# Mini Projet 2E200


**Par:**

- **Imad Eddine MAROUF**
- **Zyber DEDJA**

**Responsable de l'UE 2E200:**
**Bertrand GRANADO**

# Sommaire:

# Introduction:

Dans le cadre de l'UE "Electronique Numérique , Combinatoire et séquentielle 2E200" , nous avons été amené à réaliser un jeux Casse-Brique.

L'objectif de ce mini-projet étant de:

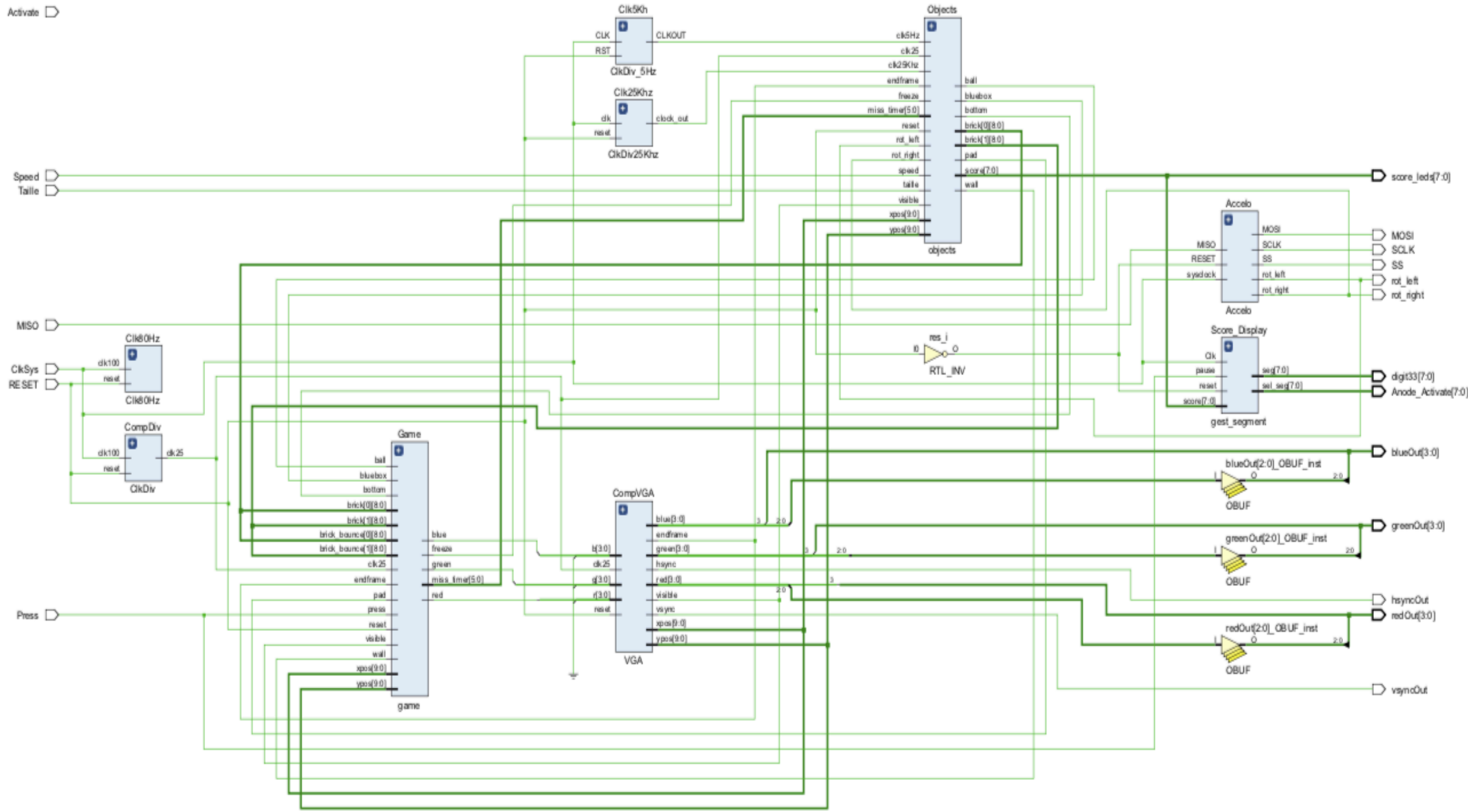
> Mettre en pratique nos connaissances théoriques acquises en cours.

