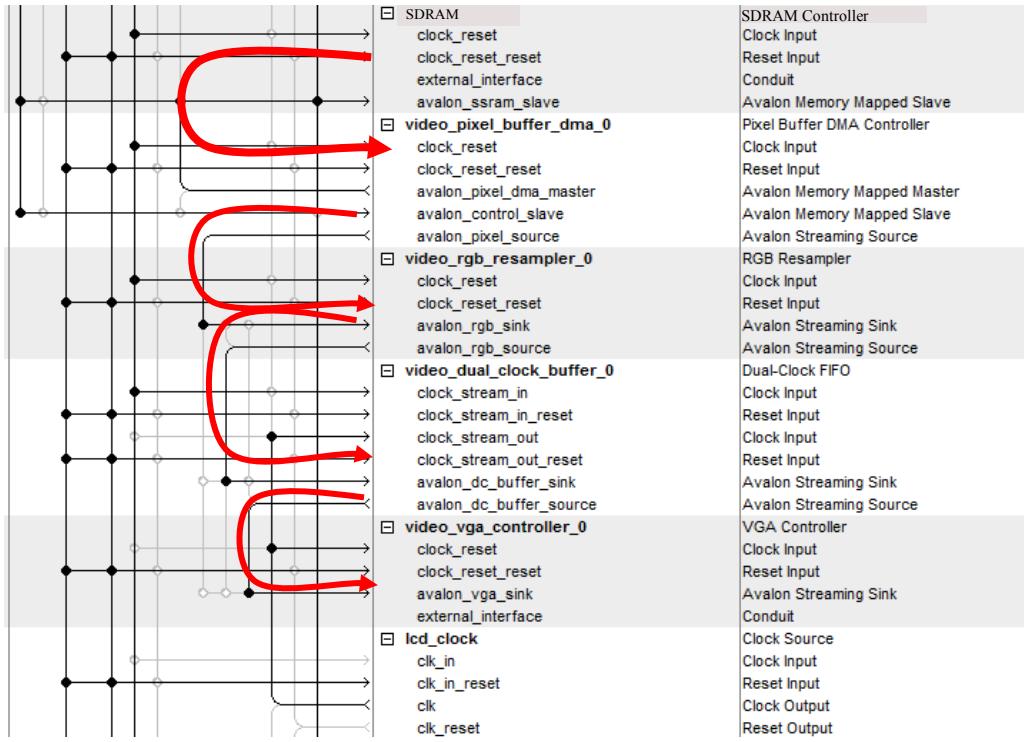
TP conception FPGA

Il vous est demandé de résoudre un problème de conception numérique, d'utiliser pour cela le langage de spécification d'architecture VHDL pour évaluer des solutions architecturales et mesurer leurs performances.

On se propose de concevoir un système d'affichage, en utilisant la carte DE1 et un écran graphique type TFT 800x480 pixels en RVB, ne disposant pas d'intelligence. Cela signifie que la mémoire d'écran n'est pas intégrée mais qu'on utilisera la mémoire de la carte FPGA. Une brique IP matérielle aura en charge de lire cette mémoire dite d'image (mémoire SDRAM implantée sur la carte DE1), de convertir le format des pixels en format compatible écran et ensuite construire des trames de données à envoyer à l'écran TFT par un protocole reposant sur de l'adressage séquentiel (remplissage pixel par pixel). On répétera cette opération suffisamment vite de sorte qu'une image soit visible par un humain (par persistance rétinienne).

En pratique, le transfert de la mémoire jusqu'à l'écran nécessite plusieurs transformations et donc plusieurs briques IPs. La première étape consiste à lire un flux continu de pixel en format 16 bits (par pixel, format couleurs RVB nombre de bits par couleur : 5-6-5). La seconde consiste à convertir le format RGB 5-6-5 d'un pixel en format compatible du TFT (à savoir que chaque couleur est codée sur 10 bits). Enfin, un buffer permet de faire le lien entre les données pixels (3 mots de 10 bits pour RVB) et le protocole des signaux de contrôle de l'écran TFT (signaux : Horloge pixel, synchronisation verticale et horizontale, bus I2C de configuration …etc). Toutes ces briques sont déjà faites dans un projet Quartus qui vous sera donné. On va se concentrer sur le remplissage de la mémoire d'image pour créer des motifs graphiques simples.

La figure ci-dessous illustre l'architecture numérique composée de briques IPs disponibles dans la bibliothèque des IPs Altera.



