Rubik's Cam Project PHY573a

Parakian Alexis, Skrzypczak Nathan

15 décembre 2013

Résumé

Votre résumé commence ici... ...

1 Présentation

Rubik's cam est un projet développé au cours de l'Enseignement d'Approfondissement : "Conception expérimentale micro et nanoélectronique" partie FPGA. Le but étant comme son nom l'indique, d'afficher un rubik's cube à l'écran en 3 dimensions et de le résoudre à l'aide de mouvements effectués devant la caméra. Ce projet ambitieux nécessite alors la réalisation de deux parties bien distinctes qui nous permirent de se départager le travail. Nathan Skrzypczak prit en charge la réalisation d'un moteur 3D et Alexis Parakian, celle de la détection de mouvements via la caméra.

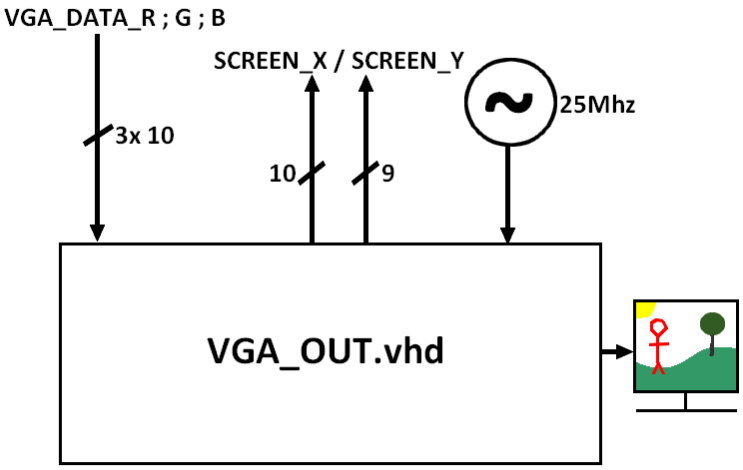
2 Architecture

2.1 Contexte et Entrées Sorties

2.1.1 Sortie VGA

La première partie du projet à être réalisée aura été la sortie VGA. Elle l'a été sous la forme d'un composant pour plus de facilité d'utilisation. Il est contenu dans le fichier VGA_OUT.vhd

Le point à observer en particulier est la fréquence d'horloge, en effet celle-ci conditionne la conception du reste du circuit. On cherche à cadencer le rafraichissement à 60Hz. Une trame comprends 795 colonnes et 525 lignes (pour 640x480 utiles) soit 417900 points sur l'écran. Obtenir 60Hz par trame revient alors à **cadencer les points à 25.074Mhz**. On choisira ainsi une **fréquence utile de 25Mhz** pour rester en phase avec le reste du circuit, soit 59.82Hz par frame (on commet une erreur de 0.3% qui est admise dans la norme VGA.

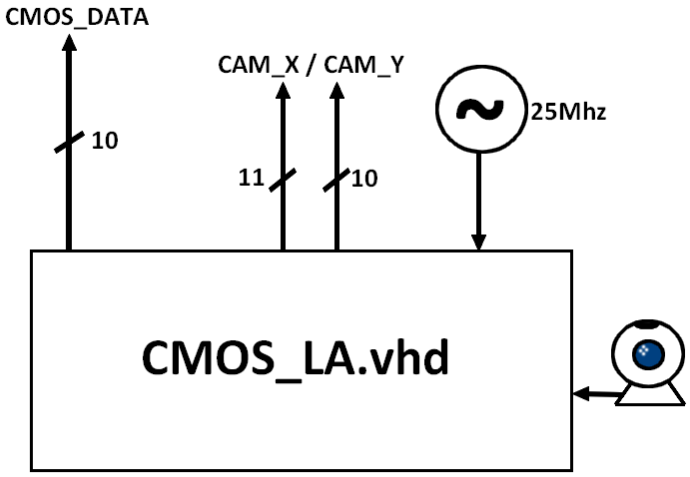


- Ce composant est cadencé sur une horloge à 25Mhz (fréquence d'affichage d'un pixel)
- Il sort deux signaux entiers donnant la position du pixel courant : SCREEN_X [0..639] et SCREEN_Y [0..479] sur 10 et 9 bits respectivement.
- la couleur du pixel courant est donnée en entrée sur trois bus de 10 bits : VGA_DATA_R ; VGA_DATA_G ; VGA_DATA_B

2.1.2 Entrées : Camera CMOS

Concernant l'entrée par la camera, nous nous sommes inspirés d'un exemple de la bibliothèque, que nous avons quasiment entièrement réécrit. Il s'agit du programme de dialogue avec le CMOS, paramétrage de la caméra et formatage des données.

