INF8500 Systèmes embarqués : conception et vérification

Laboratoire # 2 :

Synthèse haut niveau

Automne 2016

1 Objectifs

L'objectif de ce second laboratoire est d'introduire la synthèse haut niveau à l'aide du logiciel **Vivado HLS** de **Xilinx**. Le logiciel **Vivado HLS** est spécifiquement fait pour les composants FPGA de **Xilins** tels que l'**Arix7**, **Kintex 7** et **Virtex 7**.

La synthèse haut niveau permet de grandement accélérer le développement de circuit numérique. Cette approche permet la traduction d'algorithme de haut niveau décrite en *C/C++/SystemC* vers une description RTL (*VHDL*, *Verilog* ou *SystemC*). Cette version RTL peut alors être utilisée afin de produire de la logique numérique exécutant le même traitement que l'algorithme initial. Ce circuit permet alors d'accélérer ce traitement par la création de matériel (ASIC). Il s'agit d'une étape importante dans l'approche codesign qui sera vue au prochain laboratoire.

Pour ce laboratoire vous devrez :

1. Vous familiariser avec le concept de la synthèse haut niveau à l'aide du logiciel **Vivado HLS** de **Xilinx**
2. Comprendre l'algorithme *SystemC* d'un module de transformée inverse en cosinus discret (IDCT).
3. Générer diverses solutions du même coprocesseur selon divers requis.

La totalité de ce travail pratique s'échelonnera sur une séance (trois semaines étant donné la semaine de relâche). La première partie sera consacrée aux tutoriels alors que la seconde sera consacrée à l'étude de cas du CoProcesseur IDCT.

2 Contexte

On utilise aussi le terme synthèse algorithmique pour désigner ce genre d'outil. Ils sont issus d'un besoin de combler le déficit de productivité des concepteurs de système dans un marché très compétitif. Ce type d'outil permet d'alléger le flux de conception et améliorer grandement le « time to market » d'un nouveau produit. Ces outils sont également bénéfiques en termes de partitionnement logiciel / matériel, comme ils utilisent un langage commun, tel le langage C, pour définir les éléments d'un système.

De plus, l'automatisation présente dans la synthèse de haut niveau permet à un concepteur de produire rapidement des versions alternatives d'un modèle donné. Par exemple, une version pourrait implémenter différents niveaux de parallélisme, un autre différent protocole de lecture mémoire et un autre évaluant l'avantage du pipelinage. Cela permet d'explorer l'espace de conception afin de déterminer la meilleure solution selon les compromis préalablement identifiés tel que l'espace, la consommation et la rapidité de calcul.

Il faut par contre garder en tête que les designs dans le contexte de la synthèse haut niveau ont pour but l'implémentation vers la logique programmable. Il est donc primordial d'analyser les solutions dans cette optique, car ceci est différent d'un code compilé pouvant être exécuté sur un processeur tel que **ARM** ou **Intel** x86.

