unit
    port (
        clk          : in std_logic;
        resetn       : in std_logic;
       restart     : in std_logic;
        end_counter   : out std_logic
     );
end component;
 begin

 --Affectation des signaux du testbench avec ceux de l'entite à tester
 uut: counter_unit
        port map (
        clk => clk,
        resetn=>resetn,
        restart=>restart,
        end_counter => end_counter
    );

 --Simulation du signal d'horloge en continue
 process
```

```
   begin
      wait for hp;
      clk <= not clk;
   end process;
   process
   begin
      -- TESTS A EFFECTUER
      resetn <= '1';
      wait for 10ns;
      resetn <= '0';
--
wait for 300 ns;
   end process;
end behavioral;
```

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

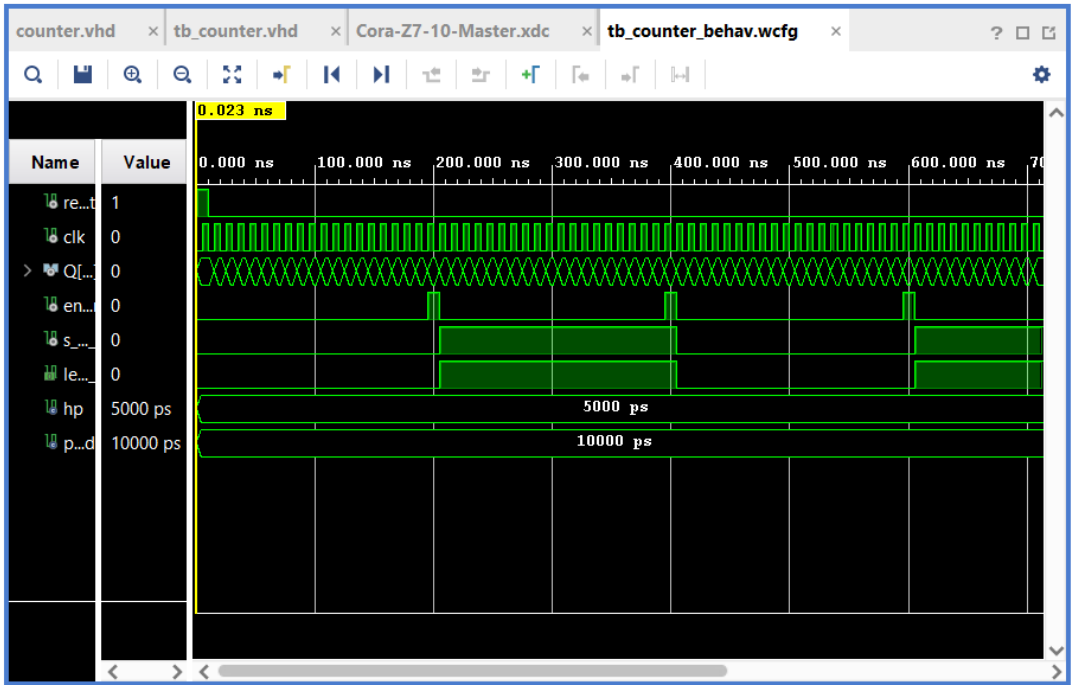
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity counter_unit is
   port (
                clk                     : in std_logic;
      resetn            : in std_logic;
      restart     : in std_logic;
      end_counter       : out std_logic
    );
end counter_unit;
```

```
architecture behavioral of counter_unit is

        --Declaration des signaux internes
    constant cte : positive := 20;
    signal Q : std_logic_vector(27 downto 0);
   signal end_count : std_logic;
begin
                --Partie sequentielle
                process(clk,resetn)
                begin
                        if(resetn = '1') then
                        Q       <=(others=>'0');
                        elsif(rising_edge(clk)) then
                        if(end_count= '1'or restart='1') then
                         Q       <=(others=>'0');
                         else
                           Q <= Q+ std_logic_vector(to_signed(1,28));
                        end if;
                        end if;
                end process;
                --Partie combinatoire
           end_count <= '1' when Q =  std_logic_vector( to_unsigned(cte-1, 28) )
             else '0';
    end_counter <= end_count;
end behavioral;
```

