الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

Présenté par

EL HAOUARI Hichem

Pour l'obtention du diplôme Master en Électronique Traitement de l'Information & Système Electronique (TISE)

Implémentation sur FPGA de techniques de détection de contour et de tatouage d'images

Proposé par : Pr BAHOURA Mohammed

Année Universitaire 2015-2016

Remerciements

A dieu tout Grace

Pour m'avoir guidé et honoré par la lumière et la compréhension et de m'avoir accordé la connaissance de la science. Louange à Dieu tout puissant qui m'a donné santé et courage pour achever ce modeste travail.

Je tiens à exprimer ma profonde gratitude à Pr BAHOURA Mohammed pour sa supervision à ce travail, je le remercie infiniment pour sa rigueur, sa simplicité, sa générosité et sa disponibilité durant la réalisation de ce travail. Ses conseils précieux m'ont permis de réaliser mon mémoire de fin d'étude.

Mes remerciements les plus vifs s'adressent aussi à Monsieur le président et les membres du jury qui ont accepté d'examiner et d'évaluer mon travail.

Mes remerciements s'adressent à tout enseignant ayant contribué à ma formation depuis mon premier cycle au primaire jusqu'au dernier cycle à l'université.

Merci à toutes les personnes qui m'ont aidé de près ou de loin à réaliser ce mémoire.

ملخص: الهدف من هذه الدراسة هو تنفيذ تقنيات معالجة الصور على شريحة FPGA، ثم مقارنة النتائج المتحصل عليها مع تلك التي حصلت مع SIMULINK. واحدة من هذه التقنيات الرئيسية هي العلامة المائية للصور الرقمية التي لديها كم هائل من الخوارزميات حيث اختيرت واحدة منها و ثم تحسينها لكي تنفذ علي بطاقة التطوير Nexys-4. التحدي الكبير في هذا المشروع، هو ترجمة هذه الخوارزميات من مخططات SIMULINK باستخدام كتل XSG، بتقليل عدد البتات من اجل الحد من الموارد المادية علي شريحة FPGA.

كلمات المفاتيح: FPGA, معالجة الصور، العلامة المائية للصور، XSG, مويجه

Résumé : L'objectif de ce mémoire est d'implémenter un ensemble de techniques de traitement d'images sur un circuit FPGA, puis de comparer les résultats de leur implémentation avec ceux obtenus avec *MATLAB/SIMULINK*. L'une des principales techniques est le tatouage d'image numérique, qui possède énormément d'algorithmes, a été choisi et amélioré pour ensuite être implémenté sur une carte de développement Nexys-4. Le grand défi dans ce projet est de pouvoir traduire ces algorithmes en diagrammes, *SIMULINK* utilisant les blocs *Xilinx System Generator* (XSG), en minimisant le nombre des bits afin de réduire les ressources matérielles requises sur la puce FPGA.

Mots clés : FPGA, Traitement d'image, Tatouage d'image, XSG, Ondelette

Abstract: The objective of this thesis is to implement a set of image processing techniques on an FPGA chip, and then compare the implementation results with those obtained with *MATLAB/SIMULINK*. One of the main techniques is digital image watermarking, which has a huge number of algorithms, has been chosen and improved to be then implemented on the Nexys-4 development board. The great challenge in this project is to be able to translate these algorithms into *SIMULINK* diagrams using *Xilinx System Generator* (XSG) block set, by minimizing the number of bits in order to reduce the required hardware resources on the FPGA chip.

Keywords: FPGA, Image Processing, Image watermarking, XSG, Wavelet

Listes des abréviations

1D	1-Dimension
2D	2-Dimension
ASIC	Circuit Intégré à Application Spécifique
CPU	Unité centrale de traitement
CRC-32	Contrôle de redondance cyclique 32bits
CWT	Transformée en ondelettes Continue
db2/4/20	Ondelette de Daubechies 2/4/20
dB	Décibel
DCT	Transformée en cosinus discrète
DFT	Transformée de Fourier discrète
DSP	Processeur de traitement numérique du signal
DT-CWT	Discrétisation temporelle de la transformée en ondelettes continue
DWT	Transformée en ondelettes discrètes
E/S	Entrée/Sortie
EEPROM	Mémoire ROM programmable effaçable électriquement
EZW	*Embedded Zerotree Wavelet*
FFT	Transformée de Fourier Rapide
FPGA	Réseau de portes logiques programmable
HDL	Langage de description matériel
HVS	Système de vision humaine
Hw	Matériel
IDWT	Transformée en ondelettes discrètes inverse

IOB	Bloc Entrée/Sortie
IRM	Imagerie par Résonance Magnétique
JPEG	*Joint Photographic Experts Group*
JTAG	*Joint Test Action Group*
LED	Diode électro-luminescente
LSB	Le bit le moins significatif
LUT	Table de correspondance
Max	Maximum
MD-4	Message Digest 4
MD-5	Message Digest 5
Min	Minimum
MPEG	*Moving Picture Experts Group*
MSB	Le bit le plus significatif
PLL	Boucle de verrouillage de phase
RGB	Rouge Vert Bleu
SHA-1	*Secure Hash Algorithm* 1
SRAM	Mémoire Statique à accès aléatoire
USB	bus universel en série
VGA	Tableau de Graphics vidéo
VHDL	Langage de description matériel de grande vitesse pour circuits intégrés
VT	Vecteur de transformation
XSG	*Xilinx System Genrator*
YUV	Luma et les deux composantes de chrominance

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Dans le monde actuel, la photo ou l'image fait partie de notre vie de tous les jours qu'elle soit personnelle ou professionnelle. Avec l'évolution technologique, l'acquisition, l'archivage et le transfert de l'image se font en numérique (image numérique). Toutefois, l'image numérique acquise nécessite des opérations supplémentaires qui permettent, par exemple, de la rendre plus claire afin d'extraire efficacement les informations utiles incluses dans cette image. Cette opération est appelée traitement d'image.

Le traitement d'image a beaucoup changé notre vie et a donné un nouveau sens à l'image. Par exemple, l'apport de l'imagerie médicale au bon diagnostic de plusieurs maladies et l'usage de l'imagerie satellite dans la météo, la cartographie et la navigation. Pour pouvoir traiter une image, elle doit passer généralement par une étape de prétraitement qui consiste soit à éliminer le bruit qui la contamine ou à égaliser son histogramme. Après cette étape, l'image est prête pour le traitement.

Le traitement d'image est un domaine très vaste qui utilise de nombreuses techniques dont certaines sont ou peuvent être installées sous forme d'applications sur nos smartphones. Toutefois, le partage d'images dans un réseau local ou sur internet nécessite la protection de certaines informations, comme l'identité du patient d'une image IRM, etc. Comment y arriver ?

Pour protéger les informations personnelles d'un patient ou garantir le droit d'auteur (signature) d'une image, il y a une technique qui a été utilisée depuis des années avec la version papier et qui est utilisée maintenant avec les images numériques. C'est le tatouage d'images numériques, appelé en anglais « *Watermarking* ». Cette technique de traitement protège les images contre la copie pirate. Elle protège également les informations professionnelles comme dans le cas

de la télémédecine où le médecin intègre les données de patient dans l'image (IRM, Radio…etc.) et protège ces données avec une clé afin de partager cette image avec un autre collègue en toute confidentialité.

Ces techniques ont déjà été implémentées en logiciel (software), principalement sous *MATLAB*, mais des efforts sont en cours pour les implémenter en matériel (hardware) dans le but de réaliser des systèmes embarqués fonctionnant en temps-réel. L'objectif de ce projet est d'implémenter ces techniques sur un circuit FPGA en utilisant le logiciel haut-niveau *Xilinx System Generator* (XSG) et la carte de développement Nexys-4. L'utilisation de cet outil de programmation permet, entre autres, de profiter des facilités de simulation de *MATLAB/SIMULINK*.

Ce mémoire est organisé en trois chapitres. Le premier présente la théorie de traitement d'images. Le second chapitre est consacré spécialement au tatouage d'images. L'implémentation sur FPGA et les résultats de simulation sont présentés dans le troisième chapitre. Le mémoire se termine par une conclusion générale et des perspectives.

Chapitre 1

Chapitre 1 TECHNIQUES DE TRAITEMENT D'IMAGES

Les images numériques sont des captures électroniques prises d'une scène ou numérisées à partir de documents, tels que des manuscrits, des textes imprimés, des illustrations, et des photographies. L'image numérique est échantillonnée et tracée comme une grille de points ou éléments de base (pixels) de l'image. À chaque pixel est attribuée une valeur tonale (noir, blanc, nuances de grise ou couleur), qui est représentée en code binaire (0 et 1) [1].

Les images numériques sont constituées d'éléments d'image, appelés pixels. En règle générale, les pixels sont organisés dans un tableau rectangulaire commandé. La taille d'une image est déterminée par les dimensions de cette matrice de pixels. La largeur de l'image représente le nombre de colonnes, alors que la hauteur de l'image représente le nombre de lignes dans la matrice. Ainsi, la matrice de pixels est une matrice de M colonnes x N lignes [2].

Après avoir défini le nombre de pixels, MxN, notre image sera définie seulement par une forme rectangulaire. Un autre paramètre, l'intensité, est nécessaire pour définir vraiment cette image. Chaque pixel a sa propre valeur en intensité ou en luminosité. Si tous les pixels ont la même valeur, l'image aura une teinte uniforme ; tout en noir, blanc, gris, ou d'une autre ombre [2].

Le traitement des images numériques est un sujet très répandu. Il permet de modifier les images pour améliorer leur qualité (restauration), extraire une information (analyse, reconnaissance) et changer leur structure (composition, retouche d'image) [3].

1.1 Conversion couleur en niveau de gris

1.1.1 Définition

Niveau de gris (*Grayscale*) est une gamme de nuances de gris sans couleur apparente. La nuance la plus sombre possible est le noir, ce qui est l'absence totale de lumière transmise ou réfléchie. La teinte la plus claire possible est le blanc. Les nuances intermédiaires du gris sont représentées par l'égalité de luminosité des trois niveaux des couleurs primaires (rouge, vert et bleu) pour la lumière transmise, ou des quantités égales des trois pigments primaires (cyan, magenta et jaune) pour la lumière réfléchie [4].

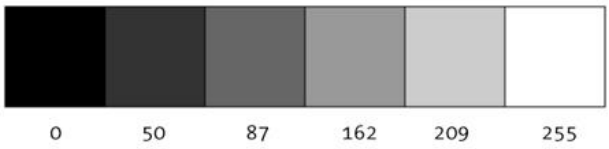


Figure 1.1. Différents niveaux de gris [5]

1.1.2 Méthodes de conversion

La conversion en niveaux de gris se fait dans différentes manières. Les meilleures d'entre elles utilisent une combinaison linéaire des valeurs RGB (rouge-vert-bleu) d'un pixel, mais pondérée en fonction de la façon dont nous percevons l'intensité des couleurs. Une méthode non linéaire de conversion en niveaux de gris est d'utiliser les valeurs des pixels. En général, les mêmes principes sont applicables à cette méthode comme il est fait pour la présentation de ces renseignements perceptuels. Si un Colormap de valeurs de plus en plus monotones est choisi, il sera imprimé d'une manière raisonnable en niveaux de gris [6].

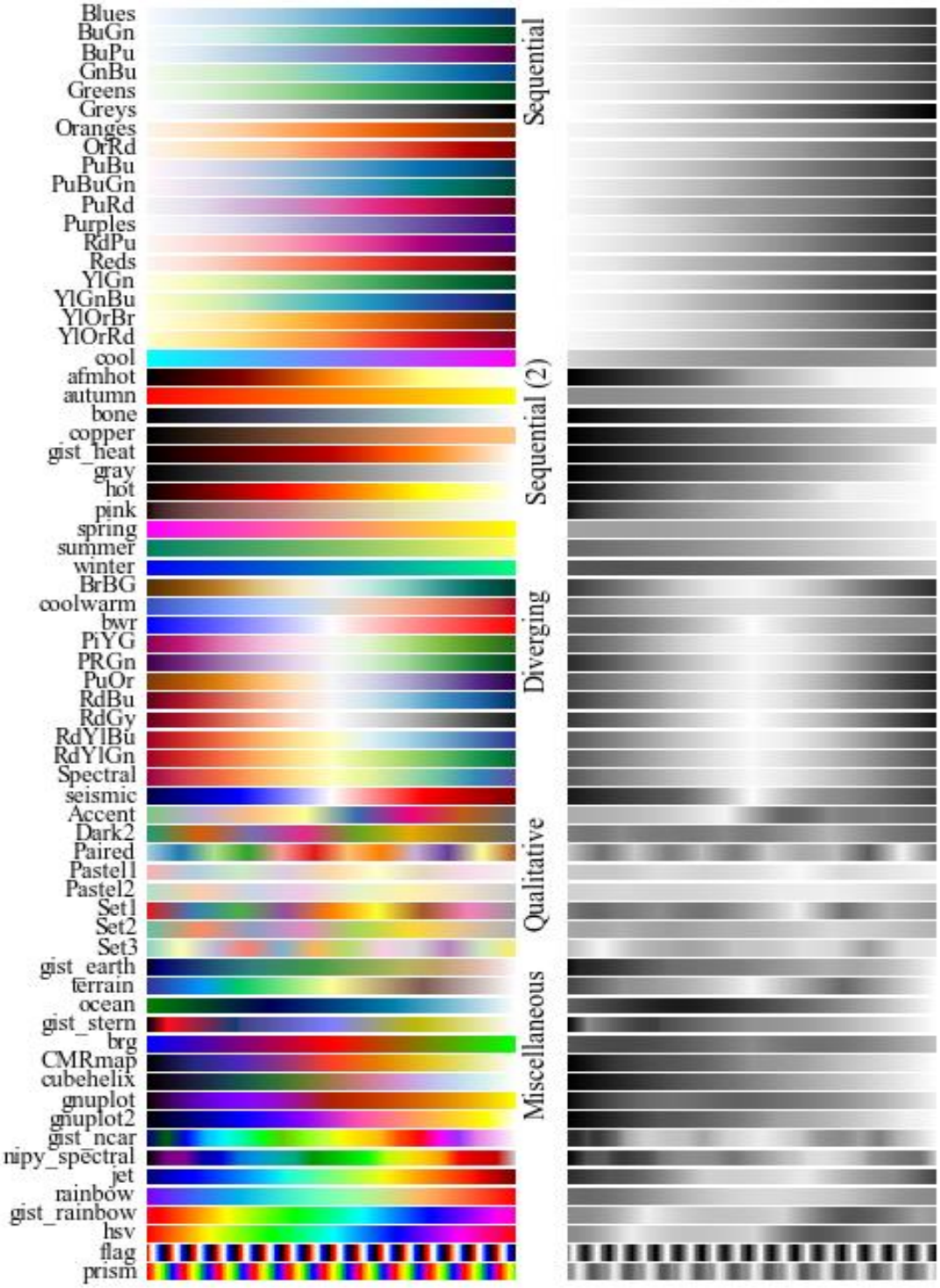


Figure 1.2. Colormap [6]

Il existe les deux méthodes suivantes pour le convertir :

- Méthode de la moyenne
- Méthode pondérée ou la méthode de luminosité

a. Méthode de la moyenne

La méthode de la moyenne est la plus simple, car elle se base simplement sur la moyenne des trois couleurs. Depuis une image RGB, cela signifie que nous avons à mélanger R avec G avec B, puis diviser par 3 pour obtenir notre image en niveaux de gris souhaitée [7].

$$\text{Grayscale} = (R + G + B / 3) \qquad (1.1)$$

Les résultats ne sont pas comme prévu. Nous avons voulu convertir l'image en niveaux de gris, mais cela est avéré être une image plutôt noir.

b. Méthode pondérée

Nous avons vu le problème qui se produit dans la méthode de la moyenne. La méthode pondérée présente une solution à ce problème, puisque la couleur rouge a une longueur d'onde plus grande que les trois couleurs et le vert est la couleur qui a, non seulement, moins de longueur d'onde que le rouge, mais aussi donne un effet plus apaisant pour les yeux [7].

Cela signifie que nous devons réduire la contribution de la couleur rouge, accroître la contribution de la couleur verte, et mettre la contribution de couleur bleue entre ces deux.

Ainsi, la nouvelle équation a la forme suivante :

$$\text{Grayscale} = ((0{,}3 \times R) + (0{,}59 \times G) + (0{,}11 \times B)) \qquad (1.2)$$

(a) (b) (c)

Figure 1.3. (a) image original ‘RGB’ (b) Méthode de la moyenne (c) Méthode pondérée

1.2 Image complémentaire

1.2.1 Définition

Une image positive est une image claire. Une image négative est une inversion totale, dans laquelle les zones lumineuses apparaissent sombre et vice-versa. Une image négative de couleur est en outre la couleur inversée, avec des zones rouge apparaissant cyan, vert apparaissant magenta et bleu apparaissant jaune.

