TP de programmation DSP pour le traitement d'image ISBS2/ST10 2005-2006

Introduction

1.1 Objectifs

Le TP sur carte IEKC6x11 s'étend sur une séance de 4 heures. Vous aurez l'occasion durant ces séances d'expérimenter :

- l'utilisation d'une carte DSP pour le traitement d'image basée sur le processeur Texas Instruments TMS320C6211 ou TMS320C6711,
- de faire de l'acquisition d'images vidéo
- d'implanter sur cette plate-forme différents algorithmes de traitement d'images,

ATTENTION **NE JAMAIS CONNECTER OU DÉCONNECTER** L'ALIMENTATION OU LE CONNECTEUR JTAG OU FAIRE UN RESET DE LA CARTE TANT QUE CODE COMPOSER EST LANCE : IL Y A RISQUE DE **DESTRUCTION DU DSP**
AVANT CES MANIPULATIONS VOUS DEVEZ TOUJOURS VERIFIER DANS LE **GESTIONNAIRE DE PROGRAMME** QUE CCS_APP.EXE N'EST PAS EN COURS D'EXECUTION
(le raccourcis KILL_CCS vous permet d'ouvrir un gestionnaire de programme)

1.2 Description de la plate-forme d'expérimentation IEK6711

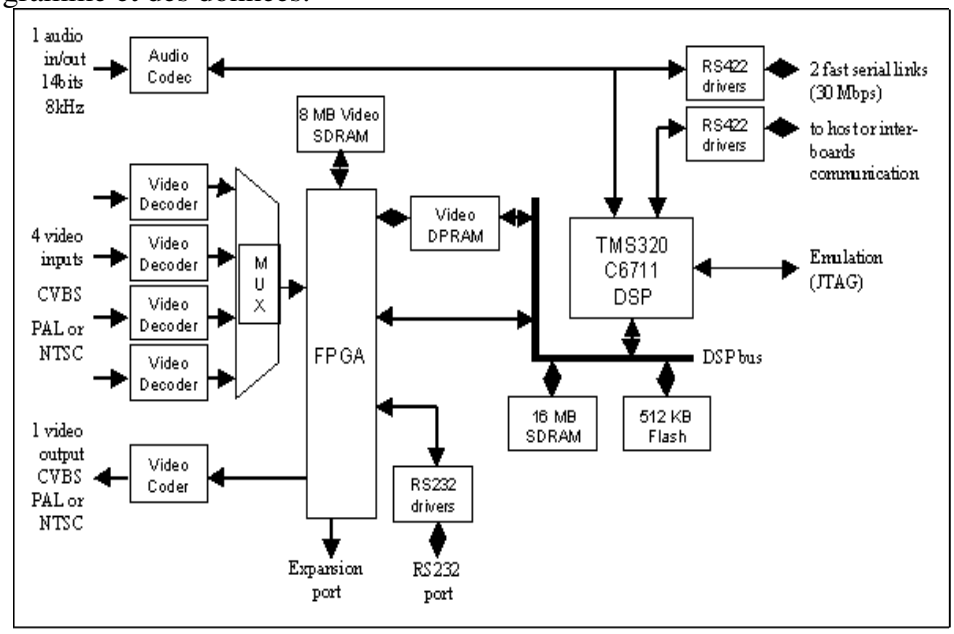
La carte IEKC6711 peut être connectée à 4 sources vidéo analogiques en entrées et possède une sortie vidéo analogique. Elle repose sur un DSP Texas Instruments TMS320C6711, un FPGA 50000 cellules et plusieurs bancs mémoires RAM (Cf. figure ci-dessous).

