Compte rendu Eve CHAR du 04/05/2023 – 09/05/202

Compte rendu de – TP02 – Compteurs

1.1 Objectif de ce TP

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.

1) Question 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 ?

T (période) = 1/Fréquence

- F= 100Mhz donc T= 1/F = 10^-8 secondes = 10*10^-9 secondes = 10 ns
- Pour attendre 2 secondes il faut compter **N***10ns donc N=2*10^8 ➔ Il faut 200 millions d'impulsions pour attendre les deux secondes.

N=200 000 000

- Pour calculer le nombre de bit :

Nbr de bit = Log2(2x10^8)= 27,5 donc il faut 28 bits. Il nous faut donc au minimum 28 Bits.

Nbr de bit =28 bits

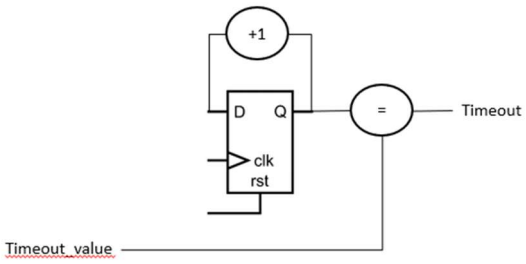
2) Question 2 : Dessinez le schéma RTL de ce compteur. Si le compteur atteint la valeur calculée précédemment, un signal end_counter passe à 1, sinon end_counter vaut 0. N'oubliez pas de mettre sur chaque signal son nombre de bits. Commencez par réaliser une boucle d'incrémentation : +1 à chaque coup d'horloge.

Comme vu en cours : je me base sur le principe de **watchdogs** qui est un compteur qui permet d'interrompre un processus lorsque ce dernier ne réponds pas au bout d'un certain temps (ici 2 secondes par exemple).

Pour observer notre compteur à la sortie, quand on arrive à 200 000 000 front montant d'horloge, on va utiliser un comparateur à la sortie.

Si les deux valeurs d'entrées sont égales, alors la sortie est à 1 sinon elle est à 0.

➔il s'agit d'un timer ou watchdog



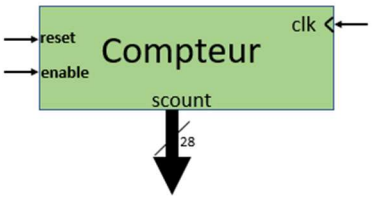
Compteur (Composant synchrone)

Composant Synchrone avec une reset asynchrone –> process avec sensibilité sur (clk, reset)

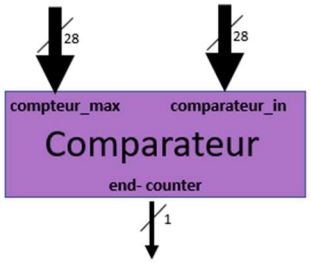
Comparateur (composant combinatoire)

Compteur = valeur maximum ➔ alors : signal end_counter passe à 1, sinon end_counter vaut 0

3) Question 3 et Question 4 et Question 5.

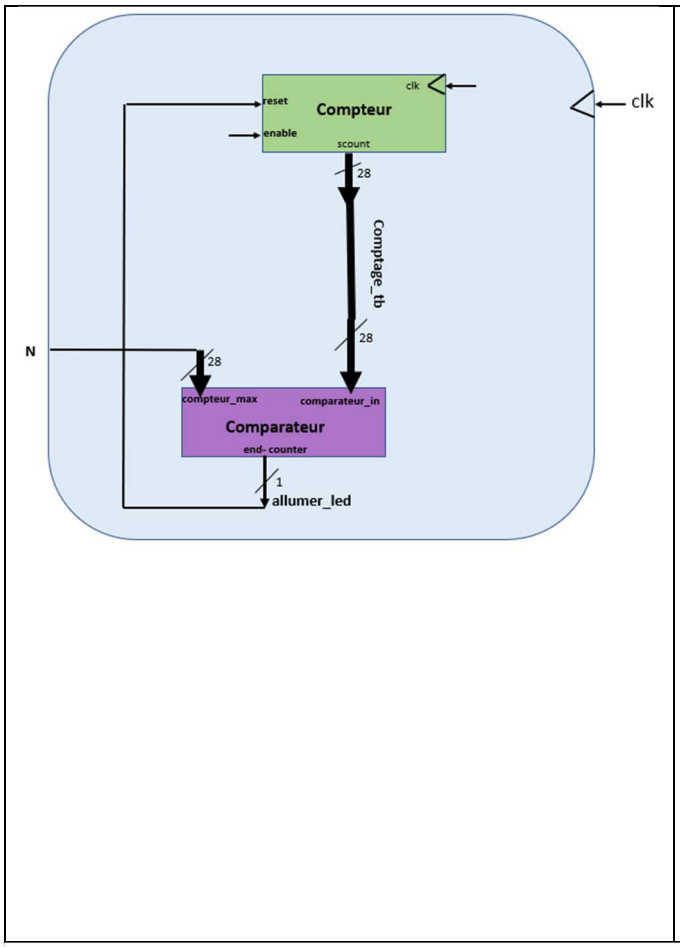



```
---tp compteur-temporisation 04/05/2023:
-- Declaration de ma bibliothèque
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

--## declaration premier composant : compteur ##

entity compteur is
    Port ( clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           enable : in STD_LOGIC;
           scount : out STD_LOGIC_VECTOR (27 downto 0));
end compteur;

----Description comportementale

architecture arch_compteur of compteur is

 signal comptage : integer := 0; -- indicateur de comptage interne

begin
    process (clk,Reset,enable)
    begin                   -- Comptage à chaque front montant de l'horloge
        if rising_edge (clk) then
            if Reset ='1' then
                comptage <= 0;
            elsif enable ='1' then
             -- pas de condition de fin de comptage et remise à zéro.
                comptage <= comptage + 1;
            end if;
        end if;
    end process;

 -- scount recopie valeur finale du compteur = la sortie du comptage

 scount <= std_logic_vector(to_unsigned(comptage,28));
```

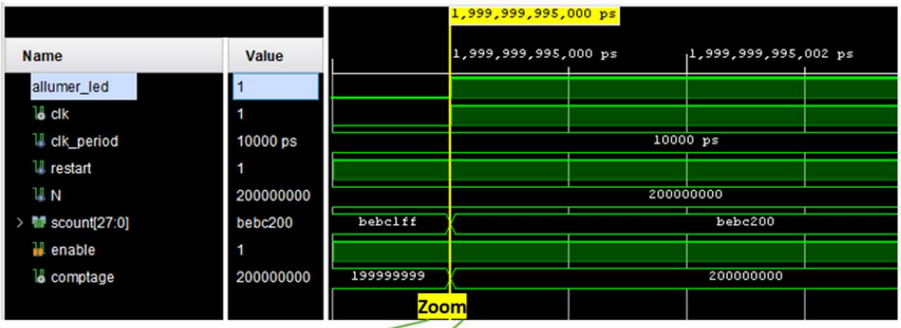



```
 ----------------------------------------------------------
---tp compteur-temporisation 04/05/2023:
-- Declaration de ma bibliothèque
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

--## --declaration deuxième composant : comparateur

-- Declaration de ma bibliothèque
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity comparateur is

    Port ( comparateur_in : in STD_LOGIC_VECTOR (27 downto 0);
           compteur_max : in STD_LOGIC_VECTOR (27 downto 0);
           end_counter : out STD_LOGIC);
end comparateur;

--Description comportementale
-----------------------------------------------
 architecture arch_comparateur of comparateur is

begin -- on va Comparer la valeur de comparateur_in avec la valeur de comparateur_max

     end_counter <= '1' when comparateur_in = compteur_max else '0';

end arch_comparateur;
```

