الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث 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

DJEMOUI Mohammed

&

BENAOUADI Arezki

pour l'obtention du diplôme Master en Électronique option système de vision et robotique

Thème

Filtrage des séquences vidéo sur FPGA avec l'outil X.S.G (xilinx system generator)

Proposé par : BOUGHERIRA KHETIB Nadia & ABDELLI Lotfi

Année Universitaire 2011-2012

Remerciements

En premier lieu, nous remercions Dieu qui nous a procuré la santé, la patience et la connaissance pour aboutir à ce succès.

A travers ce modeste travail, nous tenons à remercier vivement notre

Promotrice Madame N. BOUGHERIRA pour l'intéressante documentation qu'elle a mise à notre disposition, pour ses conseils précieux et pour toutes les commodités et aisances qu'elle nous a apportées durant notre étude et réalisation de ce projet.

On tient à remercier monsieur L. ABDELLI Co-promoteur enseignant à l'université de Saad Dahleb (Blida), pour sa disponibilité, et son aide précieuse tout au long de ce travail de recherche.

Nos remerciements les plus vifs s'adressent aussi aux messieurs le président et les membres de jury d'avoir accepté d'examiner et d'évaluer notre travail.

Nous exprimons également notre gratitude à tous les professeurs et enseignants qui nous ont aidés à réaliser ce travail, sans omettre bien sur de remercier profondément tous ceux qui ont contribué de près ou de loin à réaliser ce présent travail.

Et enfin, que nos chers parents et familles, et bien avant tout, trouvent ici l'expression de nos remerciements les plus sincères et les plus profonds en reconnaissance de leurs sacrifices, aides, soutien et encouragement afin de nous assurer cette formation dans les meilleurs conditions.

DEDICACES

Je dédie ce travail à mes chers parents pour leur amour, leurs Encouragements, leur soutien indéfectible durant toutes mes années d'étude.

Je dédie également ce travail :

A mon frère : Smail, et mes sœurs Fatima, Ouiza et Faiza, je leur souhaite toute la réussite dans leurs études ainsi dans leur vies.

A toute ma famille et proches.

A tous mes amis : Tahar, Hachmi, hamza et sans oublier tous mes amis de classe SVR : Samir, Mustapha, Abdelkader, …….pour leurs ambiance chaleureuse et leurs patience.

Arezki

Je dédie ce travail
A Celle qui m'a mis au monde, m'a suivi avec son Cœur, son esprit et m'a toujours soutenu pendant les moments difficiles, ma très chère Maman.
A Celui que je respecte beaucoup et à qui je dois ce que je suis aujourd'hui, mon très cher père
A mes grands parents et toute ma famille
A mes frères omar,abdelkarim,merwane et rafik Ouvaid, mes sœurs siham et amina
A la mémoire de mon cousin
Spécialement à mouloud,mohamed,madjid et nabil
A tous mes amis et mes enseignants, avec lesquels je partage mes meilleurs souvenirs.
A tous qui me sont chers
A mon binôme arezki
Je dédie aussi ce travail, à toute personne qui m'a aidé dans ce mémoire.

Mohamed

ملخص

الهدف من مشروعنا هو تصفية الفيديو في الوقت الحقيقي باستخدام مرشح غير خطي " مرشح المتوسط " والتي تفرض علينا استخدام أجهزة منطقية قابلة للبرمجة (الدوائر الإلكترونية السريعة) قادرة على التعامل مع كميات كبيرة من المعلومات التي قدمها مصدر الفيديو.

للقيام بذلك استخدامنا FPGA لقدرتها على حساب كميات كبير من المعلومات

للتقليل من وقت حساب المعلومات استخدامنا أربع RAMs بحديث في كل لحظة وليدنا إشارات التحكم بحيث في كل لحظة لدينا " 3 RAMs» في وضع القراء ة و " 1 RAMs" في وضع الكتابة.

Résumé : Le but de notre projet est le traitement vidéo en temps réel par un filtre non linéaire «le filtre médian», ce qui nous a obliger à utiliser des circuits logiques programmables (des circuits électroniques rapides), Capables de manipuler les grandes quantités d'informations générées par la source vidéo. Pour cela, on a travaillé avec La carte FPGA ayant cette capacité de calcul.

Pour diminuer le temps de traitement on a utilisé quatre RAMs blocs pour chaque instant on génère les signaux de commande des RAMs de sortie que on a sélectionné 3 RAMs on mode lecture et une en mode écriture.

Abstract :

The goal of our project is filtering a video in real time by a nonlinear filter "the median filter," which oblige us to use programmable logic devices (electronic circuits faster) Capable of handling large amounts of information generated by the video source. To do this, we work with the FPGA board having the ability to calculate.

To reduce processing times we used four RAMs blocks for every moment we generate the control signals of RAMs fate was selected RAMs are three playback modes and write mode.

Listes des acronymes et abréviations

FPGA: Fieled Programmable Gate Array.

VHDL: Hardware Description Language.

MPEG: Motion Picture Expert Group.

CODEC: Compression /Décompression.

RGB : Rouge / Vert / Blue.

Y U V : Y – luminance et U et V les chrominances.

Table des matières

Titre

Chapitre 02 : Notions sur les extensions vidéo

Chapitre 03 : Filtrage dune séquence Vidéo par Co-simulation

Liste des figures

Chapitre 1 : Rappels sur le traitement d'images et description VHDL de filtrage médian

Chapitre 2 : Notions sur les extensions vidéo

Chapitre 3 : Filtrage dune séquence Vidéo par Co-simulation

Liste des tableaux

Introduction générale

Les applications temps réel sont présentes dans de nombreux domaines tels que les transports ferroviaires ou l'aéronautique, l'automobile, les télécommunications, la robotique, les cartes à puce, etc.

Dans le domaine de traitement vidéo, les applications temps réel, la chaîne contient tout le cycle d'acquisition, de traitement et de restitution d'un signal vidéo provenant d'une source vidéo comme les caméras ou autre. Donc le temps réel pour nous est le fait d'avoir le même flux d'images entre la source vidéo en entrée et le moniteur de restitution en sortie. L'acquisition et le traitement ne devant pas introduire des grands retards afin de ne pas ralentir le flux de restitution des images traitées.

Une application temps réel est une application pour laquelle le facteur temps est la principale contrainte à respecter. L'application doit fournir un résultat juste et sans délai. Il ne s'agit pas, par contre, de rendre le résultat le plus vite possible, mais simplement à l'échelle du temps relative à la contrainte temporelle varie d'une application à l'autre : elle peut être par exemple de l'ordre de la microseconde dans des applications de contrôle radar, mais de l'ordre de quelques minutes pour une application de contrôle de processus chimique. Par ailleurs, le système temps réel peut être qualifié de système embarqué ou enfoui.

Les algorithmes de traitement en temps réel nécessitent des puissances de calcul au de là des performances des processeurs classiques. Pour palier à ces contraintes on peut exploiter des outils de traitement rapides basés sur la co-simulation.

Le but du travail est dans un premier temps la réalisation d'un filtre médian sur FPGA, dont le rôle est de filtrer une image 512*512.

Dans un second temps l'application du filtrage médian des séquences vidéo sur FPGA avec l'outil Xilinx System Generator.

Cette application va permettre le traitement des séquences vidéo en temps réel.

Le mémoire du projet est constitué de trois chapitre qui présente quelque rappels sur le traitement d'images et introduit la notion de filtrage sur FPGA, il contient également l'implémentation du filtre médian choisi dans ce travail.

Le second chapitre présente quelques extensions des vidéos.

Le troisième chapitre introduit la notion de la co-simulation qui est utilisé dans notre cas simulink et Xilinx System Generator.il présente également le système de traitement vidéo que nous avons développé et nous donnerons les résultats obtenus.

Ce travail sera clôturé par une conclusion.

Chapitre 01 : Rappels sur le traitement d'image et description VHDL d'un masque de convolution

Introduction

Ce chapitre a pour objectif d'exposer quelques notions sur les techniques de traitement d'images et de mettre le point sur des techniques couramment utilisées dans le traitement bas niveau, il traitera également le filtrage médian implémenté sur FPGA.

1.1. Les notions de base

1.1. 1. Définition d'une image

Une image numérique est une fonction à support discret et borné, et a valeurs discret. Le support est multidimensionnel, en générale 2D ou 3D. Les valeurs peuvent êtres scalaire (image en niveau de gris), ou bien vectorielles (images multi composantes, image couleurs).

Une image numérique est associée a un pavage de l'espace, en général rectangulaire. Chaque élément de pavage appelé **pixel,** Une image 2D est représentée par un tableau de matrice $\mathbf{I}$, de $\mathbf{H}$ lignes et de $\mathbf{W}$ colonnes, le pixel est désignée par un couples **(i, j)** où **i** est l'indice de colonnes $\mathbf{i \in [0, H-1]}$, et **j** l'indice de ligne $\mathbf{j \in [0, W-1].}$ $\mathbf{W}$ est largeur et $\mathbf{H}$ est l'hauteur de l'image $\mathbf{I,}$ par convention le pixel origine (0 ,0) est en général en haut à gauche (voir Figure 1.1) $\mathbf{(i, j) \in [0, Nmax -1],}$ Nmax est le nombre de niveau de gris. [1]

- Une image numérique est une image quantifiée et échantillonnée. [1]

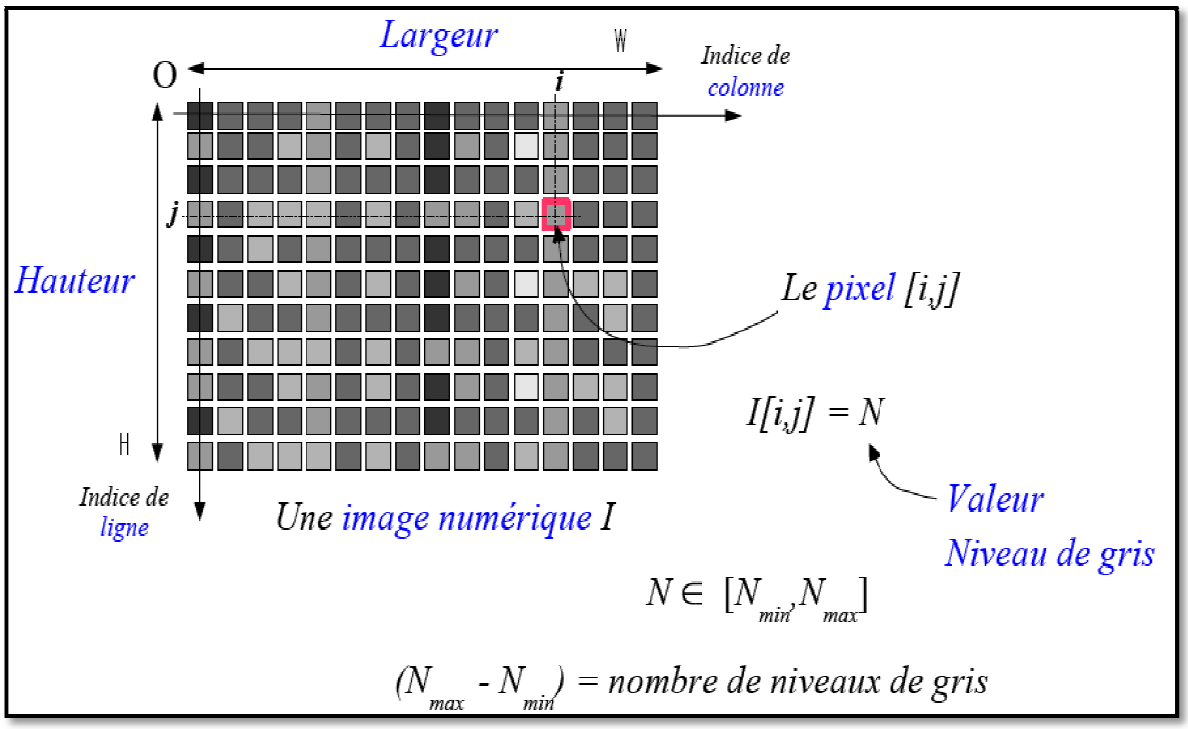


Figure 1.1 **:** une image numérique

1.1. 2. Echantillonnage

L'échantillonnage est une étape fondamentale qui doit prendre en compte le contenu de l'image à analyser. Intuitivement, on conçoit bien qu'une « structure fin », c'est-à-dire une partie de l'image comportant des oscillations de petite période spatiale, nécessitera plus de pixel qu'une partie présentant moins de variation.

Mais si l'on ne dispose pas d'assez de pixel pour représenter la dite structure fin , dans ce cas, il est souhaitable que les quelque pixels utilisés pour représenter la structure rendent compte d'un « comportement moyen » de la région concernée Cependant, si l'on contente de sous-échantillonner une région trop complexe ce n'est pas de tout ce qui se produit : en général, de nouvelle structures apparaissent. Qui ne correspondent pas a l'image réelle. [1]

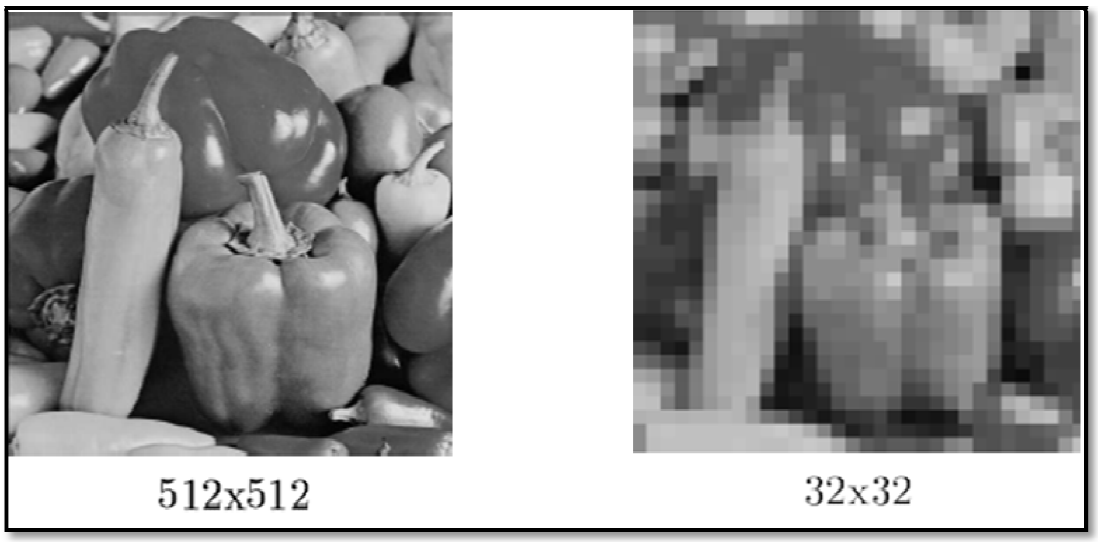


Figure 1.2 **:** effet d'échantillonnage

1.1. 3. La quantification

La quantification désigne la discrétisation tonal correspondant a la limitation du nombre de valeurs différent que peut prend chaque pixel. [1]

Exemple :

Une image en niveau de gris (0-255) chaque pixel quantifié sur 8 bits.

Une image numérique est donc une image quantifiée.

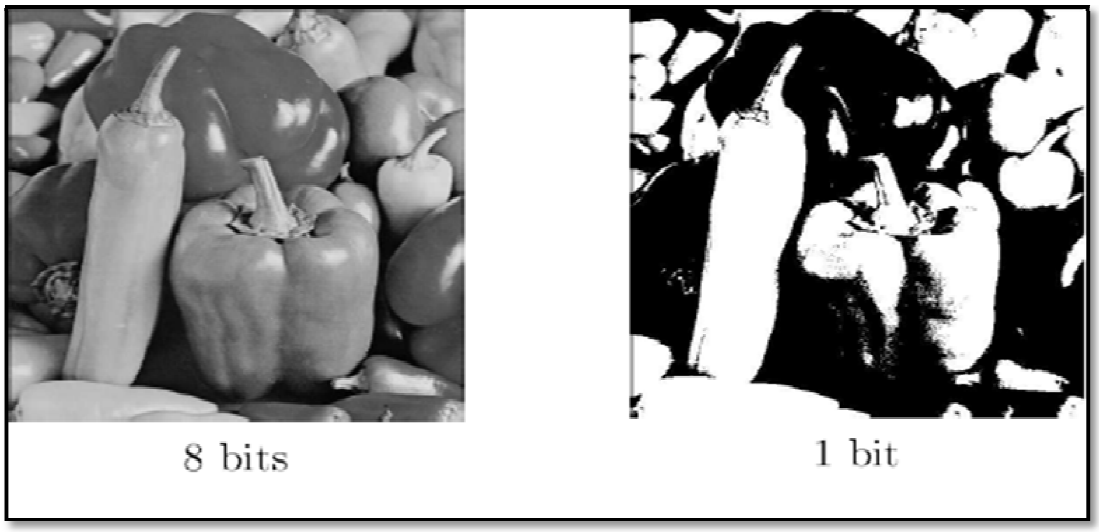


Figure 1.3 **:** effet de quantification

1.1.4. Définition de bruit

A chaque étape de l'acquisition d'une scène, des perturbations (rayures, poussières, caméra, amplification, quantification) vont détériorer la qualité de l'image. Ces Perturbations sont regroupées sous le nom de "bruit d'image". Ce bruit d'image peut être considéré de la manière suivante : [6]

- son caractère aléatoire (moyenne, variance).
- additif (ex. gaussien, impulsionnel), multiplicatif (ex. grain).
- sa fréquence.
- son caractère centré ou non.

1.1. 5. Différentes origines du bruit

Il existe plusieurs types des bruits comme *Bruit thermique* et le *Bruit gaussien, triangulaire, Uniform et le bruit* sel et poivre…etc. et chaque bruit a son origine.

Chaque bruit représente un effet indésirable sur les images, les bruits les plus célèbres sont :

a Bruit thermique

Le *bruit thermique*, ou *bruit de Johnson-Nyquist*, est dû à l'agitation naturelle des électrons, qui augmente avec la température du capteur. Ce phénomène est appelé *courant d'obscurité*. Les fabricants de caméras le quantifient par le nombre d'électrons. Sur des appareils fixes de laboratoire ou de studios professionnels, ce bruit peut être réduit efficacement par refroidissement du capteur, en utilisant l'effet Peltier, ou bien par ventilation afin d'assurer le maintien à température ambiante. En astronomie, certains équipements sont même refroidis à -196 °C grâce à l'azote liquide. Sur les appareils-photos grand public, les fabricants réduisent l'impact de Ce Bruit en adoptant ces deux solutions:

- ajout d'un filtre infrarouge juste devant le capteur, ce qui limite l'agitation électronique,
- soustraction du *courant d'obscurité* moyen fourni par des *pixels de calibrage* situés au bord du capteur et ne recevant aucune lumière.
- soustraction des *pixels chauds*, repérés par une pose obturateur fermé, faite automatiquement après une pause longue : aucune lumière n'entrant,

les pixels envoyant un signal sont ceux qui doivent être affaiblis sur la photographie. Cela s'appelle également le signal thermique, car il est reproductible. Contrairement au bruit, qui n'est pas reproductible.

- La figure 1.4 illustre l'Influence de la température du capteur sur le bruit thermique. De gauche à droite, ces 4 photos ont été prises à des températures évoluant progressivement de 4 °C à 25 °C.

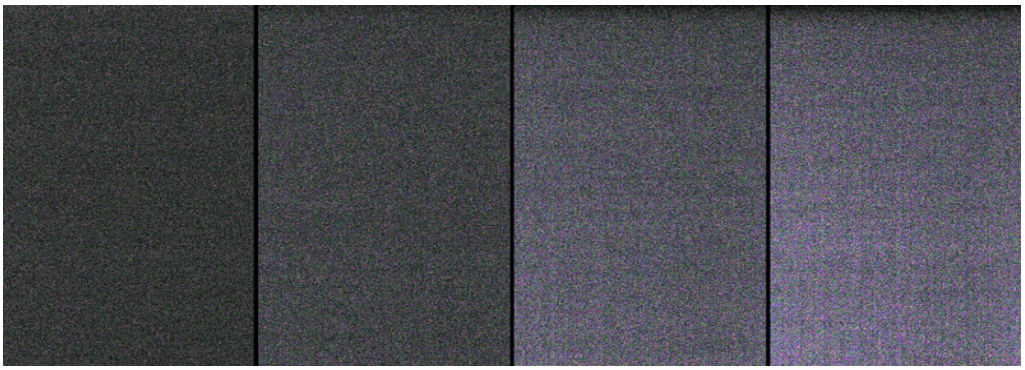


Figure 1.4 : effet de bruit thermique

b Bruit gaussien additif et multiplicatif

Le bruit Gaussien est obtenu en ajoutant à chaque pixel une valeur aléatoire suivant une loi de probabilité gaussienne.

Une variable aléatoire gaussienne a été ajoutée par le système à l'image "idéale". Dans le cas précédent des photos numériques, la source est la précision du capteur CCD ou CMOS mise en évidence par un gain élevé.

Figure 1.5 : Image de Lena avec bruit gaussien

c Bruit de grenaille

Le bruit de **grenaille**, ou bruit de **Schottky** ou bruit **quantique** est un bruit électronique. Il se produit lorsque le nombre fini de particules transportant l'énergie (électrons dans un circuit électronique, ou photons dans un dispositif optique) est suffisamment faible pour donner lieu à des fluctuations statistiques perceptibles. Le bruit des photons est la principale source de bruit dans les images prises par les appareils photo numériques actuels. La photo ci-contre (figure 1.6) a été traitée pour modéliser le résultat d'un appareil photo idéal : rendement quantique = 1, pas de bruit de lecture ni de bruit thermique perceptible. Ensuite, de gauche à droite, le nombre moyen de photons/pixel a été simulé sur la totalité de l'image : 0,001, 0,01, 0,1 (en haut) ; 1, 10, 100 (au milieu) : 1.000, 10.000 et 100.000 (en bas). On observe une amélioration importante de la qualité de l'image au delà de 10 photons/pixel (l'image source a été enregistrée avec un capteur d'une capacité de 40.000 électrons/pixel).

Figure 1.6 : l'effet Bruit de grenaille

d bruit sel et poivre

Le bruit impulsionnel, également appelé sel et poivre, est une dégradation de l'image sous la forme de pixels noirs et blancs répartis au hasard. Ce bruit est dû soit à des erreurs de transmission de données, soit à la défaillance d'éléments du capteur CCD, soit à la présence de particules fines sur le capteur d'images.

