

1. Dans un fichier *.vhd*, créez un module *Counter_unit* à partir du compteur du TP1. Le module prendra en entrée un signal d'horloge et de resetn, et donnera en sortie le signal *end_counter*. Utilisez un paramètre *generic()* pour définir le nombre de coup d'horloge à compter.

Le module Counter_Unit est décrit dans le fichier Counter_Unit.vhd.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.NUMERIC_STD.ALL;

entity counter_unit is
    generic ( constant Cst : positive := 200000000 );
    port (
        clk          : in std_logic;
        resetn       : in std_logic;
        End_Counter  : out std_logic;
    );
end counter_unit;

architecture behavioral of counter_unit is

    --Declaration des signaux internes
    signal D_out : positive range 0 to Cst ;
    Signal Cmd   : std_logic := '0';

begin
    --Partie sequentielle
    process(clk,resetn)
    begin
        if(resetn = '1') then
            D_out <= 0;
        elsif(rising_edge(clk)) then
            D_out <= D_out + 1 ;
            if (Cmd = '1') then
                D_out <= 0;
            end if;
        end if;
    end if;
    end process;

    --Partie combinatoire
    Cmd <= '1' when (D_out = Cst-1 )
        else '0';
    End_Counter <= Cmd;

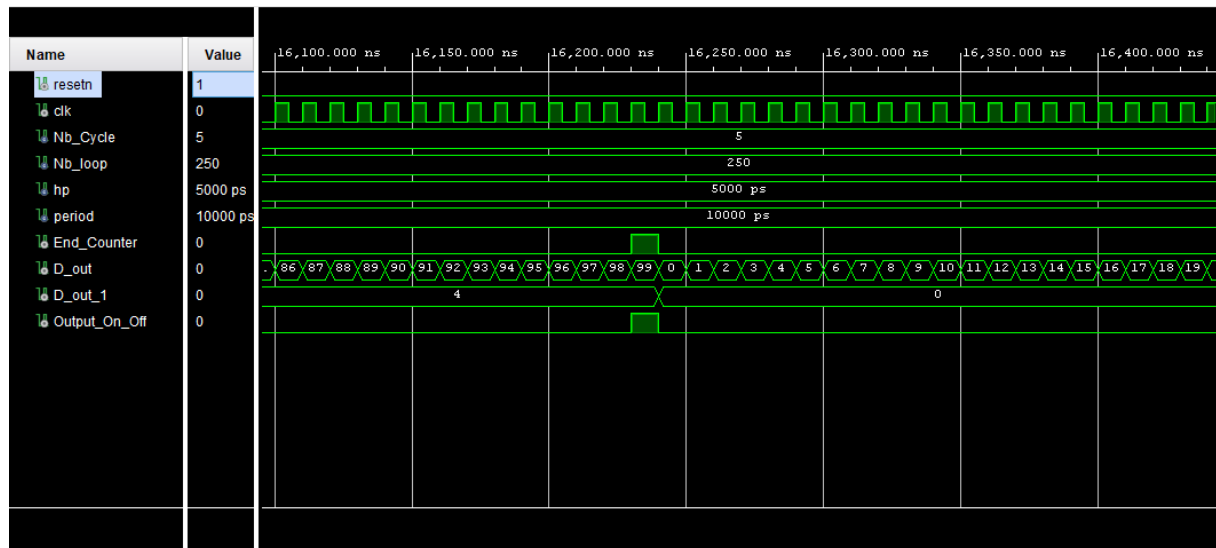
end behavioral;
```

-