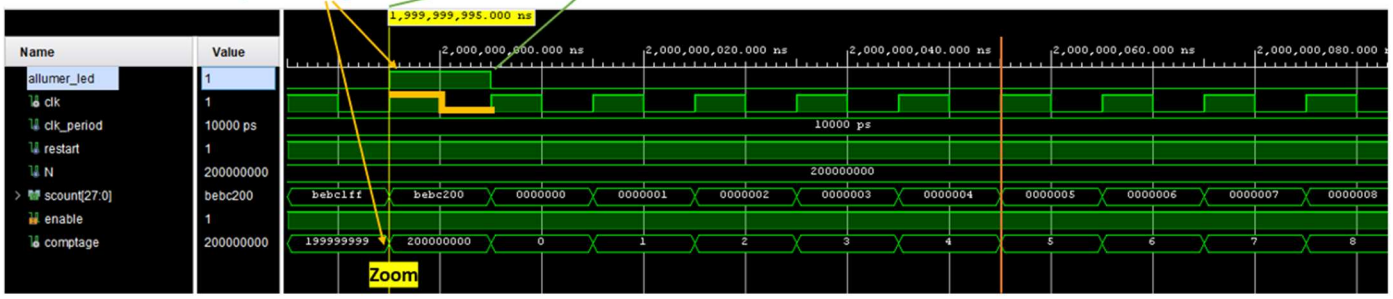
4) Question 6 et Question 7 et Question 8




```
--Le fichier de test-bench :
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE work.ALL;
 ----Description externe**************************
ENTITY compteur_temporisation_tb IS
PORT();
End compteur_temporisation_tb ;

ARCHITECTURE arch_compteur_temporisation_tb OF compteur_temporisation_tb IS

-- ## Component Declaration for the Unit Under Test (UUT)
    constant N : integer := 200000000;

-----déclaration des signaux Inputs ****************
    signal clk : std_logic := '0';

------déclaration des signaux Outputs **************
    signal allumer_led : std_logic := '0';
    signal comptage_tb : std_logic_vector (27 downto 0);

----- Clock period definitions ********************
    constant clk_period : time := 10 ns;
  Begin
   -- Instantiate the Unit Under Test (UUT)
    uut_compteur : entity compteur port map(
          clk => clk,
          Reset => allumer_led,
          enable => '1',
          scount => comptage_tb );
   uut_comparateur : entity comparateur port map(
          comparateur_in => comptage_tb,
          --compteur_max => N,
          compteur_max => std_logic_vector(to_unsigned(N,28)),
          end_counter => allumer_led );
 --------- Stimulus process*************************
    ---Clock process definitions
    clk_process : process
    begin
    clk <= '0';
    wait for clk_period/2;
    clk <= '1';
    wait for clk_period/2;
    end process;
end arch_compteur_temporisation_tb;
```



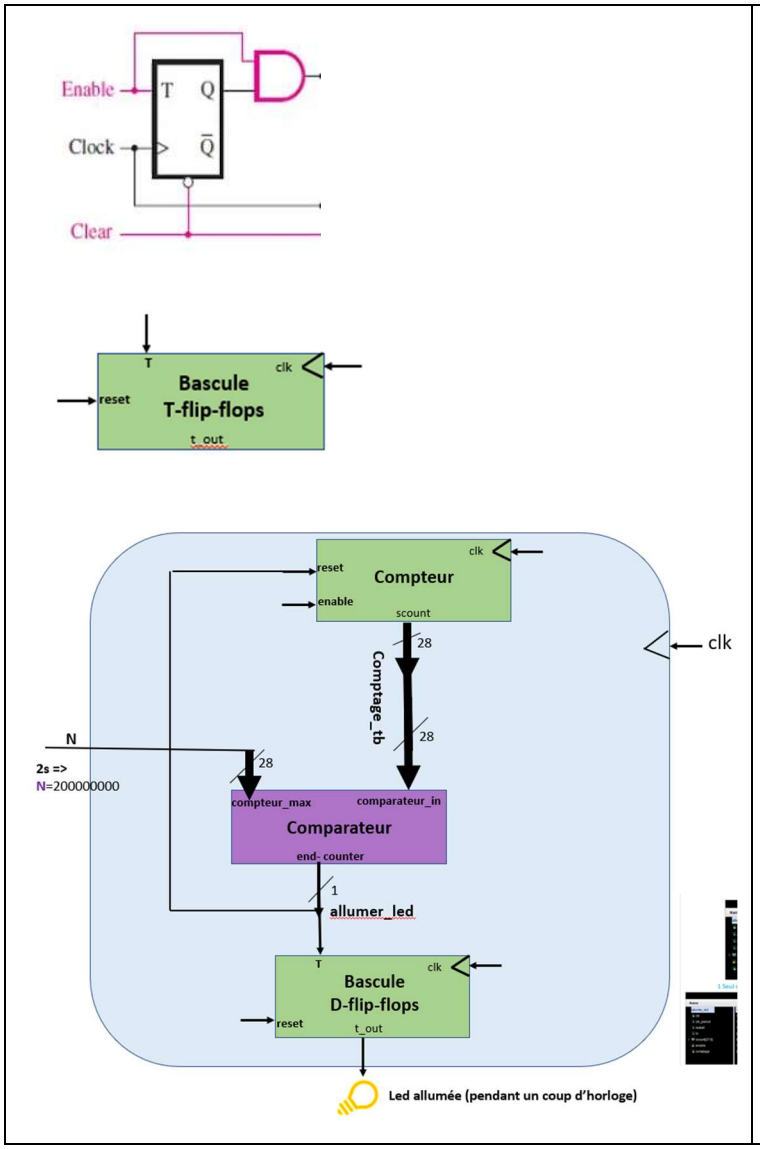
1 Seul coup d'horloge



5) Question 9 et Question 10 et Question 11

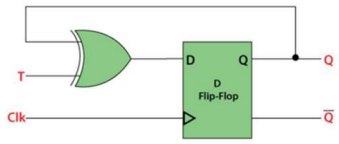
- LED clignote telle que : allumée 2s, éteinte 2s.
- signal restart sera une entrée du design associé à un bouton

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




```
--Le fichier de test-bench :
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE work.ALL;

 ----Description externe*****************************
ENTITY compteur_temporisation_tb IS
PORT(led0_b : inout STD_LOGIC );
End compteur_temporisation_tb ;

ARCHITECTURE arch_compteur_temporisation_tb OF compteur_temporisation_tb IS

-- ## Component Declaration for the Unit Under Test (UUT)
    constant N : integer := 200000000;

-----déclaration des signaux Inputs *****************
    signal clk : std_logic := '0';
    signal restart : std_logic := '0';

------déclaration des signaux Outputs ***************
    signal allumer_led : std_logic := '0';
    signal comptage_tb : std_logic_vector (27 downto 0);
  --signal led0_b : STD_LOGIC := '0';

----- Clock period definitions *********************
    constant clk_period : time := 10 ns;

Begin

 -- Instantiate the Unit Under Test (UUT)
   uut_compteur : entity compteur port map(
          clk => clk,
          Reset => allumer_led,
          enable => '1',
          scount => comptage_tb );

   uut_comparateur : entity comparateur port map(
          comparateur_in => comptage_tb,
          --compteur_max => N,
          compteur_max => std_logic_vector(to_unsigned(N,28)),
          end_counter => allumer_led );

    uut_Bascule_T_Flip_Flops : entity Bascule_T_Flip_Flops port map(
          clk => clk,
          Reset => restart,
          T => allumer_led,
          t_out => led0_b);

--------- Stimulus process*************************

    ---Clock process definitions
    clk_process : process
    begin
    clk <= '0';
    wait for clk_period/2;
    clk <= '1';
    wait for clk_period/2;
    end process;

end arch_compteur_temporisation_tb;
```