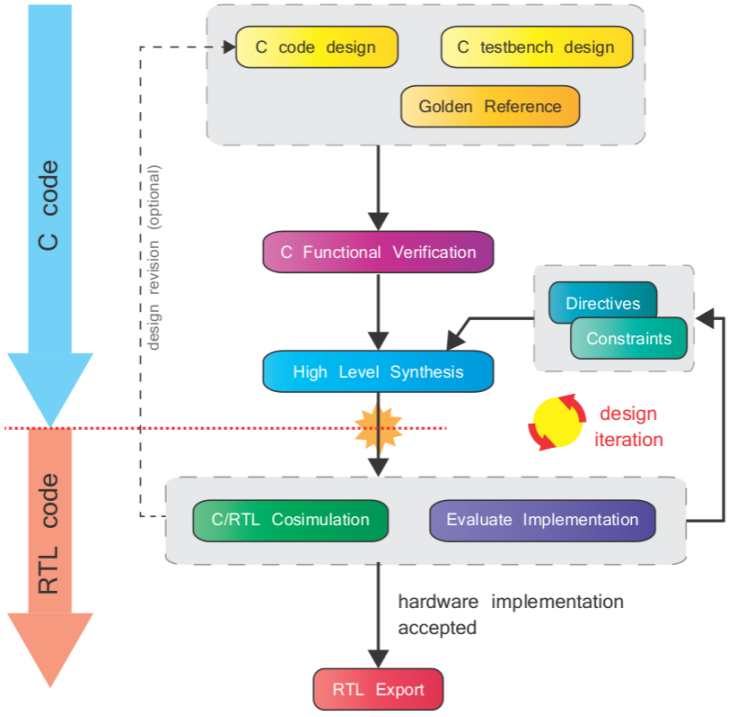


Figure 1 - Flow de travail Vivado HLS

3 Tutoriel HLS

Le but de cette partie est de vous familiariser avec le logiciel **Vivado HLS** ainsi que les concepts de la synthèse haut niveau. L'énoncé de cette partie se trouve dans le dossier tutoriel dans l'archive fourni avec cet énoncé. Suivez attentivement le tutoriel de la page **1 à 57**. Arrêtez-vous après la vérification RTL.

Pour ce laboratoire, nous allons utiliser la version **2012.4** de **Vivado HLS**. Afin de lancer l'application :

- Aller dans c:\Logiciels\Xilinx\Vivado_HLS\2012.4\Win_x86\bin\
- Démarrer Vivado HLS en double cliquant sur le programme **vivado_hls_bin.exe**.

Vous pouvez vous créer un raccourci pour cet exécutable en cliquant à l'aide bouton de droite de la souris sur l'exécutable et choisir "créer un raccourci".

4 Étude de cas HLS : Module IDCT

Le but de cette partie est d'analyser un module développé en *SystemC* afin de l'instrumenter correctement à l'aide de **Vivado HLS** pour en faire un bloc matériel optimisé selon une suite de requis.

L'étude de cas se fera sur la transformée en cosinus discret inverse.

4.1 Traitement d'image

Le traitement d'image est une forme de traitement de signal numérique où l'entrée est une image. Le traitement a pour but de modifier, améliorer ou extraire de l'information de celle-ci. Par exemple, il est possible d'augmenter les détails d'une radiographie en temps réel ou bien extraire de l'information afin de fabriquer une liste de mots clés associés aux formes ou objets se trouvant à l'intérieur d'une image.

En général, on retrouve un niveau de traitement d'image sur une majeure partie des soutiens visuel dans le domaine informatique. Que ce soit pour les images (*jpeg*, *png*, etc.) ou la vidéo (*MJPEG*, *MPEG-1/2*, *H.264*, etc.). Ce type d'opération est donc courant pour les systèmes embarqués utilisés dans le domaine du multimédia. Juste à penser aux téléphones intelligents, les systèmes de surveillance, les appareils photo, les consoles de jeux et plusieurs autres.

4.2 DCT / IDCT

4.2.1 Description

En traitement d'image, la transformée en cosinus discrète permet principalement de représenter un groupe de pixels en terme d'une somme de fonctions de cosinus oscillant à des fréquences différentes. Il s'agit donc d'une transformation du domaine spatial vers le domaine fréquentiel tout comme la transformée de Fourier discrète (DFT).

Ces fréquences représentent les variations de couleur de l'image. Les hautes fréquences indiquent un changement brusque des couleurs associé aux détails d'une image alors que les basses fréquences indiquent une uniformité dans les couleurs.

$$DCTCoeff(u,v) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} pixel(x,y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2x+1)v\pi}{2N}\right]$$

$$C(x) = \frac{1}{\sqrt{2}} \; si \; x = 0$$
$$C(x) = 1 \; si \; x > 0$$

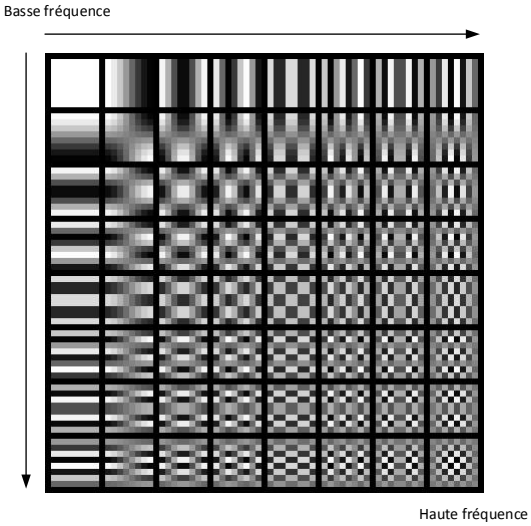


Figure 2 - Résultat d'une DCT 8x8

Ces transformés sont principalement construits afin d'exploiter les limites du système visuel humain. L'œil est plus sensible aux basses fréquences qu'aux hautes. Avec le résultat de la DCT il est possible de réduire l'information provenant des hautes fréquences tout en conservant l'information importante des basses fréquences. Avec la norme JPEG et les autres compressions avec perte, ceci est fait à l'aide d'une matrice de quantification. Ceci dit, la DCT, comme toute transformée, est à elle seule une opération sans perte et complètement réversible.

