SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Présentation du mini projet (Casse brique/Pong)

Membres du groupe (B1):

1. **Sara Brahami**
2. **Ghiles Oudjebour**

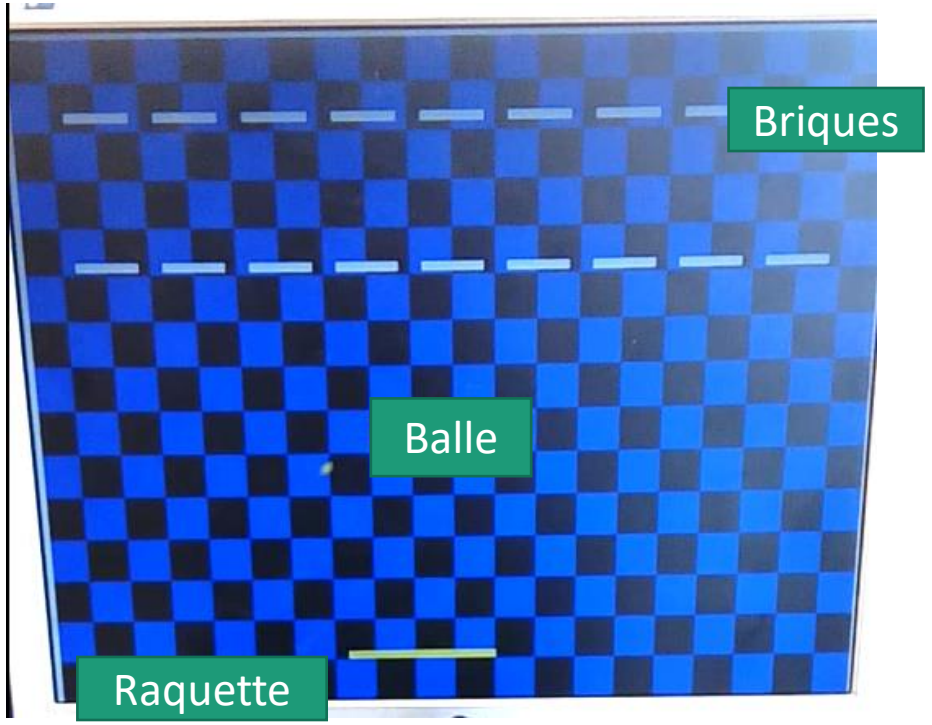
Mercredi 20 janvier 2021

Année universitaire :2020/2021.

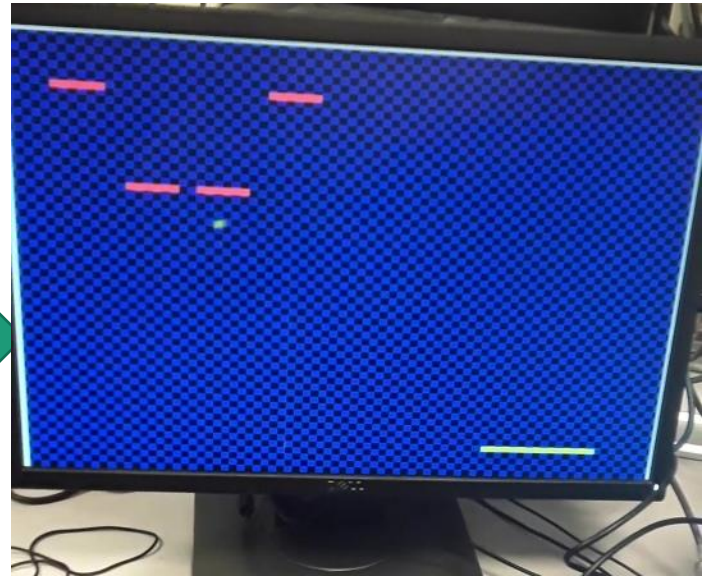
BREAKOUT

- Jeu d'arcade developpé par Atari, Inc. (1976)

Objectif du jeu: Détruire un mur de briques en le percutant à l'aide d'une balle lancée par une raquette.



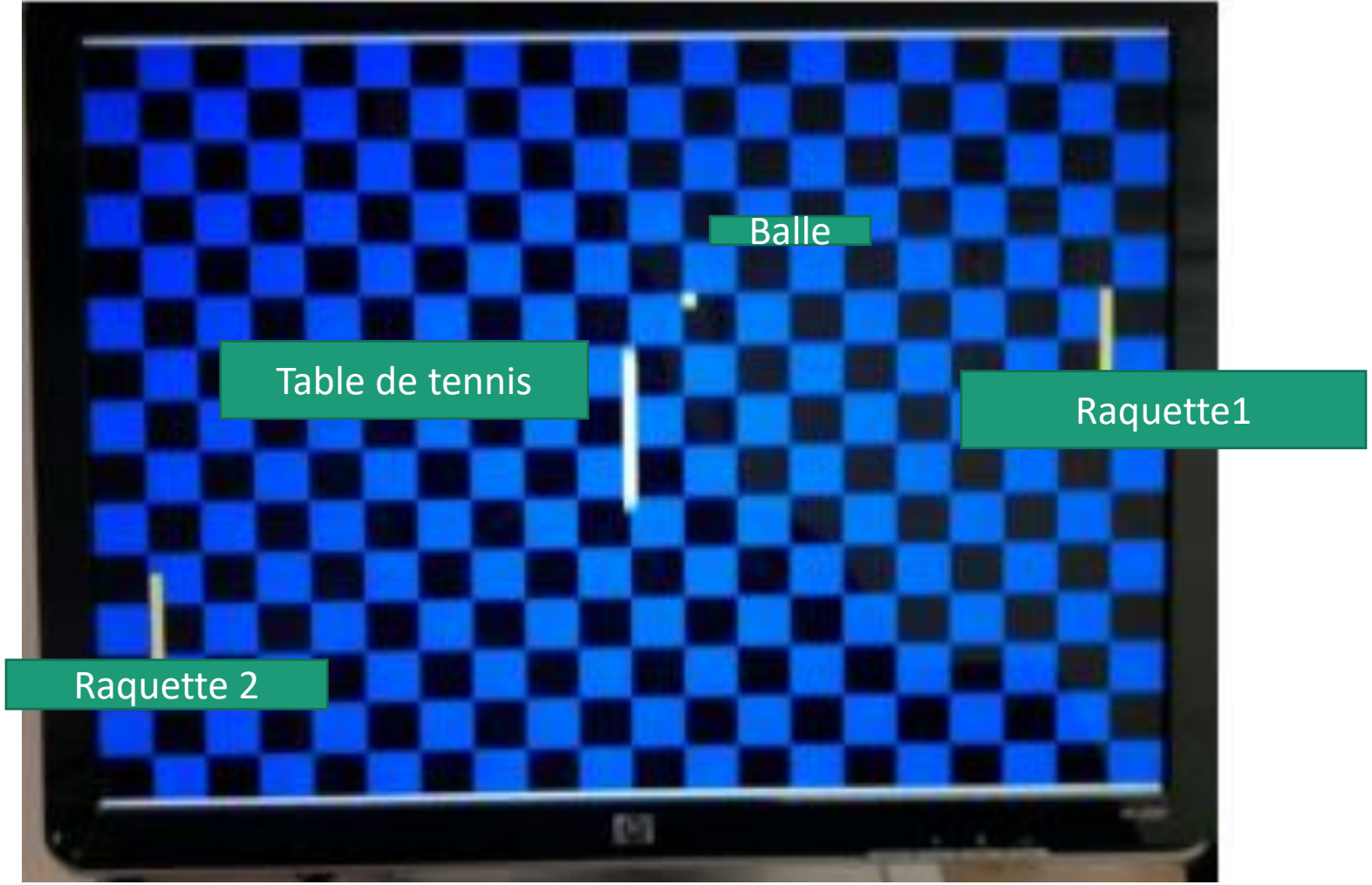
Niveau suivant



Pong

- Jeu d’arcade developpé par Allan Alcorn ,Inc. (1972)

Objectif du jeu: **Garder la balle dans le terrain , en jouant avec deux joueurs ,chacun commande une raquette en la faisant glisser verticalement entre les deux extrémités .si la balle frappe la raquette elle rebondit vers l’autre joueur .**

