REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR- ANNABA UNIVERSITY

UNIVERSITE BADJI MOKHTAR ANNABA

جامعة باجي مختار- عنابة

Année : 2019

Faculté: Sciences de l'ingéniorat

Département: Electronique

MEMOIRE

Présenté en vue de l'obtention du diplôme de : MASTER

Intitulé

Implémentation d'un algorithme d'estimation de flot optique sur plateforme reconfigurable (Xilinx)

Domaine : Sciences et Technologie

Filière : Electronique

Spécialité: Electronique Des Systèmes Embarqués

Par : Taifi Abdelkader

DEVANT Le JURY

PRESIDENT:	**S.Toumi**	**PR**	**UBMA**
DIRECTEUR DE MEMOIRE:	**M.Boughazi**	**PR**	**UBMA**
EXAMINATEURS:	**A.Boulamiz**	**M.C.B**	**UBMA**
	N.Kaddeche	**M.C.A**	**UBMA**

Remerciements

Je tiens tout d'abord à remercier dieu le tout puissant de m'avoir donné le courage, la force et la patience pour élaborer ce travail.

Mes vifs remerciements particulièrement à mon encadreur monsieur pr .Boughazi Mohammed pour son aide précieuse, ses conseils constructifs et ses orientations bénéfiques et objectives et surtout sa confiance durant les moments d'efforts pour la réalisation de mon projet.

Je tiens aussi à exprimer mes remerciements les plus respectueux à messieurs les membres de jury : messieurs S.Toumi, Md Boulamiz, Md kaddeche pour avoir acceptés d'examiner et juger ce travail.

Enfin, je remercier aussi toutes les personnes qui ont apporté leur aide pour l'accomplissement de ce travail.

Dédicaces

Je n'aurais jamais une occasion meilleur que celle-ci pour exprimer mon profond respect ,ma reconnaissance et mon grand amour à légard des personnes que j'aime qui m'ont toujours encouragé et soutenus.

Au symbole de la tendresse et l'affection,à celle qui s'est privée de tous ma combler,à ma défunte «ma mére» l'être la plus chère du monde, quoique je dise ou je fasse, cela n'exprime pas le degré de ma gratitude et on amour en vers elle.

A un grande homme pour lequel je garde un estime particulier …mon père celui qui tant sacrifié pour que j'arrive à me bâtir : je lui dois énormément.

A mes frères ainsi que tout la famille, sans oublié tous mes amis en particulier ceux qui ont aidés et encourager durant mon travail

ملخص

الهدف من نهاية مشروع الدراسة هو تنفيذ خوارزمية تقدير التدفق البصري على نظام مضمن يعتمد على FPGA. يعد تقدير التدفق المرئي أحد الأساليب المستخدمة لتحليل الحركة توجد هذه العملية في العديد من المجالات: الصناعة ، الروبوتات... إلخ. التدفق البصري هو موضوع تم علاجه في الثمانينيات من قبل العديد من العلماء (Lucas_Kanade Horn et Schunck)، لكنه لا يزال ضمن إطار التقدير او التقييم بمعنى اخر لم ينتج عنه بعد حل مرضٍ عام. تعتبر معالجة الفيديو والصور امر تقليدي في البرمجيات ، في حين أن التصميم FPGA يعتمد على الأجهزة او العتاد. لسد الفجوة، من الضروري التفكير في الخوارزميات ليس في حد ذاتها، ولكن من حيث البنية المعلوماتية الأساسية. في هذا العمل ، نحاول كتابة خوارزمية لتقدير التدفق البصري باستخدام الحل المقترح من طرف Horn et Schunck على FPGA مع مراعات خصائص هذه الاخيرة.

Résumé

L'objectif de ce projet de fin d'étude est d'implémenter un algorithme d'estimation de flot optique sur un system embarqué qui basée sue l' FPGA. L'estimation de flot optique est l'une des méthodes qui est utilisée pour l'analyse de mouvement dans des séquences d'images. on trouve ce procédé dans plusieurs domaines : industrie, robotique …etc. Le flot optique est un sujet qui a été traité dans les années 1980 par plusieurs savants (Horn & Schunck, et Lucas & Kanade) mais on reste toujours dans le cadre d'estimation c-à-d on n'arrive jamais à une solution satisfaisante et générale. Le traitement des séquences vidéo et des images est traditionnellement considéré comme une tâche du domaine logiciel (soft), alors que la conception basée sur FPGA est un résolument dans le domaine du matériel (hard). Pour remplir le vide il est nécessaire de penser aux algorithmes non pas en eux même, mais plutôt en termes d'architecture informatique sous-jacente. Dans ce travail on a essai d'écrire un algorithme pour estimer le flot optique à l'aide de solution proposée par Horn Et Schunck sur une plateforme FPGA, en tenant compte des caractéristiques de celle-ci.

