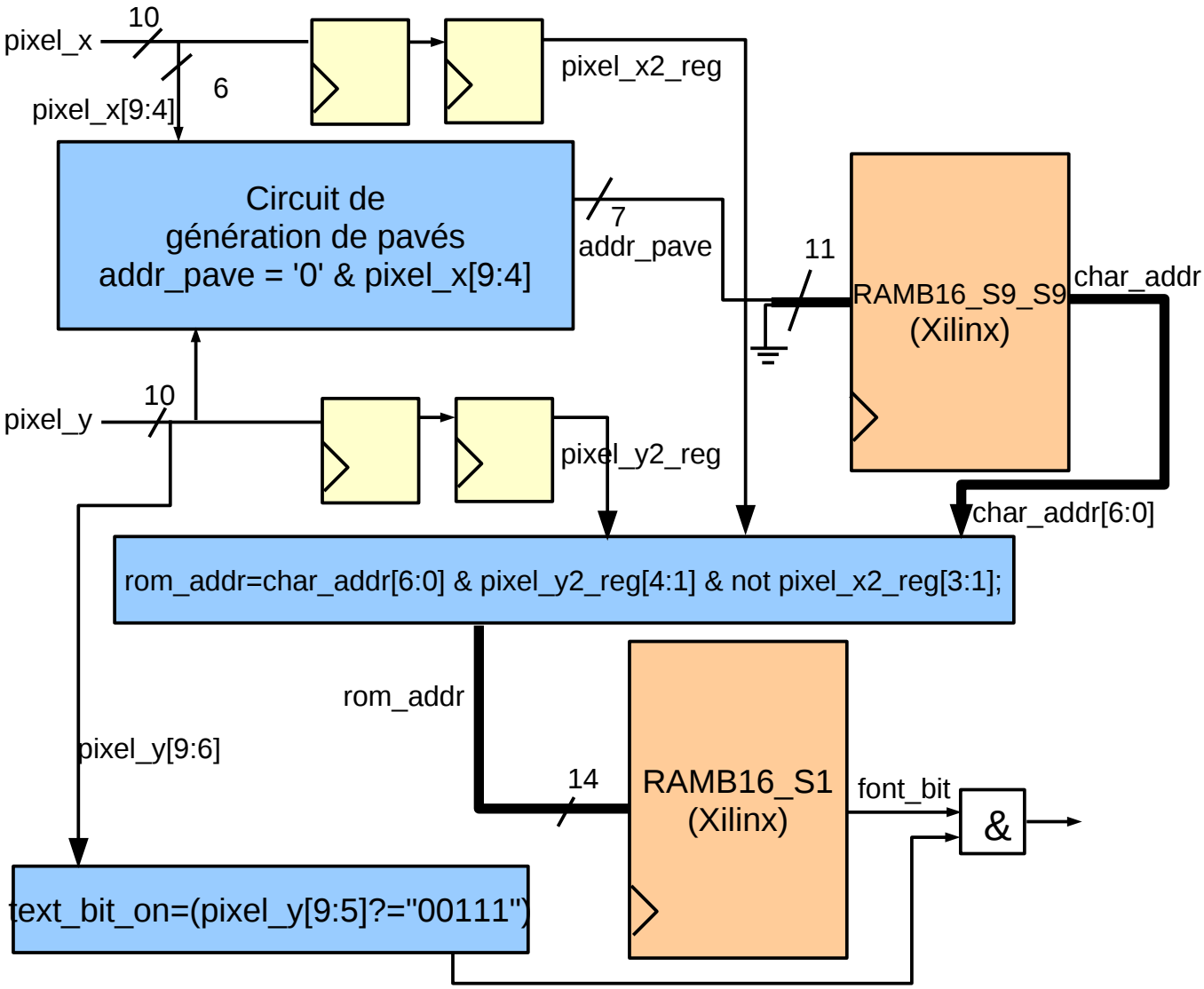
Introduction

L'objectif de cet examen est de réaliser (partiellement) un jeu vidéo appelé casse-brique. Il s'agit d'une balle rebondissant sur les murs et d'une raquette destinée à empêcher la balle de passer. Des briques sont présentes (ou pas) dans le jeu qu'il faut détruire. Le mouvement de la raquette sera encore réalisé par le bouton rotatif de votre carte (cela rappelle un peu le médian).
Dans cette épreuve , on ne vous demande pas de vous poser des questions sur l'utilité ou non des modifications mais seulement de les exécuter et de montrer que le travail résultant fonctionne toujours. Le but de l'épreuve est ainsi d'évaluer comment vous appréhendez des modifications à partir d'un schéma complexe décrit en VHDL.
Vous disposez d'une feuille réponse sur laquelle vous répondez et sur laquelle l'enseignant notera ce qu'il a vu fonctionner et sinon, quelques indications sur l'état d'avancement de votre travail.
Les ressources se trouvent dans le répertoire "final2014" du fichier ressources utilisé en TP.