> Améliorer nos compétences de langage VHDL.

> Se familiariser avec le logiciel VIVADO.

> Se confronter à réalité du terrain (debugging).

# Fonctionnement du jeux:



# 2 – VGA:

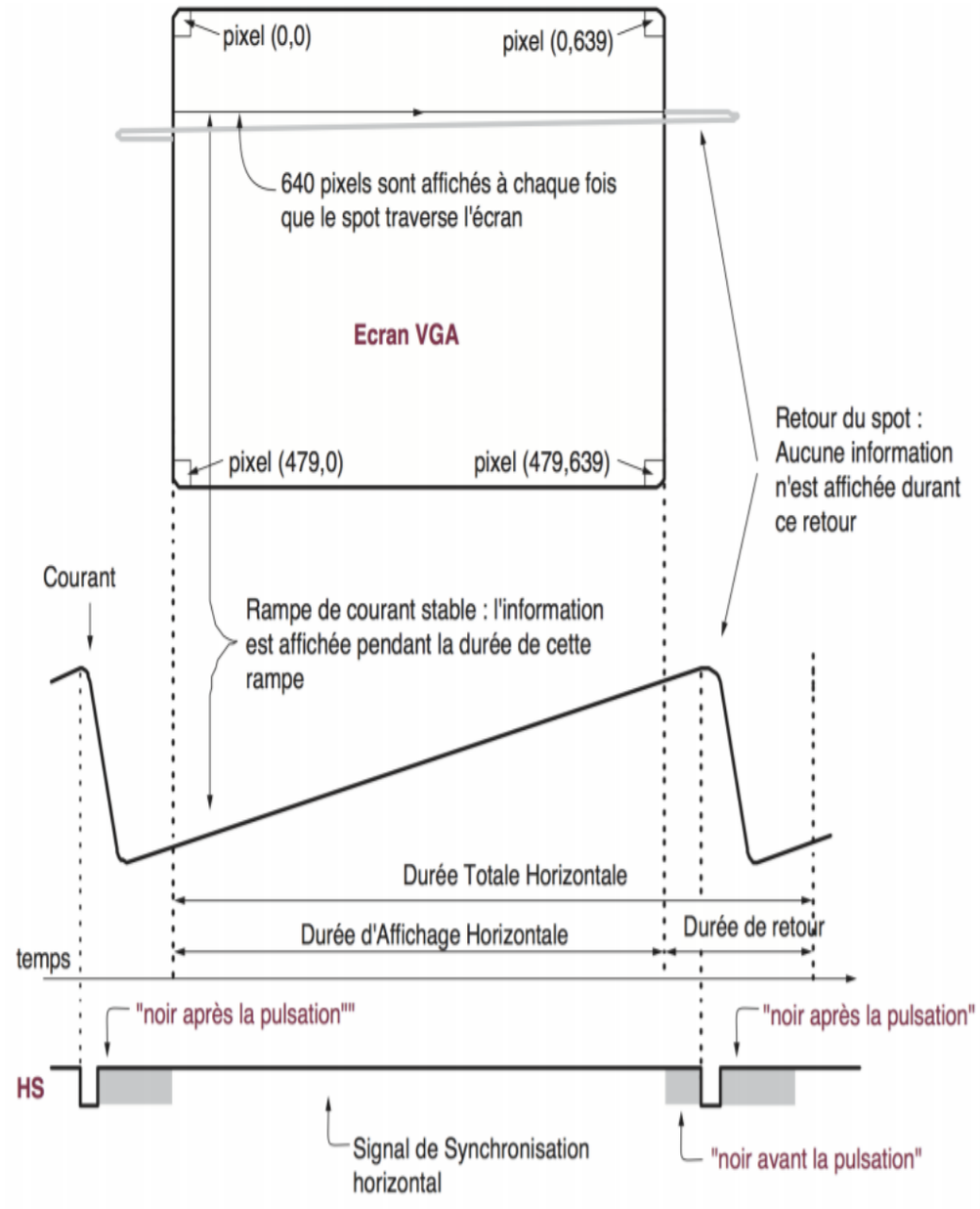
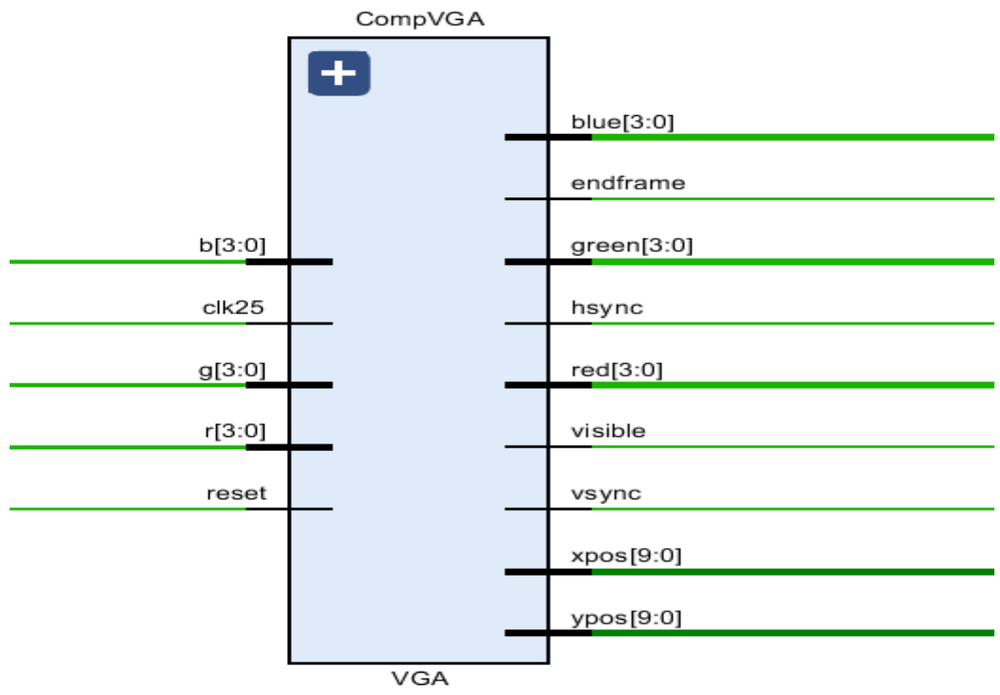