Abstract

The objective of this final is to implement an optical flow estimation algorithm on an embedded system based on the FPGA. Optical flow estimation is one of the methods that is used for motion analysis in image sequences. we find this process in several fields: industry, robotics ... etc. The optical flow is a subject that was treated in the 1980s by several scientists (Horn & Schunck, and Lucas & Kanade) but we always remain in the framework of estimation ie we never reach a solution satisfactory and general. The processing of video sequences and images is traditionally considered a soft software task, whereas the FPGA-based design is a hard hardware one. To fill the void it is necessary to think about the algorithms not in themselves, but rather in terms of the underlying IT architecture. In this work we tried to write an algorithm to estimate the optical flow using a solution proposed by Horn and Schunck on an FPGA platform, taking into account the characteristics of this one.

Sommaire

Chapitre 3:applicaton et discussion de résultat

Tableau des Figures:

Liste des Tableau

Liste des Abréviations :

ECFO : Equation de contrainte de flot optique

CFO: contrainte Flot optique

BM: Block matching

SOC : System on chip

DSP : Digital signal processor

EP : Embedded processor

SoPC: System on Programmable Chip

GPP :General Purpose Processor

FPGA: Field programmable gate arrays

PLD : programmable logic device

VHDL : VHSIC Hardware description language

VHSIC : very High speed integrated circuits

XSG : Xilinx System generator

Introduction générale :

Depuis le début des années 70, la grande majorité des puces grandes public (CPU, GPU….) possède un circuit imprimé <<statique>>, Autrement dit, les circuits imprimés possèdent une architecture figée épaulée par des macro-instructions permettant l'exécution d'une tache spécifique tel que la compression/décompression, l'accélération 3D … etc. les applications logiciels doivent alors être programmés en fonction de l'architecture de la puce. Ces applications peuvent ensuite être optimisées pour une architecture spécifique, ce qui leur permet d'exécution plus rapidement. Une puce avec un circuit imprimé classique a l'avantage de pouvoir faire n'importe quelle application conçue pour son architecture, mais malgré les optimisations possibles, les applications fonctionneront toujours plus lentement avec ce type de circuits qu'avec un circuit spécialement conçu pour une application (accélération matérielle au lieu de logiciel). En effet, en informatique une solution matérielle est beaucoup plus rapides qu'une solution logicielle il serait donc intéressant de pouvoir modifier la structures interne et donc le fonctionnement de la puce électronique pour accélérer <<matériellement>> une application. C'est que proposent les puces FPGA. Ces caractéristiques de vitesse et d'autres de flexibilité et de Reprogramabilité rendre FPGA une plateforme d'implémentation performante pour les applications d'imageries intégrées.

L'étude de mouvement est devenir un procédé irremplaçable dans différentes domaines et exploiter pour plusieurs travaux comme la vidéo surveillance, la compression vidéo, l'imagerie médicale, la robotique, l'interaction homme machine, l'analyse de séquence sportives ... etc. L'analyse de mouvement est un domaine très vaste qui contient plusieurs problématiques parmi ces problématiques est estimé à partir d'une séquence d'images le mouvement apparent des objets composants une scène tridimensionnelle

Dans ce travail, on va étudier l'une des méthodes d'estimation de mouvement qui est l'estimation de flot optique par l'approche différentielle. Cette dernière a plusieurs méthodes qui calcule les vecteurs vitesse et chaque méthode a des avantages et des inconvenants. Nous intéresserons à la méthode de HORN & SCHUNK et trouver la meilleure façon pour son implémentation sur une plate forme FPGA.

Nous décomposons ce travail en 3 chapitres :

Chapitre 1 : L'estimation de mouvement

On présente des généralités sur l'estimation de mouvement et les différentes méthodes pour faire cette procédure.

Chapitre 2 : Conceptions des systèmes embarqués

De connaissances fondamentales pour les systèmes embarqués et des circuits logique programmable /FPGA.

Chapitre 3 : Implémentation et résultats expérimentaux

Etude comparatif entre les résultats obtenu par la simulation Simulink et l autre de XSG

1.1 Introduction :

L'étude de mouvement dans des séquences temporelles d'images Bidimensionnelles (2D) et tridimensionnelles (3D) est un des problèmes fondamentaux en traitement et analyse d'image, ce domaine de recherche récent est à l'origine de nombreux problèmes ouverts pour lesquelles il n'existe aucune solution satisfaisante et générale.

Parmi les opérations qui peuvent être appliquées dans une séquence vidéo on peut citer :

- détection de mouvement : connaitre les objets / des structures en mouvement.
- Quantifier le mouvement : mesure de vitesse / accélération des objets.
- Reconstruire le mouvement véritable des objets.

Ce chapitre débute par un aperçu sur l'estimation de mouvement, ensuite en va voir les problématiques entraves pour un bon diagnostic de mouvement et quelles sont les méthodes les plus connues dans ce domaine avec une étude détaillée de méthode différentielle proposée par Horn & Schunck.

1.2 Concepts et Terminologies :

Un document vidéo n'est qu'une suite d'images, il contient une structure hiérarchique équivalente d'un ouvrage, en général, on peut décomposer la vidéo en plusieurs niveaux : Objet, plan, scène et séquence qui sont considérés comme les niveaux de base pour l'analyse de documents vidéo.

L'Objet vidéo se rapporter à un Objet 3D du monde réel projeté sur un plan.