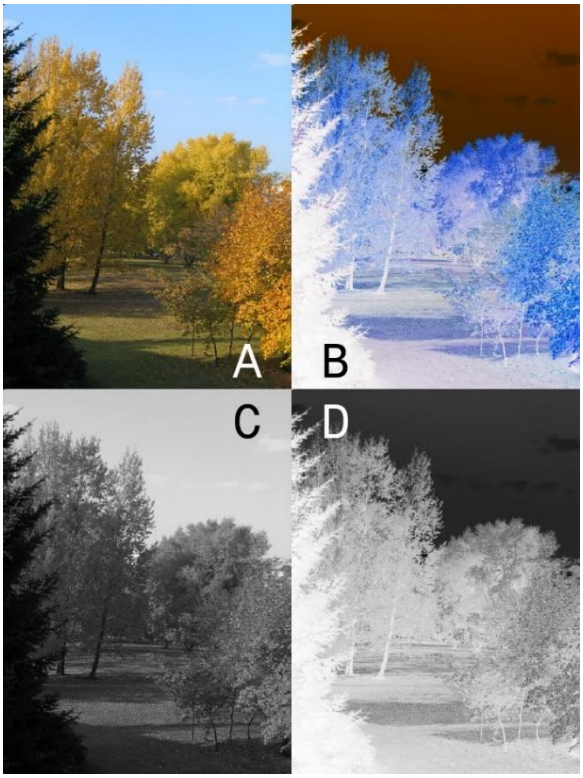


Figure 1.4. (A) Image couleur positive, (B) négatif de A, (C) monochrome image positive, (D) négatif de C [8]

Convertir une image couleur en négatif est très simple. Tout ce que nous avons à faire est de répéter 3 étapes simples pour chaque pixel de l'image.

- Obtenir la valeur RGB du pixel.
- Calculer la nouvelle valeur RGB comme indiqué ci-dessous.

$$R = 255 \oplus R$$
$$G = 255 \oplus G \qquad (1.3)$$
$$B = 255 \oplus B$$

avec $\oplus$ est l'opérateur logique XOR

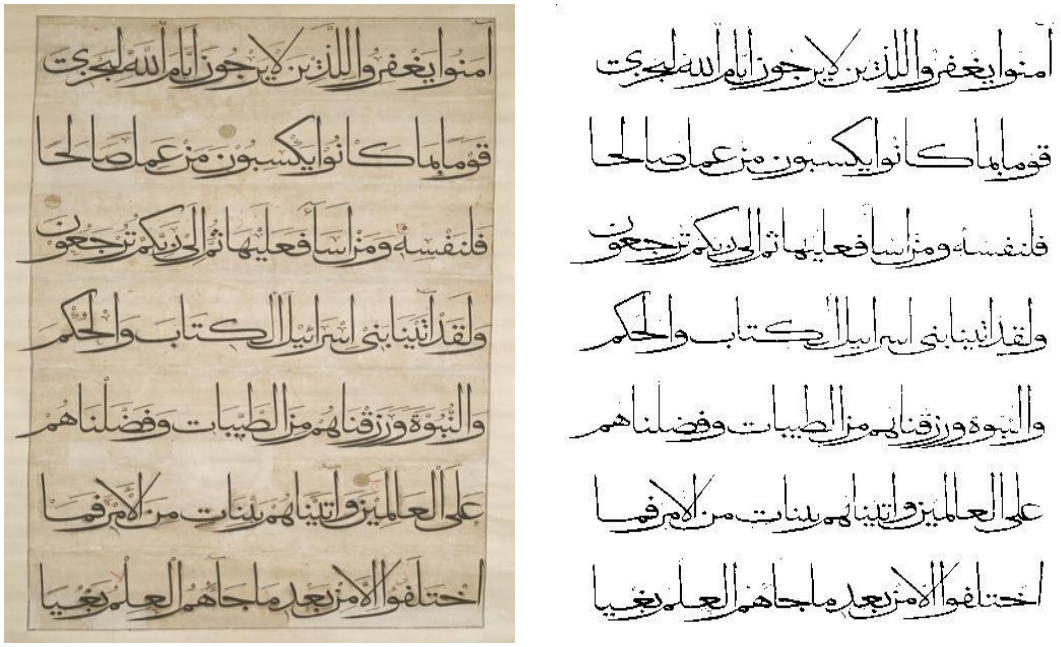
- Enregistrer la nouvelle valeur RGB dans le pixel.

1.3 Seuillage

1.3.1 Définition

Le Seuillage est un procédé de conversion d'une image d'entrée en niveaux de gris vers une image à deux niveaux (binaire) à l'aide d'un seuil optimal.

Le but de seuillage est d'extraire les pixels d'une certaine image qui représente un objet (texte ou d'autres données d'image tels que des graphiques ou des cartes). Bien que l'information soit binaire, les pixels représentent une gamme d'intensité. Ainsi, l'objectif de binariser est de marquer des pixels qui appartiennent à de véritables régions de premier plan avec une seule région d'intensité et ceux du fond avec des intensités différentes [9].



(a) (b)

Figure 1.5. (a) image original ‘texte’ (b) image ‘texte’ après seuillage (T=100) [10]

1.3.2 Méthode de seuillage d’image

Le procédé le plus simple de seuillage est de remplacer chaque pixel d'une image avec un pixel noir si son l'intensité I(i,j) est inférieure à un certain seuil T, ou avec un pixel blanc si son l'intensité est supérieure à cette constante [11].

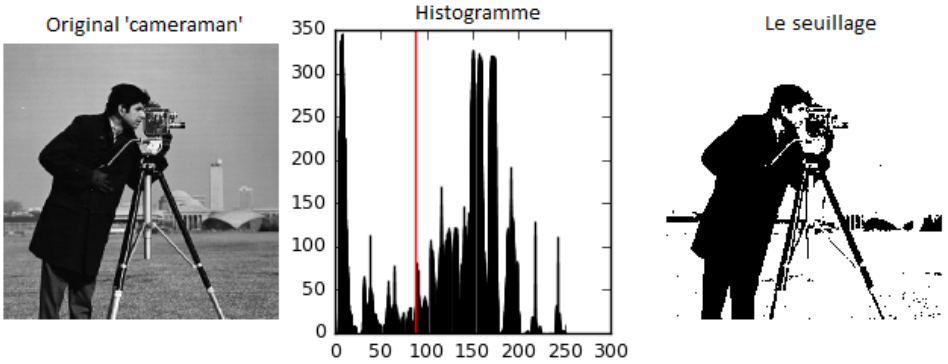


Figure 1.6. Choix du seuil à partir de l'histogramme

$$g(i,j) = \begin{cases} 1 \; si \; I(i,j) > T \\ 0 \; si \; I(i,j) \leq T \end{cases} \quad (1.4)$$

où I(i,j) l'image originale et g(i,j) est l'image après seuillage.

a. Choix des seuils

La difficulté principale dans toute méthode de seuillage est le choix du seuil (ou de l'intervalle de seuillage). En utilisant un intervalle trop large, on obtient des faux positifs, c'est-à-dire l'image obtenue par seuillage contient des pixels qui ne font pas partie des objets d'intérêt ; il s'agit généralement de bruit, ou des structures d'une autre nature, qui ont un niveau de gris avoisinant celui des objets recherchés. L'utilisation d'un intervalle trop étroit permet d'obtenir des faux négatifs, c'est-à-dire certains objets d'intérêt n'apparaissent pas, ou seulement en partie, dans l'image obtenue par seuillage. Quelques méthodes pour la sélection de l'intervalle de seuillage sont décrites ci-dessous [12].

a.a. Méthodes de seuillage basées sur l'histogramme

- **ISODATA**

Cette méthode consiste à trouver un seuil en séparant l'histogramme en deux classes, itérativement avec la connaissance à priori des valeurs associées à chaque classe. On détermine l'intervalle [min, max] des valeurs non nulles de l'histogramme. Après, on fait une estimation des valeurs moyennes initiales en divisant l'intervalle en deux parties équidistantes et en prenant m1 et m2 comme la moyenne arithmétique de chaque classe.

À chaque itération, on calcule le seuil T en prenant l'entier le plus proche de la moyenne des deux moyennes :

$$T = \frac{m_1+m_2}{2} \qquad (1.5)$$

Puis, on met à jour les moyennes en calculant la moyenne statistique pour chaque classe :

$$m_1 = \frac{\sum_{j=min}^{T} j \times h(j)}{\sum_{j=min}^{T} h(j)} \qquad (1.6)$$

$$m_2 = \frac{\sum_{j=T+1}^{max} j \times h(j)}{\sum_{j=T+1}^{max} h(j)} \qquad (1.7)$$

On recalcule les seuils et les moyennes jusqu'à ce qu'il n'y aura aucun changement.

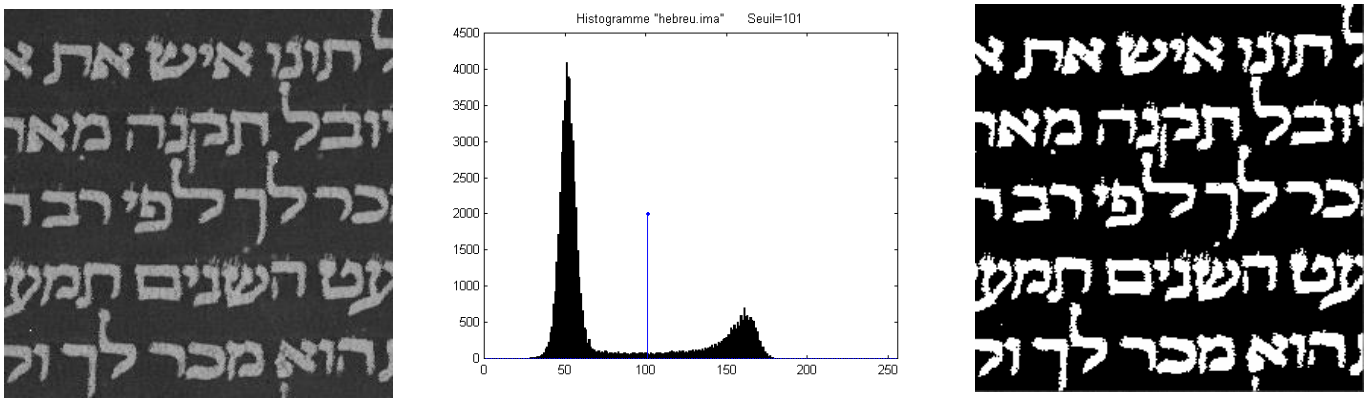


Figure 1.7. Résultats de la méthode ISODATA

- **Centres locaux**

  Elle consiste à déplacer, à chaque itération, les valeurs de l'histogramme vers le centre de gravité le plus proche en obtenant, à la fin, un nouvel histogramme dont les deux classes sont plus différenciées qu'avant.

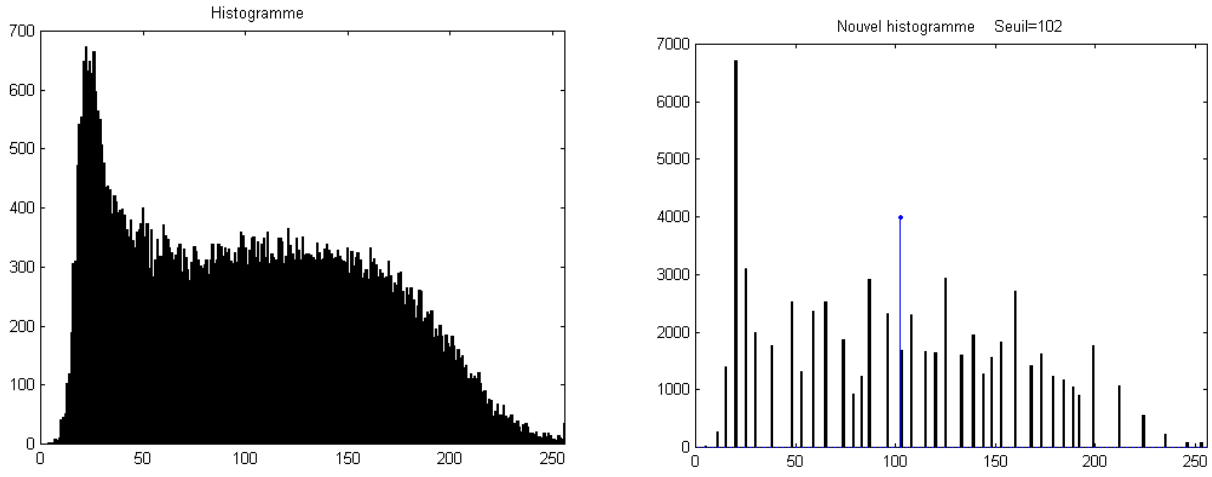
  On calcule, pour chaque index de l'histogramme, la moyenne sur une fenêtre de taille 2a+1 (moyenne glissante).

  Après, on utilise ces moyennes pour accumuler les valeurs h(k) de l'histogramme de même moyenne glissante. Dans ce nouvel histogramme, les

index sont les moyennes glissantes possibles. Une valeur quelconque de cet histogramme $h^{i+1}(k)$ garde la somme des valeurs $h^{i}(k)$ de l'histogramme précédant qui ont une moyenne glissante ‘m’ égale à k.

$$h^{i+1}(k) = \sum_{k_{i=k1}}^{k2} h^{i}(k_i)\delta\left(k - m^{i}(k_i)\right) \qquad (1.8)$$

On refait les calculs des moyennes glissantes du nouvel histogramme itérativement tant qu'il n'y a aucun changement entre histogrammes. Sur ce dernier, on peut observer le déplacement de parties des masses de l'histogramme initial vers leurs centroïdes locaux.



(a) (b)

Figure 1.8. (a) L’histogramme original ‘centres locaux’ (b) histogramme après seuillage ‘centres locaux’

- **Analyse discriminante**

Elle consiste à trouver un seuil en maximisant la séparabilité entre les deux classes. Pour chaque seuil T possible, on détermine la moyenne pour chacune des deux classes et la probabilité qu'un point de l'image lui appartienne :

$$P_0 = \sum_{i=1}^{T} p(i) \qquad (1.9)$$

$$P_1 = \sum_{i=T+1}^{256} p(i) \qquad (1.10)$$

$$\mu_0 = \sum_{i=1}^{T} i \times p(i) \qquad (1.11)$$

$$\mu_1 = \sum_{i=T+1}^{256} i \times p(i) \qquad (1.12)$$

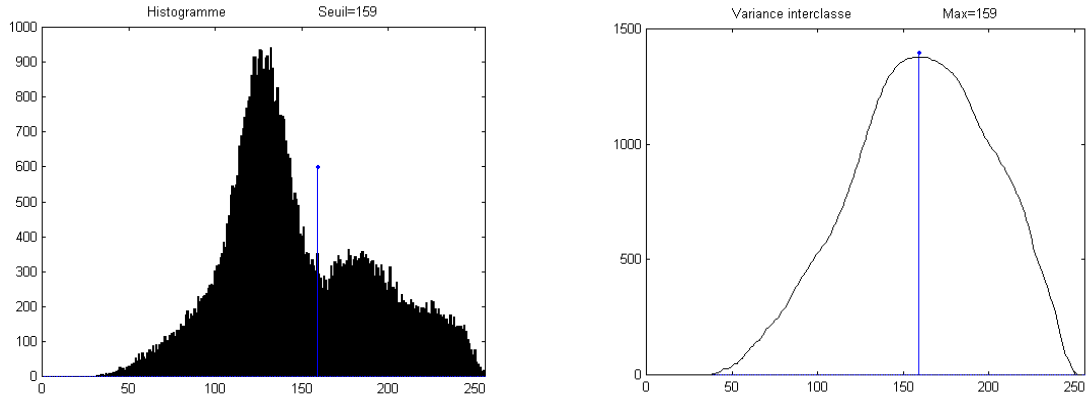
Puis, on détermine la variance entre ces deux classes en faisant :

$$\sigma^2 = P_0(\mu_0 - \mu)^2 + P_1(\mu_1 - \mu)^2 \qquad (1.13)$$

Notons qu'il faut calculer une variance pour chaque seuil. Finalement, on calcule le seuil T_{opt} pour lequel la variance interclasse est maximale :

$$\sigma^2(T_{opt}) = \max_{T=1\ldots256}\{\sigma^2(T)\} \qquad (1.14)$$

Cette relation ne nous donne pas seulement la valeur de T qui permet de séparer les classes 0 et 1, mais aussi le degré de dissimilarité qui sépare les classes.



(a) (b)

Figure 1.9. (a) L'histogramme original 'analyse discriminante' (b) histogramme après seuillage 'analyse discriminante'

a.b. Méthode d'un algorithme simple

- Choisir un seuil T initial (moyenne, médiane, ...).
- 2 groupes de pixels de moyenne µ1 et µ2.
- On calcule $T = \frac{\mu_1 + \mu_2}{2}$ (1.15)

- On itère jusqu'à ce que T soit constant.