Pour cela on peut utiliser T-flip-flops schéma RTL, vu en cours : Sur front montant d'horloge, si l'entrée du registre est à 1, alors la valeur courante du registre est inversée. Autrement, la valeur du registre est inchangée. Si Ti=1 ➔ on a Qi= not(Qi-1) Sinon Ti=0 ➔ on a Qi= Qi-1

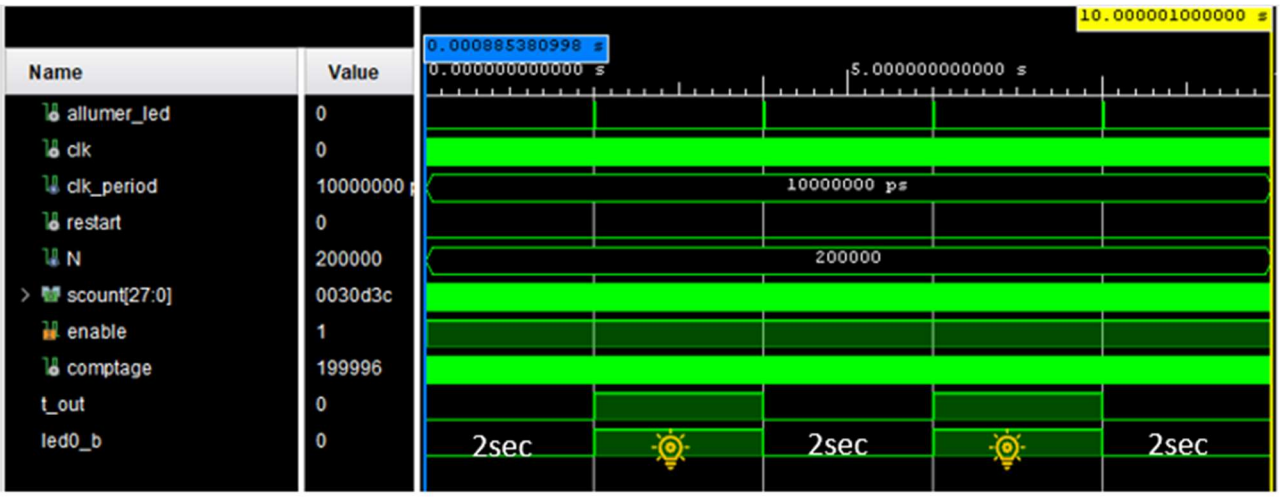


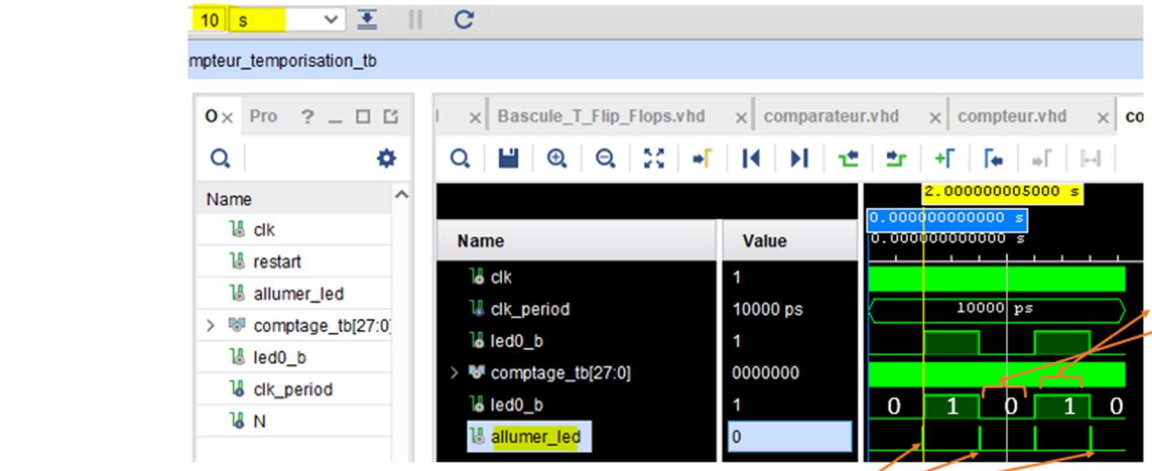
La table de vérité associée :

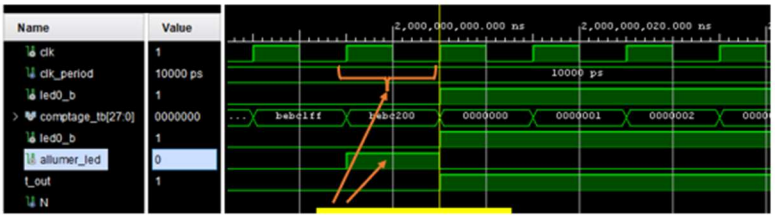
CLK	T	Q n+1
↑	0	Q n
↑	1	Q n'

Le nombre de bascules utilisées dans la conception du compteur série détermine le nombre maximal de comptage : Ce nombre sera égal à '2' exposants le nombre de bascules utilisées ➔ ici on aura **28 bascules** T flip-flop pour attendre 2 secondes.

Démonstration Vivado + explication:

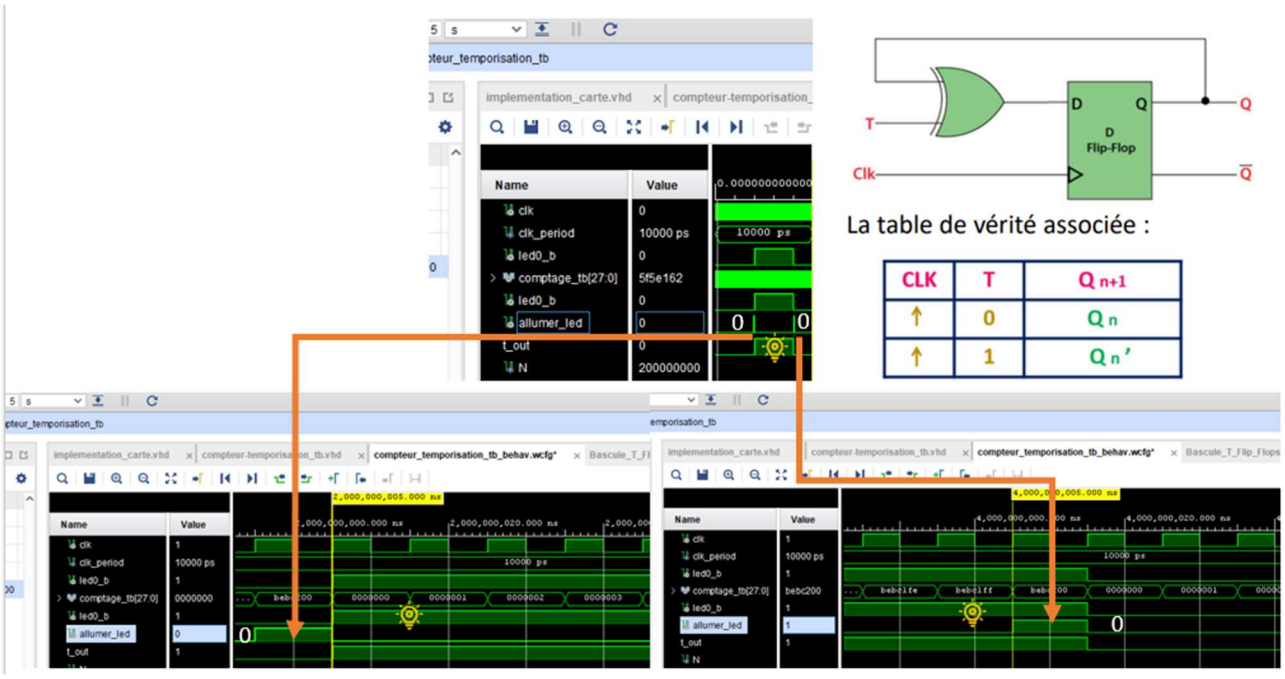






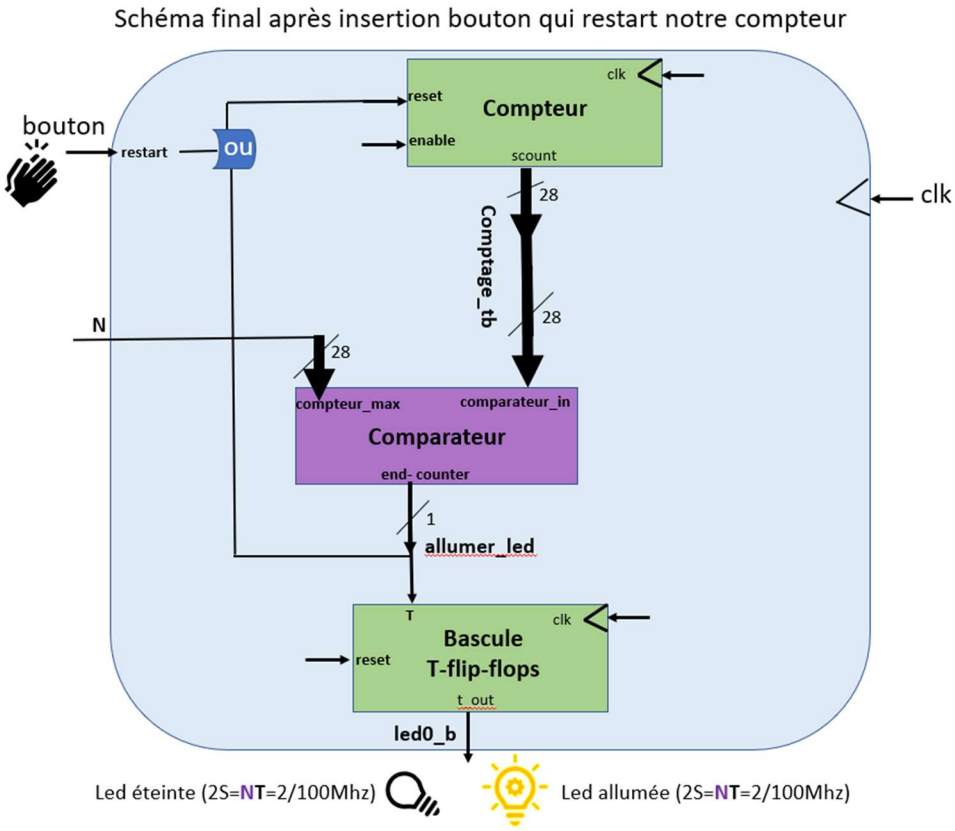
Allumer led passe à 1 pendant **un coup d'horloge** à la fin de chaque comptage 200000000.
La bascule T flip-flop maintien l'etat 1 (led allumée) pendant 200000000 coup d'horloge, ensuite au prochain état 1 elle inverse l'état à 0 et le maintient pendant 200000000 coup d'horloge

Suite :



Allumer led passe à 1 à la fin de chaque comptage 200000000.Ensuite notre bascule T flip-flop maintien l'etat 1 (led allumée) pendant 200000000 coup d'horloge, ensuite au prochain état 1 elle inverse l'état à 0 et le maintient pendant 200000000 coup d'horloge

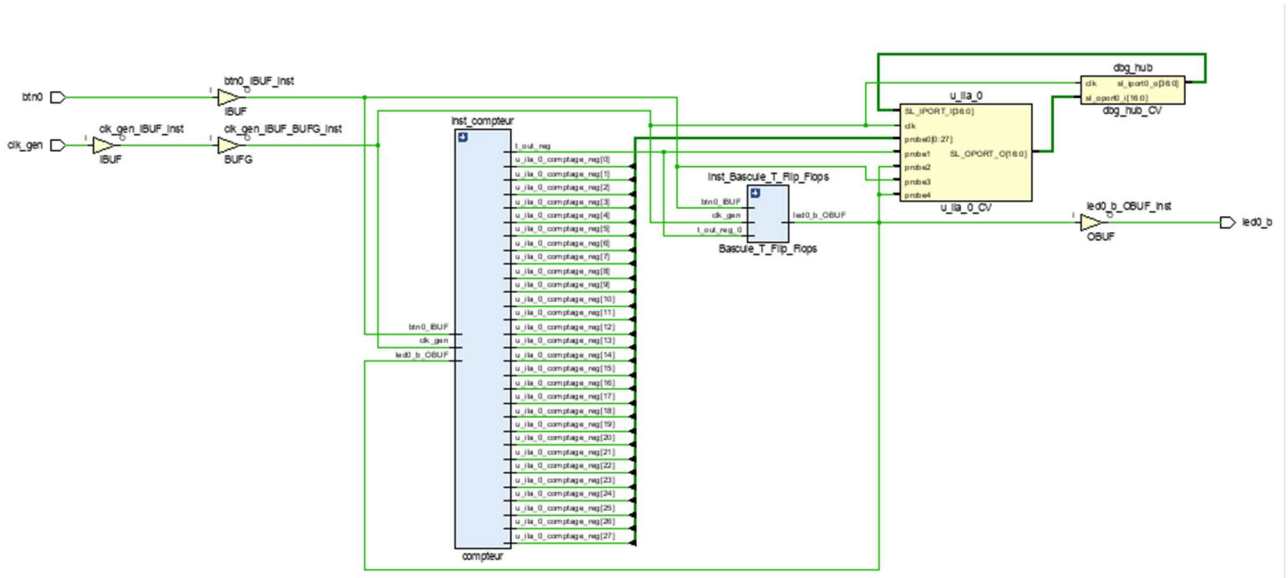
6) Question 12 : Mettez à jour votre test Bench puis vérifier votre design avec une simulation. Quels sont les signaux que vous devez observer ?