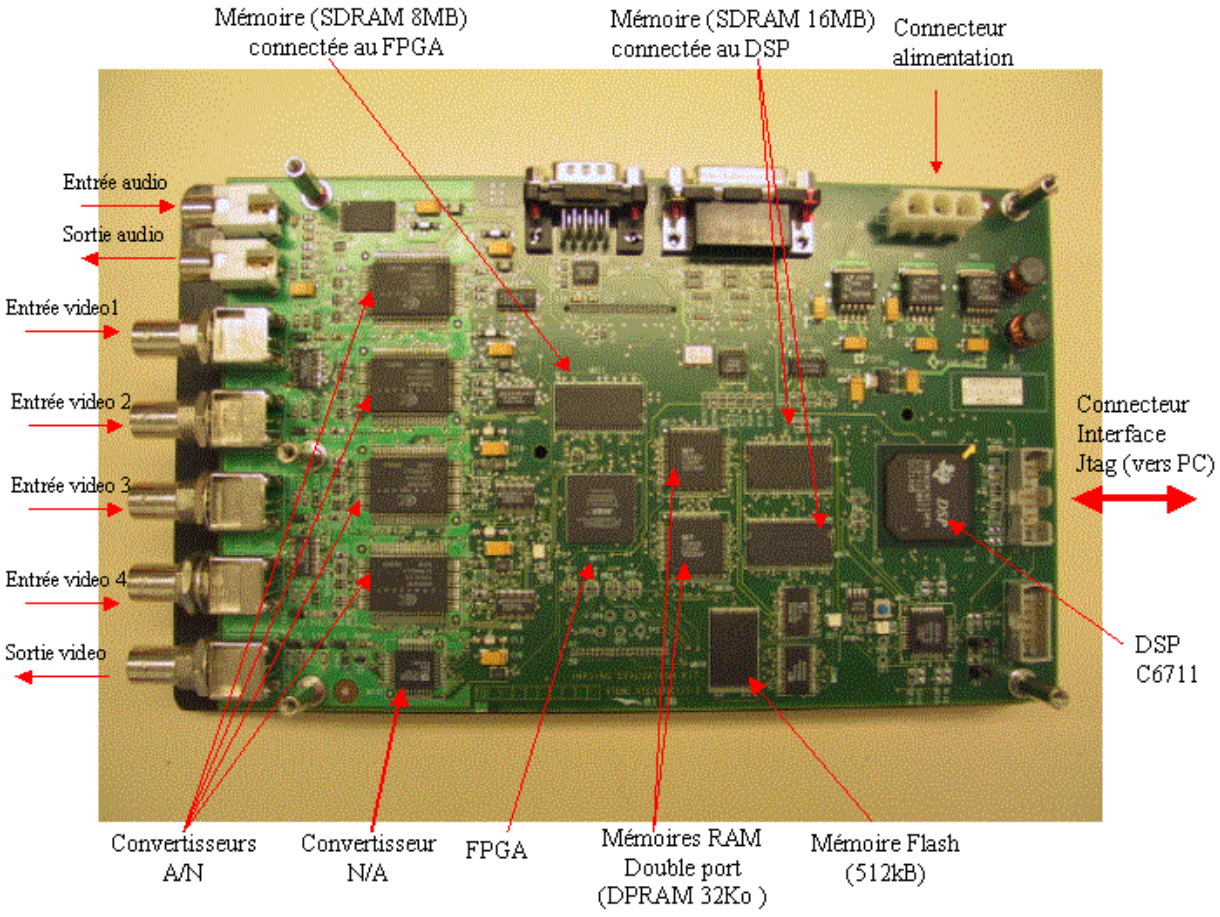
Cette carte peut fonctionner, selon la configuration du FPGA, soit en acquisition d'images, soit en restitution .

En mode acquisition le signal d'une des 4 entrées analogiques est convertit en un flot numérique par le convertisseur A/N correspondant. Ce flot est ensuite stocké dans la RAM du FPGA par le FPGA. Une fois qu'une image entière est stockée dans la RAM FPGA il faut la transférer dans celle du DSP pour qu'il puisse y faire des traitements (filtrage, compression, etc). Pour cela le FPGA et le DSP sont connectés ensembles par une mémoire RAM double ports (DPRAM) de 16Ko. Cette taille étant insuffisante pour transférer une image entière en une fois ce sont des blocs de 8 lignes qui sont échangés. Côté DSP, on utilise plutôt le DMA pour effectuer rapidement les copies des blocs de 8 lignes vers la RAM du DSP sans gaspiller de puissance CPU.