$$Pixel(x,y) = \frac{1}{\sqrt{2N}}\left[\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)DCTCoeff(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2x+1)v\pi}{2N}\right]\right]$$

$$C(x) = \frac{1}{\sqrt{2}}\ si\ x = 0$$
$$C(x) = 1\ si\ x > 0$$

4.2.2 Application MJPEG

L'IDCT utilisé pour ce laboratoire, fait partie d'une application de décodage de flux MJPEG (Motion JPEG). Le MJPEG est un ancêtre des normes MPEG utilisant principalement la compression de flux vidéo par encodage JPEG. La norme JPEG utilise la redondance spatiale d'une image afin de réduire l'information à transporter.

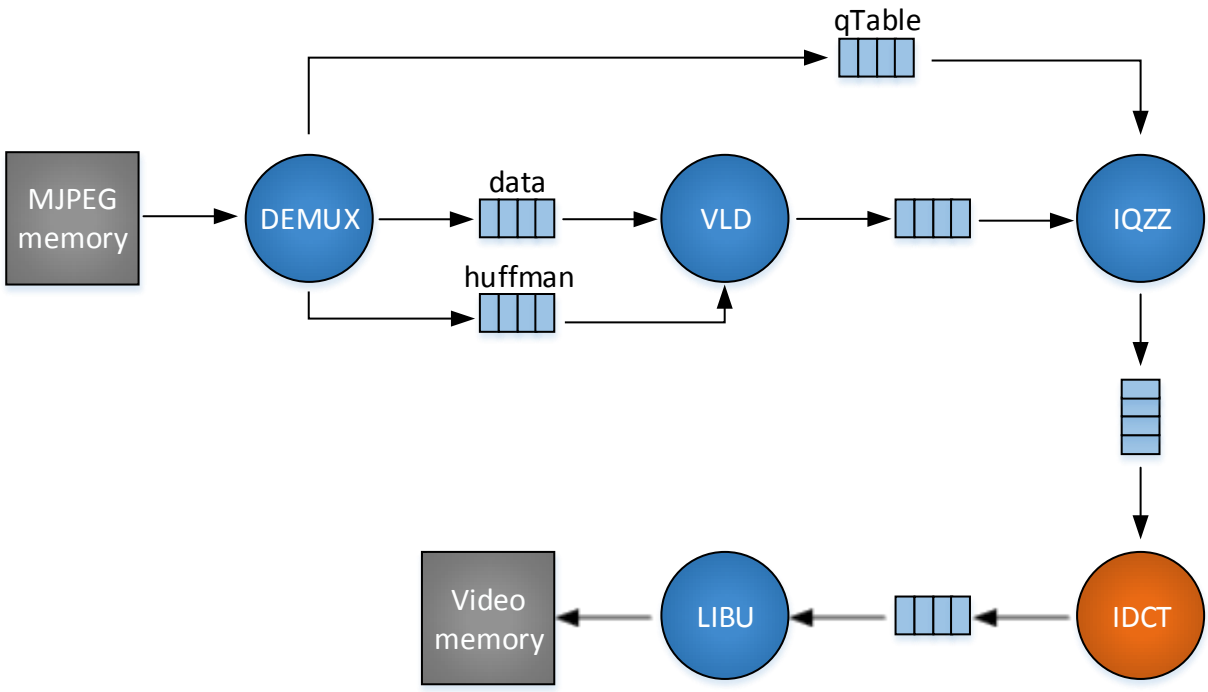


Figure 3 - Flow haut niveau d'une application MJPEG

Tâche	Description
DEMUX	DEMUX vérifie le flux MJPEG pour les marqueurs JPEG. Il envoie les tables de quantification au module IQZZ, les tables de Huffman vers le module VLD et les données vers VLD. Non mentionné dans le schéma, il agit aussi à titre de chef d'orchestre pour l'application. Il communique donc aussi avec LIBU et IDCT afin de synchroniser l'application lors de nouvelle trame à décoder
VLD	VLD effectue le décodage Huffman et envoie le flux décodé vers le module IQZZ
IQZZ	IQZZ fait la quantification inverse et la transformation en zigzag. Le flux résultant est envoyé au module IDCT
IDCT	IDCT effectue une transformée en cosinus discret inverse et envoie le résultat au module LIBU
LIBU	LIBU (LIne BUilder) transforme les blocs de données reçus en ligne d'image pour le contrôleur VGA
COLOR_METRICS	Le module COLOR_METRICS calcule des statistiques sur les couleurs de l'image