Le bruit « poivre-et-sel » consiste à mettre, aléatoirement, plusieurs pixels aux valeurs 255 ou 0 (valeurs extrêmes de l'intervalle des niveaux de gris).

Note : la commande matlab suivant peut crée le bruit *sel et poivre.*

```
IB = imnoise (I,'salt & pepper');
```

Figure 1.7 : Image originale et image bruitée

1.2. Généralité sur les filtres

1.2.1. Définition d'un filtrage

L'objectif du filtrage est de réduire les variations d'intensité au sein de chaque région de l'image tout en respectant l'intégrité des scènes : les transitions entre régions homogènes et les éléments significatifs de l'image doivent être préservés au mieux. Différentes méthodes de filtrage ont été développées suivant le type et l'intensité du bruit, ou les applications auxquelles on destine l'image. Les premières et les plus simples de ces méthodes sont basées sur le filtrage linéaire stationnaire (invariant par translations), mais les limitations de ces techniques (en particulier leur mauvaise conservation des transitions) a conduit au développement des filtres non linéaire. Certains filtres détecteurs de contours Contiennent d'ailleurs cette fonction de filtrage (ou lissage) dans leur algorithme.

Dans les cas présentés dans ce chapitre (filtrage moyenneur, Smooth, médian), le filtrage consiste à Balayer l'image par une fenêtre d'analyse de taille finie (noyau ou kernel) : le calcul du nouveau niveau de gris du pixel considéré ne prend en compte que les plus proches voisins de celui-ci.

1.2.2. Les filtres linéaires

Ces opérateurs sont caractérisés par leur réponse impulsionnelle h(x,y) (ou h(i,j) dans le cas discret), la relation entrée '**E**' et la sortie **'S'** étant donnée par :

$$S[i,j] = \sum_{u,v=-\infty}^{u,v=+\infty} (E[i,j] * h[i-u, j-v])$$

pour u, v variant de moins l'infini à plus l'infini.

a Filtre moyenneur

Dans le filtre moyenneur le niveau de gris du pixel central est remplacé par la moyenne des niveaux de gris des pixels environnants. La taille du kernel dépend de l'intensité du bruit et de la taille des détails significatifs de l'image traitée. [6]

Exemple de taille du masque (kernel) :

1	1	1
1	1	1
1	1	1

(3x3)

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

(7x7)

Les effets du filtre moyenneur varient avec la taille du noyau : plus les dimensions Du noyau seront importantes, plus le bruit sera éliminé ; mais en contrepartie, les détails fins seront eux aussi effacés et les contours étalés.

En effet, ce type de filtrage passe-bas consiste à atténuer les composantes de l'image en hautes fréquences (pixels foncés). Il implique donc une réduction des variations brutales dans l'image.

Pour celles altérées par un bruit concentré dans les hautes fréquences et possédant peu de variations brutales. [6]

- Exemple d'application de filtre moyenneur

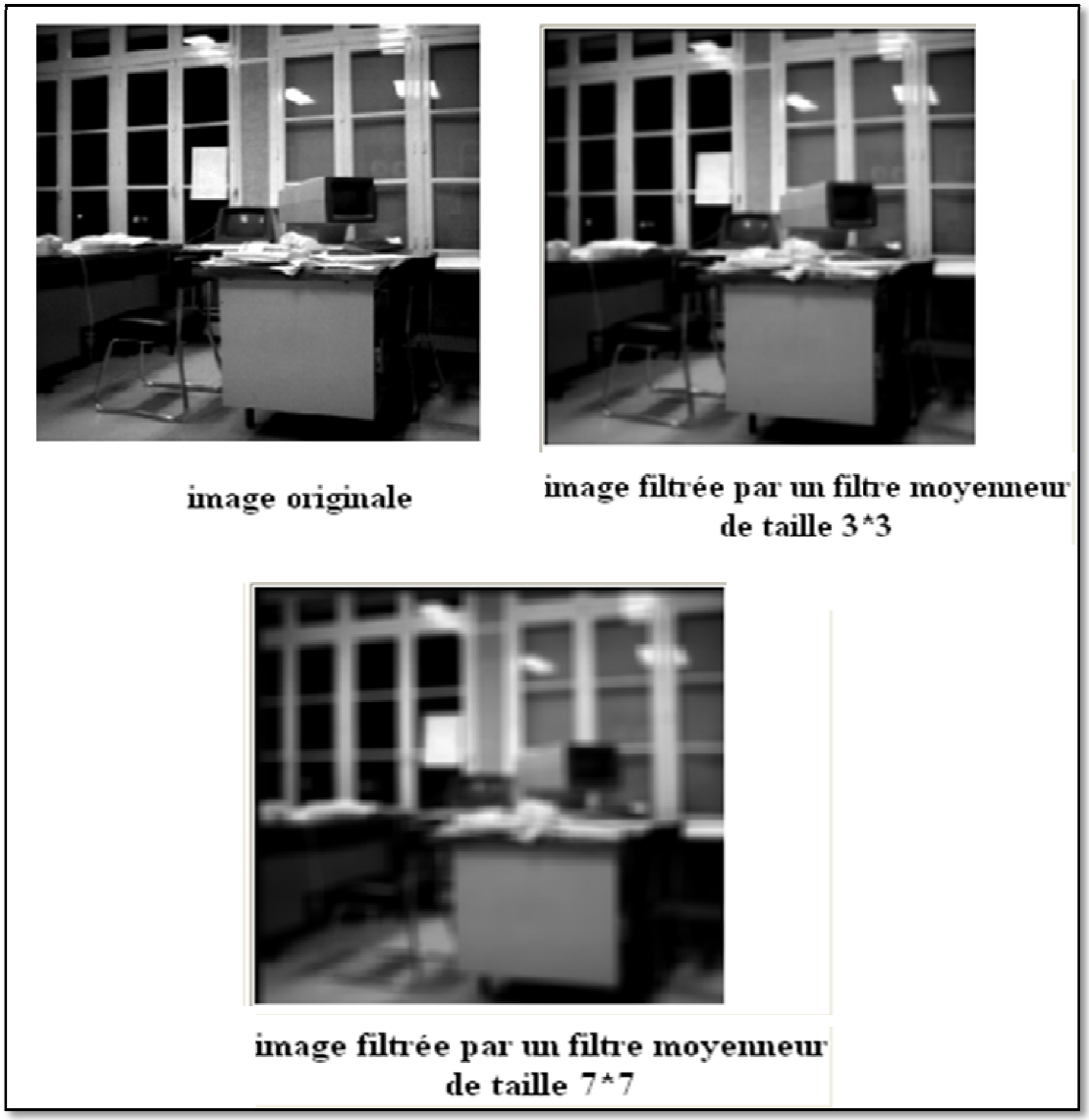


Figure 1.8 : Résultats d'application du filtre moyenneur pour les tailles du masque (3*3) et (7*7)

b Filtre de smooth

Dans le filtre de smooth le niveau de gris du pixel central est calculé de même façon que le filtre moyenneur mais par des coefficients différents.

Les coefficients sont calculés en utilisant des pondérations gaussiennes.

1	2	1
2	4	2
1	2	1

Des itérations successives permettent d'obtenir le smooth 5*5, et le smooth 7*7.

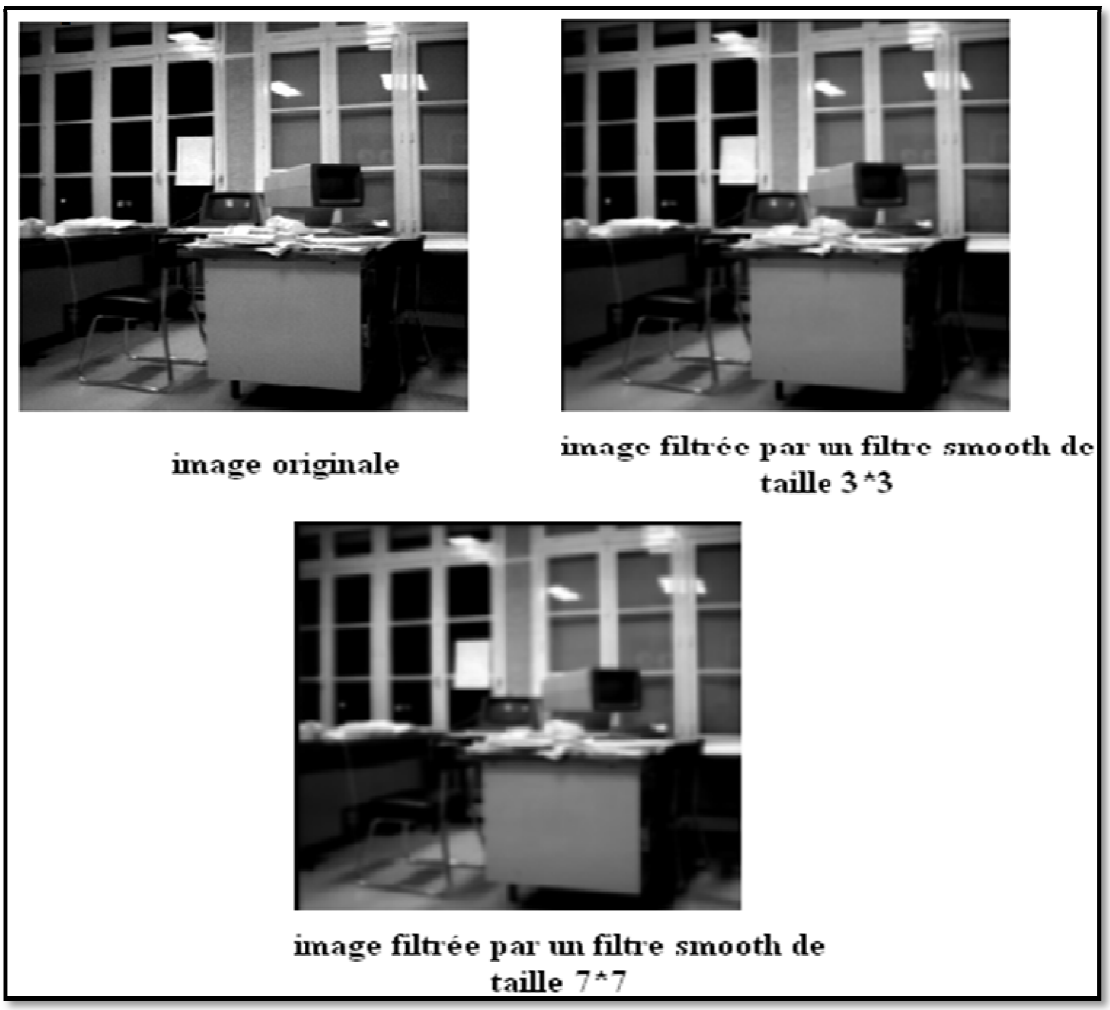


Figure 1.9 : Résultats d'application du filtre smooth pour les tailles du masque (3*3) et (7*7)

1.2. 3. Filtres non linéaires

Il existe plusieurs filtres qui sont pas linéaire parmi celle-ci on trouve par exemple le filtrage morphologiques, Filtre de Nagao, min-max et Filtre médian … (etc.).

1.3. Etude de filtre médian

1.3.1 Définition

Le filtrage médian est un filtre non linéaire de la famille des filtres d'ordre. Le filtrage médian est très robuste à différents types de bruit, comme le bruit gaussien ou le bruit impulsionnel.

Le principal inconvénient du filtre médian est sa complexité mathématique pour calculer le médian de n éléments. Son autre inconvénient est que dans sa formulation

classique (sous forme de tri) il est impossible de paralléliser certaines parties de l'algorithme à cause des dépendances de données. [6]

1.3.2. Principe du filtrage

Le niveau de gris du pixel central est remplacé par la valeur médiane de tous les pixels de la fenêtre d'analyse a filtré selon la taille du filtre. [6]

Exemple :

On illustre le fonctionnement d'un filtre médian de fenêtre d'analyse 3*3 ci-dessous par la (figure 1.10) : en suivant les étapes suivantes :

Fenêtre sur l'image originale Classer par ordre croissant.

Affectation de la valeur médiane à la place du pixel central.

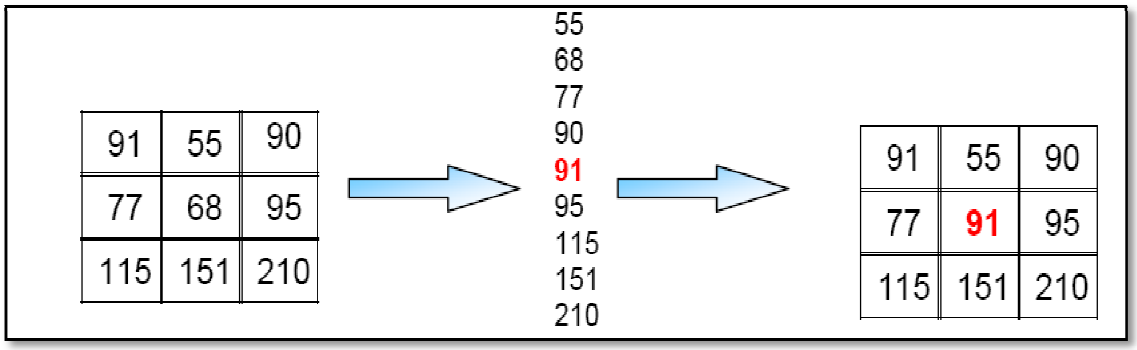


Figure1.10 : la valeur médiane d'une fenêtre de 3*3

Valeur médiane : 91 au pixel central

Les propriétés attendues de ce filtre sont :

- plus le masque est grand, plus le filtrage est efficace mais plus l'image est Déformée.
- il préserve les contours.
- il supprime les valeurs illogiques dans la suite.
- il est très efficace sur du bruit impulsionnel (type poivre et sel).

Remarque :

Pour les fenêtres de taille paire (2 K valeurs) : Après ordonnancement croissant des valeurs des pixel,on prendre la moyenne des 2 valeurs centrales :

Valeur de sortie = (Kième valeur ordonnée + (K+1)ième valeur ordonnée)/2.

1.3.3. Propriétés du filtre Médian

a Linéarité du filtre Médian

Le filtre respecte la loi multiplication de *f* par un scalaire

***a M* [*a f*] = *a M* [*f*] .**

En effet, la multiplication de l'ensemble des valeurs du voisinage ne modifie pas l'ordonnancement de ces valeurs pour ***a* > 0**.

Pour **a < 0**, l'ordonnancement est renversé, mais le centre de la classification décroissante reste le même.

Le filtre Médian ne respecte pas l'addition :

M* [f +*g]* ≠ *M* [f*]* + *M* [*g] car le classement d'une somme de valeurs n'est pas égal à la somme des classements dans le cas général. [9]

b Aspects non-linéaires du filtre Médian

C'est un filtre croissant, autodual, idempotent lorsque le filtrage conduit à une image totalement ordonnée (cas qui ne peut s'obtenir que par un très grand nombre de passages successifs), qui respecte l'union et l'intersection. Le filtre Médian est considéré comme faiblement non- linéaire par rapport à d'autres Modèles. [9]

c Réponse impulsionnelle:

Par définition, la réponse impulsionnelle est obtenue pour une image source ne comprenant qu'un seul point non-nul. Etant donné que la taille d'un médian est d'au moins *N*=3 pour un filtre symétrique, l'impulsion ne peut faire partie du médian.

La réponse impulsionnelle du médian est donc *nulle*.

De façon générale, toute information du type contraste local (noir sur blanc ou blanc sur noir) de taille inférieure à **(N−1) /2**, ne peut être médian. Une telle information disparaît donc du résultat. Le choix de la dimension spatiale du filtre médian est essentiellement lié à cette notion. [9]

d Respect des contours

Considérons l'action d'un filtre Médian unidimensionnel avec N = 3 appliqué sur une fonction f(x).comme illustré par la figure (1.11)

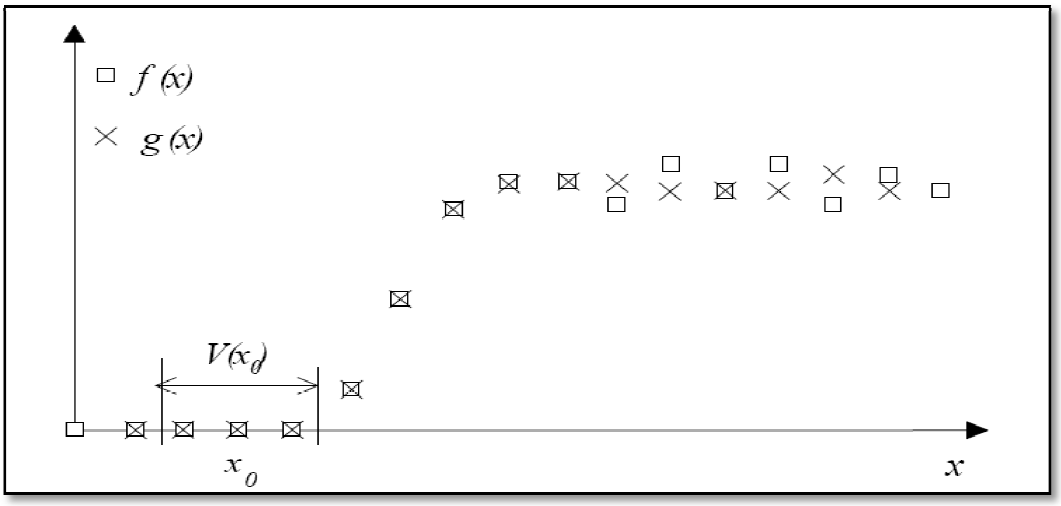


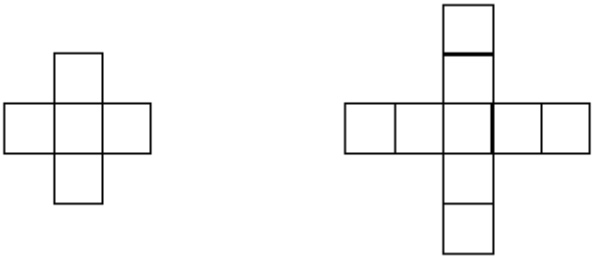
Figure 1.11 : Action du filtre médian unidimensionnel avec N=3

On remarque que le filtre Médian ne produit aucune action sur un contour non bruité: en effet, le contour représente une structure ordonnée autour du point courant *x0*, il en résulte que la classification du voisinage est déjà réalisée avec comme élément central de la classe *F(x0).* En particulier, le contour parfait type *marche* est préservé. Cette différence fondamentale par rapport aux filtres linéaires permet son emploi dans le domaine de l'*imagerie vidéo*, comme débruiteur par exemple. [9]

e *Rejet des valeurs extrêmes*

Le filtre Médian a la propriété de rejeter les valeurs extrêmes puisqu'elles se trouvent en bord de classement et non au centre.

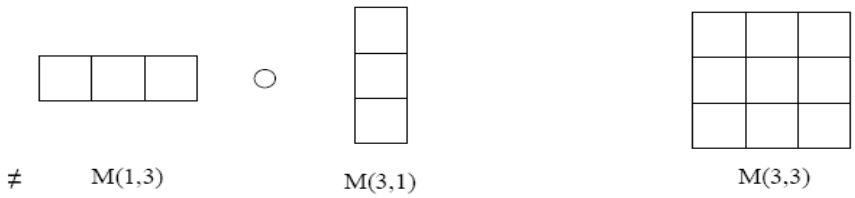
A noter que la valeur médiane est toujours parmi les valeurs *majoritaires* du voisinage. Cet aspect est très important car ce filtre rejette toute *information minoritaire* par rapport à la taille du filtre. Par exemple, pour un filtre (3,3), les informations de taille inférieure à 4 pixels sont éliminés Pour minimiser cet effet de rejet, on utilise les formes de voisinages suivants: [9]



- ***Loi de composition***

Contrairement aux filtres linéaires, le passage successif de 2 ou plusieurs filtres Médian n'est pas équivalent à Médian unique: $f \circ M1 \circ M2 \neq f \circ M$

Par exemple, le passage d'un filtre horizontal puis d'un filtre vertical ne donne pas le même résultat qu'un 3x3.



En effet, le premier schéma est un choix en deux étapes de sélection, alors que le médian (3,3) est un choix direct. Rien ne permet d'affirmer que la valeur retenue dans le voisinage (3,3) fait partie des valeurs retenues lors du passage du (1,3) puisque les ensembles ne sont pas les mêmes dans les deux cas. En particulier, la notion d'information majoritaire n'a pas le même poids pour les deux méthodes

Pour trouver le médian des valeurs d'un masque il ya plusieurs méthode de tris parmi celle-ci on trouve : [9]

- tri par Sélection.
- tri par tri rapide.
- tri à bulles.