Une image peut contenir plusieurs objets sémantiques mais seulement quelque Objet d'intérêt.

Un plan est défini comme une séquence d'images prises sans interprétation temporelle par la caméra.

Les scènes sont définies comme des collections de plans, adjacents dans le temps, et qui Sont liées sémantiquement **[1].**

Une séquence d'image est une série de N images ou frames, acquises à des instants du temps discrets $T_k = T_0 + k\Delta T$, ou ΔT est un intervalle de temps fixe, et K =0, 1,2,…N-1 (nombres de frames) **[2]**.

Définition de mouvement :

Mouvement $\Longleftrightarrow$ **variation d'intensité**

Si l'on suppose que les conditions d'illumination ne varient pas et que les surfaces sont lambertiennes, les changements de luminance dans l'image sont dus à un mouvement relatif entre la caméra et la scène :

- caméra mouvement devant une scène statique.
- Parties de la scène en mouvement devant une caméra statique.
- camera et objet mouvant ensemble **[3]**.

1-3 interprétations spatiales et temporelles des séquences d'image :

1-3-1 détection de mouvement : dans une séquence d'images on cherche à distinguer les zones fixes et mobiles d'une scène, d'autres manières identifier dans chaque image les pixels apparents à des objectes mobiles.

1-3-2 Estimation : calculer le mouvement apparent (vitesse instantanée) de chaque pixel **[4]**

1-3-3 Segmentation : cette tache va au-delà de la détection de mouvement puisqu' il s'agit de segmenter chaque image en régions qui représente une homogénéité du mouvement apparent. Cette opération est généralement réalisée à partir d'une estimation du flot optique **[5]** ou des dérivées spatiotemporelles de l'intensité lumineuse.**[6]**

1-3-4 L'étiquetage de régions (ou étiquetages en composantes connexes) : est opération fondamentale du traitement d'images. Ce traitement affecte un numéro d'identification, ou étiquette à chaque composante connexe d'une image binaire **[7,8]**

1-3-4 L'analyse : pour l'étude de mouvement, il existe plusieurs méthodes que l'on peut les regrouper en 2 grandes catégories :

- des méthodes globales.
- des méthodes différentielles.

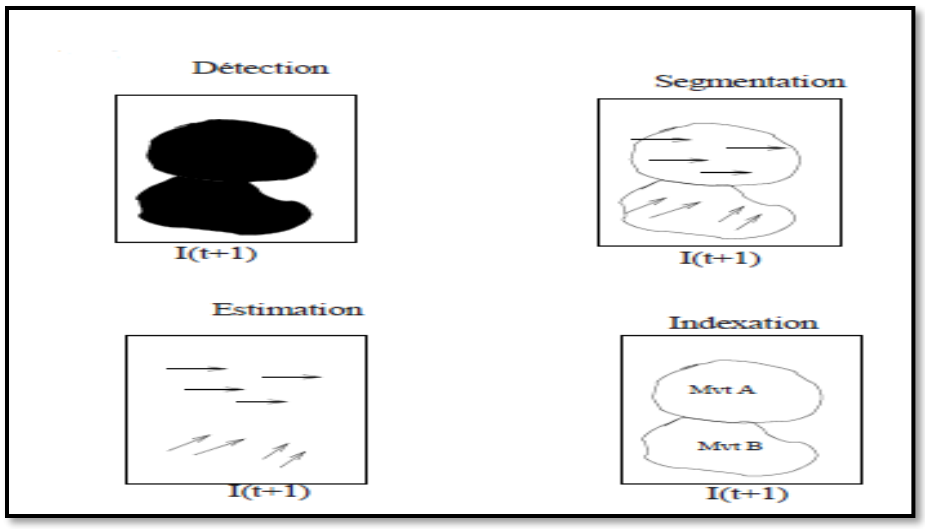


Figure 1.1 la différente phase d'interprétation spatiotemporelle d'une séquence d'image

1-4 Méthodes globales :

1-4-1 Méthode de Fourier : cette méthode repose sur L'analyse de fréquence des images (au lieu du spatial), il permet l'analyse de mouvement globale affine. **[9]**

La formule :

$$\hat{\boldsymbol{I}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{t}) = \int_{-\infty}^{+\infty} \boldsymbol{I}\,(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})\boldsymbol{e}^{-2\pi\boldsymbol{i}(\boldsymbol{xu}+\boldsymbol{yv})}\,\boldsymbol{dx}\,\boldsymbol{dy}$$

En suppose que dans l'instant t+Δt (Δt : variation de temps) en fait un mouvement uniforme de translation τ = (Δx, Δy).