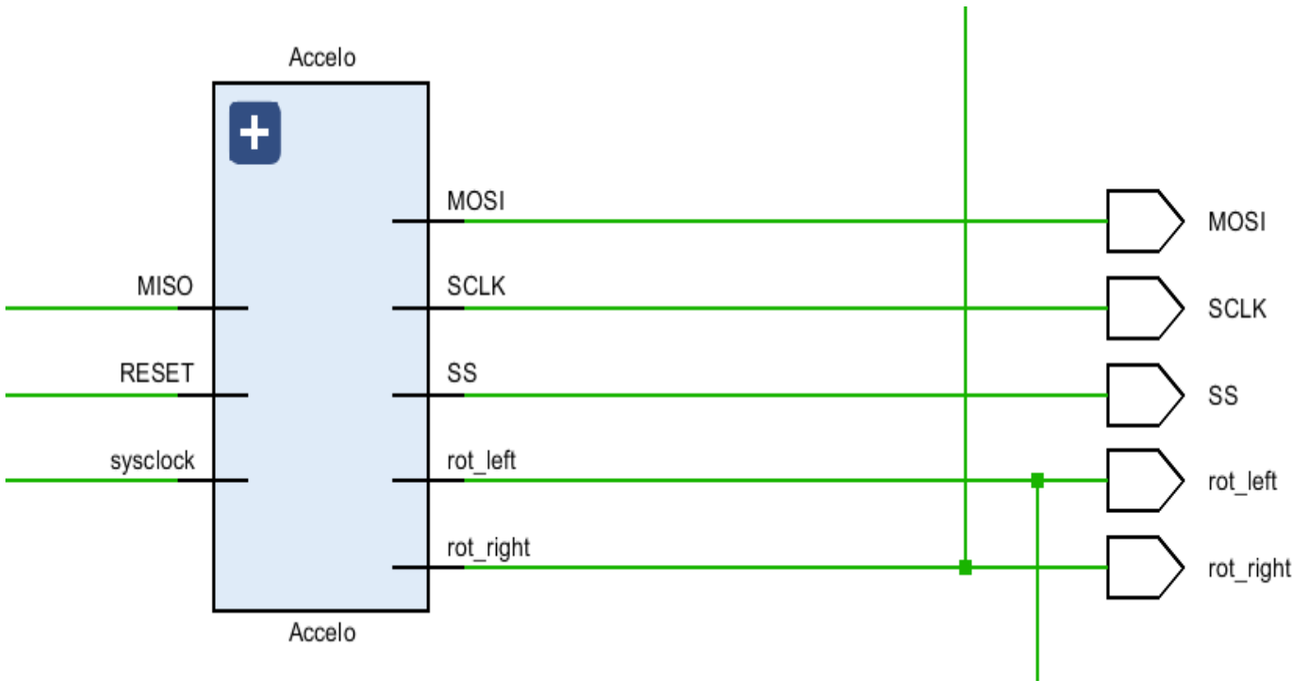

Figure 1: Principe de l'affichage VGA

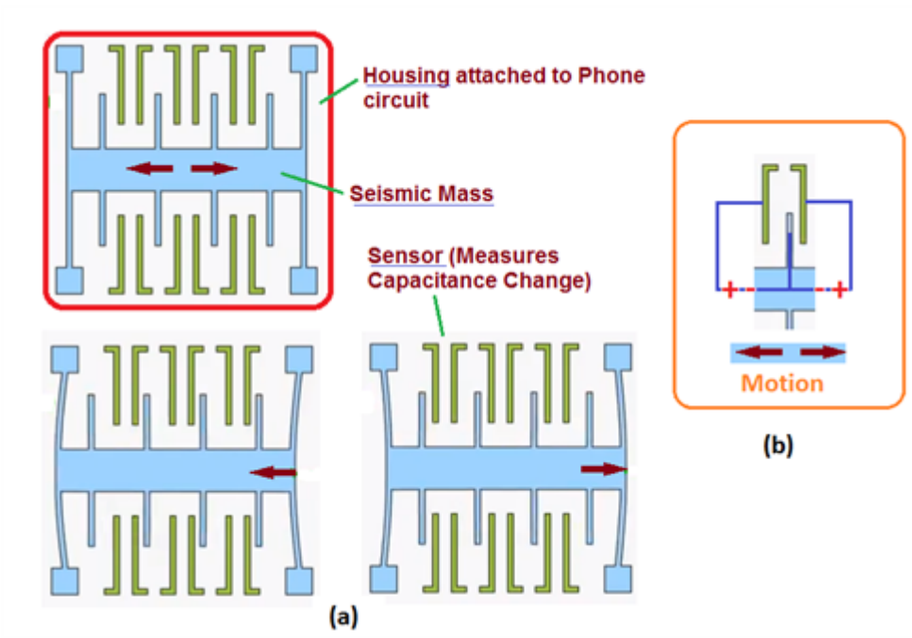



-

```
--------------Calling the VGA_Component----------------------
CompVGA: entity work.vga(archi)
port map( clk25 => clk_25, reset => RESET,
          r => red, g => green, b => blue,
          red => redOut, blue => blueOut, green => greenOut,
          hsync => h_sync, vsync => v_sync,
          visible => visible, endframe => endframe, xpos => xpos, ypos => ypos);
-------------------------------------------------------------
```

