GORI Aurélien

A la recherche de l'âme sœur

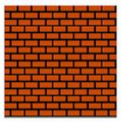
1) Le principe du jeu.

Le jeu est un plateformer : il s'agit de déplacer un personnage jouable sur une carte, tout en arrivant à un point d'arrivée en évitant les obstacles présents sur cette carte. Le point d'arrivée est modélisé par le haut de l'écran, en y = 0.

Pour ce jeu, j'ai utilisé des sprite, comme vu en TP. Je gère un sprite pour l'écran d'acceuil, 3 sprite sur l'écran de jeu, 1 sprite de game over et un sprite de fin de jeu (victoire). Une petite animation avec les leds sera possiblement faite si le temps me le permet.

2) Sprite utilisés :

Ce sprite représente le personnage jouable, qui va se déplacer sur la map.



Premier obstacle de la carte, un mur qui sera fixe. Toute collision avec ce mur est un game over.

Deuxième obstacle de la carte, une voiture qui se déplace sur la carte. Plus dure à éviter que le mur, cette dernière sera le plus grand danger du personnage.

Ecran d'accueil du jeu : Ce dernier représente le joueur avant son périple. Instruction à effectuer pour lancer le jeu (Appuyer sur un bouton).

Ecran de Game Over. Quand le personnage entre en collision avec un obstacle, il ne rencontrera pas son âme sœur, il est donc rempli de tristesse (et nous aussi).

Dernier sprite du jeu, la consécration ultime : notre âme sœur. Ce sprite apparaît lorsque le joueur est arrivé à destination. Nous sommes tous remplis de bonheur (et de chantillée).

Malheureusement, je n'ai pas pu afficher l'ensemble de mes sprites, car je n'ai pas su comment réduire leur qualité. En effet, en ayant discuté avec les professeurs, j'ai vu qu'il était possible de définir une image de petite taille, et de remplir l'écran avec les pixels de cette image, mais je n'ai pas réussi à l'implémenter en VHDL. Mon jeu ne comportera donc malheureusement pas d'écran de Game Over. A la place, j'ai mis et un break. En effet l'écran

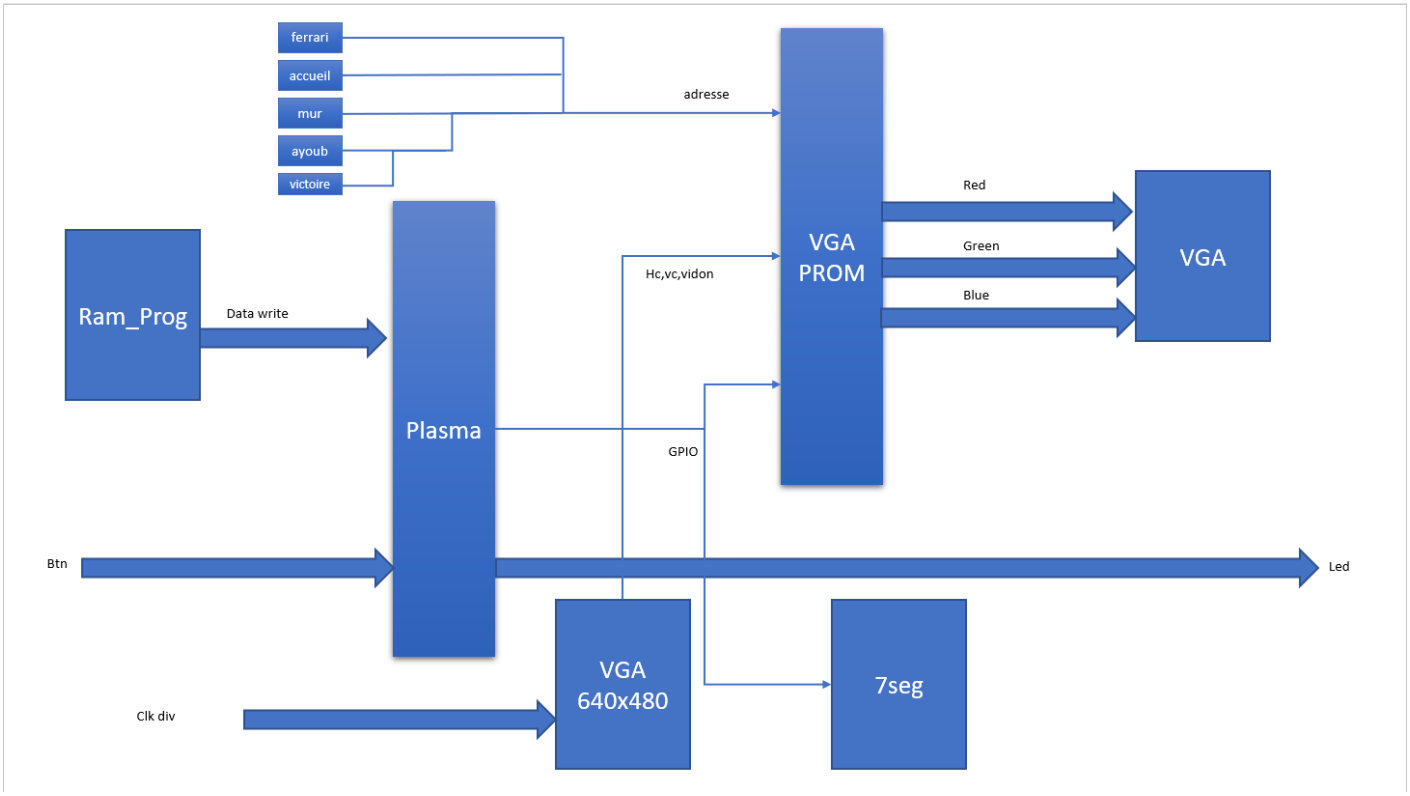
de Game Over est remplacé par l'écran de jeu, fixe, montrant la collision entre le personnage jouable et un obstacle.