Dans l'espace cartésien en a :

$$\boldsymbol{I}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t}) = \boldsymbol{I}(\boldsymbol{x}+\Delta\boldsymbol{x},\boldsymbol{y}+\Delta\boldsymbol{y},\boldsymbol{t}+\Delta\boldsymbol{t}) \quad (dans\ lespace\ temporelle)$$

Dans l'espace fréquentielle :

$$\hat{\boldsymbol{I}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{t}) = \iint_{\boldsymbol{\omega}} \boldsymbol{I}\,(\boldsymbol{x}+\Delta\boldsymbol{x},\boldsymbol{y}+\Delta\boldsymbol{y},\boldsymbol{t}+\Delta\boldsymbol{t})\boldsymbol{e}^{-2\pi\boldsymbol{i}(\boldsymbol{xu}+\boldsymbol{yv})}\,\boldsymbol{dx}\,\boldsymbol{dy}$$

$$\boxed{\hat{\boldsymbol{I}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{t}) = \hat{\boldsymbol{I}}(\boldsymbol{U},\boldsymbol{V},\boldsymbol{t}+\Delta\boldsymbol{t})\boldsymbol{e}^{-2\pi\boldsymbol{i}(\boldsymbol{xu}+\boldsymbol{yv})}\,\boldsymbol{dx}\,\boldsymbol{dy}}$$

Extraction de terme exponentiel de cette équation :

Par corrélation, posons que : $\hat{\boldsymbol{j}}(\boldsymbol{u},\boldsymbol{v}) = \hat{\boldsymbol{I}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{t}+\Delta\boldsymbol{t})$ **:**

$$\boxed{\textbf{Corrélation } (\hat{\boldsymbol{I}},\hat{\boldsymbol{j}}) = \frac{\bar{\hat{\boldsymbol{J}}}\hat{\boldsymbol{I}}}{|\hat{\boldsymbol{I}}||\hat{\boldsymbol{j}}|} = \boldsymbol{e}^{2\boldsymbol{i}\pi(\boldsymbol{u}\Delta\boldsymbol{x}+\boldsymbol{v}\Delta\boldsymbol{y})}}$$

1-4-2 Détection par la transformée de Hough :

-Principe : comptabiliser tous les déplacements de point possible (admissibles) dans un espace d'accumulation puis faire l'analyse.

-En utilisant la convention de la brillance (luminance) comme critère de déplacement admissible.

-**algorithme :**

En considérons qu'en a 2 plans successifs It et It+Δt , H(t) est l'accumulateur d' Hough tel que :

$$\forall(a,b), Ht(a,b) = 0$$

Pour tout couple (a, b)∈ $\boldsymbol{\omega h}$ fair :

Si : $\boxed{I\text{t(x, y)} = It + \Delta t(x+a, y+b) \text{ alors : } \quad Ht(a,b) = Ht(a,b) + 1}$

-Les avantages et les inconvénients de cette méthode :

Avantages	Inconvénients
- Aucune information n'a priori sur le mouvement et nécessaire. - Analyse simultanée de différents objets possibles	- Fort cout algorithmique. - L'espace de Hough obtenir et très bruites : l'interprétation difficile (localisation des points d'accumulation).

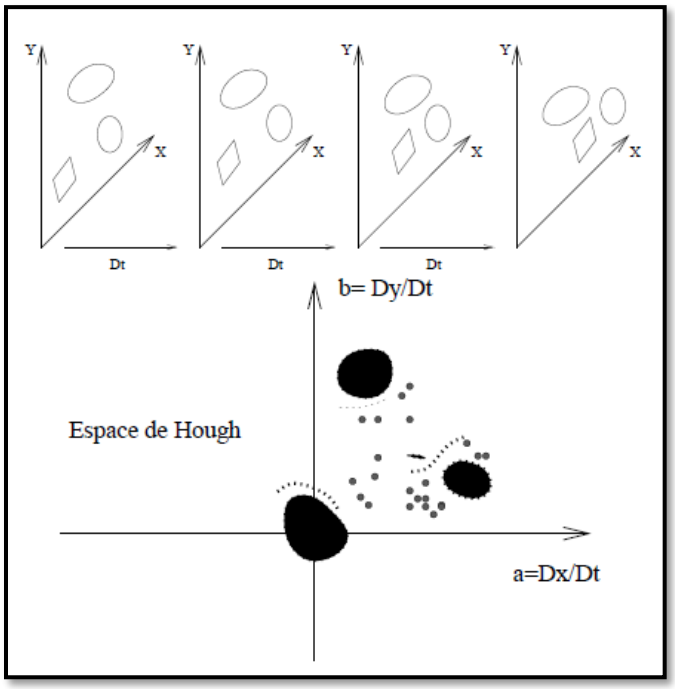


Figure 1.2 Exemple de déplacement et leur impact dans l'espace d'Haugh

1-4-3 Détection par différence d'image :

-Historiquement : parmi les premières méthodes.

-But : analyser les différences temporelles pixel à pixel entre 2 plans.

-hypothèse basique : les zones actives dans le temps correspondent à un mouvement :

Elles induisent une variation temporelle des niveaux de gris.