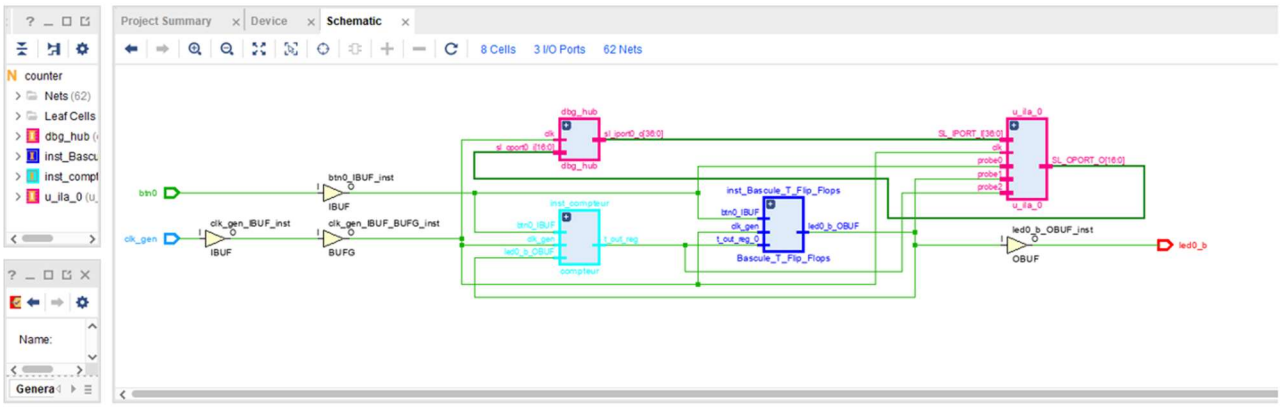


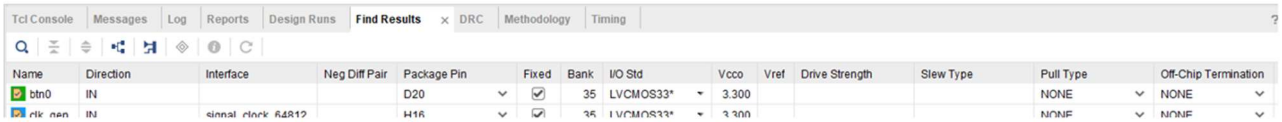
Signale à observer : signal allumer_led & signal comptage-tb & signal led0_b_intern.

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

1.2 Schématique :



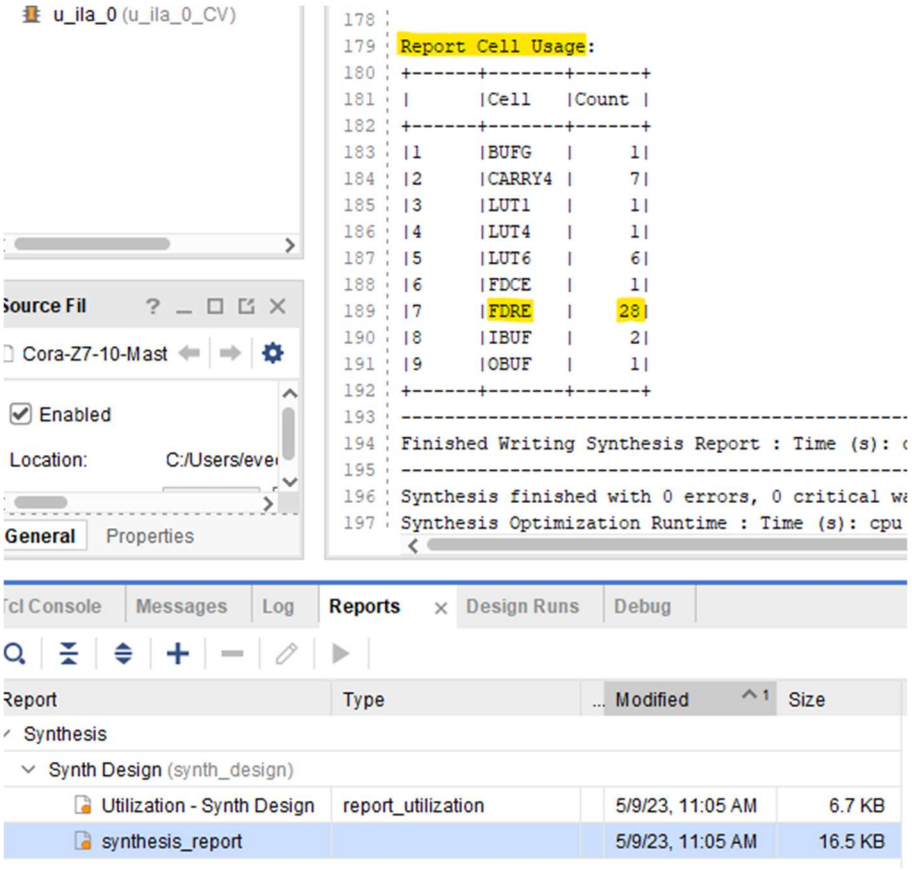




Name	Cell	Cell Pin Co...
btn0_IBUF_inst	IBUF	2
clk_gen_IBUF_BUFG_inst	BUFG	2
clk_gen_IBUF_inst	IBUF	2
dbg_hub	dbg_hub	55
inst_Bascule_T_Flip_Flops	Bascule_T_Flip_Flops	4
inst_compteur	compteur	4
led0_b_OBUF_inst	OBUF	2
u_ila_0	u_ila_0	58

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

1.3 Rapport de synthèse




```
Report Cell Usage:
+------+--------+------+
|      |Cell    |Count |
+------+--------+------+
|1     |BUFG    |     1|
|2     |CARRY4  |     7|
|3     |LUT1    |     1|
|4     |LUT4    |     1|
|5     |LUT6    |     6|
|6     |FDCE    |     1|
|7     |FDRE    |    28|
|8     |IBUF    |     2|
|9     |OBUF    |     1|
+------+--------+------+
Finished Writing Synthesis Report : Time (s): c
Synthesis finished with 0 errors, 0 critical wa
Synthesis Optimization Runtime : Time (s): cpu
```

```
Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists
```

```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |    8 |     0 |     17600 |  0.05 |
|   LUT as Logic          |    8 |     0 |     17600 |  0.05 |
|   LUT as Memory         |    0 |     0 |      6000 |  0.00 |
| Slice Registers         |   29 |     0 |     35200 |  0.08 |
|   Register as Flip Flop |   29 |     0 |     35200 |  0.08 |
|   Register as Latch     |    0 |     0 |     35200 |  0.00 |
| F7 Muxes                |    0 |     0 |      8800 |  0.00 |
| F8 Muxes                |    0 |     0 |      4400 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

```
5. Clocking
-----------

+------------+------+-------+-----------+-------+
| Site Type  | Used | Fixed | Available | Util% |
+------------+------+-------+-----------+-------+
| BUFGCTRL   |    1 |     0 |        32 |  3.13 |
| BUFIO      |    0 |     0 |         8 |  0.00 |
| MMCME2_ADV |    0 |     0 |         2 |  0.00 |
| PLLE2_ADV  |    0 |     0 |         2 |  0.00 |
| BUFMRCE    |    0 |     0 |         4 |  0.00 |
| BUFHCE     |    0 |     0 |        48 |  0.00 |
| BUFR       |    0 |     0 |         8 |  0.00 |
+------------+------+-------+-----------+-------+
```

```
1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     | _            | -           | -            |
| 0     | _            | -           | Set          |
| 0     | _            | -           | Reset        |
| 0     | _            | Set         | -            |
| 0     | _            | Reset       | -            |
| 0     | Yes          | -           | -            |
| 0     | Yes          | -           | Set          |
| 1     | Yes          | -           | Reset        |
| 0     | Yes          | Set         | -            |
| 28    | Yes          | Reset       | -            |
+-------+--------------+-------------+--------------+
```

```
2. Memory
---------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| Block RAM Tile |    0 |     0 |        60 |  0.00 |
|   RAMB36/FIFO* |    0 |     0 |        60 |  0.00 |
|   RAMB18       |    0 |     0 |       120 |  0.00 |
+----------------+------+-------+-----------+-------+
```

```
3. DSP
------

+-----------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+-----------+------+-------+-----------+-------+
| DSPs      |    0 |     0 |        80 |  0.00 |
+-----------+------+-------+-----------+-------+
```

