GRANDE ÉCOLE D'INGÉNIEURS

Année 2024-2025

Rapport de projet :

Morpion VGA en VHDL

Réalisé par :

VADOT Antoine

ALZAIX Florian

Encadré par :

PONCET Xavier

Sommaire :

Table des matières

GRANDE ÉCOLE D'INGÉNIEURS

I – Cahier des charges

L'objectif initial était de créer un jeu de morpion sur un écran avec la technologie VGA en se basant sur la carte Basys3. Le tout coordonné par un FPGA décrit par le langage VHDl.

Contenu attendu du projet :

- Affichage d'une grille de morpion sur l'écran VGA de tailles paramétrables.
- Affichage de croix ou de ronds en fonction du joueur qui joue.
- Réinitialisation de l'écran et des cases de la grille en cas de bug ou fin de la partie (bouton up de la croix de boutons).
- Pour la partie jeu :
  - Détection de fin de partie due à la victoire d'un joueur ou à l'égalité.
  - Le joueur qui joue doit pouvoir choisir l'emplacement de son symbole (croix ou rond) à l'aide des switches de la carte (de 1 à 9).
  - Soumission de la case choisie en pressant le bouton central de la carte (bouton au centre de la croix de boutons).
  - Une fois qu'un joueur a joué ; changement de joueur et donc de symbole.
  - Si un joueur tente de jouer sur une case déjà occupée, alors la soumission ne marche pas et le joueur doit continuer de jouer.
  - Si un joueur tente de cocher plus d'une case à la fois la soumission doit être rejetée de même et il continue de jouer.
  - S'il ne coche pas de case, alors sa soumission est aussi rejetée.

Exemple typique du déroulement attendu :

- Le joueur n°1 (J1) joue.
  - Il sélectionne sa case avec les switches.
  - Presse le bouton central.
  - Une croix s'affiche dans la case choisie de l'écran.
- C'est donc au tour du joueur n°2 (J2)
  - De la même manière que le J1, il sélectionne une case.
  - Si elle est déjà occupée, alors il doit en choisir une autre.
  - S'il en sélectionne plus ou moins qu'une case, alors il doit en choisir une autre.
  - Pour choisir une autre case, c'est le processus normal : switches puis bouton central.
  - Un rond s'affiche dans la case sélectionnée.
- Une fois toutes les cases remplies ou trois symboles alignés alors l'ajout de symboles est bloqué et la seule option possible est le reset (bouton haut de la croix de boutons).
- Une fois le bouton pressé, alors la partie recommence.

II – Démarche suivie

Comme première initiative, nous avons découpé le travail en deux blocs majeurs :

- Ecran VGA
- Machine à état

Puis nous avons procédé de la manière suivante : documentation, test puis correction et ainsi de suite.

Pour l'écran VGA, nous nous sommes basés sur des cours en ligne sur le protocole de communication VGA. Nous avons d'abord affiché un pixel puis un carré et enfin une grille.

Par la suite nous avons découpé le code pour l'écran en deux entités :

- vga_sync
- graphic_manager

Leurs utilités seront détaillées plus loin.

Pour la machine à état, nous avons conçu un système permettant de gérer les différentes phases du jeu de morpion (sélection des cases, validation des mouvements, vérification des conditions de victoire et d'égalité, passage au tour d'un joueur à un autre...). La machine à état est divisée en plusieurs états correspondant à chaque étape du jeu.

III – Documentation technique

Description des entités :

1. Ecran VGA

Vga_sync : l'objectif de l'entité est de faire l'interface entre le protocole VGA et le script. Nous avons sélectionné une résolution de 640x480 et le taux de rafraîchissement de 60Hz. De plus, un excédent de pixels est ajouté au protocole qui constitue des pixels invisibles (les zones tampons pour gérer les mouvements du spot) donc nous avions à afficher 800x525 pixels. Pour ce faire en respectant le 60Hz il faut donc une clock de fréquence 25.2Mhz. Or, une clock de 100Mhz était à notre disposition. Nous avons donc utilisé un compteur modulo 4 pour diviser sa fréquence par 4, soit 25Mhz. Malgré l'écart de 0.2Mhz l'affichage fonctionne tout de même. Puis nous avons deux autres compteurs modulo 800 pour la coordonnée x et modulo 525 pour y. Ces compteurs retournent les valeurs de leurs valeurs soit x et y puis émettent une impulsion lorsqu'ils atteignent leurs valeurs limites. Ces impulsions sont les signaux hsync et vsync qui signifient fin de ligne et fin d'image pour que l'écran change de ligne ou change d'image.