En mode restitution, le processus est totalement symétrique : l'image à afficher est supposée d'abord contenue dans la RAM du DSP ou sa RAM exterieure. Il s'agit donc de la découper en blocs de 8 lignes pour la transférer dans la RAM du FPGA par le biais de la petite DPRAM. Quand une image entière a été transférée, le FPGA la transmet à l'unique convertisseur numérique-analogique qui est lui même connecté au moniteur vidéo.

Comme tous les processeurs récents, le DSP est équipé d'une **interface JTAG** qui permet, depuis Code Composer Studio exécuté sur un PC, de contrôler très précisément le DSP : mis en place de points d'arrêts, lecture-écriture des registres internes, accès en lecture-écriture à tous l'espace mémoire. Cette dernière fonctionnalité sera d'ailleurs utilisée pour charger la mémoire du DSP avec un programme et des données.





1.3 Préparation :

Avant de commencer il faut récupérer le fichier compressé TP_ISBS2_DSP_2004-05.zip que vous trouverez sur http://www.esiee.fr/~grandpit/ pour le copier et le décompresser dans le répertoire c:\temp de votre machine. Vous n'oublierez pas de nettoyer ce répertoire à la fin de chaque TP après en avoir effectué une sauvegarde sur votre compte.

Note : il n'est pas possible à code composer d'accéder directement à votre compte c'est pourquoi il faut toujours travailler sur le répertoire local temp.

Une fois décompresser vous obtiendrez les répertoires suivants :

- capture : ce répertoire contient le projet "TP-ISBS" (faire menu "Project"→"open") permettant de faire l'acquisition d'images puis l'affichage dans une boucle infini

- Test : ce répertoire contient des fichiers binaires déjà compilés (fichiers "affiche-image.out", "affiche-vidéo.out") afin de tester rapidement la carte quand vous avez un

doute sur son fonctionnement. Il suffit, après avoir correctement ré-initialisé la carte si nécessaire, de charger (menu "File" → "Load Program") puis exécuter ces programmes (menu " Project" → "run") en ayant préalablement connecté et allumé le téléviseur qui doit être sur le mode A/V).

- MaintenanceIEK67 : ne pas exécuter les programmes contenu dans ce répertoire. Ils servent principalement à configurer le FPGA de la carte en mode acquisition ou restitution.

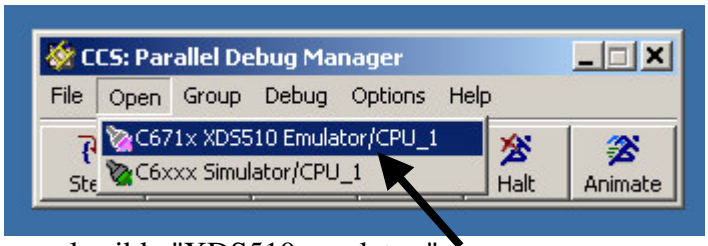
- docs : documents techniques sur la programmation du DSP C6x et de l'EDMA.

2 Traitement d'images

2.1 Chargement du projet

1. Lancer CCS. Pour s'exécuter CCS requière des privilèges spéciaux, il s'exécute donc sous les droits d'un utilisateur différents : **login=ccs, password=ccs**

