Tache Bonus Mini-Projet LU3EE100

Console SU-EE100 sur FPGA

Réalisé par :

- **KADI Koussaila**
- **CHABANE Sabrina**

Groupe B2

Enseignant de TP :

Mr. Orlando Chuquimia Camacho

Année universitaire 2019-2020

Module Moving Colors

1-diviseur d'horloge :

Générer une horloge qui va cadencer le changement de teinte en sortie du module.

- ✓ ***Description et simulation de diviseur d'horloge sous Modelsim :***

Afin d'obtenir une horloge de fréquence 10MHz on a utilisé un diviseur d'horloge qui permet de divise sur 10 la fréquence de l'horloge de la carte **Nexys 4** qui est cadencée à 100 MHz.

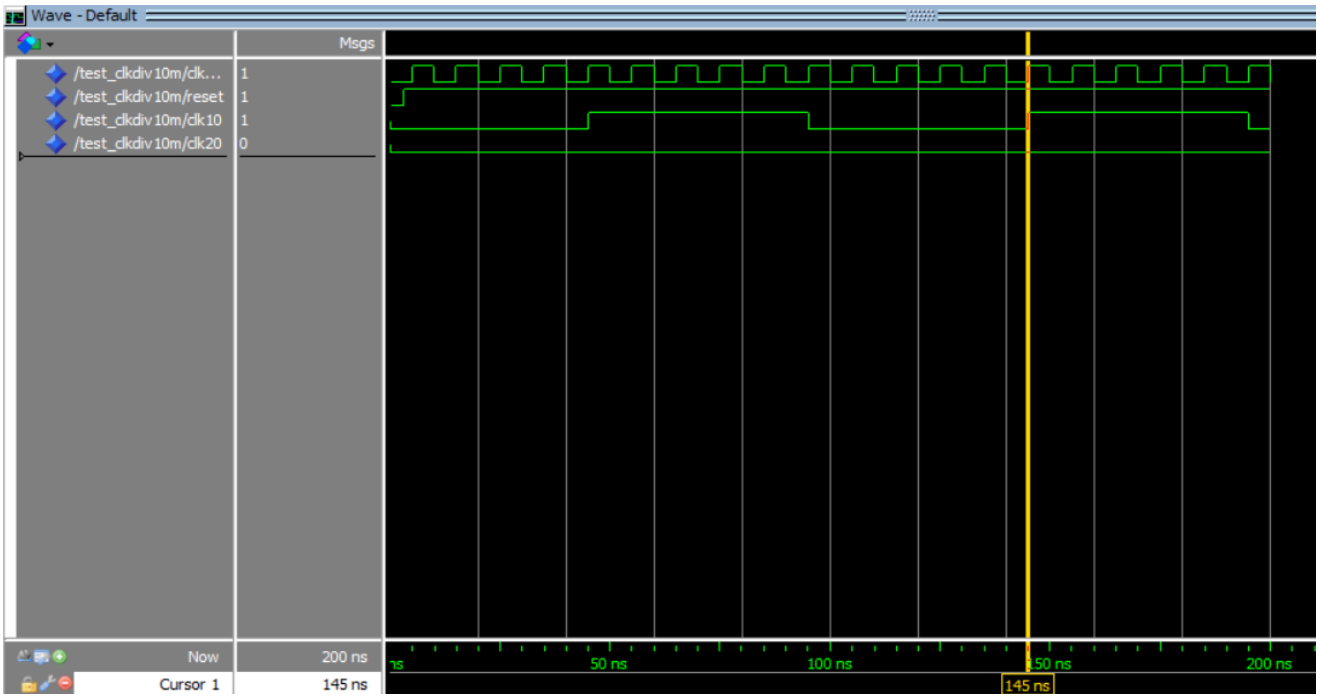
- ✓ **Le code de diviseur d'Horloge : 100 MHz --> 10 MHz « clkDiv10.vhd »**

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

entity clkDiv10M is
port(reset,clk100 : in std_logic;
     clk10MHz: out std_logic);
end  clkDiv10M;

architecture archi of clkDiv10M is
signal cpt: std_logic_vector(2 downto 0):=(others=> '0'); -- compteur pour 10 Mhz avec periode de 100ns
signal cpt20Hz: std_logic_vector(22 downto 0):=(others=> '0'); -- compteur pour 20Hz de 50ms donc pour visualiser il faut
signal clk10MHz_tmp : std_logic :='0';                        --mettre le pas de simulation
signal clk20Hz_tmp : std_logic :='0';