- The diagram illustrates a digital circuit for a counter unit. It features a main block labeled "Counter_unit" with inputs "clk" and "resetn", and outputs "end Counter" and "out ON OFF". The "out ON OFF" output is connected to a circular component labeled "+1". The output of the "+1" component is connected to a 2-to-1 multiplexer. The other input of this multiplexer is labeled "on maintiens notre valeur". The output of the multiplexer is connected to another 2-to-1 multiplexer. The other input of this second multiplexer is labeled "RAZ". The output of the second multiplexer is connected to the "clk" input of a counter block. The counter block has a "resetn" input and an output labeled "out ON OFF". The "out ON OFF" output of the counter is connected back to the "out ON OFF" output of the main "Counter_unit" block, completing a feedback loop.

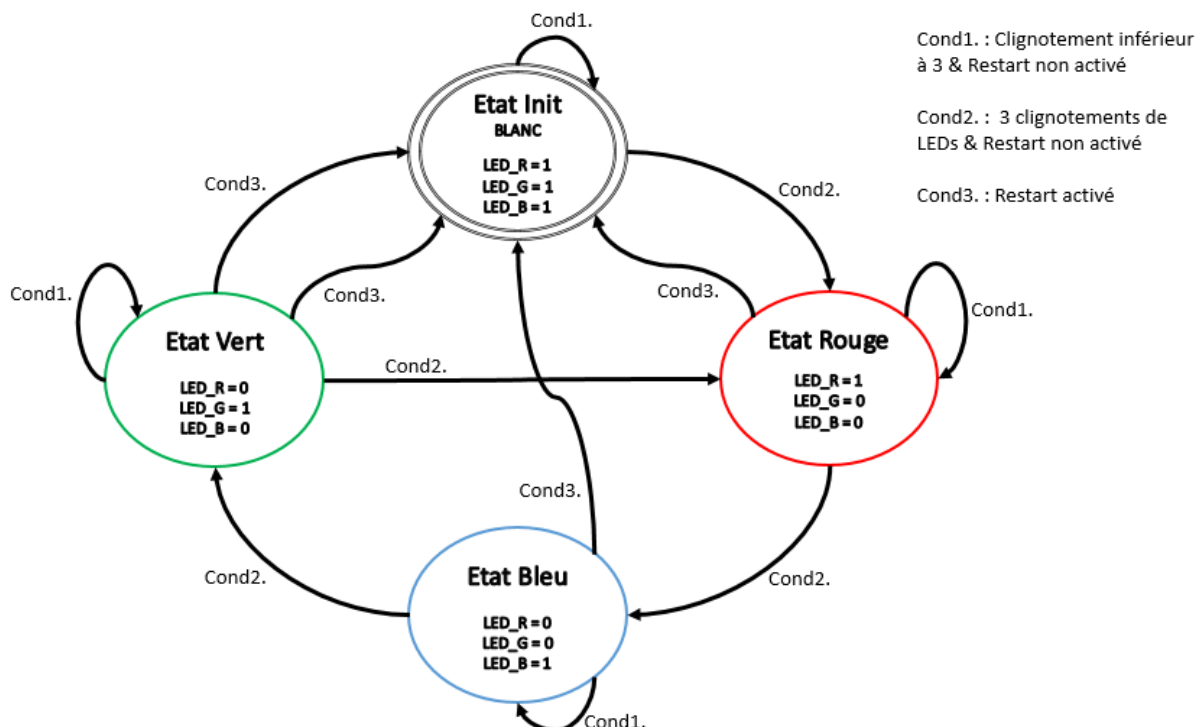
- Il en est / sera de même pour son test bench associé « tb_tp_fsm - Question 3.vhd »

- | Name | Value | |
|---------------|----------|--|
| resetn | 1 | |
| clk | 0 | |
| Nb_Cycle | 5 | |
| Nb_loop | 250 | |
| hp | 5000 ps | |
| period | 10000 ps | |
| End_Counter | 0 | |
| D_out | 0 | |
| D_out_1 | 0 | |
| Output_On_Off | 0 | |