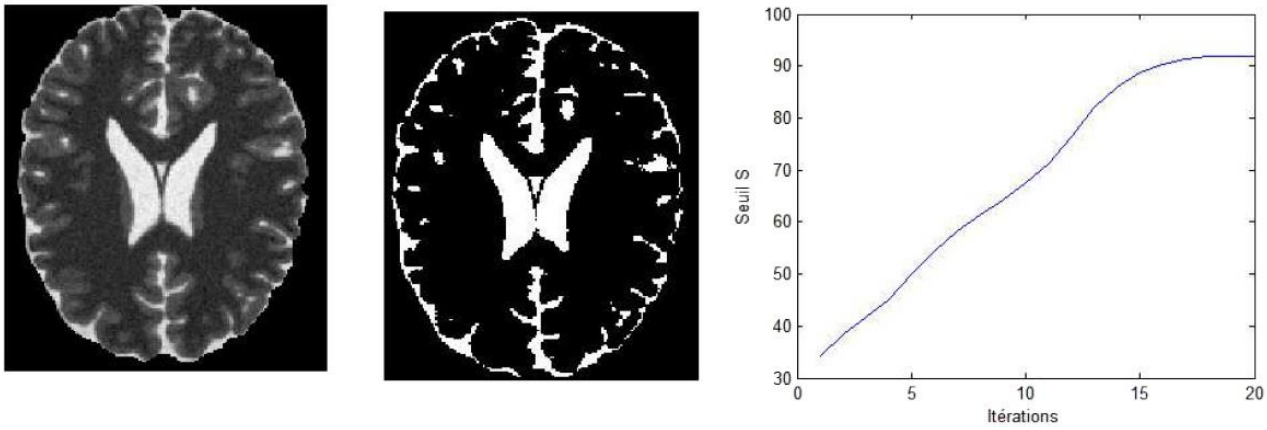


Figure 1.10 Résultats d'un algorithme simple

a.c. Méthode d'Otsu (1979)

- Un seuil T définit deux groupes de pixel : C_1 et C_2
- On cherche alors le seuil qui minimise la variance intra-classe :

$$\sigma_\omega^2 = \omega_1(T)\sigma_1^2(T) + \omega_2(T)\sigma_2^2(T) \qquad (1.16)$$

- Les poids $\omega_1(T)$ et $\omega_2(T)$ représentent les probabilités.
- Les σ_i^2 sont les variances de ces classes.

Figure 1.11 Résultats de la méthode d'Otsu [13]

1.4 Détection de contour

1.4.1 Définition

La détection de contour est un traitement très important dans le domaine de la vision par ordinateur. Le contour définit les limites entre régions dans une image, ce qui contribue à la segmentation et à la reconnaissance d'objets. Les contours peuvent montrer où les ombres tombent dans une image ou tout autre changement notable de l'intensité d'une image.

Etant donné que les différents détecteurs de contour fonctionnent mieux dans des conditions différentes, il serait idéal d'avoir un algorithme qui fait usage de multiples détecteurs à bord et qui permet l'application du bon détecteur lorsque les conditions de la scène sont les plus idéales pour sa méthode de détection. Afin de créer ce système, on doit d'abord savoir quels détecteurs de contour fonctionnent mieux sous quelles conditions [14].

Certains opérateurs utilisés pour la détection de contours sont :

- CANNY
- DERICHE
- SOBEL
- PREWITT
- ROBERT CROSS
- HARRIS OPERATOR

1.4.2 L’opérateur (Filtre) SOBEL

Le filtre de Sobel est un opérateur couramment utilisé en traitement d'image pour la détection de contours. C’est un des opérateurs les plus simples qui donne toutefois de bons résultats.

L’opérateur calcule le gradient de l'intensité de chaque pixel. Ceci indique la direction de la plus forte variation du clair au sombre, ainsi que le taux de changement dans cette direction. Par conséquent, on détermine les points du changement soudain de luminosité, correspondant éventuellement à des bords, ainsi que l'orientation de ces bords.

En termes mathématiques, le gradient d'une fonction de deux variables (ici l'intensité en fonction des coordonnées de l'image) est un vecteur à 2 dimensions,

dont les coordonnées sont les dérivées selon les directions horizontale et verticale. À chaque point, le gradient pointe dans la direction du plus fort changement d'intensité, et sa longueur représente le taux de variation dans cette direction. Le gradient dans une zone d'intensité constante est donc nul. Au niveau d'un contour, le gradient traverse le contour, des intensités les plus sombres aux intensités les plus claires [15].

a. Conception du filtre SOBEL

La plupart des méthodes de détection de contour fonctionnent sur l'hypothèse que le contour se produit lorsqu'il existe une discontinuité en fonction de l'intensité ou un gradient d'intensité très élevé dans l'image. En utilisant cette hypothèse, si nous prenons la dérivée des valeurs d'intensité à travers l'image afin de trouver le point où la dérivée est maximale, le contour pourrait théoriquement être localisé [16].

Les valeurs changent avec la distance dans les directions x et y. Ainsi, les composantes du gradient peuvent être trouvées en utilisant les approximations suivantes :

$$\Delta x = \frac{\partial f(x,y)}{\partial x} \tag{1.17}$$

$$\Delta y = \frac{\partial f(x,y)}{\partial y} \tag{1.18}$$

où Δx et Δy mesurent la distance respectivement le long des directions x et y. En images discrètes, on peut considérer Δx et Δy en termes de nombre de pixels entre deux points.

$$\Delta x = f(i+1,j) - f(i,j) \tag{1.19}$$

$$\Delta y = f(i,j+1) - f(i,j) \tag{1.20}$$

Afin de détecter la présence d'une discontinuité du gradient, on calcule le changement dans le gradient du pixel (i, j) selon :

$$M = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{1.21}$$

et la direction est donnée par :

$$\theta = arctan\left(\frac{\Delta y}{\Delta x}\right) \tag{1.22}$$

L'opérateur Sobel est un opérateur de différenciation discrète qui permet le calcul d'une approximation du gradient de l'intensité de l'image. Les différents opérateurs en eq. (1.17) et (1.18) correspondent à la convolution de l'image avec les masques suivants :

$$\Delta x = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \tag{1.23}$$

$$\Delta y = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \tag{1.24}$$

Après cela, il se fait :

- Le coin supérieur gauche du masque approprié est superposé au-dessus de chaque pixel de l'image.
- Une valeur est calculée pour Δx ou Δy, en utilisant les coefficients de masque d'une somme pondérée de la valeur du pixel (i, j) et ses voisins.
- Ces masques sont appelés masques de convolution ou parfois noyaux de convolution. Maintenant, les masques décrits ci-dessus sont un peu petits et peuvent être très pratiques à utiliser. En outre, un avantage d'utiliser une plus grande taille du masque est que les erreurs dues aux effets du bruit sont réduites en faisant la moyenne locale dans le voisinage du masque.

L'avantage d'utiliser un masque de taille plus grand est que les opérateurs sont centrés et peuvent, par conséquent, fournir une estimation basée sur le centre de pixel (i, j). Les masques de l'opérateur Sobel sont donnés par :

$$\Delta x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{1.25}$$

$$\Delta y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1.26}$$

Figure 1.12. Image original ‘coin’

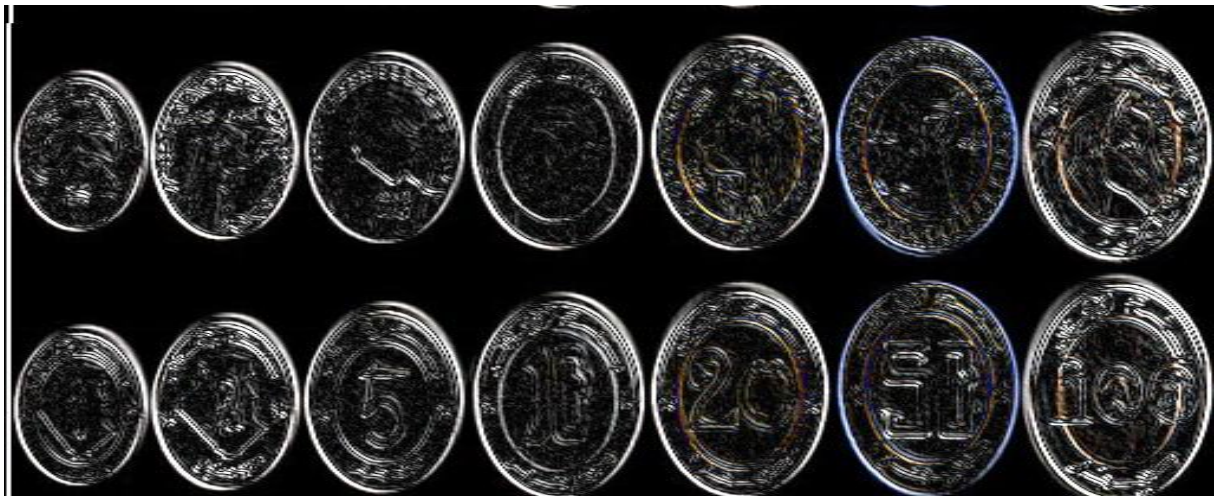


Figure 1.13. Image ‘coin’ après l’application de l’opérateur de SOBEL XY

1.5 La transformée en ondelettes discrète (DWT)

La transformée en ondelettes discrète (DWT) est une implémentation discrète de la transformée en ondelettes utilisant le plus souvent l'algorithme de Mallat [17]. La figure 1.14 représente le diagramme de décomposition d'image (ligne et colonne) en arbre à un seul niveau.

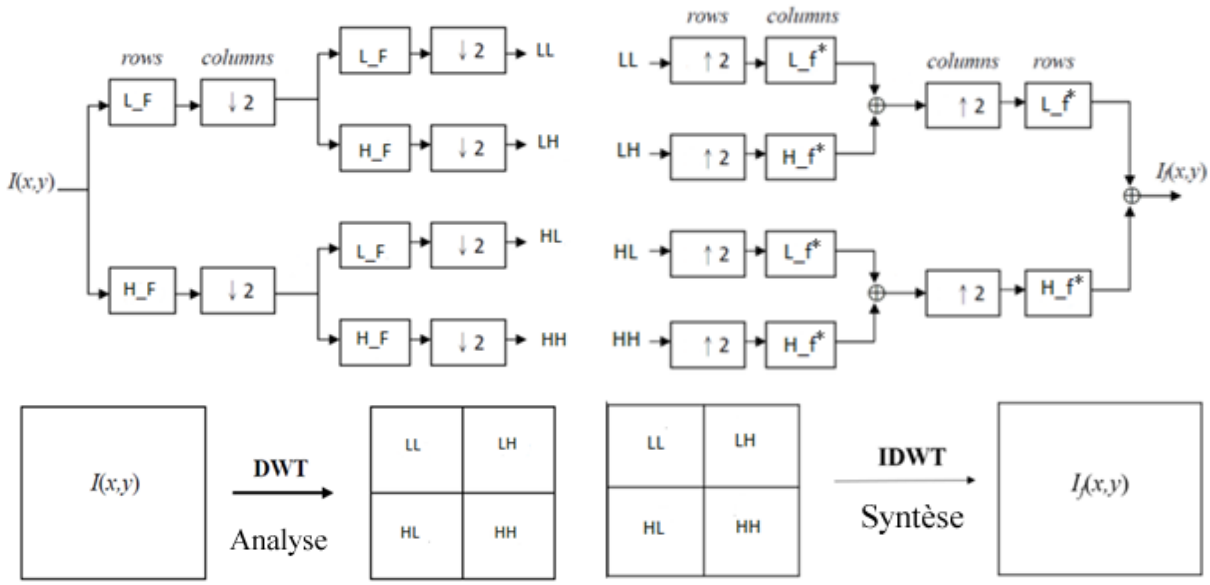


Figure 1.14. Décomposition de l'image par l'algorithme de Mallat [18]

Pour la partie décomposition (analyse), 'L_F' est un filtre passe-bas et 'H_F' un filtre passe haut d'analyse avec le symbole ↓2 représente le sous-échantillonnage par 2. La sortie 'LL' représente l'approximation de l'image, alors que LH, HL et HH représentent ses détails.

Pour la partie reconstruction (synthèse), 'L_F*' est un filtre passe-bas et 'H_F*' un filtre passe haut de reconstruction avec le symbole ↑2 représente le sur-échantillonnage par 2.

Les filtres d'analyse ou de la reconstruction sont calculés à partir d'une ondelette analysante, appelée ondelette mère.

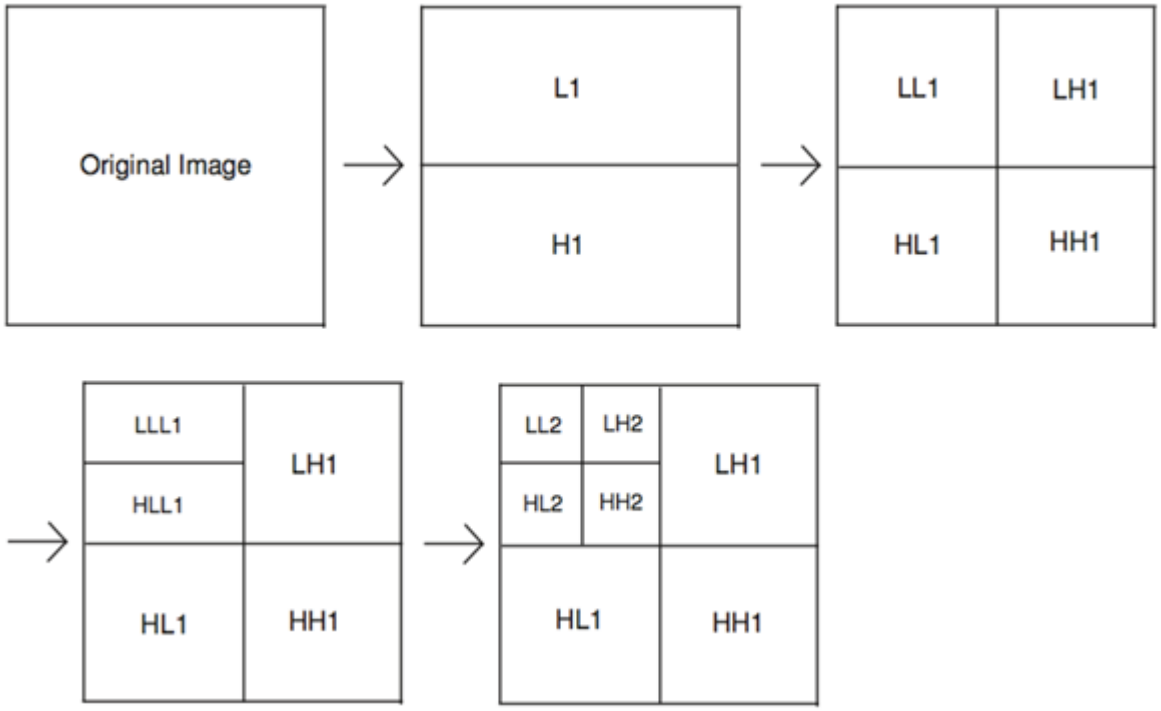


Figure 1.15 Différentes étapes (ligne et colonne) de la décomposition d'image à 2 niveaux [19]

Figure 1.16 Exemple d'ondelette de Daubechies (db1) du 1er ordre

Chapitre 2

Chapitre 2 Tatouage des images numériques

L'important développement des réseaux numériques pose le problème de la protection de la propriété intellectuelle des documents et a motivé de nombreuses recherches à s'intéresser au tatouage numérique. Cette technique consiste à insérer, à l'intérieur d'un document numérique, une marque invisible ou visible, contenant un code, robuste face à toute attaque susceptible de modifier la donnée tatouée. De nombreux algorithmes de tatouage ont été proposés cherchant à optimiser un compromis entre robustesse et invisibilité. Cependant, aucun d'eux ne satisfait un cahier de charges idéal. Actuellement, aucun modèle fonctionnel universel de tatouage n'a été défini.

2.1 Généralité sur le tatouage

2.1.1 Historique

Le terme « *digital watermarking* » (tatouage numérique) fut pour la première fois employé en 1992 par Tirkel et Osborne. En fait, le terme utilisé par Tirkel et Osborne est originaire du Japon : 'denshi sukashi' qui se traduit en anglais par « *Electronic watermark* » [20].

2.1.2 Stéganographie

La Stéganographie est la pratique de cacher un fichier, un message, image ou vidéo dans un autre fichier, message, image ou vidéo. Le mot stéganographie combine les mots *steganos*, qui signifie "couverts, cachés ou protégés" et *graphein* qui signifie « écriture » [21].

La Stéganographie comprend la dissimulation d'informations dans des fichiers informatiques. Dans la stéganographie numérique, les communications électroniques peuvent inclure le codage steganographic à l'intérieur d'une couche de transport, comme un fichier document, fichier image, programme ou protocole. Les fichiers

multimédias sont idéals pour la transmission de la stéganographie en raison de leur grande taille [21].

2.1.3 Les Types de tatouages

Il existe deux types de tatouage numérique : visible et invisible.

a. Tatouage numérique visible

Un tatouage numérique visible est une marque que l'on peut voir à l'œil nu, qui peut être un texte écrit en gros en plein milieu de l'image avec le nom de l'auteur ou bien seulement un copyright « © » en bas à droite de l'image. Le choix réside dans le compromis entre beauté et sécurité. En effet, un tatouage sera plus ou moins facile à enlever en fonction de la façon dont il a été inséré ou de la Robustesse du tatouage [22].

b. Tatouage numérique invisible

Le tatouage numérique dit « invisible » n'est pas visible à l'œil nu sans modification de l'image. Il est en priorité utilisé pour protéger une image sans gâcher sa qualité pour autant, mais la protection ne sera pas aussi efficace que dans le cas d'un tatouage visible [22].

Il existe aussi un tatouage numérique invisible qui permet de savoir si l'image en question a été modifiée sans l'autorisation du possesseur.

c. Robustesse du tatouage

- Un Tatouage numérique est appelé fragile s'il ne parvient pas à être détectable après la moindre modification. Les Tatouages fragiles sont couramment utilisés pour la détection de sabotage (preuve d'intégrité).
- Un Tatouage numérique est appelé semi-fragile s'il résiste à des simples transformations, mais échoue à la détection après des transformations malignes. Les Tatouages semi-fragiles sont couramment utilisés pour détecter les transformations malignes.

- Un Tatouage numérique est appelé robuste si elle résiste à une classe désignée de transformations. Les Tatouages robustes peuvent être utilisés dans des applications de protection contre la copie.