1.4. Les différentes méthodes de Tri :

1.4.1. Médian par la méthode de tri sélection :

Principe de la méthode : Sélectionner le minimum du tableau en parcourant le tableau de la fin au début et en échangeant tout couple d'éléments consécutifs non ordonnés. [11]

➲ Exemple

Pour trier : 101, 115, 30, 63, 47, 20 On va avoir les boucles suivantes :

i=1_101, 115, 30, 63, 47, 20

101, 115, 30, 63, 20, 47

101, 115, 30, 20, 63, 47

101, 115, 20, 30, 63, 47

101, 20, 115, 30, 63, 47

i=2 _20, 101, 115, 30, 63, 47

i=3 _20, 30,101, 115, 47, 63

i=4 _20, 30, 47, 101, 115, 63

i=5 _20, 30, 47, 63, 101, 115

Donc en sortie: 20, 30, 47, 63, 101, 115.

a L'algorithme de tri par sélection

Déclaration i, k : Naturel

```
01 :  Début
02 :  Pour i ← 0 à nbElements-1 faire
03 :      Pour k ← nbElements-1 à i+1 faire
04 :          Si  t[k] < t[k-1]  alors
05 :           Echanger (t[k],t[k-1])
06 :          Fin _si
07 :      Fin _ pour
08 :  Fin _ pour
09 : fin
```

1.4.2. Médian rapide

a Présentation de l'algorithme

Lorsque le masque est un carré, il existe un algorithme bien plus astucieux et qui peut être paralléliser par partie. L'algorithme est le suivant (dans notre cas, nous avons 3 × 3 valeur à trier) :

Soient les 9 valeurs suivantes à trier [11]

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

– étape 1 : trier chaque ligne, tri de {a, b, c} tri de {d, e, f} et tri de {g, h, i}

– étape 2 : trier chaque colonne, tri de {a, d, g}, tri de {b, e, h}, tri de {c, f, i}

– étape 3 : trier la diagonale secondaire, {g, e, c}

➲ le médian se trouve alors au milieu donc 'e'.

b Démonstration

Chaque étape crée une relation d'ordre partiel par ligne (respectivement par colonne). La figure (1) décrit la construction de ces relations d'ordre. La flèche signifiant la relation "supérieur à".[11]

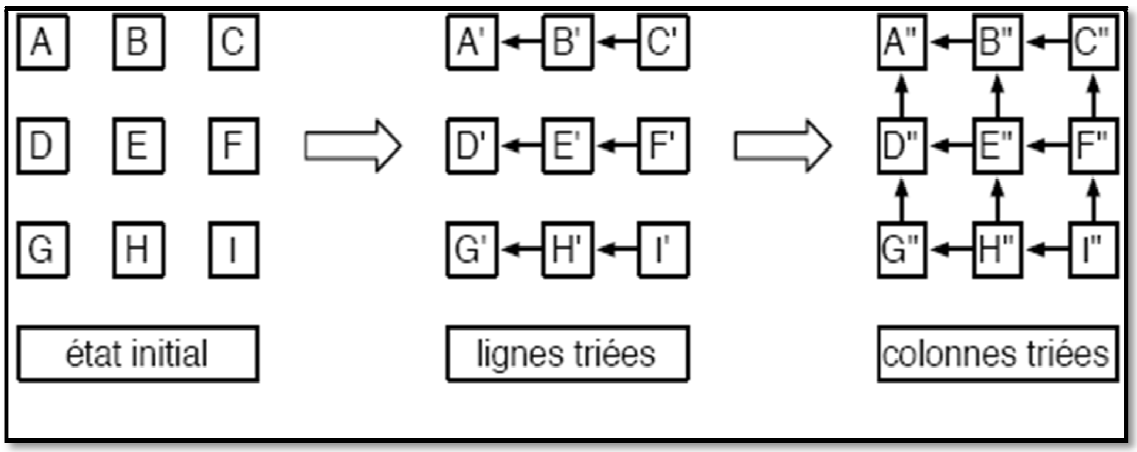


Figure 1.12 **:** tri des lignes et des colonnes

En appliquant une rotation de 45 degré au schéma, (figure 1.13), on voit que les trois points du haut (A", D" et B") sont les trois plus petits (même si D" et B" ne sont pas triés). De même, les trois points du bas (H", F" et I") sont les trois plus grands. La médiane est donc forcément parmi les trois points du milieu (G", E" et C") qu'il ne reste plus qu'à trier. [12]

Une ces trois points triés, la valeur est en « E ».

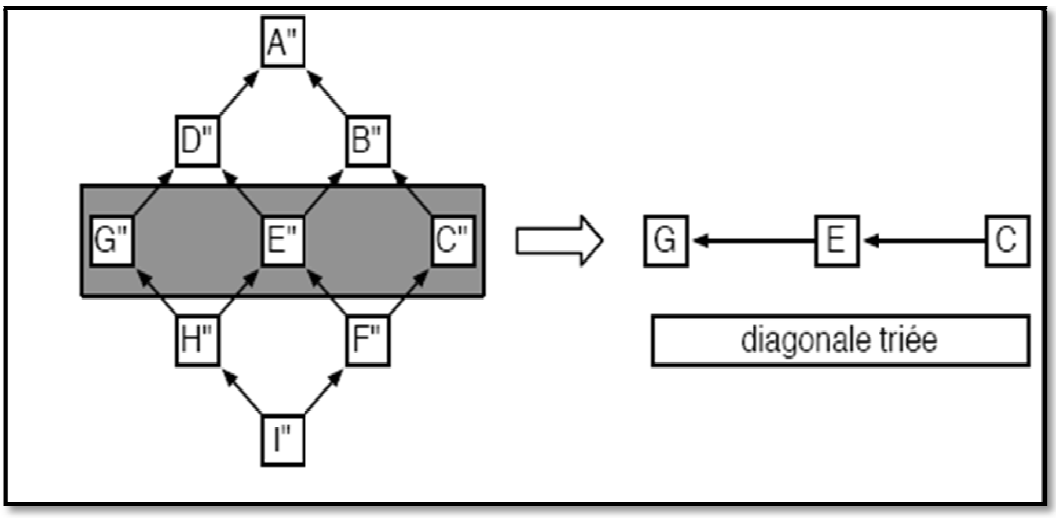


Figure 1.13 : Extraction du médian

c Algorithme de Tri rapide pour le filtre médian

1 : [N, M]= taille d'image ;

2 : **pour j=2 ; j<M ; j++ faire**

3 **: pour i=2 ; i<N ; i++ faire**

4 :%%%%%%%%% TRIER DES LIGNES

5 : p1 = image (i-1,j-1), p2 = image(i,j-1), p3 = image(i+1,j-1),
p4 = image (i-1, j) , p5 = image(i,j), p6 = image(i+1,j),
p7 = image (i-1, j+1), p8 = image (i,j+1), p9 = image(i+1,j+1);

6 : max = Max (p1, p2, p3) ;

7 : min = Min (p1, p2, p3) ;

8 : e1 = (p1 ou p2 ou p3) et (e1 /= max) et (e1 /= min) ;

9 : L1= [max e1 min] ;

10 : max = Max (p4, p5, p6) ;

11 : min = Min (p1, p2, p3) ;

12 : e1 = (p4 ou p5 ou p6) et (e1 /= max) et (e1 /= min) ;

13 : L2= [max e1 min] ;

14 : max = Max (p7, p8, p9) ;

15 : min = Min (p7, p8, p9) ;

16 : e1 = (p7 ou p8 ou p9) et (e1 /= max) et (e1 /= min) ;

17 : L3=[max e1 min] ;

18 : L= classement horizontale (L1, L2, L3) ;

19 : %%%%%%%%% TRI DES COLONNES

20 : p1 = L(1,1), p4 =L(1,2), p7 =L(1,3),
p2 = L(2,1) , p5 = L(2,2), p8 = L(3,2),
p3 = L(3,1), p6 = L(3,2), p9 = L(3,3);

21 : max = Max (p1, p2, p3) ;

22 : min = Min (p1, p2, p3) ;

23 : e1 = (p1 ou p2 ou p3) et (e1 /= max) et (e1 /= min) ;

24 : c1=[max e1 min] ; C1 = c1' ; % ligne → colonne

25 : max = Max (p4, p5, p6) ;

26 : min = Min (p1, p2, p3) ;

27 : e1 = (p4 ou p5 ou p6) et (e1 /= max) et (e1 /= min) ;

28 : c2 = [max e1 min] ; C2=c2' ; % ligne → colonne

29 : max = Max (p7, p8, p9) ;

30 : min = Min (p7, p8, p9) ;

31 : e1 = (p7 ou p8 ou p9) et (e1 /= max) et (e1 /= min) ;

32 : c3 = [max e1 min] ; C3 = c3' ; % ligne → colonne

33 : C= classement verticale (C1, C2, C3) ;

34 : %%%%%%%%% TRIER LE DIAGONAL

35 : max = Max (C(3,1), C(2,2), C(3,1));

36 : min = Min (C(3,1), C(2,2), C(3,1));

37 : e1 = ((3,1) ou C(2,2) ou C(3,1)) et (e1 /= max) et (e1 /=min) ;

38 : la valeur médian = e1 ;

39 : Image filtrée (i, j) = la valeur médian ;

40 : **fin pour**

41 : **fin pour**

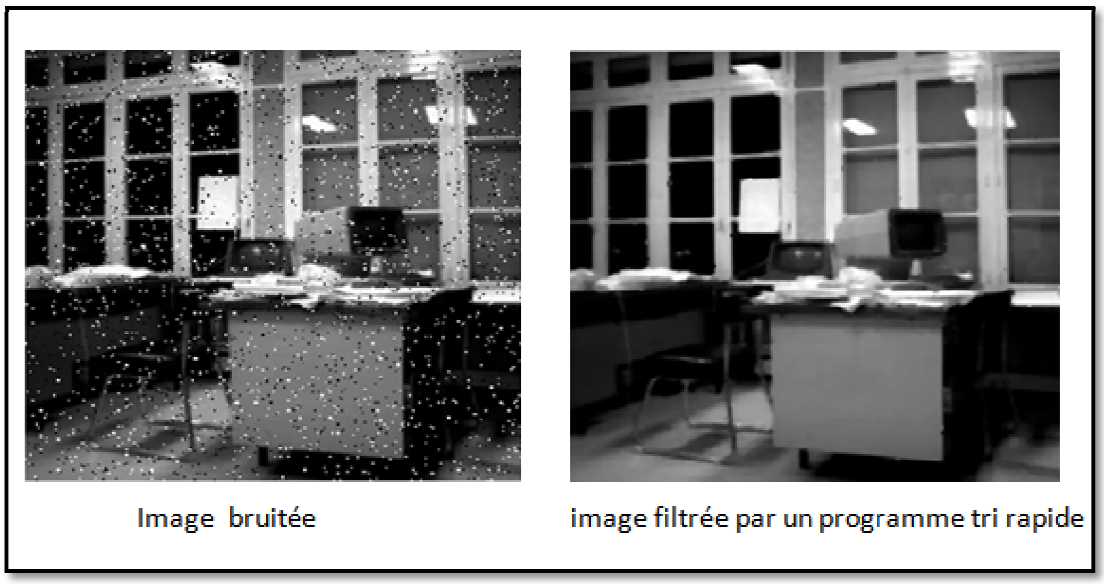


Figure 1.14 **:** Application d'algorithme de tri rapide sur limage Bureau

Remarque

Existe une commande matlab directe pour l'application d'un filtre médian sur une image cette commande est **medfilt2.**

Examples

```
01:  I = imread ('bureau.gpg');
02:   II = medfilt2 (I);
03:   Figure, imshow (II)
```

Dans la figure (1.4) on trouve l'image bureau filtrée par la commande matlab **medfilt2** ainsi la comparaison entre les deux images, image filtré par notre programme et l'image filtré par la commande matlab, la comparaison sur matlab se fait tout simplement par la différence (-).

Figure 1.15 : application **medfilt2** sur image bureau puis la comparaison

1.4.3. Principe du tri à bulles

Au lieu d'utiliser le tri par sélection ou le tri par insertion pour trier les valeurs par 3 (en ligne ou en colonne), c'est le tri à bulles qui va être utilisé car il a la particularité d'avoir une structure itérative régulière. Il s'implante particulièrement bien en VHDL. Soit l'opérateur mM (pour minMax) qui prend en entrée deux valeurs et qui renvoie les deux valeurs triées (la valeur de gauche est la valeur min, la valeur de droite est la valeur max) : [12]

mM(a, b) = (min(a, b), max(a, b))

L'opérateur mM peut aussi être vu comme une simple permutation si b < a (dans le cas contraire rien n'est fait) :

$$mM(a,b) = \begin{cases} (a,b)\ si\ \ a \leq b \\ (b,a)\ \ si\ \ non \end{cases}$$

Trier a, b, c consister à :

1. trier a, b : mM(a, b)
2. trier b, c : mM(b, c)
3. trier a, b : mM(a, b)

A la fin la plus petite valeur est en '**a**' (min), la plus grande est en '**c**' (max) et la valeur médiane au milieu. [12]

C'est le principe du tri à bulles qui fait remonter les plus petites valeurs en début de tableau, et les plus grandes en fin de tableau (figure 3).

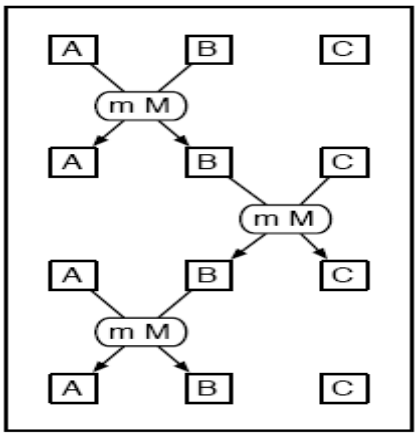


Figure 1.16 : principe du tri a bulles appliqué a 3 valeurs

1.5. Description VHDL du filtre médian

1.5.1. Schéma fonctionnel du filtre médian

Le filtrage médian est à base des comparateurs, ils (les comparateurs) sont utilisés pour classer les pixels (X1, X2….X9) sélectionnés par le masque du filtre dans un ordre croissant ou bien décroissant.

Il est constitue de « 30 » comparateurs (figure 1.17).

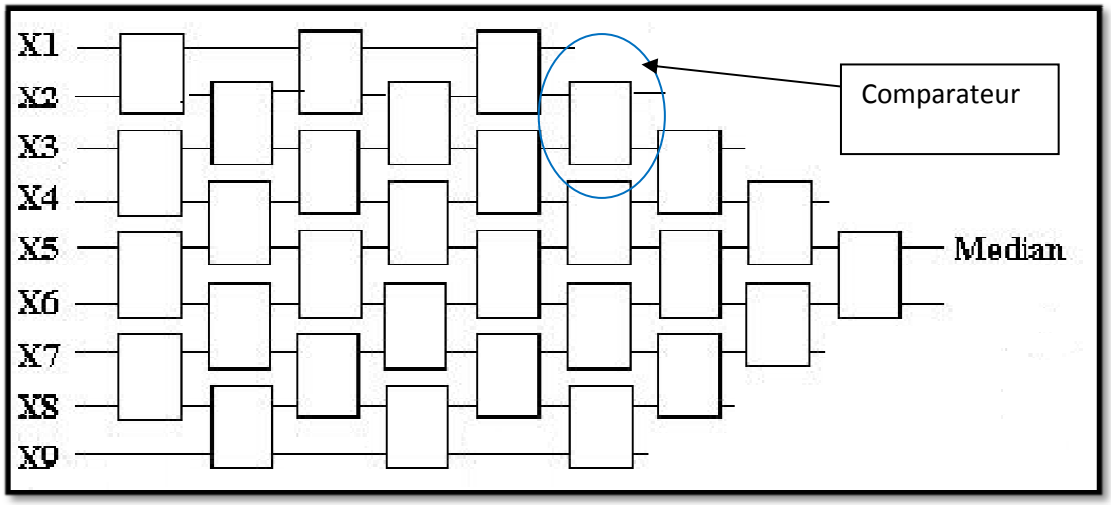


Figure 1.17 **:** schéma bloc du filtre médian à base des comparateurs

Chaque comparateur du filtre effectue une sélection des valeurs deux à deux. L'implantation du circuit de la **figure (03) compare** en entrée deux valeurs (A, B) et renvoie en sortie les deux valeurs triées (Max (A, B), Min (A, B)).

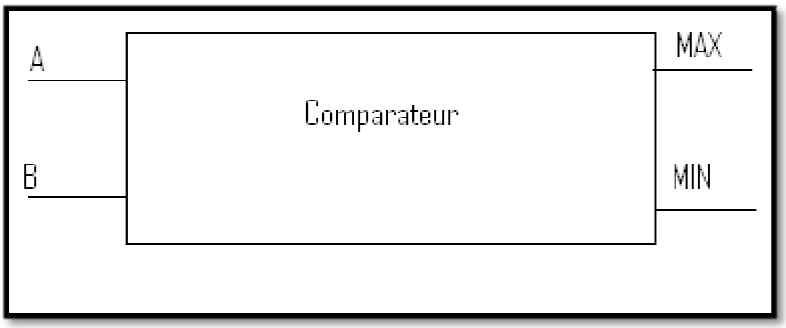


Figure 1.18 : implantation du circuit de comparaison

1.5.1 Principe de fonctionnement du filtre médian

Etant donné l'image à filtrer, il s'agit de prendre un masque de sélection 3X3 et de le faire balayer sur toute l'image en commençant par les trois premières lignes , et a chaque 3X3 pixels on sélectionne par tri la valeur médiane qui va occuper la position centrale du masque dans une nouvelle image.

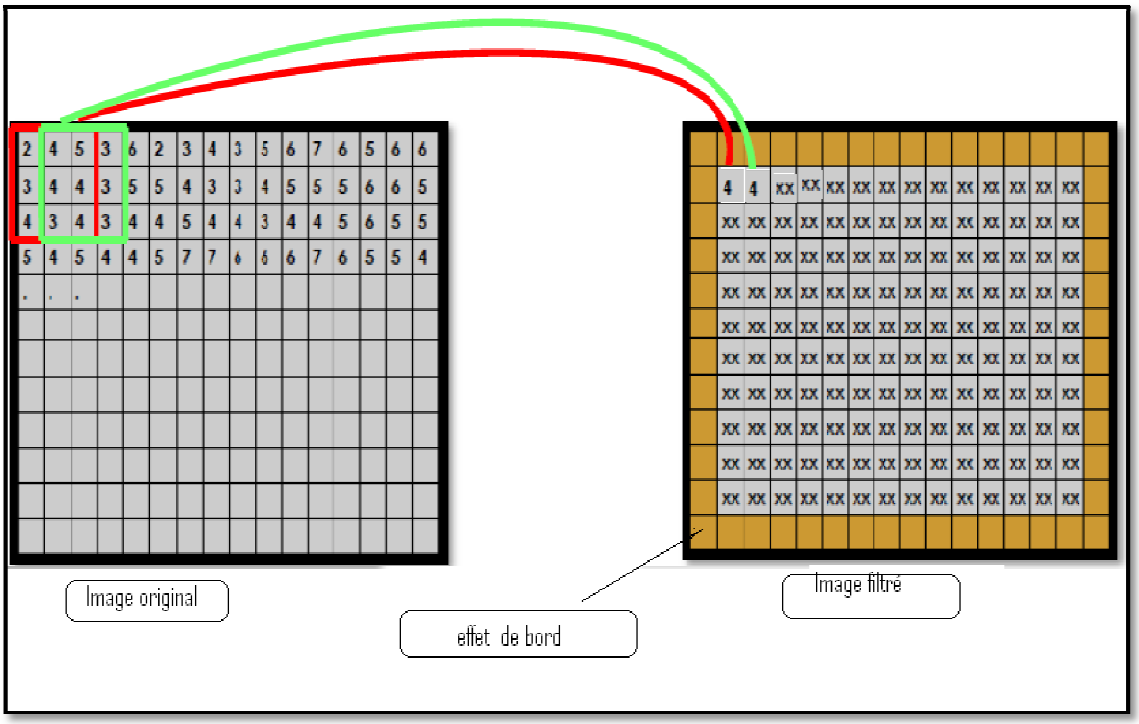


Figure 1.19 : le Principe du balayage du filtre médian

L'opération du tri consiste à comparer dans un premier temps tous les pixels sélectionnés par le masque deux à deux cette opération est réitérée huit fois, pour donner à la cinquième ligne de comparaison au niveau de la sortie max du dernier comparateur la valeur médiane (figure 1.19).

1.5.2 Organigramme de comparaison et Description VHDL du filtre

L'organigramme de la cellule de base de comparaison ainsi que La description VHDL correspondante à une comparaison du filtre médian sont donnés dans les figures (1.20 et 1.21) respectivement.

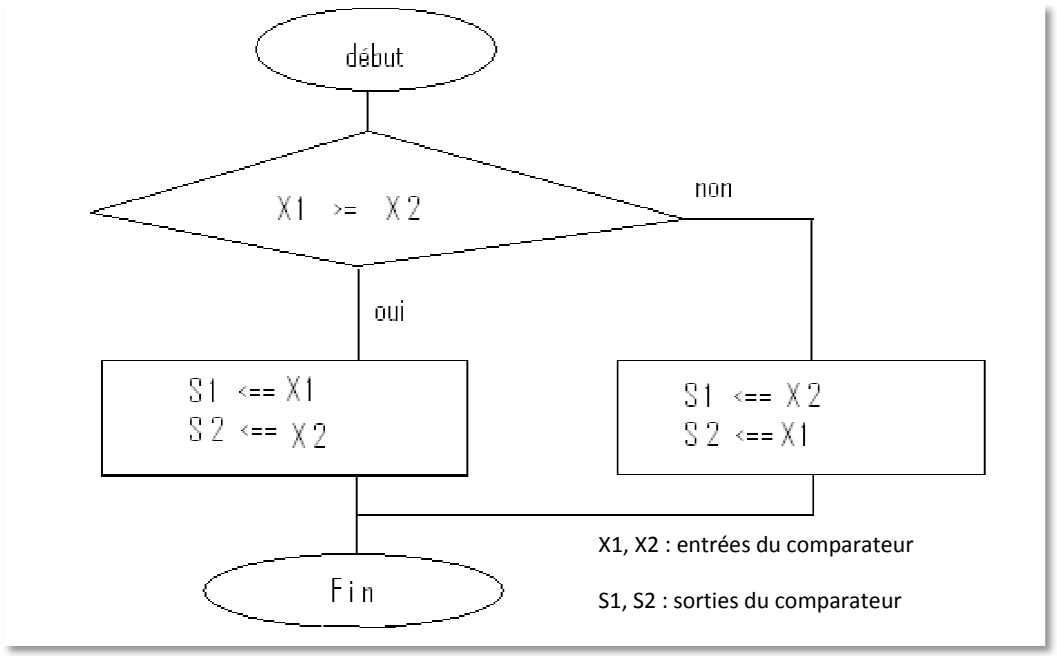


Figure 1.20 **:** L'organigramme dune cellule de base (comparateur)

```
Process (clk)
Begin
        if(clk'event and clk ='1') then
        if(X1 >=  X2)then
        S1 <= X1;
        S2 <= X2;
        else
        S1 <= X2;
        S2 <= X1;
End if;

End if;
```

Figure 1.21 : La description VHDL d'un comparateur

1.6 Gestion de flux de donnée

La comparaison pratiquée pour traiter l'image utilise un flux qui est généré à partir des lignes de l'image.