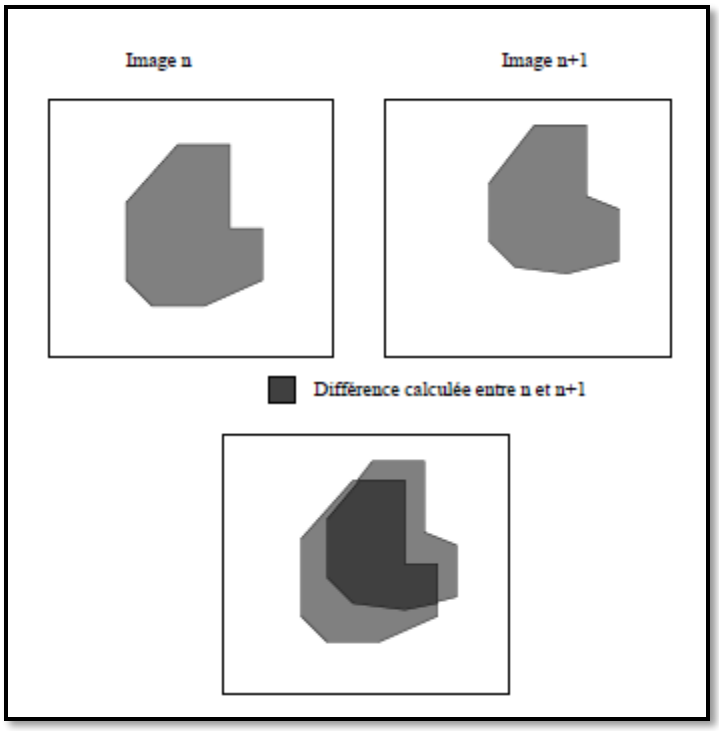


Figure 1.3 Différance d'images

Définition de l'image de différence entre un plan J et un plan K :

$$DPjk(x,y) = \begin{cases} 1 \\ 0 \end{cases} \Leftrightarrow |I(x,y,j) - I(x,y,k)| > \tau$$

D'autres choix sans possibles :

- Ajout d'une clause de connexité ou voisinage.
- Seuillage plus souple : évaluer la quantité :

$$\gamma(x,y) = \frac{\frac{\sigma_j^2 + \sigma_k^2}{2} + (\frac{\mu_j - \mu_k}{2})^2}{\sigma_j \sigma_k}$$

- (μ_j, σ_j) moyenne et variance sur le voisinage du pixel dans le plan J
- (μ_k, σ_k) moyenne et variance sur le voisinage du pixel dans le plan K

- Différances Cumulées :

-caractéristique :

-Cette méthode ne peut que détecter les zones de mouvement

-Amélioration de la méthode pour estimer qualitativement le mouvement en utilisant plusieurs plans comparés à un plan de référence R.

-**principe:**

Définition de l'image de différence cumulée de K-iémé plan par rapport au plan de référence R

$$ADP°(x, y) = 0$$

$$ADP^k(x, y) = ADP(\mathrm{k} - 1) + \mathrm{DP\ rk(x, y)}$$

L'image ADPk contient la trace d'un Object en mouvement.

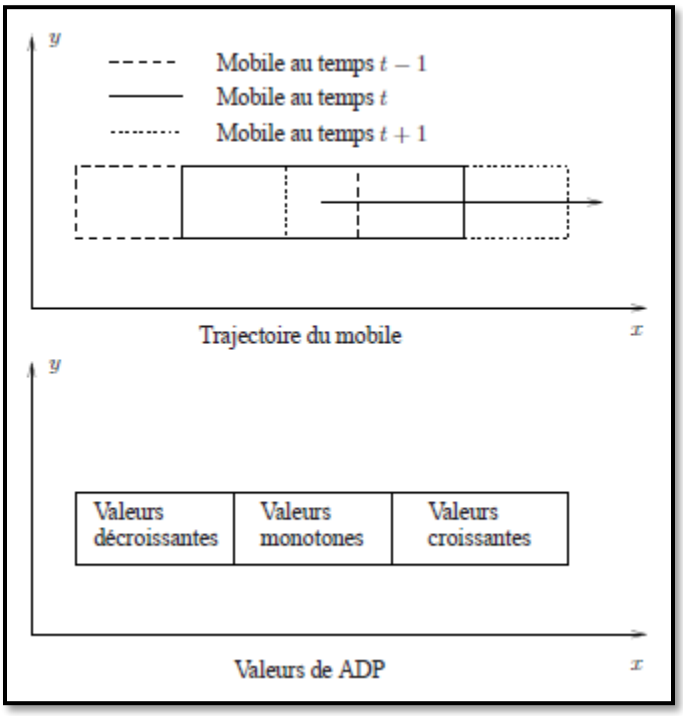


Figure 1.4 – Différance d'image Cumulées

- Avantages :

-Détection et estimation du mouvement : intensité et direction.

-Peux couteux, relativement efficace : bon pour tempes réelles.

-Fortes hypothèses : mouvement rectiligne & constant, point à point.

-implémentation sur suivre balistique.

-illumination globale constante.

1-4-4 La méthode de mise en correspondance (Block Matching) :

Elle cherche à déterminer des similarités entre des points, des contours ou des régions, présents dans deux images successives. Pour cela, une fonction de corrélation doit être maximisée ce sont en général des régions constituées de petits blocs carrées qui sont considérées, seules les translations sont tolérées, car ce modèle suppose que tous les pixels appartenant à une même région subissent un déplacement identique les norme MPEG (par exemple) recourt à cette méthode.

-**Définition de corrélation :**

Soit X=(x,y) : les cordonnés d'un point.