De plus, j'ai utilisé l'afficheur 7 segment. Pendant l'écran d'accueil, j'affiche "btnD" pour indiquer au joueur d'appuyer sur le bouton du bas afin de lancer le jeu.

Pendant la partie, l'afficheur montre "go" pour indiquer au joueur de bouger le personnage.

Si le personnage touche un obstacle, il est alors affiché "noob" sur l'afficheur, pour inciter le joueur à être meilleur.

3) Architecture matérielle



Reprise de l'architecture classique vue en TP et en TD. J'ai rajouté des images et utilisé les périphériques extérieurs (btn, led). On reprend l'utilisation du contrôleur VGA vu en TD avec les composants VGA 640x480, Plasma et VGA PROM.

4) Organigramme du Code C

Mon code se trouve dans le fichier sprite.c. Ce dernier s'organise de la manière suivante :

Un main dans lequel j'implémente une boucle while(1) pour que le jeu tourne en continue.

Dans un premier temps je définis toutes les variables de type int dont j'aurais besoin dans le code. Ensuite, il y a 2 conditions : la première me permet de dire si on lance le jeu on ou non.

La deuxième découle de la première. Si l'on a appuyé sur le bouton du bas, on lance alors le processus du jeu. Si l'on n'appuie pas sur le bouton, on reste sur l'écran d'accueil.

Dans un premier temps, j'implémentes l'afficheur 7 segments avec l'affichage des textes désirés. Ensuite, je fais varier les coordonnées du sprite du personnage. Une fois ses réglages fait, j'affiche ensuite tous mes sprites, puis la temporisation.

Une fois mes sprite affichés, je gère le Screen Saver de la Ferrari, et je gère les collisions entre tous les sprites.

Une fois les collisions codées, j'implémente une condition de victoire. Si cette dernière est réalisée, j'affiche l'écran de victoire. Sinon, le jeu se bloque sur l'endroit de la collision et le joueur doit appuyer sur le bouton reset pour relancer le jeu.

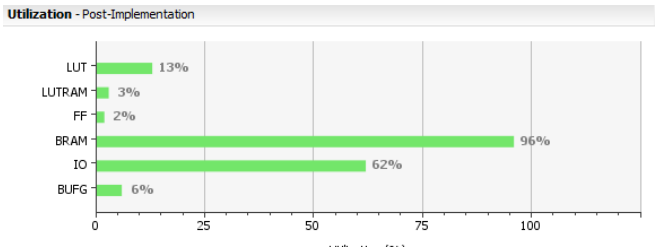
L'ensemble des fonctions que j'ai codé se situent dans le no_os.c, on y retrouve les fonctions d'affichage des sprites qui sont identiques, à l'exception des GPIO utilisées. On retrouve aussi la fonction des 7 segments, la fonction de colorisation de l'écran et les fonctions qui permettent de récupérer les données des boutons.

J'ai remarqué que lorsque que je "lance mon jeu trop tôt", les collisions sont légèrement bancales. La seule explication que je peux trouver à cela est le temps d'exécution du script en C. En effet, le moment ou je gère les collisions est assez éloigné dans le code du moment ou je gère les positions du personnage, donc je pense que si je bouge trop vite le personnage, le code n'aura pas eu assez de temps pour arriver aux lignes des collisions.

5) Résultats

A premières vues, les résultats sont satisfaisants. En effet, dès que le personnage rencontre un obstacle, le code se met en état bloquant montrant la fin de la partie. Si le joueur atteint l'arrivée (la partie haute de l'écran) alors l'écran de fin de jeu apparaît et les leds clignotent, annonçant la victoire du joueur. De plus, l'affichage est bien réalisé.

Niveau Hardware, j'ai trouvé que tout l'intérêt de ce projet résidait dans l'utilisation des ressources. En effet, j'ai pu voir que ces dernières étant limitées, il était primordial de bien savoir les gérer. Malheureusement, je n'ai pas pu afficher tous mes sprites car je ne disposais pas d'assez de RAM. Effectivement, après génération du bitstream final, voici les résultats que j'obtiens en sortie de project summary :



Resource	Utilization	Available	Utilization %
LUT	2654	20800	12.76
LUTRAM	256	9600	2.67
FF	806	41600	1.94
BRAM	48	50	96.00
IO	66	106	62.26
BUFG	2	32	6.25

On voit bien que l’on rentre assez juste dans la RAM de la basys, et cela peu s’expliquer par la taille que j’ai donné aux écrans de victoire et à la Ferrari.

Pour conclure, ce jeu a été un bon projet en autonomie nous permettant de faire le lien entre le hardware (VHDL, FPGA) et le software (code de C, compilateur). Ce dernier nous a appris à gérer des contrôleur vidéo tel que le VGA. Enfin, ce dernier nous a permis de faire preuve d’imagination et d’auto-correction, qui sont des qualités primordiales chez un ingénieur.