Ici, on valide bien le fonctionnement du reset. Lorsque le reset est activé, le compteur s'arrête de compter. Et lorsque le compteur redémarre, on attends bien 5 cycle de comptage du signal End_Counter avant d'activer le signal de sortie Output_On_Off (ce dernier se fait lors de la dernière période du dernier cycle demandé). Voir image ci-dessous.



5. Créez en RTL une machine à états (FSM) permettant de faire clignoter une LED RGB en rouge puis bleu et enfin en vert avant de recommencer le cycle (rouge, bleu, vert, ...). Dans chaque état la LED devra clignoter 3 fois. De plus, si le bouton restart est appuyé, on retourne dans l'état initial quel que soit l'état dans lequel on se situe. L'état initial est l'état dans lequel on se situe au démarrage, on passe à l'état rouge après 3 clignotements de la LED en blanc (rouge, vert et bleu actifs en même temps).



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

Les signaux d'entrée sont :

- Clk : in std_logic ; → Horloge de notre système
- Resetn : in std_logic ; → Reset de notre système
- Restart : in std_logic ; → RAZ de la machine d'état

Les signaux de sortie sont :

- Output_On_Off : out std_logic ; → Sortie du compteur de notre Syst.
- LED_OUT : out std_logic_vector (2 downto 0) → LEDs RGB pour la carte CORA

Les signaux internes sont :

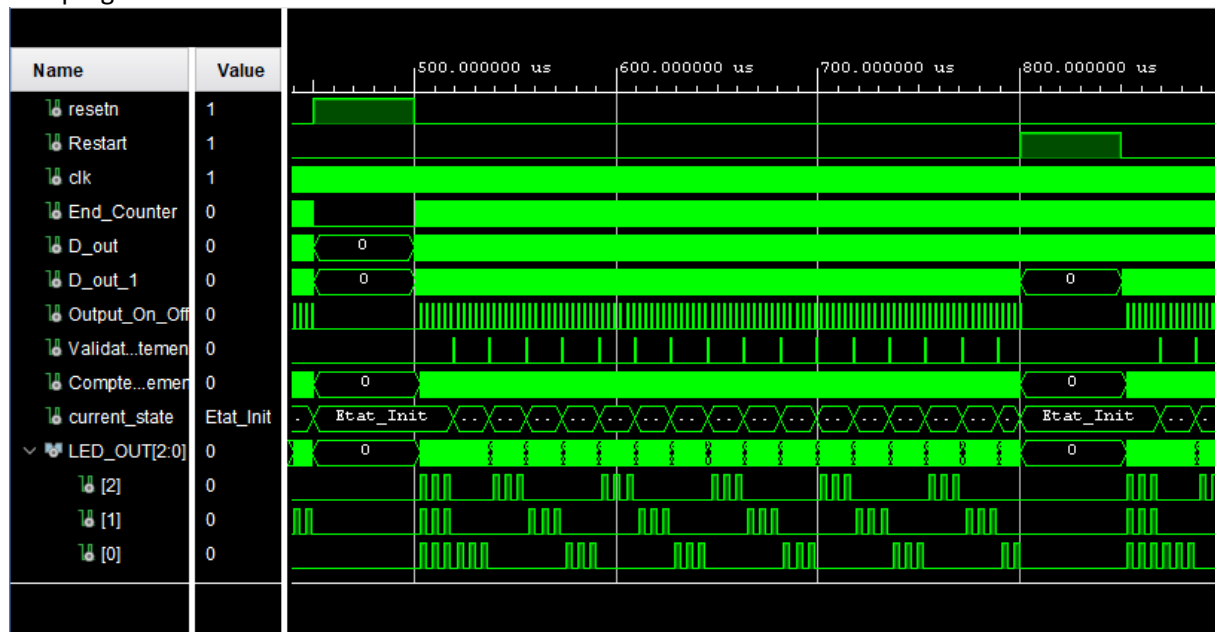
- S_LED_OUT : std_logic_vector (2 downto 0) ; → Signal interne pour LED_OUT