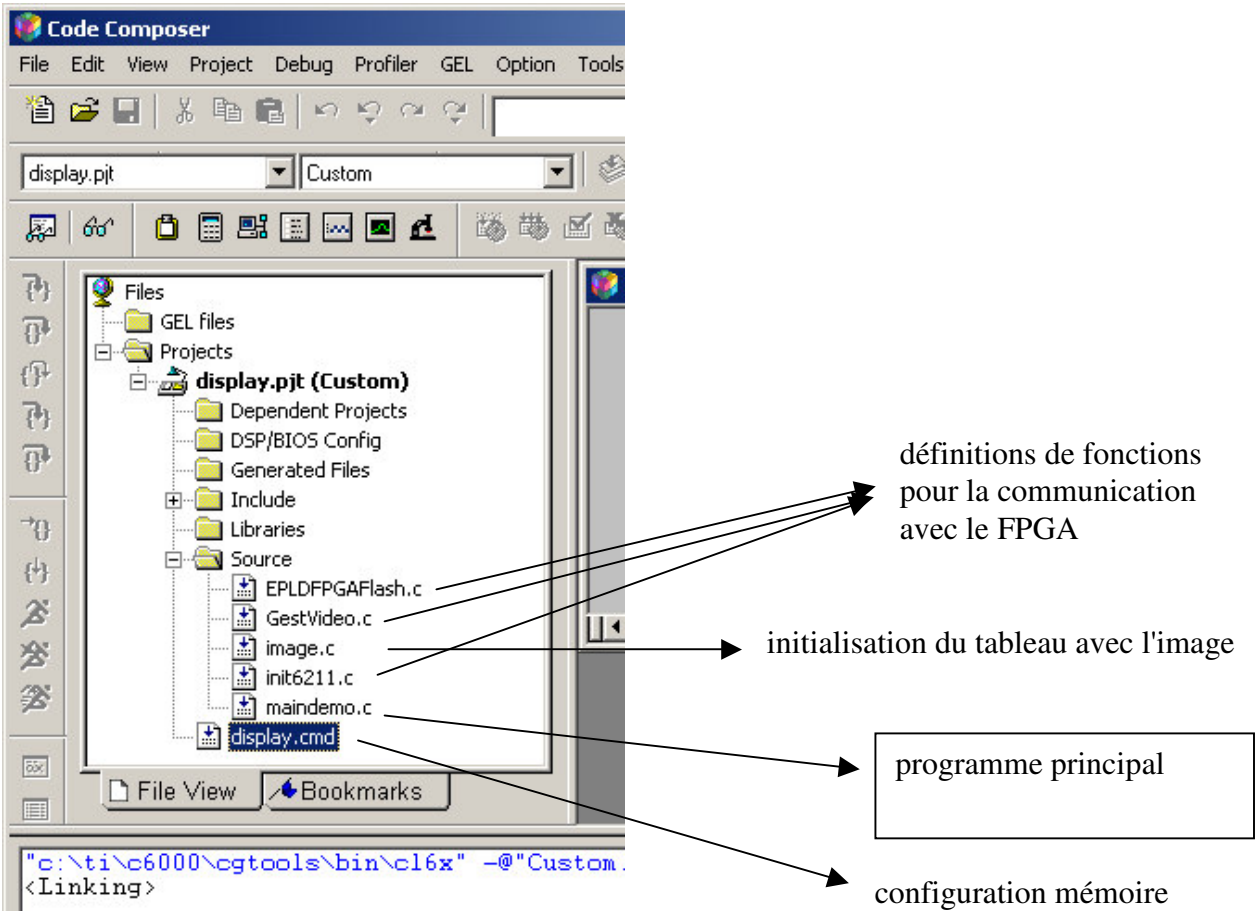
2. Si le logiciel ouvre la fenêtre suivante :



il faut alors sélectionner la cible "XDS510 emulator "

3. Charger le projet "display" se trouvant dans le répertoire ``c:/temp/affiche_image/display.pjt''

Ce projet contient les fichiers suivants :

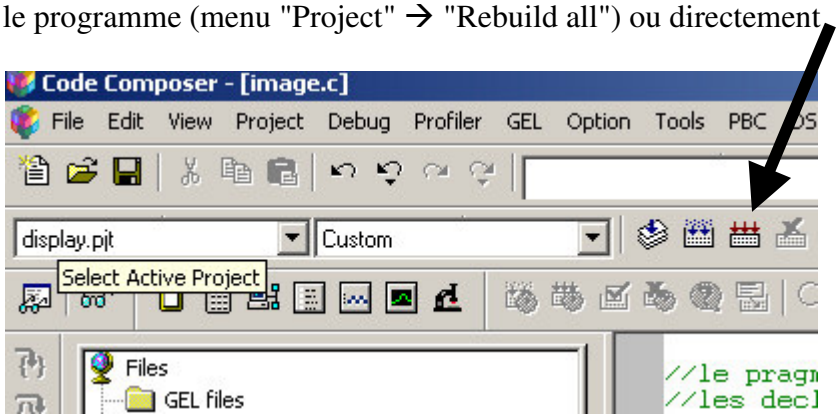


Dans le cadre de ce TP nous nous intéresserons uniquement au fichier maindemo.c
Le cœur de ce module est la fonction C "main( )" (ouvrir le source du fichier maindemo.c à l'aide du navigateur dans la partie de gauche de Code Composer Studio)