```
7. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| FDRE     |   28 | Flop & Latch        |
| CARRY4   |    7 | CarryLogic          |
| LUT6     |    6 | LUT                 |
| IBUF     |    2 | IO                  |
| OBUF     |    1 | IO                  |
| LUT4     |    1 | LUT                 |
| LUT1     |    1 | LUT                 |
| FDCE     |    1 | Flop & Latch        |
| BUFG     |    1 | Clock               |
+----------+------+---------------------+
```

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	8	0	17600	0.05
LUT as Logic	8	0	17600	0.05
LUT as Memory	0	0	6000	0.00
Slice Registers	29	0	35200	0.08
Register as Flip Flop	29	0	35200	0.08
Register as Latch	0	0	35200	0.00
F7 Muxes	0	0	8800	0.00
F8 Muxes	0	0	4400	0.00

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
1	Yes	-	Reset
0	Yes	Set	-
28	Yes	Reset	-

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
1	Yes	-	Reset
0	Yes	Set	-
28	Yes	Reset	-

7. Primitives

Ref Name	Used	Functional Category
FDRE	28	Flop & Latch
CARRY4	7	CarryLogic
LUT6	6	LUT
IBUF	2	IO
OBUF	1	IO
LUT4	1	LUT
LUT1	1	LUT
FDCE	1	Flop & Latch
BUFG	1	Clock

1.4 Rapport de timing Static Timing Analysis (STA)

il y n' y a pas une violation de set up
➔ Total Negative Slack =TNS = 0

Design Timing Summary

WNS(ns)	TNS(ns)	TNS Failing Endpoints	TNS Total Endpoints	WHS(ns)	THS(ns)
3.787	0.000	0	3665	0.018	0.000

All user specified timing constraints are met.

Select an object to see properties

Tcl Console | Messages | Log | Reports | Design Runs | Power | DRC | Methodology | Timing

Report	Type	Options	Modified
Power - Route Design	report_power		5/9/23, 3:01 PM
Route Status - Route Design	report_route_status		5/9/23, 3:01 PM
Timing Summary - Route Design	report_timing_summary	max_paths = 10;	5/9/23, 3:01 PM

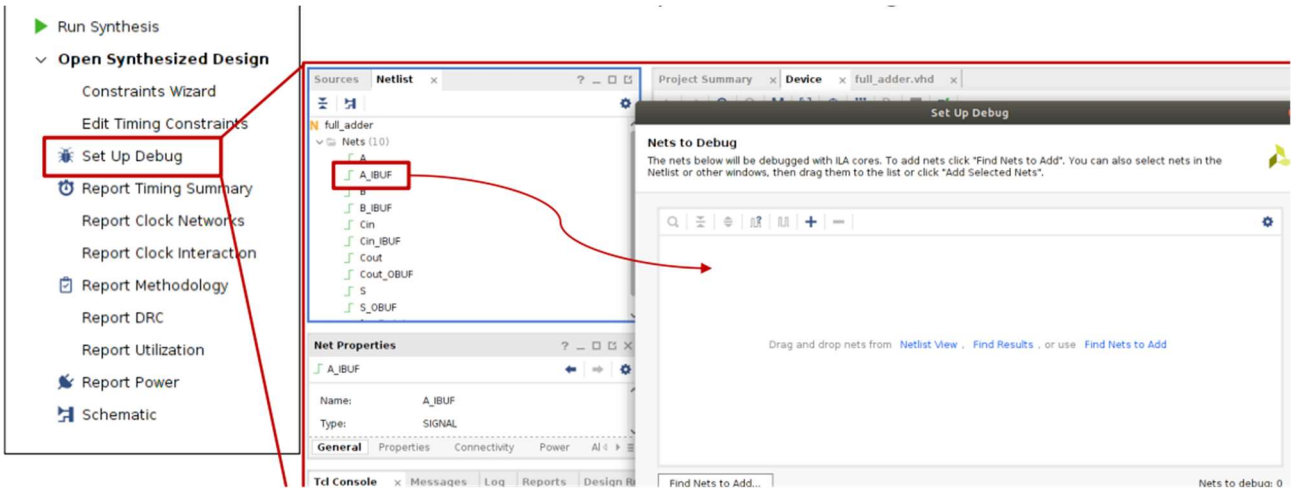
```
----------------------------------------------------------------------------------------------------
From Clock:  dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK
  To Clock:  dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK

Setup :            0  Failing Endpoints,  Worst Slack       26.165ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.063ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack       15.250ns,  Total Violation        0.000ns
----------------------------------------------------------------------------------------------------
```

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

Clock        Waveform(ns)       Period(ns)      Frequency(MHz)
-----        ------------       ----------      --------------
sys_clk_pin  {0.000 5.000}      10.000          100.000
```

Description de la forme de l'horloge Ici, changement d'état de l'horloge toutes les 0 et 5 ns avec une période de 10 ns

Fréquence de l'horloge

Question15. Ouvrez le Set Up Debug. Placez des sondes sur les signaux à observer que vous avez défini à la question 12.



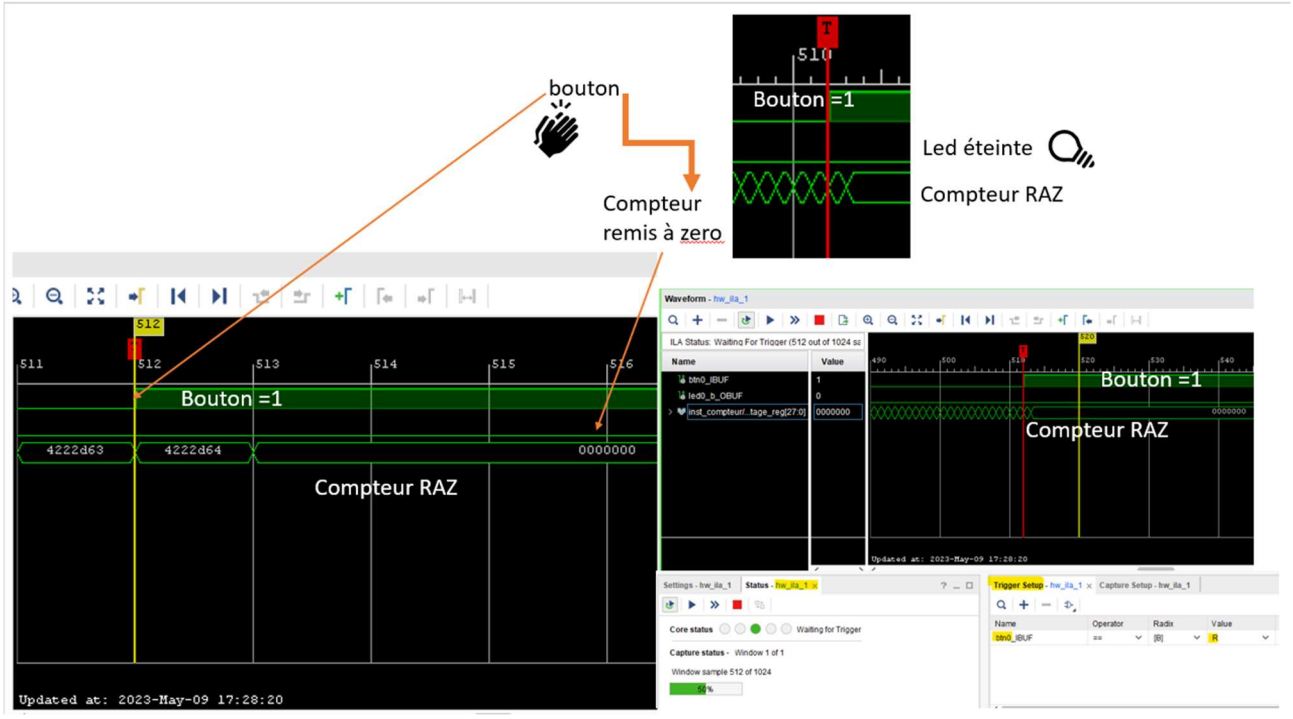
Question 16. Lancez l'implémentation puis étudiez le rapport de timing (vérifiez les violations de set up et de hold et identifiez le chemin critique).