begin
 process(reset,clk100)
 begin
  if reset='0' then clk10MHz_tmp<='0';clk20Hz_tmp <='0';
  elsif rising_edge(clk100) then cpt<=cpt+1;cpt20Hz<=cpt20Hz+1;
  if cpt=4 then clk10MHz_tmp<= not clk10MHz_tmp; cpt<=(others=>'0') ; end if;
  if cpt20Hz=2499999 then clk20Hz_tmp<=not clk20Hz_tmp; cpt20Hz<=(others=>'0'); end if; --pour faire une horloge de 20HZ
                                                                                       --il suffit juste de déclarer clk20 comme sortie
                                                                                       --et la brancher au signal clk20Hz_tmp
  end if;
 end process;
clk10MHz<=clk10MHz_tmp;
end archi;
```

- ***Le code de testbench test_clkdiv10.vhd :*** **tester le bon fonctionnement de diviseur d'horloge 10MHz.**

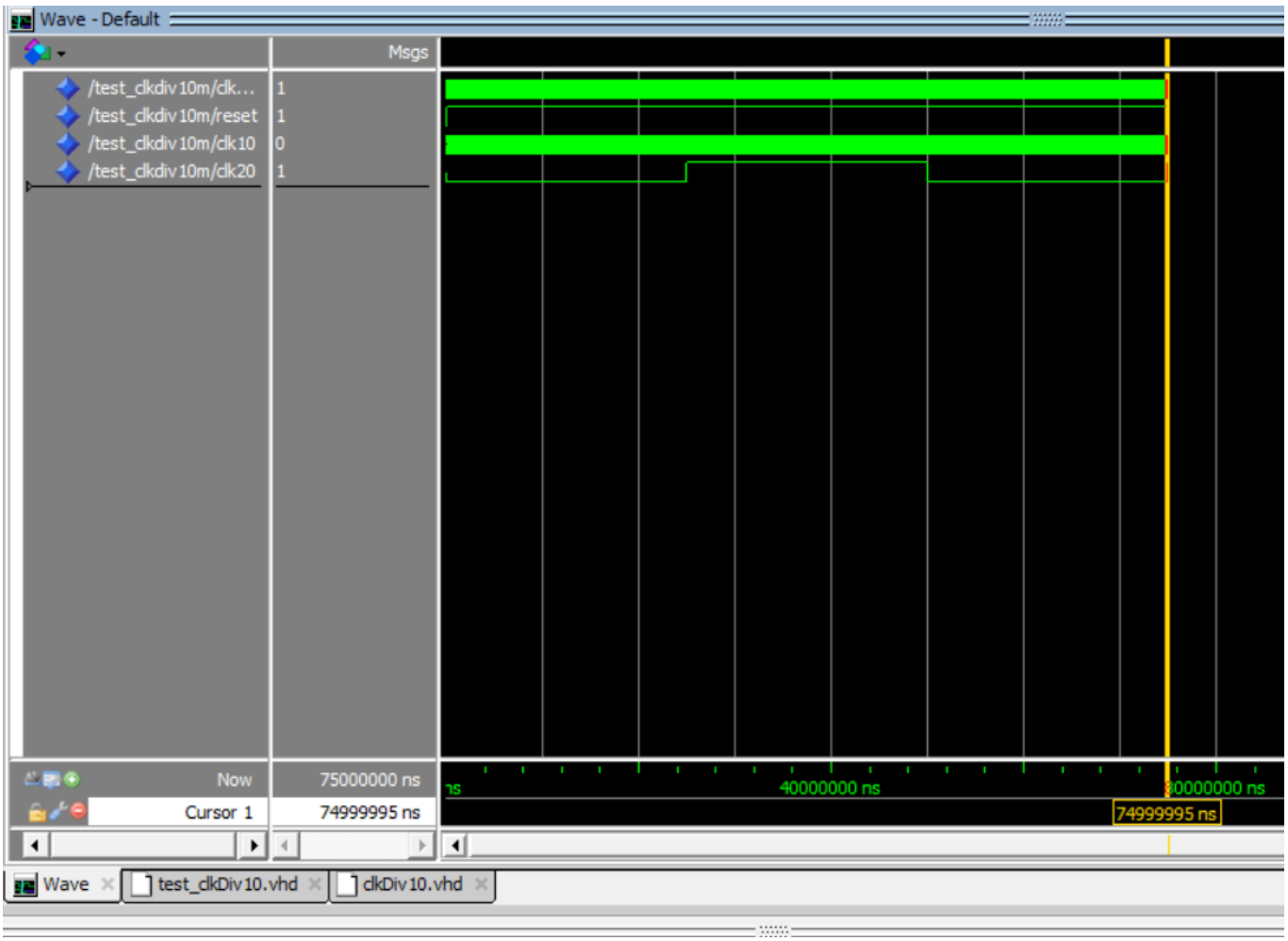
```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.ALL;

entity test_clkDiv10M is
end  test_clkDiv10M;

architecture archi of test_clkDiv10M is
signal clk100: std_logic :='0';
signal reset: std_logic  :='0';
signal clk10 : std_logic;
begin
label1 : entity work.clkDiv10M
         port map(reset,clk100,clk10);