Graphic_manager : reçoit x et y et en fonction de ces coordonnées il désigne la couleur à envoyer à l'écran. Nous avons trois parties : la grille, la croix et le rond ainsi que le reste en noir en respectant la zone de dessin au milieu. Il reçoit un tableau de 18 bits pour afficher soit une croix soit un rond (ex : si les deux premiers bits du tableau valent « 00 » alors pas de croix ni de rond dans la première croix, si « 01 » alors affiche une croix dans la première case, si « 10 » alors affiche un rond dans la première case. De plus, nous avons rendu la taille de la grille paramétrable : la position de la grille (x, y), la taille des cases (case_size), la taille de la marge entre la case et la grille (margin), la taille des barres de la grille (border).

2. Machine à état

Le Game Manager est l'entité dans laquelle se trouve la machine à états permettant de créer et gérer le jeu. Cette machine à états que nous avons conçue est composée de six états. Initialement, nous avons tenté d'implémenter une machine de type Moore, mais cela n'a jamais abouti à un résultat satisfaisant. Nous nous sommes donc orientés vers la méthode de Mealy, qui s'est avérée fonctionnelle. Pour mener à bien le Game Manager, nous avons utilisé plusieurs variables : « choice_i » qui est une entrée pour indiquer la case choisie par le joueur. Cette entrée est sur 9 bits car il y a 9 cases. « changed_bit_index » qui est une entrée sur 9 bits également qui permet de sauvegarder l'indice de la case sélectionnée par le joueur. « cases_full » qui est une entrée sur 2 bits permettant d'indiquer si toutes les cases sont pleines afin de détecter si un match nul est présent.

- **J1** : Cet état correspond à l'état où le joueur n°1 joue, c'est-à-dire que dans cet état, la machine attend que le joueur effectue un coup valide. Les conditions à respecter pour passer à l'état suivant sont les suivantes : la case choisie doit être libre, le bouton de validation (bouton central) doit être pressé pour confirmer le choix de la case, et le joueur ne doit sélectionner qu'une seule case à la fois. Si ces conditions sont remplies, alors une croix bleue est écrite sur la case sélectionnée et la machine passe à l'état suivant. Si l'une des conditions n'est pas remplie (case occupée, bouton non pressé ou sélection multiple), la machine à état reste sur l'état J1 et le joueur doit recommencer son tour. Cet état est également l'état de départ du jeu, où le joueur n°1 commence à jouer. De plus, si une partie est réinitialisée à l'aide du bouton reset (bouton du haut), la machine retourne à cet état J1, permettant ainsi de recommencer une nouvelle partie.

- **Valid_j1** : Cet état intervient après le coup joué par le joueur n°1 et permet d'analyser l'évolution de la partie. Dans cet état, la machine à état vérifie si le coup effectué par le joueur n°1 a entraîné une victoire ou un match nul. Pour ce faire, un processus de vérification des conditions de victoire (La combinaison des croix

de la partie est similaire à l'une des 9 combinaisons possibles à la victoire) ou de match nul (toutes les cases occupées sans gagnant) est effectué. Si l'une de ces conditions est remplie, l'état suivant sera directement lié à la situation : si une victoire est détectée, la machine passera à l'état de victoire, et si un match nul est constaté, l'état suivant sera celui de l'égalité. Si aucune de ces conditions n'est validée, cela signifie que la partie continue, et l'état suivant sera l'état du joueur n°2, lui permettant ainsi de jouer son tour.