# 3 – Accelo.vhd



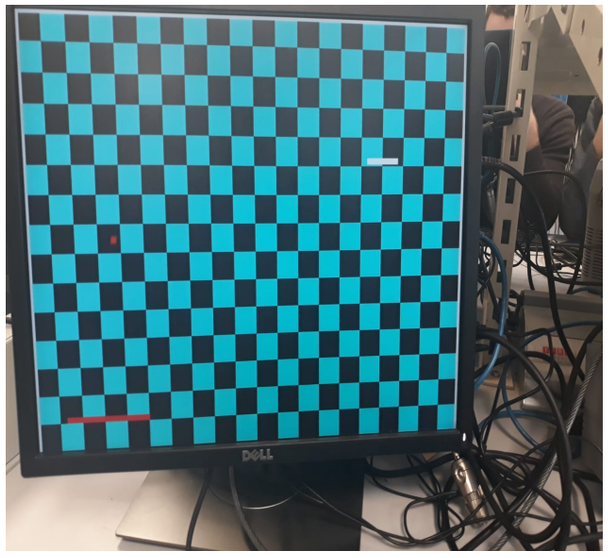




```
------------------Calling  Accelo Component-----------------
Accelo: entity work.Accelo(Behavioral)
Port Map( sysclock => ClkSys, RESET => res, LED_Y => accel_y, SCLK => SCLK,
        MOSI => MOSI, MISO => MISO, SS => SS, rot_left => left, rot_right => right);
```

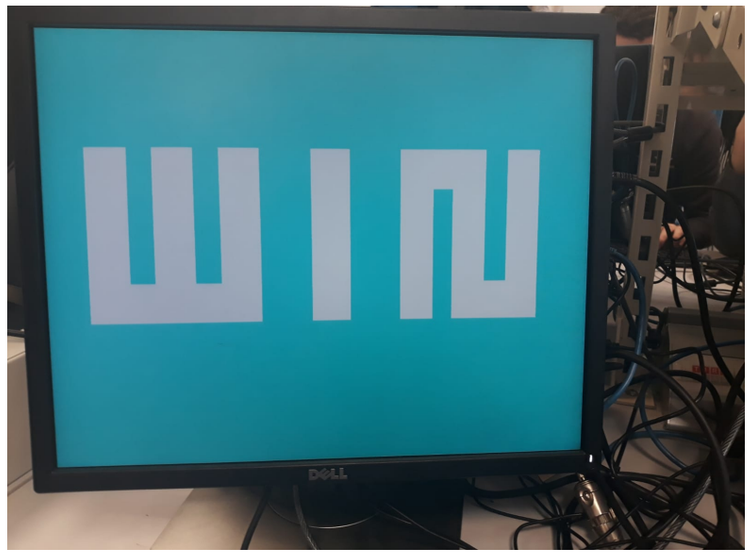
# LISTE DES AMELIORATIONS:

- Changement de décor (couleurs).
- Ecrire un message (WIN-LOSE).
- Affichage du score dans les afficheurs 7 segments.
- Affichage du nombre de vie.
- Création d’un second niveau (augmentation de la vitesse).