clk100<= not clk100 after 5 ns;
reset <= '1' after 3 ns;
end archi;
```

- ***La simulation :*** Vérifier le bon fonctionnement du module de division d'horlage.



On faisant le zoome sur le chronogramme obtenue lors de la simulation on constate que la période de l'horloge de la carte Nexys 4 qui est de 10 ns est multiplier par dix afin d'obtenir une période de 100 ns qui correspond à une fréquence de 10MHz

On peut aussi obtenir une horloge qui génère une fréquence de 20 Hz :



2-les compteurs :

On a besoin de 3 compteurs de 5 bits, cadencés par l'horloge issue du diviseur d'horloge (10 MHz) Chaque compteur va représenter la teinte de rouge, de vert ou de bleu à envoyer en sortie.

- ***Déscription et simulation des compteurs :***

❖ **Le Code du compteur_moving_colors «compteur_moving_colors.vhd » :**

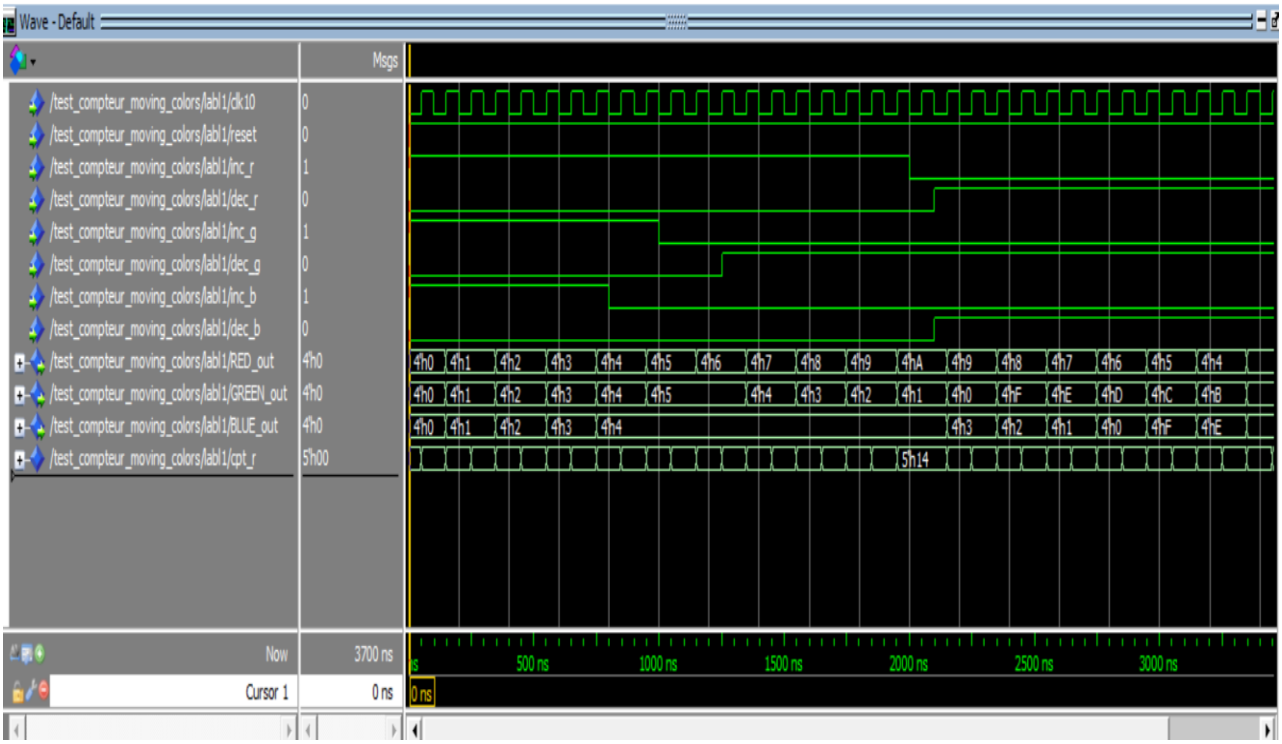
```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.ALL;

entity compteur_moving_colors is
port(clk10,reset : in std_logic;
     inc_r , dec_r :in std_logic;
     inc_g , dec_g : in std_logic;
     inc_b , dec_b : in std_logic;
     RED_out : out std_logic_vector(3 downto 0);
     GREEN_out : out std_logic_vector(3 downto 0);
     BLUE_out : out std_logic_vector(3 downto 0));
end compteur_moving_colors;

architecture archi of compteur_moving_colors is
signal cpt_r : std_logic_vector(4 downto 0);
signal cpt_g : std_logic_vector(4 downto 0);
signal cpt_b : std_logic_vector(4 downto 0);
signal mod_r : std_logic_vector(2 downto 0);
signal mod_g : std_logic_vector(2 downto 0);
signal mod_b : std_logic_vector(2 downto 0);
begin
--compteur de la couleur rouge :
mod_r<=reset&inc_r&dec_r;
process(clk10,mod_r)
begin
if reset='0' then cpt_r<=(others=>'0');
elsif rising_edge(clk10) then
   case(mod_r) is
     when "000" => cpt_r<=(others=>'1');  -- initialisation
     when "100" => cpt_r<=cpt_r;    -- mémorisation
     when "110" => cpt_r<=cpt_r+1; -- incrémentation
     when "101" => cpt_r<=cpt_r-1; -- décrémentation
     when others=> NULL;
   end case;
```