$\delta = (\Delta x, \Delta y)$: Les vecteurs de déplacement.

I(x,t) : la fonction spatio-temporelle des valeurs de niveaux de gris d'une séquence.

-La corrélation de point X pour un déplacement δ et donnés par la relation :

$$C(x,\delta) = \sum_m W(m)[F\big(I(x+m,t)\big) \Diamond F\big(I(x+m+\delta, t+\Delta t)\big)]$$

F: opérateur quelconque.

$\Diamond$: Opérateur de comparaison.

W : une fenêtre de calcul.

Direct :

$$Cd(X,\delta) = \sum w(m) I(X+m,t) * (X+m+\delta, t+\Delta t)$$

Normaliser la relation précédente par rapport a la moyenne :

$$CM(x,\delta) = \sum W(m)(I(x+m,t) - \bar{I}\,(x,t)) * (I(x+m+\delta, t+\Delta t) - \bar{I}\,(x+\delta, t+\Delta t))$$

$\bar{I}$: moyenne d'I sur la fenêtre.

-Variation normalisée :

$$Cv(x,\delta) = \frac{Cm(x,\delta)}{\overline{var}\big(I(x,t)\big) * \overline{var}\big(I(x+\delta, t+\Delta t)\big)}$$

C'est la corrélation statique usuelle

Corrélation binaire :

$$Cb(x,\delta) = \sum W(m) \boldsymbol{B} It(x+m) * \boldsymbol{B} I(t+\Delta t) \Delta(x+m,\delta)$$

-Corrélation binaire issue d' I(image des contours, seuillage).

Corrélation sur un filtre laplacien :

$$Cl(x,\delta) = \sum W(m)\boldsymbol{L}It(x+m) * \boldsymbol{L}It + \Delta t(x+m+\delta)$$

Avec :

$$\mathbf{L}I = \Delta I = \frac{\delta^2 I}{\delta x^2} + \frac{\delta^2 I}{\delta y^2}$$

Erreur moyenne (statique d'ordre 1)

$Ce(x,\delta) = \sum\ W(m)(I(x+m,t) - I(x+m+\delta,t+\Delta t)\ )^2$ .....etc.

Grande variabilité de corrélation :

Dépend de la situation, des propriétés des Objet à reconnaître.

-Quelques avantages et inconvénients :

-Peut travailler à déférentes échelles (pixel, voisinage, gros Objet).

-Facile à implémenter.

-Inadapté à fournir un champ dense de vitesse sur une image.

-par contre : très efficace sur la poursuite d'objet rigide ou peu déformable.

- les calculs sont lords

-un cout minimisé.

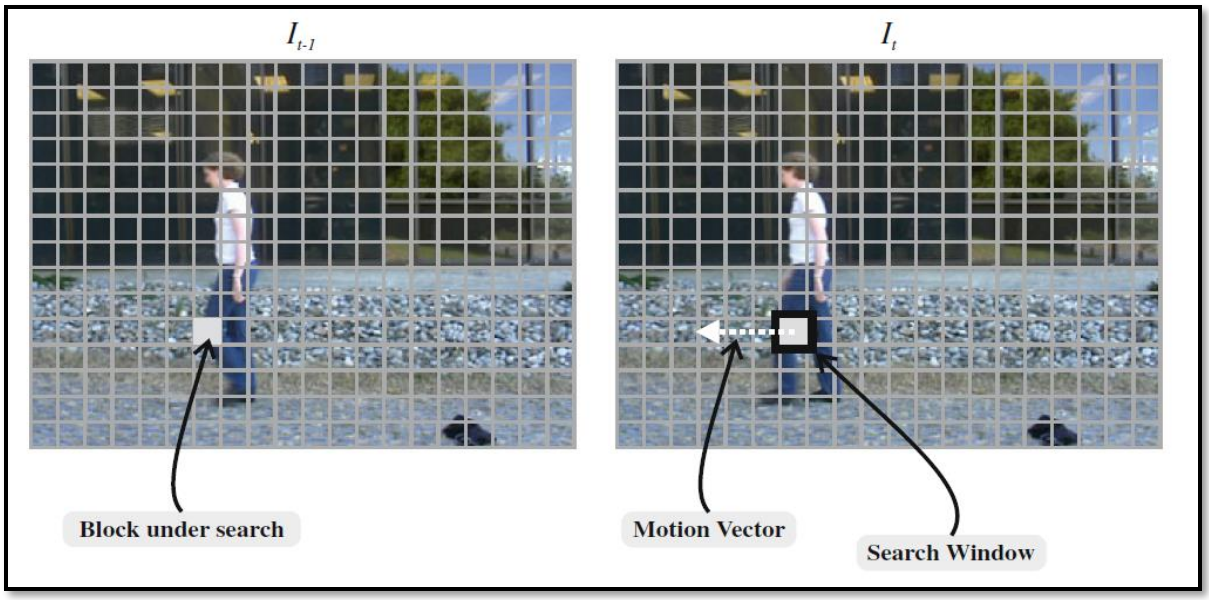


Figure 1.5 procédure de mise en correspondance (block matching)

1-4-5 Détections par seuillage statique :

Opération consiste au test du rapport de vraisemblance.