Mémoire Image

Transfert des 16 bits couleur du pixel vers un Buffer de pixels

Transformation : RGB 16 bits vers 3x10 bits

Buffer qui alimente le transfert des pixels vers l'écran

Interface de sortie de de synchronisation sur l'écran TFT

Horloge Pixel

On se propose de réaliser le système d'affichage d'un rectangle plein (en couleur) sur l'écran. On implantera cela sous 2 versions : une version logicielle (sur le NIOS II) et une version purement matérielle en VHDL. On utilisera les coordonnées pixels (X1,Y1) et (X2,Y2), inclus, pour réaliser le remplissage du rectangle avec une couleur RVB prédéfinie. Pour la version matérielle on pourra utiliser les interrupteurs (Switchs) de la carte pour changer la couleur.

Il est demandé d'implanter ce système d'affichage d'un rectangle plein à l'écran sous les deux variantes :

A- En utilisant le processeur NIOS-II, réaliser la fonction d'affichage en langage C sur ce processeur. Ce processeur sera connecté au bus mémoire AVALON. Cette version est dite software à l'inverse de la solution B qui sera dite hardware.

B- Implanter la brique matérielle (en VHDL) qui accède au bus d'interface AVALON (bus de communication ALTERA) pour aller remplir la mémoire d'images (SDRAM) directement.

Quelques considérations techniques :

- La mémoire SDRAM permet de stocker l'image qui sera affichée sur l'écran TFT.
- Chaque pixel est codé sur 2 octets (RVB 5:6:5) et la mémoire est adressable en octets. L'adressage de chaque pixel devra donc être pair si on veut lire un mot de 16 bits (voir détail de la brique External Bus to Avalon Bridge).
- L'organisation de l'image en mémoire est telle qu'il faut 10 bits pour indexer le numéro de colonne X (0 à 799) et 9 bits pour indexer le numéro de ligne (0 à 479).
- L'adresse de base de la mémoire SDRAM est 0x00000000 (sur un bus d'adresse AVALON 32 bits).
- L'algorithme pour remplir un rectangle en mémoire image est :

```
rect(x1, y1, x2, y2,couleur){
        entier x,y;
        for(y=y1; y<=y2; y++){
                for(x=x1;x<=x2;x++){
                        addr= concaténer Y&X&'0';
                        Mémoire[addr] ← couleur;
                }
        }
}
```

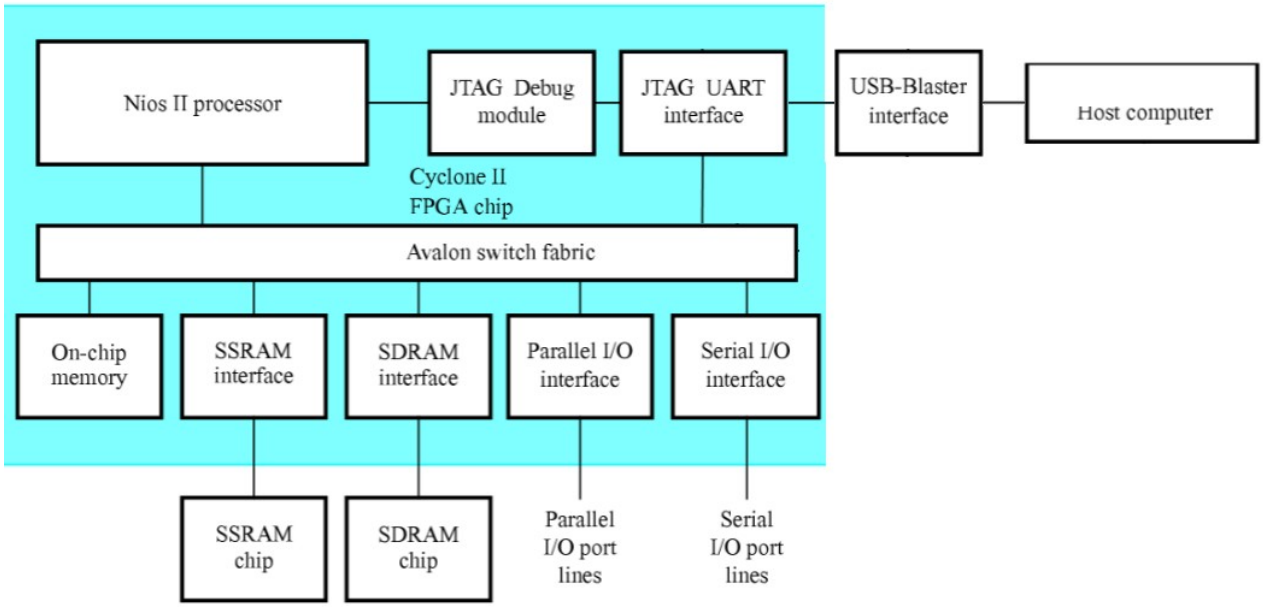
- Pour une implantation matérielle de cet algorithme, il nous faudra prévoir un signal **Start** pour lancer le système de dessin et un signal **Stop** pour indiquer que le dessin est terminé.
- L'accès à la mémoire SDRAM dispose d'un signal **ACK** qui indique que la mémoire a bien réalisé l'action (écriture ou lecture). Ce signal passe à 1 lorsque l'opération est terminée.
- L'écriture en mémoire SDRAM nécessite d'utiliser le signal **Write** et la donnée sera sur le bus **WDATA**.
- Pour nos tests, on va dessiner un carré plein de 100x100pixels, comme référence de mesure.