I. Prise de contact avec la gestion d'un écran VGA (sans processeur)

Préparation de l'écran VGA en mode texte
Le principe de base est réalisé par le schéma ci-dessous :



Pas de panique cet ensemble vous est donné (dans CasseBriqueFinal2014.vhd) !
Le point essentiel est l'apparition de deux mémoires :
- une ROM de caractères (RAMB16_S1) dans laquelle on stocke la forme de ces caractères qui seront

dessinés sur l'écran,
- une RAM vidéo (RAMB16_S9_S9) dans laquelle on stocke ce qui sera écrit sur l'écran.
En ce qui concerne cette épreuve, seule la partie basse de l'écran sera remplie par des caractères.

Travail de préparation (2 pts exo0)

Les coordonnées graphiques de l'écran dans le mode choisi sont $0 \leq x \leq 639$ et $0 \leq y \leq 479$. Donner sur votre feuille réponse le nombre de bits nécessaires pour coder x et y en binaire.

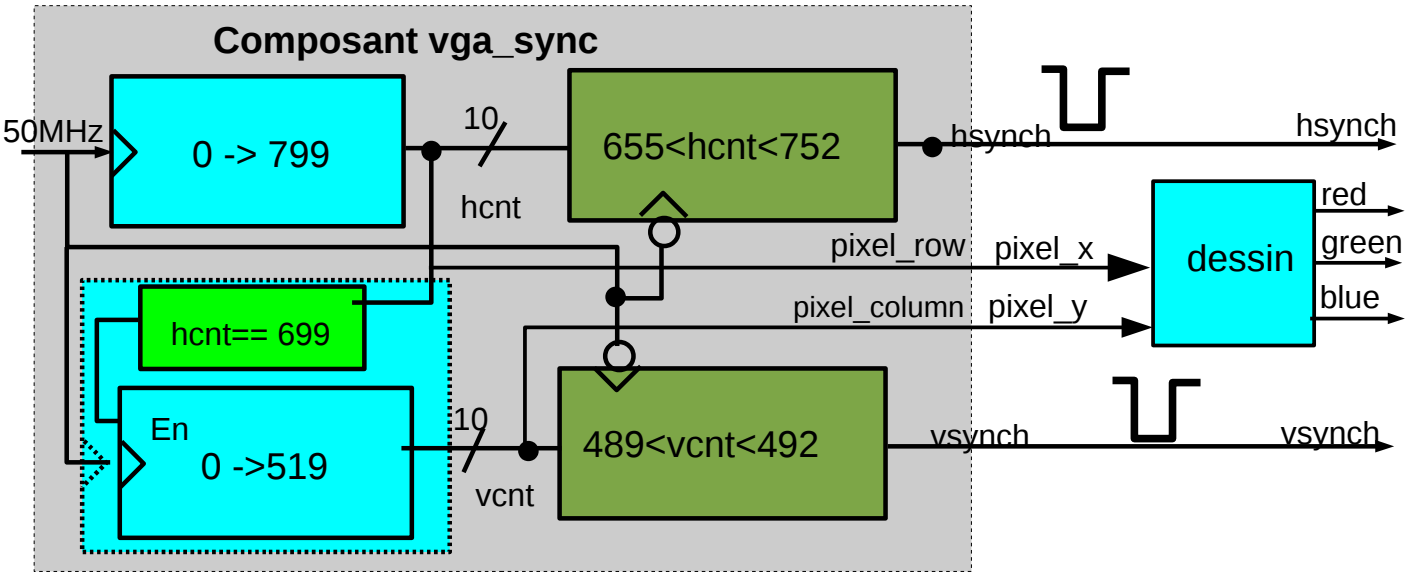
Réponse sur feuille réponse

Travail à réaliser sans processeur (4 pts exo1)