## 1 – Décor:



## 2 – Ecrire un message:



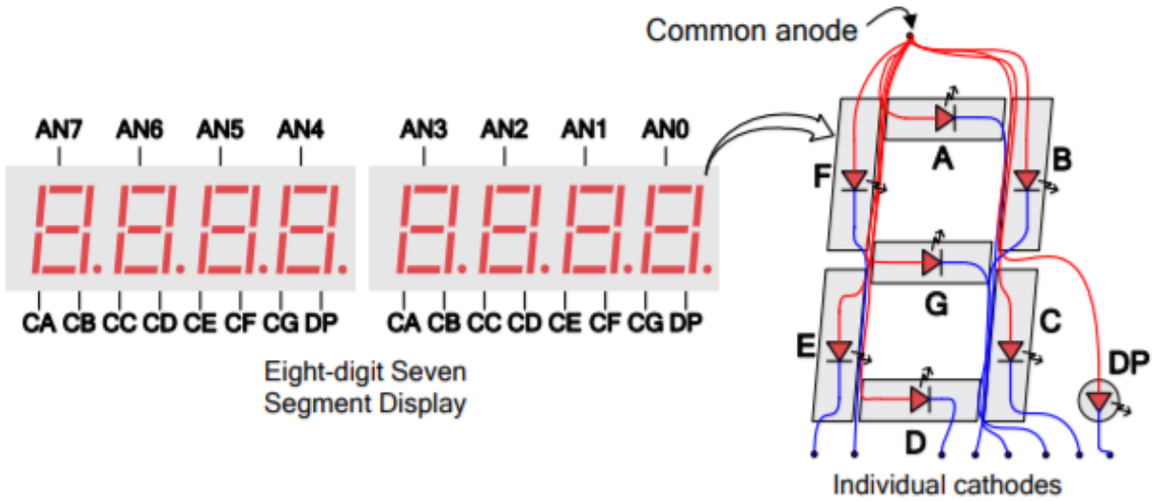


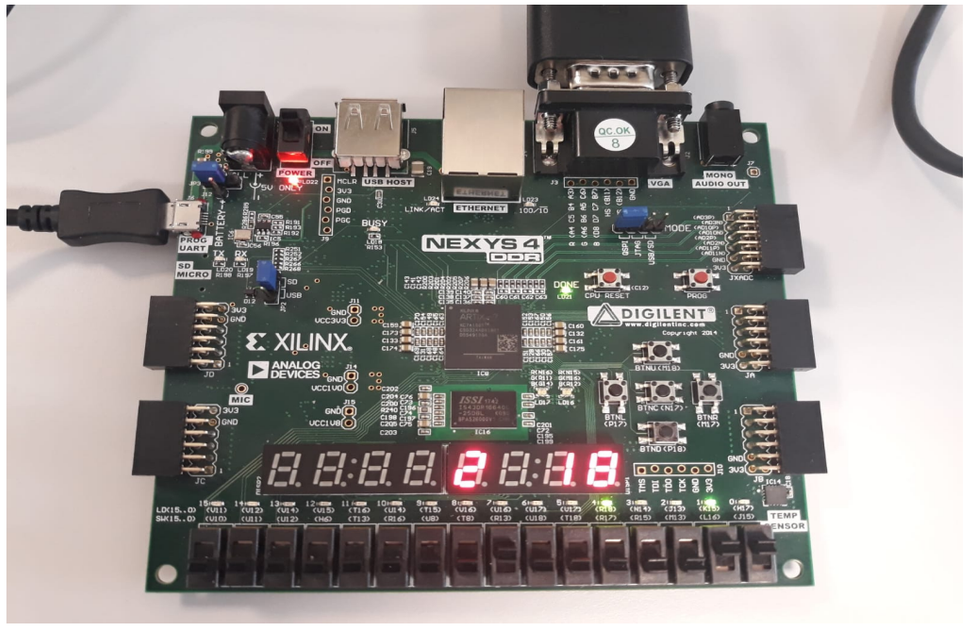
```
if win = '1' then
        red <= '0'; green <= '1'; blue <= '0';
  if( (xpos>47 and xpos<95 and ypos>123 and ypos<317) or (xpos>122 and xpos<169 and ypos>123 and ypos<317) or (xpos>197 and xpos<244 and ypos>123 and yp
   or (xpos>94 and xpos<123 and ypos>272 and ypos<317) or (xpos>168 and xpos<198 and ypos>272 and ypos<317) or (xpos>302 and xpos<349 and ypos>123 and y
   or (xpos>408 and xpos<449 and ypos>123 and ypos<317) or (xpos>471 and xpos<512 and ypos>123 and ypos<317) or (xpos>537 and xpos<577 and ypos>123 and
   or (xpos>448 and xpos<472 and ypos>123 and ypos<168) or (xpos>511 and xpos<538 and ypos>272 and ypos<317)
  )then
          red <= '1'; green <= '1'; blue <= '1';
  else
          red <= '0'; green <= '1'; blue <= '0';
  end if;

  elsif miss='1' then
        red <= '1'; green <= '0'; blue <= '0';
  if( (xpos>172 and xpos<220 and ypos>47 and ypos<214) or (xpos>219 and xpos<289 and ypos>173 and ypos<214) or (xpos>314 and xpos<346 and ypos>47 and yp
   or (xpos>396 and xpos<428 and ypos>47 and ypos<214) or (xpos>345 and xpos<397 and ypos>47 and ypos<92) or (xpos>345 and xpos<397 and ypos>169 and ypo
   or (xpos>172 and xpos<289 and ypos>249 and ypos<290) or (xpos>172 and xpos<289 and ypos>314 and ypos<351) or (xpos>172 and xpos<289 and ypos>375 and
   or (xpos>172 and xpos<211 and ypos>288 and ypos<315) or (xpos>248 and xpos<289 and ypos>350 and ypos<376) or (xpos>314 and xpos<431 and ypos>249 and
   or (xpos>314 and xpos<346 and ypos>249 and ypos<376) or (xpos>314 and xpos<431 and ypos>314 and ypos<351) or (xpos>314 and xpos<431 and ypos>375 and
    )then
        red <= '1'; green <= '1'; blue <= '1';
    else
        red <= '1'; green <= '1'; blue <= '0';
    end if;

  end if;
```

# 3 – Affichage de score et nombres de vie:




```