- Compteur_clignotement : positive :=0 ; → Comptage du Nb de clignotement
- Validation_clignotement : std_logic ; → Permet la validation du chgt d'état

7. Ajoutez à votre code VHDL les éléments que vous venez de créer.

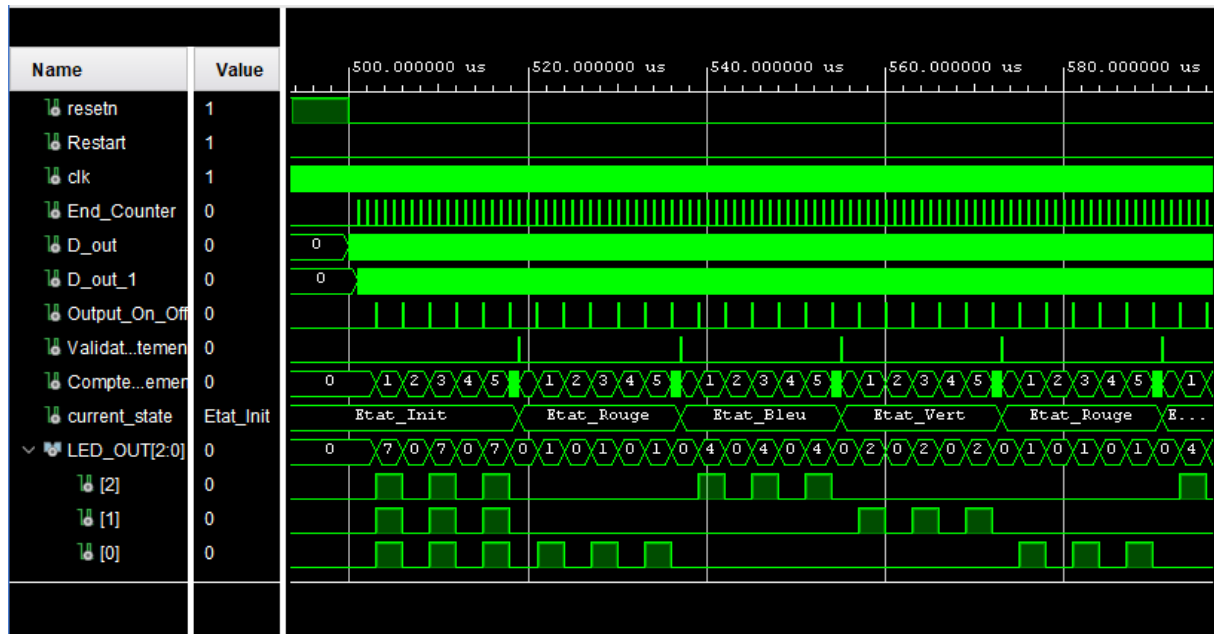
Cf. Code complet du projet.

8. Ecrivez un testbench pour tester votre architecture. Vérifiez à la simulation que vous obtenez le résultat attendu.

Validation des signaux resetn et Restart : lorsque l'un des deux signaux est activé, tous les signaux de comptage retournent à 0 et la machine d'état revient à l'état initial avec les LEDs en OFF.



Validation de la machine d'état : Ici, on voit bien qu'on suit le chemin de notre machine à savoir Etat_Init → Etat_Rouge → Etat_Bleu → Etat_Vert puis on revient à Etat_Rouge car aucun restart n'est activé.



9. Exécutez la synthèse et relevez les ressources utilisées (y compris la FSM). Sur la schématique, identifiez où se situe votre compteur de cycle.

Dans le fichier : Synthesis_report, on retrouve :

- La description de notre machine d'état

| State | New Encoding | Previous Encoding |
|------------|--------------|-------------------|
| etat_init | 00 | 00 |
| etat_rouge | 01 | 01 |
| etat_bleu | 10 | 10 |
| etat_vert | 11 | 11 |

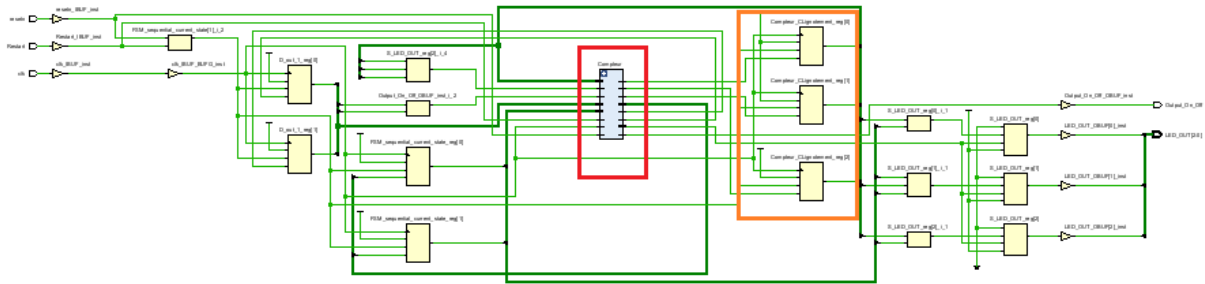
- Les composants qui permettent de constituer notre schéma RTL