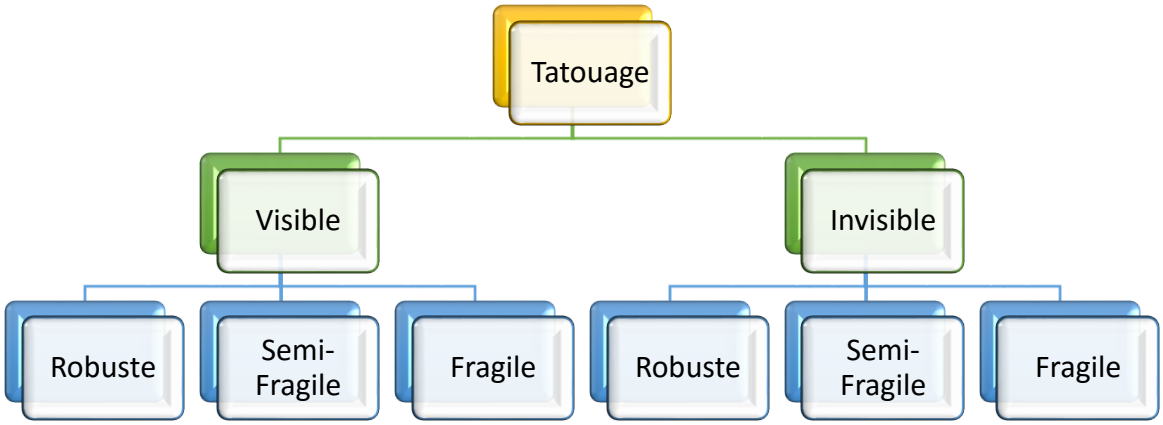


Figure 2.1 Classification du tatouage

2.1.4 Fonctionnement du tatouage numérique

Le tatouage numérique est une technique utilisée dans le traitement de signal numérique l'intégration des informations dans les données multimédia (cachées). Ces informations ne sont pas généralement visibles, seules les personnes qualifiées peuvent voir et extraire la marque. La figure suivante montre les différents niveaux du tatouage numériques.

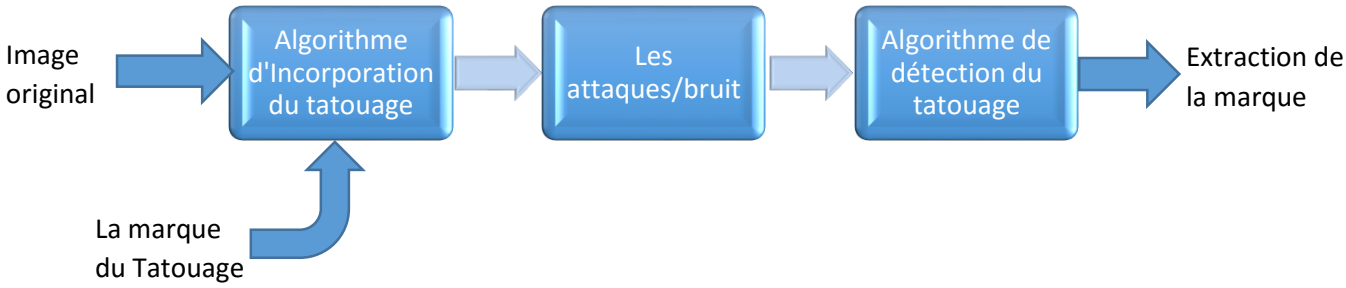


Figure 2.2 Les différents niveaux du tatouage numérique

2.2 Techniques du tatouage numérique d'image

Le tatouage numérique des images contient diverses techniques pour protéger le contenu numérique. Toutes ces techniques travaillent dans deux domaines, le domaine spatial qui agit directement sur les pixels. Il incorpore la marque dans l'image en modifiant les valeurs des pixels, la plus utilisée c'est la technique LSB.

Le deuxième domaine consiste à utilise les transformées (DFT, DWT, DCT), dont la plus utilisée dans ce domaine est la DCT.

2.2.1 Domaine spatial

Les premiers algorithmes de tatouage numérique des images ont été conçus pour opérer dans le domaine spatial. Un algorithme développé dans ce domaine consiste à insérer la marque en modifiant l'intensité lumineuse d'un nombre donné de pixels pour le cas des images à niveaux de gris. Cependant, dans le cas des images couleurs, une ou plusieurs composantes d'un espace colorimétrique quelconque vont être modifiées. Les méthodes les plus couramment utilisées dans ce domaine sont :

a. Technique de la somme de contrôle

En Anglais « *checksums* », une des premières techniques utilisées pour vérifier l'intégrité d'une image, visait à insérer des valeurs de « *checksums* » dans les bits les moins significatifs (LSB) des pixels de l'image.

L'algorithme proposé par Walton en 1995 consiste à sélectionner, de manière pseudo-aléatoire (en fonction d'une clé), des groupes de pixels et de calculer, pour chacun d'eux, une valeur de « *checksum* ». Ces valeurs sont obtenues à partir des nombres formés par les 7 bits les plus significatifs (MSB) des pixels sélectionnés, et sont ensuite insérées sous forme binaire au niveau des bits de poids faible. L'algorithme original [23] est détaillé par les étapes suivantes :

- Soit N suffisamment grand ;
- Diviser l'image en blocs de taille 8 × 8 pixels ;
- Pour chaque bloc Bi :
  - Définir un ordre de parcours pseudo-aléatoire (selon par exemple une clé secrète et l'indice du bloc Bi) des 64 pixels (p1, p2, ..., p64) ;
  - Générer une séquence pseudo-aléatoire de 64 entiers (a1, a2, ..., a64) du même ordre de grandeur que N ;
  - La valeur de *checksum* S est alors calculée de la manière suivante :

$$S = \sum_{j=1}^{64} \left(a_j . g(p_j) \right) mod(N) \qquad (2.1)$$

avec $g(p_j)$ est le niveau de gris du pixel p_j en ne tenant compte que des 7 MSB.

- Coder et crypter S en binaire ;
- Insérer la séquence binaire résultante au niveau des LSB des pixels du bloc

L'algorithme de vérification est le même que celui d'insertion. Il consiste à vérifier pour chaque bloc, la valeur de « *checksum* » recalculée à partir des MSB des pixels de l'image testée, avec celle de l'image originale codée au niveau des LSB [24].

b. Techniques du hachage

Le rôle principal des fonctions de hachage est de permettre de vérifier l'intégrité d'un document numérique sans avoir recours au document l'original. Une fonction de hachage opère généralement sur un message M de longueur arbitraire pour fournir une valeur de hachage h de taille fixe. Pour qu'une telle fonction soit considérée comme sûre, elle doit vérifier les propriétés suivantes :

- Il est « facile » de calculer h connaissant M,
- Il est « difficile » de retrouver M connaissant h,
- Il est « difficile » de trouver un message M' (différent de M)

  en ayant comme valeur de hachage h' = h.

En d'autres termes, une fonction de hachage sert à produire un condensé (ou une empreinte) unique, représentatif du document original. Il existe de nombreuses fonctions de hachage parmi lesquelles on peut citer : MD-4, MD-5 (Message Digest), CRC-32 (32 bits Cyclic Redundancy Check), SHA-1 (Secure Hash Algorithm), etc. [25]

Les deux méthodes « *Row-Coumn hash function* » et « *Block based hash function* » proposée par Wolfgang et Delp. [26]. La 1er Méthode consiste à calculer une valeur de hachage pour chaque ligne et chaque colonne de l'image originale. Lorsque l'on souhaite vérifier l'intégrité d'une image, on recalcule les valeurs de hachage des lignes et des colonnes de l'image à tester et on les compare avec celles de l'image originale. Pour localiser les éventuelles disparités, il suffit d'identifier les lignes et les colonnes qui sont différentes. Cependant, dans le cas où plusieurs zones de l'image ont été modifiées, on n'est plus capable de les localiser sans ambiguïté, c'est-à-dire que des régions intègres seront considérées comme altérées, ce qui réduit considérablement l'intérêt de cette technique [26].

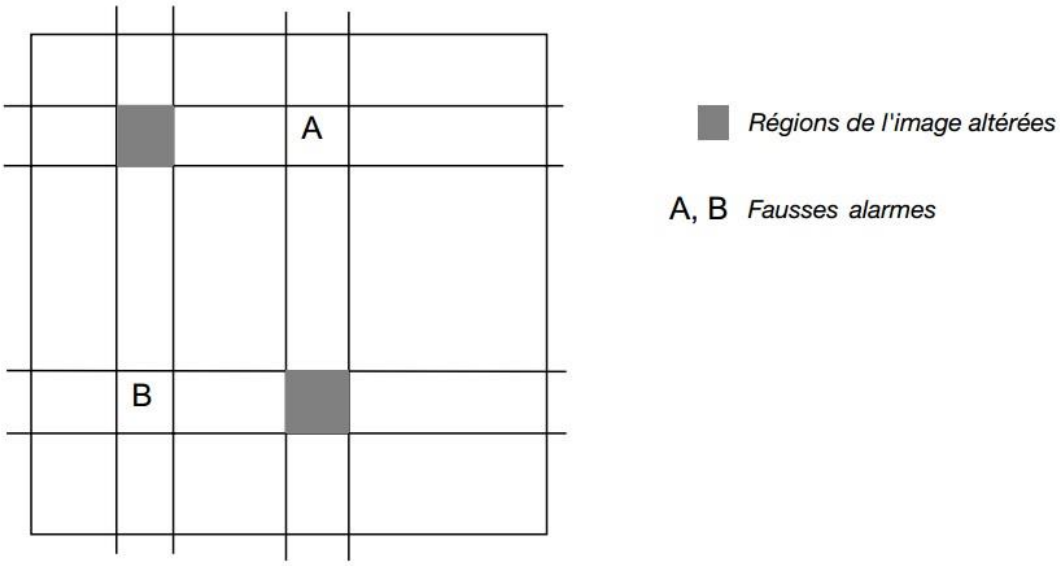


Figure 2.3 Ambiguïté dans la localisation des régions altérées de l'image [26]

Le principe de la 2eme méthode est similaire à celui décrit précédemment, à la différence près qu'il n'opère plus sur les lignes ou les colonnes de l'image, mais sur les blocs. Ainsi, lorsque l'on constate des différences dans les valeurs de hachage, il suffit de se reporter aux blocs concernés pour localiser les zones de l'image qui ont été manipulées. Les fonctions de hachage ont la particularité d'être extrêmement sensibles à la moindre variation ; en effet, il suffit de modifier la valeur d'un pixel pour changer radicalement la valeur de hachage du bloc associé. Elles ne permettent donc pas de distinguer les manipulations malveillantes des manipulations bienveillantes [26].

c. Technique de Patchwork

Patchwork est une technique de dissimulation de données développée par Bender *et al.* [27].

Elle fonctionne en intégrant invisiblement une statistique spécifique, avec une distribution gaussienne, dans l'image originale. Deux ensembles (le premier A et le second B) des pixels, ou des patches, de l'image sont choisis. Ensuite, l'algorithme fonctionne en des points légèrement lumineux en A, et des points sombre en B [27].

Pour déterminer les points que nous devons toucher à l'image, un générateur de nombres pseudo-aléatoires est utilisé, avec une clé secrète partagée par l'émetteur et le récepteur, car l'algorithme de décodage doit visiter les mêmes points dans le même ordre pour extraire les données intégrées [27].

$$S(n) = \sum_{i=1}^{n} a_i - b_i \qquad (2.2)$$

où a_i et b_i sont les i^{eme} points, dans le patch A et B et n représente le nombre de pixels dans les deux ensembles.

Après avoir fait n fois les opérations :

$$a_i = a_i + delta\ b_i \qquad (2.3)$$

où 'delta' est la distance entre les pixels dans le patch. La valeur attendue va être décalée « shift » vers la droite de

$$2(delta)n \qquad (2.4)$$

Donc, si n est assez grand, et si on calcule la valeur de S(n) pour une image, nous allons avoir une certitude assez forte si l'image a été tatoué ou non par patchwork [27].

En effet, nous pouvons attribuer un (niveau de sécurité) liée à n au processus de codage, ce qui représente la confiance que nous pouvons donner aux réponses de décodage. Le problème avec l'augmentation de n pour obtenir une marque plus forte, est que plus on répète le processus, plus le tatouage devient visible.

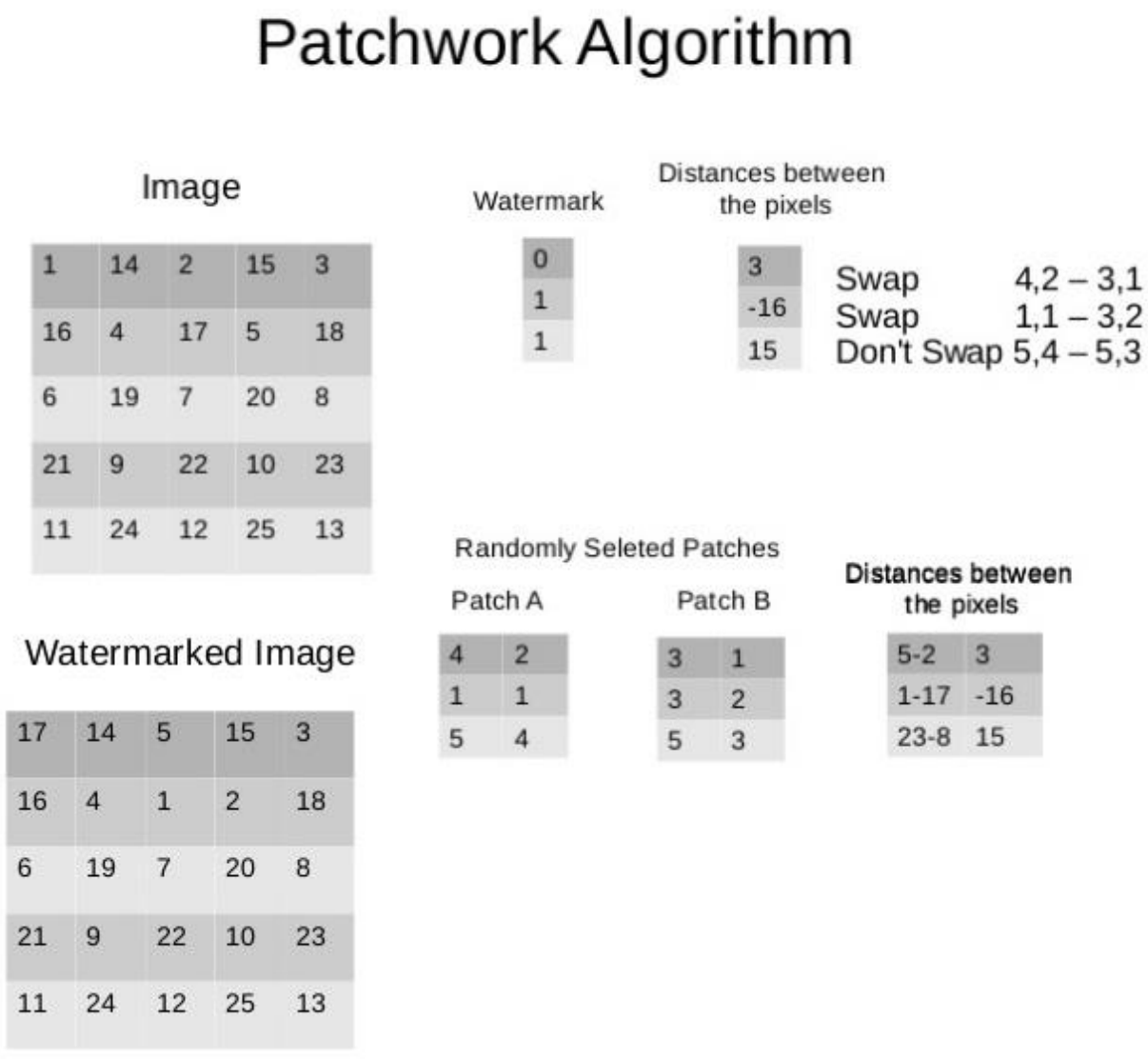


Figure 2.4 Exemple de la technique patchwork [27]

2.2.2 Domaine fréquentiel

D'un autre côté, les techniques de tatouage numérique conçues pour travailler dans le domaine fréquentiel sont plus robustes et plus complexes, mais elles sont largement utilisées. La marque est insérée en modulant les coefficients de la transformée fréquentielle. Parmi les transformées utilisées dans les algorithmes du tatouage numérique des images, on peut citer : La transformée en ondelette discrète (DWT) et la transformée en cosinus discrète (DCT).

a. Techniques de la DWT

L'un des premiers travaux de tatouage numérique des images basé sur la transformée en ondelettes est celui de Kundur et Hatzinakos [28]. L'image originale est décomposée en utilisant la DWT jusqu'au niveau de résolution L. La même procédure est faite pour la marque sauf que le niveau de résolution est égal à un. Les coefficients de détails de tous les niveaux de résolution, sont divisés en blocs de taille 8x8 pixels. Ils sont ensuite modifiés par l'insertion des coefficients DWT de la marque. Un seuil adaptatif est utilisé dans cette étape. Dans cet algorithme la marque utilisée est une image binaire de taille 16 x 16 pixels. Les résultats expérimentaux montrent la robustesse du schéma contre la compression JPEG, l'ajout de bruit et le filtrage [28].