A partir d'une image 512*512 on extrait les lignes séquentiellement puis les pixels sont à leur tour extraits à partir des lignes. Le flux de données est synchronisé par l'horloge du FPGA. [2]

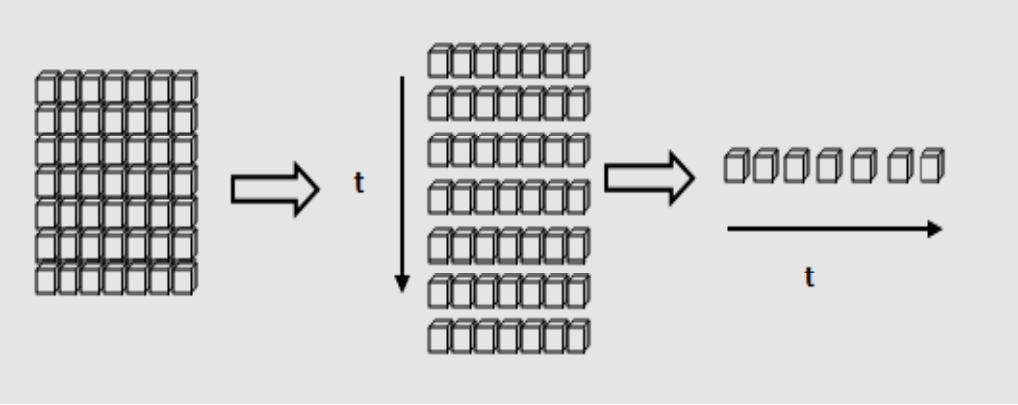


Figure 1.22 : flux de données de la source.

L'extraction des lignes consiste à construire un module qui permet de marquer la succession de ces dernières. Un retard ligne de pixels est généré pour la réalisation de la convolution, il nécessite des mémoires de type FIFO.

Le FPGA spartan 3E offre une RAM bloc qu'on peut la considérer comme une FIFO, sa description VHDL est la suivante : [2]

```
RAMB1: RAMB16_S9
generic map (
INIT => X"000", -- Value of output RAM registers at startup
SRVAL => X"000", -- Ouput value upon SSR assertion
WRITE_MODE => "WRITE_FIRST", -- WRITE_FIRST, READ_FIRST or NO_CHANGE
-- The following INIT_xx declarations specify the initial contents of the RAM
-- Address 0 to 511
INIT_00 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_01 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
port map (
DO => DO_1, -- sortie de la donné sur 8-bits.
DOP => DOP_1, -- 1-bit sortie de parité.
ADDR => add, -- addressage sur 11-bits.
CLK => clk, -- horloge.
DI => d_in, -- entrée de la donné sur 8-bits.
DIP => "0", -- 1-bit entrée de parité.
EN => '1', --bit activation de la RAM.
SSR => '0', -- Reset entrée de synchronisation.
WE => we_1 -- 1 bit d'activation en écriture.
);
```

Conclusion

Pour le traitement des images on a choisi le filtre médian car il est plus adapté que le filtrage linéaire pour réduire les bruits surtout le bruit poivre et sel (bruit impulsionnel). On a présenté le développement sous VHDL du filtre choisi.

Dans ce chapitre on s'est basé sur le filtrage d'une image mais le but principal est de traiter une vidéo en temps réel (25 images par seconde selon le standard cinéma), donc ce la nous oblige à donner quelques notions de base sur les vidéos (Caractéristiques d'un signal vidéo, l'extension, le codec,…), cette partie est étudiée dans le chapitre qui suit.

Notions sur les extensions vidéo

Introduction

Le projet consiste à réaliser un filtre sur FPGA

et appliquer sur une sequence vidéo .

l'entrée de filtre doit provenir d'une source vidéo.

La vidéo regroupe l'ensemble des techniques permettant l'enregistrement ainsi que la restitution d'images animées, accompagnées ou non de son, sur un support électronique et non de type photochimique. Le mot « vidéo » vient du latin « video » qui signifie : « je vois »..

Dans ce chapitre alors on va citer les différents types des extentions vidéo les plus souvent utilisées pour la codec vidéo (compression et la décompression), pour faire un bon choix de la source vidéo de notre projet.

2.1 Le Signal Vidéo Numérique

Le principe de la numérisation d'une image vidéo est assez simple. La première étape consiste à sous diviser chaque image vidéo selon une résolution donnée(normalement 720 x 486 pixels pour une image vidéo normale) et à associer une valeur numérique à chacun des éléments qui forment la couleur de ce pixel(YUV ou RGB) en utilisant une table de conversion de couleurs(normalement 24 bits par pixels pour 16 millions de couleurs possibles en chaque point). [13]

Ce procédé de conversion doit se faire très rapidement étant donné qu'une image vidéo traditionnelle contient plusieurs milliers de pixels et que la vidéo analogique pour le mode NTSC défile à près de 30 images par seconde (30 images de 720 x 576).

Si un signal vidéo de 720x486 pixels de résolution est numérisé en utilisant la norme YUV, le fichier résultant sera de 1025,16 Ko par image ou 30,03 Mo/sec. C'est ce qu'on appelle le format non compressé pixels par seconde pour le mode PAL, Ces valeurs sont calculées en annexe. [13]

On remarque qu'avec un débit d'environ 30 Mo/sec, la vidéo numérique non compressée exigerait donc plus de 1.8 Go d'espace, pour capter une (01) seule minute de vidéo ce qui rend la capacité de stockage trop importante. Ceci d'une part, d'autre part dans la transmission vidéo le transfert de ces données en format numérique. Certaines technologies permettent actuellement le transfert des données vidéo numériques non-compressées sauf qu'elles ne sont pas toujours facilement accessibles. Pour les technologies plus accessibles il faut donc penser a réduire le débit des données. Une solution peut être envisagée ; c'est celle de la compression de données [13]

2.2. La compression de données

De nombreux formats de compression disponibles La compression des images et des données vidéo peut suivre deux approches différentes :

2.1.1 la compression sans perte (lossless)

Dans le cas d'une compression "lossless", C'est-à- dire sans perte, chaque pixel est maintenu intact. L'image obtenue après compression est donc identique à l'original. Cependant, le prix à payer est que le gain, en termes de réduction des données, est très limité. Un format de compression "sans perte" bien connu est le format GIF. Du fait de son faible taux de compression, ce format ne convient guère Aux solutions de vidéo sur IP nécessitant l'archivage et la transmission de quantités importantes d'images. [10]

Voilà pourquoi plusieurs méthodes et normes de compression ont été développées comme la compression “lossy.[10]

2.1.2 La compression avec perte (Lossy)

Le principe fondamental de la compression “lossy” est de réduire les éléments invisibles à l’œil humain et d’accroître ainsi considérablement le taux de Compression.

Les méthodes de compression suivent également deux approches différentes par rapport aux normes de compression : compression des images fixes et compression vidéo. Normes de compression des images fixes toutes les normes de compression des images fixes ont la particularité de se concentrer Sur une seule image à la fois. La norme la plus connue et la plus répandue en la matière est JPEG. [10]

2.3 Présentation de quelques techniques de compression vidéo

Pour la compression vidéo on peut procéder par plusieurs façons :

- On réalise une compression d’images : ici la taille des images est réduite et elles sont compressés au format JPEJ, l'inconvénient de cette technique est la perte de la qualité d’image.
- Suppression des informations inutiles : les élément identique d’une images l’autre sont enlevés et on ne garde que les partie en mouvement de l’images, , l'inconvénient de cette dernière est la perte des détails.
- On peut réalisé une réduction du nombres des images par seconde, ici on supprime de temps en temps une image par seconde (ex : 1 image sur 5),par conséquent on perte en qualité de l’animation.

2.4 Les extensions des formats vidéo

Il existe plusieurs formats de vidéo numérique. Ces formats compressent les fichiers car sans la compression les fichiers seraient énormes. Voici la taille sans compression 1sec = 22Mo. La vidéo est compressée par un CODEC (Codeurs/Décodeurs). Il existe deux types de CODEC, les CODEC logiciels et les CODEC matériels. Pour qu'une vidéo soit lisible sur un ordinateur, il faudra donc qu'un CODEC approprié soit installé. Une fois le Codec installé, une simple application telle que le Windows Media Player ou Winamp suffira pour la lire.

Ces formats se reconnaissent par leurs extensions ci dessous:

2.4.1 AVI (Audio Vidéo Interleave)

Format de fichier d'animations mis au point par Microsoft, qui fonctionne sur n'importe quelle machine, et ne nécessite pas de carte d'extension particulière. Dans ce format, la compression est toujours effectuée image par image, dans l'en-tête d'un fichier AVI se trouve un champ qui définit le type de compresseur utilisé par le flux vidéo enregistré. [16]

2.4.2 MOV

Ces Fichiers peuvent être lus par le logiciel QuickTime.

Le format a été créé par Apple Computer pour travailler avec des fichiers multimédia. Bien que les fichiers MOV sont souvent trouvés sur le web, de les jouer sur un ordinateur Windows on doit installer un composant supplémentaire ou convertir en un autre format. [8]

MOV est un format conteneur et peut contenir de la vidéo, l'animation, le graphisme, L'avantage de MOV est la capacité à contenir des références abstraites de données pour les données multimédia. Cela signifie qu'ils peuvent être facilement éditées - pas besoin de réécrire toutes les données des médias après l'édition. [8]

2.4.3 La technique INDEO

La compression INDEO inclut des techniques de compression identiques à celles utilisées dans la norme MPEG-2 comme la prédiction bidirectionnelle, elle inclut aussi d'autres Caractéristiques telles que la transparence. La résolution maximum est de 320 x 240 avec un affichage de 30 images par seconde. [14]

2.4.4 H.263

La technique de compression H.263 est conçue pour une transmission
vidéo à débit fixe. L'inconvénient du débit fixe est que l'image perd de sa qualité lorsque les objets sont en mouvement. La norme H.263 était initialement destinée aux

applications de vidéoconférence où il ya pas beaucoup de mouvement et non à la surveillance où les détails ont plus d'importance. [14]

NOTE : Avec la compression de série H, l'image d'une personne en mouvement ressemble à une mosaïque. L'arrière-plan, qui est généralement sans intérêt, conserve toutefois une bonne qualité d'image et une bonne clarté.

2.4.5 H.264

La toute dernière norme de compression vidéo H.264, est appelée à devenir la norme vidéo de référence au cours des prochaines années. Elle a déjà été intégrée avec succès Dans des gadgets électroniques tels que les téléphones mobiles et les lecteurs vidéo Numériques. Dans le secteur de la vidéo surveillance, le H.264 offre de nouvelles possibilités en termes de réduction des frais de stockage et de renforcement de l'efficacité globale. [14]

Le H.264 (également connue sous l'appellation MPEG-4 Partie 10/AVC) est une norme Ouverte sous licence, compatible avec les techniques de compression vidéo les plus efficaces d'aujourd'hui. Un encodeur H.264 peut réduire la taille d'un fichier vidéo Numérique de plus de 80 % par rapport à la norme Motion JPEG et de 50 % par rapport à la norme traditionnelle MPEG-4 Partie 2, sans que la qualité d'image ne soit compromise. L'importance de ces gains rend le H.264 extrêmement utile pour les applications de vidéosurveillance. [14]

2.4.6 MPEG (Motion Picture Expert Group)

Groupe d'experts chargés de mettre au point un format de compression vidéo. Et, par extension, les formats eux-mêmes (extension « .MPG » sous MS-DOS).

La norme MPEG (fondée par le Motion Picture Experts Group à la fin des années 1980) est la plus connue des techniques de transmission directe audio et vidéo. Dans cette section, nous nous limiterons à la partie vidéo de la norme MPEG. [16]

Le principe de base du MPEG consiste à comparer entre elles deux images compressées destinées à être transmises sur le réseau. La première des deux images servira de trame de référence. Sur les images suivantes, seuls seront envoyées les

zones qui différent de la référence. L'encodeur réseau reconstruit alors toutes les images en fonction de l'image de référence et de la "plage de différence".[16]

Bien que plus complexe que la technique Motion JPEG, la compression vidéo MPEG produit de plus petits volumes de données à transmettre via le réseau. À la page suivante, nous vous proposons une illustration de cette technique consistant à ne transmettre que les différences existant entre la deuxième et la troisième trame.

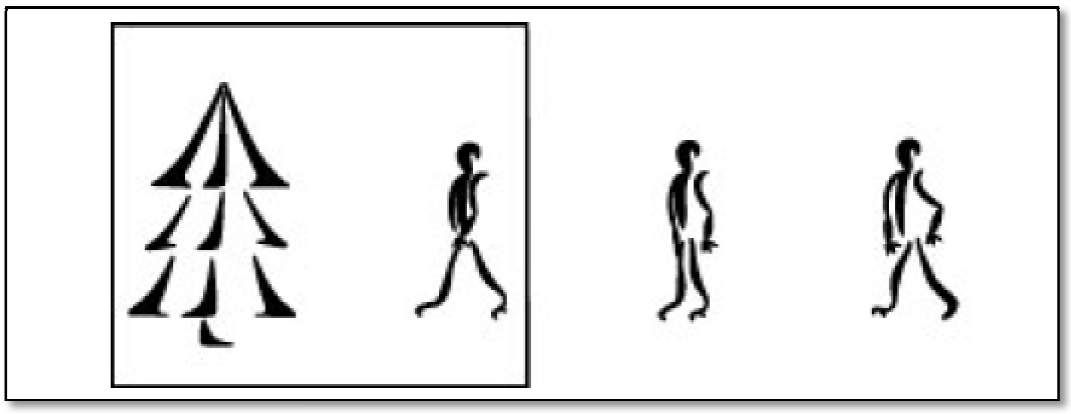


Figure.2.1 **:** illustration la technique de compression MPEG

MPEG est en réalité bien plus complexe que l'ébauche ci-dessus. Cette méthode implique bien souvent des techniques ou des outils supplémentaires permettant de gérer certains paramètres tels que la prédiction du mouvement dans une scène ou l'identification des Objets. [7], [16]

Dans le format MPEG, on distingue plusieurs méthodes d'encodage (compression et décompression de données, qu'on appelle aussi codec), qui permettent d'optimiser le rapport qualité/taille. Parmi ces formas on distingue les suivants :

a MPEG1

Format 360 X 240 (pixels), appelé aussi format VCD (vidéo

CD). Lancée en 1993 et destinée à l'archivage des données vidéo numériques sur CD. La plupart des encodeurs et des décodeurs MPEG-1 sont conçus pour un débit d'environ 1,5 Mbit/s en résolution CIF. MPEG-1 met surtout l'accent sur le maintien d'un débit relativement constant, au détriment de la qualité d'image, laquelle est variable et Comparable à la qualité vidéo VHS. En MPEG-1, la fréquence d'image est plafonnée à 25 Images par seconde. [7], [16]

b MPEG2

Format 760 X 540 (pixels), environ 700 Mo/10 mn, (11.5Mo/s),

Appelé aussi format DVD. Approuvée en 1994, était destinée à la vidéo numérique de qualité supérieure (DVD), à la télévision haute définition (HDTV), aux supports d'enregistrements interactifs (ISM), aux systèmes d'émission vidéo numérique et à la télévision par câble (CATV).

Le format MPEG-2 visait à accroître la technique de compression de la norme MPEG-1 afin de couvrir des images plus grandes et de meilleure qualité, mais aux dépens d'un taux de Compression plus faible et d'un débit d'images plus rapide. La fréquence est plafonnée à 25 images par seconde, tout comme en MPEG-1. [7], [16]

c MPEG3

Le MPEG3 était à l' origine destinée aux très hauts débits mais ne vit pas le jour en tant qu'entité puisqu'il fut assimilé au standard MPEG2. [7]

d MPEG-4

Représente une évolution substantielle par rapport au forma

MPEG-2. Les outils permettant de réduire le débit d'images de manière à atteindre une certaine qualité pour une application ou une scène déterminée sont beaucoup plus nombreux en MPEG-4. En outre, la fréquence n'est plus limitée à 25 ou 30 images par seconde. Soulignons cependant que la plupart des outils actuels permettant de réduire le débit ne concernent que les applications en temps réel.

Ceci est dû au fait que ces outils requièrent des capacités telles que les durées d'encodage et de décodage (temps de latence) les rendent quasiment impossibles à utiliser à d'autres fins que pour l'encodage de films en studio, de films d'animation, etc.

En réalité, la majorité des outils MPEG-4 destinés aux applications en temps réel sont les mêmes que ceux qui existent pour les formats MPEG-1 et MPEG-2. L'essentiel est de choisir une norme de compression largement utilisée, qui assure une bonne qualité d'image, soit par exemple M-JPEG ou MPEG-4. [7] ; [16]

e MPEG7 et MPEG21

Le MPEG7 est le dernier né de la famille des codecs MPEG, et a reçu sa première esquisse officielle en Septembre 2000. Le MPEG7 ne concerne plus vraiment la compression vidéo mais traite essentiellement de contenu multimedia et d'interactivité. Le MPEG21 étend le MPEG7, et est encore en pleine naissance. [7]

2.5 La comparaison entre quelques types des extensions vidéo et sélection d'une technique de compression.

Le but principal de compression vidéo est d'augmenter le débit de transmission. Ainsi une vidéo compressée est plus protégée qu'une vidéo non compressée.

La figure suivante montre les différents débits en (Kbits/s) de quelques types des extensions vidéo.

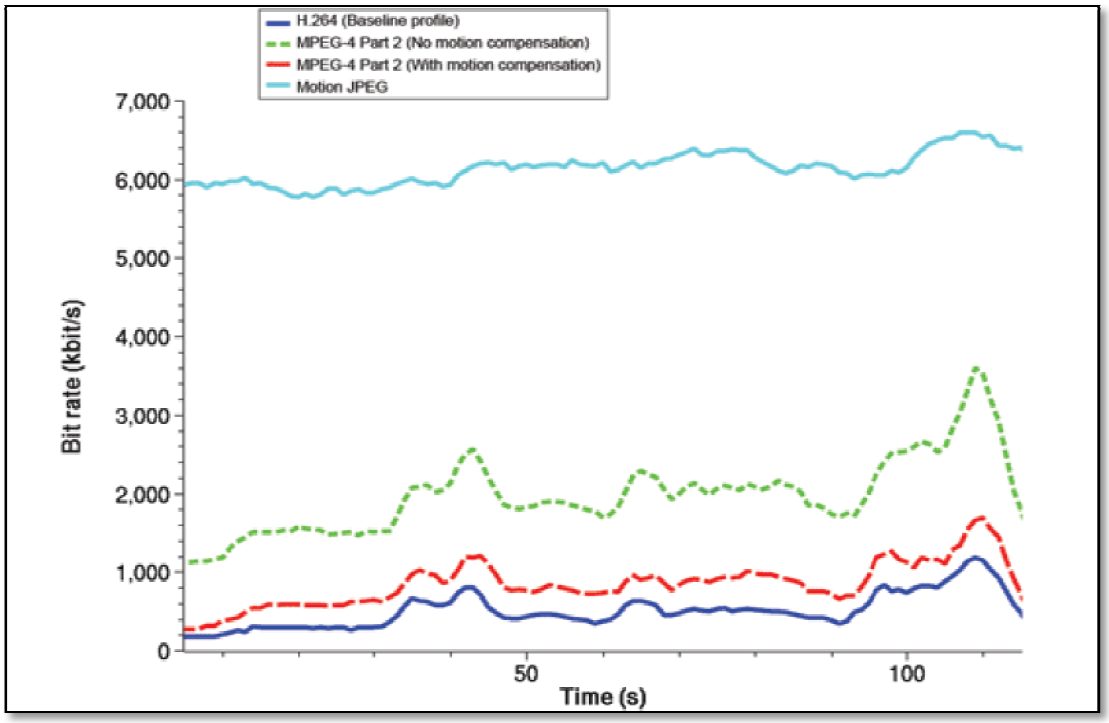


Figure.2.2 : La comparaison entre quelques types des extensions vidéo

A partir de la figure 2-6 précédente on remarque que l'extension MPEG a une chance d'être notre source, car elle a un débit très important et par conséquence un taux de compression très élevé.

Donc comment fonctionne le format MPEG et quelle est leur technique de compression, cette partie est développée ci dessous.

2.6 Techniques communes au MPEG 1/2/4

2.6.1 Espaces de couleurs

Pour travailler sur des images numériques, on discrétise ces images. On représente donc une image par une fonction à deux variables, les coordonnées, qui renvoient la couleur au point demandé de l'image. La représentation classique d'une image numérique utilise un espace de couleur dit RGB. Cela correspond a une représentation discrète de l'image où l'on quantifie la couleur en chaque point par trois valeurs : Rouge, Vert et Bleu (RVB ou RGB en anglais).

Cette présentation a cependant plusieurs défauts : elle utilise une grande quantité d'information et ne tient pas compte du fait que l'œil humain ne perçoit pas les couleurs en RVB mais au travers de deux types de cellules :

Les une perçoivent la luminosité (en noir et blanc), et les autres la coloration. Les codecs MPEG ne compressent donc pas des images RVB, mais utilisent un espace de couleurs plus approprié, de façon à mieux profiter du format de la vision humaine. Le format adopté est un format dit YUV. [4]

Y représente la luminosité U représente la première valeur de chrominance(ou chroma) V représente la deuxième valeur de chrominance(ou chroma)

Il faut évidement 3 composantes indépendantes pour pouvoir avoir un espace de couleur complet. [4]

Les formules permettant de passer d'un espace RVB a YUV et inversement sont assez simples, et correspondent à des produits matriciels (Ce sont des changement de bases, puisque ce sont des espace vectoriels de dimension 3) [4].