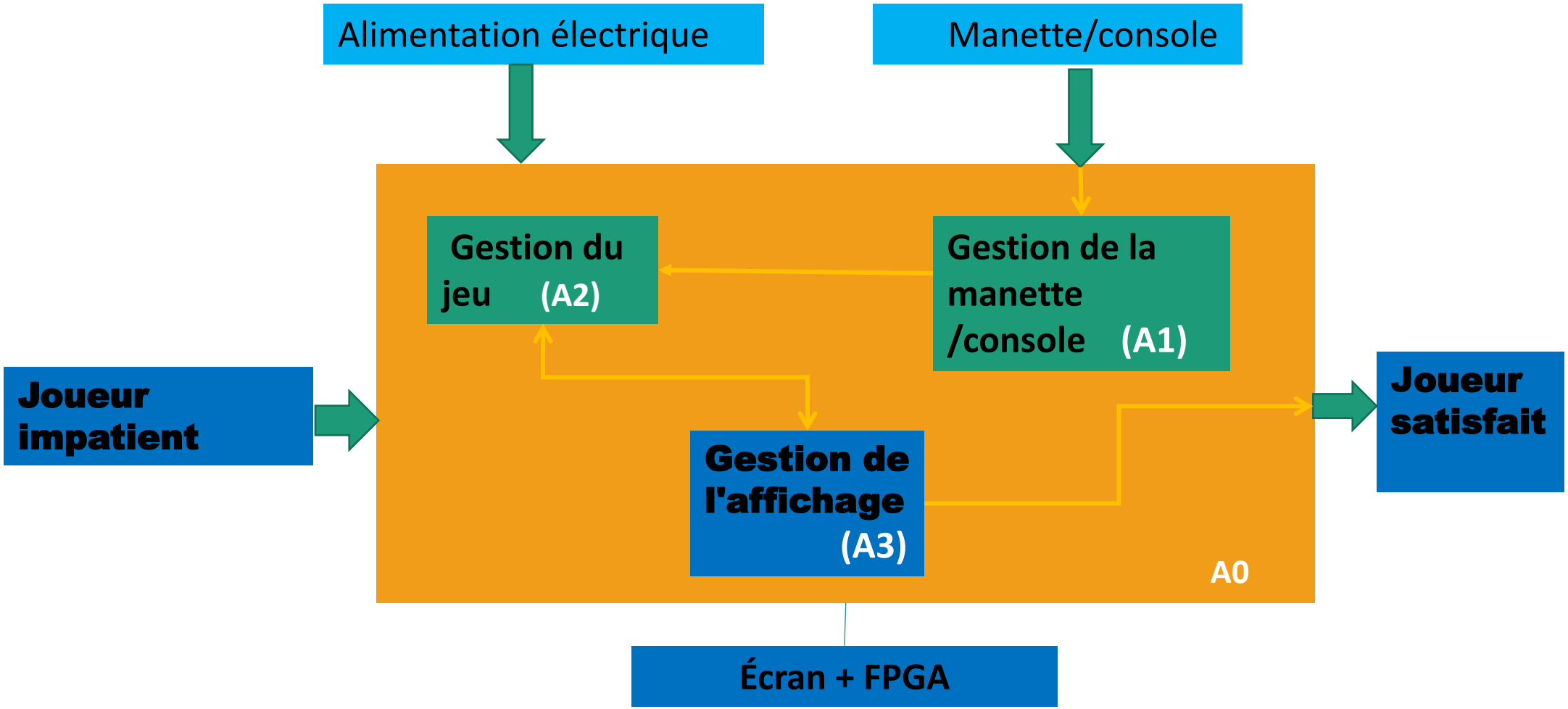


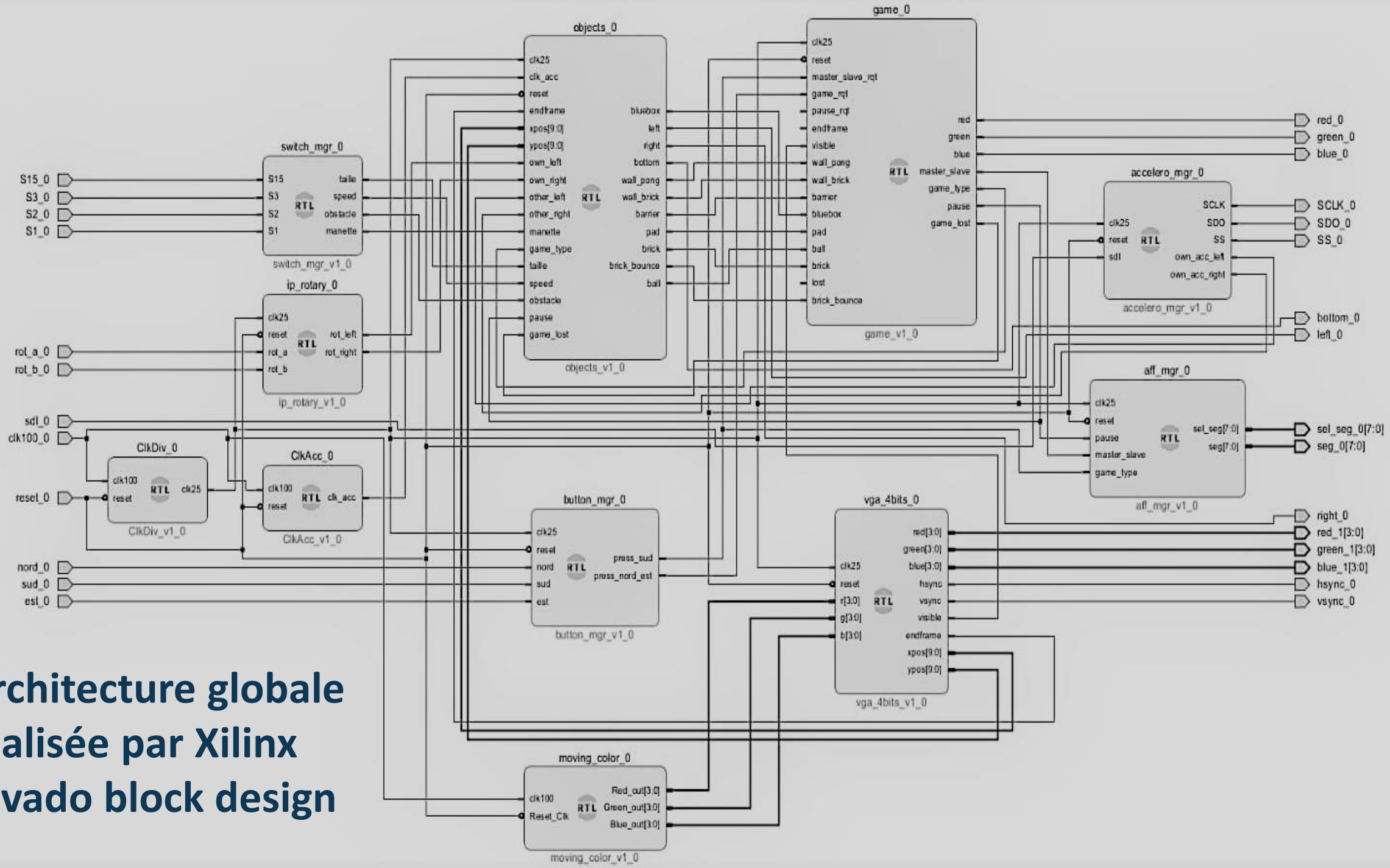
Sommaire

- 1. Diagramme SADT niveau A-0 et A0 du casse Brique et pong
- 2. Instruction générale de la console .
- 4.Présentation des taches 1 et 2 .
- 5.Les parties non validés.
- 5.Les contraints .
- 6.Conclusion.

The Nexys4 DDR

Diagramme SADT niveau A0 du casse brique et pong





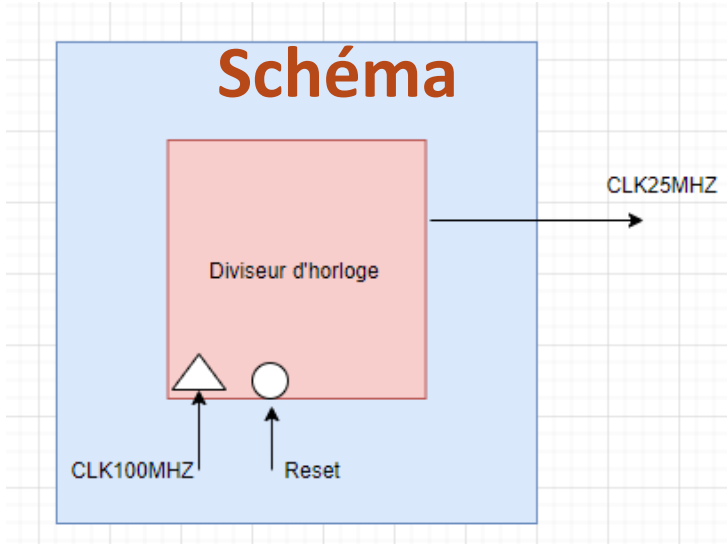
Architecture globale réalisée par Xilinx Vivado block design

<< ClkDiv 25 MHZ>>

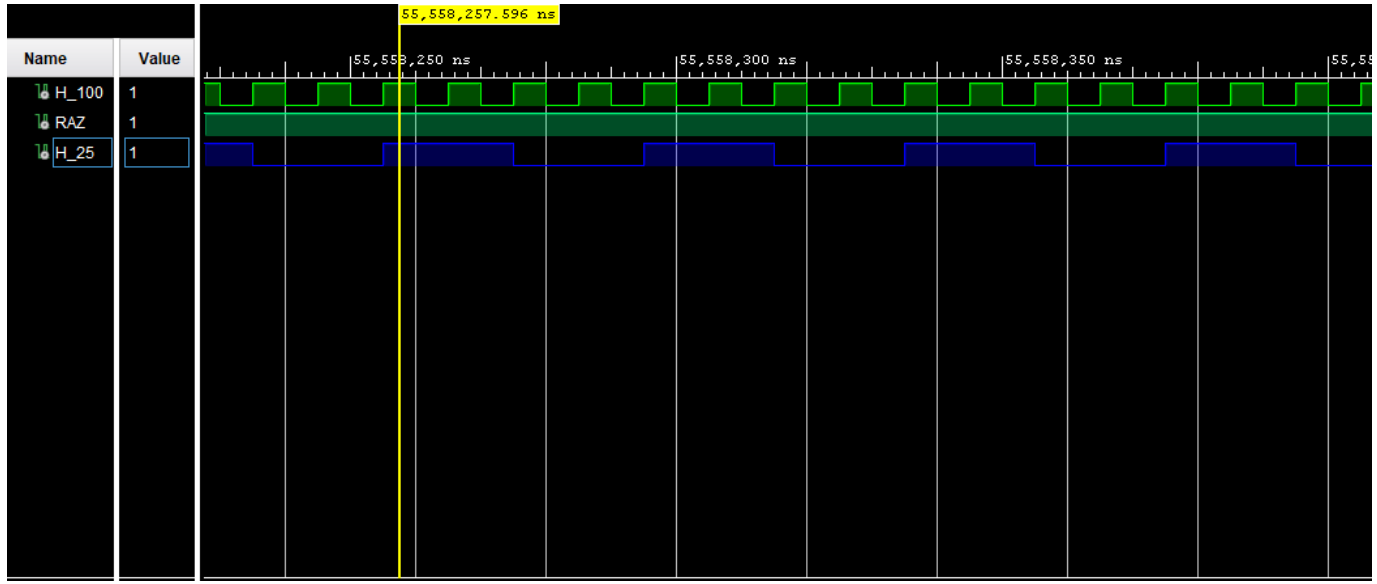
code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ClkDiv is
    Port ( clk100,reset : in  STD_LOGIC;     -- Horloge 100 Mhz et Reset Asynchrone
           clk25 : out  STD_LOGIC);          -- Horloge 25 MHz
end ClkDiv;
architecture Behavioral of ClkDiv is
-- Registre Horloge 25 MHz
signal clk: std_logic_vector(1 downto 0);
begin
-- Affectation Port de Sortie
clk25<=clk(1);
-----------------------------------------
-- DIVISION PAR 4 DE L'HORLOGE 100 MHZ
process(clk100,reset)
begin
if reset = '0' then
clk <= "00";
elsif rising_edge(clk100) then
clk <= clk + '1';
end if;
end process;
end Behavioral;
```

Schéma



Simulation

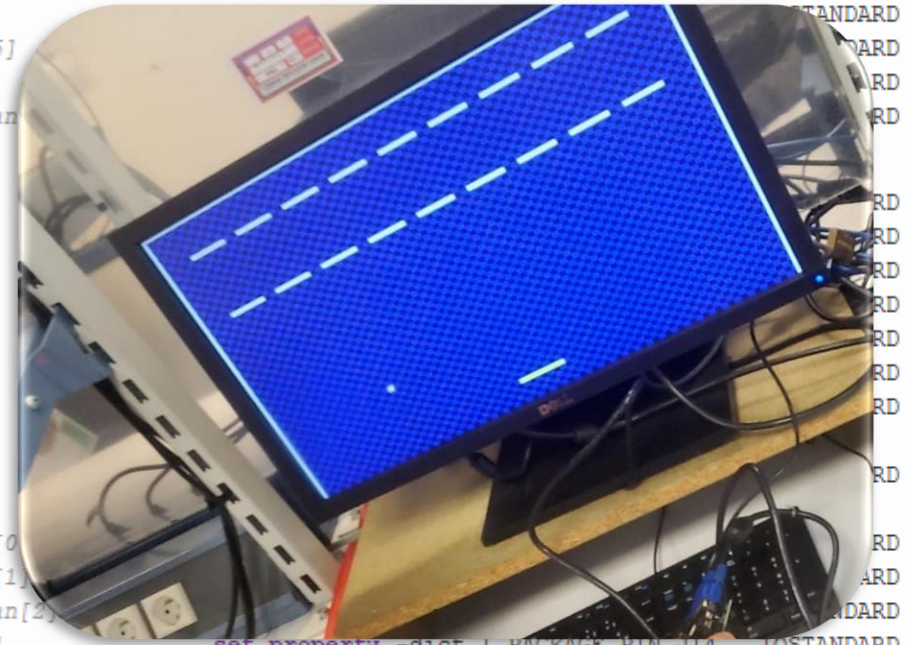


Console utilisée: Nexys4DDR.Xdc

Sources

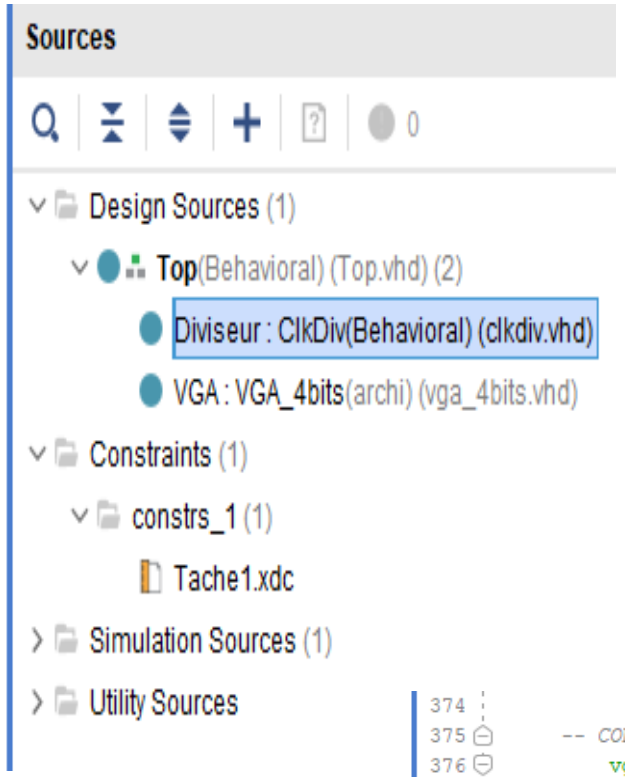
- top(Behavioral) (top.vhd) (10)
  - clk25MHz : ClkDiv(Behavioral) (clkdiv.vhd)
  - clk25Hz : ClkAcc(Behavioral) (ClkAcc.vhd)
  - switch : switch_mgr(Behavioral) (switch_mgr.vhd)
  - buttons : button_mgr(Behavioral) (button_mgr.vhd)
  - aff : aff_mgr(Behavioral) (aff_mgr.vhd)
  - codeur : ip_rotary(Behavioral) (ip_rotary.vhd) (1)
  - accelero_mgr : accelero_mgr(Behavioral) (accelero_mgr.vhd) (1)
  - obj_ctrl : objects(Behavioral) (objects.vhd) (5)
  - game_ctrl : game(Behavioral) (game.vhd) (3)
  - vga_ctrl : VGA(archi) (VGA.vhd)

Résultat obtenu sur écran VGA



<<Contrôleur VGA avec couleur sur 4 bits>>

L'objectif c'est de changer l'intensité de chaque couleur rouge, bleu ou vert.



On a modifié le module VGA en mettant red,green,bleu, sur 4 bits, plus on a changé le nom du module VGA sous le VGA_4bits

```
-- Affichage Couleur
red <= r when image_visible = '1' else "0000";
green <= g when image_visible = '1' else "0000";
blue <= b when image_visible = '1' else "0000";
```

Instanciation VGA_4bits dans top