Xia et al. [29] ont présenté un algorithme de tatouage numérique basé sur la transformée en ondelettes de Haar. L'image originale est décomposée en deux niveaux de résolution. La marque utilisée est une séquence pseudo-aléatoire (bruit gaussien) : elle est ajoutée aux coefficients des six sous-bandes sauf les coefficients des basses-fréquences (LL). Lors de l'étape d'extraction, la détection de la marque se fait par la méthode d'inter-corrélation entre l'image originale et l'image tatouée. Les résultats expérimentaux montrent que cet algorithme est robuste contre l'ajout de bruit et la en utilisant l'algorithme de codage EZW [29].

Xianghong *et al.* [30] ont proposé d'insérer la marque dans la sous-bande d'approximation LL. Toutefois, l'insertion dans cette sous-bande cause une grande distorsion. Pour remédier à ce problème, une transformée vectorielle (VT) est utilisée afin de transformer les coefficients basses fréquences de la DWT. Les résultats expérimentaux montrent la robustesse contre l'ajout de bruit, le filtrage, la correction gamma, l'accentuation de la netteté, la mise à l'échelle et la compression JPEG [30].

b. Techniques de la DCT

Le premier algorithme de tatouage numérique des images efficace basé sur cette transformée en cosinus discrète (DCT) a été développé par Koch et al. [31]. Dans cet algorithme, l'image originale est divisée en blocs de taille 8x8 pixels puis la DCT est calculée pour chaque bloc. Ensuite, une paire parmi douze paires de coefficients de moyennes fréquences d'un bloc est sélectionnée. Le choix du bloc et de la paire se fait aléatoirement. Selon la valeur du bit à insérer, les coefficients

sont modifiés de telle sorte que la différence entre eux soit positive ou négative. Les résultats expérimentaux montrent la robustesse de ce schéma contrela compression JPEG avec un facteur de qualité supérieure à 50% [31].

Bors et Pitas [32] ont présenté un schéma qui modifie les coefficients DCT. Après avoir divisé l'image originale en blocs de taille 8x8 pixels, certains blocs de moyennes fréquences sont sélectionnés conformément à la décision du classificateur du réseau gaussien puis modifiés. Bors et Pitas [32] ont utilisé des images couleurs pour présenter les résultats expérimentaux où la composante de luminance de l'espace colorimétrique YUV a été utilisée pour l'insertion de la marque. Le schéma proposé est robuste contre la compression JPEG [32].

Tao et Dickinson [33] ont introduit le premier algorithme de tatouage numérique adaptatif dans le domaine de la DCT en exploitant les caractéristiques du système visuel humain (HVS). Il assigne à chaque région spatiale un indice de sensibilité au bruit et la marque est insérée dans chaque bloc de la DCT en fonction de cet indice. L'algorithme de classification exploite les propriétés de masquage de la luminance, le contour et la texture du système visuel humain. Pour résister au bruit, la marque est insérée dans N composantes de la DCT. Les résultats expérimentaux montrent la robustesse de cet algorithme contre la compression JPEG avec un facteur de qualité supérieure à 5% et l'ajout de bruit [33].

Tableau 2.1 Récapitulatif des techniques du tatouage numérique

Propose par	Année	Domain	Visibilité	Robustesse
Walton	1995	Spatial (LSB)	Invisible	Fragile
Bors et Pitas	1996	Fréquentiel (DCT)	Invisible	Robuste
Bender	1996	Spatial (Patchwork)	Invisible/Visible	Robuste
Kundur et Hatzinakos	1997	Fréquentiel (DWT)	Invisible/Visible	Semi-Fragile
Tao et Dickinson	1997	Fréquentiel (DCT)	Invisible	Robuste
Xia	1997	Fréquentiel (DWT)	Invisible	Robuste
Koch	1999	Fréquentiel (DCT)	Invisible	Robuste
Xianghong	2004	Fréquentiel (DWT)	Invisible	Robuste

2.3 Attaques

Comme il y a des recherches pour la conception d'un algorithme de tatouage numérique des images le plus optimal, il y a aussi des recherches pour trouver les vulnérabilités et les faiblesses des algorithmes.

Il existe dans la littérature deux principales classifications détaillant, de manière plus précise, les différentes attaques que peut subir une image. La première classification a été proposée en 1999 par Hartung et al. [34] : on classifie les attaques qui ne détériorent pas, de manière significative, la fidélité perçue de l'image originale. Cette classification distingue les quatre catégories suivantes (voir Figure 2.5) :

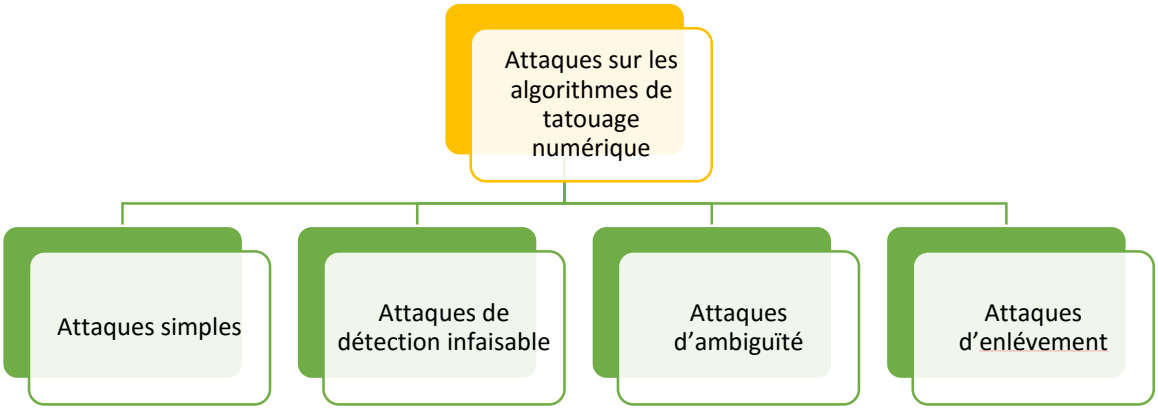


Figure 2.5 Classification des attaques selon Hartung et al. [34]

La deuxième classification est celle proposée par Voloshynovskiy et al. [35]. Elle est représentée à la figure 2.5.

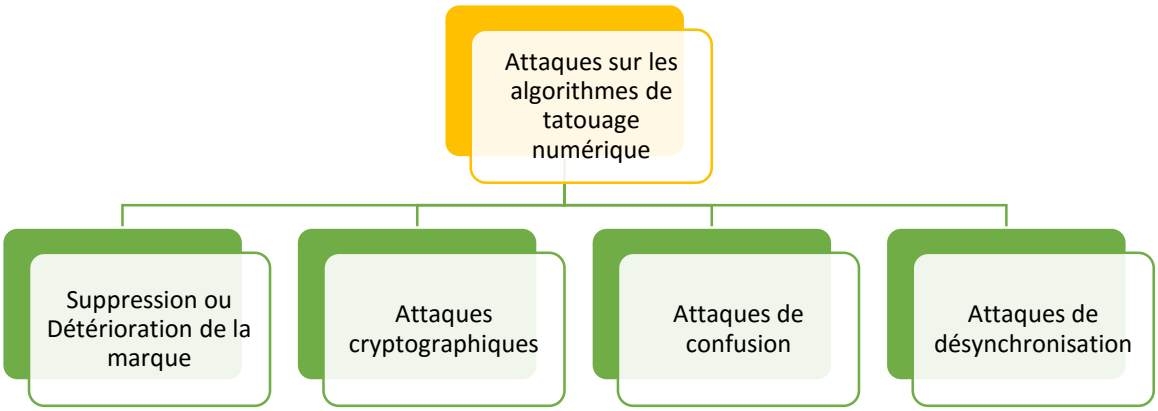


Figure 2.6 Classification des attaques selon Voloshynovskiy et al. [35]

2.4 Applications du tatouage numérique

Les applications du tatouage numérique sont nombreuses parmi celle-ci. On peut citer :

- Protection du droit d'auteur
- Les empreintes digitales
- La limitation de la copie

La principale raison pour laquelle le tatouage d'images est très utile réside dans la protection de la propriété intellectuelle. Ainsi, la protection du droit d'auteur a été

la première application du tatouage et elle est toujours le plus populaire. Néanmoins, des moyens supplémentaires sont nécessaires afin de faciliter l'application, comme les outils techniques de protection, le propriétaire peut intégrer une marque numérique afin de protéger son innovation partout dans le monde. Cette opération est nécessaire de nos jours, après la grande explosion d'Internet qui rend l'accès à l'information de plus en plus facile [36].

Une autre application du tatouage d'images est l'empreinte digitale, cette technique est similaire au tatouage privé, mais la différence réside dans le niveau de sécurité. Les empreintes sont intégrées pour chaque livraison d'images, elles peuvent être associées à un réseau serveur/client (Peer2Peer). Dans ce cas, l'empreinte et située du côté serveur. Le serveur connaît l'identité du client, et l'incorpore dans l'image avant la livraison [36].

Cette application est utile pour tirer les ressources de copies illégales. Ainsi, le propriétaire peut intégrer des marques différentes dans les copies fournies à différents clients. Cette technique peut être comparée à l'incorporation d'une série de nombre liée à l'identité du client dans les images protégées. Cela permet la protection intellectuelle de l'image afin d'identifier les clients qui ont brisé leur accord en fournissant les données à des tierces parties [36].

Chapitre 3

Chapitre 3 Implémentation et Résultats

Dans ce chapitre, nous allons implémenter sur circuit FPGA certaines techniques décrites dans les deux premiers chapitres en utilisant *Xilinx System Generator* (XSG) et la carte de Développement Nexys-4. Le choix d'utiliser XSG et pas un langage de description du matériel (HDL) est motivé par la facilité de programmation et de simulation avec cet outil dans l'environnement SIMULINK.

3.1 Xilinx System Generator (XSG)

Xilinx System Generator (XSG) est un outil développé *Xilinx* pour la conception d'algorithmes de traitement numérique du signal (DSP) dans l'environnement MATLAB/SIMULINK, en vue de les implémenter sur un circuit FPGA. Une expérience antérieure avec des circuits FPGA ou des méthodologies de conception RTL ne sont pas nécessaires lorsqu'on désire utiliser XSG. Les diagrammes sont construits, en utilisant un ensemble spécifique de blocs SIMULINK. Toutes les étapes de mise en œuvre de FPGA, y compris la synthèse et le routage sont réalisées automatiquement pour générer le fichier de programmation de FPGA [37].

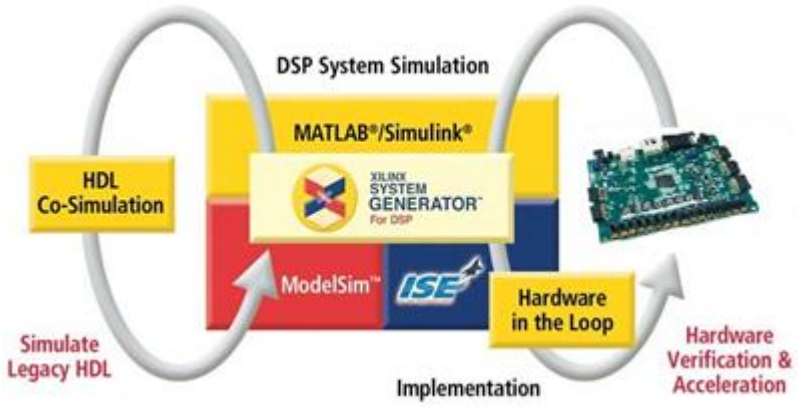


Figure 3.1 Outil de programmation haut-niveau [37]

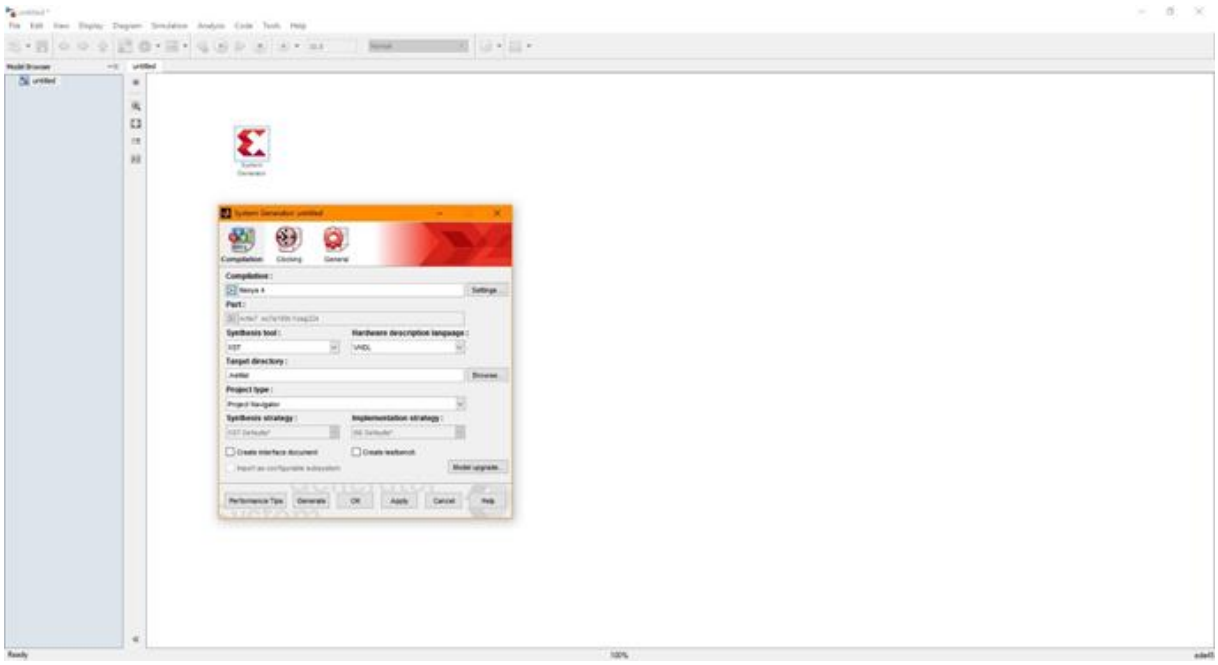


Figure 3.2 Interface XSG (Xilinx ISE 14.7)

Plus de 90 blocs DSP sont fournis dans le *blockset Xilinx* dans SIMULINK. Ces blocs comprennent les blocs communs de construction DSP, tels que des additionneurs, multiplicateurs et des registres. Sont également inclus un ensemble de blocs complexes de construction DSP, tels que la correction d'erreur, des FFT, des filtres et des mémoires. Ces blocs exploitent les générateurs de base *IP Xilinx* pour obtenir des résultats optimisés pour le périphérique sélectionné [37].

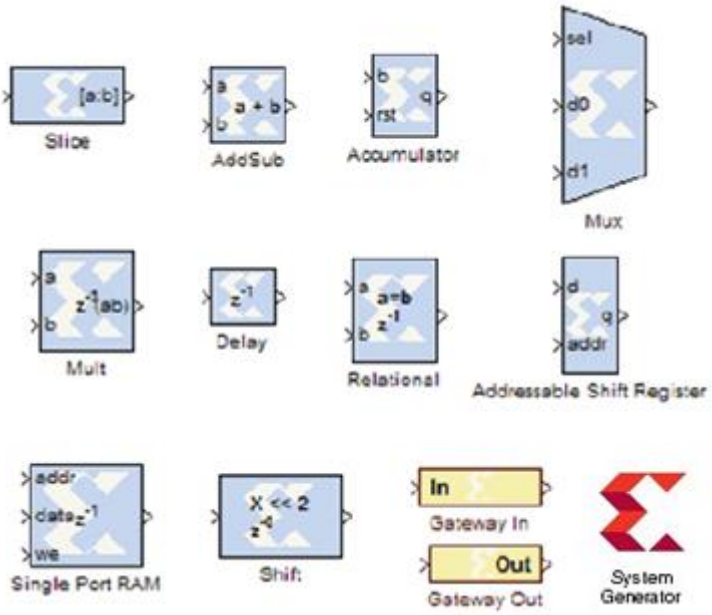


Figure 3.3 les blocs XSG

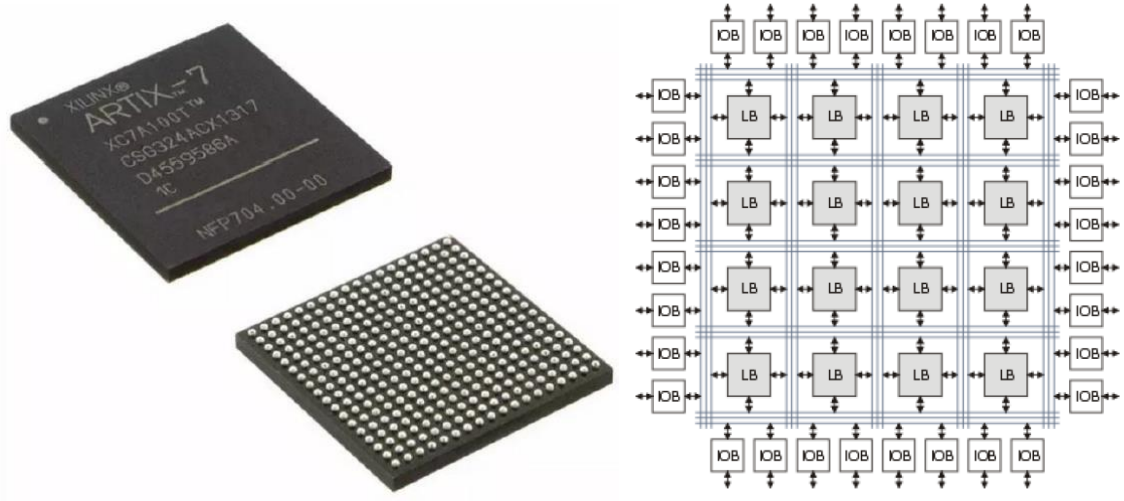
3.2 Matériel utilisé

Dans ce projet, nous allons utiliser avec un circuit FPGA (Nexys 4)

3.2.1 Field-Programmable Gate Array (FPGA)

FPGA est un circuit intégré conçu pour être configuré par le concepteur après la création de la conception. La configuration du FPGA est en général spécifiée en utilisant un langage de description de hardware (HDL), similaire à celui utilisé pour un circuit intégré à application spécifique (ASIC) [38].

