



FACULTÉ DES SCIENCES APPLIQUÉES

ELEN0040-1 : DIGITAL ELECTRONICS

---

## Projet VHDL : Tic-Tac-Toe

---

*Professeur et assistant de projet*

REDOUTÉ Jean-Michel  
PEERS Thibaud

*Groupe :*

DELPORTE Guillaume  
FIRRINCIELI Maxime  
LAMBERMONT Romain  
LOUIS Arthur

15 mai 2021

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Signaux</b>	<b>2</b>
<b>3</b>	<b>Schéma fonctionnel de l'implémentation</b>	<b>3</b>
<b>4</b>	<b>Pseudo-code de l'implémentation</b>	<b>4</b>
<b>5</b>	<b>Description du code</b>	<b>7</b>
<b>6</b>	<b>Description des composants</b>	<b>8</b>
<b>7</b>	<b>Problèmes rencontrés et solutions</b>	<b>9</b>

## Table des figures

<b>1</b>	<b>Schéma d'un jeu de morpion</b>	<b>1</b>
<b>2</b>	<b>Exemple de partie de morpion</b>	<b>1</b>
<b>3</b>	<b>Schéma fonctionnel de l'implémentation</b>	<b>3</b>
<b>4</b>	<b>Schéma d'utilisation du compteur décodeur CD4026</b>	<b>8</b>

# 1 Introduction

Dans le cadre de ce projet pour le cours d'électronique numérique, nous avons décidé de réaliser un jeu assez simple, le morpion (tic-tac-toe en anglais) et de l'implémenter en VHDL sur notre FGPA. Le but du morpion est d'aligner trois de ses jetons sur une même ligne (verticale, horizontale ou en diagonal) tout en empêchant l'adversaire de faire de même.

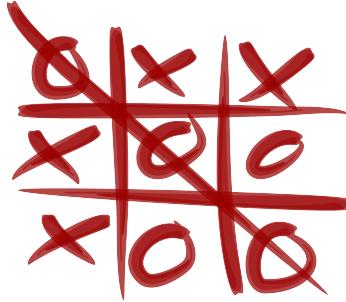


FIGURE 1 – Schéma d'un jeu de morpion

Dans ce rapport, nous allons détailler notre implémentation selon plusieurs aspects (I/O, pseudo-code, schéma fonctionnel,...)

Pour jouer, nous disposons simplement de 4 boutons pour se déplacer, jouer ou reset la partie. Tout se passe directement sur notre matrice de LED et les scores de la partie en cours sont affichés sur des afficheurs 7-segments. La matrice de LED montre les endroits où les pions ont déjà été joués et leur couleur correspondante (rouge ou vert). Un curseur orange (couleur créée par le mélange de rouge et vert) indique où le curseur du joueur est actuellement situé pour indiquer où il compte jouer son coup.

Dans l'image ci-dessous, on assiste à une partie que le joueur vert va gagner, changeant les scores à 1-2 :

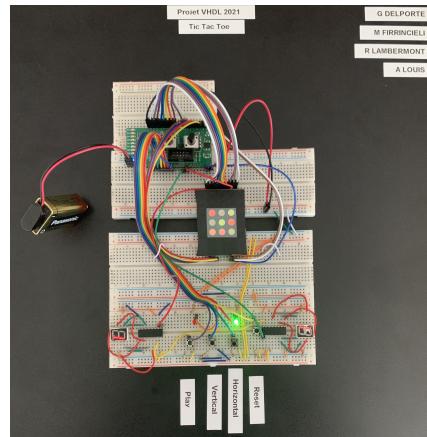


FIGURE 2 – Exemple de partie de morpion

## 2 Signaux

Dans cette section, nous allons décrire les différents types de signaux utilisés.