-But de cette technique : être capable de décider un changement significatif entre 2 images.

Un changement significatif : la répartition de niveaux de gris est différente entre les 2 images : test d'inégalité des lois sur les images.

1-5 Méthodes différentielles :

-Définition de flot optique : le flot optique est la distribution de la vitesse par rapport à l'observateur, en chaque point de l'image.

Autre définition : mouvement apparent dans les images, transport de la luminosité.

-équation de flot optique :

Il faut mettre en compte :

-Les considérations humaines

-Les considérations physiques

-Hypothèse de flot optique :

Les séquences d'image sont (localement) invariantes en luminosité.

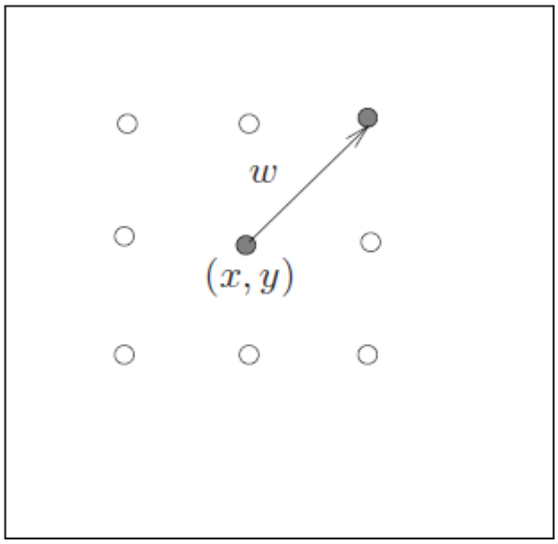


Figure 1.6 Transport de la luminosité

-La formule :

Selon l'hypothèse en l'écrite :

$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$

Développement de Taylor :

Ici très petit en l'Ignorer

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\sigma I}{\sigma x}\sigma x + \frac{\sigma I}{\sigma y}\sigma y + \frac{\sigma I}{\sigma t}\sigma t + HOT\ (heigt\ order\ term)$$

L'équation devienne:

$$I(x, y, t) = I(x, y, t) + \frac{\sigma I}{\sigma x}\sigma x + \frac{\sigma I}{\sigma y}\sigma y + \frac{\sigma I}{\sigma t}\sigma t$$

$$\frac{\sigma I}{\sigma x}\sigma x + \frac{\sigma I}{\sigma y}\sigma y + \frac{\sigma I}{\sigma t}\sigma t = 0$$

$\omega = (\sigma x = Vx, \sigma y = Vy)$: Déplacement de point I (selon x et y) au temps t.

$$\left(\frac{\sigma I}{\sigma x}\sigma x + \frac{\sigma I}{\sigma y}\sigma y + \frac{\sigma I}{\sigma t}\sigma t\right) * \left(\frac{1}{\sigma t}\right) = 0 * \left(\frac{1}{\sigma t}\right)$$

Ix Iy It

$$\frac{\sigma I}{\sigma x}\frac{\sigma x}{\sigma t} + \frac{\sigma I}{\sigma y}\frac{\sigma y}{\sigma t} + \frac{\sigma I}{\sigma t}\frac{\sigma t}{\sigma t} = 0$$

$\nabla I = \left(\frac{\sigma I}{\sigma x}, \frac{\sigma I}{\sigma y}\right)$:Le gradient d'intensité spatial

$$\omega \nabla I + It = 0$$

L'équation de contrainte de flot optique (E.C.F.O)

-Problème de l'ouverture :

-Il manque une contrainte sure ω pour le système soit résolvable (pas sa dépendance linière).

-Pas de réponse universelle quelle contrainte justifiée prendre ?

1 ère solution (ECFO) :

En fait plusieurs acquisitions de la même scène :

$$\omega \nabla I^i + It^i = 0$$

Les I^i sont différentes acquisitions la mémé scène obtenue par :

-Image multi spectrale : est une image qui capte les données d'images a des longueurs d'onde spécifique du spectre électromagnétique, les longueurs d'ondes pouvant être séparés par des filtres aux l'utilisation des instruments sensibles a des longueurs d'onde particulaires.

-Image a différents points de vue.

-Image en stéréovision : lorsque en utilise plusieurs caméras pour l'acquisition des images de la mémé scène en appeler ça stéréovision

-dériver des images : obtenir plusieurs images à traiter à l'aide de filtrages (ex : Laplacien).

Des contraintes supplémentaires :

Si poussons-en à une seule séquence ? Dans ce cas en utilisent des filtres pour créer des nouvelles images.

Si i=1,2 trouver condition nécessaire et suffusion pour flot optique soit bien posé.

CNS pour inversion de CFO:

Selon L'équation précédant :

$$\begin{cases} UI_x^1 + VI_Y^1 + I_t^1 = 0 \\ UI_x^2 + VI_Y^2 + I_t^2 = 0 \end{cases} \Leftrightarrow \quad A\omega = F$$

Est inversible si detA≠0 donc : $I_x^1 I_Y^2 \neq I_x^2 I_Y^1$.