La plupart des nouveaux circuits FPGA sont basés sur des cellules SRAM, aussi bien pour le routage du circuit que pour les blocs logiques à interconnecter. Un bloc logique configurable (CLB), est de manière générale, constitué d'une table de correspondance « *LUT ou Look-Up-Table* » et d'une bascule « *Flip-Flop* ». La LUT sert à implémenter des équations logiques ayant généralement 4 à 6 entrées et une sortie. Elle peut toutefois être considérée comme une petite mémoire, un multiplexeur ou un registre à décalage. Le registre permet de mémoriser un état (machine séquentielle) ou de synchroniser un signal « *pipeline* ». [38]



(a) (b)

Figure 3.4 : (a) Puce FPGA (la famille ARTIX-7) [37], (b) Structure type d'un FPGA [38]

3.2.2 La carte Nexys-4

La carte Nexys4 est une plate-forme complète, prête à l'emploi du développement de circuit numérique basé sur les dernières Artix-7 (XC7A100T-1CSG324C) (FPGA) de *Xilinx*. Avec sa grande capacité, mémoires externes, port USB, Ethernet, et d'autres ports, la Nexys4 peut accueillir des circuits allant de circuits combinatoires d'introduction aux processeurs embarqués puissants. Plusieurs périphériques intégrés, y compris un accéléromètre, capteur de température, microphone numérique, un amplificateur de haut-parleur et beaucoup de périphériques d'entrées sorties (E/S) permettent au Nexys4 à être utilisé pour une large gamme de modèles sans avoir besoin d'autres composants.

Le FPGA Artix-7 est optimisé pour la logique de haute performance, il offre une plus grande capacité, des performances plus élevées, et plus de ressources que les modèles précédents. Les ressources de la Artix-7 100T [39] sont :

- 15 850 tranches logiques, chacune avec quatre LUT à 6 entrée et 8 bascules (Flip-Flops)
- 4 860 Kbits de block RAM rapides
- 6 tuiles de gestion d'horloge, chacune avec PLL
- 240 tranches DSP
- Horloge interne avec une vitesse dépassant le 450Mhz

Figure 3.5 La carte Nexys-4

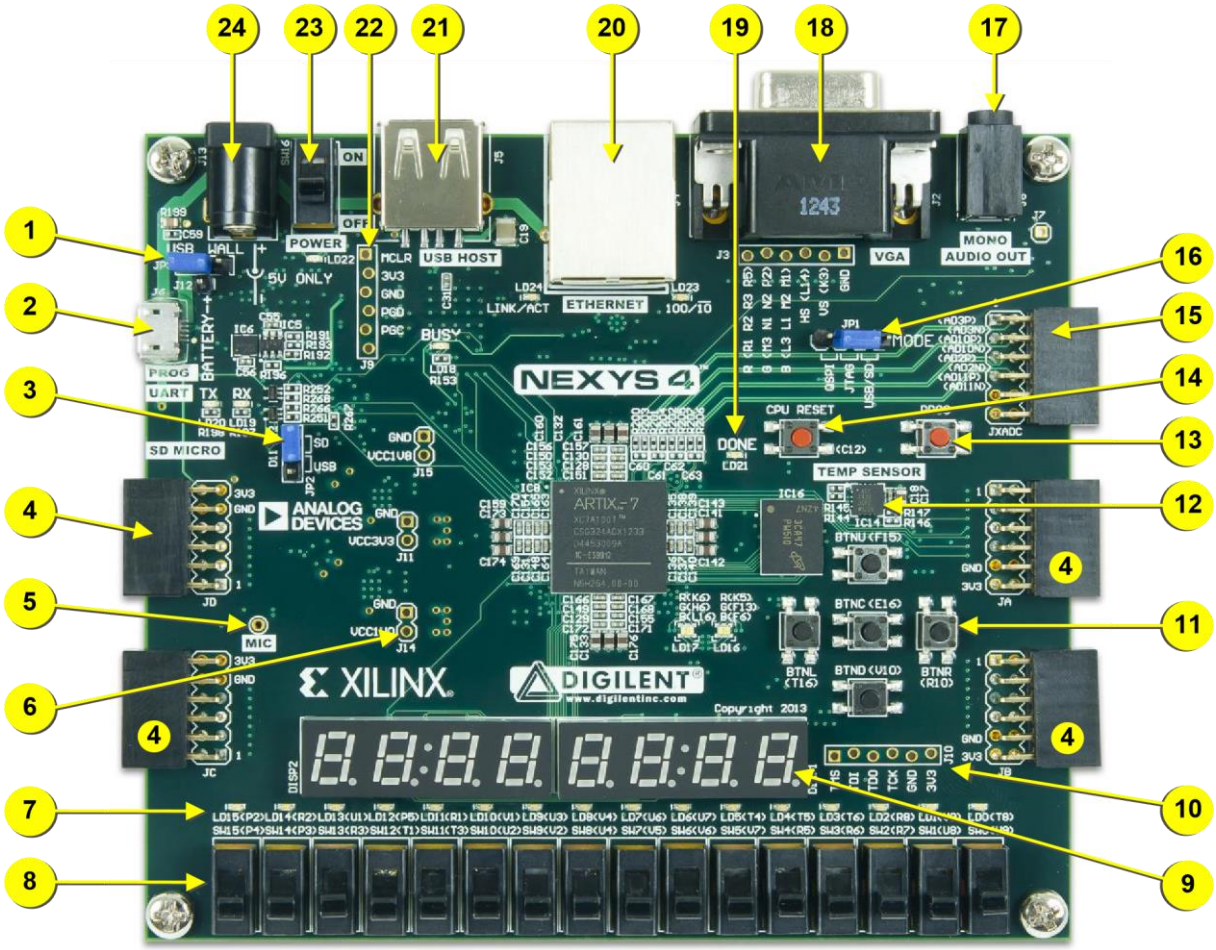


Figure 3.6 Les composants de la carte Nexys-4 [39]

Tableau 3.1 Les caractéristiques de la Nexys-4 [39]

Légende	Description du composant	Légende	Description du composant
1	Jumper de la source d'alimentation	13	Bouton reset la configuration FPGA
2	Adept USB (JTAG & UART)	14	Bouton reset CPU
3	Jumper de configuration externe SD / USB	15	Connecteur Pmod signal Analogique
4	Connecteur Pmod	16	Jumper de mode de Programmation
5	Microphone	17	Connecteur Audio
6	Point de teste d'alimentation	18	Connecteur VGA
7	LEDs (16)	19	LED programme FPGA terminé(DONE)
8	Interrupteurs (16)	20	Connecteur Ethernet
9	8 afficheur 7-seg	21	Connecteur USB
10	Port JTAG pour les câbles externe	22	Port PIC24 (Programmation)
11	5 bouton poussoirs	23	Bouton power
12	Capteur de température	24	Connecteur d'alimentation

3.3 Implémentation

Après avoir présenté la théorie de certaines techniques de traitement d'image, nous passons à la partie simulation en créant un petit laboratoire de traitement d'image qui implémente les techniques suivantes :

3.3.1 Implémentation conversion couleur en niveau de gris

La méthode utilisée est la méthode pondérée. Le bloc « *Convert to a serial stream* » fait la conversation du 2D vers 1D, c'est-à-dire transformer une matrice en un vecteur de lignes, en même temps, il fait la conversion du type de données en double. Les blocs « *R_In, G_In, B_In* » sont des blocs de XSG nommés 'Gateway In, Gateway out', qui font la liaison entre les blocs de classiques de SIMULINK et ceux de XSG. Le bloc « *Recreate Image* » va recréer l'image en fraisant la conversion du type de données à 'uint8' et recréer la matrice. Le bloc « *System Generator* » est le bloc de contrôle qui permet de choisir le type de la cible FPGA ainsi que la commande à exécuter. Ces blocs seront toujours utilisés avec n'importe quelle technique de traitement d'image. Finalement, le bloc « *RGB2Grayscale* » qui fait traduire l'équation (1.2) en hardware en utilisant XSG.

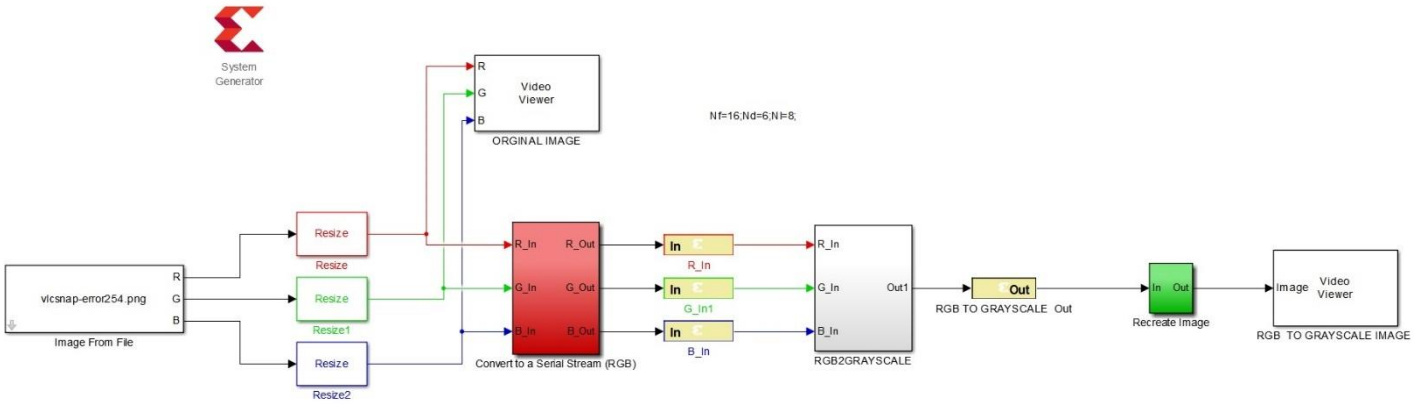


Figure 3.7 Schéma de conversion couleur en niveau de gris (XSG)

3.3.2 Implémentation conversion négative

Dans un premier temps, nous allons toujours utiliser les mêmes blocs importants pour que le changement se fasse au niveau du bloc « *Negative* » qui va traduire l'équation (1.3). Nous utilisons des blocs de XOR.

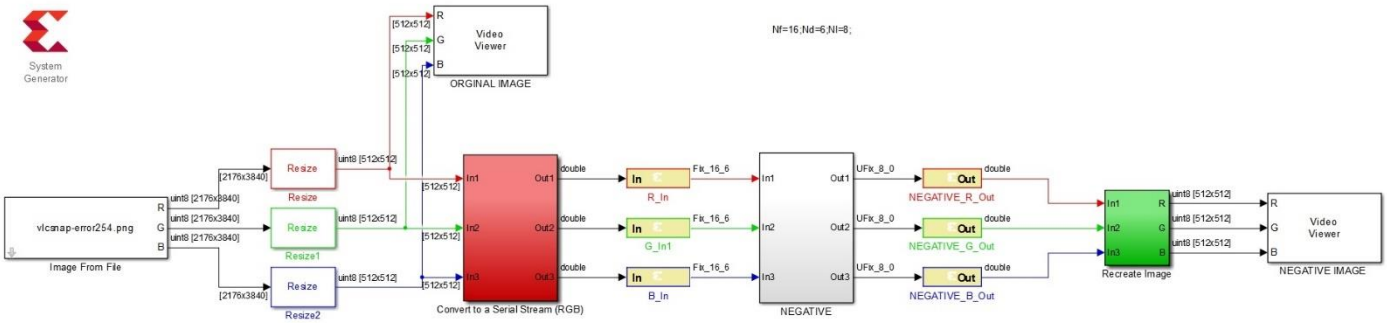


Figure 3.8 Schéma de conversion négative (XSG)

3.3.3 Implémentation seuillage

Avant de commencer le seuillage, il faut choisir le seuil. Nous utilisons la méthode la plus simple qui se base à l'histogramme (T=90), puis nous traduisons l'équation (1.4) pour la mettre sous forme de blocs. Dans le bloc « *Thersholding* », nous utilisons des blocs relationnels qui comparent les pixels avec le seuil puis le bloc de multiplexage (Mux) va choisir la valeur (1 ou 0) pour avoir la nouvelle valeur de pixel.

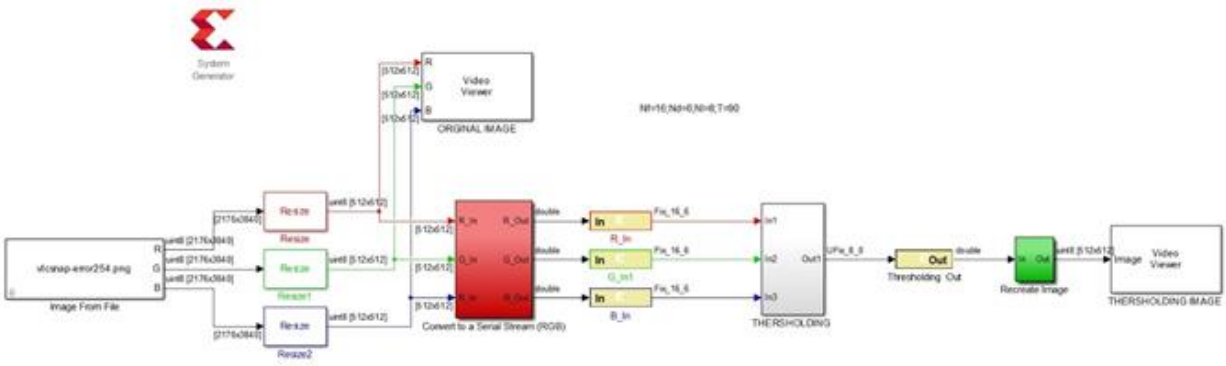


Figure 3.9 Schéma du seuillage (XSG)

3.3.4 Implémentation détection de contour

La détection de contour se base sur plusieurs opérateurs. Dans ce cas, nous allons nous baser sur l'opérateur Sobel et nous l'appliquons sur les lignes et les Colonnes (SobelXY). Dans la bibliothèque de XSG, ou il y a un bloc « 5x5 filtre » qui contient déjà les paramètres du Filtre Sobel prédéfini. Dans cette technique, nous ajoutons un bloc 'Delay' car nous utilisons des filtres qui causent un décalage à l'image et pour l'éliminer nous utilisons ces blocs.

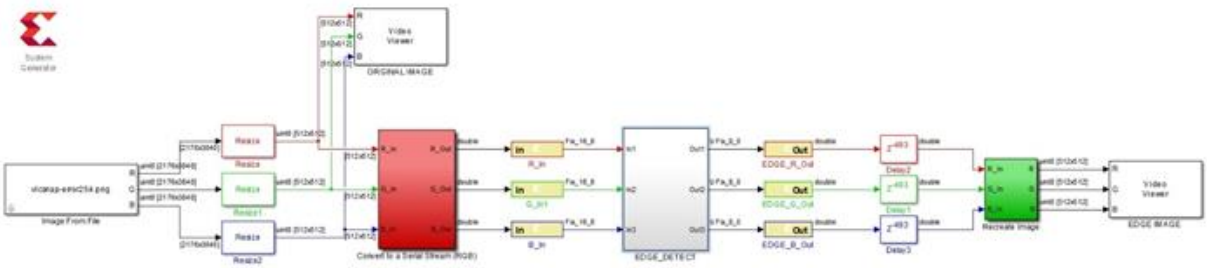


Figure 3.10 Schéma détection de contour (XSG)

3.3.5 Implémentation tatouage numérique

L'algorithme utilisé est celui de Xia [29] auquel nous appliquons des modifications. Il se résume dans les étapes suivantes :

- Décomposer l'image originale et la marque avec l'ondelette (db2) à un seul niveau.
- Appliquer l'équation suivante pour additionner les approximations (LL).

$$LL' = LL_I + (w \times LL_m) \qquad 0 < w < 1 \qquad (3.1)$$

où w est le poids de tatouage avec lequel nous pouvons déterminer si l'image est visible ou non.

S'il tend-vers 0, c'est un tatouage invisible et vice-versa.

- Reconstruire avec la même ondelette (db2) sauf que pour (LL), nous utilisons le nouveau (LL').

Comme XSG ne contient pas de bloc (DWT), nous devons les créer en utilisant des filtres FIR à base des coefficients (db2), puis nous traduisons l'équation (3.1). Pour la (IDWT), nous appliquons le processus inverse et nous devons créer le bloc. Pour l'extraction ou la détection de la marque), nous appliquons le processus inverse avec la décomposition de l'image originale.