11. Associez la nouvelle entrée restart à un bouton.

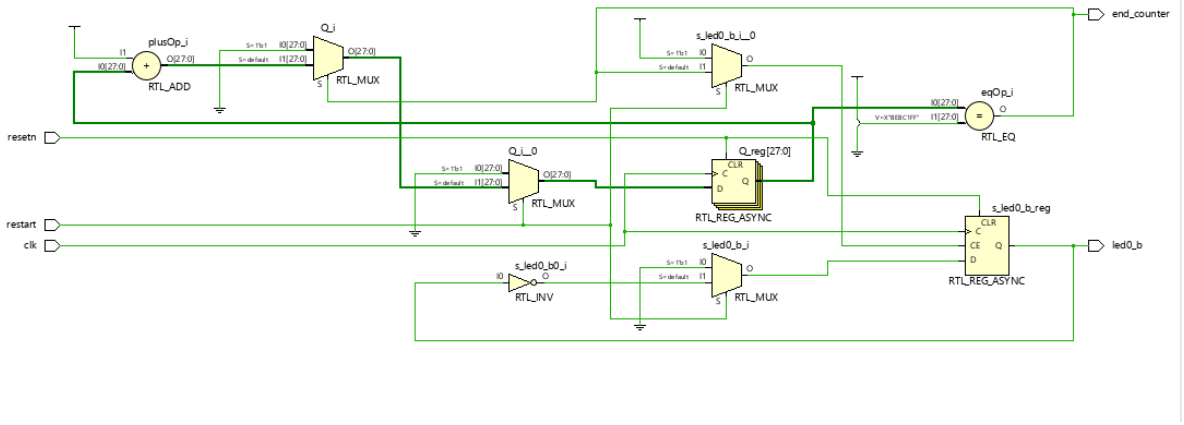
12. Mettez à jour votre testbench puis vérifier votre design avec une simulation. Quels sont les signaux que vous devez observer ?

Clk, led, end_couteur, restart,resetn




Les chronogrammes suivent bien le résultat attendu. On observe bien l'allumage de la led passe à un (allumé) lorsque end_counter passe à 0 (à la fin du comptage). Et respecte bien les 2s avant de se réactiver.

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




On remarque deux additionneurs, 4 multiplexeurs, un registre et un bloc de registres de 28 bits.

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

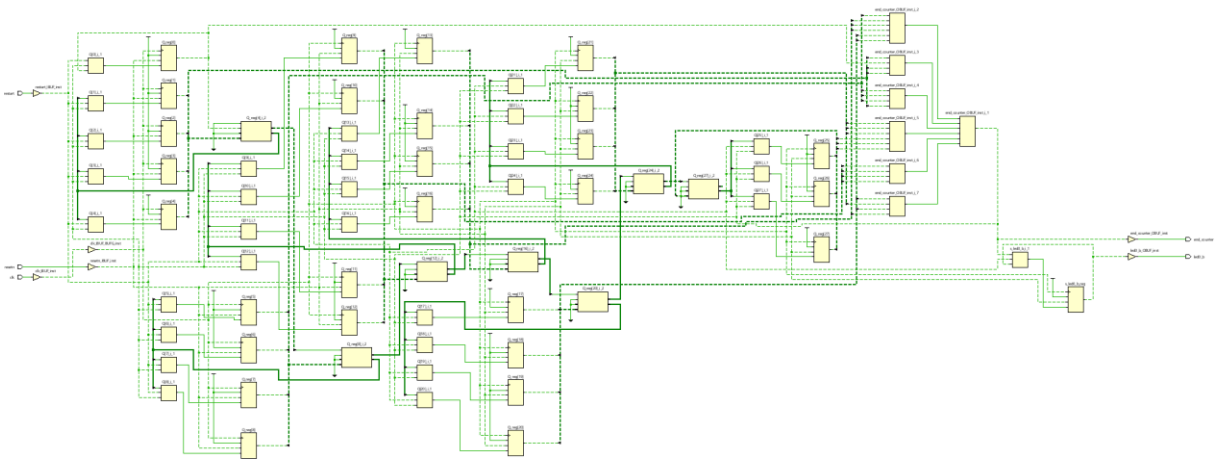
```
Finished RTL Optimization Phase 2 : Time (s): cpu = 00:00:19 ; elapsed = 00:00:28 . Memory (MB): peak = 1001.645
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------------
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
       2 Input    1 Bit        Muxes := 1
```

```
| |BlackBox name |Instances |
+-+--------------+----------+
+-+--------------+----------+

Report Cell Usage:
+------+-------+------+
|      |Cell   |Count |
+------+-------+------+
|1     |BUFG   |     1|
|2     |CARRY4 |     7|
|3     |LUT3   |    29|
|4     |LUT4   |     4|
|5     |LUT6   |     3|
|6     |FDCE   |    29|
|7     |IBUF   |     3|
|8     |OBUF   |     2|
+------+-------+------+
-------------------------------------
```