```
374
375     -- CONTROLEUR VGA 4 bits
376         vga_ctrl: entity work.VGA_4bits
377         port map (              clk25   => clk25,       -- Horloge 25 MHz
378                 reset   => reset,       -- Reset Asynchrone
379                 r           => VGA_red_i,           -- Commande Affichage Rouge
380                 g           => VGA_green_i,     -- Commande Affichage Vert
381                 b           => VGA_blue_i,          -- Commande Affichage Bleu
382                 red         => VGA_red,     -- Affichage Pixel Rouge
383                 green   => VGA_green,   -- Affichage Pixel Vert
384                 blue        => VGA_blue,    -- Affichage Pixel Bleu
385                 hsync   => hsync,       -- Synchro Horizontale
386                 visible     => visible,     -- Zone Visible de l'Image
387                 endframe    => endframe,    -- Fin de l'Image Visible
388                 xpos    => xpos,            -- Coordonnee X du Pixel Courant
389                 ypos    => ypos);       -- Coordonnee Y du Pixel Courant
390
```

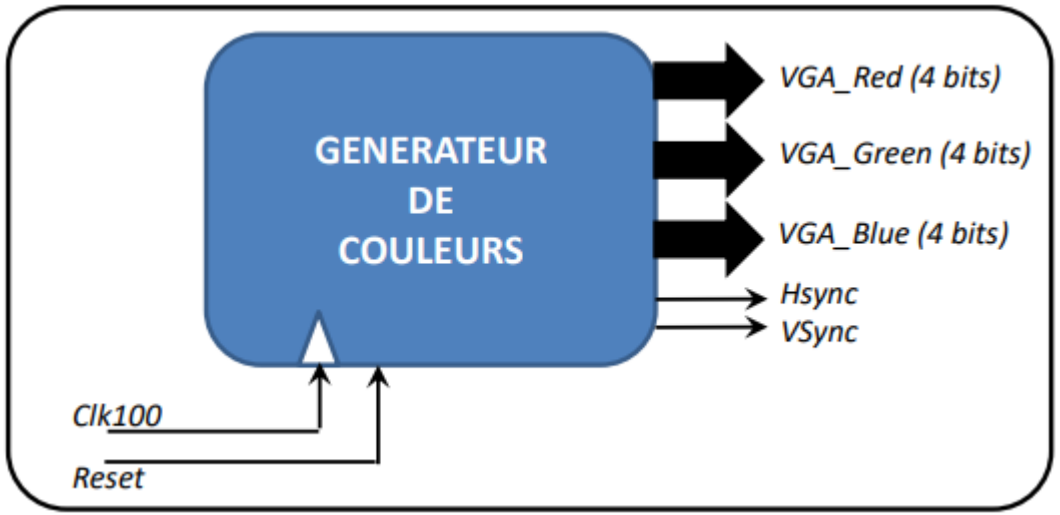
• L’affichage des couleurs sur écran VGA

- En jouant avec les interrupteurs de la carte nexys 4DDR on obtient les différents couleurs

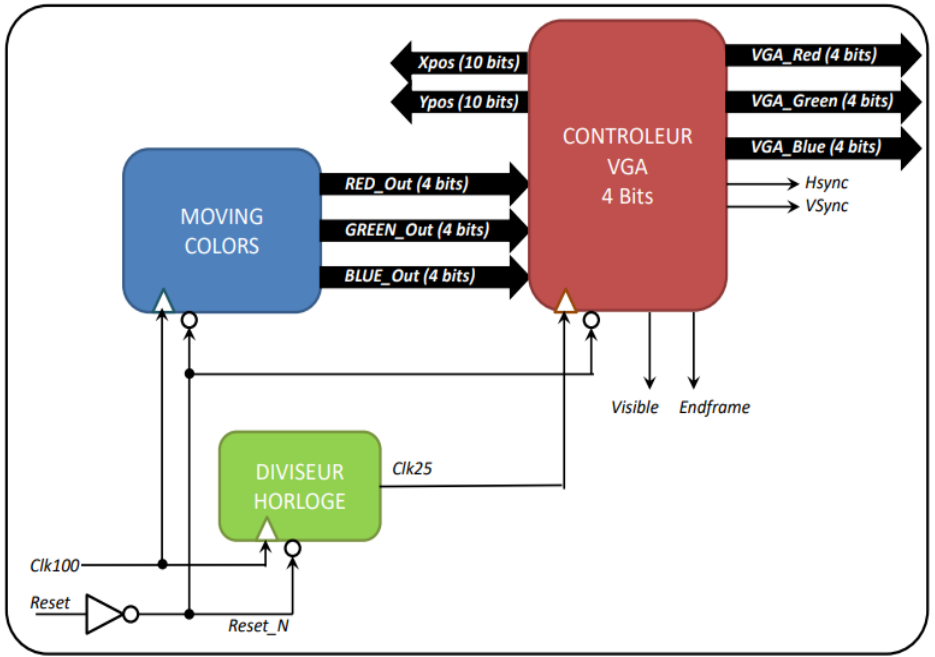
C:\Users\braha\OneDrive\Documents\sara\video-changement de teinte.mp4

<< Ajout d'un générateur de couleurs >>

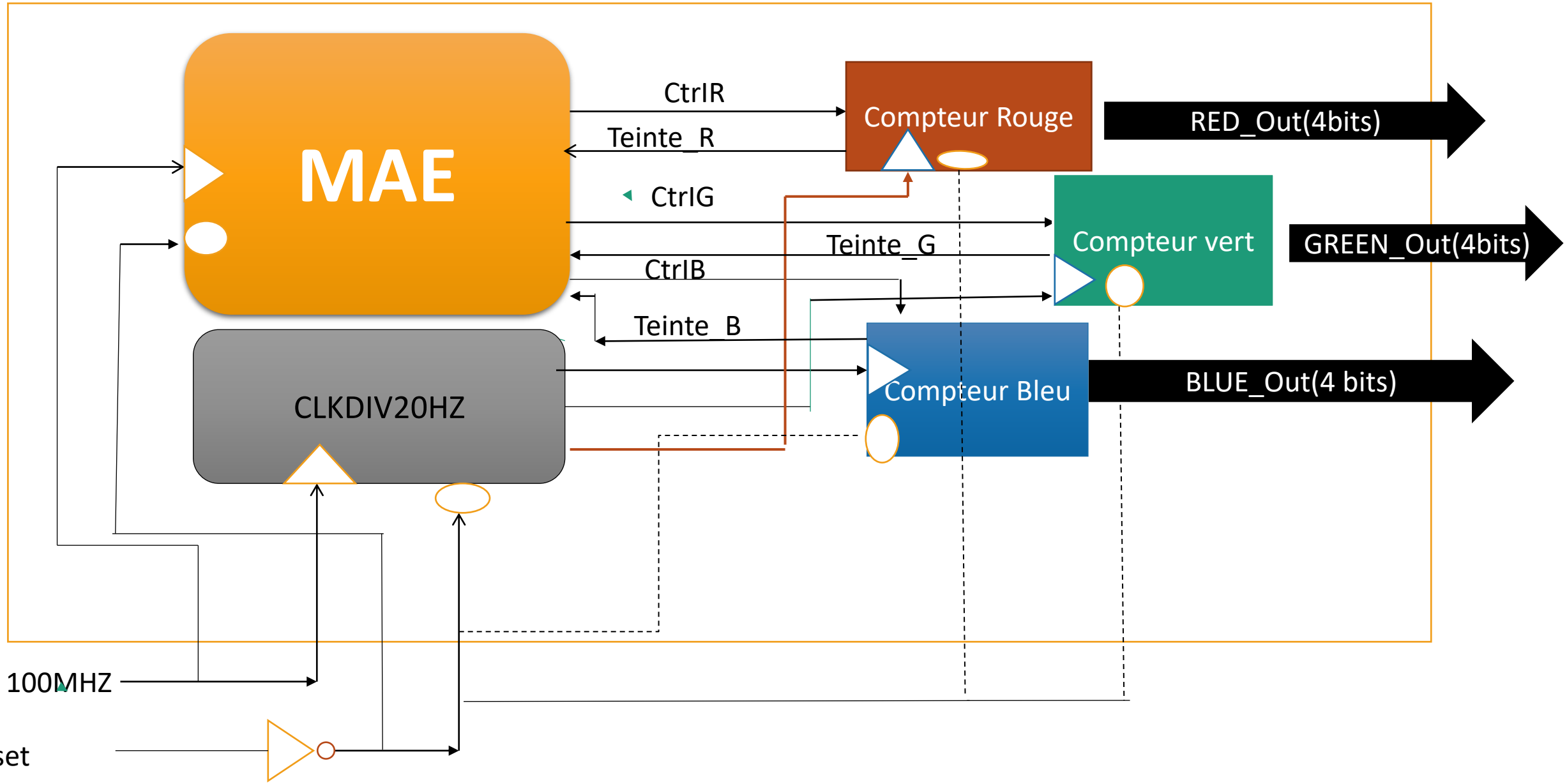
Maintenant, on va ajouter un générateur de couleur pour générer une série de couleurs sur 12 bits ,sans utiliser les interrupteurs pour modifier les teintes , pour réaliser cela on va utilisé une machine à état



À l'intérieur du module générateur de couleur on va instancier 3 modules



- Module **Moving Colors** comprend 5 sous modules :



Le code des différents modules

compteur_Rouge

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity compteur_Bleue is
port( Clk10,Reset:in std_logic;
      ComB: in std_logic_vector (1 downto 0);
      cpt_bleue: out std_logic_vector(4 downto 0));
end compteur_Bleue;
architecture archi of compteur_Bleue  is
signal cptB: std_logic_vector(4 downto 0);
begin
process(Clk10,Reset)
begin
if(Reset='0') then cptB <="00000";
elsif(rising_edge(Clk10))then
 case (comB) is
            when "00" => cptB<= cptB - '1';
            when "01" => cptB <= cptB  + '1';
            when "10" => cptB<= cptB ;
            when others => NULL;
                    end case;
end if;
end process ;
cpt_Bleue<=cptB;
end archi ;
```

Compteur_Bleu

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4
5  entity compteur_Rouge is
6  port( Clk10,Reset:in std_logic;
7        ComR: in std_logic_vector (1 downto 0);
8        cpt_Rouge: out std_logic_vector(4 downto 0));
9  end compteur_Rouge;
10 architecture archi of compteur_Rouge  is
11 signal cptR: std_logic_vector(4 downto 0);
12 begin
13 process(Clk10,Reset)
14 begin
15 if(Reset='0') then cptR <="11111";
16 elsif(rising_edge(Clk10))then
17 case (comR) is
18           when "00" => cptR<= cptR - '1';
19           when "01" => cptR <= cptR  + '1';
20           when "10" => cptR<= cptR ;
21           when others => NULL;
22       end case;
23 end if;
24 end process ;
25 cpt_Rouge<=cptR;
26 end archi ;
```

Compteur_Vert

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity compteur_vert is
port( Clk10,Reset:in std_logic;
      Comv: in std_logic_vector (1 downto 0);
      cpt_vert: out std_logic_vector(4 downto 0));
end compteur_vert;
architecture archi of compteur_vert  is
signal cptv: std_logic_vector(4 downto 0);
begin
process(Clk10,Reset)
begin
if(Reset='0') then cptv<="00000";
elsif(rising_edge(Clk10))then
 case (comv) is
            when "00" => cptv<= cptv - '1';
            when "01" => cptv <= cptv + '1';
            when "10" => cptv<= cptv;
            when others => NULL;
       end case;
 end if;
end process ;
cpt_vert<=cptv;
end archi ;
```

```vhdl
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned;
entity MAE is
port (
      clk100: in std_logic;  --Horloge cadencé à 100MHz
      teinteR: in std_logic_vector (4 downto 0); --entrée indiquant l'état du compteur rouge
      teinteG: in std_logic_vector (4 downto 0); --entrée indiquant l'état du compteur vert
      teinteB: in std_logic_vector (4 downto 0); --entrée indiquant l'état du compteur bleu
      CtrlR: out std_logic_vector (1 downto 0);  --sortie ordonnant l'augmentation, la diminution ou le maintien de la valeur de la teinte
      CtrlG: out std_logic_vector (1 downto 0);  --sortie ordonnant l'augmentation, la diminution ou le maintien de la valeur de la teinte
      CtrlB: out std_logic_vector (1 downto 0);  --sortie ordonnant l'augmentation, la diminution ou le maintien de la valeur de la teinte
      reset: in std_logic);
end MAE;
architecture behavioral of MAE is
type etat is (Phase1,Phase2,Phase3);
signal EP, EF: etat;
begin
--registre des états
process(clk100, reset)
begin
    if reset = '0' then EP <= Phase1;
    elsif rising_edge(clk100) then EP <= EF;
    end if;
end process;
```

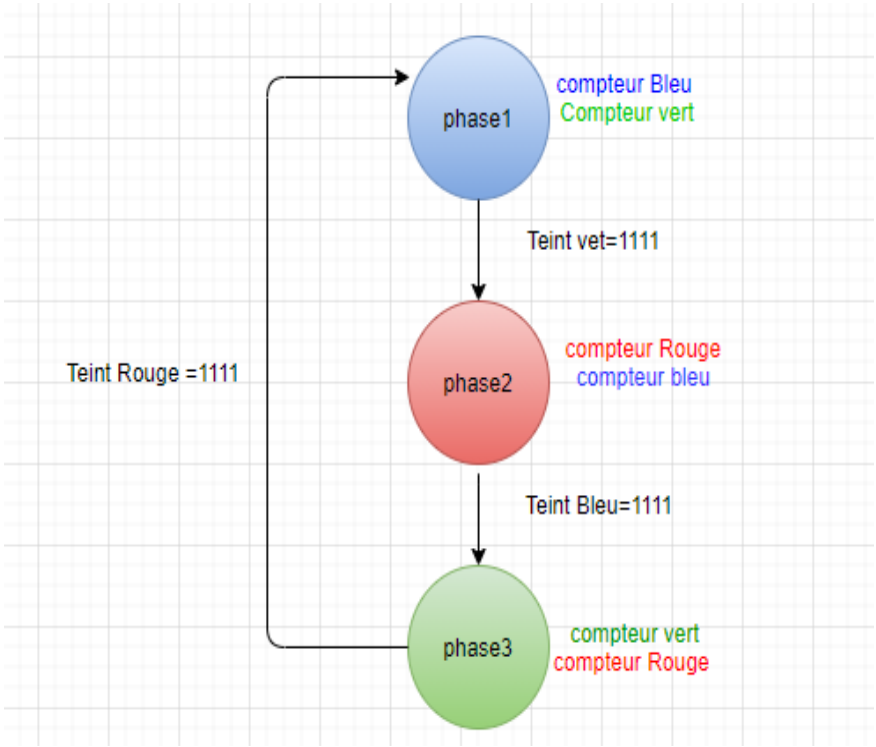
Code MAE

Phase	Teinte Rouge	Teinte Verte	Teinte Bleue
1	Diminution	Augmentation	Constante
2	Constante	Diminution	Augmentation
3	Augmentation	Constante	Diminution