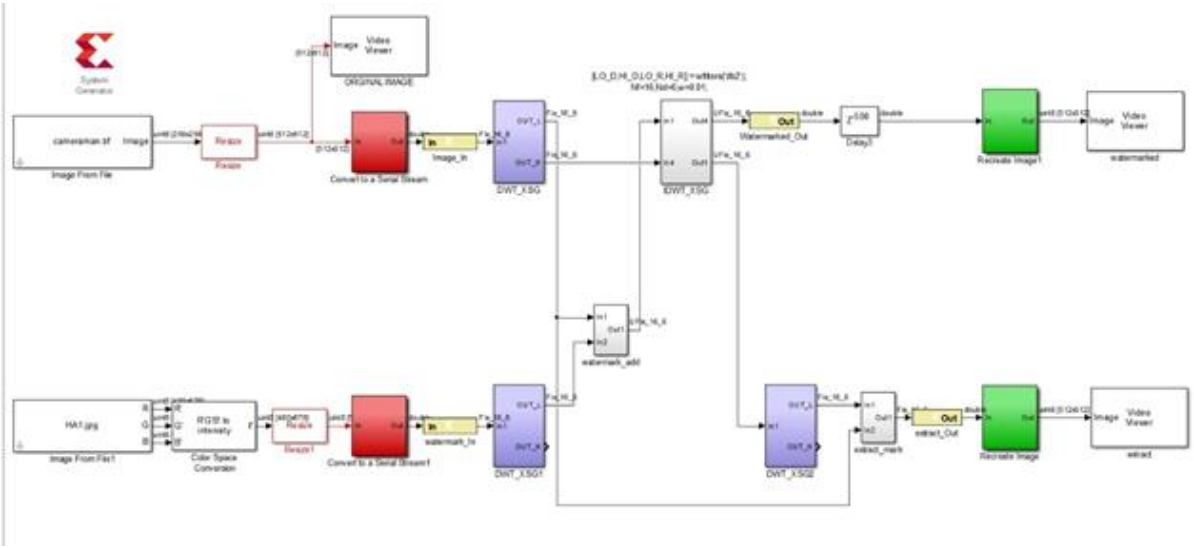


Figure 3.11 Schéma du tatouage numérique (XSG)

3.3.6 Laboratoire de Traitement d'images

Après avoir réalisé tous ces techniques, nous devons les rassembler dans un seul fichier pour avoir un laboratoire de traitement d'image. Afin de rendre la comparaison entre le logiciel (software) et le matériel (hardware) plus complète, nous devons créer trois laboratoires sous trois différentes plateformes (*Simulink*, XSG, *hardware Co-Sim*).

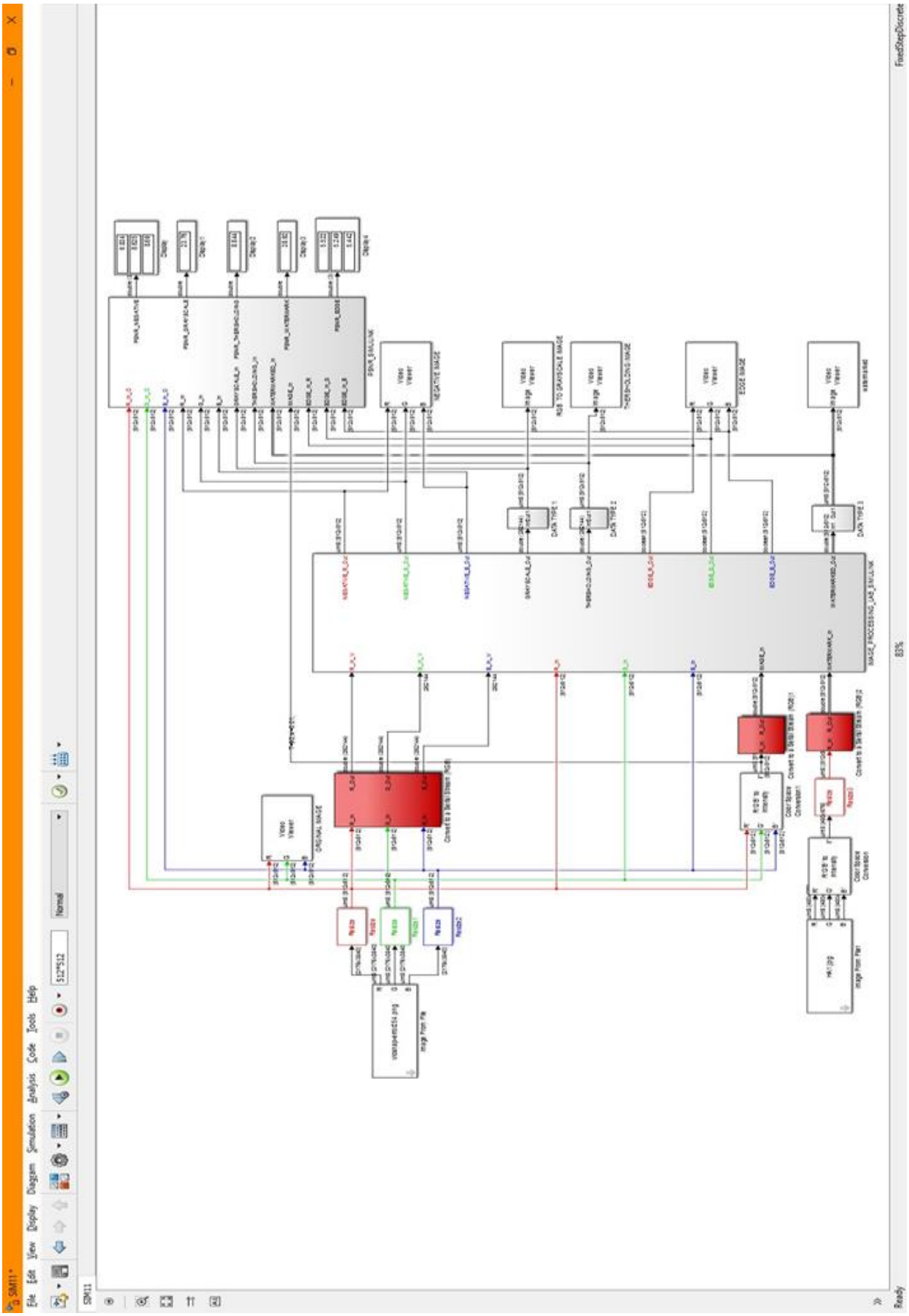


Figure 3.12 Laboratoire de traitement d'images (Simulink)

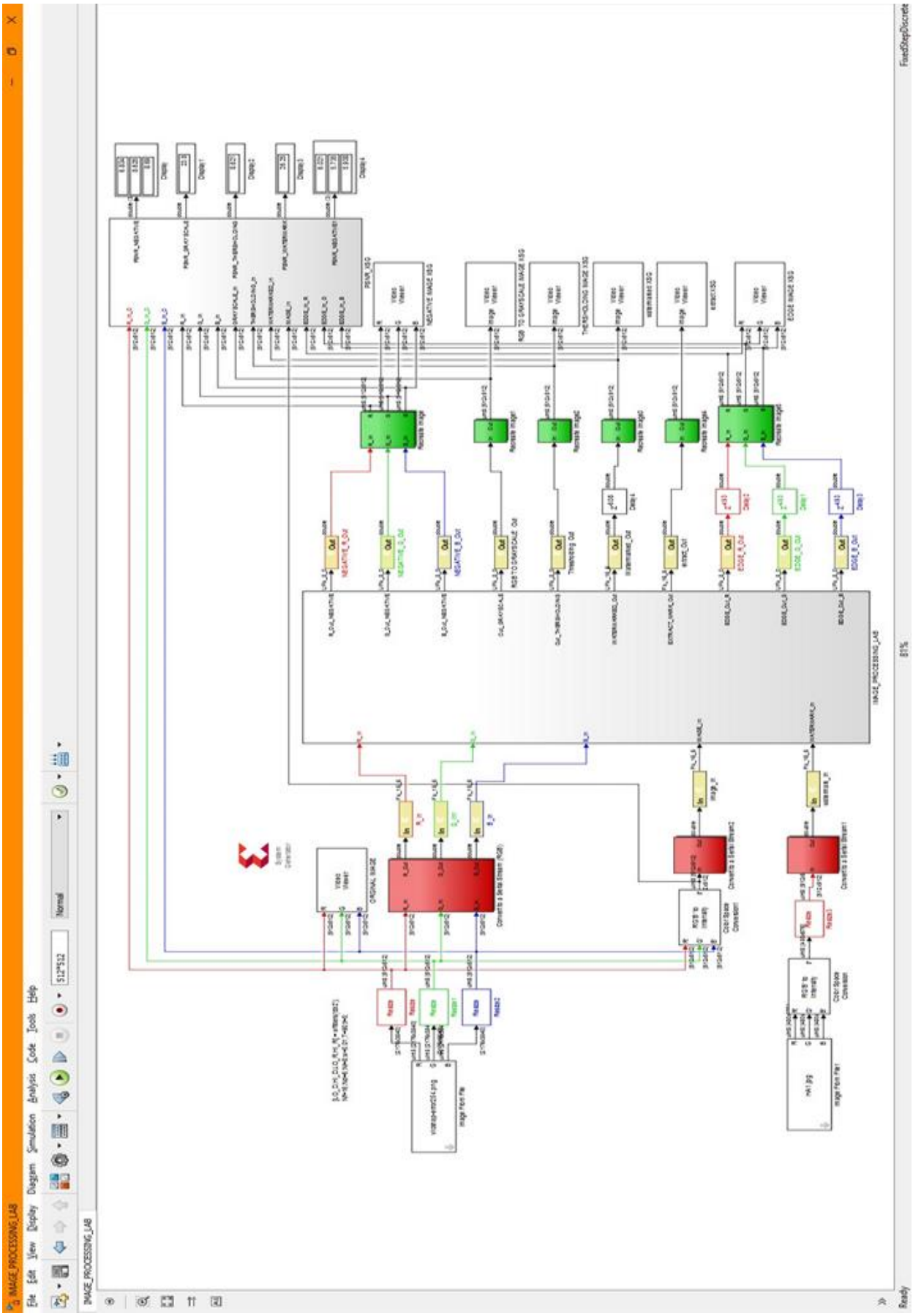


Figure 3.13 Laboratoires de traitement d'images (XSG)

Pour générer le bloc (*hw_Co-Sim*), il faut passer par l'étape de compilation de *Bitstream* pour savoir si nous avons suffisamment de ressources pour passer à l'étape de compilation hardware.

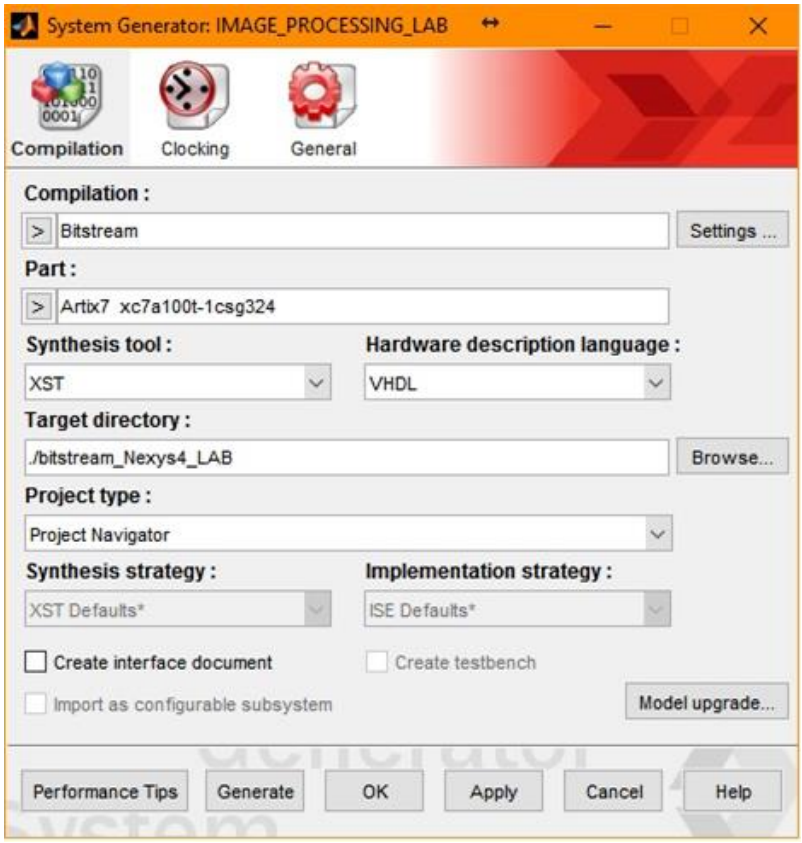


Figure 3.14 Compilation de Bitstream

Tableau 3.2 Résultats de compilation Bitstream

Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	2,068	126,800	1%
Number of Slice LUTs	3,106	63,400	4%
Number of occupied Slices	1,213	15,850	7%
Number of bonded IOBs	177	210	84%
Number of RAMB18E1	27	270	10%
Number of BUFG	1	32	3%
Number of OLOGICE2	24	300	8%
Number of DSP48E1s	39	240	16%
Average Fanout of Non-Clock Nets	2.41		

Figure 3.15 Compilation Hardware

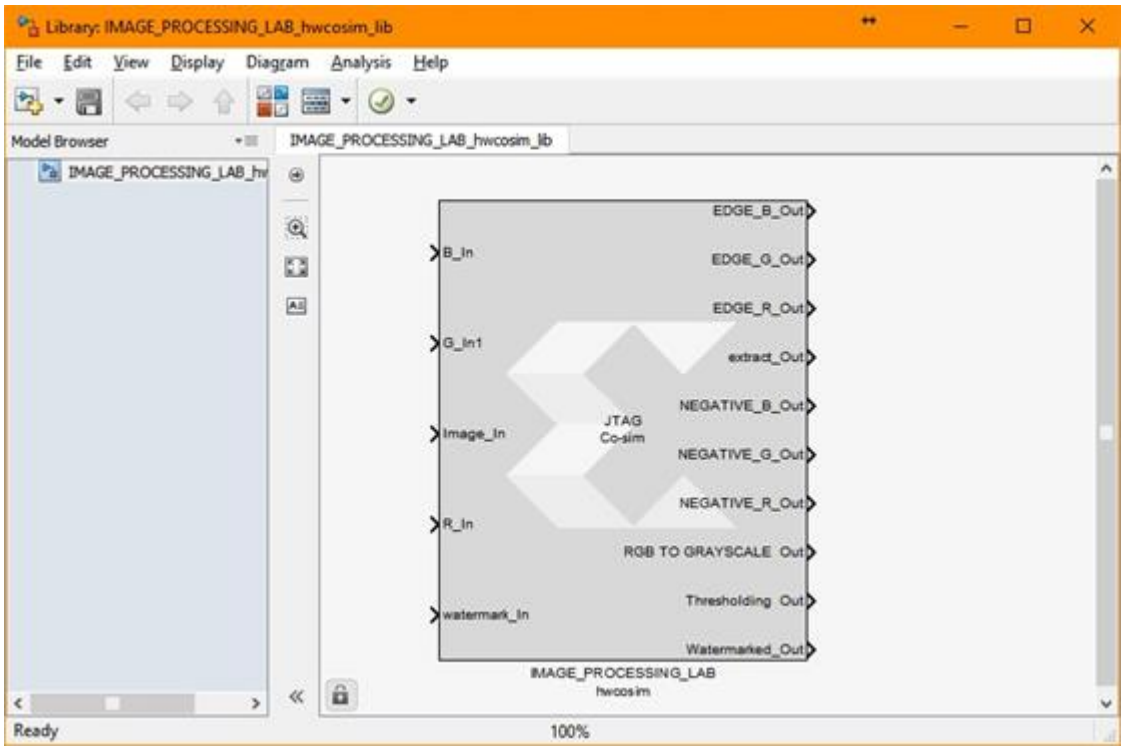


Figure 3.16 Bloc hw_cosim généré.