Y = (0.257 * R) + (0.504 * G) + (0.098 * B) +16

Cr = V= (0.439 * R) - (0.368 * G) - (0.071 * B) +128

Cb = U= - (0.148 * R) - (0.291 * G) + (0.439 * B) + 128

Et

B = 1.164 * (Y - 16) + 2.018 (U - 128)

G = 1.164 (Y - 16) – 0.813 (V-128) – 0.391 (U - 128)

R = 1.164 (Y - 16) + 1.596 (V - 128)

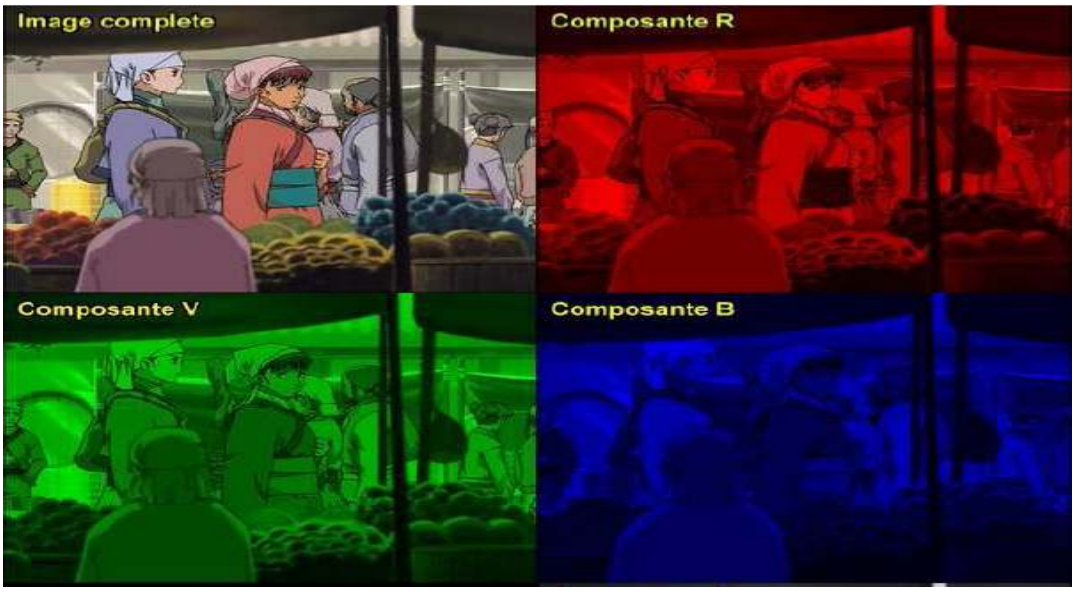


Figure.2.3 : différentes composantes R G B

Figure 2.4 : différentes composantes Y U V

L'utilisation du format YUV permet de profiter d'une caractéristique de l'œil humain sa capacité de distinction des couleurs est plus faible que celle de distinction de la luminosité. Cela permet de réduire la quantité d'information des espaces de

chrominance par rapport à celle de luminosité, on fait donc ce qu'on appelle du sous-échantillonnage sur les composantes U et V. Cela correspond en général à les réduire de moitié de résolution (et donc à diviser par 4 leur taille en mémoire), pour une perte de qualité assez faible(le gain en espace est important) [4]

Figure.2.5 : Les composantes U et V sont sous-échantillonnées

On distingue plusieurs espaces YUV aux caractéristiques différentes, utilisées par les formats MPEG 1/2/4. Les compressions vidéos MPEG 1/2/4 utilisent toutes à la base le format dit YV12 (YUV 4 :2 :0). Cependant depuis le MPEG2, le support de formats professionnels de haute qualité, dit YUV 4 :2 :2, est possible. En pratique ceux-ci ne sont quasiment jamais utilisé. [4]

2.6.2 transformée en cosinus discrète, quantification et compression du bloc

La DCT permet de transformer un bloc d'une composante, en un ensemble de fréquences décrivant le même ensemble (c'est un changement de représentation isomorphe). Une fois de plus le but final et de profiter des faiblesses de l'œil humain qui remarque beaucoup moins une perte de données réparties que localisée. (Un peu de bruit dans l'image généra beaucoup moins que quelques pixels complètement faux)
La définition formelle de la DCT à deux dimensions est la suivante :

F(u,v) est la transformée, c'est la fonction qui donne la valeur pour le couple de fréquence (u,v). [4]

$$\mathrm{F(u,v)} = \frac{2}{\mathrm{N}} * \mathrm{C(u)} * \mathrm{C(v)} * \sum_{\mathrm{x=0}}^{\mathrm{N-1}} \sum_{\mathrm{y=0}}^{\mathrm{N-1}} \mathrm{F(x,y)} \cos(\frac{(2x+1)u\pi}{2N}) \cos(\frac{2y+1}{2N})$$

Avec : u, v, x, y = 0, 1, 2,….N – 1.

Ou x et y sont les coordonnées spatiale ou u et v sont les coordonnées dans la transformée

Et avec $C(u).C(v) = f(x) = \begin{cases} \frac{1}{\sqrt{2}} pour \ \ u,v = 0 \\ 1 \ \ si \ \ non \end{cases}$

Et la transformation inverse, l'IDCT est défini ainsi

$$\mathrm{F(x,y)} = \frac{2}{\mathrm{N}} * \sum_{\mathrm{u=0}}^{\mathrm{N-1}} \sum_{\mathrm{v=0}}^{\mathrm{N-1}} \mathrm{C(u)} * \mathrm{C(v)} * \mathrm{F(u,v)} \cos(\frac{(2x+1)u\pi}{2N}) \cos(\frac{(2y+1)u\pi}{2N})$$

En pratique dans les encodages MPEG, la DCT est utilisée sur des blocs. Le bloc est transformé dans le domaine fréquentiel par la DCT. Au moment du décodage on applique ce que l'on Apelle l'IDCT (Inverse Direct Cosine Transform) qui permet de repasser au domaine spatial. La quantisation intervient juste après avoir transforme un bloc via la DCT.

La quantisation revient a diviser un tableau par un autre tableau. On divise chaque valeur obtenu par DCT par la valeur correspondante dans un tableau. La méthode classique utilisée par le MPEG4, le H263, divise par une même valeur tout les coefficients. Ceci permet de "simplifier" l'information contenue, et donc de rendre la compression plus facile. En effet, on ramène l'ensemble des valeur à un ensemble plus petit, ce qui le rendra plus aisement compressible par une compression de type entropique classique (comme huffman). [4]

Exemple 1 *Soit le bloc de taille 8x8 suivant :*

$$\begin{pmatrix} 150 & 170 & 132 & 185 & 147 & 190 & 215 & 220 \\ 165 & 185 & 130 & 190 & 175 & 196 & 223 & 199 \\ 155 & 163 & 180 & 220 & 202 & 173 & 197 & 170 \\ 143 & 154 & 160 & 170 & 211 & 185 & 190 & 166 \\ 130 & 140 & 172 & 190 & 193 & 150 & 180 & 140 \\ 135 & 164 & 198 & 180 & 177 & 141 & 172 & 135 \\ 170 & 190 & 163 & 140 & 165 & 132 & 160 & 140 \\ 160 & 200 & 145 & 135 & 170 & 199 & 190 & 129 \end{pmatrix}$$

Une fois transformé par DCT on obtient :

$$\begin{pmatrix} 338 & -49 & -39 & 10 & -23 & 12 & -64 & 5 \\ 62 & -78 & 16 & -13 & 35 & 24 & -6 & -42 \\ 8 & -7 & 76 & 27 & -27 & -13 & -10 & -44 \\ -31 & 1 & 8 & -40 & 6 & 12 & 4 & 3 \\ -9 & -34 & 0 & 24 & -10 & -7 & 7 & -8 \\ -9 & 10 & 14 & -10 & 10 & -13 & 10 & 11 \\ 11 & 3 & -29 & -20 & -7 & 14 & 9 & 0 \\ -8 & 14 & -8 & -17 & 16 & 13 & -2 & 2 \end{pmatrix}$$

On applique alors la quantisation qui revient a diviser chacun des coefficients par la matrice du JPEG par exemple :

$$\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

On obtient :

$$\begin{pmatrix} 21 & -4 & -3 & 0 & 0 & 0 & -1 & 0 \\ 5 & -6 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

On parcourt ensuite la matrice de façon d'obtenu un zigzag comme indiqué dans la figure 2. .

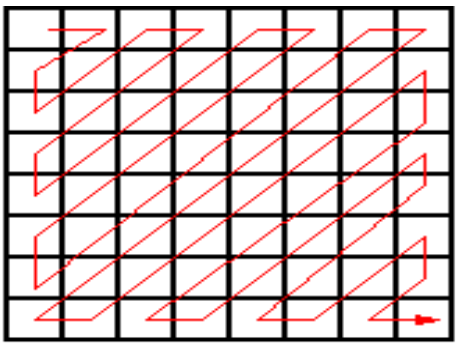


Figure 2.6 : le chemin de zigzag

Exemple 2 *Par exemple dans l'exemple précédent on obtient :*

21
-4 5
0 -6 -3
0 1 0 -2
0 0 4 0 0
0 1 1 0 -1 0
0 0 0 -1 0 0 -1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0
0 0 0 0
0 0 0
0 0
0

On obtient donc une suite de coefficients, que l'on compresse à l'aide d'une compression entropique du type huffman. Grâce à la quantisation on a de nombreux zéros, et la compression est bonne.

En jouant sur les coefficients de quantisation on peut facilement "rajouter ou enlever" de l'information et ainsi Controller la taille du bloc compressé. Le résultat à évidemment un aspect de plus en plus différent de l'image d'origine par ce processus, ou l'on augmente la quantisation, puisque l'on élimine de l'information par arrondi. [4]

2.6.3 Type de frames et fonctionnement global

Il existe trois types de frame dans la compression MPEG : Les I frames, dites Intra Frames, ou Key frames Les P frames, dites Inter Frames, Delta Frames, ou prédiction frames Les B frames, ou Bidirectionnel frames.

a les I Frames

Les I frames ressemblent beaucoup à une compression de type JPEG. Leur codage revient simplement à prendre l'image, la découper en blocs de 8x8 pixel, et à coder chaque bloc via DCT, quantisation et compression (cf Figure). On modifie le coefficient de quantisation si l'on veut changer la taille finale, puisque ce coefficient permet de jouer sur l'information restante présente. Leur nom, Intra, dérive du fait qu'elles sont complètement indépendante de toute autre information dans la vidéo ces frames sont bien entendue très rapidement gourmandes en mémoire, mais possèdent l'avantage d'avoir une image entière codée. [4]

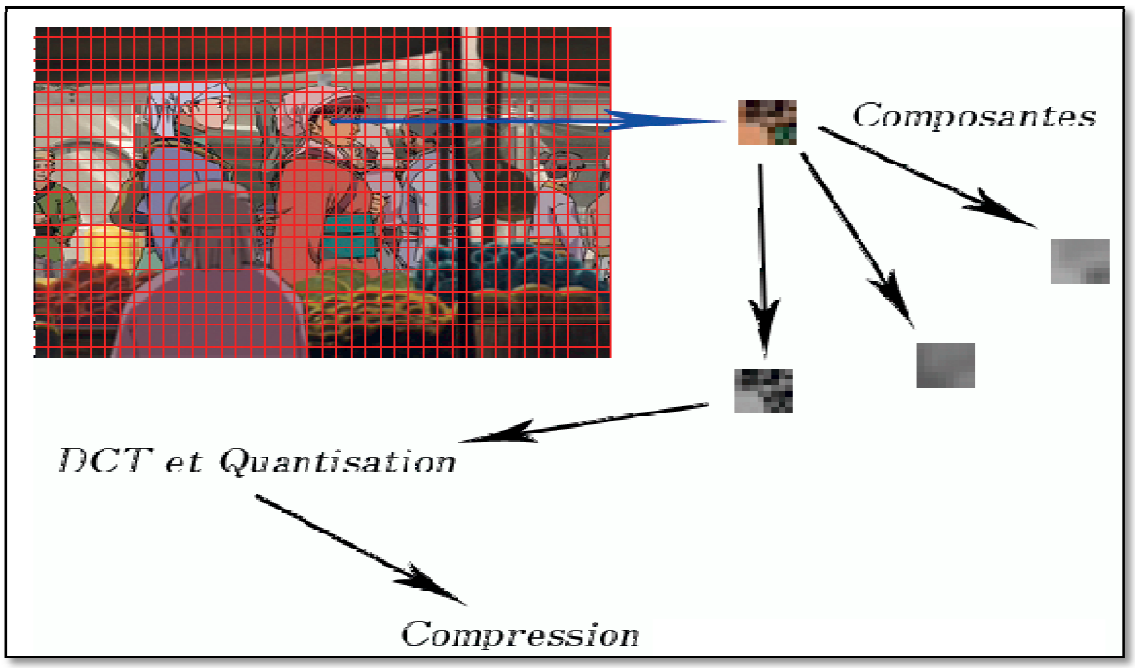


Figure 2.7 **:** Compression d'une I Frame

b Les P-Frames et B Frames

Les P frames sont obtenus de manière plus complexe. On va d'abord chercher à trouver ce que l'on appelle des "motion blocks", des blocs de mouvement. Ces blocs

sont en fait des blocs que l'on trouve dans la frame précédente et qui ont simplement bougé. Cela correspond a l'idée que la frame suivante sera en grande partie similaire au frame existante, et que de nombreux morceaux n'auront que bougés. Cette idée permet de gagner largement en compression puisque chaque bloc trouvé par compensation de mouvement n'aura pas besoin d'être entièrement recodé, il suffira de stocker son vecteur. On cherche donc un maximum de ces blocs pour pouvoir réduire la quantité d'information nécessaire. L'information qui ne peux être trouvée par compensation de mouvement sera codée traditionnellement ; comme dans une I frame, par un codage DCT. La P frame est donc une mosaïque de blocs composés d'un vecteur (pour indiquer ou se situait le bloc dans la frame précédente) et de blocs complets. En pratique, ces vecteurs, appelé "motion vectors", sont recherché sur une taille fixée de 16x16 pixel (c'est-à-dire 4 blocs) (en dessous, le vecteur aura une efficacité bien inferieure a celle d'un codage complet). [4]

Les B frames poussent le concept des P frames plus loin ; alors qu'une P frame ne se base que sur la frame précédente, la B frame utilise également des frames "dans le future" pour faire de la compensation de mouvement. Cela permet encore un nouveau gain(en particulier lorsque la compression est forte) mais demande à pouvoir décoder des frames à l'avance. [4]

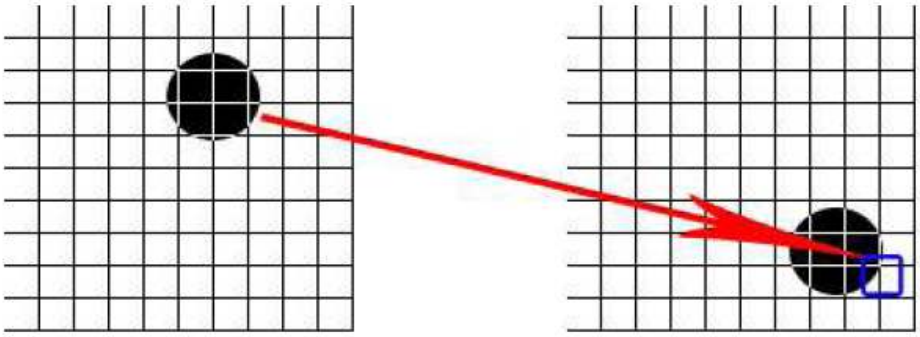


Figure 2.8 : Vecteur de compensation de mouvement dans une P Frame

c L'ordre

Comme les B Frames demandent à connaître à l'avance d'autres frames, on les stocke dans un ordre différent de l'ordre d'affichage, comme indiqué sur la figure 2.9. [4]

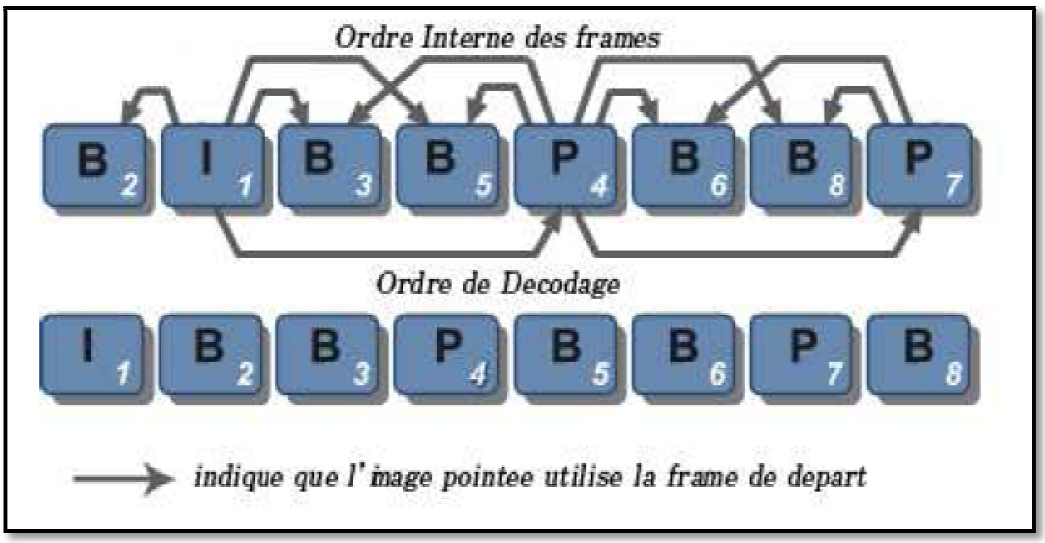


Figure 2.9 : Référencements des frames entres elle dans un flux video MPEG

d Estimation du mouvement

Pour calculer la compensation de mouvement, on précalcule l'estimation du mouvement. Cette technique fonctionne en pré-calculant les vecteurs de mouvement. Pour cela on prend un bloc et on le compare dans un rayon donné à tout les blocs aux alentours dans la frame suivante. Si il correspond a un bloc alors on calcule le vecteur qui lui correspond. Ce vecteur permettra ensuite de déterminer la P frame/la B frame suivante, bien plus aisément. [4]

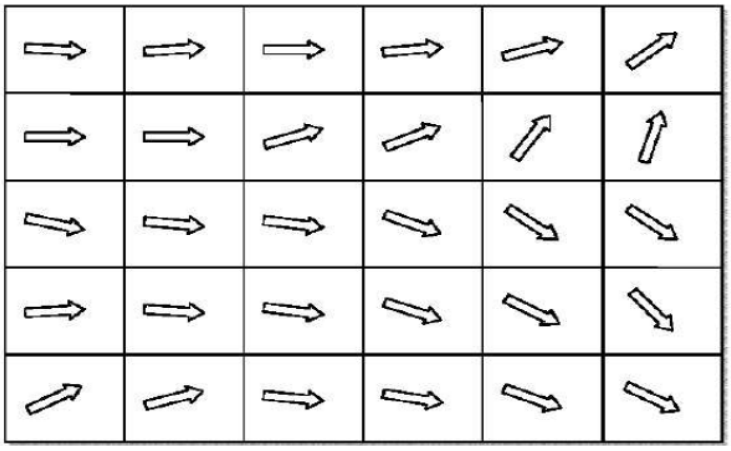


Figure 2.10 : Champs de vecteur de compensation

Conclusion

Nous aurions aime choisir le format AVI qui est pratique le plus répondu ; mais le manque sévère de documentation nous a oblige à opter pour le format MPEG.

Nous avons donc vu la diversité des formats vidéo en étudiant les plus répandus (AVI, MPEG, MOV, H263..) et en constatant leurs différences.

Filtrage d'une séquence vidéo par Co-simulation

Introduction

Le but du projet est la réalisation d'un système de traitement en temps réel qui fait un filtrage d'une séquence vidéo. Le filtre adapté est le filtre médian.

Ce travail est implémenté avec l'outil de Xilinx System Generator, cette dernière combine les fonctionnalités de MathWorks (simulink) et celles du langage VHDL.

Le présent chapitre présente dans un premier temps l'outil de développement System Generator, puis seront décrites les unités qui composent notre système de traitement. Nous présenterons également les résultats obtenus.

3.1. Logiciel Xilinx ISE 12.3

L'offre logicielle dans le domaine de conception des circuits numériques est très variée et l'un de ces environnements que nous allons exploiter au cours de ce travail est Xilinx ISE qui est un logiciel de création et de gestion de projet pour les circuits FPGA. [3]

C'est un logiciel multi taches qui possède dans son soft différents outils permettant la création de système sous circuits numériques, l'introduction de projets de manière textuelle ou graphique en vue d'une intégration dans un circuit logique programmable.

Ce logiciel Xilinx ISE permet la simulation de la description et la synthèse du circuit logique équivalent puis le placement et le routage du circuit sur un prototype correspondant a une technologie FPGA bien précise et enfin lorsque toutes les vérification sont faite vient l'implantation sur un FPGA réel ce qui correspond à générer le fichier de configuration du circuit cible choisi afin d'établir les interconnections des cellules logiques correspondantes au circuit logique conçu avec optimisation de ressources disponibles au niveau du circuit programmable FPGA. [3]

D’une manière générale, le Xilinx ISE permet de réaliser toutes les étapes de conception et de programmation des circuits FPGA de Xilinx et même pour d’autres circuits programmables tel que les CPLD. [3]

La conception de circuits sur Xilinx ISE met en œuvre 4 outils :

- **Un éditeur de texte ou entrée graphique :** permet la description dans les logiciels CAO c'est-à-dire de dessiner ou de décrire le circuit avec une interface graphique ou textuelle. [3]
- **Un simulateur :** la simulation du système est faite pour vérifier la validité du code avant-synthèse, après synthèse et placement –routage. [3]
- **Un synthétiseur** : L’étape de synthèse et routage succéderons par la suite où la synthèse consiste à faire la transcription de la description d’une forme texte vers une graphique (RTL) à base de portes logiques. [3]
- **Un placeur routeur** : Cette étape est une adaptation du circuit logique synthétisé sur les ressources disponibles dans le circuit FPGA ciblé. La figure 3.2 montre les différentes étapes pour réaliser un circuit sur FPGA. [3].