--Convert from Integer to Unsigned using Std_Logic_Arith
remainder <= score_int MOD 10;
ones_vector <= conv_std_logic_vector(remainder, ones_vector'length);

decs <=  score_int / 10 ;
decs_vector <= conv_std_logic_vector(decs, decs_vector'length);
```

```
case (counter) is
        when 00000 =>
               score_inter <= ones_vector;
               anode_act <= not "00000001";
        when 10000 =>
               score_inter <= decs_vector;
               anode_act <= not "00000010";
        when 20000 =>
               score_inter <= lives_int;
               anode_act <= not "00001000";
        when 30000 =>
               score_inter <= ones_vector;
               anode_act <= not "00000001";
        when 40000 =>
               score_inter <= decs_vector;
               anode_act <= not "00000010";
        when 50000 =>
               score_inter <= lives_int;
               anode_act <= not "00001000";
        when 60000 =>
               score_inter <= ones_vector;
               anode_act <= not "00000001";
        when 70000 =>
               score_inter <= decs_vector;
               anode_act <= not "00000010";
        when 80000 =>
               score_inter <= lives_int;
               anode_act <= not "00001000";
        when others => NULL;
```

```
case (score_inter) is
    when "00000000" =>
         seg <= "11000000";
    when "00000001" =>
         seg <= "11111001";
    when "00000010" =>
         seg <= "10100100";
    when "00000011" =>
         seg <=  "10110000";
    when "00000100" =>
         seg <=  "10011001";
    when "00000101" =>
         seg <=  "10010010";
    when "00000110" =>
         seg <=  "10000010";
    when "00000111" =>
         seg <=  "11111000";
    when "00001000" =>
         seg <=  "10000000";
    when "00001001" =>
        seg <= "10010000";
    when others =>
       seg <= "10010000";
end case;
```

