Lors de ce projet, nous devions réaliser un jeu vidéo simple, codé en C, avec une partie hardware en VHDL. Ce jeu devait fonctionner sur une FPGA, ici Basys 3. Nous partions donc d'une base composée de plasma_basys, réalisé lors des séances de TP, auquel nous devions réaliser des modifications matérielles et logicielles, afin de faire fonctionner le jeu. Nous avons choisi de réaliser le jeu PONG, qui fut l'un des premiers jeux vidéo d'arcade. C'est un jeu simple, mais auquel nous allons apporter quelques fonctionnalités supplémentaires.

I) Présentation générale du système :

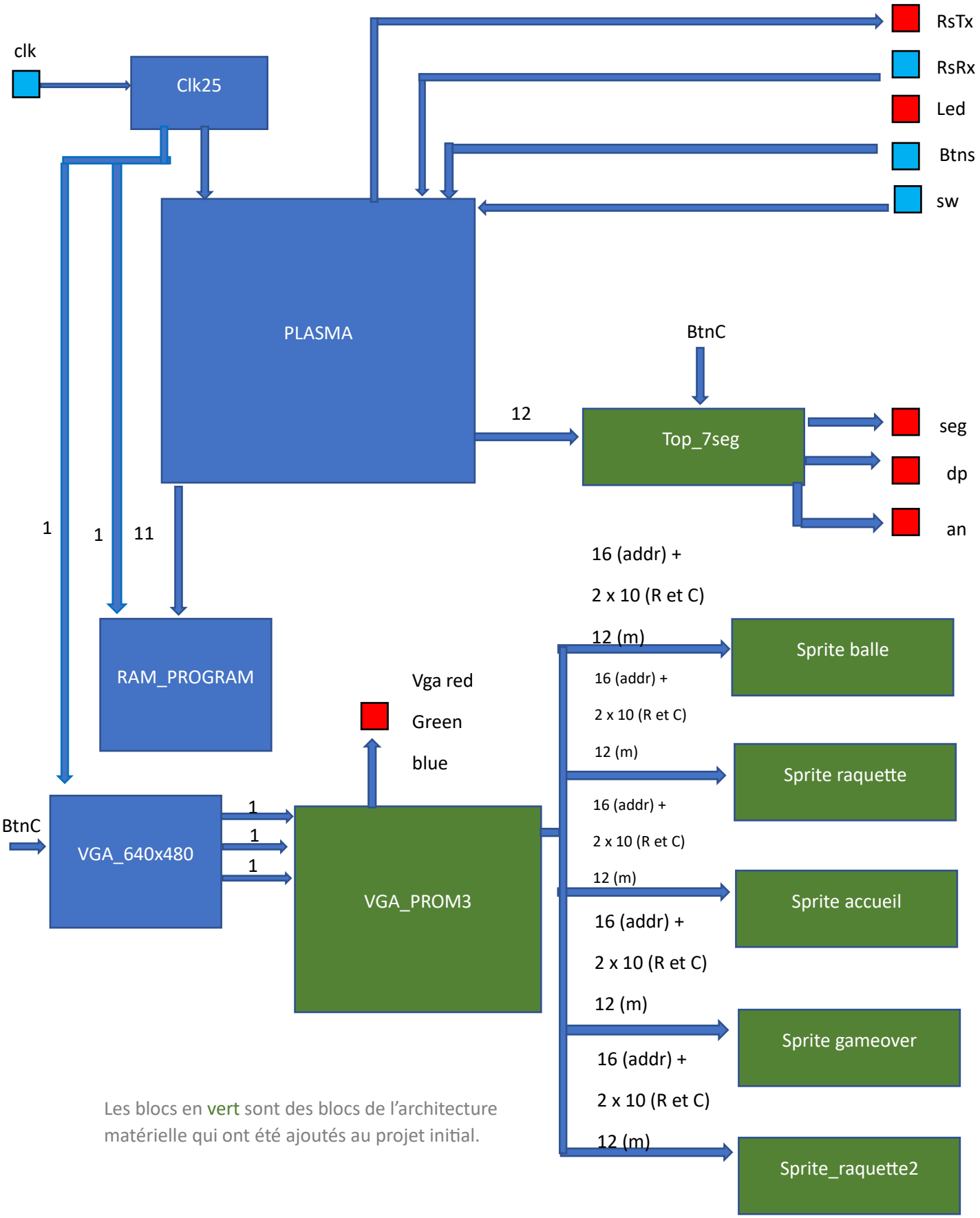
Lors du démarrage du jeu, nous avons une page d'accueil, celle-ci indique au joueur qui faut choisir un mode de jeu à l'aide des switchs.