```
end process;

--compteur de la couleur verte
mod_g<=reset&inc_g&dec_g;
process(clk10,mod_g)
begin
if reset='0' then cpt_g<=(others=>'0');
elsif rising_edge(clk10) then
   case(mod_g) is
     when "000" => cpt_g<=(others=>'1');  -- initialisation
     when "100" => cpt_g<=cpt_g;    -- mémorisation
     when "110" => cpt_g<=cpt_g+1; -- incrémentation
     when "101" => cpt_g<=cpt_g-1; -- décrémentation
     when others=> NULL;
   end case;
end if;
end process;

--compteur de la couleur blue
mod_b<=reset&inc_b&dec_b;
process(clk10,mod_b)
begin
if reset='0' then cpt_b<=(others=>'0');
elsif rising_edge(clk10) then
   case(mod_b) is
     when "000" => cpt_b<=(others=>'1');  -- initialisation
     when "100" => cpt_b<=cpt_b;    -- mémorisation
     when "110" => cpt_b<=cpt_b+1; -- incrémentation
     when "101" => cpt_b<=cpt_b-1; -- décrémentation
     when others=> NULL;
   end case;
end if;
end process;
-----------------------------

RED_out<=cpt_r(4 downto 1);
GREEN_out<=cpt_g(4 downto 1);
BLUE_out<=cpt_b(4 downto 1);

end archi;
```

- **Testbench du compteur_moving_colors « test compteur_moving_colors.vhd »**

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

entity test_compteur_moving_colors is
end test_compteur_moving_colors;

architecture archi of  test_compteur_moving_colors is
signal clk10 : std_logic := '0';
signal reset :std_logic :='0';
signal inc_r,dec_r: std_logic;
signal inc_g , dec_g :std_logic;
signal inc_b , dec_b :std_logic;
signal RED_out : std_logic_vector(3 downto 0);
signal GREEN_out : std_logic_vector(3 downto 0);
signal BLUE_out : std_logic_vector(3 downto 0);
begin
labl1 : entity work.compteur_moving_colors
        port map(clk10,reset,
                 inc_r , dec_r,
                 inc_g , dec_g ,
                 inc_b , dec_b ,
                 RED_out ,
                 GREEN_out ,
                 BLUE_out);

clk10<=not clk10 after 50 ns;
reset<= '1' after 1 ns;
inc_r<='1' , '0' after 2000 ns;
dec_r <='0' , '1' after 2100 ns ;
inc_g<='1' , '0' after 1000 ns;
dec_g <='0' , '1' after 1250 ns ;
inc_b<='1' , '0' after 800 ns;
dec_b <='0' , '1' after 2100 ns;
end archi;
```

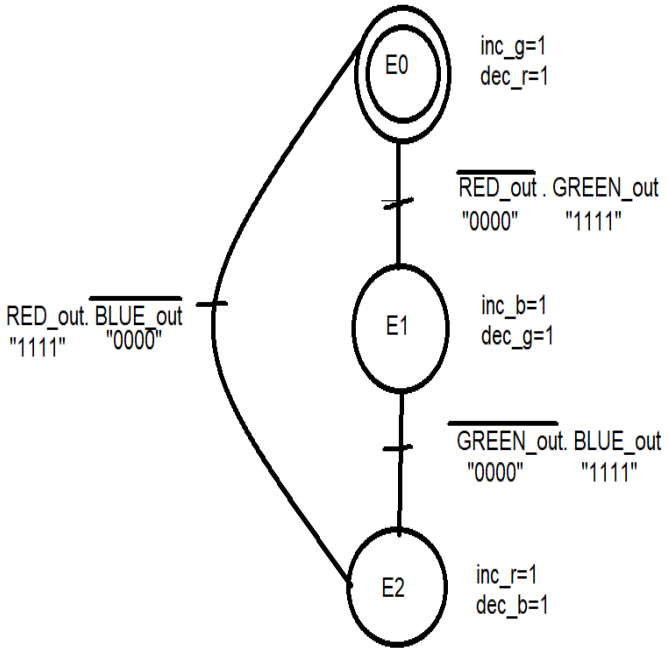
- ***La simulation :*** Vérifier le bon fonctionnement du compteur_moving_colors:



3)-Machine à états :

Permet de commander les compteurs de teinte, elle est cadencée par l'horloge à 100 MHz.