On a déduit le graphe d'état à partir de ce tableau

Graphe d'état MAE

```vhdl
    process(EP,teinteR, teinteG, teinteB)    --combinatoire des états futurs
    begin
        case(EP) is
            when Phase1 => EF <=Phase1; if teinteG = "11111" then EF <= Phase2; end if;
            when Phase2 => EF <=Phase2; if teinteB = "11111" then EF <= Phase3; end if;
            when Phase3 => EF <=Phase3; if teinteR = "11111" then EF <= Phase1; end if;
        end case;
    end process;
    process(EP)                    --combinatoire des sorties
    begin
        case(EP) is
            when Phase1 =>
                CtrlR <= "00";
                CtrlG <= "01";
                CtrlB <= "10";
            when Phase2 =>
                CtrlR <= "10";
                CtrlG <= "00";
                CtrlB <= "01";
            when Phase3 =>
                CtrlR <= "01";
                CtrlG <= "10";
                CtrlB <= "00";
        end case;
    end process;
    end behavioral;
```



Code Moving_colors

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
entity Moving_Colors is
  Port (Clk100: in std_logic;
        Reset: in std_logic;
  RED_Out, GREEN_Out, BLUE_Out: out std_logic_vector (3 downto 0));
end Moving_Colors;
architecture archi of Moving_Colors is
signal clk10 : std_logic;
signal teinteR_signal : std_logic_vector (4 downto 0);
signal teinteG_signal : std_logic_vector (4 downto 0);
signal teinteB_signal : std_logic_vector (4 downto 0);
signal CtrlR_signal : std_logic_vector (1 downto 0);
signal CtrlG_signal : std_logic_vector (1 downto 0);
signal CtrlB_signal : std_logic_vector (1 downto 0);
begin
RED_Out <= teinteR_signal(4 downto 1);
GREEN_Out <= teinteG_signal(4 downto 1);
BLUE_Out <= teinteB_signal(4 downto 1);
my_a1: entity work.ClkDiv10
                port map(
                    clk100 => Clk100,    -- Horloge 100 Mhz
                    reset => Reset,    -- Reset Asynchrone
                    clk10 => clk10      -- Horloge 20 Hz
```

```
                );
CompteurR : entity work.compteur_Rouge
                port map(
      clk10 => clk10,  --Horloge de 20Hz en entrée
      reset => Reset, --Reset asynchrone actif à l'état bas
      cpt_Rouge=> teinteR_signal, --Teinte indiqué à la sortie du compteur
      ComR => CtrlR_signal  --Commande qui contrôle l'incrémentation, la décrémentation ou le maintien de la valeur du compteur
                );
CompteurG : entity work.compteur_vert
                port map(
                 clk10 =>clk10,
                 reset => Reset, --Reset asynchrone actif à l'état bas
                 cpt_vert=> teinteG_signal, --Teinte indiqué à la sortie du compteur
               Comv => CtrlG_signal  --Commande qui contrôle l'incrémentation, la décrémentation ou le maintien de la valeur du compteur
                );
CompteurB : entity work.compteur_bleue
                port map(
                    clk10 =>clk10,
                    reset => Reset, --Reset asynchrone actif à l'état bas
                    cpt_Bleue => teinteB_signal, --Teinte indiqué à la sortie du compteur
               ComB => CtrlB_signal  --Commande qui contrôle l'incrémentation, la décrémentation ou le maintien de la valeur du compteur
                );
MAE : entity work.MAE
       port map(
           clk100 => Clk100,  --Horloge cadencé à 100MHz
           teinteR => teinteR_signal, --entrée indiquant l'état du compteur rouge
```

Instanciation compteur, Rouge, vert, Bleu dans le module moving_colors

```
52              teinteG => teinteG_signal, --entrée indiquant l'état du compteur vert
53              teinteB => teinteB_signal, --entrée indiquant l'état du compteur bleu
54              CtrlR => CtrlR_signal,  --sortie ordonnant l'augmentation, la diminution ou le maintien de la valeur de la teinte rouge
55              CtrlG => CtrlG_signal,  --sortie ordonnant l'augmentation, la diminution ou le maintien de la valeur de la teinte verte
56              CtrlB => CtrlB_signal,  --sortie ordonnant l'augmentation, la diminution ou le maintien de la valeur de la teinte bleu
57              reset => Reset  --reset asynchrone
58          );
59      end archi;
60
```

Instanciation MAE

testbench.vhd

VHDL Testbench

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5  entity simu is
6  end simu;
7
8  architecture archi of simu is
9
10 signal Clk100,Reset: std_logic:='0';
11 signal clkmove: std_logic;
12
13 begin
14
15 -- Instanciation du Module à Simuler
16 My_Adder: entity work.Clk
17 port map(
18              clk100      => Clk100,
19              reset   => Reset,
20              clkmove => clkmove
21         );
22
23 --Valeur des Entrées
24 Clk100 <= not Clk100 after 50 ns;
25 Reset <= '1' after 10 ns;
26
27 end archi;
28
```

design.vhd

VHDL Design

```
11
12 -- Compteur pour Horloge 20 Hz
13 signal CPT20: std_logic_vector(21 downto 0);
14
15 -- Signal Tampon pour l'horloge 20HZ
16 signal Clk20: std_logic;
17
18
19 begin
20
21 -- Affectation Horloge 20hz
22 Clkmove <= Clk20;
23
24
25 ------------------------------------------------
26 -- GESTION DES COMPTEURS DE DIVISION
27 --       ET GENERATION DE L'HORLOGE 20HZ
28 process(clk100,reset)
29
30     begin
31
32         if reset = '0' then
33
34             Clk20 <= '0'; CPT20 <= (others => '0');
35
36         elsif rising_edge(clk100) then
37
38             CPT20 <= CPT20+1;
39
40             if (CPT20 = "10011000100101101000000") then
41                 CPT20 <= (others => '0');
42                 Clk20<= not Clk20;
43             end if;
44
45         end if;
46
47 end process;
48
49 end archi;
50
```

Code de l'horloge 20hz utilisé pour l'implémentation

testbench.vhd

VHDL Testbench

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity simu is
end simu;

architecture archi of simu is

signal Clk100,Reset: std_logic:='0';
signal clkmove: std_logic;

begin

-- Instanciation du Module à Simuler
My_Adder: entity work.Clk
port map(
            clk100      => Clk100,
            reset   => Reset,
            clkmove => clkmove
        );

--Valeur des Entrées
Clk100 <= not Clk100 after 50 ns;
Reset <= '1' after 10 ns;

end archi;
```

Teste_bench

design.vhd

VHDL Design

```vhdl
-- Signal Tampon pour l'horloge 10Mhz
signal Clk10: std_logic;


begin

-- Affectation Horloge 10Mhz
Clkmove <= Clk10;


-----------------------------------------
-- GESTION DES COMPTEURS DE DIVISION
--       ET GENERATION DE L'HORLOGE 10MHZ
process(clk100,reset)

    begin

        if reset = '0' then

            Clk10 <= '0'; CPT10 <= (others => '0');

        elsif rising_edge(clk100) then

            CPT10 <= CPT10+1;

            if (CPT10 = 101) then
                CPT10 <= (others => '0');
                Clk10<= not Clk10;
            end if;

        end if;

end process;

end archi;
```

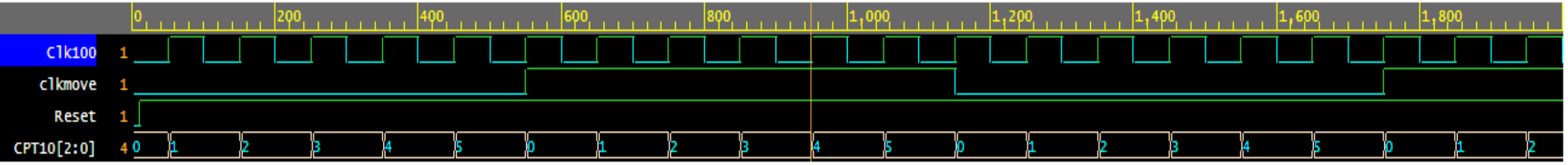
Code de l'horloge 10 Mhz utilisé pour la simulation

EPWave

From: 0ns To: 2,000ns

Get Signals | Radix | 100% | 950ns | simu/Clk100

Simulation



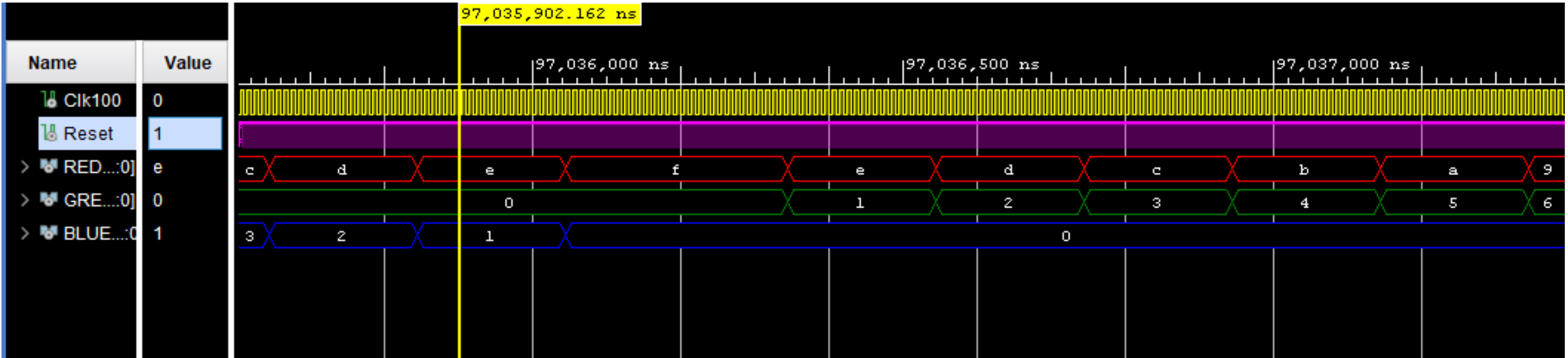
Teste_bench de Moving_Colors