Ce jeu sera composé de 3 modes à choisir avec les switchs de la FPGA: le premier est un mode « solo », le joueur joue seul contre l'algorithme : à l'aide des boutons btnU et btnD, il devra bouger la raquette afin de renvoyer la balle à l'adversaire. Nous avons réalisé 3 niveaux de difficulté : le premier est un mode de prise en main des commandes où la balle va lentement, pour sélectionner ce mode, il faudra mettre les switches à 0x0001. Pour les 2 autres niveaux, les switchs devront être à 0x0002 ou 0x0003, la balle va à une vitesse moyenne à élevée (pour plus de challenge, il faudra choisir le mode difficile). Pour ces modes solos, le score du joueur s'affichera sur les segments de la FPGA.

Ensuite, nous avons 2 modes en multijoueur : le premier (sw = 0x0004) est un Pong où les 2 joueurs devront s'échanger la balle sans la perdre. Ce mode est une prise en main pour le second mode multijoueur (sw = 0x0008), où le premier des deux joueurs devront essayer de faire toucher la balle sur le mur vertical adverse. Le premier arrivé à un score de 10 gagne la partie. Les scores sont affichés sur les segments (partie gauche : score de joueur 1, partie droite : score du joueur 2). Lorsque la partie est finie, l'identifiant du joueur s'affichera sur les switchs. La première balle étant une balle d'échauffement aura une vitesse réduite.

Lorsque le joueur est en mode solo, lorsque la balle arrive vers le camp adverse et qu'elle a dépassé la moitié de l'écran, la raquette adverse utilise les coordonnées de celle-ci afin de la renvoyer, puis une fois renvoyé, se replace vers le milieu de l'écran. La vitesse de cette raquette est fixée au pas : 1 pixel pour tour de boucle du jeu. Bien sûr, pour permettre au joueur de gagner, lorsque le score vaut 14, la raquette aura une position cible faussé, ce qui va permettre au joueur de gagner. Tant que le joueur n'a pas sélectionné le mode, le jeu ne peut pas démarrer.

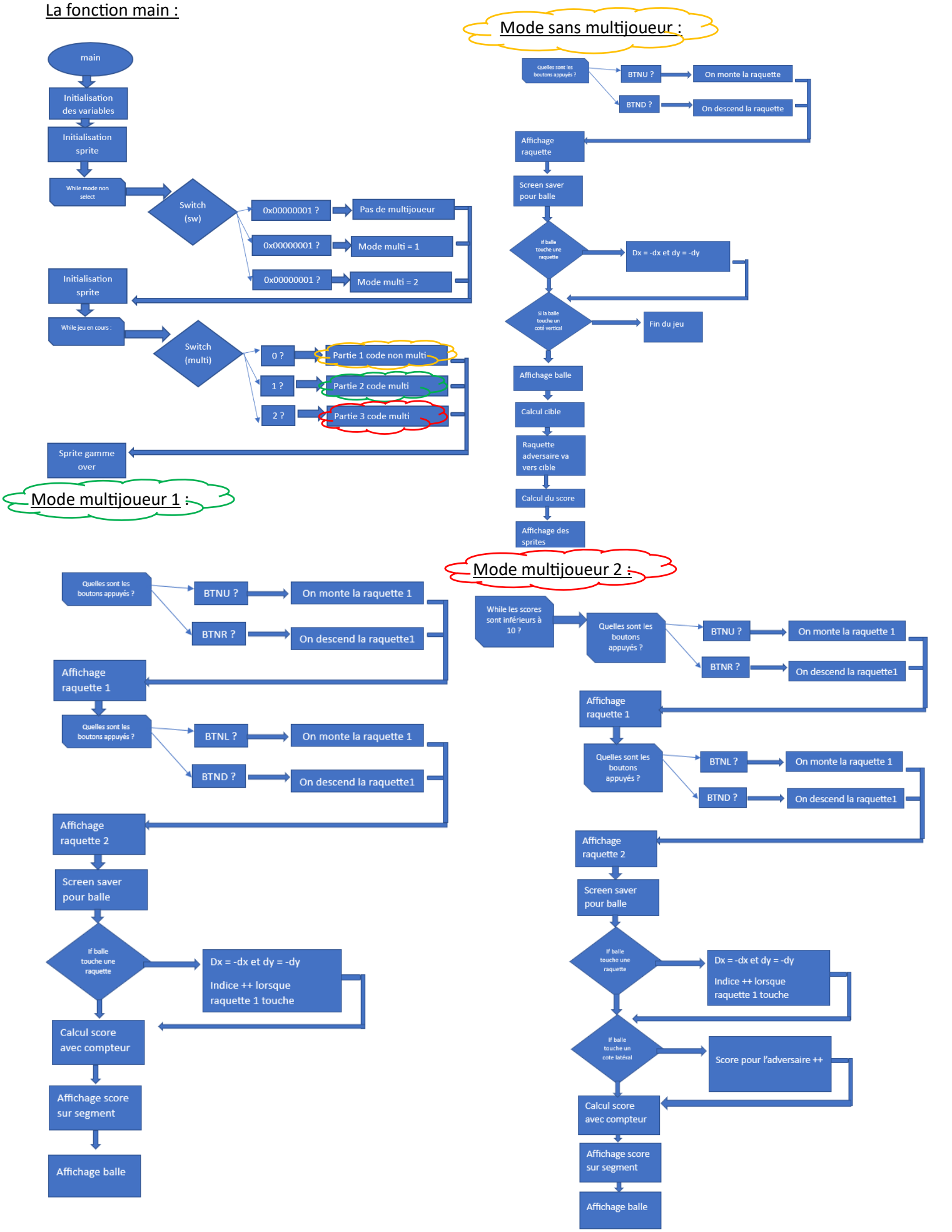
II) Schéma de l'architecture matérielle :