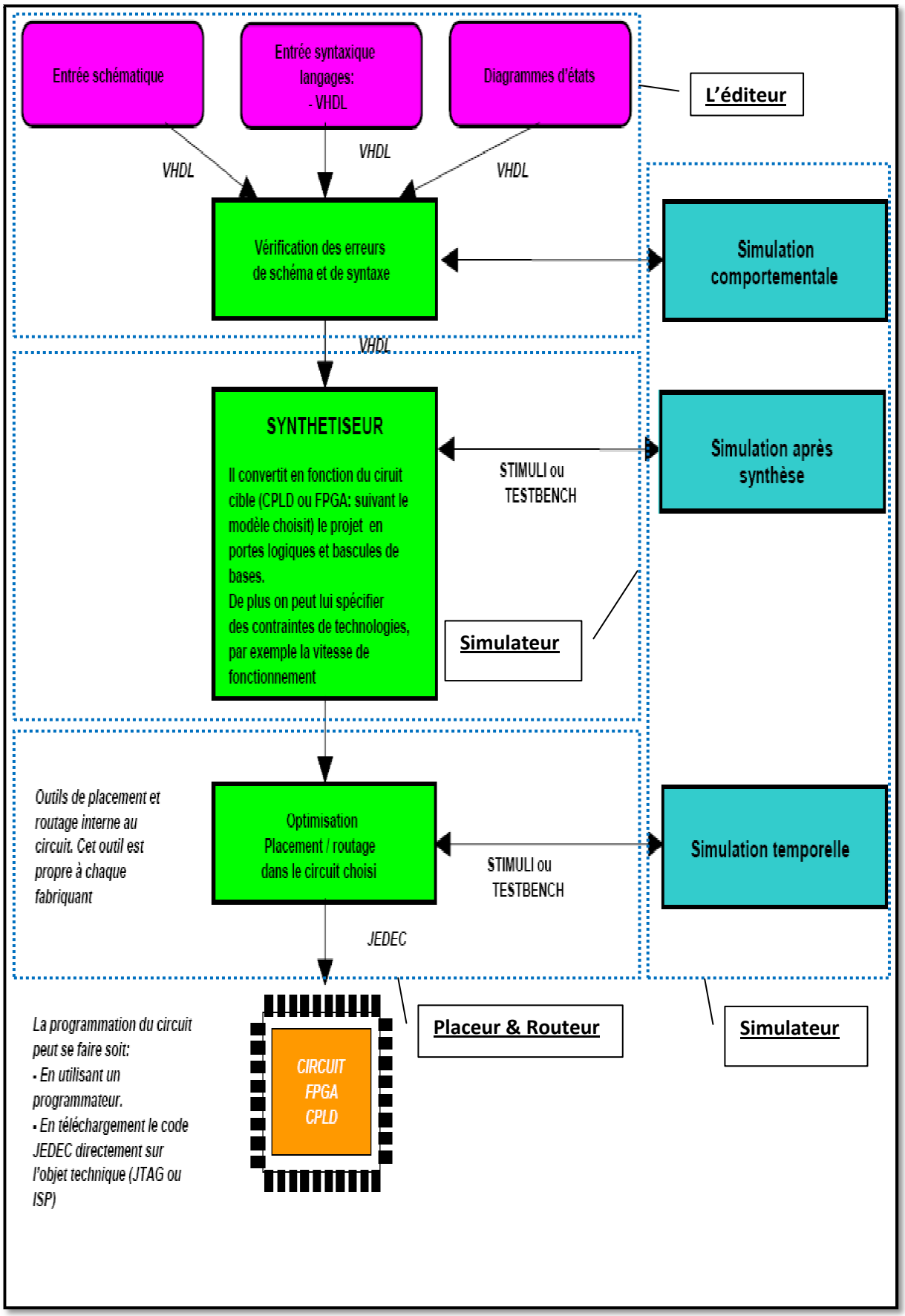


***Figure 3.2*:** différents étapes pour la réalisation d'un circuit par FPGA.

3.2. Le logiciel Xilinx System Generator

System Generator est un outil de conception DSP de Xilinx qui permet l'utilisation de Math Works basé sur des modèles Simulink pour la conception FPGA.

Il est utilisé dans ce projet dans le but de combiner les deux logiciels MathWorks basé sur Simulink et Xilinx (VHDL) pour une conception FPGA.

En effet dans ce projet il y a des parties qui sont développées sous Simulink et d'autres sous VHDL comme il sera décrit dans le schéma global du traitement par filtrage appliqué à une séquence vidéo.

Les schémas sont capturés dans la modélisation de l'environnement DSP Simulink à l'aide de blocs spécifiques de Xilinx. Toutes les étapes en aval de la mise en œuvre FPGA y compris la synthèse sont automatiquement effectuées en vue de créer un fichier de programmation.

Plus de 90 blocs de construction DSP sont fournis dans le jeu de blocs DSP de Xilinx pour Simulink [Figure 3.3]. Ces blocs comprennent les blocs de construction commune DSP tels que des additionneurs, des multiplicateurs et des registres. D'autres blocs sont également inclus tels que les blocs FFT, filtres, etc...

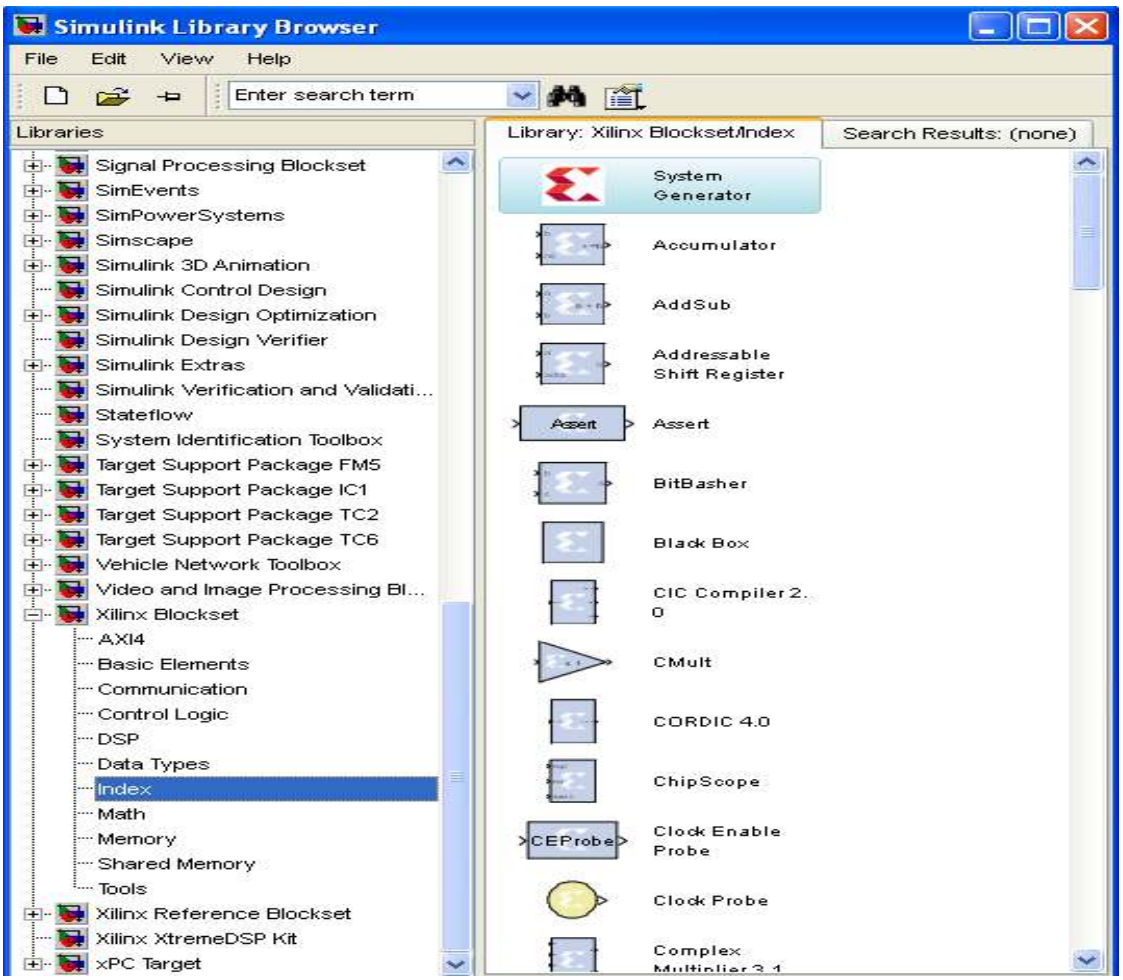


Figure 3.3 **:** Les blocs de Xilinx pour Simulink.

3.2.1. Définition du bloc system Generator

Le bloc System Generator permet de contrôler les paramètres du système et la simulation, et est utilisé pour appeler le générateur de code. Tous les modèles Simulink contenant les blocs de Xilinx doivent contenir au moins un bloc System Generator. Une fois ce dernier ajouté à un modèle, il est possible de spécifier la manière dont la génération de code et de simulation doivent être manipulée [Figure 3.4]. [5]

Figure 3.4: Le bloc system generator

3.2.2. Les options de bloc System Generator:

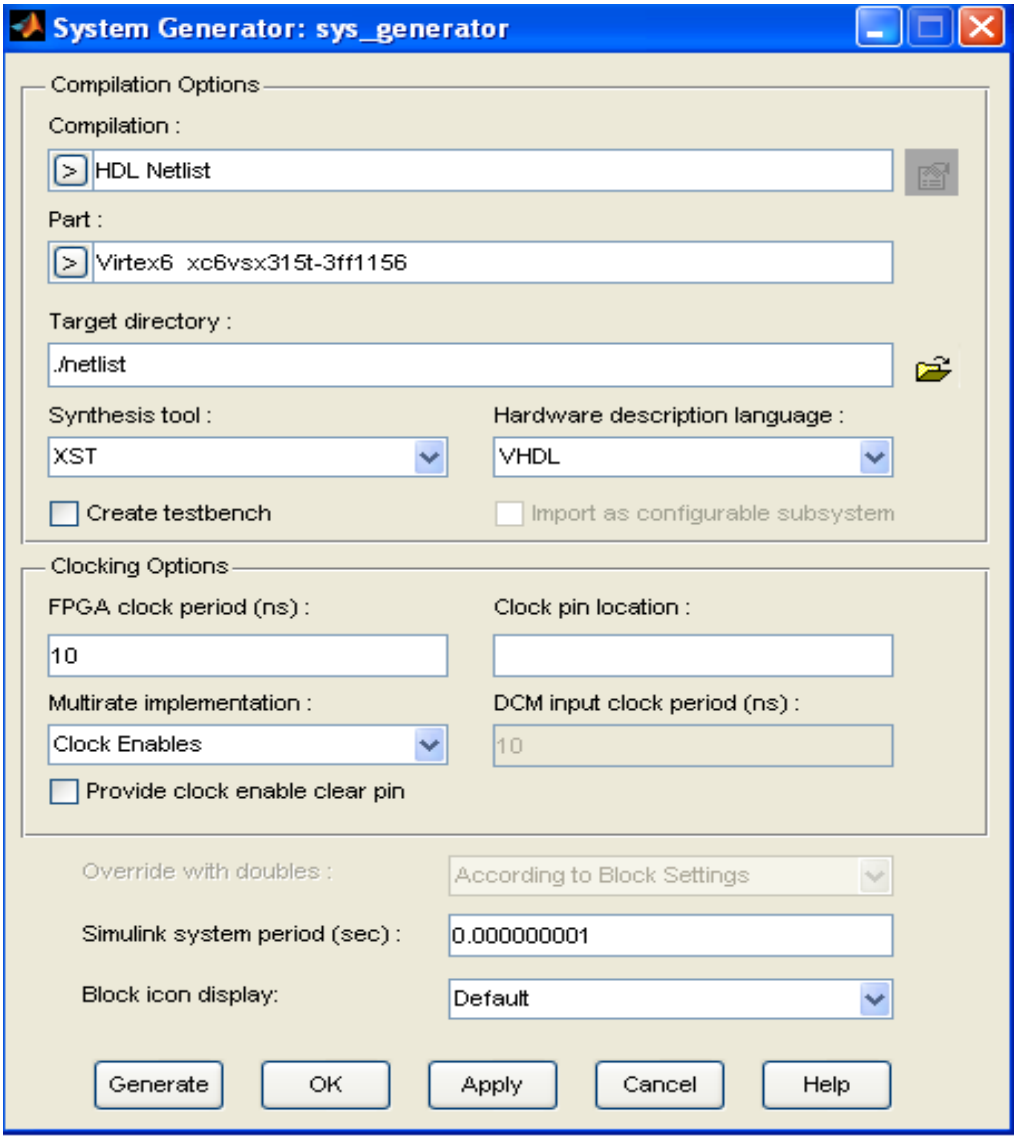


Figure3.5 **:** options de bloc System Generator

- ***Compilation :*** Indique le type de résultat de compilation qui devrait être produite lorsque générateur de code est invoqué. [5]
- ***part :*** Définit le type de FPGA utilisé. [5]

- ***Target directory :*** Définit l'emplacement où System Generator devrait écrire les résultats de la compilation. [5]

Comme le système générateur et les outils FPGA de conception physique génèrent de nombreux fichiers, il est préférable de créer un répertoire cible distinct, à savoir, un répertoire autre Que le répertoire contenant nos fichiers de modèles Simulink. [5]

- ***. Synthésis tool :*** Indique l'outil à utiliser pour synthétiser la conception. Les Possibilités sont Synplicity Synplify Pro, Synplify, et XST de Xilinx. [5]
- ***. Hardware Description Language :*** Indique le langage HDL utilisé pour la compilation de la Conception. Les possibilités sont VHDL et Verilog. [5]
- ***. Mutirate implémentation*** *:* Cette fonction de System Generator permet de créer un banc de VHDL. Simuler le banc de test dans un simulateur HDL permet de comparer les résultats de simulation Simulink avec ceux obtenus à partir de la version compilée de la conception. Pour construire des vecteurs de test, System Generator simule la conception dans Simulink, et enregistre les valeurs observées au niveau des passerelles. Le premier fichier HDL pour le banc de test est nommé testbench.vhd qui est un nom dérivé de la partie de la conception à l'essai. [5]

Note:

Cette option n'est pas prise en charge lorsque des blocs de mémoire partagée sont inclus dans la conception.

- **Période d'horloge FPGA (ns):** Définit la période en nanosecondes de l'horloge matérielle. La valeur ne doit pas être un nombre entier. [5]
- ***L'emplacement broche Clock :*** Définit l'emplacement des broches pour l'horloge matérielle. Cette information est transmise à la mise en oeuvre des outils Xilinx vers un fichier de contraintes. Cette option ne devrait pas être spécifiée si la conception du système générateur doit être incluse dans le cadre d'un grand projet de conception HDL. [5]
- ***. Générateur d'horloge (DCM) :***Crée une enveloppe d'horloge avec un DCM qui peut piloter Jusqu'à trois ports d'horloge à des taux différents pour Virtex-4 et Virtex-5 et jusqu'à deux ports d'horloge pour Spartan-3A. [5]

- ***. DMC entrée période de l'horloge (ns):*** Préciser si différente de la période d'horloge FPGA (ns) option.[5]

Autres Options :

- ***. période système Simulink (s):*** Définit la période dusystème Simulink en unités de Secondes. La période du système Simulink est le plus grand diviseur commun des périodes de l'échantillon qui apparaissent dans le modèle. Ces périodes d'échantillonnage sont définies explicitement dans les boîtes de dialogue de bloc, héritées selon les règles de propagation Simulink, Dans le matériel, un bloc ayant un taux d'échantillonnage de plus d'un processus de ses intrants a un rythme plus rapide que les données. Par exemple, un bloc séquentiel multiplicateur avec un taux de sur-échantillonnage de huit implique une période d'échantillonnage (Simulink) qui est un huitième du temps du bloc multiplicateur de réelle de l'échantillon dans Simulink. Ce paramètre peut être modifié que par un bloc maître. [5]

3.2.3 HDL Black Box

« HDL Black-Box » est un bloc qui représente dans l'environnement « Simulink HDL » une unité qui sera Co-simulé à l'aide de Simulink et Xilinx System Generator

Pour faire une Co-simulation VHDL dans Simulink, son interface doit être créée et décrite dans un certain format qui permet le flux de données entre les simulateurs (unités crées par simulink et celles crées par System Generator).

On fait appel à HDL black box lorsqu'on fait implémentation sur FPGA d'un algorithme dont le temps d'exécution de calcul est très important sur une machine à base d'un CPU core (i3 par exemple)

Lorsque cette situation se présente, on développe une description VHDL (très rapide) qui sera « mise en boite » par l'outil System Generator sous forme de système qui admettra des entrées et des sorties.

Pour faire entrer la description VHDL dans le HDL black box, il suffit d'ajouter le composant black box à partir de la bibliothèque « blocs Xilinx System Generator » (Figure 3.6) dans une nouvelle fenêtre, puis l'outil System Generator va déplacer le script en VHDL dans le HDL black box à partir de son emplacement.

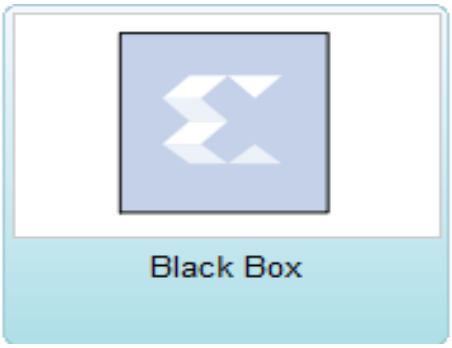


***Figure 3.6*:** bloc black box

Une fois le HDL black box chargé par le code VHDL on aura un nouveau bloc avec des entrées/sorties selon notre description prêt à l'utilisation. La figure 3.7 montre le bloc HDL black box.

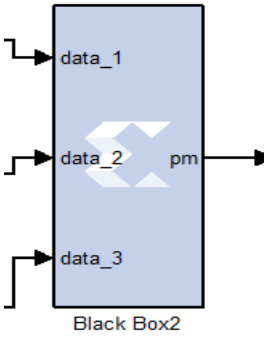


***Figure 3.7*:** bloc HDL black box chargé par le code VHDL

Il est noté que le HDl black box doit contenir une entrée clock enable cette dernière permet l'activation de l'horloge (synchronisation).

3.3 Schéma global du traitement par filtrage d'une séquence vidéo

L'objectif du notre projet est le développement d'un système qui fait le traitement de filtrage en temps réel d'une séquence vidéo. Le schéma synoptique de ce dernier est donné par la figure 3.8

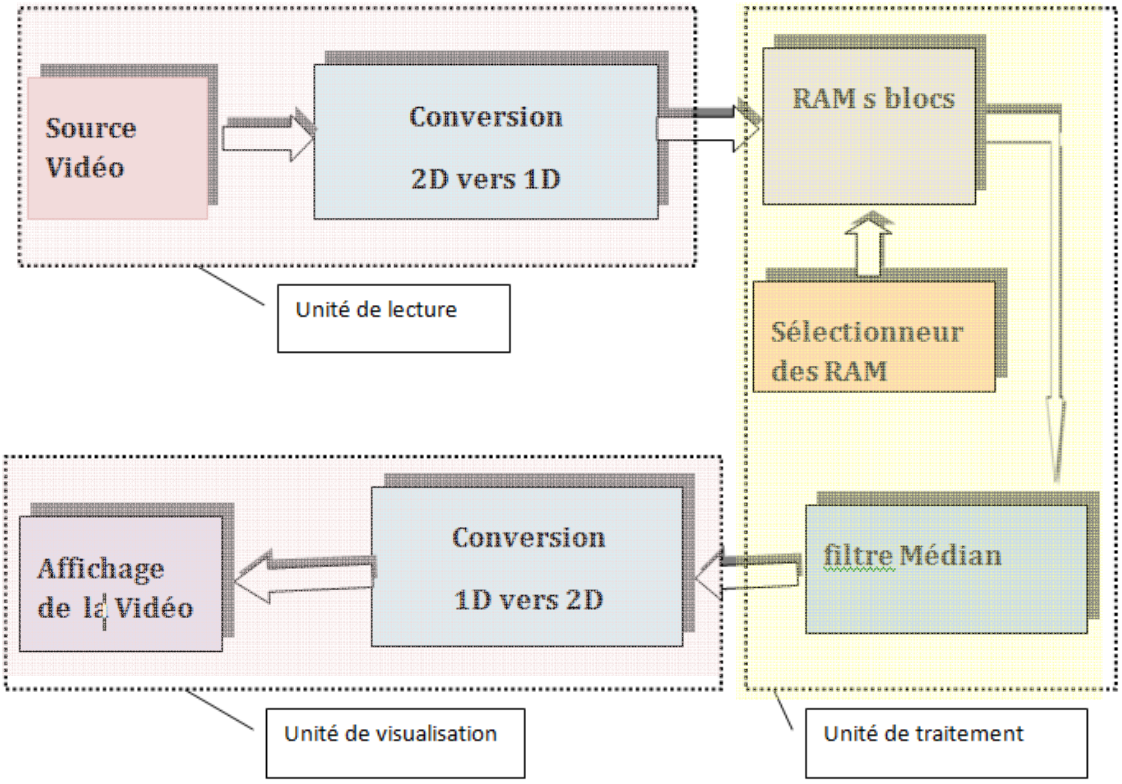


Figure 3.8 : Schéma synoptique pour les blocs du projet

Le système est composé de trois étages dont deux sont développées avec simulink. Ce sont :

- L'unité de lecture de la séquence vidéo
- L'unité de visualisation après le traitement, figure 3.8

Et le troisième qui constitue de l'unité de traitement, cette dernière est développée en VHDL. Les paragraphes qui suivent vont décrire chaque étage

3.3.1 Description d'unité de lecture

Cette unité est réalisée dans l'environnement MathWorks .

A partir de la figure 3.9, on distingue les éléments suivants :

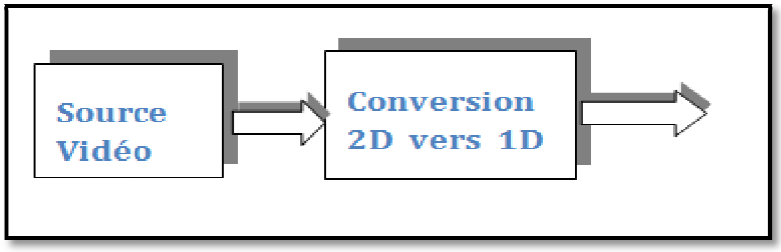


Figure 3.9 : unité de lecture

a Source Vidéo

La source vidéo peut être une camera ou une webcam ou un fichier vidéo enregistré dans un PC, mais pour traiter ce fichier il faut éliminer l'entête ainsi la compression de ce fichier pour obtenir seulement l'information.qu'elle doit etre sous un fichier binaire

Le bloc simulink de matlab «from multimedia file » Figure (3.10), nous a offre ces conditions.

Figure 3.10: bloc from multimedia file

Il peut manipuler avec des fichiers multimédia sous Windows ou sous des plateformes non-Windows :

- Sous Windows, il lit les images vidéo et / ou des extraits audio d'un fichier multimédia compressé ou non compressé. Les fichiers multimédias peuvent contenir audio, vidéo ou données audio et vidéo.
- Les plateformes non-Windows, il lit les images vidéo et / ou des échantillons audio d'un fichier AVI non compressé.

Les paramètres du bloc "from multimédia file "sont réglés selon la figure 3.11.