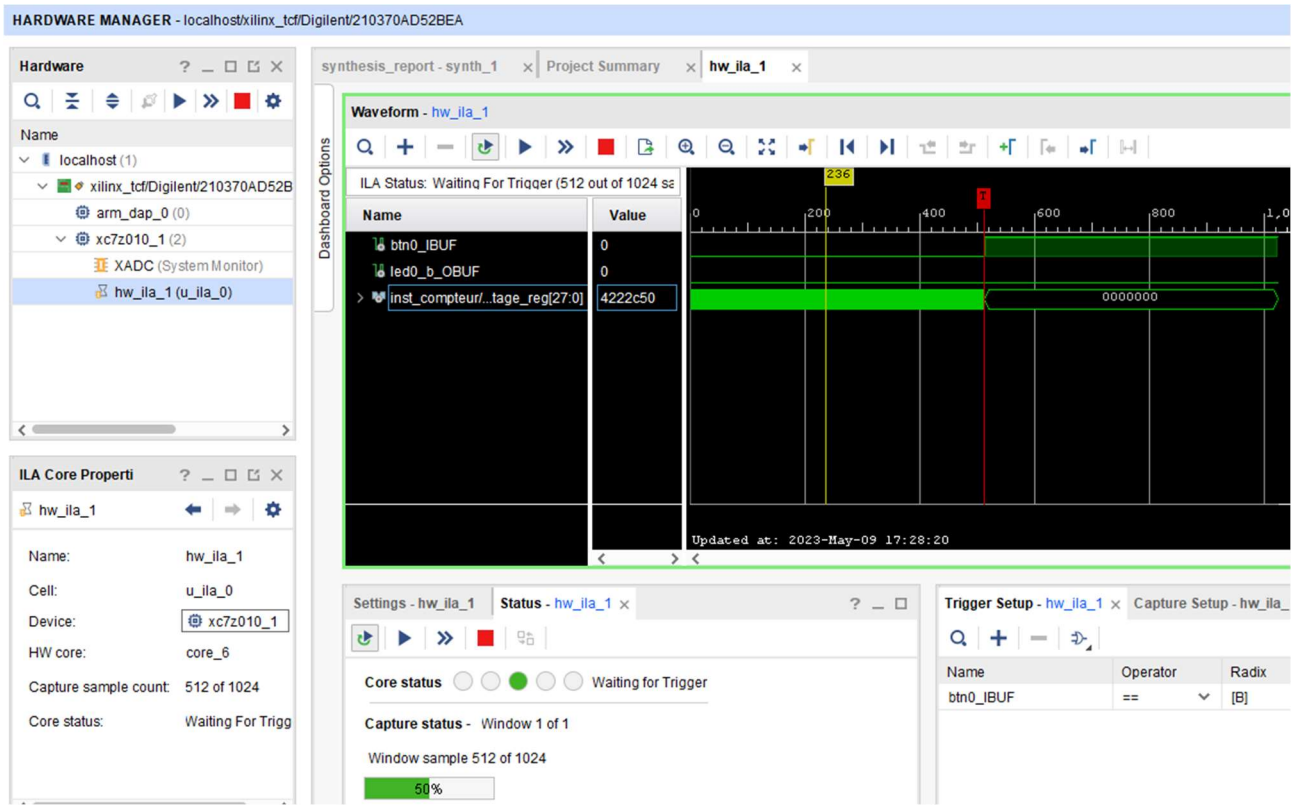
Description de notre chemin critique :

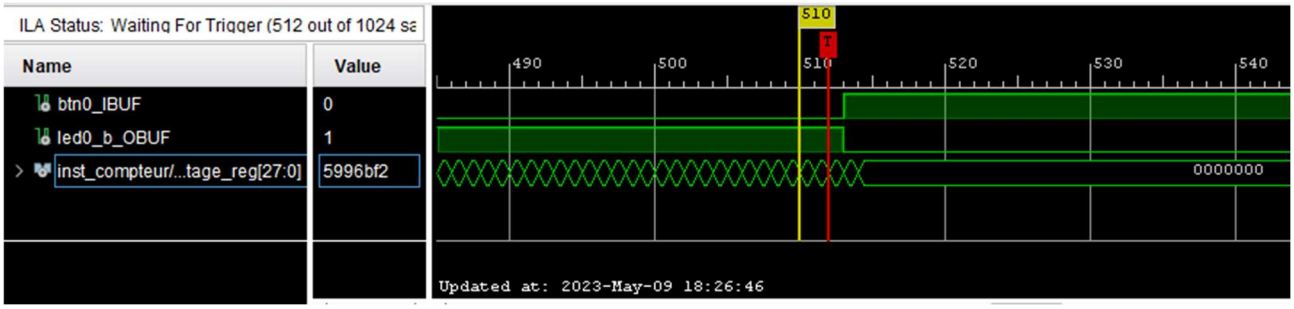
```
Setup :            0  Failing Endpoints,  Worst Slack       26.165ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.063ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack       15.250ns,  Total Violation        0.000ns
---------------------------------------------------------------------------------------------------


Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             26.165ns  (required time - arrival time)
  Source:                 dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/state_reg[1]/C
                            (rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/
  Destination:            dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_switch/portno_temp_re
                            (rising edge-triggered cell FDRE clocked by dbg_hub/inst/BSCANID.u_xsdbm_id/
  Path Group:             dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bs
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            33.000ns  (dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIE
  Data Path Delay:        6.770ns  (logic 1.721ns (25.421%)  route 5.049ns (74.579%))
  Logic Levels:           5  (CARRY4=1 LUT3=1 LUT4=1 LUT5=1 LUT6=1)
  Clock Path Skew:        -0.061ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    3.071ns = ( 36.071 - 33.000 )
    Source Clock Delay      (SCD):    3.508ns
    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
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

Vérification de comportement de ma carte en temps réel + Explication :

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

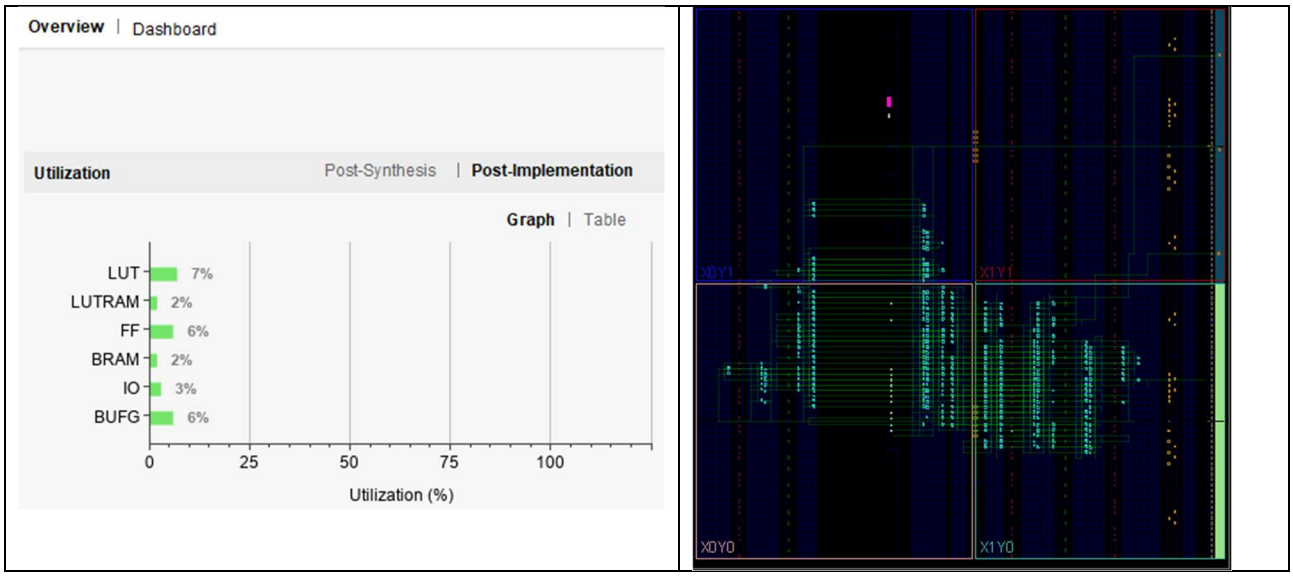



HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210370AD52BEA
Hardware
Name
localhost (1)
xilinx_tcf/Digilent/210370AD52B
arm_dap_0 (0)
xc7z010_1 (2)
XADC (System Monitor)
hw_ila_1 (u_ila_0)
ILA Core Properti
hw_ila_1
Name: hw_ila_1
Cell: u_ila_0
Device: xc7z010_1
HW core: core_6
Capture sample count: 512 of 1024
Core status: Waiting For Trigg
synthesis_report - synth_1
Project Summary
hw_ila_1
Dashboard Options
Waveform - hw_ila_1
ILA Status: Waiting For Trigger (512 out of 1024 sa
Name
Value
btn0_IBUF
0
led0_b_OBUF
0
inst_compteur/...tage_reg[27:0]
4222c50
236
0
200
400
600
800
0000000
Updated at: 2023-May-09 17:28:20
Settings - hw_ila_1
Status - hw_ila_1
Core status
Waiting for Trigger
Capture status - Window 1 of 1
Window sample 512 of 1024
50%
Trigger Setup - hw_ila_1
Capture Setup - hw_ila_
Name
Operator
Radix
btn0_IBUF
==
[B]


ILA Status: Waiting For Trigger (512 out of 1024 sa
Name
Value
btn0_IBUF
0
led0_b_OBUF
1
inst_compteur/...tage_reg[27:0]
5996bf2
510
490
500
510
520
530
540
0000000
Updated at: 2023-May-09 18:26:46

Compte rendu Eve CHAR du 04/05/2023 – 09/05/202

Annexe



Code :