### 1. Inputs :

- clk : clock de notre système qui varie entre 50 et 266 Hz
- rst : input reçu lors du déclenchement du bouton reset, sert à remettre le jeu et les scores à 0
- play : input reçu lors du déclenchement du bouton play, sert à jouer le coup désiré si possible
- hor : input reçu lors du déclenchement du bouton de mouvement horizontal, sert à se déplacer dans la matrice de LED de gauche à droite
- ver : input reçu lors du déclenchement du bouton de mouvement vertical, sert à se déplacer dans la matrice de LED de haut en bas

### 2. Outputs :

- colr : output envoyé pour qu'une colonne spécifique de la matrice de LED s'allume en rouge (combiné avec l'output colg, la colonne correspondante devient alors orange)
- colg : output envoyé pour qu'une colonne spécifique de la matrice de LED s'allume en vert (combiné avec l'output colr, la colonne correspondante devient alors orange)
- row : output servant à allumer une ligne spécifique dans la matrice de LED
- green\_led : output envoyant un signal vers la LED afin que celle-ci s'allume quand c'est au tour du joueur vert
- red\_led : output envoyant un signal vers la LED afin que celle-ci s'allume quand c'est au tour du joueur rouge
- compteur1 : output envoyant un signal au compteur-décodeur qui gère le 7-segment display lorsque le joueur rouge gagne la partie
- compteur2 : output envoyant un signal au compteur-décodeur qui gère le 7-segment display lorsque le joueur vert gagne la partie

### 3. Signaux :

- player : signal déterminant quel joueur est en train de placer son pion
- playerCol : signal déterminant la colonne où se trouve actuellement le curseur du joueur
- playerRow : signal déterminant la rangée où se trouve actuellement le curseur du joueur