1-1) On vous demande de réaliser un projet qui aura comme fichier source "cassebriqueFinal2014.vhd" donné et de le tester. N'oubliez pas de réaliser l'ucf qui va bien. Qu'est-ce qui est affiché sur l'écran ?
Puis modifier la RAMB16_S9_S9 pour que soit affiché " UTT" en lieu et place de "GEII". Cherchez les codes ASCII des caractères sur Internet pour ce travail.

ATTENTION : la RAMB16_S9_S9 s'initialise à partir de la droite.

1-2) Vous avez dans la figure ci-dessous la gestion de la partie graphique supérieure de l'écran. C'est le petit rectangle bleu clair complètement à droite. Le reste sert à synchroniser l'écran et était donc présent dans la question précédente.



Le composant bleu reçoit les coordonnées du point de balayage courant (pixel_x,pixel_y) et doit sortir les couleurs si ces valeurs sont comprises entre certaines valeurs prédéfinies.
Réaliser le dessin d'un rectangle rouge au milieu de l'écran si la résolution est 640x480.
Indication : Ce travail se fera en VHDL directement dans le code de "CasseBriqueFinal2014.vhd". Comme il y a déjà du dessin (de texte) sur l'écran, votre travail se fera en 3 étapes :
- déclaration d'un signal "sred1" par exemple
- réalisation d'une équation combinatoire à partir de pixel_x et pixel_y du genre :

```
sred1 <= '1' when  pixel_x > ??? and pixel_x < ??? and pixel_y > ??? and pixel_y < ??? else
          '0';
```

- ajout de sred1 dans le ou logique (qu'il vous faut trouver juste avant la sortie red).

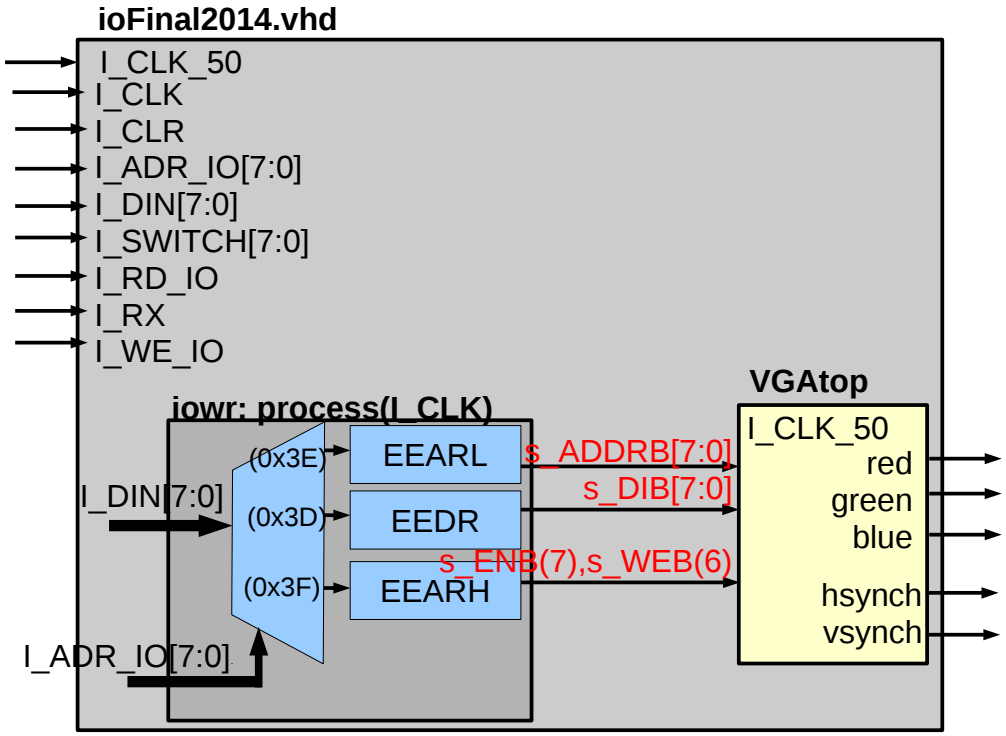
Si vous voulez un deuxième rectangle, vous faites de même et vous faites un OU logique entre les couleurs avant de les sortir. On ne vous demande pas de deuxième rectangle mais retenez cette façon de procéder.