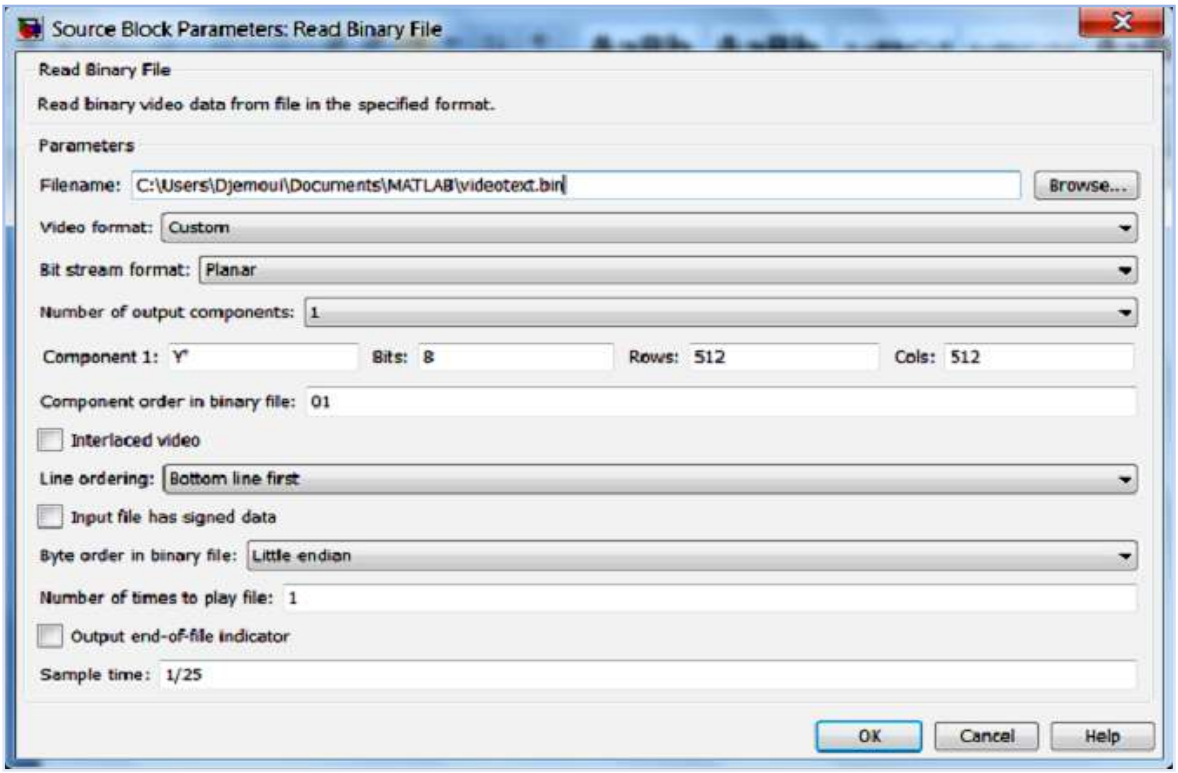


Figure 3.11: les paramètres du bloc "from multimedia file ''

b *Conversion 2D vers 1D*

La sortie de la source est procurée sous forme d'une suite de matrices (images matricielles), or l'étage de traitement utilise les données sous forme vectorielle, d'où la nécessité défaire une conversion 2D vers 1D, la figure 3.12.présente le black box correspondant

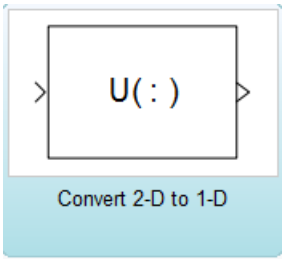
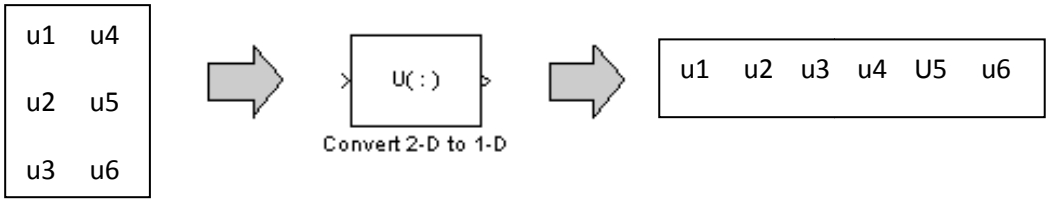


Figure 3.12: bloc de conversion de 2d vers 1d.

Exemple : Conversion 2D vers 1D.



Les paramètres du bloc “convert 2-d to 1-d” sont donnés automatiquement par simulink comme le montre la figure 3.13

Figure 3.13 : les paramètres du bloc “ convert 2-D to 1-D ”

3.3.2. Description d’unité de visualisation

Cette unité est réalisée dans l’environnement MathWorks.

A partir de la figure 3.14, on distingue les éléments suivants :….

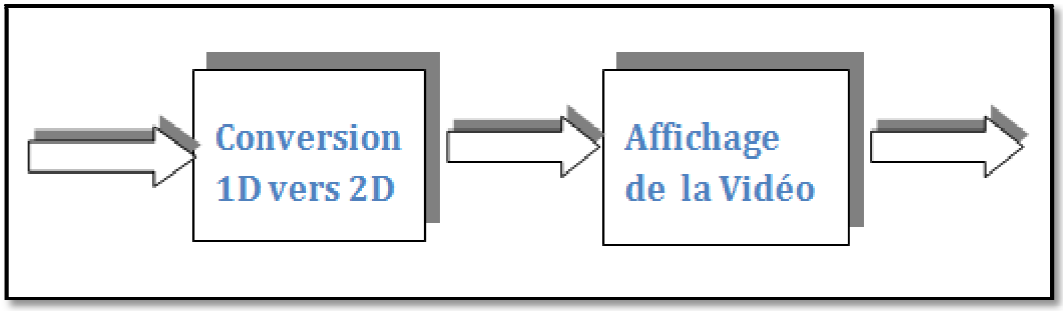


Figure 3.14 : unité de lecture

a Conversion 1D vers 2D

Pour afficher notre vidéo traitée on a obligé du faire la conversion inverse (1D vers 2D), cette conversion est effectuer par le bloc matlab Reshape.

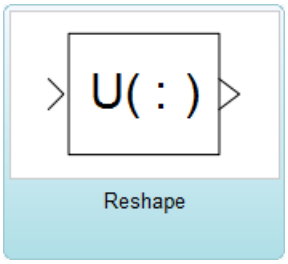
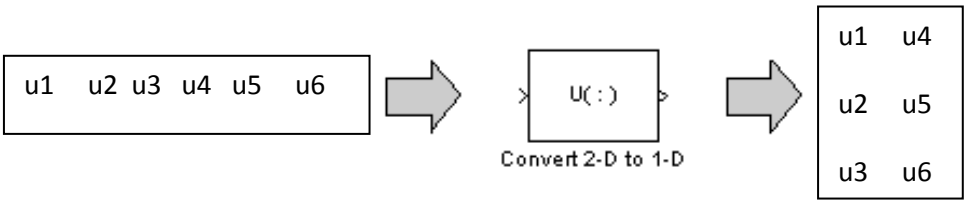


Figure 3.14 : le bloc Reshape de conversion 1D vers 2D)

Exemple :



3.3.3. L'unité de traitement :

Le traitement effectué dans ce travail est le filtrage médian, en VHDL cette opération à nécessité l'utilisation de :

- RAM blocs (présentées dans le chapitre 1)
- un étage de sélection de RAMs
- étage de filtrage (figure 3.12)

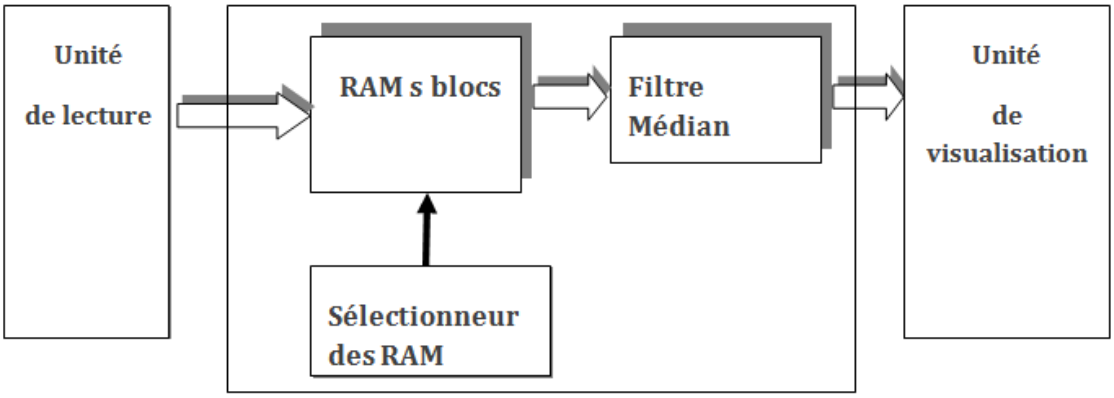


Figure 3.12 : Unité de traitement

a RAMs Blocs

Dans le chapitre 1, nous avons vu que pour réaliser le filtrage median il faut stocker a chaque fois 4 nouvelles lignes image. Ce dernier (le stockage), est orchestré par la génération des signaux de contrôle des RAMs

- **Génération des signaux de contrôle des RAM's**

Les filtres adaptés pour notre traitement utilisent des masques de fenêtre 3x3, donc pour le calcul de la valeur médian on a besoin de stocker trois (03) lignes pour le balayage horizontal du masque.

Il est nécessaire de prendre une quatrième ligne pour considérer le déplacement vertical du masque.

L'alternance de la sélection des mémoires en lecture/écriture, est faite de telle sorte, que l'on ait trois signaux sélectionnés en mode lecture et un signal de sélection en mode écriture.

Le contrôle des signaux de sélection nécessite un système séquentiel qui est composé d'un compteur modulo 2048 et d'une circuiterie (comparateur) qui procure une sortie sur deux bits, figure (3.13).

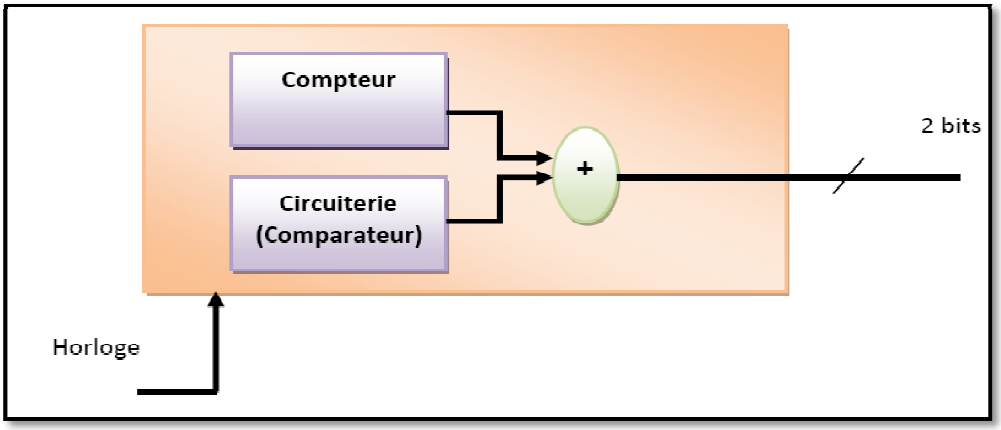


Figure3.13 **:** circuits de génération des signaux du contrôle

Ce système fonctionne à la cadence de l'horloge de FPGA, il offre en sortie tous les 512 tops d'horloge un code sur 2 bits qui contrôle la sélection des RAMs ; ces bits sont dits bits de contrôle ou d'état (état1, état2, état3 et état4).

L'organigramme suivant (figure 3.14) montre le séquencement de génération des signaux de contrôle des RAMs.

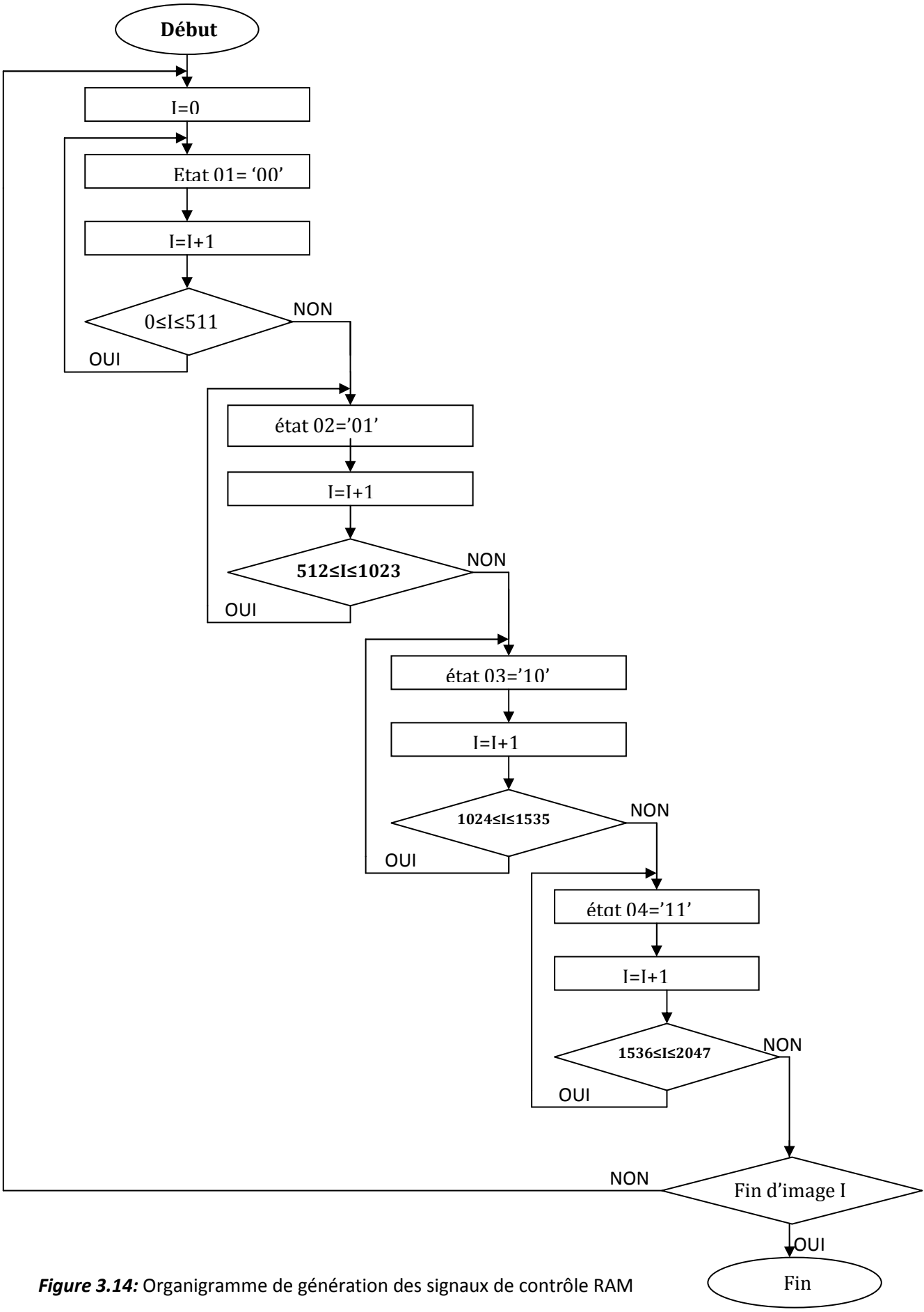


Figure 3.14: Organigramme de génération des signaux de contrôle RAM

b Sélection des RAMs

Les 4 états générés, permettent d'obtenir des signaux de sélection des RAMs. De chaque état, on génère 4 signaux qui définissent le mode de sélection des RAMs. Chaque état, correspond à trois (03) sélections en mode lecture et une (01) en mode écriture.

Les modes de sélection des RAM's sont donnés par :

we_1, we_2, we_3, et we_4.

La sélection est faite comme le montre la table de vérité suivante :

états	We_1	We_2	We_3	We_4
00	0	0	0	1
01	1	0	0	0
10	0	1	0	0
11	0	0	1	0

Tableau 3.1 : Table de vérité de mode de sélection des RAM's

On prend la sélection par :

- We_x='0' en mode lecture
- We_x='1' en mode écriture

L'organigramme suivant (figure 3.15) permet de voir les modes des sélections des RAMs en lecture/écriture.

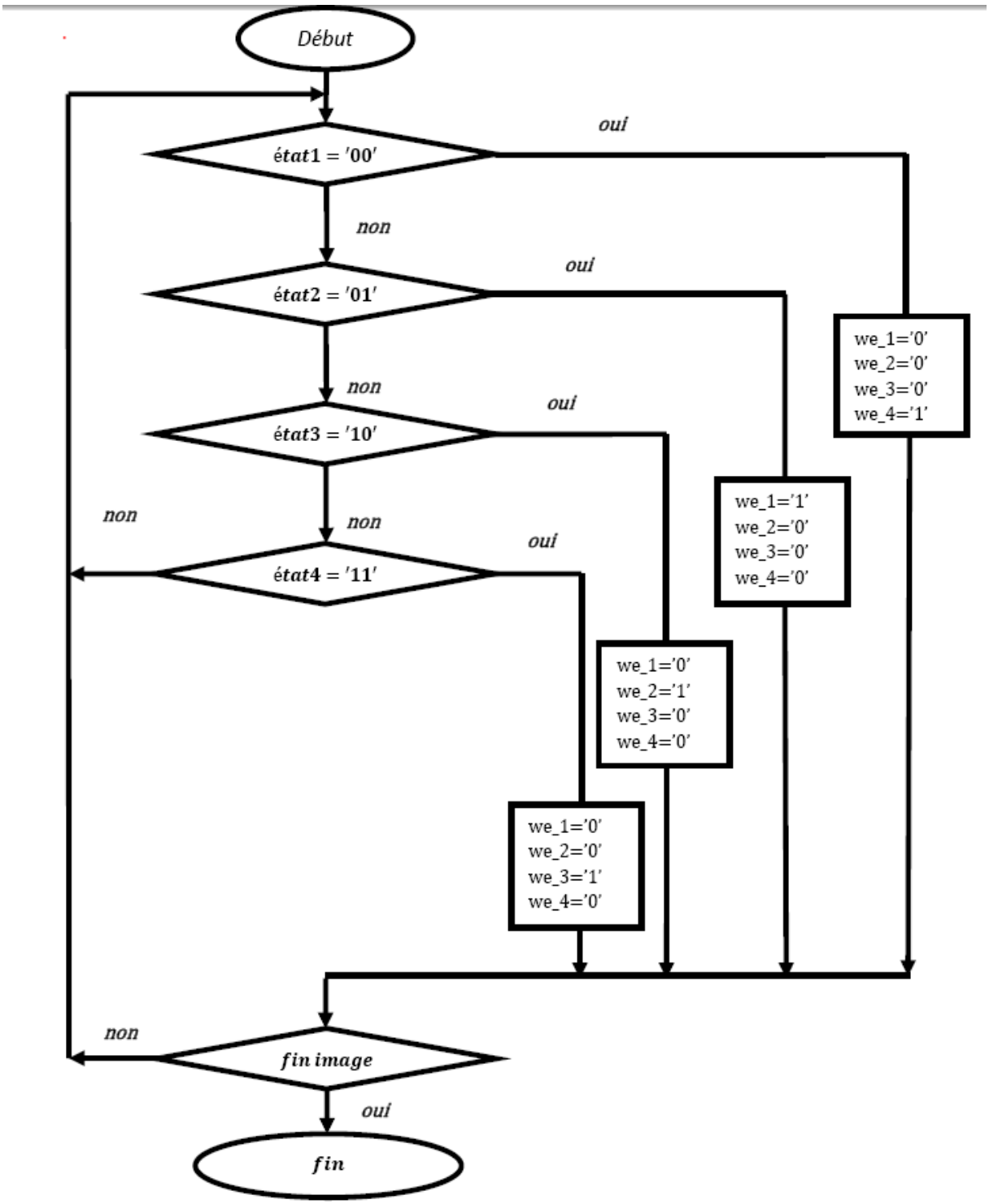


Figure 3.15 **:** Organigramme du mode de sélection RAM

Le schéma de fonctionnement correspond à cette opération est donnée par la figure 3.16.

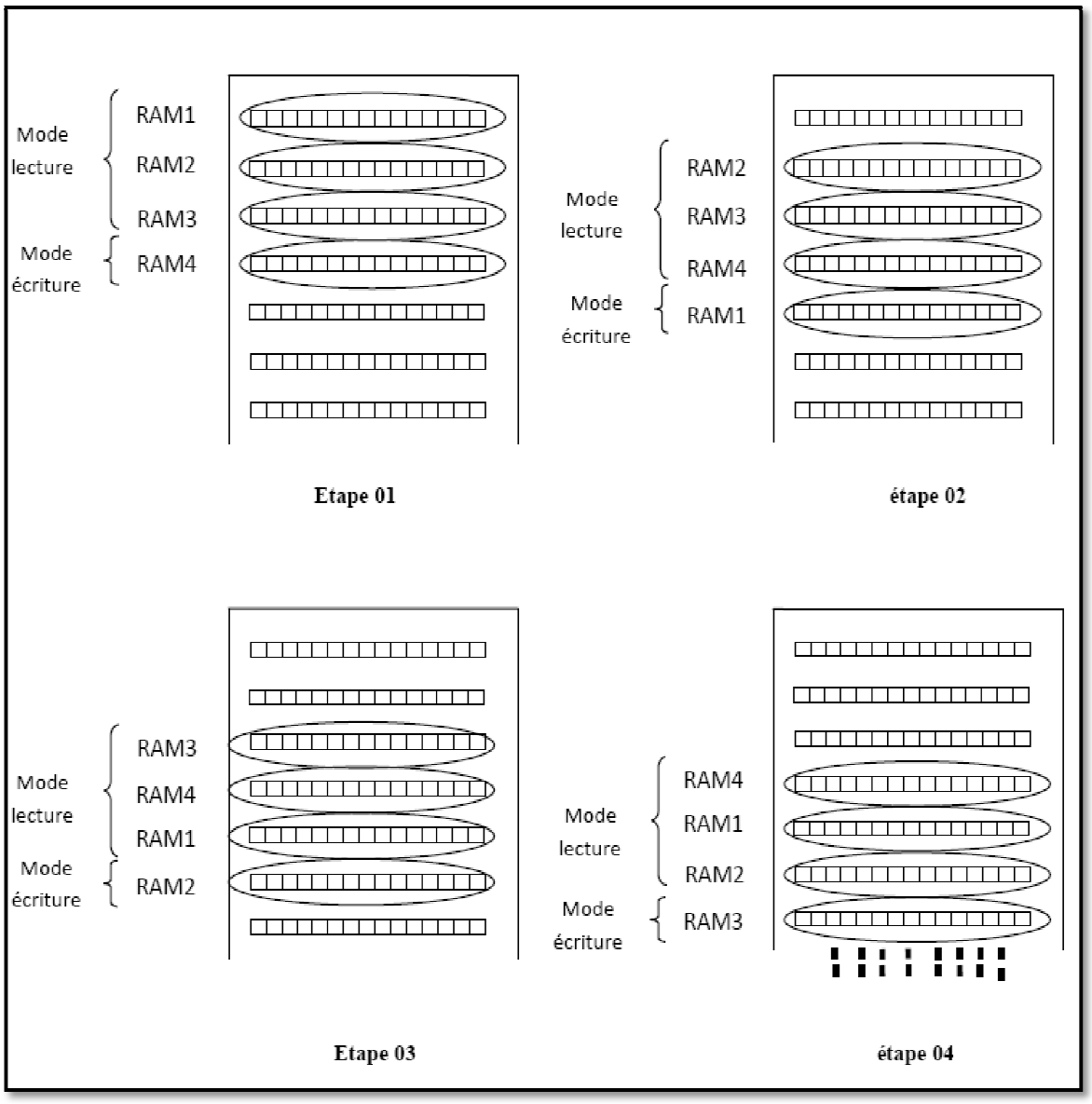


Figure 3.16 : Schéma fonctionnelle de sélection des RAM's.