## 3 Schéma fonctionnel de l'implémentation

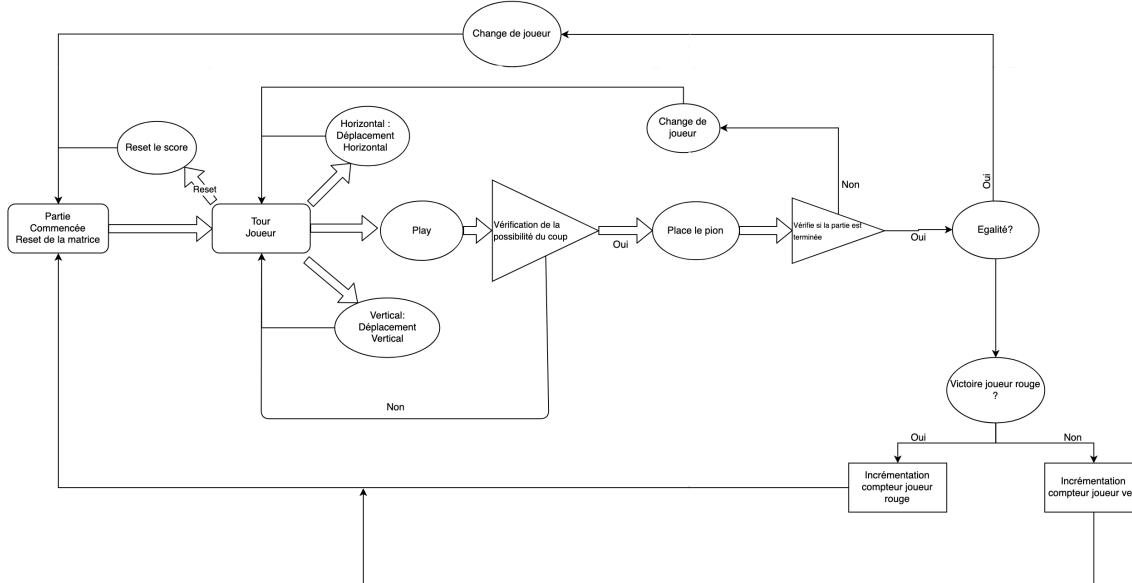


FIGURE 3 – Schéma fonctionnel de l'implémentation

## 4 Pseudo-code de l'implémentation

```
-- process pour faire defiler les rangees de la matrice de LED
liste de sensibilite : clk

while clk = 0 and clk'last_value = 1      -- faire une rotation tous les 3
    if clk_count = 2                      -- fronts montants d'horloge de la
        clk_count = 0                      -- rangee de la matrice de LED a
        if row_count = 5                   -- allumer
            row_count = 3
        else
            row_count = row_count + 1
        end if
    else
        clk_count = clk_count + 1
    end if
end while

-- process pour allumer une rangee bien precise de la matrice de LED
liste de sensibilite : row_count

row = off toutes les lignes
row(row_count) = on
switch (row_count) :
    case 3 :
        colg = colg3
        colr = colr3
        if playerCol = row_count
            colg(playerRow) = on -- affiche le curseur en orange
            colr(playerRow) = on
        end if
    end case
    case 4 :
        colg = colg4
        colr = colr4
        if playerCol = row_count
            colg(playerRow) = on -- affiche le curseur en orange
            colr(playerRow) = on
        end if
    end case
    case 5 :
        colg = colg5
        colr = colr5
        if playerCol = row_count
            colg(playerRow) = on -- affiche le curseur en orange
```

```

        colr(playerRow) = on
    end if
end case
end switch

-- process pour reagir a l'utilisation d'un bouton en cours)
-- winner: integer (verifier si qqn gagne)
-- countTurn: integer (compter le nombre de jetons places.)
liste de sensibilite : clk

if clk = 1 and clk'last_value = 0
    if winner = 1
        countTurn = 9 --remettre le plateau de jeu a zero
        impulsion dans compteur1
    else if winner = 2
        countTurn = 9 --remettre le plateau de jeu a zero
        impulsion dans compteur2
    end if
    if player1 a gagne
        winner = 1
    else if player 2 a gagne
        winner = 2
    else
        winner = 0
    end if

if click sur le bouton play
    switch (playerCol)
        case 3 :
            if position pas encore jouee
                if player = 2
                    colg3(playerRow) = on
                    turn = false
                else if player = 1
                    colg3(playerRow) = on
                    turn = false
                end if
            end if
        end case
        case 4 :
            if position pas encore jouee
                if player = 2
                    colg4(playerRow) = on
                    turn = false
                else if player = 1

```

```

                colg4(playerRow) = on
                turn = false
            end if
        end if
    end case
    case 5 :
        if position pas encore jouee
            if player = 2
                colg5(playerRow) = on
                turn = false
            else if player = 1
                colg5(playerRow) = on
                turn = false
            end if
        end if
    end case
end switch
if turn = false
    countTurn = countTurn + 1;
    if player = 1
        player = 2
    else if player = 2
        player = 1
    end if
end if
end if

if click sur le bouton rst or countTurn = 9
    colr3 = off
    colr4 = off
    colr5 = off
    colg3 = off
    colg4 = off
    colg5 = off
    playerRow = 3
    playerCol = 4
    countTurn = 0

else if click sur le bouton hor
    if playerCol = 5
        playerCol = 3
    else
        playerCol = playerCol + 1;
    end if

```

```

else if click sur le bouton ver
    if playerRow = 4
        playerRow = 2
    else
        playerRow = playerRow + 1;
    end if
end if

-- process pour allumer la LED correspondante au joueur qui joue

if player = 1
    green_led = off
    red_led = on
else if player = 2
    green_led = on
    red_led = off
end if

```

## 5 Description du code

Nous avons créer différents process pour implémenter notre jeu. Nous allons les décrire dans cette section.

- process scanning : ce process tient à jour un index qui se rafraîchit tout les 3 fronts montants d'horloge. Cet index nous permet d'allumer seulement une ligne à la fois sur la matrice de LED. Il est sensible au signal clk.
- process row\_on : ce process allume la ligne qui porte l'index défini par le process précédent. Il gère aussi l'affichage du curseur pendant la partie.
- process button\_and\_turn : ce process gère un tour complet. Il réagit à l'appui des boutons pour déplacer le curseur et gère aussi le bouton reset et le bouton play. Il est sensible au signal row\_count. De plus, ce process vérifie si la partie est gagnée ou si l'écran est complet et met à jour les scores en fonction.
- process led\_player : ce process gère les deux LEDs qui indiquent qui doit jouer. Il est sensible au signal player.