# 4 – Création des niveaux:

```
 ------------------- Passer à un niveau sup -------------------
niveau <= '1' & ('0' or int_speed);
process(winout)
 begin
    if (winout = '1')then
         niveau <= unsigned(niveau) + 1;
    end if;
end process;

int_speed <= niveau(0);
int_taille <= niveau(1);
-----------------------------------------------------------------
```

# Intelligence Artificielle pour le jeux Case Brique:

Stratégie Essayé: 1 – Implémentation de CaseBrique sur le PC.
2 – Implémentation de l'algorithme Q-Learning.
3 – Transmètre les paramètres des réseaux de neurons sur la carte (inputs/weights, activation function).


Architecture:
1 - Actions: Gauche, Droite, Rien.
2 - Récompense: le score.
3 - Règles terminal: avec un nombre de vies, le jeux termines si la balle tombe, où le temps termine.
4 - Règles de transitions: la fonction publié dans la feuille de Mnih & al.

| | B. Rider | Breakout | Enduro | Pong | Q*bert | Seaquest | S. Invaders |
|---|---|---|---|---|---|---|---|
| **Random** | 354 | 1.2 | 0 | −20.4 | 157 | 110 | 179 |
| **Sarsa** [3] | 996 | 5.2 | 129 | −19 | 614 | 665 | 271 |
| **Contingency** [4] | 1743 | 6 | 159 | −17 | 960 | 723 | 268 |
| **DQN** | **4092** | **168** | **470** | **20** | **1952** | **1705** | **581** |
| **Human** | 7456 | 31 | 368 | −3 | 18900 | 28010 | 3690 |
| **HNeat Best** [8] | 3616 | 52 | 106 | 19 | 1800 | 920 | **1720** |
| **HNeat Pixel** [8] | 1332 | 4 | 91 | −16 | 1325 | 800 | 1145 |
| **DQN Best** | **5184** | **225** | **661** | **21** | **4500** | **1740** | 1075 |

Table 1: The upper table compares average total reward for various learning methods by running an $\epsilon$-greedy policy with $\epsilon = 0.05$ for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an $\epsilon$-greedy policy with $\epsilon = 0.05$.

http://www.nintendoninja.com/

(2) https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

# Conclusion:

Ce projet nous a permis de :

- Appliquer les notions acquises dans cette UE.
- Travailler en groupe, meilleur communication.

Il a aussi stimulé notre créativité ; En effet on a implémenté plusieurs améliorations, néanmoins plusieurs autres idées restent inachevées tel que :

- L'implémentation d'IA (Reinforcement Learning) pour le jeu
- Travailler sur l'interface graphique et la rectification de bug potentiels.

# Références:

1 – FPGA BreakOut par Doug Cumbie, UCF Fall 2008.
2 – Playing Atari with Deep Reinforcement Learning par DeepMinds Technologies.
3 – Design and Implementation of Neural Network in FPGA Sep. 2012 ISSN 1813- 7822
4 – Neural Network Based Reinforcement Learning Acceleration on FPGA Platforms , par Jiang Su et al.
5 – Stanford CS229 Final Report Reinforcement Learning to Play Mario, par Yizheng Liao, et al.
6 – Implementing Neural Networks on Field Programmable Gare Array par Hannes Kinks.
7 – FPGA Accelerator Architecture for Q-Learning and its applications in Space Exploration par Pranay Reddy Gankidi.
8 – A General Neural Network Hardware Architecture on FPGA.
9 – Asynchronous Methods for Deep Reinforcement Learning par Google DeepMind.
10 – A hardware-Based FPGA AI for Super Mario Bros, ECE5760, Cornell University.

Merci pour votre attention ( ^_^ )