```
1
2   -- Testbench ALU--
3   library IEEE;
4   use IEEE.std_logic_1164.all;
5   use IEEE.std_logic_unsigned.all;
6   entity SIMU is
7   end SIMU;
8   architecture ARCHI of SIMU is
9   signal Clk100,Reset:std_logic:='0';
10  signal RED_Out,GREEN_Out,BLUE_Out:std_logic_vector (3 downto 0);
11  begin
12  my_ALU: entity work.Moving_Colors
13   port map(Clk100,Reset,RED_Out, GREEN_Out, BLUE_Out);
14  Clk100 <= NOT (Clk100) after 5 ns;
15  Reset <= '1' after 2 ns;
16  end archi;
```

Scope | Sources

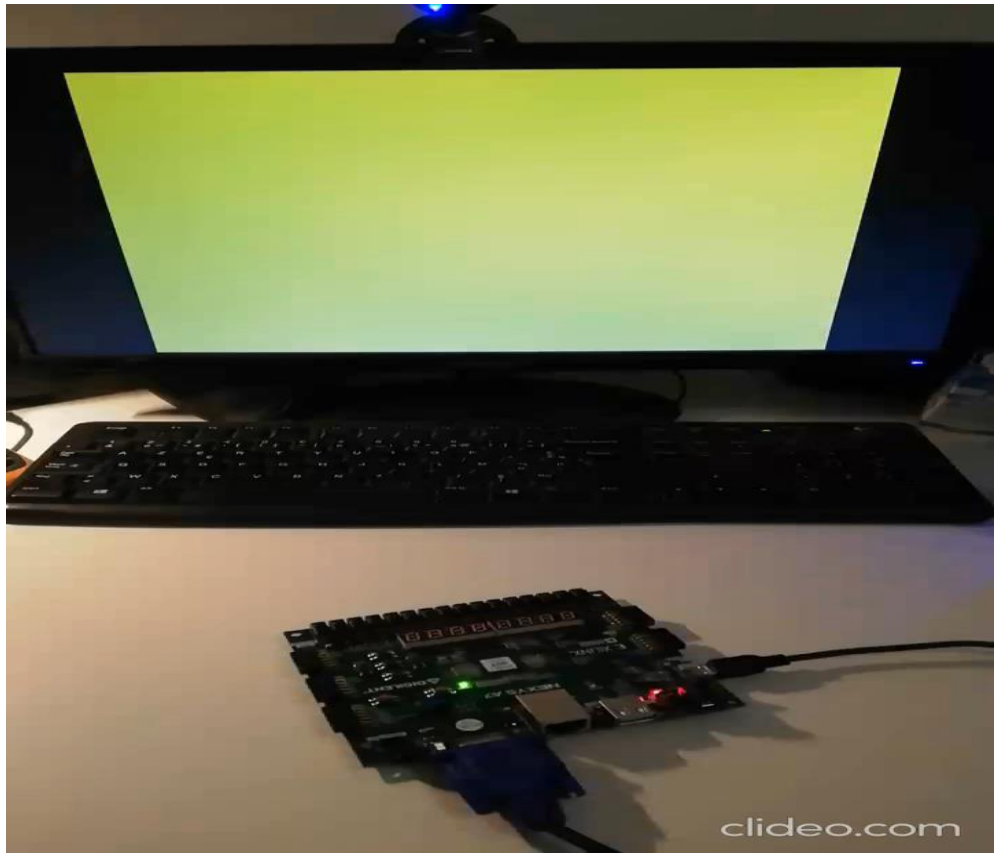
- Design Sources (1)
  - **Moving_Colors**(archi) (moving_colors.vhd) (5)
    - my_al : ClkDiv10(archi) (Clkdiv10.vhd)
    - CompteurR : compteur_Rouge(archi) (compteur_Rouge.vhd)
    - CompteurG : compteur_vert(archi) (compteur_vert.vhd)
    - CompteurB : compteur_Bleue(archi) (compteur_Bleue.vhd)
    - MAE : MAE(behavioral) (MEA.vhd)
- Constraints
- Simulation Sources (1)
  - sim_1 (1)
    - **SIMU**(ARCHI) (test_bench.vhd) (1)
      - my_ALU : Moving_Colors(archi) (moving_colors.vhd) (5)
- Utility Sources

Simulation



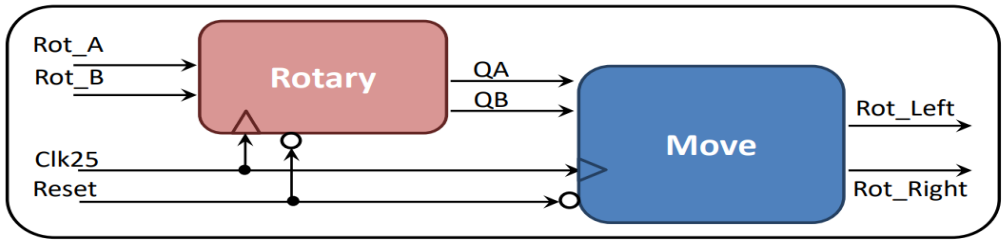
Cette fois-ci sans jouer avec les interrupteurs les niveaux des teintes changent tout seul à cause de la machine à état qui permet de changer les états d'une phase à une autre .

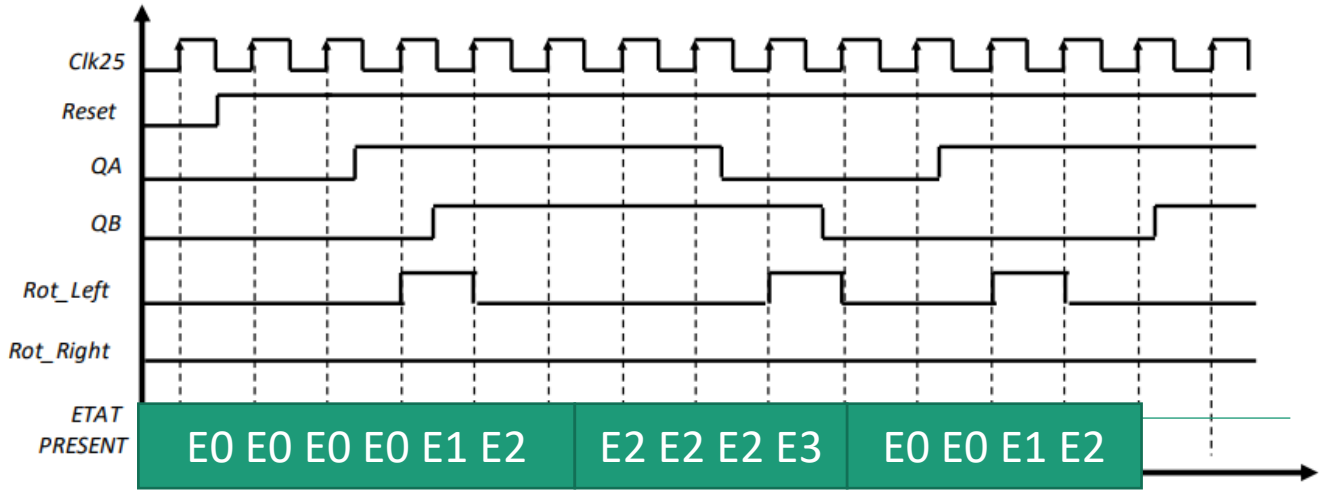
C:\Users\braha\OneDrive\Documents\move-colors\video_jem9.mp4

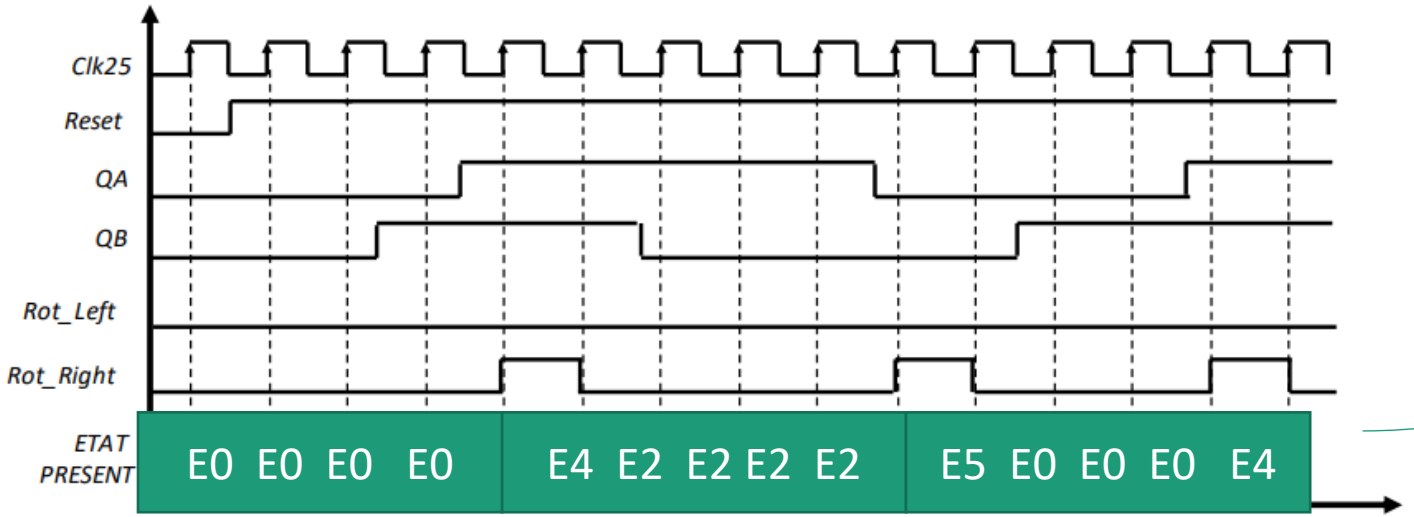


<<Gestion de la raquette avec l'encodeur rotatif >>

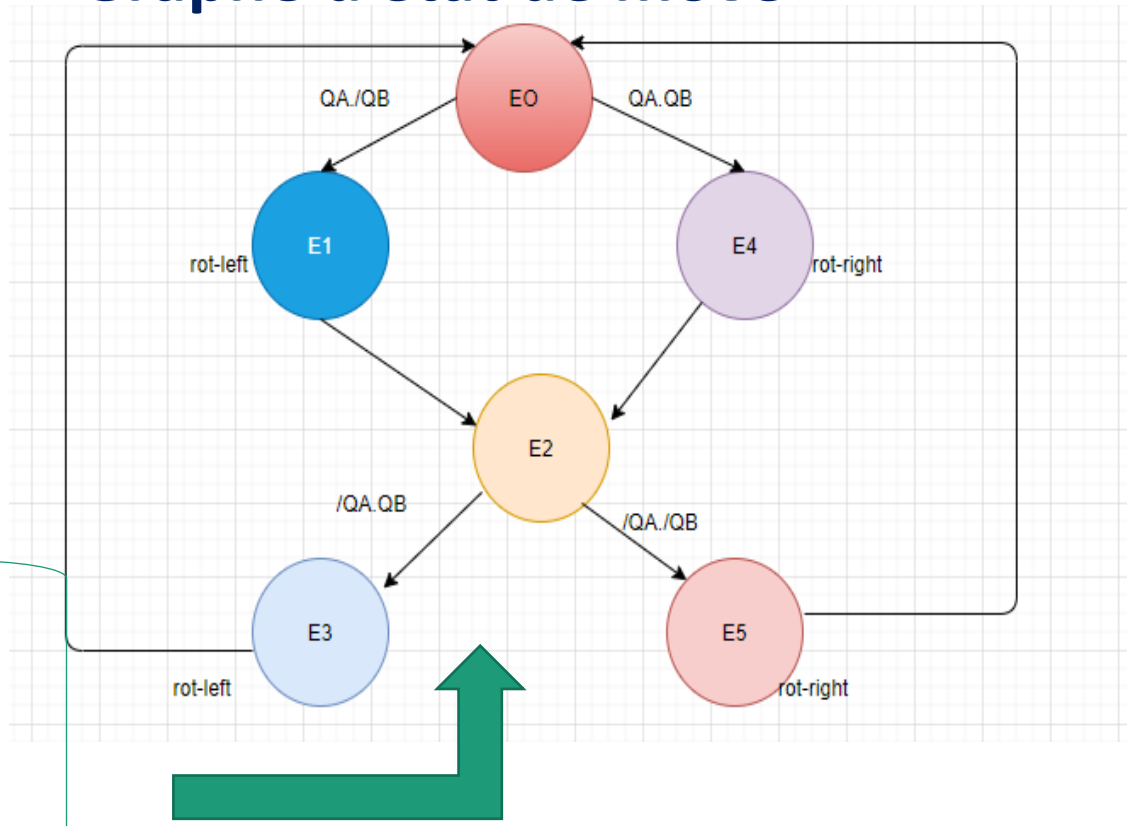
Module IP_Rotatry







Graphe d'état de move



A partir de ces deux chronogrammes on a déduit le graphe d'état

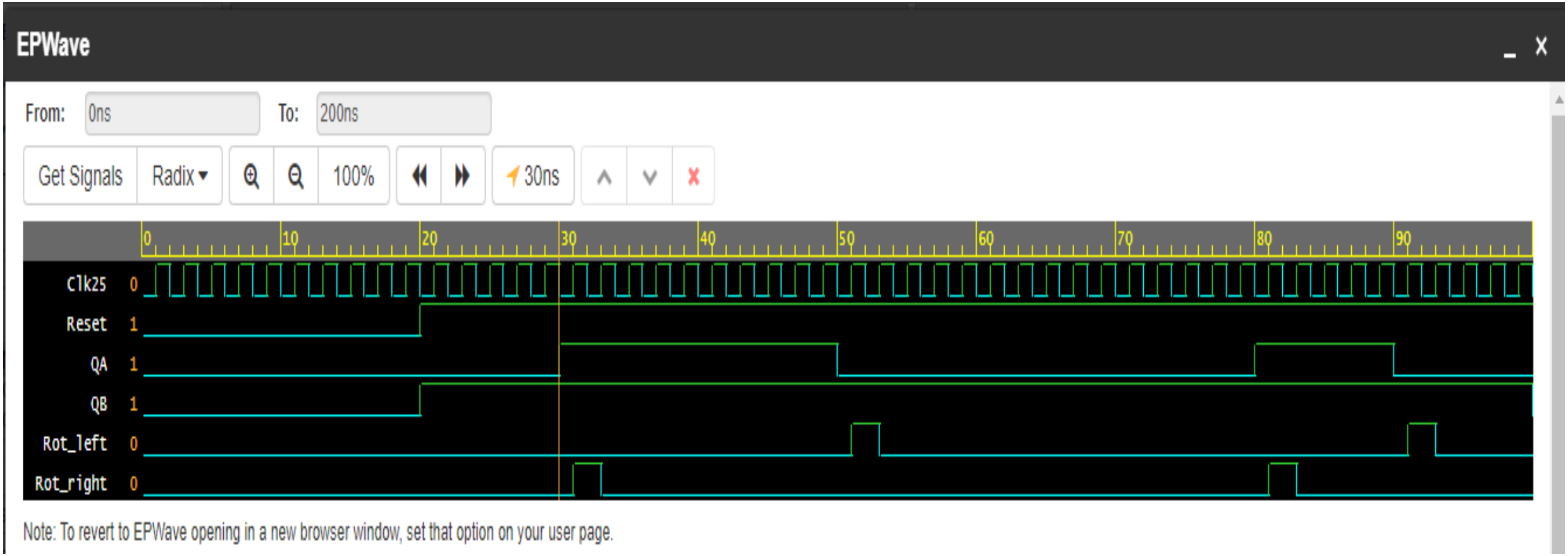
Code move