On Trouve les mêmes éléments sur le RTL que dans le rapport de synthèse.

29 registres à reset , 3 input buffer( resetn,clk,restart) et 2Out buffer(Led et end_conter).

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



On remarque le trajet des signaux sur le schéma.

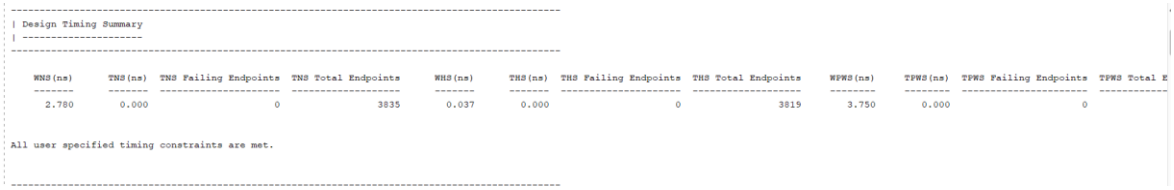
16. Lancez l'implémentation puis étudiez le rapport de timing (vérifiez les violations de set up et de hold et identifiez le chemin critique).

Clock

```
------------------------------------------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------------------------------------------

Clock                                                                                                Waveform(ns)       Period(ns)      Frequency(MHz)
-----                                                                                                ------------       ----------      --------------
dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK           {0.000 16.500}     33.000          30.303
sys_clk_pin                                                                                          {0.000 5.000}      10.000          100.000


------------------------------------------------------------------------------------------------
| Intra Clock Table
| -----------------
------------------------------------------------------------------------------------------------
```

On vérifie bien que la période est à 10ns et la fréquence est de 100MHz

Valeur à 0, il n'y a pas de violation du set up et du hold.



```
------------------------------------------------------------------------------------------------
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------------------------

    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints      WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints     WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total E
    -------      -------  ---------------------  -------------------      -------      -------  ---------------------  -------------------     --------     --------  ---------------------  ------------
      2.780        0.000                      0                 3835        0.037        0.000                      0                 3819        3.750        0.000                      0

All user specified timing constraints are met.


------------------------------------------------------------------------------------------------
```

Chemin critique :

Source: dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/state_reg[0]/C
(rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK {rise@0.000ns fall@16.500ns period=33.000ns})
Destination: dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/portno_temp_reg[4]/D
(rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK {rise@0.000ns fall@16.500ns period=33.000ns})

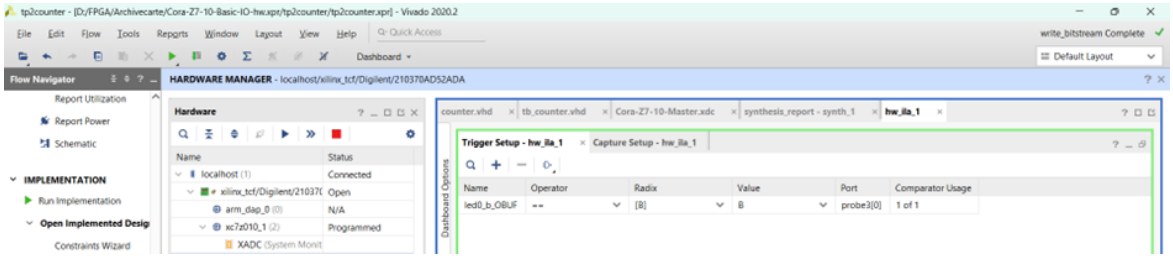
```
Max Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             26.626ns  (required time - arrival time)
  Source:                 dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/state_reg[0]/C
                            (rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_in
  Destination:            dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/portno_temp_reg[4]/D
                            (rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_in
  Path Group:             dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            33.000ns  (dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK ris
  Data Path Delay:        6.318ns  (logic 1.843ns (29.170%)  route 4.475ns (70.830%))
  Logic Levels:           5  (CARRY4=1 LUT3=1 LUT4=1 LUT5=1 LUT6=1)
  Clock Path Skew:        -0.052ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    3.067ns = ( 36.067 - 33.000 )
    Source Clock Delay      (SCD):    3.494ns
    Clock Pessimism Removal (CPR):    0.375ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
```