- **Le graphe d'états de la MAE :**



- ***Déscription et simulation du graphe d'états « MAE » sous Modelsim :***

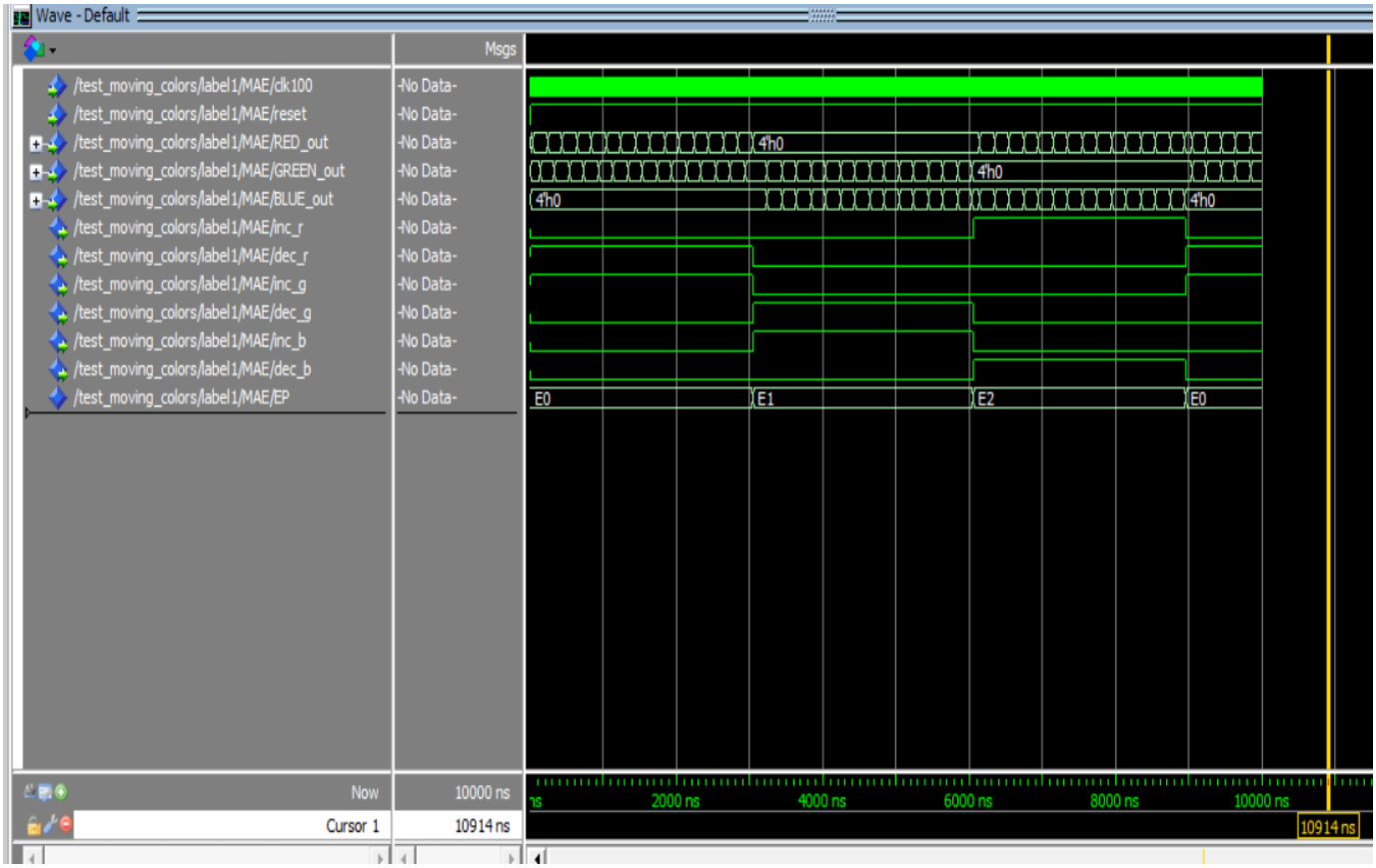
Le code de MAE « mae.vhd » :

```
library ieee;
use ieee.std_logic_1164.All;
use ieee.std_logic_unsigned.ALL;

entity MAE_moving_colors is
port(clk100,reset : in std_logic;
     RED_out : in std_logic_vector(3 downto 0);
     GREEN_out : in std_logic_vector(3 downto 0);
     BLUE_out : in std_logic_vector(3 downto 0);
     inc_r , dec_r : out std_logic;
     inc_g , dec_g : out std_logic;
     inc_b , dec_b : out std_logic);
end MAE_moving_colors ;

architecture archi of MAE_moving_colors is
type etat is (E0,E1,E2);
signal EP,EF :etat;
begin
--registre d'etat :
process (clk100,reset)
begin
if reset = '0' then EP<=EP;
elsif rising_edge(clk100) then EP<=EF;
end if;
end process;
--combinatoire des entrées et sorties:
process(EP,RED_out,GREEN_out,BLUE_out)
begin
case (EP) is
    when E0 => inc_r <='0';dec_r<='1';inc_g<='1';dec_g<='0';inc_b<='0';dec_b<='0';
               if RED_out = "0000" then
                   if GREEN_out="1111" then EF<=E1; end if;
               else EF<=E0;
               end if;
    when E1 => inc_r <='0';dec_r<='0';inc_g<='0';dec_g<='1';inc_b<='1';dec_b<='0';
               if GREEN_out="0000" then
                  if BLUE_out="1111" then EF<=E2; end if;
               else EF<=E1;
               end if;
    when E2 => inc_r <='1';dec_r<='0';inc_g<='0';dec_g<='0';inc_b<='0';dec_b<='1';
               if BLUE_out ="0000" then
                   if RED_out= "1111" then EF<=E0; end if;
               else EF<=E2;
               end if;
 end case;

end process;

end archi;
```