3.4 Résultats

34.1 Signaux de sélection des RAMs blocs

La *figure 3.17,* représente les quatre signaux de sélection RAM's, ces signaux sont donnés par we_1, we_2, we_3, we_4

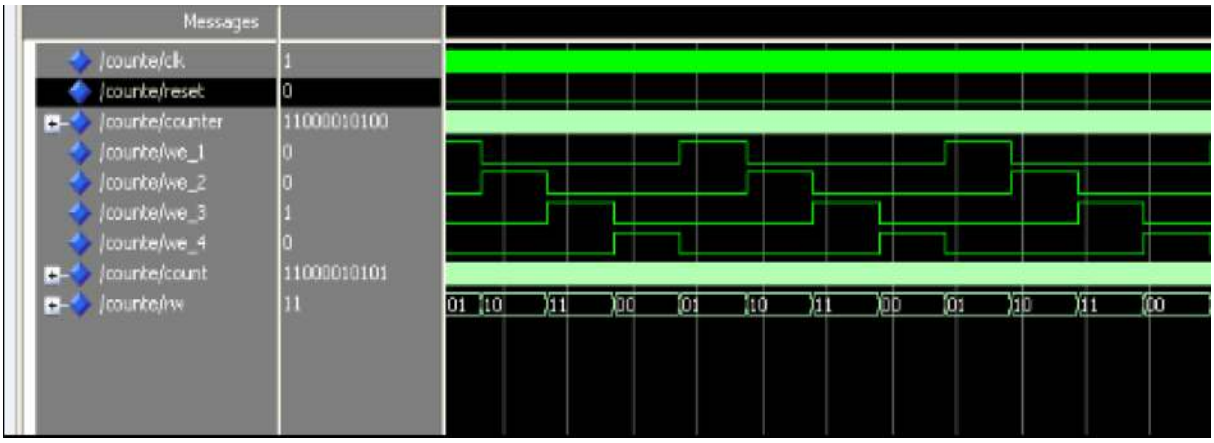


Figure 3.17: Signaux de sélection des RAMs

3.4.2 Ecriture des données sur la RAM

Un module d'adressage est nécessaire pour chaque RAMs, ce dernier est réalisé par un compteur synchrone modulo 512, l'incrémentation du compteur se fait par pas de 1, ce dernier permet d'écrire les données sur la RAM, ces données mémorisées correspondent à une ligne de l'image par RAM. Les résultats obtenus sont donnés par la figure 3.18

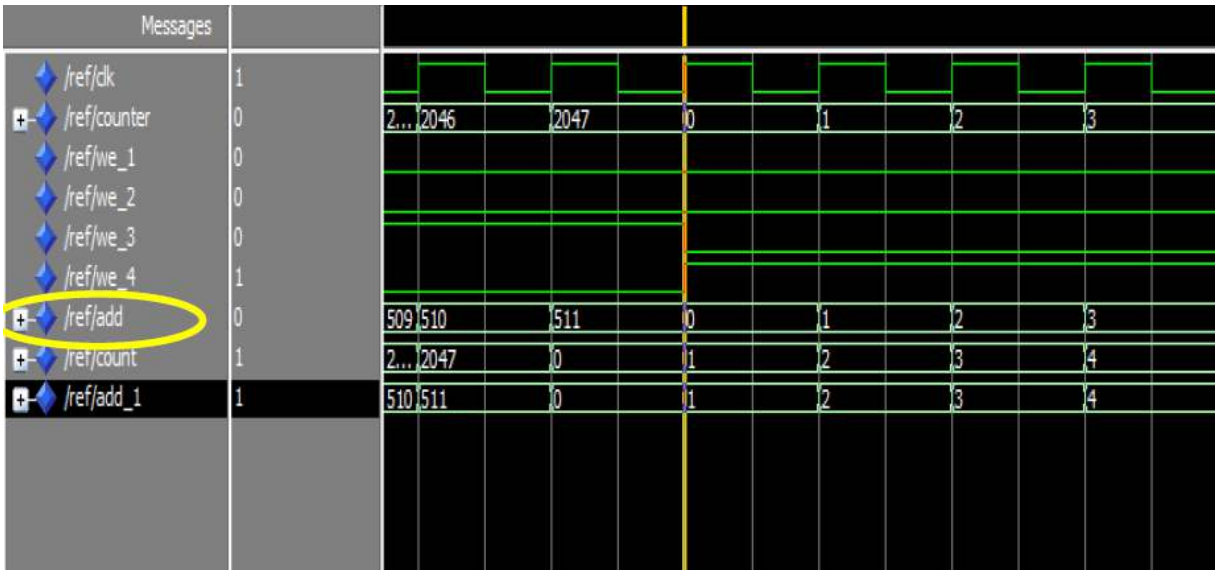


Figure 3.18 : Signaux d'adressage des RAMs blocs

La sortie du module d'adressage est mentionnée dans le chronogramme par « add ».

3.4.3 Le filtre médian

Le principe du filtrage d'une vidéo (25 images par seconde) est très similaire à un filtrage d'une image le principe consiste à balayer chaque séquence d'image par le filtre (le principe est cité dans le 1er chapitre), donc pour rester dans les normes d'un filtrage en temps réel on doit balayer 25 images par seconde.

Les entrées-sorties du filtre médian sont illustrées dans la figure (3.19)

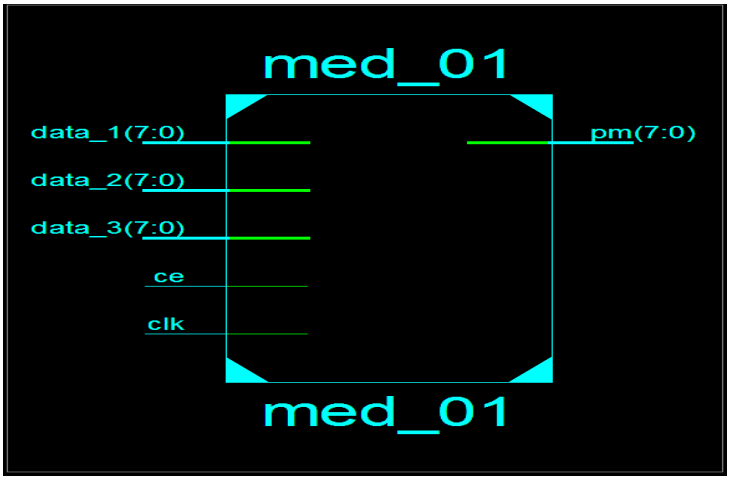


Figure 3.19 : La vue schématique de filtre médian

Les resultats obtenus sous VHDL du filtre médian sont donnés par la figure3.20

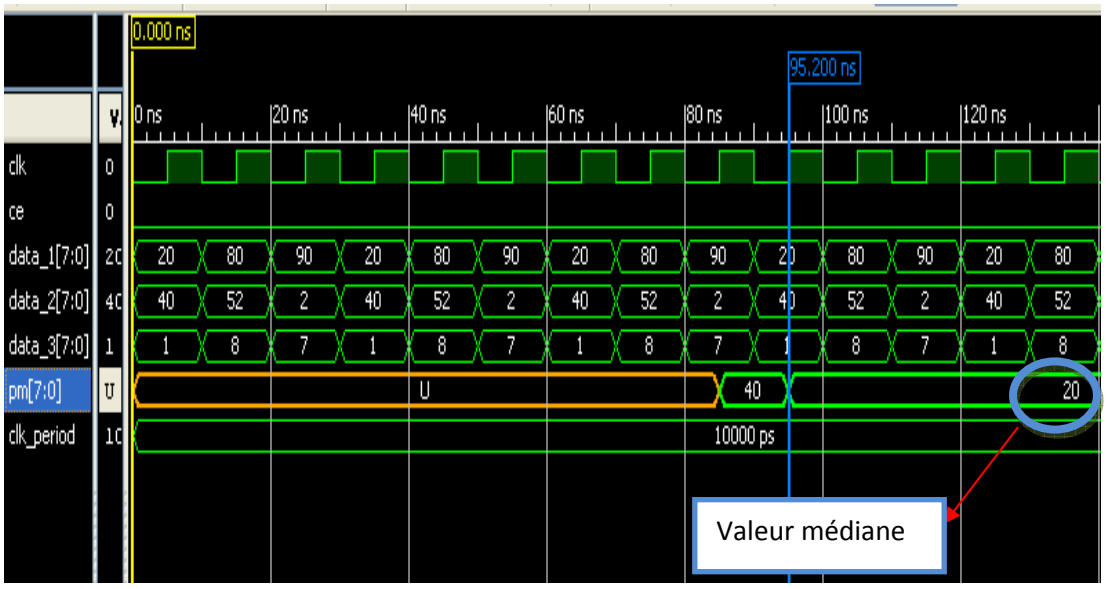


Figure 3.20 : La simulation numérique de filtre médian

On remarque que les résultats de la simulation prennent des résultats réels qu'après 9 tops d'horloges, ce retard est causé par le chargement des RAMs blocs.

3.4.4 Module de traitement vidéo Co-simulé par (simulink/System Generator)

Nous avons développé un système Co-simulé en utilisant l'outil simulink et System Generator qui réalise le filtrage médian d'une séquence vidéo. Le schéma de

Co-simulation est présentée ci-dessous

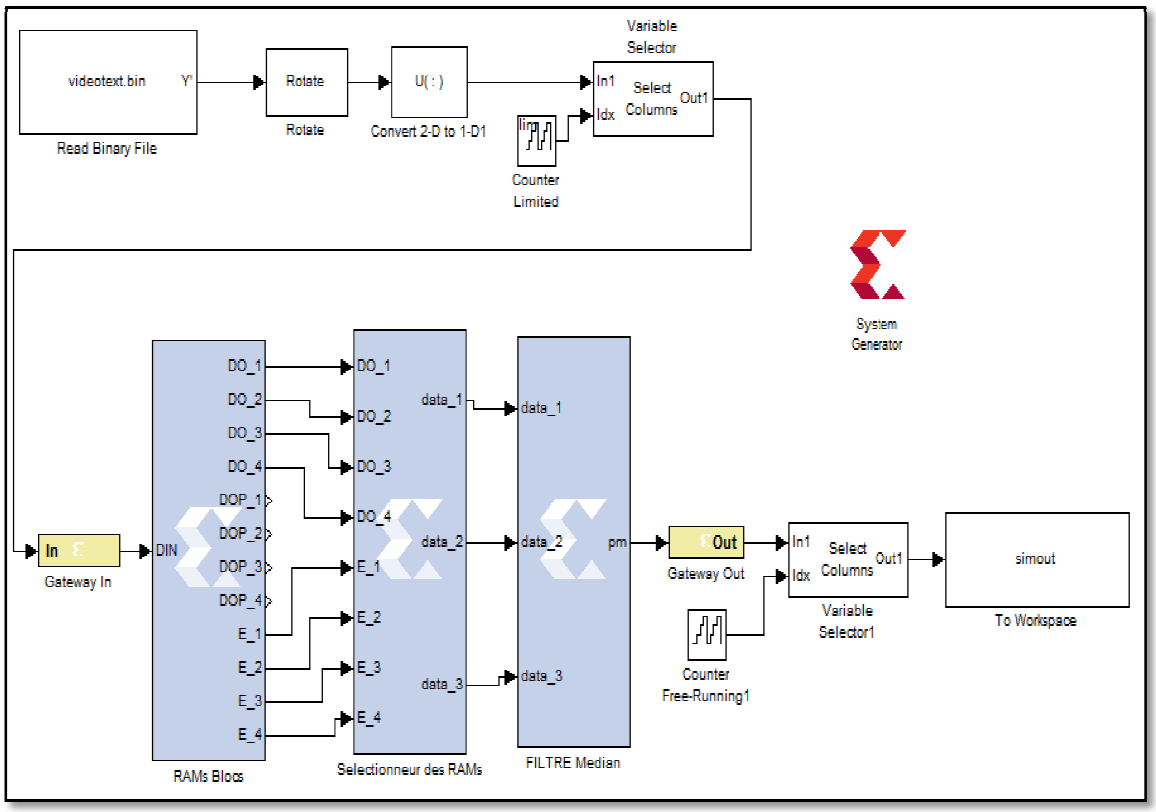


Figure 3.24 **:** module de traitement vidéo Co-simulé par (simulink/Système Generator)

Ce système Co-simulé a été appliqué sur une séquence vidéo de trois secondes, il nous a donné les résultats qu'on espérait puisque nous avons pus récupérer la même séquence filtrée .la figure 3.25 montre une image récupérer à partir de la vidéo traité.

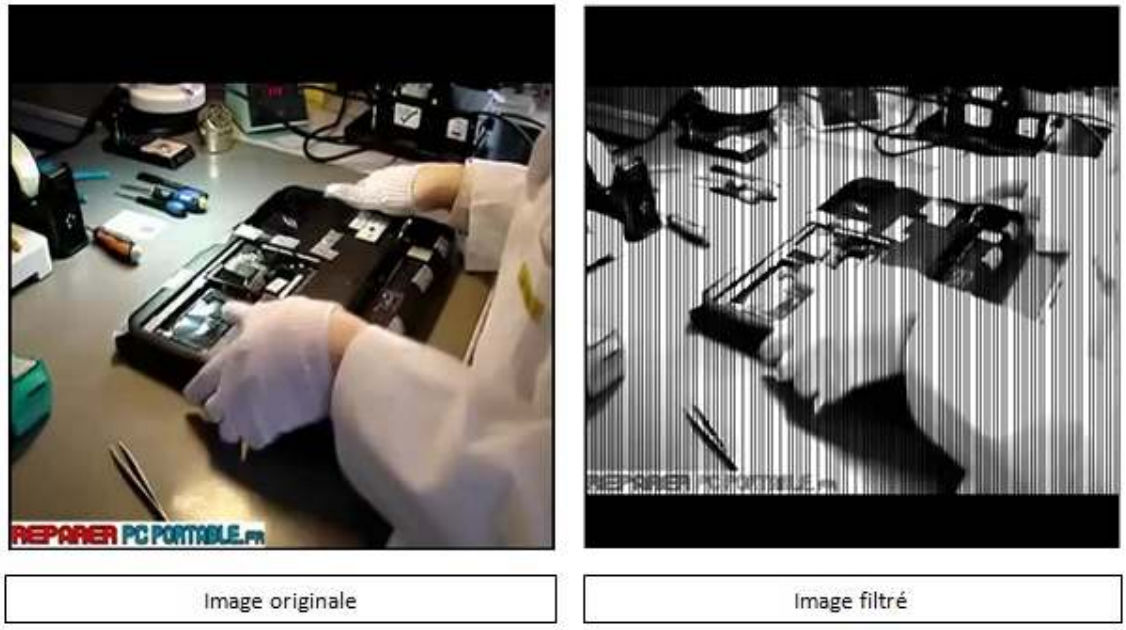


Figure 2.25 *:* image récupérer à partir de la vidéo traité

Il est à noter que le temps mis pour effectuer ce traitement est très important (de l'ordre de 48h) ceci est complètement normal vu que ce système de co-simulation n'est pas encore hard car il est implémenté sur un micro ordinateur .

Ceci ne devrait pas poser de problème lorsqu'on passera à la co-simulation hard i.e. lorsque la carte FPGA sera introduite comme le montre la figure 3.26

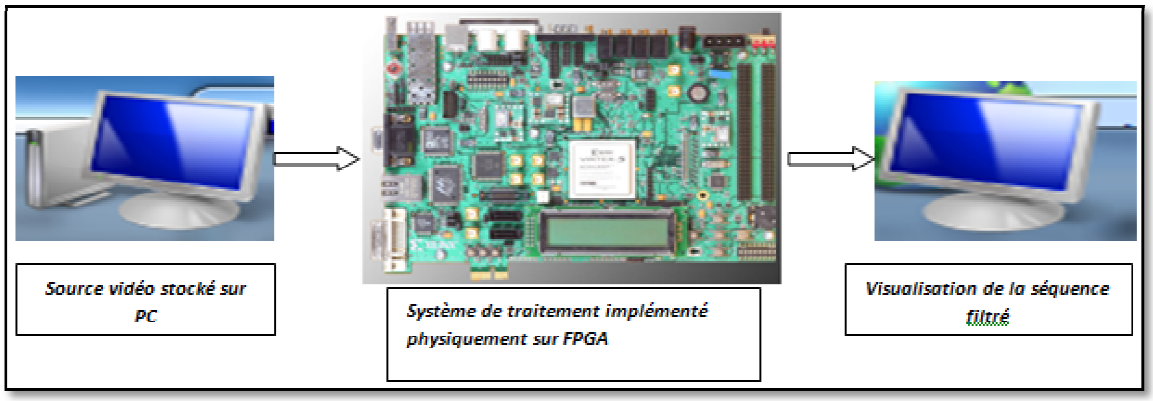


Figure 2.26 : les différentes unités de projet (lecture, filtrage, visualisation)

Conclusion

Dans ce chapitre nous avons expliqué les différentes parties développées, à savoir les étages de lecture et de visualisation et celui de traitement et nous avons également exposé les résultats obtenus à chaque étape de réalisation. Et nous avons déduit que la Co-simulation peut être très efficace pour régler pas mal de problèmes de génération de code et traitement en temps réel.

Conclusion générale

Dans ce travail nous avons réalisé un système de traitement vidéo basé sur le principe de la Co-simulation afin de rendre cette dernière (traitement vidéo) très rapide.

En effet le temps de calcul d'un filtre médian sur une séquence vidéo est très important si on utilise un processeur classique, la co-simulation hard permet de faire ce traitement en temps réel.

Le cahier de charge a été respecté puisque nous avons dans un premier temps pu filtrer une image avec ce système de traitement (Co-simulation) , ceci sans nous soucier de la qualité du filtrage, car ce n'était le but recherché dans ce travail, dans un second temps, nous avons également réussi le filtrage d'une séquence vidéo par Co-simulation dans cette partie on a remarqué que le filtrage a duré assez long temps car l'implantation n'est pas faite sur la carte FPGA.

Nous attendons les résultats de la Co-simulation hard (avec la carte FPGA) dans les jours qui vont venir en principe les traitements devront être en temps réel.

Ce projet a été d'un bénéfice et intérêt, il a permis d'enrichir nos connaissances dans le domaine de traitement d'image, néanmoins ce travail nous a permis d'apprendre l méthodologie de la conception des architecture de la Co-simulés a faite de MathWorks simulink et Xilinx system Generator.

On pense que ce travail ne devrait pas rester sans continuité, il peut effectivement constituer un point de départ pour les futurs diplômés intéressés au domaine de la Co-simulation.

Annexes

Exemple de numérisation d'une séquence vidéo

1720 pixels X 486 pixels X 24 bits/pixel= 1,049,760 octets/image.

Conversion octets/image en K octets/image, 1, 049, 760 octets/image X 1 Ko/1024 octets= 1025,16 Ko.

Conversion K octets par image en K octets par seconde, 1025,16 Ko/image X 30 images/sec.= 30754,69 Ko/sec.

Conversion K octets par seconde en M octets par seconde, 30754,69 Ko/sec. X 1 Mo/1024 Ko = 30,03 Mo/sec.

rs/OPI_fr_M04_C05/co/Contenu_04.html

Bibliographie

[1]-A. Manzanera, chapitre(01), les images numériques, ENSTA/UEI.

[2]-Amimer Kheireddine, implantation sur fpga de deux filtres de traitement d'image, 78, 2011.

[3]- Olivier Gras, didacticiel du logiciel Xilinx ISE,2007

[4]-Mathias Ortner,' Techniques de compression vidéo des standards MPEG', Université de NICE-SOPHIA ANTIPOLIS, 27, 2004.

[5] -marechal , 'implémentation de l'oscillateur Collpitts sur FPGA, 28, 2006.

[6]- Kristen PAPIN, *Analyse comparative d'algorithmes de traitements d'images,* Projet de Fin d'Etude, Ecole Nationale Supérieure d'Electricité et de Mécanique *(ensem),* Nancy, France.

[7]- Patrick Bas, Compression d'Images Fixes et de Séquences Video cours ENSERG/INPG,22 , 2005.

[8]- H http://www.CoolUtils/MOV fichier desccription format.html

[9]-http://perso.telecom-paristech.fr/~maitre/BETI/filtres_lin_nlin/filtres/fitre median.html

[10]-http://www.edaboard.co.uk/matlab-simulink-system-generator-hdl-co-simulation- t413580.html

[11]- Mohammed Talibi Alaoui**,** *cours, Chapitre 2, Filtrage,* Département Mathématiques et Informatique, Oujda, Maroc

[12]- http://bwrc.eecs.berkeley.edu/classes/cs152/handouts/tris/tutorials_book.pdf

[13]- http://www.optique-ingenieur.org/fr/cours/Video%20Numerique.html.html
[14]- http://www.axis communication.html

[15]- http://www.axis communication.html

[16]-http://stats.deskshare.com/piwik/piwik.php?idsite=2"style="border:0"alt="" /></p>