Un point particulièrement important dans l'utilisation de ce périphérique connecté au GPIO de la carte DE2 est **la gestion des horloges**. En effet, la caméra prends en entrée une horloge cadencée à 25Mhz, et retourne les données lues sur les pixels CMOS à cette même fréquence. Cependant, ce processus a une certaine durée d'exécution, surtout à cause de la présence de la nappe, et du câblage interne, qui ajoute un retard moyen de 10ns (soit $\frac{1}{4}$ de periode à 25Mhz). L'horloge générale utilisée pour le reste du circuit devra donc tenir compte de ce retard de 10ns de manière à assurer la synchronisation des données. (Voir la section Horloge pour plus de précisions)



- Ce composant est cadencé sur une horloge à 25Mhz (acquisition des pixels)
- Il sort deux signaux entiers donnant la position du pixel courant : CAM_X $[0..(480 * 2 - 1)]$ et SCREEN_Y $[0..(640 * 2 - 1)]$ sur 10 et 11 bits respectivement.
- la couleur du pixel courant est donnée par l'organisation du capteur

G	R	...
B	G	...
⋮	⋮	⋱

2.1.3 Mémoires

Le choix des mémoires aura posé de nombreuses difficultés, dans la mesure où les composants présents fixent de nombreuses contraintes, et que le matériel disponible (pour les mémoires) possède également les siennes. On souhaite donc afficher sur la sortie VGA une figure 3D projetée dans le plan de l'écran. Les éléments qui seront donc sujet à une forte occupation mémoire, et qui ne peuvent pas se résoudre par l'utilisation de bascules sur la carte seront donc le stockage des points de cette figure et de l'affichage de l'écran si celui-ci doit être stocké dans un buffer.

1. **Figure 3D**
   - Pour la représentation en 3D de la figure, la solution la plus simple est de la décomposer en triangles (dans la mesure ou ils sont nécessairement plans). Ces triangles (ensembles de 3 points, donc 9 coordonnées de l'espace) seront tout stockés indépendamment les uns des autres par souci de simplicité. Il est possible d'optimiser grandement le stockage en utilisant des recouvrements (dans la mesure ou ils partagent souvent des sommets), mais ceci génère de la complexité dans le calcul et la gestion de cette mémoire.
   - La quantité de mémoire à prévoir pour afficher un Rubik's cube et donc en première approximation **de 108 triangles soit 972 coordonnées**.

– Il faut également considérer que nous manipulons des points qui seront probablement sujet à des translations et des rotations. Il nous faut donc garantir une bonne précision pour éviter la divergence des données par approximations successives. Une précision de 32bits avec virgule fixe à 16bits semble raisonnable (Avec des coordonnées allant de -32768 à 32767 et une précision de 10^{-4}, on peut exprimer des points à 'l'infini visuel' et se prémunir des erreurs d'approximation grossières) Cette précision reste ajustable en réduisant les coordonnées accessibles.
– Avec ces observations, on atteint **32Kbits de données à stocker**, or le Cyclone II possède 100 puces M4K accessibles en lecture / écriture. On peut donc réaliser ce stockage directement sur la carte avec 8 telles puces.

Les triangles seront donc stockés sur une mémoire ***double port*** *de* ***32bits de données*** *et de* ***10bits de bus d'adresses****.*

Les coordonnées seront stockées de cette manière pour un triangle (en partant de l'adresse 0x000) :

...	x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	...