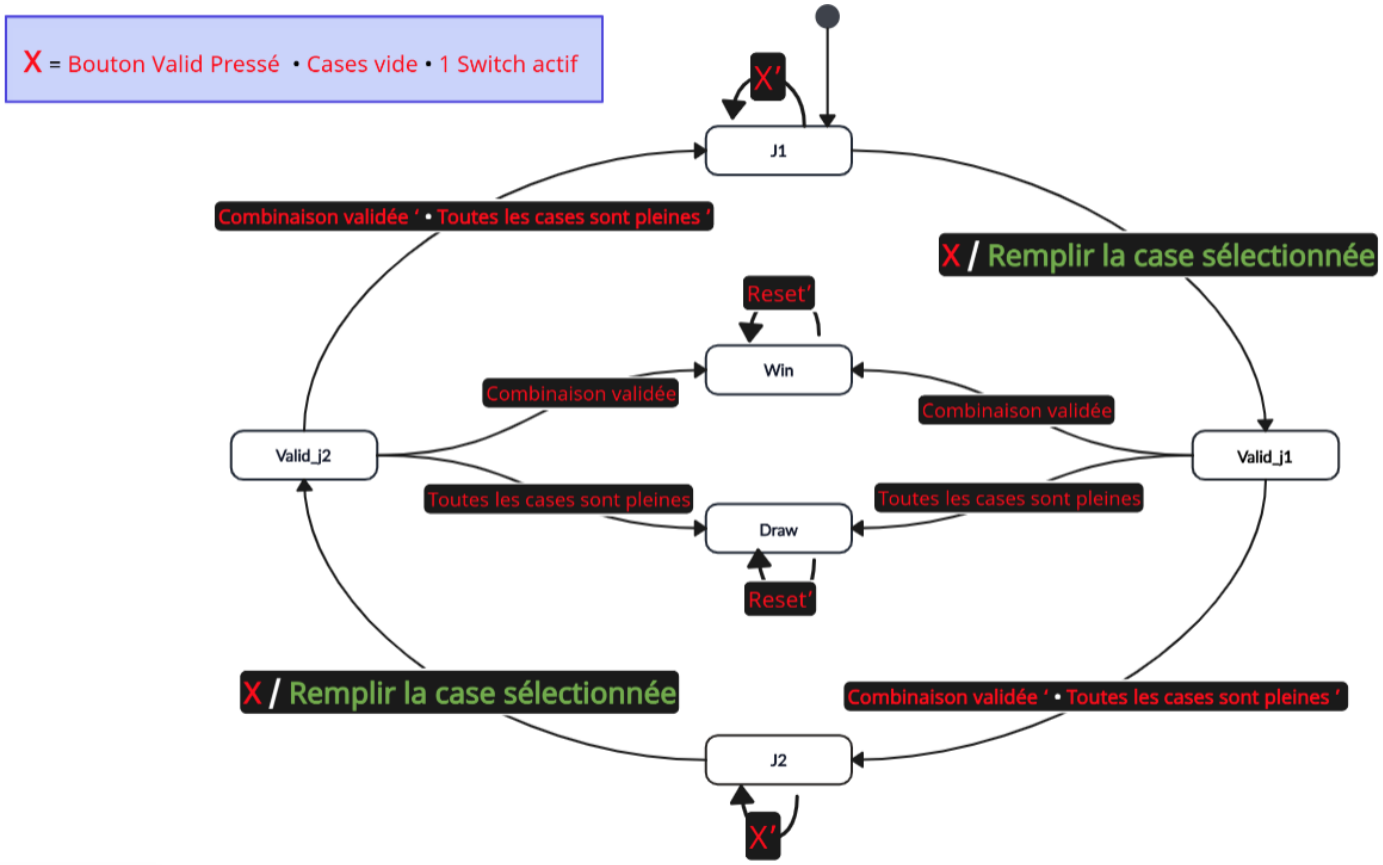
- **J2** : Cet état correspond à l'état où le joueur n°2 joue. Il fonctionne de la même manière que l'état J1, mais cette fois, le joueur n°2 choisit une case pour y inscrire son symbole (un carré rouge). Les conditions à respecter pour passer à l'état suivant sont les mêmes que pour le joueur n°1 : la case choisie doit être libre, le bouton de validation doit être pressé et une seule case doit être sélectionnée. Si ces conditions sont remplies, la machine à état enregistre le coup, affiche un rond dans la case choisie et passe à l'état suivant, qui sera l'état Valid_j2.