Bien que la norme ai été remplacée par des normes plus efficaces telles que MPEG et H.264, elle est toujours utilisée dans les applications embarquées. Par exemple, les caméras vidéo, les appareils de surveillance, les consoles de jeux portables (PSP, Nintendo 3DS) et autres. Bref, elle est moins performante (e.g. débit) que MPEG et H.264, mais beaucoup plus simple à implémenter et moins gourmande en ressources matériels.

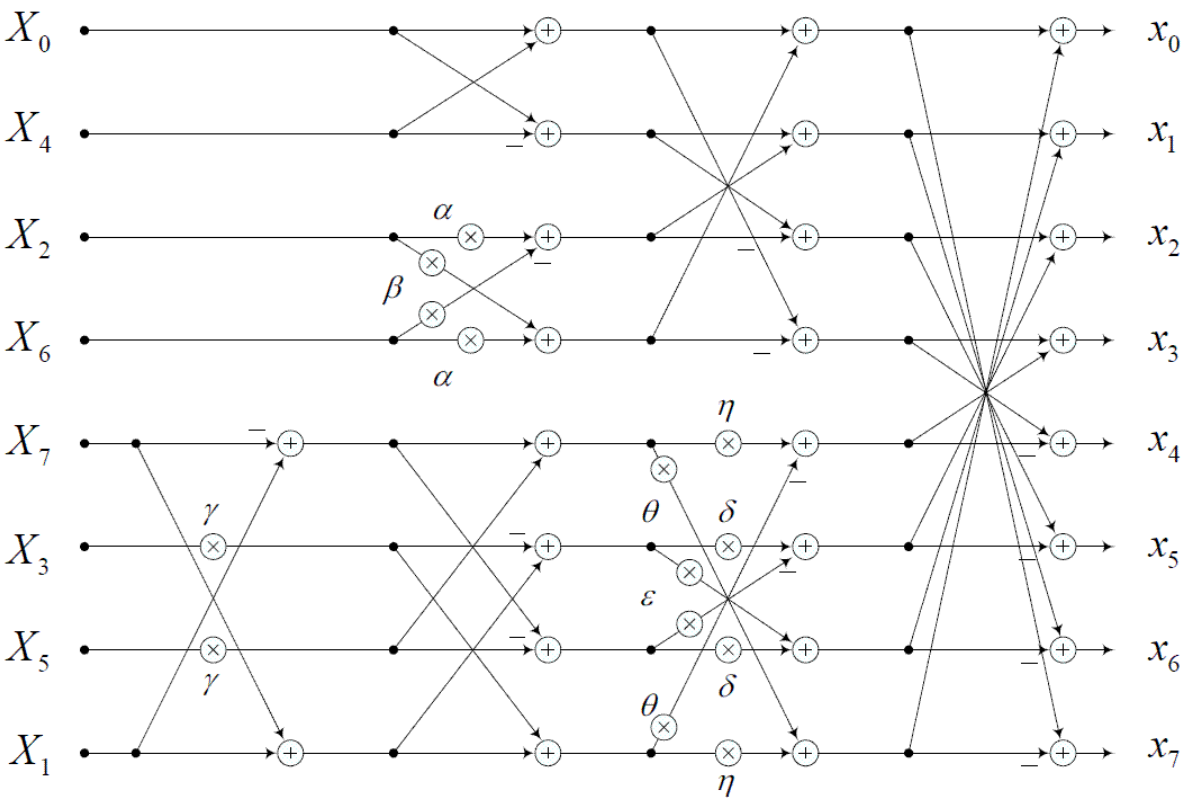
4.2.3 Détail de l'implémentation du module IDCT

Le projet **Vivado HLS** dans lequel se trouve le module IDCT du laboratoire a été généré par l'application **SpaceStudio**. Cette application permet le développement de système à haut niveau à l'aide d'une approche de codesign logiciel/matériel. Elle sera vue plus en profondeur en classe et lors du prochain laboratoire, mais prenez note que le code du module est en partie auto générée.