Testbentch de La MAE « test_mae.vhd » :

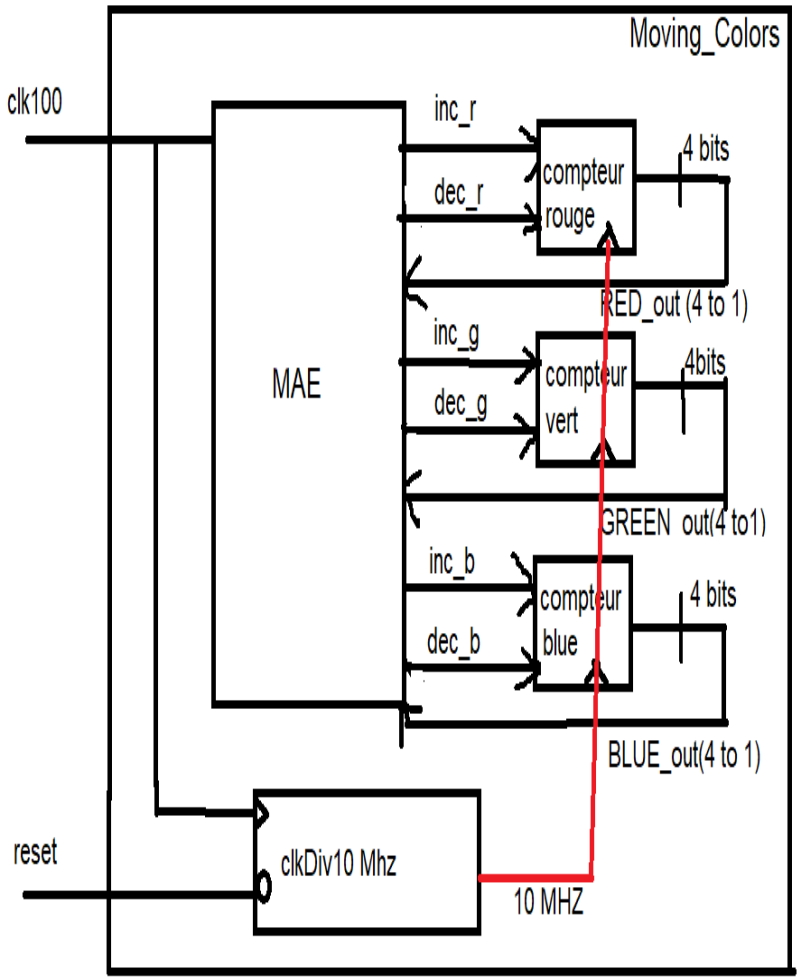
```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.ALL;

entity test_MAE_moving_colors is
end  test_MAE_moving_colors ;

architecture archi of test_MAE_moving_colors  is
signal clk100: std_logic :='0';
signal reset: std_logic  :='0';
signal GREEN_out,BLUE_out, RED_out :std_logic_vector(3 downto 0);
signal  inc_r , dec_r ,inc_g,dec_g ,inc_b , dec_b : std_logic;

begin
instance : entity work.MAE_moving_colors
          port map(clk100,reset,
                   RED_out,
                   GREEN_out ,
                   BLUE_out ,
                   inc_r , dec_r ,
                   inc_g , dec_g ,
                   inc_b , dec_b);
RED_out<="1111", "1110" after 100 ns, "1101" after 200 ns , "0000" after 500 ns, "1111" after 1000 ns;
GREEN_out<="0000","1111" after 100 ns, "1110" after 200 ns , "1111" after 500 ns , "0000" after 600 ns;
BLUE_out<= "1111","1110" after 100 ns, "1101" after 200 ns , "1111" after 600 ns , "0000" after 1000 ns;
clk100<= not clk100 after 5 ns;
reset <= '1' after 3 ns;
end archi;
```

La simulation : Vérifier le bon fonctionnement du **MAE**.