```
--##-- declaration premier composant : compteur ##
-- Declaration de ma bibliothèque
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity compteur is
    Port ( clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           enable : in STD_LOGIC;
           scount : out STD_LOGIC_VECTOR (27 downto 0));
end compteur;
----Description comportementale
architecture arch_compteur of compteur is
 signal comptage : integer :=0; --STD_LOGIC_VECTOR (27 downto 0);
-- indicateur de comptage interne

begin

process (clk,Reset,enable)
    begin                                -- Comptage sur le signal interne

if rising_edge (clk) then if Reset ='1' then

comptage <= 0; --scount <= (others => '0');

elsif enable ='1' then
          ----  if compteur = "11111111111111111111111111" then
compteur <= (others => '0'); -- pas de condition de fin de comptage
et remise à zéro.
```

```
comptage <= comptage + 1; --std_logic_vector(unsigned(scount) +
1); -- "+"(unsigned,int)
--end if;end if;end if;

end process;
 --scount <= std_logic_vector(compteur); -- scount copie la sortie =
valeur finale du compteur
 scount <= std_logic_vector(to_unsigned(comptage,28));

 end arch_compteur;

--##--declaration deuxieme composant : comparateur ##
-- Declaration de ma bibliothèque
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity comparateur is
--generic ( N : integer := 200000000);
    Port ( comparateur_in : in STD_LOGIC_VECTOR (27 downto 0);
           compteur_max : in STD_LOGIC_VECTOR (27 downto 0);
           end_counter : out STD_LOGIC);
end comparateur;

--Description comportementale
-------------------------------------------------
 architecture arch_comparateur of comparateur is

begin -- on va Comparer la valeur de comparateur_in avec la valeur
de comparateur_max


--principe de ma comparison : egal <= '1' when A=B else '0';
        end_counter <= '1' when comparateur_in = compteur_max
else '0';


--une autre façon d'ecrire ma comparison :

--if comparateur_in >= compteur_max then


--end_counter = '1';


--else


--end_counter='0';

--end if;
```

```
end arch_comparateur;

--##--declaration troisième composant : Bascule_T_flip_flops

-- Declaration de ma bibliothèque
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Bascule_T_Flip_Flops is

    Port ( clk : in STD_LOGIC;
        Reset : in STD_LOGIC;
        --enable : in STD_LOGIC;


T : in STD_LOGIC;
        t_out : inout STD_LOGIC :='0');
        --led0_b : inout STD_LOGIC :='0');
end Bascule_T_flip_flops;

--Description comportementale
-------------------------------------------------
architecture arch_Bascule_T_Flip_Flops of Bascule_T_Flip_Flops is

begin -- on va mémoriser la dernière entrée pendant N valeur de
comptage
    -- on va laisser la led allumée 2s et etteinte pendant 2S

  process(Reset,clk)
      begin
      if reset='1' then t_out <= '0';
      --if Reset='1' then led0_b <= '0';
      elsif rising_edge(clk) then
         if T ='1' then
            if t_out ='1' then
            --if led0_b ='1' then
               t_out <= '0';
               --led0_b <= '0';
            else --si t_out = '0'
               t_out <='1';
               --led0_b <='1';
            end if;

  end if;

end if;
  end process;

 end arch_Bascule_T_flip_flops;
--##--declaration liaison entre les differents composants et la carte

-- Declaration de ma bibliothèque
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE work.ALL;

entity counter is
   Port ( clk_gen : in STD_LOGIC;
          btn0: in STD_LOGIC;
          --restart : in std_logic;
          led0_b : out STD_LOGIC);
end counter;

architecture arch_counter of counter is
   --déclaration des signaux Outputs
   signal allumer_led : std_logic := '0';
   signal comptage_tb : std_logic_vector (27 downto 0);
   signal led0_b_intern : STD_LOGIC := '0';
   signal restart : std_logic := '0';
   signal reset_compteur : std_logic := '0';
   -- Clock period definitions
   constant N : integer := 200000000;
begin

   -- Gestion des inputs
   -- clk_gen => pas de traitement
   restart <= btn0 ;
   reset_compteur <=(allumer_led or restart);

   inst_compteur : entity compteur port map(
        clk => clk_gen,
        Reset => reset_compteur,
        enable => '1',

scount => comptage_tb );

   inst_comparateur : entity comparateur port map(
        comparateur_in => comptage_tb,

compteur_max => std_logic_vector(to_unsigned(N,28)),

end_counter => allumer_led );

inst_Bascule_T_Flip_Flops : entity Bascule_T_Flip_Flops port map(
         clk => clk_gen,
         Reset => restart,
         T => allumer_led,
         --t_out => led0_b_inter);
         t_out => led0_b_intern);

led0_b <= led0_b_intern;

end arch_counter;
```