```
entity move is
    Port ( clk25 : in STD_LOGIC;
           Reset : in STD_LOGIC;
           QA : in STD_LOGIC;
           QB : in STD_LOGIC;
           rot_left : out STD_LOGIC;
           rot_right : out STD_LOGIC);
end move;
architecture Behavioral of move is
type etat is (E0,E1,E2,E3,E4,E5);
signal EP,EF:etat ;
begin
--process du registre d'etat--
process(clk25,Reset)
begin
if Reset='0' then EP <= E0;
elsif rising_edge(clk25) then EP<= EF;
end if ;
end process;
--combinatoire des états futurs--

process(EP,QA,QB)
begin
case(EP) is
when E0 => EF <= E0; if (QA='1' and QB='0') then EF <= E1;
                     elsif (QA='1' and QB='1') then EF <= E4;
                     end if;
when E1 => EF <= E2 ;
when E2=> EF<= E2 ; if(QA='0' and QB='0') then EF<=E5;
                    elsif(QA='0' and QB='1') then EF<=E3;
                    end if;
 when E3 => EF <= E0;
 when E4 => EF <= E2;
 when E5 => EF <= E0;
 end case ;
 end process;
 --combinatoir des sorties --
 process(EP)
 begin
 case(EP) is
 when E0 => Rot_left <= '0'; Rot_right <='0';
 when E1 => Rot_left<=  '1';   Rot_right<='0';
 when E2 => Rot_left <= '0'; Rot_right <='0';
 when E3 => Rot_left <= '1'; Rot_right <='0';
 when E4 => Rot_left <= '0'; Rot_right <='1';
 when E5 => Rot_left <= '0'; Rot_right <='1';
 end case ;
 end process;

end Behavioral;
```

Test_bench move

```
-- Testbench
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity SIMU is
end SIMU;

architecture ARCHI of SIMU is

signal Clk25,Reset:std_logic:='0';
signal QA,QB:std_logic:='0';
signal Rot_left,Rot_right:std_logic;

begin

my_ALU: entity work.move
 port map(Clk25,Reset,QA,QB,Rot_left,Rot_right);
QA <= '1' after 30 ns, '0'after 50 ns ,'1'after 80 ns,'0' after 90 ns;
QB <= '1' after 20 ns ,'1' after 70 ns ,'0' after 100 ns;
Clk25<= NOT (Clk25) after 1 ns;
Reset<= '1' after 20 ns;
end archi;
```

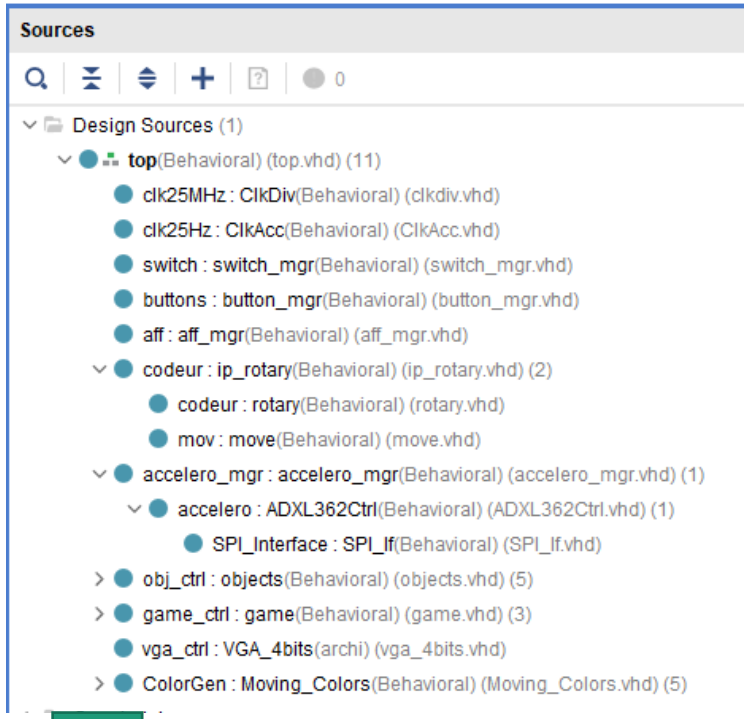
Simulation teste bench du move



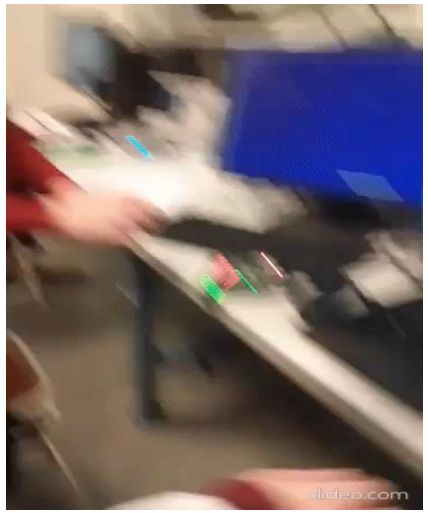
L’instanciation du module move dans le module ip_rotary

```
40      ------------------------------------------------------------------
41      --L'INSTANCIATION DU MODULE MOVE --
42  mov: entity work.move
43          port map (
44              clk25       => clk25,       -- Horloge
45              Reset       => reset,       -- Reset Asynchrone
46              Rot_left     => rot_left,       --
47              Rot_right =>    rot_right,      -- Switch B du Codeur
48              QA          => qa,          -- Comportement du Switch A (Filtre)
49              QB          => qb);         -- Comportement du Switch B (Filtre)
50
```

Move prend sa place dans la hiérarchie de la console

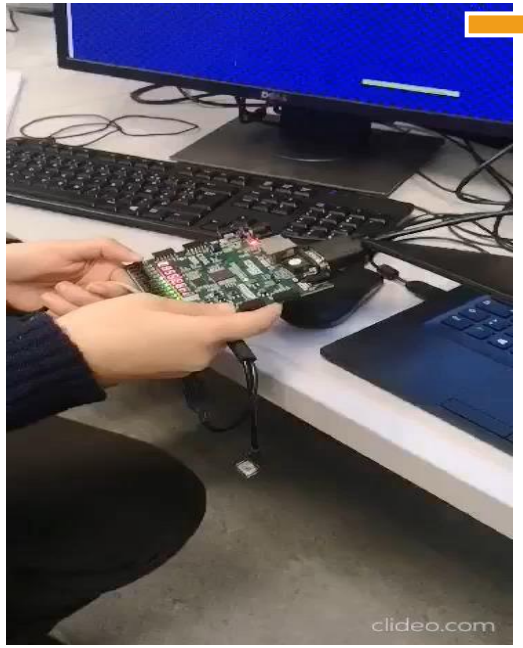


C:\Users\braha\OneDrive\Documents\sara\video-rotatry_.mp4

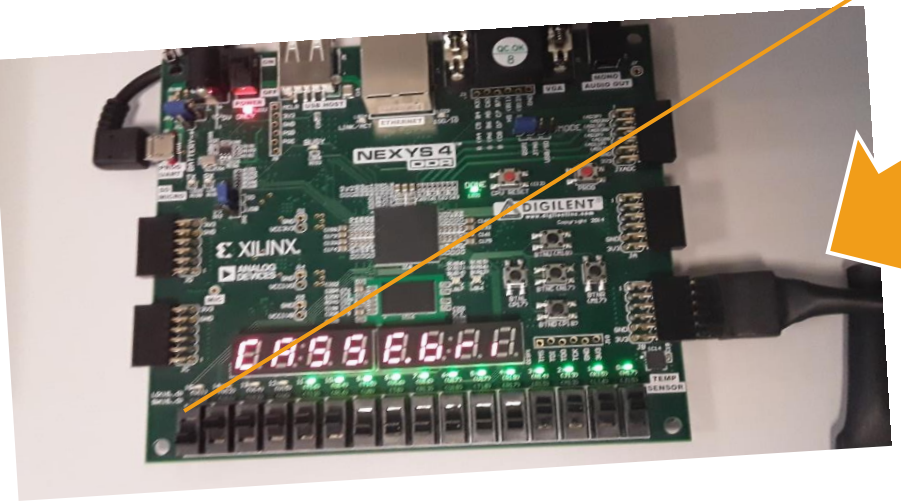


Après l’implémentation sur la carte la nouvelle version on a eu ce résultat

C:\Users\braha\OneDrive\Documents\ rotatry/video-1608824191_fVGLLOYK_Ge17.mp4



On peut également déplacer la raquette en utilisant l'encodeur rotatif et en choisissant l'interrupteur **S15** vers le bas de la carte nexys 4DDR



On y connecte l'encodeur rotatif en mode console , pmdoB

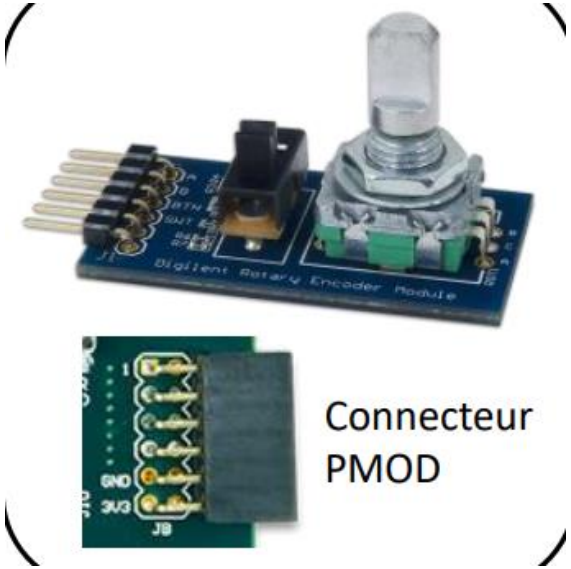


Figure extrait du manuel de l'utilisateur

- **<u>Les parties qui ne sont pas validées :</u>**

<<Gestions du jeux >>

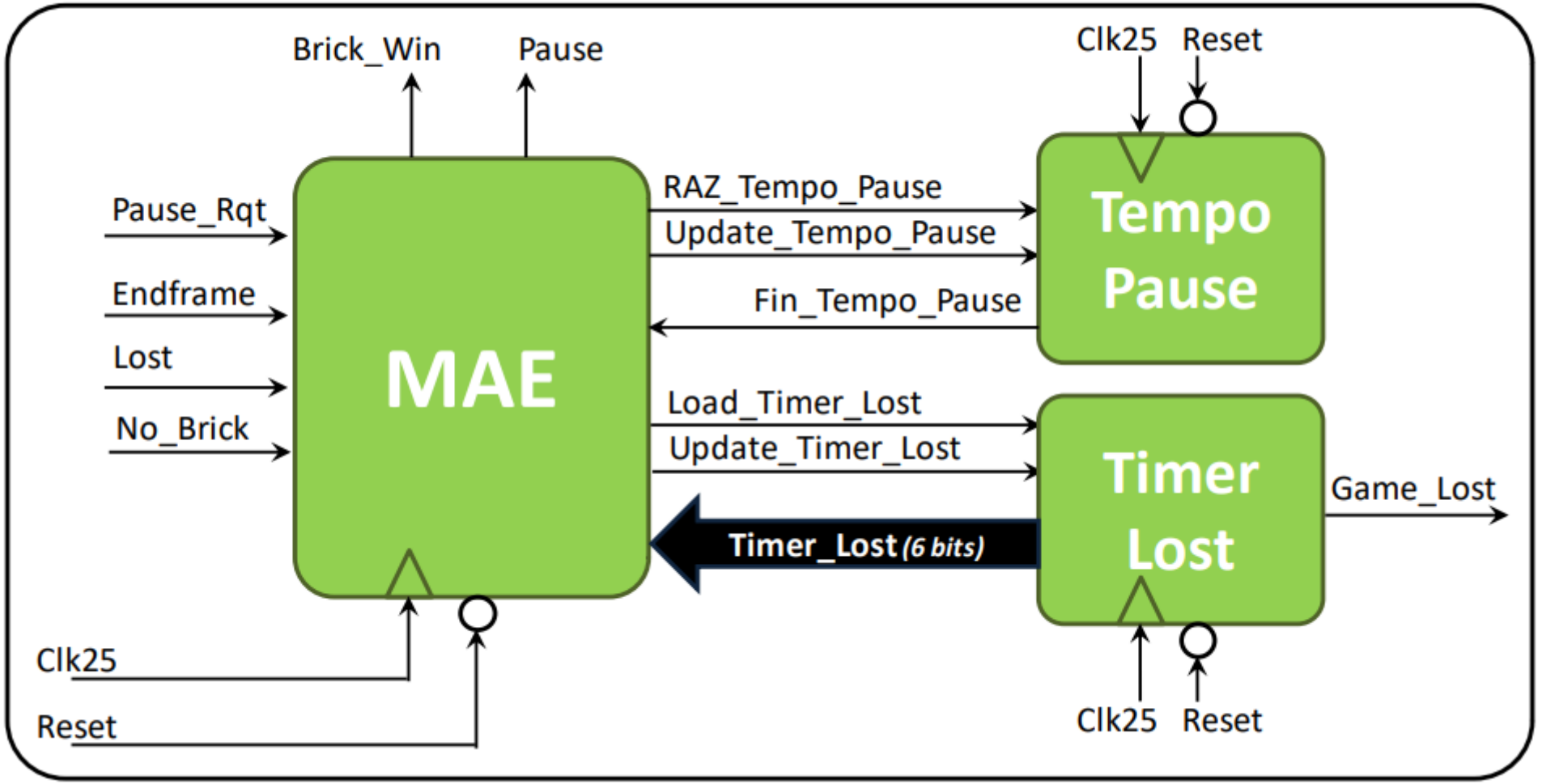


Figure extrait du manuel TP

testbench.vhd

VHDL Testbench

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;

entity simu is
end simu;

architecture archi of simu is
signal clk25,Reset:std_logic:='0';
signal
Update_Timer_Lost,Load_Timer_Lost,Game_Lost:std_logic:='
0';
begin
 My :  entity work.Timer_Lost
port map(Update_Timer_Lost,Load_Timer_Lost,clk25,Reset,
Game_Lost);

clk25<= not(clk25) after 1 ns ;
Reset <= '1' after 1 ns ;
Update_Timer_Lost <= '1' after 36 ns ;
Load_Timer_Lost <= '1' after  3 ns ,'0' after  40 ns;

end archi;
```