2. **Ecran**
   – Ayant vu ceci, il est nécessaire de parcourir l'ensemble des M4K occupés et de traiter les points présents pour obtenir l'image à afficher à l'écran. La fréquence d'affichage des pixels étant de 25Mhz et comme il est déraisonnable d'espérer faire fonctionner la mémoire interne à plus de 100Mhz, il est impossible de compter sur un traitement temps réel : **Nécessairement les différents pixels devront être stockés en mémoire**
   – Reste donc à définir une telle mémoire pour un écran de 640x480 pixels. La mémoire la plus simple d'utilisation offrant tout de même une capacité raisonnable est la SRAM. En effet, pour rentrer dans les 100 x M4K de la mémoire on-chip avec une profondeur de couleur de 10bits en niveaux de gris, il faudrait se limiter à 160x120...
   – La SRAM permet de stocker 256 kilomots de 16 bits ce qui est largement suffisant pour une profondeur de 10bits (niveaux de gris ou couleurs réduites) sur 640x480 pixels. Cette mémoire peut également être accédée jusqu'à 100Mhz, ce qui permet une réactivité suffisante pour la sortie VGA.
   – Pour ce qui est de la répartition des données, cette dernière est un peu subtile dans la mesure ou les plages d'adresses et de données ne sont pas paramétrables de la même façon que sur le carte. On se limite dans une première approximation à une plage de pixel de 320x240 pour simplification, mais la modification de la formule permettrait d'étendre à la résolution supérieure. L'adresse se calcule de la manière suivante :

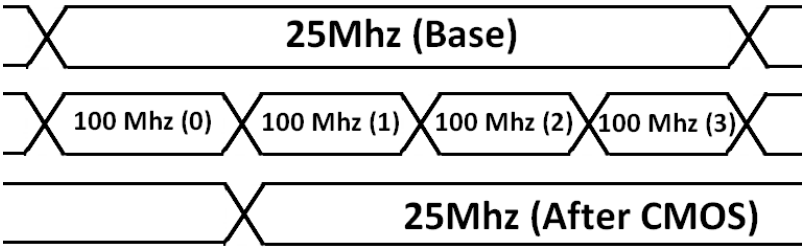
$$Adresse_{18bits} = \frac{x}{2} + 320.\frac{y}{2}$$

2.2 Horloge

Étant donné les éléments en jeu dans ce circuit, **l'horloge principale sera donc cadencé à 100Mhz** (fréquence maximale de fonctionnement de la SRAM), de manière à permettre à cette dernière une lecture et une écriture dans une

étape de calcul à 25Mhz (Une écriture dure en effet 3 cycles de 10ns et une lecture un cycle de 10ns).

La caméra induit un décalage de 10ns, on cadence donc cette dernière sur le la base 25Mhz alors que l'ensemble du circuit suit une cadence avec un cycle de 100Mhz de retard (donc 10ns)



La synchronisation est réalisée par un signal comptant les fronts d'horloge dans chacun des composants, et agissant comme clock enable pour le front à sélectionner.

2.3 Acquisition

2.4 Moteur d'affichage

2.5 Détection de mouvement

Le programme de détection de mouvement est assez simpliste comparé à ce qui existe déjà mais l'idée principale était de faire quelque chose d'assez léger qui puisse tourner en temps réel et sans trop utiliser de mémoire (La SRAM et les M4K étant déjà mises à rude épreuves par le moteur 3D). Nous nous sommes principalement inspirés de l'algorithme de Viola et Jones de détection d'objet dans une image numérique. Cette méthode est l'une des plus connues et plus utilisées dans la détection de visages et de personnes.

Elle consiste au parcours de l'ensemble d'une image en calculant un certain nombre de caractéristiques dans des zones rectangulaires qui se chevauchent. Pour calculer rapidement et efficacement ces caractéristiques sur une image, les auteurs proposent également une nouvelle méthode, qu'ils appellent "image intégrale. C'est une représentation sous la forme d'une image de même taille que l'image d'origine, qui en chacun de ses points contient la somme des pixels situés au-dessus de lui et à sa gauche.

Les caractéristiques sont calculées à toutes les positions et à toutes les échelles dans une fenêtre de détection de petite taille, typiquement de 24 x 24 pixels.