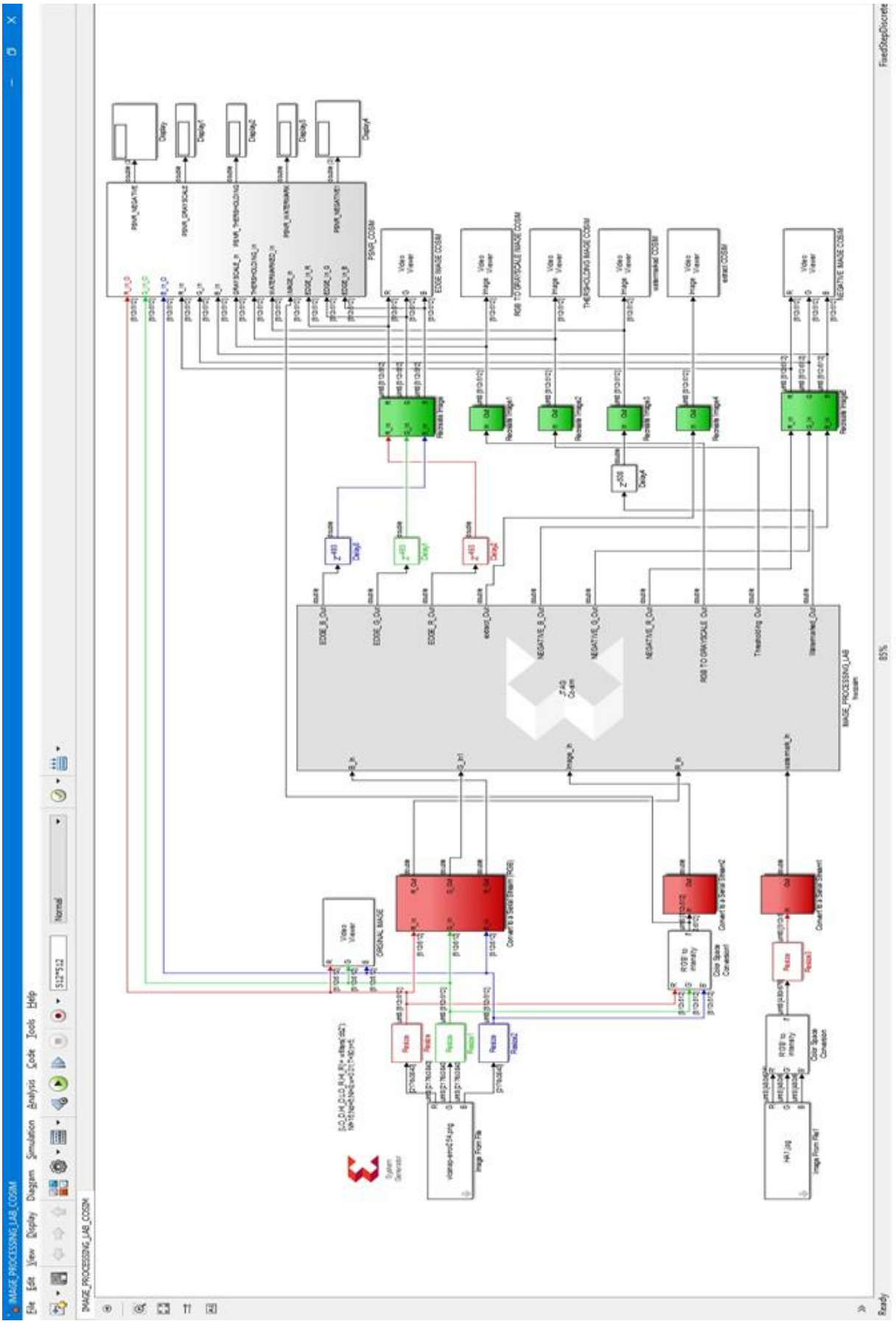
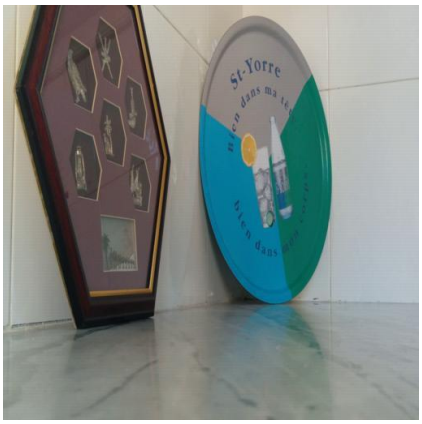
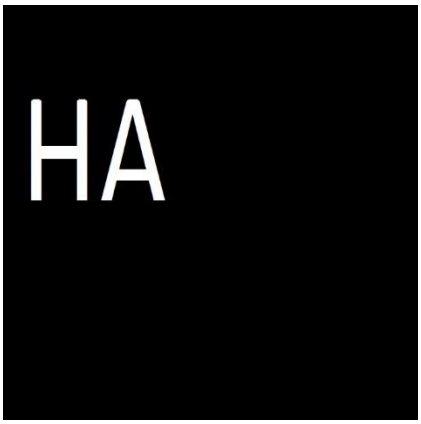
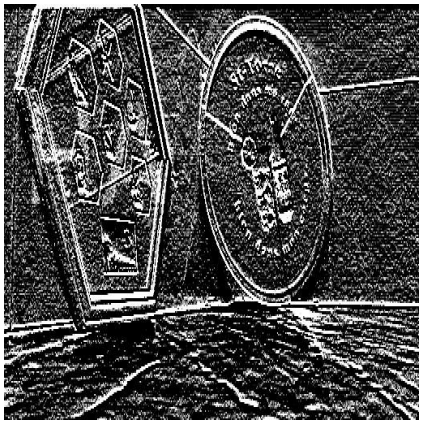


Figure 3.17 Laboratoire de traitement d'images (hw_cosim)

3.4 Résultats et discussion







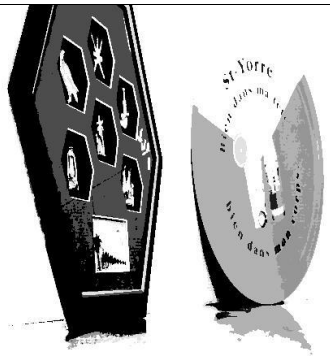
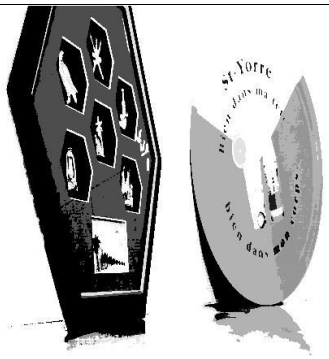
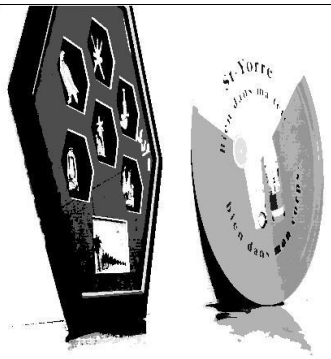
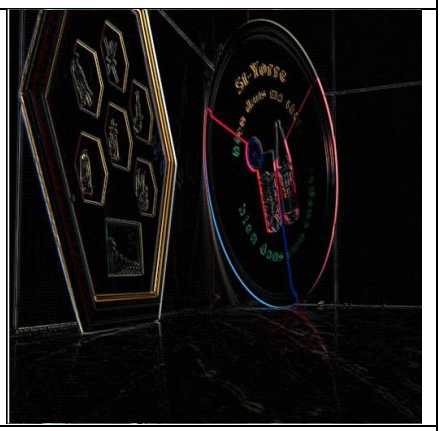
(a) (b) (c)

Figure 3.18 (a) Image orignal 'teste ' (b) la marque utilise (c) la marque extrait

Tableau 3.3 Résultats d'implémentation avec le taux PSNR

	Simulink	**XSG**	**hw_Co-sim**
RGB to Grayscale			
PSNR	23,76 dB	23,8 dB	23,8 dB
Négative			
PSNR	R=6,834/G=8,625/B=8,69 (dB)	R=6,834/G=8,625/B=8,69 (dB)	R=6,834/G=8,625/B=8,69 (dB)

Tableau 3.3 Résultats d'implémentation avec le taux PSNR (suite)

Seuillage			
PSNR	8,644 dB	8,621 dB	8,621 dB
Détection de contour			
PSNR	R=5,522/G=5,249/B=5,442 (dB)	R=6,021/G=5,735/B=5,938 (dB)	R=6,021/G=5,735/B=5,938 (dB)
Tatouage Numérique			
PSNR	28,92 dB	26,25 dB	26,25dB

3.4.1 Discussion

Dans ce chapitre, le but est d'implémenter les techniques de traitement d'image sur une carte FPGA (Hardware) et comparer les résultats obtenus par le hardware avec ceux obtenus avec le software. Les résultats montrent que l'implémentation a été faite avec succès. Toutefois, nous remarquons une légère différence entre les deux (software et hardware), qui est due au fait que le software (Simulink) utilise un grand nombre de bits (64 bits), par contre XSG utilise un nombre réduit de bits (erreur de quantification).

Nous avons implémenté l'algorithme proposé par Xia [29] et nous avons essayé de l'améliorer pour obtenir de bons résultats. Le but de rajouter le poids w et de le faire varier entre 0 et 1 et permettre que le tatouage soit invisible ou visible et, en même temps, voir la robustesse du tatouage (voir tableau 3.4).

Tableau 3.4 Les intervalle de poids du tatouage numérique (w)

$w \in$	Robustesse	Visibilité
]1 ; 0.02]	Robuste	Visible
]0.02 ; 0.003]	Semi-Fragile	Invisible
]0.003 ; 0[	Fragile	Invisible

Conclusion générale

Le but du projet est l'implémentation sur FPGA d'un certain nombre de technique de traitement d'images comme la détection de contour et le tatouage d'image.

Dans un premier temps, le travail parait facile dans la mesure où la programmation se fait dans l'environnement *MATLAB* et que les techniques paraissent très simples. Toutefois, il fallait revoir les algorithmes sous l'angle d'un programmeur qui dispose de composantes matérielles.

Dans un deuxième temps, il fallait optimiser le nombre de bits qui permet de trouver un compromis entre les performances de traitement et les ressources matérielles requises. Une fois les contraintes de temps et de ressources ont été satisfaites, le bloc *hardware Co-Sim* a été généré.

Finalement, nous avons réussi à créer un petit laboratoire de traitement d'images avec différents techniques. Les résultats de simulation sur le logiciel (XSG) et le matériel (FPGA) ont été très concluants.

Comme prochaine étape, il serait intéressant de faire fonctionner ses algorithmes sur la carte FPGA indépendamment de l'environnement *MATLAB/ SIMULINK* en envoyant, par exemple, les images traitées via le port VGA de la carte FPGA.

Annexes

L’histogramme de l’image numérique

En imagerie numérique, l’histogramme représente la distribution des intensités (ou des couleurs) de l'image. C'est un outil fondamental du traitement d'images, avec de très nombreuses applications. Les histogrammes sont aussi très utilisés en photographie et pour la retouche d'images. Pour une image en niveau de gris, c'est-à-dire à une seule composante, l'histogramme est défini comme une fonction discrète qui associe à chaque valeur d'intensité le nombre de pixels prenant cette valeur. La détermination de l'histogramme est donc réalisée en comptant le nombre de pixel pour chaque intensité de l'image. On effectue parfois une quantification, qui regroupe plusieurs valeurs d'intensité en une seule classe, ce qui peut permettre de mieux visualiser la distribution des intensités de l'image. Pour les images couleurs, on peut considérer les histogrammes des 3 composantes de façon indépendante, mais cela n'est en général pas efficace. On construit plutôt un histogramme directement dans l'espace couleur. Les classes de l'histogramme correspondent désormais à une couleur (ou un ensemble de couleurs, en fonction de la quantification), plutôt qu'à une intensité. On parle alors parfois d'histogramme de couleur.

Ondelette de Daubechies

Nommées d'après leur créatrice Ingrid Daubechies, les ondelettes de Daubechies sont une famille d'ondelettes orthogonales définissant une transformée en ondelettes discrète (DWT), caractérisées par un nombre maximal de moments dissipant pour un support donné. Pour chaque type d'ondelette de cette classe, il existe une fonction d'échelle (appelée aussi ondelette mère) qui génère une analyse multi résolution orthogonale.

Peak Signal-to-Noise Ratio (PSNR)

Le PSNR est une mesure de distorsion utilisée en image numérique, tout particulièrement en compression d'image. Il s'agit de quantifier la performance des codeurs en mesurant la qualité

de reconstruction de l'image compressée par rapport à l'image originale. PSNR est définie par l'erreur quadratique moyenne (MSE)

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

Le PSNR (on dB) est définie par :

$$PSNR = 10.log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

Autant la valeur de PSNR augment autant y'a pas distorsion c'est-à-dire y'a pas différence dans le cas ou MSE=0 ➔ PSNR=∞ , par contre si le PSNR démunie implique que la distorsion augmente.

Bibliographie et Webographie

[1] https://www.library.cornell.edu/preservation/tutorial/intro/intro-01.html

[2] https://sites.google.com/site/learnimagej/image-processing

[3] A. Gupta, H. Vaishnav, H Garg ”Image Processing using Xilinx System Generator (XSG) in FPGA”. Vol. II, Issue IX, Sept. 2015, pp. 119-125.

[4] http://whatis.techtarget.com/definition/grayscale

[5] http://py.processing.org/tutorials/color/imgs/grayscale.jpg

[6] http://matplotlib.org/1.4.1/users/colormaps.html

[7] https://commons.wikimedia.org/wiki/File:Pozytyw_i_negatyw.jpg

[8] http://www.tutorialspoint.com/dip

[9] Art and History Trust Collection, Photo courtesy of the Freer Gallery of Art, Smithsonian Institution, Washington, D.C. (Page from a Koran)

[10] H. K. Anasuya Devi, “Thresholding : A Pixel-Level Image Processing Methodology Preprocessing Technique for an OCR System for the Brahmi Script”. Ancient Asia, 1, pp.161–165, 2006

[11] S.S. Al-amri, N.V. Kalyankar and S.D. Khamitkar “Image Segmentation by Using Thershod Techniques” Journal of Computing, Vol. 2, Issue 5, May 2010, pp. 83-86.

[12] E.ESCODA et A.ALONSO, SEUILLAGE D'HISTOGRAMME, 1999

[13] https://sites.google.com/site/insyslearning/otsu-s-method

[14] S.Shubham, K.Bhavesh, S.Bhatia, COMPARATIVE STUDY OF IMAGE EDGE DETECTION ALGORITHMS, School of Computing Science and Engineering, Vellore Institute of Technology, India

[15] J. Canny, “A Computational Approach to Edge Detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, Nov. 1986, pp. 679-698.

[16] D. Ray, Edge Detection in Digital Image Processing, June 06, 2013.

[17] S. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation”, IEEE Trans. PAMI, vol. 11, pp. 674-693, 1989.

[18] C.Lin, S.Wang, H.Cheng, K.Fan, W.Hsu and C.Lai, “ Bimodal Biometric Verification Using the Fusion of Palmprint and Infrared Palm-Dorsum Vein Images”, MDPI, Sensors 2015.

[19] http://www.posterus.sk/?p=10983

[20] A.Z.Tirkel, G.A. Rankin, R.M. Van Schyndel, W.J.Ho, N.R.A.Mee, C.F.Osborne. “Electronic Water Mark”. DICTA93, Macquarie University. p.666-673, 1993.

[21] Wayner, Peter . Disappearing cryptography: information hiding: steganography & watermarking. Amsterdam: MK/Morgan Kaufmann Publishers. ISBN 1-558-60769-2, 2000.

[22] A.Khan. and A.M.Mirza. Genetic perceptual shaping: Utilizing cover image and conceivable attack information during watermark embedding. Inf. Fusion 8, 4, 354-365, Oct. 2007.

[23] S. Walton, “Information Authentication for a Slippery New Age”, Dr. Dobbs Journal, vol. 20, no. 4, pp. 18–26, Apr 1995.

[24] J. Fridrich. Robust Bit Extraction From Images. ICMCS’99, Florence, Italy, june 1999.

[25] SHA-1, Secure Hash Standard (SHS), spécification (FIPS 180-1), http://www.itl.nist.gov/fipspubs/fip180-1.htm

[26] R.B. Wolfgang and E. J. Delp. “A watermark for digital images”. Proceedings of the International Conference on Image Processing. Vol. 3, pp. 219-222, Lausanne, Switzerland, Sept. 1996.

[27] W. Bender, D. Gruhl, N. Morimoto, A. Lu, "Techniques for Data Hiding", IBM Systems Journal, Vol. 35, No 3-4, 1996, pp. 313 – 336.

[28] D. Kundur and D. Hatzinakos, “A Robust Digital Image Watermarking Method using Wavelet-Based Fusion”, In IEEE International Conference on Image Processing, vol. 1, p. 544-547, Santa Barbara, USA, Octobre 1997.

[29] X.-G. Xia, CG. Boncelet et G.R. Arce : A multiresolution watermark for digital images. In IEEE International Conference on Image Processing, vol. 1, p. 548-551, Santa Barbara, États-Unis, octobre 1997.

[30] T. Xianghong, Li Lu, Y. Lianjie and N. Yamel : A digital watermarking scheme based on DWT and vector transform. In International Symposium on Intelligent Multimedia, Video and Speech Processing, p. 635-638, Hong Kong, Chine, octobre 2004.

[31] F. Hartung et M. Kutter : Multimedia watermarking techniques. Proceedings of the IEEE, 87(7): 1079-1107, juillet 1999.

[32] A.G. Bors, and I. Pitas: Image watermarking using DCT domain constraints. In IEEE Inter-national Conference on Image Processing, vol. 3, p. 231-234, Lausanne, Suisse, septembre 1996.

[33] B. TAO and B. Dickinson : Adaptive watermarking in the DCT domain. In IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 4, p. 2985-2988, Munich, Allemagne, avril 1997.

[34] F. Hartung, J.K. SU et B. Girod : Spread spectrum watermarking : malicious attacks and counterattacks. In SPIE Conference on Security and Watermarkinci of Multimedia Contents, vol. 3657, p. 147-158, San Jose, États-Unis, janvier 1999.

[35] S. Voloshynovskiy, S. Pereira, T. Pun, J.J. Eggers et J.K. Su : Attacks on digital watermarks : classification, estimation based attacks, and benchmarks. IEEE Communications Magazine, 39(8): 118-126, août 2001.

[36] M.S. Bouhlel, H. Trichili, N. Derbel and L. Kamoun, “On the Image Watermarking Techniques Applications, Properties and fields”, National Engineering School of Sfax, RIST Vol, 12 n°02, p. 39-46, Sfax,Tunisie 2002.

[37] http://www.xilinx.com/

[38] http://www.eecg.toronto.edu/~vaughn/challenge/fpga_arch.html

[39] https://reference.digilentinc.com/_media/nexys:nexys4:nexys4_rm.pdf