Les blocs en vert sont des blocs de l'architecture matérielle qui ont été ajoutés au projet initial.

III) Organigramme du code C :

La fonction main :



Explication du fonctionnement global :

Le code est composé de 2 switch cases : un pour déterminer si le joueur a sélectionné ou non son mode de jeu. Et un autre qui va déterminer quel doit être le fonctionnement du jeu dans ce mode. Quand le joueur joue seul, il ne pourra que bouger la raquette de droite à l'aide des bouton U et D. Lorsque la balle est envoyée du coté ennemi, l'adversaire ne commencera qu'à bouger sa raquette que lorsque la balle dépasse la moitié de l'écran et il se dirigera vers la coordonnée x de la balle du moment (fonction « Nostradamus »). Le score est calculé via un compteur : en effet, afin de réaliser des vitesses pour la balle, nous avons réalisé un compteur qui prend soit 2, 3 ou 5 tours de boucle (suivant la difficulté) avant de bouger la position de la balle. En multijoueur, grâce à une succession de if et avec des opérations de masquages, nous déterminons quelles sont les boutons qui ont été pressé, et les raquettes bougent suivant les appuis. Des boucles de test ont été ajouté afin d'éviter que la position de la raquette ne sorte de l'écran. Le code du screen saver a été utilisé pour les rebonds de la balle. Pour l'affichage du score, c'est la fonction « SEG7 » qui a été implémenté. Les fonctions « VGA_sprites » sont les fonctions qui gèrent l'affichage et le positionnement des images qui leur sont associés.

Lien entre le logiciel et le matériel :

Le code C compilé (après l'appel de la commande « make …») est en réalité un fichier binaire, rempli de 0 et 1. Ces chiffres sont stockés en mémoire sous forme de paquets, le processeur va alors lire et interpréter ce nombre comme une instruction à réaliser. Les fonctions « Memory Read » et « Memory Write », nous permettent d'accéder à un registre via le code C. Elles lisent (« Read ») ou écrivent (« Write ») dans la mémoire, Nous passons en paramètre l'adresse du registre concerné, ainsi qu'une éventuelle valeur à écrire. Les adresses des GPIOs sont stockées dans le fichier « plasma.h » qui est inclus dans le code C. Ce sont donc ces fonctions qui vont lire ou écrire dans le registre au niveau matérielle (hardware). Dans le code VHDL (partie hardware), pour écrire une valeur dans un registre, nous utilisons la ligne de code suivante :

```
elsif cpu_address(7 downto 4) = "0011" then
   gpio0_reg <= gpio0_reg or cpu_data_w;
```

Pour tout affichage d'image, il faut également créer un sprite dans la partie hardware, que l'on connectera à un nouveau GPIO. Les réglages de positions de l'image se feront alors en C, via les fonctions VGA_sprite, qui seront reliées elles même au GPIO concerné.

Conclusion : Ce projet nous a permis de réaliser une approche concrète de la programmation de système embarqué, avec la programmation de la partie hardware en parallèle de la partie software (faite ici en C). Ce jeu nous a mieux fait comprendre les enjeux de la programmation matérielles et comment elle est mise en rapport avec la partie logicielle, avec des contraintes d'espace et de performance avec la Basys 3.