Test_bench_Timer_Lost

design.vhd

VHD

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity  Timer_Lost is
Port ( Update_Timer_Lost,Load_Timer_Lost: in std_logic:='0';
           CLK25, Reset: in std_logic;
           Game_Lost: out std_logic:='0');
 end Timer_Lost;

 architecture archi of Timer_Lost is
 signal compteur: std_logic_vector(5 downto 0);

  begin

     process(Reset,CLK25)
   begin
   if(Reset='0' ) then compteur<=(others=>'0');

   elsif (rising_edge(CLK25) ) then
    if Load_Timer_Lost ='1' then compteur <=(others=>'1');
   elsif Update_Timer_Lost = '1' then  compteur <=compteur -'1';
   elsif(Load_Timer_Lost ='0'and Update_Timer_Lost  ='0') then  compteur<=
compteur;
   end if;
  end if;
   end process;
  Game_Lost <= '1' when compteur = "000000"  else '0';
```

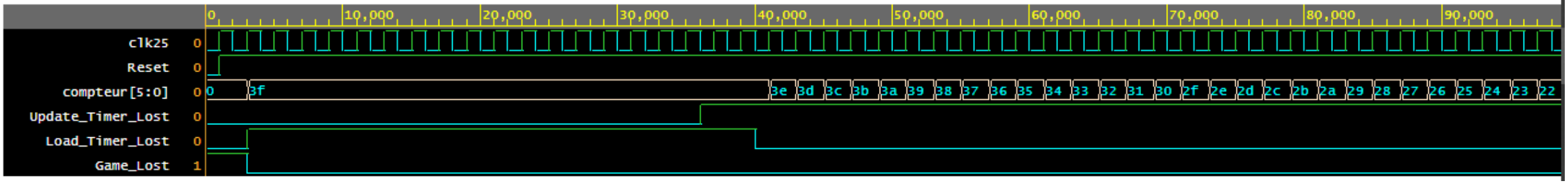
Code_Timer_Lost

EPWave

From: 0ps To: 100,000ps

Get Signals | Radix | 100% | 0ps

Simulation Timer_Lost



testbench.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.std_logic_unsigned.all;
4
5  entity simu is
6  end simu;
7
8  architecture archi of simu is
9  signal clk25,Reset:std_logic:='0';
10 signal
   RAZ_Tempo_Pause,Update_Tempo_Pause,
   Fin_Tempo_Pause:std_logic:='0';
11 begin
12  My :  entity work.tempo_pause
13 port
   map(RAZ_Tempo_Pause,Update_Tempo_Pause
   ,clk25,Reset, Fin_Tempo_Pause);
14
15 clk25<= not(clk25) after 1 ns ;
16 Reset <= '1' after 10 ns ;
17 RAZ_Tempo_Pause <= '0' after 30 ns ;
18 Update_Tempo_Pause <= '1' after 4ns ;
19
20 end archi;
21
```

VHDL Testbench

Test_bench_

design.vhd

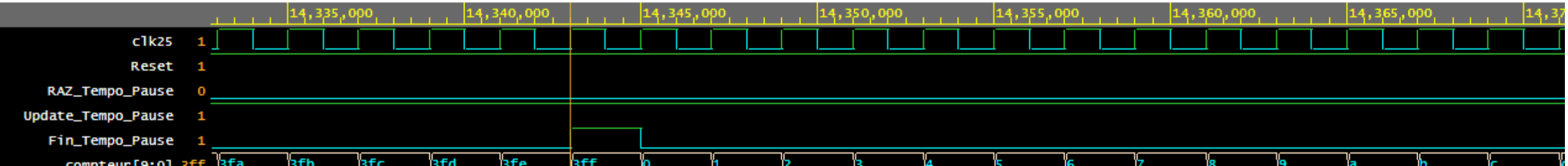
```
4
5  entity  tempo_pause is
6  Port ( RAZ_Tempo_Pause,Update_Tempo_Pause: in std_logic:='0';
7            CLK25, Reset: in std_logic;
8            Fin_Tempo_Pause: out std_logic:='0');
9   end tempo_pause;
10
11  architecture archi of tempo_pause is
12  signal compteur: std_logic_vector(9 downto 0);
13  signal k:std_logic;
14
15   begin
16
17   process(CLK25,Reset)
18
19   begin
20
21    if(Reset='0' ) then compteur<=(others=>'0');
22
23    elsif (rising_edge(CLK25) ) then
24     if RAZ_Tempo_Pause='1'then compteur <=(others=>'0');
25    elsif Update_Tempo_Pause='1'then  compteur <=compteur +'1';
26    elsif(RAZ_Tempo_Pause='0'and Update_Tempo_Pause='0') then  compteur<= compteur;
27
28    end if;
29    end if ;
30
31
32    end process;
33    Fin_Tempo_Pause <='1' when  compteur="1111111111" else '0';
34    end archi ;
```

Code_tempo_pause

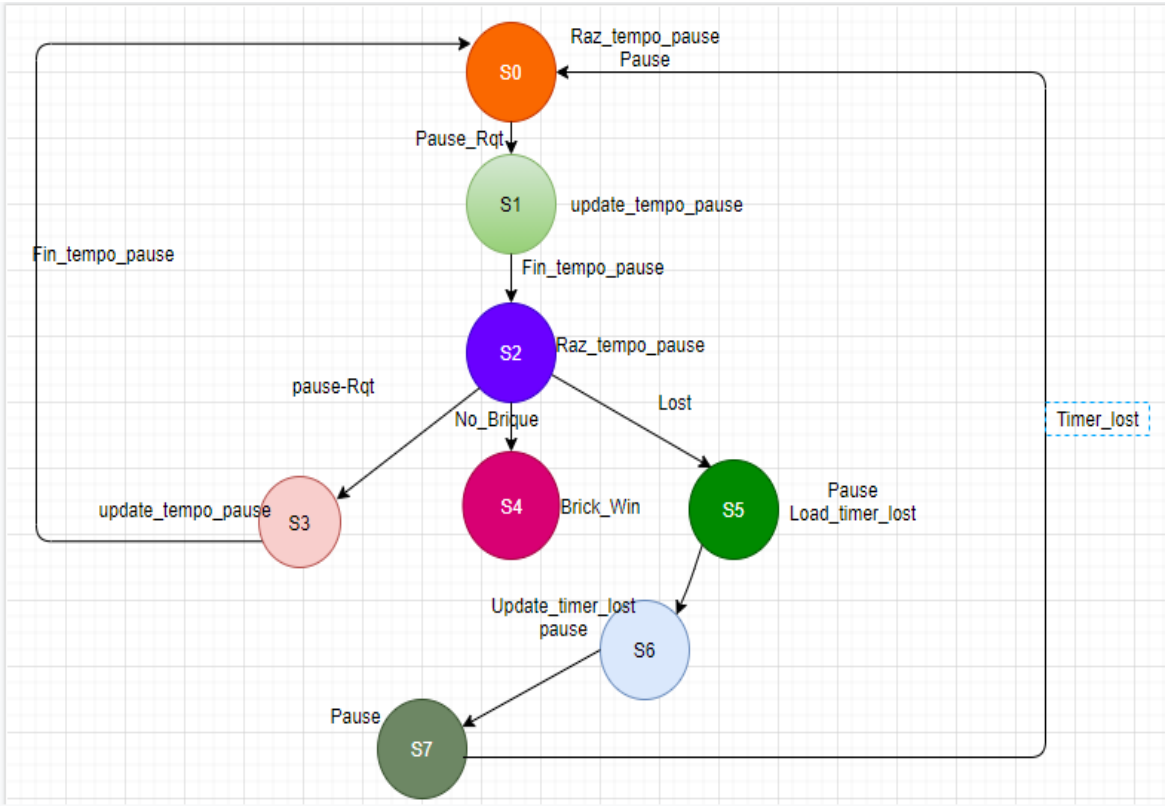
From: 0ps To: 19,982,000ps

Get Signals | Radix | 100% | 14,343,000ps

Simulation



Graphe d'état de la MAE

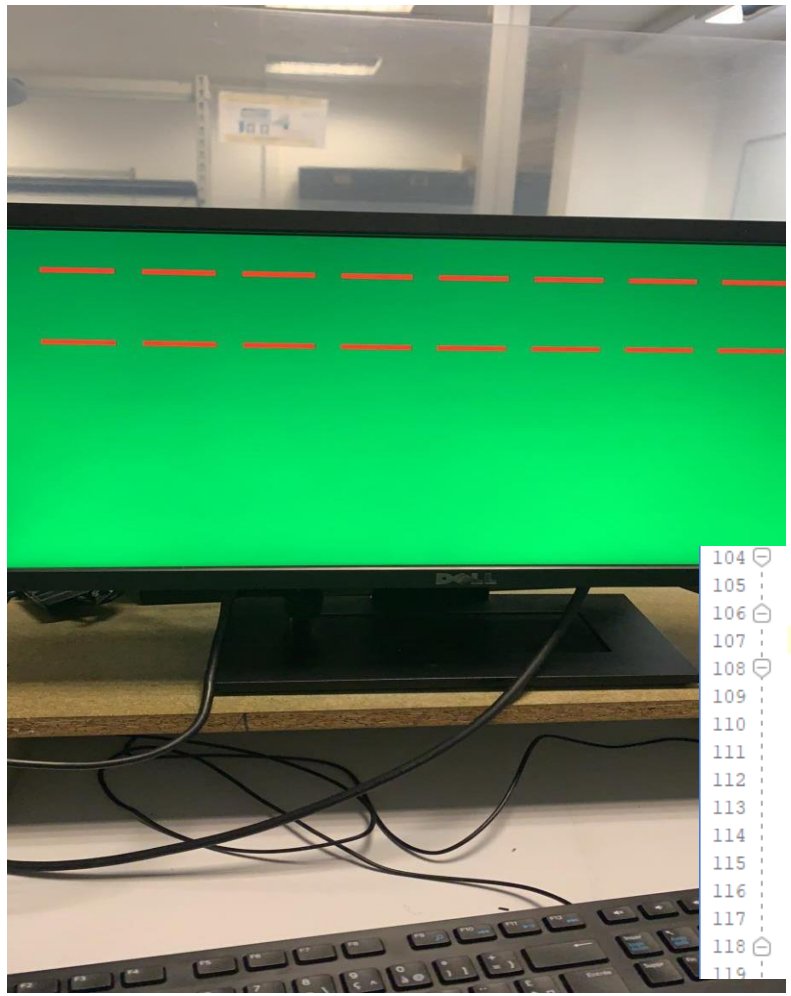


Code MAE

```
35  signal EP, EF: etat;
36  begin
37  process( Reset , Clk25)
38  begin
39  if reset ='0' then EP <= S0;
40   elsif rising_edge(Clk25) then EP <= EF;
41  end if;
42  end process;
43
44  process(EP, Endframe , Lost , No_Brick , Pause_Rqt , Fin_Tempo_Pause , Timer_Lost  )
45  begin
46  case(EP) is
47  when S0 => EF<=S0;if ((Pause_Rqt='1' ) and ( Timer_Lost ="000000") )then EF<=S1 ; end if;--pause--
48                   if((Pause_Rqt='0') and (Endframe='1') and (Timer_Lost>"000000")) then EF <= S6; end if ;
49
50  when S1=> EF<=S1;
51   if(Pause_Rqt='0' and Fin_Tempo_Pause='1') then EF <= S2; end if ;
52  when S2=>  EF<=S2; --Mode actif--
53  if (Pause_Rqt='1') then EF<=S3; end if ;
54  if (No_Brick='1')  then EF<=S4; end if;
55  if(Lost='1') then EF <= S5; end if;
56  when S3 => EF<=S3; if (Pause_Rqt='0' and Fin_tempo_Pause='1') then EF<=S0; end if ;
57  when S4=>EF<=S4;-- Win --
58  when S5 => EF<=S0 ;--Load timer lost --
59  when S6 => EF<=S0; --Game loste--
60
61  end case;
62  end process;
```

```
64  process(EP)
65  begin
66  case(EP) is
67  when S0 => RAZ_Tempo_Pause<='1';Pause <='1'; Brick_Win <='0'; Update_Tempo_Pause <='0'; Load_Timer_Lost <='0' ; Update_Timer_Lost  <='0';
68  when S1 =>RAZ_Tempo_Pause<='0';Pause <='0';  Brick_Win <='0'; Update_Tempo_Pause <='1'; Load_Timer_Lost <='0' ; Update_Timer_Lost  <='0';
69  when S2 => RAZ_Tempo_Pause<='1';Pause <='0';  Brick_Win <='0'; Update_Tempo_Pause <='0'; Load_Timer_Lost <='0' ; Update_Timer_Lost  <='0';
70  when S3 => RAZ_Tempo_Pause<='0';Pause <='0';  Brick_Win <='0'; Update_Tempo_Pause <='1'; Load_Timer_Lost <='0' ; Update_Timer_Lost  <='0';
71  when S4 =>RAZ_Tempo_Pause<='0';Pause <='0';  Brick_Win <='1'; Update_Tempo_Pause <='0'; Load_Timer_Lost <='0' ; Update_Timer_Lost  <='0';
72  when S5 => RAZ_Tempo_Pause<='0';Pause <='0';  Brick_Win <='0'; Update_Tempo_Pause <='0'; Load_Timer_Lost <='1' ; Update_Timer_Lost  <='0';
73  when S6 => RAZ_Tempo_Pause<='0';Pause <='0';  Brick_Win <='0'; Update_Tempo_Pause <='0'; Load_Timer_Lost <='0' ; Update_Timer_Lost  <='1';
74  end case;
75  end process;
76  end Behavioral;
```