C'est le cas des fonctions ModuleWrite_X et ModuleRead_X . Celles-ci sont générées afin d'établir les communications entre les modules DEMUX, IQZZ, IDCT et LIBU. Elle s'occupe de lire et d'écrire les données vers les ports de communication. Même chose pour les fonctions getNbBeats qui retourne le nombre de lectures d'entier (4 octets) d'une file nécessaire pour acquérir le nombre d'éléments demandé. Vous remarquerez que seul le type de donnée change entre ces différentes versions. Cette approche n'est pas inefficace, mais reflète plutôt la réalité du domaine matériel qui nécessite les circuits nécessaires pour accéder de façon différente aux données.

Les constructions de type `do { z2[3] = z1[0] - z1[3]; z2[0] = z1[0] + z1[3]; } while(0);` ne sont pas de réelles boucles. Ce type de construction est utilisé à la construction de macro en C afin d'y inclure plusieurs expressions. Ce que nous voyons dans ce module est l'expansion de ces macros. Ne pas en tenir compte lors de la création des directives de synthèse.

La IDCT du module utilise l'approche Papillon (Butterfly) afin de réduire les calculs. Le processus en papillon combine le calcul de plus petite DCT en une large DCT. Ces petites IDCT sont regroupées en étage. Deux points nécessitent un étage, quatre points deux étages, huit points trois étages, etc. Pour les huit points de la norme JPEG, ceci nous limite à 14 multiplications au lieu de 1024 par une approche purement matricielle.



Reznik, Hindsy, Zhangz, Yuz, and Ni, Efficient Fixed-Point Approximations of the 8x8 Inverse Discrete Cosine Transform

Figure 4 - IDCT, Papillon 8x8

La fonction `IDCT_1d` implémente une IDCT à une dimension de huit points similaires à celle présentée à l'image précédente. L'implémentation ajuste les facteurs multiplicatifs à 16bits (multiplication de 2^{14}) afin d'exécuter les calculs à virgule fixe. En effet, vous remarquerez que la table des cosinus est ajustée par un facteur de 2^{14}, ceci afin de ne pas perdre de précision. Les calculs sont par la suite remis à l'échelle lors de l'exécution de la fonction `rot` par la division du même facteur (`>> 14`).

Finalement, la fonction `IDCT_1d` est appelée une fois sur les lignes et par la suite sur les colonnes afin de produire une IDCT à deux dimensions nécessaires pour les images.

4.3 Projet Vivado

Ce travail pratique est accompagné d'un dossier IDCT sous lequel vous trouverez le projet Vivado HLS. Il faut d'abord ouvrir ce projet à partir de l'application Vivado HLS afin d'amorcer le laboratoire.

Le projet Vivado est composé de deux sections avec chacune leurs fichiers. La première section (source) contient le module IDCT et la seconde (Test Bench) contient le banc d'essai SystemC vérifiant la validité du module et son temps d'exécution.

4.3.1 Module

Dossier : Source

- **IDCT.h / IDCT.cpp** : Module SystemC effectuant une transformée en cosinus discrète inverse. Ce sont ces fichiers qui seront modifiés tout au long de ce laboratoire

4.3.2 Test Bench

Dossier : Test Bench

- **tb_init.h/tb_init.cpp** : Module permettant la mise à zero (reset) initial des modules Driver et IDCT
- **tb_driver.h/tp_driver.cpp** : Module pilot permettant l'envoi des données DEMUX et IQZZ vers le module IDCT ainsi que la réception et validation des résultats provenant de l'IDCT vers LIBU. Il émule donc la présence des Module DEMUX, IQZZ et LIBU
- **tb_top.cpp** : Module principal effectuant la connectivité entre les modules tb_init, tb_driver et IDCT
- **DEMUX_IDCT.bin** : Fichier de donnée de test du module DEMUX vers IDCT
- **IQZZ_IDCT.bin** : Fichier de donnée de test du module IQZZ vers IDCT
- **IDCT_LIBU.bin** : Fichier résultat (Golden Model) contant les données qui valideront le fonctionnement du module IDCT

5 Analyse

Lors de l'implémentation d'un module matériel, on se doit de balancer entre différents paramètres, soit l'espace (utilisation du matériel), la consommation (Watt) et la rapidité de calcul. Par exemple, le déroulage de boucle massif va grandement améliorer la rapidité de calcul au coût d'ajout d'unité arithmétique alors que le pipelinage permet un certain balancement entre le débit et l'utilisation de matériel. Il n'existe donc pas de solution ultime qui répond à tous les besoins.

