

# INTERVIEW DES ENCADRANTS, DÉFINITION DU TRAVAIL A ACCOMPLIR

---

## Modalités

---

**Date** : le Jeudi 17/09/2020 à 13h30

**Personnes présentes** : Jean-Noël Clink, Hugo Questroy, Alexandre Froehlich (Product Owner), Guillaume Leinen (Scrum Master), Jean-Christophe Le Lann, Pascal Cotret

## Objectif de la séance

---

- Faire la connaissance des encadrants du projet
- Définir la logistique du projet (salle/matériel/disponibilités)
- Définir les points marquant du projet
- Éclaircir la théorie sur ces points

## Compte-rendu de l'échange

---

### premier point : le matériel et les salles de travail

Nous aurons à notre disposition du matériel prêté par l'école, notamment des cartes FPGA (Field Programmable Gate Array) Nexys A7. Ces cartes seront suffisantes pour notre projet, et dispose de tout la connectique nécessaire, notamment un port réseau.

Cependant, il faudra acheter les cameras qui s'occuperont de la capture de la vidéo à streamer. Les cameras en question sont des cameras CSI, de référence Omnivision 7670 par exemple. Ce sont des petites cameras de définition 720p qui se connectent à la carte via un connecteur p-mode.

Concernant la salle, les encadrant se chargeront de faire en sorte de pouvoir accéder à une salle info du bâtiment N, qui est normalement fermé par badge. cette salle disposent en effet de tout le matériel nécessaire, et les encadrants préféreraient que les cartes FPGA ne sorte pas trop du labo.

*NB: nous pouvons avoir accès à la salle, de même qu'un autre groupe*

### Second point : La base théorique sur lequel il va falloir se documenter

Le but de ce projet est de fabriquer un algorithme de compression vidéo **Open-source** et libre de droits. Il sera donc bien question de regarder les différentes techniques utilisés par les codec actuel type MPEG4 (MP4), mais **de les adapter à nos problématiques et notre implémentation**. Pour cela, il serait bien d'éviter d'utiliser des bibliothèques externes, car elles

engendre un effet **boite noire** qui rendent la compréhension et la maîtrise du code inexacte. Il sera important donc de maîtriser la théorie des différentes méthodes de compression.

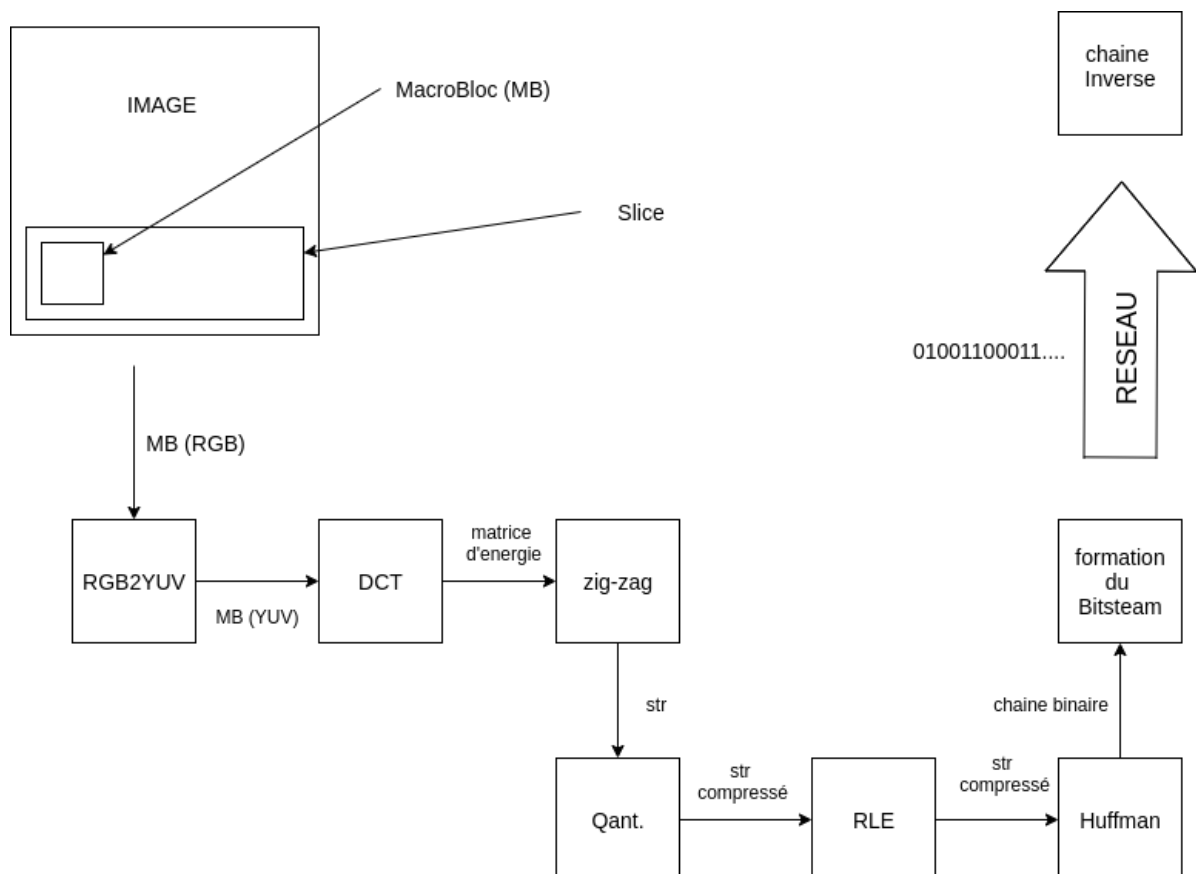
### ***Justement, quel sont les différents blocs-théoriques à maîtriser ?***