II. Gestion du texte avec le processeur

Comme vous avez pu le remarquer avec la question précédente, la partie basse de l'écran affiche du texte. Nous allons essayer de modifier le texte affiché avec un processeur maintenant.

ATTENTION : Ce travail sera obligatoirement réalisé avec la version ATMega8 qui vous est donné dans les ressources pour cet examen final 2014. Un fichier "ioFinal2014.vhd" spécifique y est utilisé et tous les PORTs extérieurs ont été retirés.

Vous allez maintenant interfacer l'ensemble de la question précédente (VGAtop) au processeur ATMega8 comme indiqué dans la figure ci-après.



Les entrées non utilisées du module VGAtop (dans "CasseBriqueFinal2014.vhd") seront reliées à 0.

Vous sortirez naturellement tous les signaux "red", "green", "blue" "hsynch" et "vsynch" du processeur.
Ce que l'on vient de réaliser est une interface entre la mémoire caractères (qui définit ce qui est affiché à l'écran) et notre processeur. La technique pour écrire dedans consiste à :
- mettre EEARH à 0
- positionner l'adresse où l'on va écrire dans EEARL
- positionner la donnée que l'on veut écrire dans EEDR
- mettre EEARH à 0xC0 (pourquoi?)
- mettre EEARH à 0

Travail à réaliser (6 pts exo2)

2-1) Réaliser la partie matérielle

Indication : I_CLK_50 du VGAtop doit absolument être relié au I_CLK_50 (50 MHz) du module io.vhd

2-2) Écrire un sous-programme qui affiche les caractères 00 à 99 suivant la valeur passée en paramètre dans la partie après "Level" de l'écran. Ce que fait ce sous-programme c'est de prendre les 4 bits de poids faible et de les afficher à la place des unités et les 4 bits de poids fors et de les afficher à la place des dizaine.

Indication : les unités sont en adresse 102 et les dizaines en adresse 101

2-3) Écrire un programme complet qui compte en BCD et affiche toutes les secondes ce comptages à l'aide du sous-programme de la question 2-2). Commencez peut-être par compter de 0 à 9 sur un digit, ... puis 2.

III. Gestion du bouton rotatif

On vous donne le programme "Left_Right_Leds.vhd" que vous avez utilisé en médian. Vous devez dans ce code repérer les décalages et les remplacer respectivement par une incrémentation et une décrémentation.