## 6 Description des composants

Pour le hardware, voici la liste de tous les composants qui ont été nécessaires à la réalisation de notre projet :

- Matrice de LED bicolore (5 par 7) : Nous avons utilisé cette matrice pour afficher notre morpion grâce à un système de balayage se reposant sur la clock interne de la carte. En effet, en rafraîchissant une rangée de la matrice tous les 3 tics d'horloge, il nous est possible d'allumer chaque LED dans la couleur souhaitée grâce à des signaux rangés dans des vecteurs (*colr*, *colg* et *row*).
- Breadboards : Ces planches d'essai nous permettent de rendre un projet assez propre et de rendre plus claires les connexions sans avoir à souder.
- Compteur-décodeur (CD4026) : Nous avons utilisé un composant permettant d'incrémenter et d'afficher directement un nombre sur un afficheur 7-segments en utilisant bien ses pin (voir schéma ci-dessous).
- Afficheur 7-segments (cathode commune) : Ceux-ci sont reliés à leurs compteurs-décodeurs afin d'afficher le score de chaque joueur, Ces afficheurs ne peuvent afficher que des chiffres et le score peut donc varier de 0 à 9.
- Résistances : Nous avons utilisé des résistances afin de ne pas abîmer les LED et les afficheurs 7-segments mais également en tant que résistance *pull-down* pour empêcher les rebonds dans les boutons.
- Boutons : Utilisés afin de déclencher quatre évènements lors d'une partie de morpion (déplacement vertical ou horizontal, jouer son coup ou reset la partie).
- LED : Une de couleur verte et une de couleur rouge, associés à une résistance et connectée directement à une pin de la carte pour indiquer quel joueur joue.

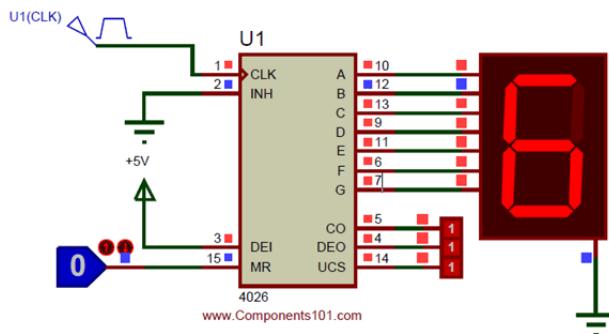


FIGURE 4 – Schéma d'utilisation du compteur décodeur CD4026

## 7 Problèmes rencontrés et solutions

Durant la réalisation de ce projet, nous nous sommes heurtés à de nombreux problèmes. Dans cette section nous allons en donner une liste non-exhaustive ainsi que les méthodes que nous avons mises en place afin de palier à ceux-ci.

- En tout premier lieu, nous avons eu du mal à trouver comment modifier un signal dans deux processus différents, donc nous avons globalisé le plus possible le rôle de chaque processus afin de ne pas modifier le même signal dans deux processus différents.
  - Ensuite, nous avons eu beaucoup de mal à déboucler nos boutons, nous avons premièrement tenter une méthode hardware avec un condensateur afin de lisser le signal, mais cela s'est avéré plus compliqué que prévu nous avons alors décidé d'utiliser une résistance pull-down associée avec du software afin de neutraliser ces fameux bonds assez ennuyants.
  - Un autre gros soucis a été de découvrir comment utiliser les afficheurs 7-segments sans gâcher toutes les pins de notre carte. Après un peu de recherche, nous avons trouvé un compteur-décodeur qui lorsqu'on lui envoie un signal, s'occupe d'incrémenter le nombre affiché sur le 7-segments. A l'aide de la datasheet, nous avons relié correctement les pins correspondantes et le tour était joué.
  - Nous avons aussi cassé un des afficheurs ainsi que grillé un des segments sur un autre car nous n'avions pas utilisé de résistances pour protéger les LEDs .
-