Il est vivement conseillé <u>d'arriver à l'heure en TP</u> et prendre le temps de lire ce document, de répondre aux premières questions (1 et 2) et en révisant quelques bases de la description VHDL. Des contrôles de connaissances seront réalisés d'une manière régulière.

<u>Implantation de la version logicielle :</u>

On utilisera pour cela un processeur virtuel IP (NIOS II) qui est sur le bus. Nous disposons également de mémoire SSRAM. Cette mémoire SSRAM sera utilisée pour héberger le programme et les données du processeur NIOS II.

La figure ci-dessous illustre l'architecture générale du système à base de processeur NIOS-II et basée sur le bus AVALON. Toute la partie délimitée en couleur sera implantée dans le FPGA. Les blocs externes sont reliés au FPGA et implantés sur la même carte (Carte DE1).



Le processeur NIOS-II a également accès à la mémoire SDRAM d'affichage de l'écran TFT à partir d'une adresse à identifier dans votre projet.

1) Ecrire le programme en C pour dessiner un rectangle plein. On respectera le prototype de la fonction indiquée précédemment. On choisira un rectangle plein de 100x100pixels. Le rectangle sera dessiné à partir des coordonnées des 2 coins diagonaux (x1,y1) (x2,y2), qui seront inclus.

On dispose d'un Timer dans l'architecture du système, interfacé au processeur NIOS, et qui nous permettra de mesurer le temps de calcul de la fonction de dessin du rectangle.

Les fonctions d'accès à ce TIMER sont celles utilisées en TP CNM en S7.

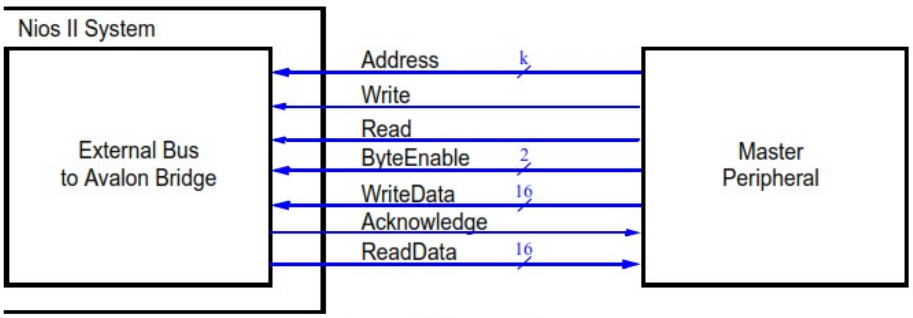
2) Mesurer le temps de calcul (en cycle d'horloge) et déduire le CPP (nombre de Cycle Par Pixel) pour cette implantation logicielle.

On va ensuite transformer l'algorithme de base pour arriver à optimiser cette fonction de dessin d'un rectangle plein. Identifiez les possibilités d'optimisations et de réduction de la complexité de cette fonction.

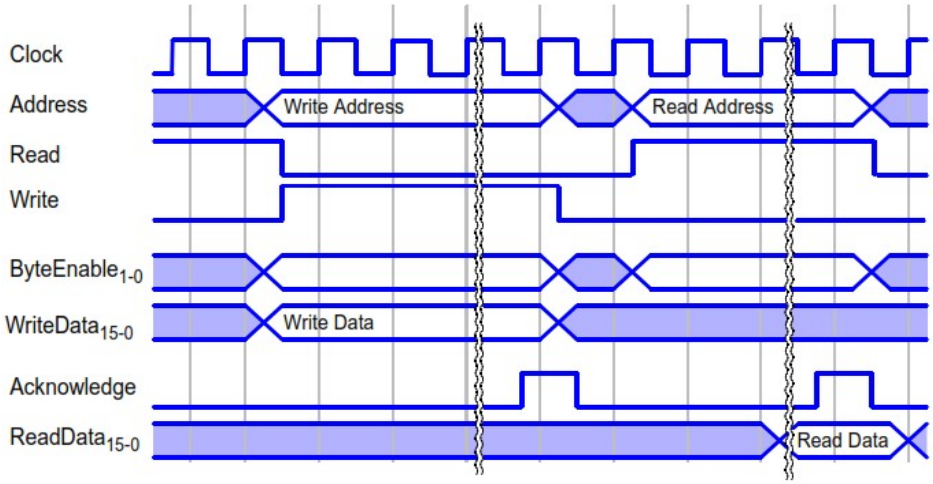
3) Explorer les différentes possibilités de modifications de cet algorithme « dessin d'un rectangle » pour l'optimiser, donc réduire le CPP.