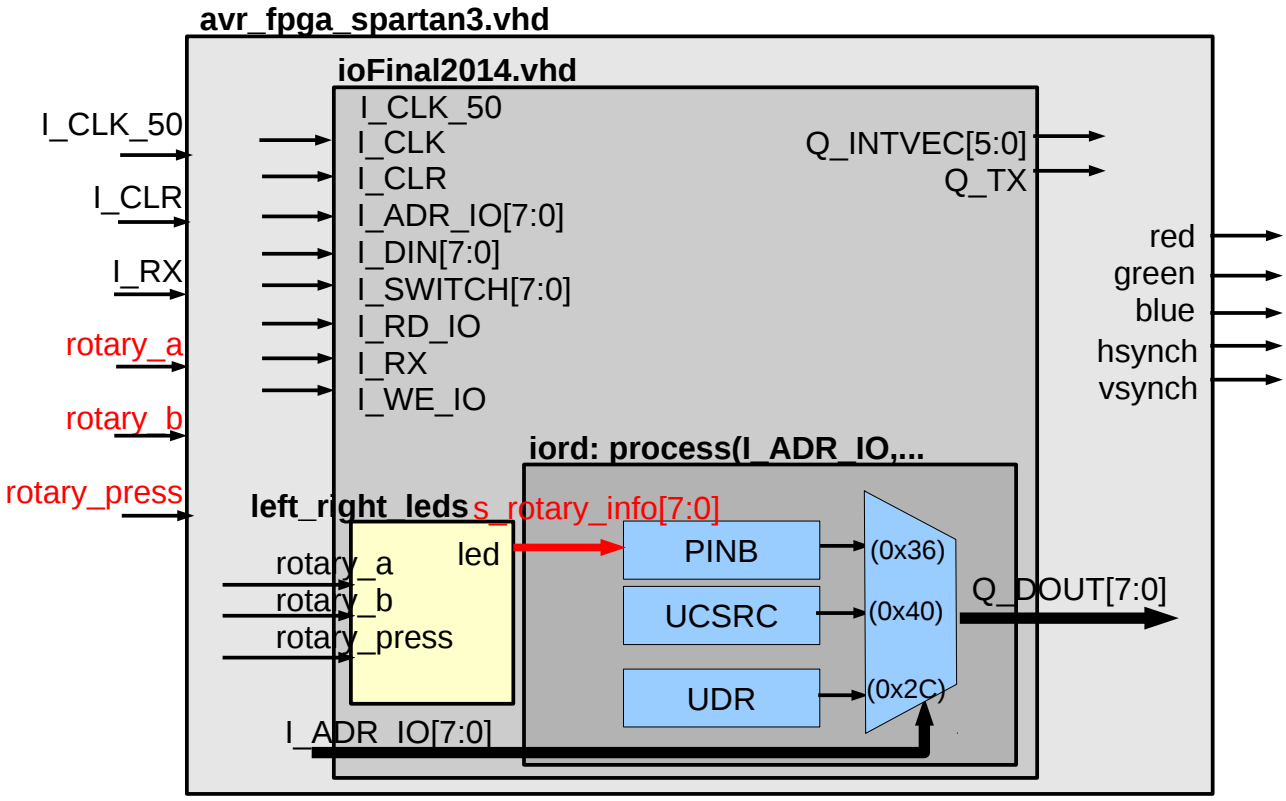
Des essais nous ont montré qu'il serait bon de réaliser une incrémentation de deux en deux.

Travail à réaliser (7 pts exo3)

3-1) Réaliser la transformation présentée. L'appui sur le bouton donnera une valeur fixée par vous.

3-2) Réaliser ensuite l'interface avec le processeur comme indiqué dans la figure ci-dessous. La valeur du 3-1) pourra donc être lue avec **PINB**.

3-3) Réaliser un sous-programme capable d'afficher en hexadécimal maintenant sur les deux afficheurs déjà utilisés en question 2 pour tester votre interface. Le chiffre affiché doit bouger en fonction de la rotation du bouton.



IV. Gestion d'une raquette à l'aide du bouton rotatif

Vous allez maintenant dessiner une raquette horizontale en bas de la partie graphique. Cela revient à faire le rectangle de la première question sauf que maintenant le processeur doit pouvoir le déplacer. Si vous avez été curieux depuis le début vous avez remarqué la présence d'un composant rectangle dans votre partie VGA. En fait il est utilisé pour dessiner une balle (carrée blanche) qui était ou pas visible mais qui ne nous intéresse pas pour la question IV). Il a aussi été utilisé pour faire la séparation entre le graphique et le texte.