- Le problème qu'on a rencontré après l'implémentation du code sur la carte nxys c'est que le système reste dans pause il ne change pas d'état .



```
-- CONTROLEUR DES JEUX CASSE BRIQUES ET PONG
-------------------------------------------------------------------------
-- REMPLACER CES 3 INSTRUCTIONS PAR L'INSTANCIATION DU MODULE MODE --

My: entity work.mode
port map (
     Pause_Rqt =>pause_rqt,
     Endframe =>endframe,
     Lost =>lost,
     No_Brick =>no_brick,
     Clk25=>clk25,
     Reset =>reset,
     Pause =>pause,
     Brick_Win =>brick_win ,
     Game_Lost =>lost_game);
```

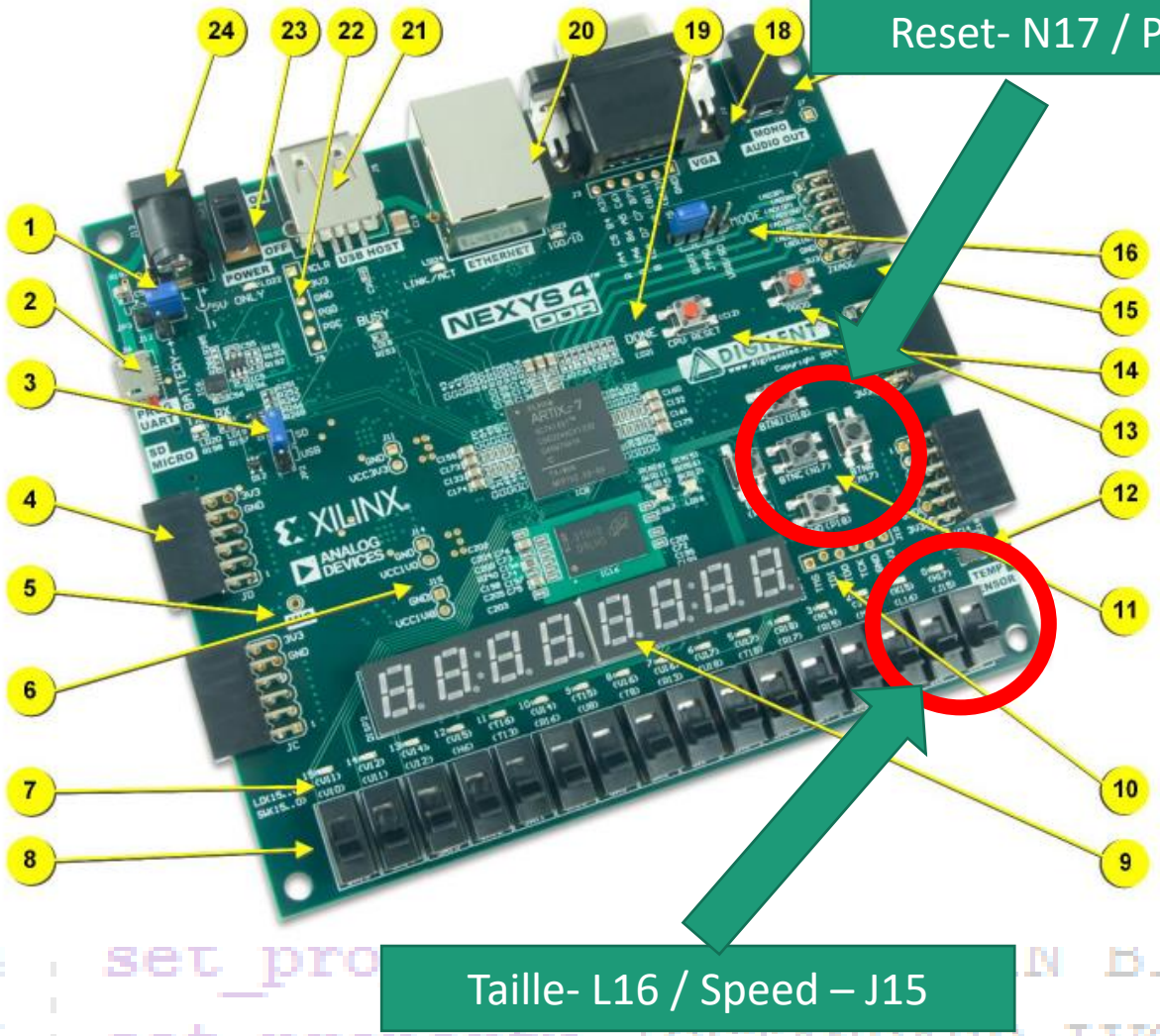
Instanciation mode dans le fichier game

```
 My: entity work.Timer_Lost
port map( Load_Timer_Lost => Load_Timer_Lost,
          Update_Timer_Lost =>Update_Timer_Lost,
          Timer_Lost => Timer_Lost,
          CLK25=>  CLK25,
          Reset=>Reset,
          Game_Lost =>Game_Lost );
   Myy : entity work. tempo_pause
   port map(  RAZ_Tempo_Pause => RAZ_Tempo_Pause,
           Update_Tempo_Pause=>Update_Tempo_Pause,
          CLK25=>CLK25,
             Reset =>Reset,
Fin_Tempo_Pause  => Fin_Tempo_Pause );
   Myyy : entity work .MAE
 port map (Clk25 =>Clk25,
 Reset =>Reset,
 Pause_Rqt=>Pause_Rqt,
 Endframe =>  Endframe,
 Lost =>Lost,
 No_Brick=>No_Brick,
 Fin_Tempo_Pause=>  Fin_Tempo_Pause,
 Timer_Lost=>Timer_Lost,
  RAZ_Tempo_Pause=>RAZ_Tempo_Pause,
  Update_Tempo_Pause=>Update_Tempo_Pause,
  Load_Timer_Lost=> Load_Timer_Lost,
  Update_Timer_Lost=>Update_Timer_Lost,
  Pause => Pause ,
  Brick_Win =>  Brick_Win);
```

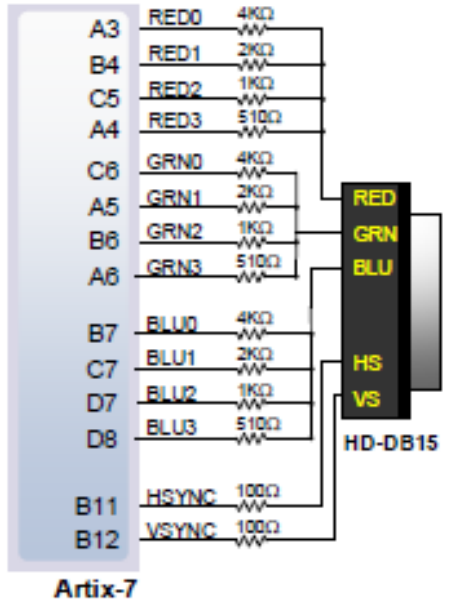
Instanciation MAE ,Time_lost,Time_ pause dans mode

```
-- ********** REMPLACER L'INSTRUCTION CI-DESSOUS PAR CELLE EN COMMENTAIRE **********
--pause_rqt            => S1,                -- Demande de Pause - Appui sur Bouton Encodeur
pause_rqt             => pause_rqt,          -- Demande de Pause - Appui sur Bouton Encodeur
-- ********************************************************************************
```

Les «contraints»



Images issues de la documentation du Nexys 4 DDR



Pour le VGA

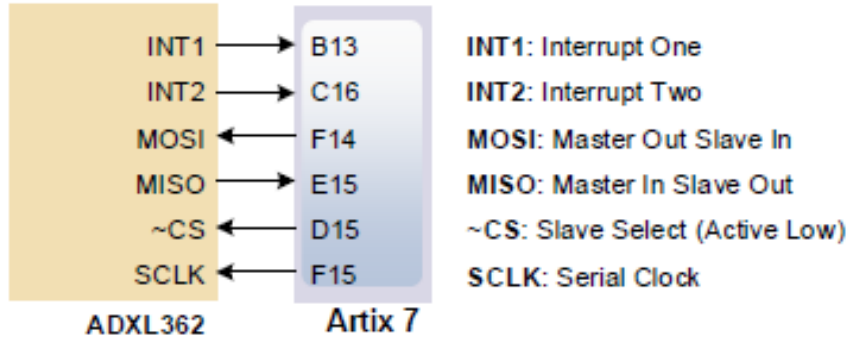
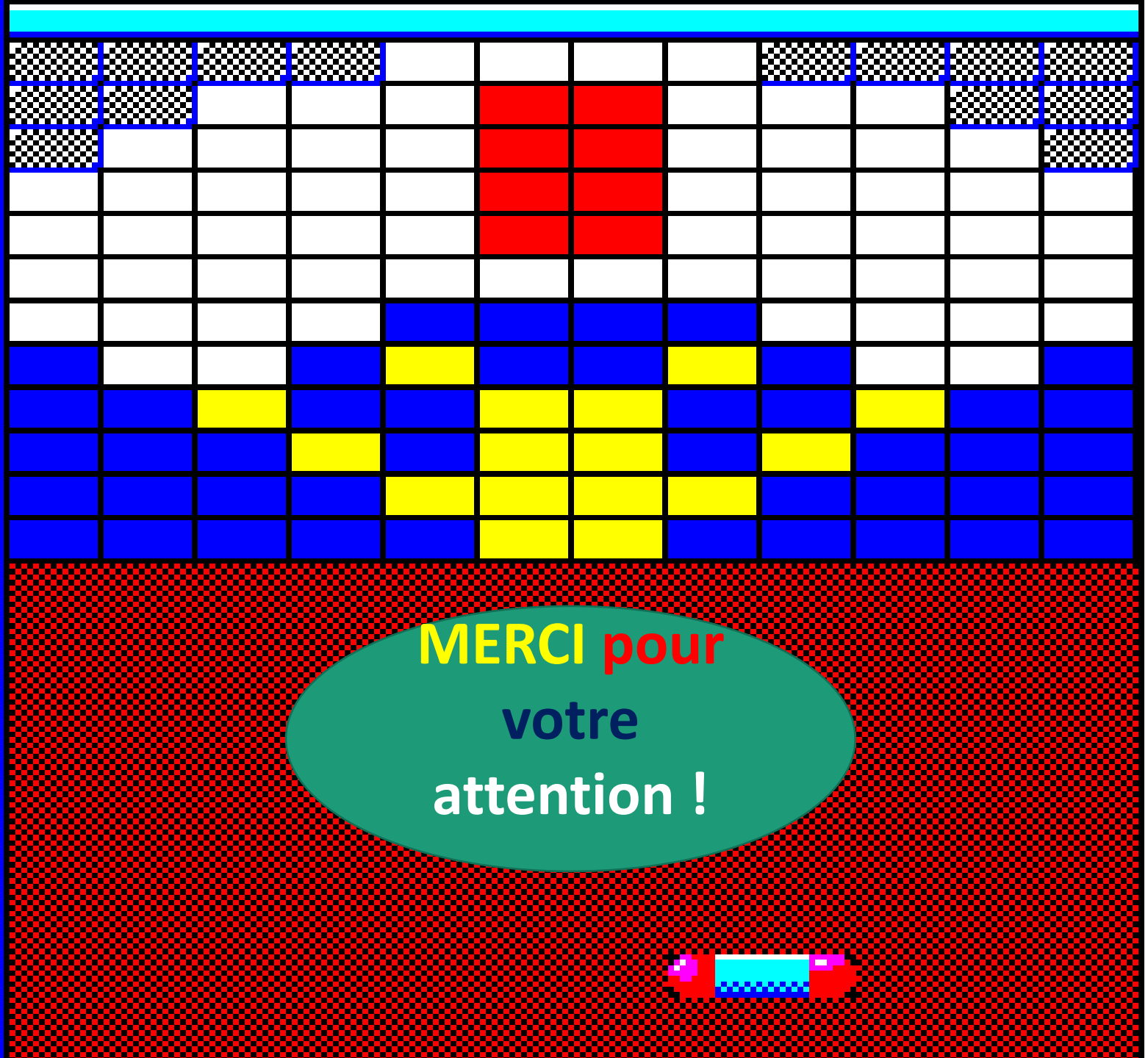


Figure 23. Accelerometer interface.

Pour l'accéléromètre

Conclusion

C’est avec regret que nous n’avons pas pu mener à termes la tache 3 et les améliorations. Néanmoins nous avons appris beaucoup de choses dans la tache 1 , 2 et le reste du jeu fonctionne bien. C’était une réelle satisfaction de pouvoir mener un bien un projet d’électronique numérique et système embarqué.



GIGASNOID

HIGH-SCORE 18590

SCORE 90120

ROUND 11

BARS 3

PARTS

Ok 1 5 1 3 2