<u>Implantation matérielle de l'Architecture de calcul pour afficher un rectangle.</u>

Pour cela, il faudra s'interfacer au bus du système qui est relié à la mémoire SDRAM et au contrôleur de l'afficheur. La figure ci-dessous illustre la brique d'interfaçage qu'on utilisera pour se connecter au bus d'interface AVALON auquel est reliée la mémoire SDRAM.



(a) External bus signals



(b) External bus timing diagram

La brique que nous devrons réaliser sera vue comme un maître sur le bus et nous utiliserons le Pont-Master MM-Bridge pour s'interfacer. Ce Pont aura un comportement similaire à un bus de processeur pour avoir accès à la mémoire d'image. On activera alors le signal **write** et on attendra le retour du signal ACK (Acknowledge) indiquant que l'opération s'est bien passée. Ainsi, en balayant en ligne et en colonne la mémoire d'image, on affichera le rectangle en respectant la couleur.
Lors des séances de TPs, des explications seront données pour vous aider à mieux répondre aux questions posées. Un modèle de projet Quartus vous sera également donné pour vous éviter tous les détails techniques inutiles au problème de conception d'architectures, but de nos TP.

4) Proposez une modification de l'algorithme de base pour y inclure toutes les recommandations techniques, en prenant en compte le signal de retour ACK de la mémoire SDRAM.
5) Découpez l'algorithme sous forme d'actions à réaliser de manière séquentielle.
6) Proposez une implantation comportementale en VHDL de cet algorithme en considérant le chemin directeur comme étant l'automate et on réalisera les actions à faire dans chacun des états.
7) Pour mesurer le temps d'affichage du rectangle, on utilisera un compteur qui démarrera au lancement du dessin et s'arrêtera à la fin. Utilisez les signaux Start et Stop pour cela.
8) Déduire le nombre de cycles d'horloge par pixel affiché, noté CPP. Cette métrique nous permettra de faire une étude comparative par la suite.

Note importante concernant la conception numérique sur FPGA : *Dans une description sur papier d'une architecture de chemin de données, il est légitime de favoriser la description d'architecture à base de bus de données et donc des buffers 3 états. Toutefois, lors de l'implantation de cette architecture sur FPGA, il est déconseillé d'utiliser des buffers 3 états et des bus bidirectionnels si ces derniers restent confinés dans le FPGA (sans en sortir sur les plots E/S).*
En effet, dans un FPGA, la notion de bus bidirectionnels est très pauvre. Il en est de même pour les Buffers 3 états. Ces concepts sont cependant très fortement représentés sur les plots d'entrées/sorties (tous les plots E/S sont configurables et potentiellement bidirectionnels).
Par conséquent, un bus bidirectionnel sera mis en œuvre sous forme de 2 bus unidirectionnels (chacun dans un sens). Des composants qui sortent leurs données sur un bus verront leurs bus de sorties multiplexés pour construire ce bus de sortie.

Implantation matérielle/logicielle dites MIXTE.

On se propose maintenant de modifier la solution logicielle pour l'interfacer à la solution matérielle. Ainsi, la brique d'affichage matérielle sera considérée comme un coprocesseur graphique par rapport au processeur NIOS.
Il suffira pour cela de :

- ajouter des périphériques type Ports de sorties pour modifier X1,Y1 et X2,Y2 avec la couleur. Il vous faudra pour cela 2 ports de sortie.
- prévoir un bit de commande pour lancer le calcul, en reproduisant le signal Start par logiciel.
- prévoir un registre de retour d'état pour lire le signal Stop.

On ajoutera pour cela des ports d'entrées/sorties, permettant au processeur d'écrire les coordonnées du rectangle (X1,Y1;X2,Y2) ainsi que la couleur. Ces informations seront écrites par le processeur (par logiciel) et accessibles par la brique d'affichage hardware du rectangle.

9) Proposer une implantation matérielle pour interfacer la brique matérielle de dessin du rectangle au processeur NIOS-II. Attention, il faudra prévoir l'attente de fin du calcul grâce au signal Stop.
10) Mesurer le temps de calcul ainsi que le CPP. Conclure.

Il sera demandé un seul compte rendu à la fin de toutes les séances de TP. IL faudra y inclure tous les résultats des études.