Il existe donc une gamme de solutions matérielles possibles au même problème. Il est donc question de balancer et de trouver le juste milieu entre les performances et les ressources utilisées. On se doit alors d'évaluer la variation des paramètres (déroulage de boucles, gestion de mémoire, etc) sur la variation des utilisations matériels et des performances afin de trouver un ajustement qui convient aux spécifications demandées.

5.1 Exercices

Dans le cadre de ce laboratoire les spécifications ne sont pas stipulées. Il sera donc question de formuler un nombre de solutions afin de voir la variation des performances en fonction des utilisations matérielles. Les performances d'une application de décodage vidéo se mesurent généralement par le débit en image par seconde. En ce qui concerne les ressources sur FPGA, ceux-ci se mesure par l'utilisation des ressources : LUT(Look up table), FF(FlipFlop), BRAM, DSP et SLICE.

Prenez le temps de regarder l'implémentation. Regarder les communications utilisées, les mémoires et l'algorithme de l'IDCT. Lancé la synthèse et regarder les résultats. Tester des directives, manipulées et modifier le code et regarder les résultats à l'aide de la cosimulation RTL. Prenez le temps de vous familiarisé avec le comportement du module en fonctions des directives d'optimisation.

1. Créer une nouvelle solution et nommez la « Initiale ». Cette solution sera la base de votre analyse. Celle-ci ne comportant pas d'accélération particulière, elle servira de comparaison.

2. Créer une gamme de solution avec des directives différentes. Faites varier la performance et grouper des directives (déroulage de boucle, pipelinage, etc.). On demande un maximum de 10 solutions. Vous pouvez utiliser le guide utilisateur afin de découvrir d'autre possibilité d'accélérations en plus de celles mentionné dans le tutoriel. Le lien pour le guide se trouve sur la page moodle du laboratoire.

3. Tracer la courbe de Paréto pour chaque ressource matérielle utilisée (BRAM, DSP, FF et LUT et SLICE) en fonction du temps d'exécution du module (latence). Faites de même pour la consommation vs le temps d'exécution. Un exemple de courbe est fournis avec ce laboratoire (Behavioral_SystemC_Implementation_of_an_MP3_Decoder.pdf)

4. On vous demande d'interpréter le résultat de vos solutions : qu'est-ce qui selon vous produit l'accélération, le ralentissement, l'utilisation d'une ou des ressource, etc. Dans le diagramme de Pareto, expliquez si possible les solutions écartées.

6 Remise

Vous devrez écrire un rapport, de maximum 5 pages, qui décriront le travail effectué.

Ce rapport doit contenir une page présentation, une introduction ainsi qu'une conclusion.

RAPPORT Qualité de la présentation et du français	**/2**
Solutions proposées et interprétation des résultats Détaillé chacune de vos solutions, leurs impacts sur les performances et les ressources utilisées. Expliquer la provenance de ces performances (dépendance de données réduite, augmentation des unité arithmétique, etc.) qu'est-ce qui selon vous produit l'accélération, le ralentissement, l'utilisation d'une ou des ressource, etc **Courbe de Paréto** Créer la courbe de Pareto pour chaque ressource en fonction de la performance (temps d'exécution)	**/10** **/8**
PÉNALITÉ (-2 pts/jour ouvrable, plus de 4 jours entraîne la note 0, si problème consultez le professeur)	
NOTE GLOBALE	**/20**

7 Références

1. Yuriy A. Reznik ; Arianne T. Hinds ; Cixun Zhang ; Lu Yu ; Zhibo Ni; Efficient fixed-point approximations of the 8x8 inverse discrete cosine transform. Proc. SPIE 6696, Applications of Digital Image Processing (September 24, 2007)

2. C. Loeffler, A. Ligtenberg, and G. S. Moschytz, Practical fast 1-D DCT algorithms with 11 multiplications, in Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'89), vol. 2, pp. 988-991, February 1989.

3. L.H Crockett and al, The Zynq Book, Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, Scotland, UK (http://www.zynqbook.com/)

4. Vivado Design Suite User Guide, High-Level Synthesis, UG902 (v2012.4) December 18, 2012

5. http://halicery.com/jpeg/idct.html

6. https://fr.wikipedia.org/wiki/Transform%C3%A9e_en_cosinus_discr%C3%A8te