Travail à réaliser (8 pts exo4)

4-1) Réaliser un rectangle rouge allongé horizontalement dont la position x peut être réalisée avec **PORTC** comme indiqué dans la figure ci-dessous. Les tailles proposées sont :

- $\Delta x = 60$

- $\Delta y = 12$

- y = 408

- x variable relié à **PORTC**

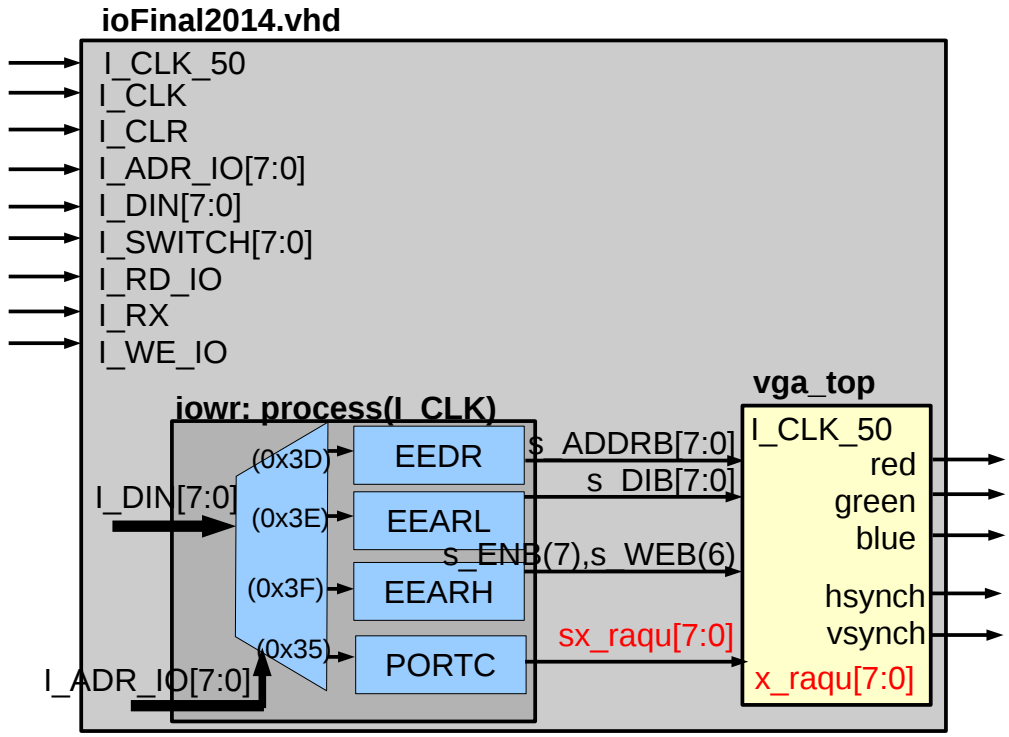
Le problème est que **PORTC** est sur 8 bits mais que la position de la raquette doit être sur 10 bits. Vous prendrez les 8 bits d'entrée du **PORTC**, donc du module VGAtop (entrée x_raqu) pour en faire en interne 10 bits en ajoutant deux 0 aux poids faibles (sous peine de ne pouvoir se déplacer sur tout l'écran). Ceci se fait très bien dans le port map du rectangle.

4-2) Modifier maintenant le programme de la question 3-3) pour qu'en plus de l'affichage on ait le déplacement réel de la raquette.

Notez sur votre feuille réponse la valeur maximale de déplacement de la raquette pour que celle-ci ne sorte pas de l'écran.

4-3) Modifier le programme 4-2 pour ne pas sortir de l'écran.

4-4) Comment pourrait-on faire pour que ce soit le module matériel "left_right_leds.vhd" qui gère les non dépassement de l'écran. On vous demande une réponse de principe (pas détaillée et complète) sur la feuille réponse en précisant bien où vous feriez la modification.



V. Gestion de l'ensemble

A partir de maintenant vous allez faire fonctionner l'ensemble raquette + balle. Dans cette partie, vous n'avez aucune réalisation matérielle à réaliser. Celle de la question IV est suffisante.

Travail à réaliser (10 pts exo5)

V-1) Lire le code de ioFinal2014.vhd donné pour essayer de voir comment on déplace la balle, quels sont les PORTs invoqués ?

Réponse sur la feuille réponse.

V-2) Écrire un sous-programme "void setXY(uint16_t x,unsigned int y);" capable de positionner la balle n'importe où sur l'écran.

V-3) Écrire un programme principal qui fait rebondir la balle sur les murs et éventuellement sur la raquette si elle est au bon endroit.