Remarque : pour ce TP, nous n'utiliserons qu'une sous-partie des fonctionnalités de la carte IEKC6211 : images CIF (352x288) en niveau de gris. La carte est toutefois capable de travailler aussi en plein format vidéo (720x576) et en couleur (YUV422).

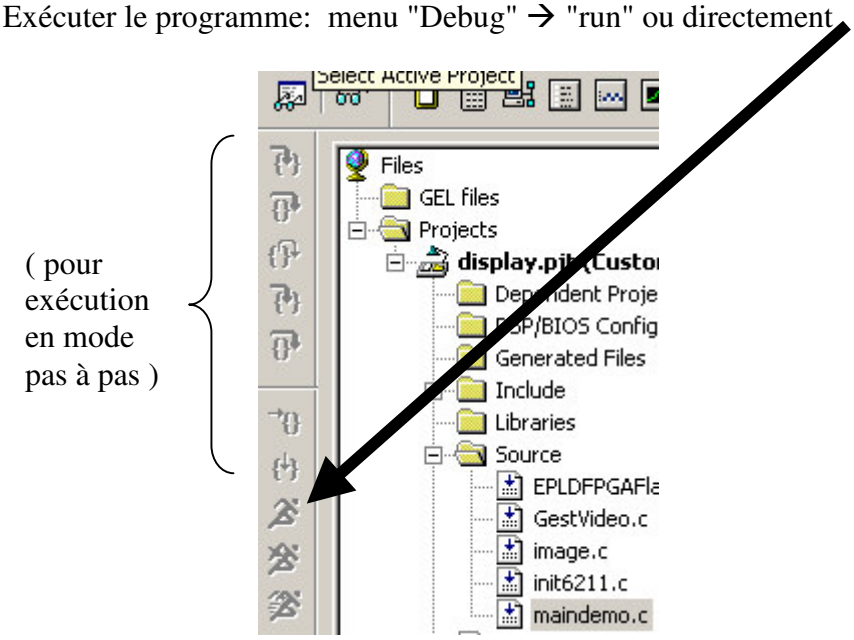
2.2 Compilation, exécution :

1. Recompiler le programme (menu "Project" → "Rebuild all") ou directement



2. Charger le programme : menu "File"→"load program"

3. Exécuter le programme: menu "Debug" → "run" ou directement



Il doit s'ensuivre l'affichage des méssages suivants dans CCS :

- Load FPGA
- Load FPGA done
- Affichage de l'image

2.3 Traitement d'images

2.3.1 Dessiner un carré

Il s'agit d'écrire 2 boucles imbriquées qui modifient la valeur des pixels de façon à dessiner un carré au centre de l'image

2.3.2 Négatif :

Il s'agit d'écrire un code qui compare chaque pixel à la valeur du *seuil* et qui remplace la valeur du pixel par 0 si sa valeur est inférieure au seuil ou par 255 si supérieure ou égale. Les arguments *image_source* et *image_destination* sont les adresses des images avant et après transformations. Les pixels sont représentés par des entiers codés sur un octet et donc compris entre 0 et 255.

2.3.3 Seuillage :

Il s'agit d'écrire un qui "inverse" la valeur des pixels de *l'image_source* de façon à avoir une image en négatif dans *image_destination*.

2.3.4 Convolutions

Il s'agit d'implanter une convolution 3x3.
Rappel : la convolution d'une image source S de taille NxN avec une matrice MxM consiste à calculer une nouvelle image D dont la valeur de chaque pixel Dij est une fonction de la valeur des pixels voisins du pixel Sij.