Déscreption du Module Moving_Colors sous *Modelsim* :
Il s'agit d'instancier le graphe d'états, les trois compteurs et le diviseur d'horloge dans un seule module.

- **Schéma bloc globale du module Moving_Colors :**



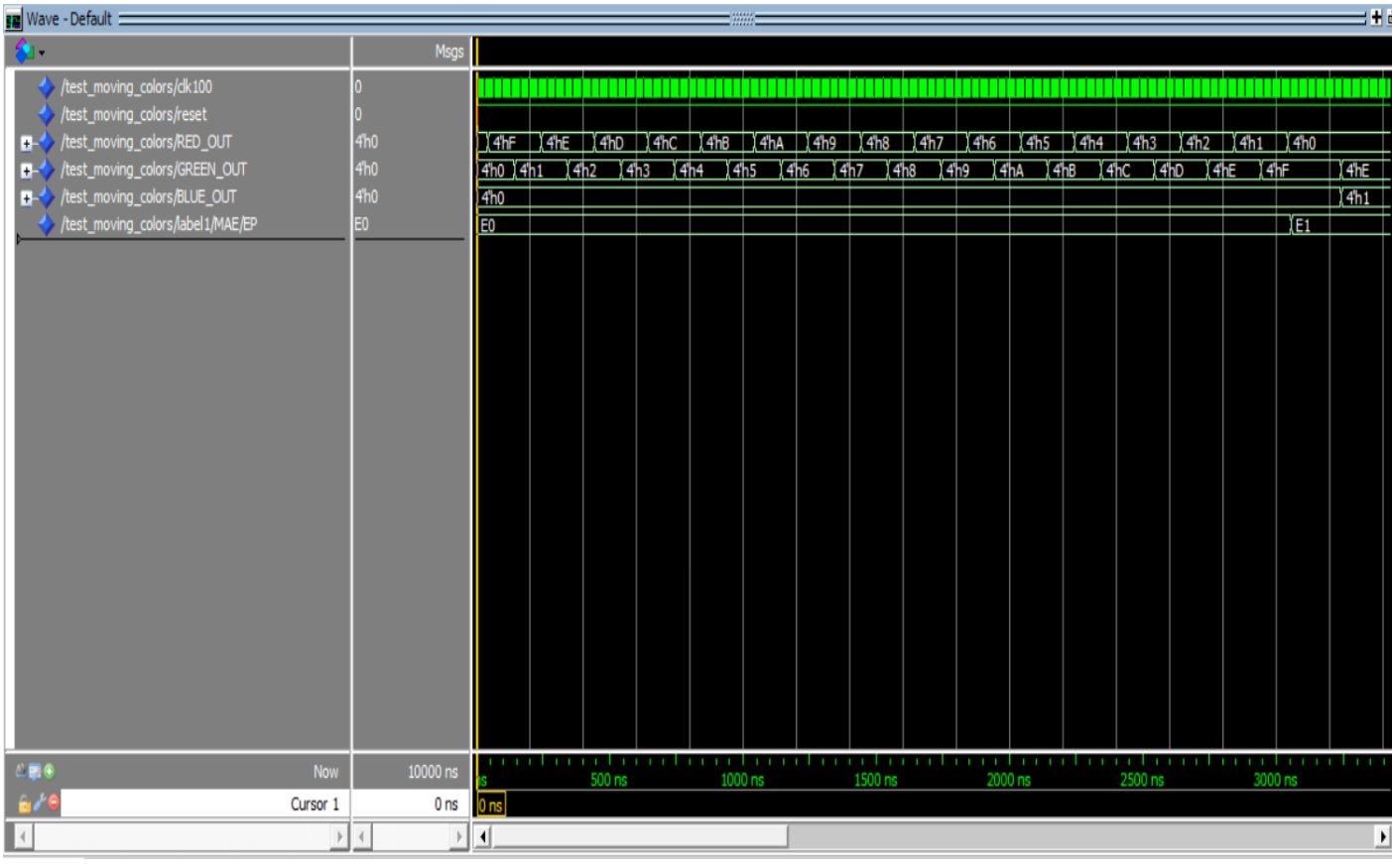
Le code «Moving_Colors.vhd »

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

entity Moving_colors is
              port (Clk100 ,reset  : in std_logic;
                    RED_OUT : out std_logic_vector(3 downto 0);
                    GREEN_OUT : out std_logic_vector(3 downto 0);
                    BLUE_OUT : out std_logic_vector(3 downto 0)
                    );
end Moving_colors;

architecture archi of Moving_colors is
signal clk10 : std_logic;
signal RED_out_tmp, GREEN_out_tmp, BLUE_out_tmp :std_logic_vector(3 downto 0);
signal inc_r , dec_r, inc_g , dec_g ,inc_b , dec_b : std_logic;
begin
clk10_MHZ : entity work.clkDiv10M
            port map (reset,clk100,clk10);