- **Valid_j2** : Cet état intervient après le coup joué par le joueur n°2 et permet de vérifier si ce coup a entraîné une victoire pour le joueur n°2 ou un match nul. Le processus de vérification est identique à celui de l'état Valid_j1 : la machine vérifie si le joueur n°2 a aligné trois symboles ou si toutes les cases sont occupées sans gagnant. Si l'une de ces conditions est remplie, la machine passe à l'état correspondant (victoire ou égalité). Si aucune de ces conditions n'est validée, l'état suivant sera l'état du joueur n°1 donc J1, et ainsi le jeu continue.

- **Win** : Cet état est activé lorsque la condition de victoire est validée lors des états Valid_j1 ou Valid_j2. Une fois que cet état est atteint, la machine arrête le jeu afin d'empêcher la transition vers un autre tour, garantissant ainsi que le jeu ne continue pas après qu'un joueur ait gagné. Dans cet état, aucune nouvelle action n'est autorisée, et la partie est considérée comme terminée. Le seul moyen de quitter cet état et de recommencer une nouvelle partie est de presser le bouton reset. Cependant, bien que cette fonctionnalité ait été prévue, elle n'a pas pu être implémentée de manière entièrement fonctionnelle dans notre projet. Ainsi, lorsqu'une combinaison gagnante est détectée pendant une partie, le jeu ne s'arrête pas et continue de fonctionner normalement, sans prendre en compte la victoire.

- **Draw** : Cet état est similaire à l'état Win, mais il est activé lorsqu'un match nul est détecté. La machine à état vérifie le nombre de cases remplies et, si toutes les 9 cases sont occupées sans qu'aucun des deux joueurs n'ait gagné, elle passe à cet

état. Cela signifie que la partie est terminée, mais qu'aucun joueur n'a remporté la victoire. Comme pour l'état Win, l'interruption du jeu dans cet état empêche la machine de continuer à passer d'un tour à un autre. Le seul moyen de quitter cet état et de recommencer une nouvelle partie est de presser le bouton reset.



3. Processus principal

Main : Instancie les entités vga_sync, graph_manager, game_manager et relient leurs entrées et sorties entre les entrées/sorties de la carte et entre eux.


Carte
switches_i
btn_center_i
btn_up_i
clock_main_i
main
inputs
clk_main_i
btn_up_i
clk_main_i
btn_up_i
btn_center_i
switches_i
vga_sync
inputs
game_manager
clk_i
reset_i
reset_i
diviseur_mod_4
reset_o
reset_i
compteur_mod_800
reset_o
cases
reset_o(h_sync)
count_o(x)
compteur_mod_525
reset_o(v_sync)
count_o(y)
outputs
x
y
h_sync
v_sync
graph_manager
vga_red
green_red
blue_red
outputs
vga_red
vga_green
vga_blue
hsync
vsync
Ecran

IV – Conclusion

Ce projet de création d'un jeu de morpion codé en VHDL et affiché à l'aide d'un écran en VGA en utilisant la carte Basys3 a permis de mettre en œuvre nos compétences techniques dans plusieurs domaines, notamment la conception de circuits numériques, la gestion d'une machine à états, et l'affichage graphique via le protocole VGA.

Malgré des efforts importants, certains problèmes persistent :

La fonctionnalité de reset entraîne, une fois sur deux, un comportement imprévisible qui rend le jeu non fonctionnel. L'état Win, censé arrêter le jeu lorsqu'une victoire est détectée, n'est pas pleinement opérationnel : le jeu continue de fonctionner même après qu'un joueur ait aligné trois symboles.

Cependant, en dehors de ces limitations, le jeu fonctionne globalement comme prévu. Les principales fonctionnalités sont opérationnelles :

- L'affichage de la grille sur l'écran VGA, avec des tailles paramétrables.

- La gestion du tour des joueurs, l'affichage des croix et des ronds en fonction des choix valides, ainsi que le basculement automatique entre les joueurs.

- La détection d'une égalité.

Ce projet représente donc une réussite satisfaisante, car nous avons pu atteindre la plupart des objectifs fixés dans le cahier des charges, tout en approfondissant notre maîtrise des concepts de base de la conception numérique et du langage VHDL.