```

-----
Start RTL Component Statistics
-----
Detailed RTL Component Info :
+---Adders :
    2 Input  3 Bit    Adders := 1
    2 Input  2 Bit    Adders := 1
+---Registers :
    3 Bit    Registers := 1
    2 Bit    Registers := 1
+---Muxes :
    2 Input  3 Bit    Muxes := 1
    4 Input  3 Bit    Muxes := 1
    2 Input  2 Bit    Muxes := 8
    4 Input  2 Bit    Muxes := 1
    2 Input  1 Bit    Muxes := 2
-----
Finished RTL Component Statistics
-----

```

Le schematic est le suivant :



En rouge on repère notre Counter_unit module et en orange le compteur qui nous permet de gérer le comptage du clignotement.

10. Modifiez le fichier de contraintes pour connecter vos entrées / sorties du système avec les broches de la carte. Réglez l'horloge pour que sa fréquence soit à 100MHz.

Dans le fichier de contrainte :

- btn[0] sera affecté au signal resetn
- btn[1] sera affecté au signal Restart
- led0_b sera affecté au signal LED_OUT[2]
- led0_g sera affecté au signal LED_OUT[1]
- led0_r sera affecté au signal LED_OUT[0]

Il faut également modifier le PL System Clock de la façon suivante :

```
# PL System Clock
set_property -dict {PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 8.000 -name sys_clk_pin -waveform {0.000 4.000} -add [get_ports clk]
#set_property -dict { PACKAGE_PIN H16 IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L13P_T2_MRCC_35 Sch=sysclk
#create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];#set
```

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

| Intra Clock Table | | | | | | |
|-------------------|---------|---------|-----------------------|---------------------|---------|---------|
| Clock | WNS(ns) | TNS(ns) | TNS Failing Endpoints | TNS Total Endpoints | WHS(ns) | THS(ns) |
| sys_clk_pin | 2.625 | 0.000 | 0 | 34 | 0.244 | 0.000 |

| Setup | Hold | Pulse Width |
|--------------------------------------|----------------------------------|---|
| Worst Negative Slack (WNS): 2.625 ns | Worst Hold Slack (WHS): 0.244 ns | Worst Pulse Width Slack (WPWS): 3.500 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: 34 | Total Number of Endpoints: 34 | Total Number of Endpoints: 35 |

All user specified timing constraints are met.

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

Le chemin critique est le suivant :

```
Max Delay Paths
-----
Slack (MET) :          2.625ns  (required time - arrival time)
Source:              Compteur/D_out_reg[3]/C
                    (rising edge-triggered cell FDCE clocked by sys_clk_pin  {rise@0.000ns fall@4.000ns p
Destination:         Compteur/D_out_reg[25]/D
                    (rising edge-triggered cell FDCE clocked by sys_clk_pin  {rise@0.000ns fall@4.000ns p
Path Group:          sys_clk_pin
Path Type:           Setup (Max at Slow Process Corner)
Requirement:         8.000ns  (sys_clk_pin rise@8.000ns - sys_clk_pin rise@0.000ns)
Data Path Delay:     5.422ns  (logic 2.567ns (47.347%)  route 2.855ns (52.653%))
```

12. Générez le bitstream pour vérifier le système sur carte.

Dans un premier temps, un bitstream à était généré avec :

- generic (constant Cst : positive := 100000000);

Le fichier bitstream est dispo au lien : TP\TP03\Bitstream\tp_fsm_3secondes.bit

Pour les besoins de la réalisation de la vidéo, on va régénérer un bitstream avec :

- generic (constant Cst : positive := 100000000/3);

Le fichier bitstream est dispo au lien : TP\TP03\Bitstream\tp_fsm_1seconde.bit