MAE :       entity work.MAE_moving_colors
            port map (clk100,reset,
                      RED_out_tmp,
                      GREEN_out_tmp,
                      BLUE_out_tmp,
                      inc_r , dec_r,
                      inc_g , dec_g,
                      inc_b , dec_b);

compteurs : entity work.compteur_moving_colors
            port map(clk10,reset,
                     inc_r , dec_r,
                     inc_g , dec_g,
                     inc_b , dec_b,
                     RED_out_tmp,
                     GREEN_out_tmp,
                     BLUE_out_tmp);

RED_OUT<=RED_out_tmp;
GREEN_OUT<=GREEN_out_tmp;
BLUE_OUT<=BLUE_out_tmp;

end archi;
```

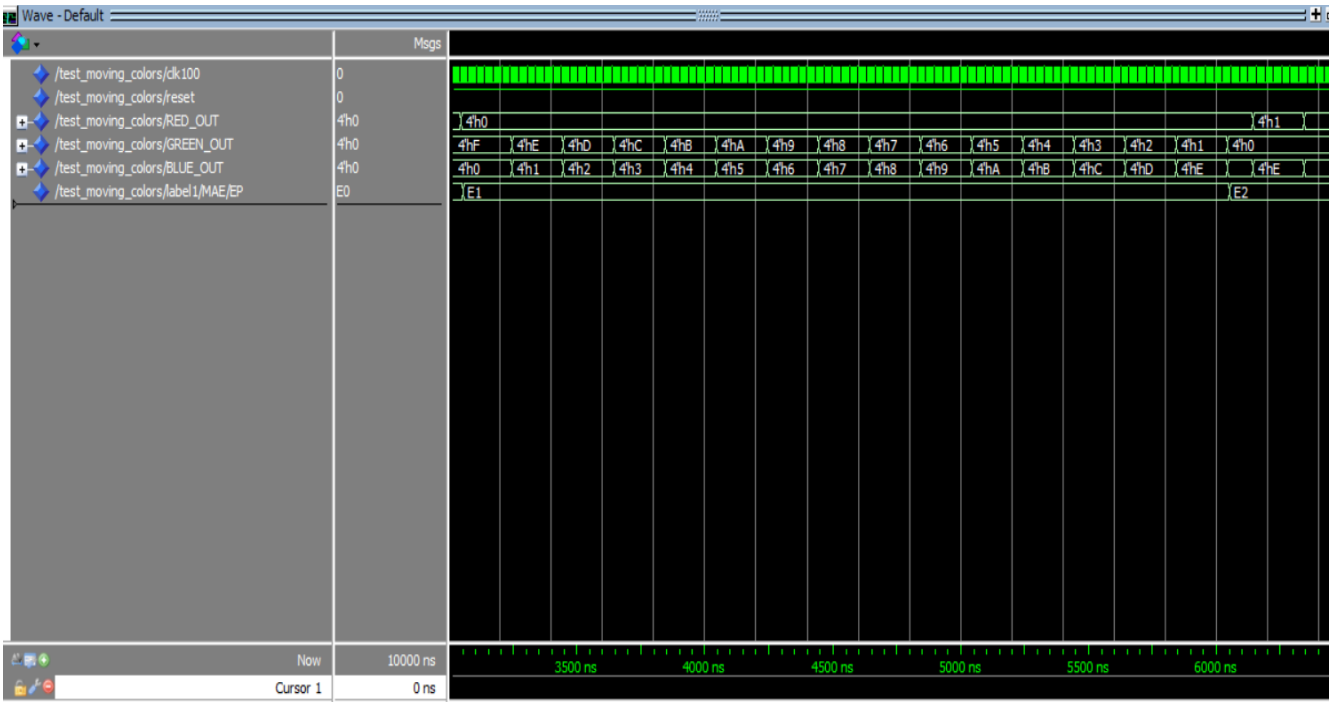
Testbench de module Moving_Colors « test_ Moving_Colors.vhdl»

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

entity test_Moving_colors is
end  test_Moving_colors;

architecture archi of test_Moving_colors is
signal clk100: std_logic :='0';
signal reset: std_logic  :='0';
signal RED_OUT,GREEN_OUT,BLUE_OUT : std_logic_vector(3 downto 0);
begin
label1 : entity work.Moving_colors
         port map (Clk100 ,Reset,
                   RED_OUT,
                   GREEN_OUT,
                   BLUE_OUT
                   );


clk100<= not clk100 after 5 ns;
reset <= '1' after 3 ns;
end archi;
```

La simulation : Vérifier le bon fonctionnement du module Moving_Colors.



Suite de la simulation de module Moving_Colors



On peut aussi tester le système complet et simuler directement Top.vhd :