- Le premier traitement consistera en la conversion d'une source RGB en la séparation en 2 canaux luminance/chrominance et selon le format YUV ( ou YCbCr).

*état de réalisation actuel : algorithme fonctionnel mais problème de balance des couleurs (décalage vers le rouge), possiblement la cause des coefficients dans la matrice de conversion.*

- Le second traitement consiste en une transformation de type DCT (Direct Cosinus Transform). Il faudra pour gagner en optimisation coder nous même la fonction (ça devrait être relativement).
- Ensuite, le "zig-zag" scan, qui consiste à parcourir une matrice de manière diagonale afin de regrouper des zéros consécutifs (générés par la DCT). Implémentation facile à priori.
- Un bloc de quantification pour regrouper plusieurs niveau de contraste peu différent en un seul afin de compresser encore plus.
- une compression RLE (Run Line encoding) permet de regrouper les données de même niveau avec leur fréquence. cela forme une paire (valeur, fréquence)
- l'encodage de huffmann permet de générer un arbre binaire a partir de ces tuples pour gagner encore en compression.

*état de réalisation actuel: algorithme de huffmann fonctionnel*



### **Diagramme de blocs succinct du traitement à réaliser**

## ***il y a t'il d'autres points d'attentions face à la conception d'un tel algorithme ?***

Il va falloir se pencher **très sérieusement** sur les images de test de notre algorithme. La conception d'une image testant les limites/extrêmes du code est quelque chose de très important. Se renseigner donc sur la création d'images synthétiques...

## **Troisième point : Devront nous maîtriser un langage de description matériel (HDL) ou existe t'il un moyen de Traduire du code haut niveau type langage C en HDL ?**

Il existe effectivement un moyen mais non complet, il s'agit des outils HLS (high Level Synthesis). Cependant ces outils ne peuvent que traduire des briques de bases, et non pas un code dans son intégralité. Il faudra donc gérer des signaux entrées-sorties et des conversion directement en bas-niveau. Il faut cependant ne pas trop en abusé car ils sont les mêmes inconvénients que l'utilisation des bibliothèques dans des langages de plus haut-niveau, c'est à dire qu'il rendent les codes '*opaques*'.

L'autre inconvénient est un recours fréquent à des ressources **payantes** nécessaires à l'utilisation de certaines fonctions comme le réseau ou un lien PCI-express. Une des approches face à ce problème consiste à comprendre la logique de la ressource grâce à la documentation et de la copier nous même.

*NB: après en avoir discuté avec un IETA de DGA techniques terrestres passionné de FPGA, ce point risque d'etre plutot compliqué à gérer sur les cartes de fabricant xilinx, les docu faisant parfois 800 pages ou plus....*

## **Dernier point : Les livrables attendus par les encadrants**

Le plus important sera la conceptions d'un ou plusieurs schémas-blocs très documentés et élaborés. l'idéal serait de faire des diagrammes SVG sur une application comme **Inkspace** avec représentation des flux, des bandes-passantes, des traitements, etc... c'est appelé du free-modeling.