Rapport Sprint 8

Issues terminées pour le sprint

Jean-Noël : gestion d'un socket en C [finished]

Difficultés rencontrées

Faire marcher le socket sur Linux ET sur Windows simultanément (gestion différente par l'OS). Beaucoup de templates de code disponibles sur internet mais pas adapté à notre projet. Phase de conversion et d'adaptation qui a été nécessaires. Extrêmement chronophage.

Fonctionnalité

Permet d'envoyer un message d'un client et à un serveur en c via l'appel Scanf, et ce peut importe l'OS utilisé.

Hussein / Alexandre: Support des images pour la DCT

Difficultés rencontrés

Pas encore de tests unitaires. Difficulté dans la manipulation des types non natifs, ainsi que dans l'entente sur l'implémentation à adopter (calcul sur canaux séparés ou tout simultané, taille de matrice, etc...).

Fonctionnalité
Permet la transformation DCT d'une matrice de taille N, ainsi que la DCT inverse. Elle intervient dans le chaine de compression.

A ajouter
Permettre un calcul préalable des coefficients de la DCT avant les transformations afin de gagner du temps (matrice constante).

Alexandre : Dévellopement des autres fonctions métiers de EVEEX en c

Difficultés rencontrés

Fuites de mémoires très chronophages, code très complexe.

Fonctionnalité
Tout les blocs de traitement de EVEEX sont implémentés : encodage Huffman, zigzag, quantization, RLE, découpage macroblocs. Seuls certains liens entre ces fonctions métiers sont encore inexistants.

A ajouter
Les liens restants, cependant le code devient incontrolable et la motivation/ probabilité sur la possibilité de finir l'encodage est de plus en plus basse. De plus il n'est pas sur qu'une implémentation orienté objet du c soit réellement plus performante qu'une implémentation dans un language de plus haut niveau.

Alexandre / Guillaume: Réaliser une cross compilation RiscV de Eveex-C

Difficultés rencontrés
Construction/syntaxe du Makefile.

Fonctionnalité
Permet de compiler EVEEX sur une architecture riscV directement depuis nos ordinateurs en x86 (cross-compilation) sans segfault. Un logger a été développé pour avoir un retour sur les bugs sur fpga.

Hugo : décodeur EVEEX python fonctionnel sur la vidéo [Non fonctionnel]

Difficultés rencontrés
Bitstream a casser pour récupérer uniquement les trames de données. Decodage de la première frame fonctionnel mais problème pour savoir quand s'arrêter à la frame suivante. Huffman est intégré au bitstream mais il est necessaire de le sortir afin de décoder la vidéo. Package Eveex interférant avec le prototype quand les 2 sont présents sur une même machine.

Fonctionnalité
Décodage d'une frame. Nous n'allons pas poursuivre sur ce point étant donné l'avancement dans le projet en C.

Guillaume : Faire fonctionner un SOC VexriscV avec un kernel Buildroot sur la carte Nexys4ddr

Difficultés rencontrés
Comprendre la composition d'une image buildroot (packet linux, packet opensbi, etc...). Ceci fait, la source majeure de problème a été de trouver un moyen d'importer mon software sur une image buildroot. Nous sommes obligés de l'importer sur la partition de boot car nous n'avons pas pu faire marcher l'accès à distance (SSH, WGET).

Fonctionnalité
Permet l'implémentation d'un noyau linux optimisé pour notre implémentation matériel. Cet OS fonctionne sur un CPU RiscV différent du précédent. Nous allons maintenant pouvoir ajouter des packets maison a ce noyau linux.

Suite de l'activité pour sprint 9

Comme écrit dans le rapport de mi-projet, le projet a été réorienté. Les issues ont donc totalement changés par rapport à l'estimation d'il y a 2 semaines.

- premierement, réaliser l'encodeur EVEEX entierement en GO => quasi terminé le jour même.
- ensuite, tout le monde se documente et réalise au moins un "helloworld" en go afin de se familiariser avec le language.
- Se procurer des cartes raspberry Pi (2 cartes en notre possesion) pour Jean-Noêl nottament, et se familiariser avec raspbian voir Buildroot

Concernant le FPGA :

- chercher à calculer un cosinus en VHDL par l'utilisation de tables LUT (ce sera très utile pour la DCT). le but est de pouvoir comparer l'execution de Huffman en VHDL ou en c sur RiscV ou

en go sur ARM.

- refaire un "Helloworld" sur l'utilisation de la HLS Vivado ou d'autres HLS et l'empacketage du résultat en IP.