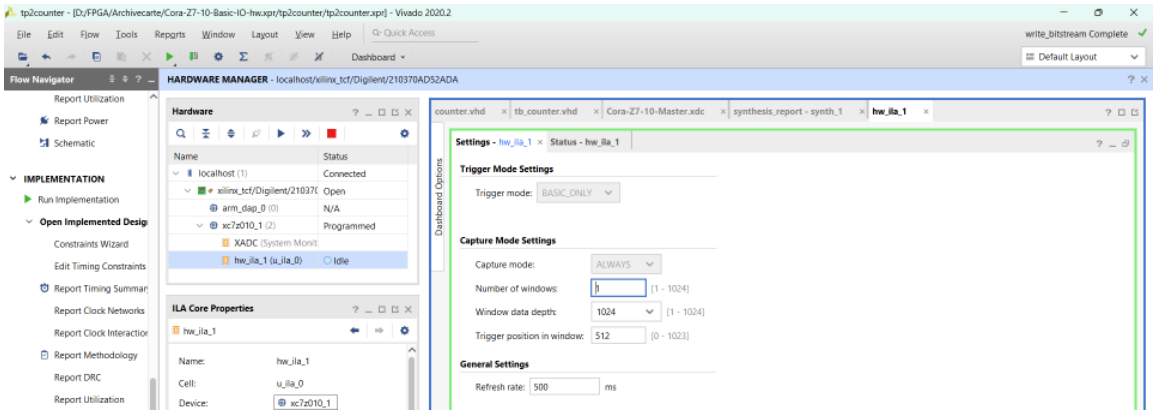
17. Générez le bitstream pour observer le système sur carte. Relevez les résultats de la ILA.

Les signaux led, et couteur,

hw_ila_1

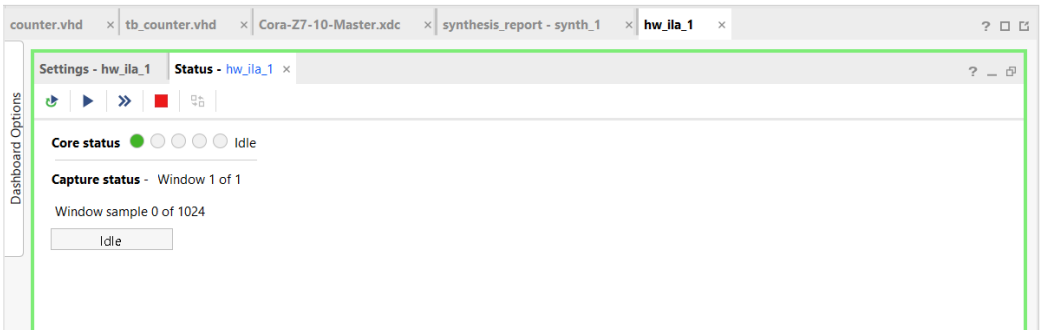
Nous demarrons l'enrégistrement de l'ILA sur le front montant et descendant du port de sortie de la led et du compteur. On procède à un changement de valeur sur le signal de déclenchement entraînera l'ILA pour commencer à enregistrer les signaux sondés (Led, et Q). Ceci est fait dans le déclencheur (trigger setup).




Cliquez sur le déclencheur Exécuter bouton (play). l'ILA se déclenche et enregistre les signaux

.




Nous voyons la ligne verticale rouge (marqueur) sur le front montant de notre signal de déclenchement (port trigger de led), et il est en position 512. Nous pouvons également vérifier que le signal compte se comporte correctement et le signal end_counter et le compteur Q.