Pas de dépendance linéaire entre ∇I^1 et∇I^2.

Autres contraints:

- Imposer un modèle de mouvement : paramétrer la fonction ω : un modèle de mouvement affine sur ω (comme exemple).

-Rechercher des zones localement homogènes.

-Parfois possible : ajouter d'autres contraintes justifiables physiquement. Les implémentations :

- variationnelles, paramétriques, probabilistes
- discrets (programmation dynamique), ...etc.

1-5-1 Formulation variationnelle:

-Méthode Horn et Schunck :

-La méthode de Horn et Schunck n'est pas limitée aux translations. C'est une méthode différentielle itérative adaptée à l'estimation des petits déplacements et basée sur le développement en série de Taylor, des gradients spatiaux et temporaux. Le but est de trouver le champ de vecteurs qui satisfait l'équation de contrainte du mouvement en chacun pixel.

-Historiquement : la première modelé différentiel [Horn et Schunck, 1981].

– méthodologie : problème d'optimisation numérique, résolu par une **formulation vibrationnelle.**

– Le modèle :

– Invariance de la luminosité (Contrainte du flot optique)

– Régularité du champ de vitesses : une contrainte sur la variation du champ des vitesses suffit à lever l'indétermination.

– Formulation sous forme une énergie à minimisé : très courant en imagerie (méthode très intuitive malgré le contexte mathématique).

Modelé :

$$E(w) = \int_{\Omega} (\omega \nabla I + It)^2 dxdy + \alpha^2 \int_{\Omega} \|\nabla w\|^2 dxdy$$

- Premier terme : contrainte du flot optique.

- Second terme : contrainte de régularité sur ω.

-E : énergie, en doit chercher la fonction qui minimise ω cette énergie.

-E plus petit que :

1- CFO plus proche de zéros.
2- Le champ de vitesses w est régulé.

$$\|\nabla w\|^2 = \|\nabla u\|^2 + \|\nabla v\|^2$$

$$\|\nabla w\|^2 = (\frac{\sigma u}{\sigma x})^2 + (\frac{\sigma u}{\sigma y})^2 + (\frac{\sigma v}{\sigma x})^2 + (\frac{\sigma v}{\sigma y})^2$$

-Résolution numérique :

-principe : équations d'Euler-Lagrange associées au problème de minimisation.

-la solution : DE (w)= 0

est le minimum de l'énergie (E doit être convexe).

Donc :

$$E(w) = \int_{\Omega} (\omega \nabla I + It)^2 dxdy + \alpha^2 \int_{\Omega} \left[(\frac{\sigma u}{\sigma x})^2 + (\frac{\sigma u}{\sigma y})^2 + (\frac{\sigma v}{\sigma x})^2 + (\frac{\sigma v}{\sigma y})^2 \right] dxdy$$

$$E(w) = \int_{\Omega} (\omega \nabla I + It)^2 dxdy + \alpha^2 \int_{\Omega} [|\nabla u|^2 + |\nabla v|^2] dxdy$$

Selon l'équation de Lagrange :

$$\begin{cases} \frac{\sigma E}{\sigma \mu} - \frac{\sigma}{\sigma x}\left(\frac{\sigma E}{\sigma\left(\frac{\sigma u}{\sigma x}\right)}\right) - \frac{\sigma}{\sigma y}\left(\frac{\sigma E}{\sigma\left(\frac{\sigma u}{\sigma y}\right)}\right) = 0 \\ \frac{\sigma E}{\sigma v} - \frac{\sigma}{\sigma x}\left(\frac{\sigma E}{\sigma\left(\frac{\sigma v}{\sigma x}\right)}\right) - \frac{\sigma}{\sigma y}\left(\frac{\sigma E}{\sigma\left(\frac{\sigma v}{\sigma y}\right)}\right) = 0 \end{cases}$$

$$\begin{array}{l|l} \mathrm{Ix}(\mathrm{IxVx} + \mathrm{IyVy} + \mathrm{It}) - \alpha^2(\Delta u) = 0 & \Delta\mu = \bar{u} - u \\ \mathrm{Iy}(\mathrm{IxVx} + \mathrm{IyVy} + \mathrm{It}) - \alpha^2(\Delta v) = 0 & \Delta v = \bar{v} - v \end{array}$$

Après remplacé ($\Delta\mu$, Δv) et développer les équations en trouve :

$$(\alpha^2 + \mathrm{Ix}^2 + \mathrm{Iy}^2)(u - \bar{u}) = -\mathrm{Ix}(\mathrm{Iy}\bar{v} + \mathrm{It} + \mathrm{Ix}\bar{u})$$

$$(\alpha^2 + \mathrm{Ix}^2 + \mathrm{Iy}^2)(v - \bar{v}) = -\mathrm{Iy}(\mathrm{Iy}\bar{v} + \mathrm{It} + \mathrm{Ixu})$$

$$\overline{\cup n} = \mathrm{A} \cup n, \quad \cup n + 1 = f(\overline{\cup n})$$

Finalement : La relation de Horn et Schunck est :

$$\mathrm{U} = \bar{\mathrm{U}} - \frac{\mathrm{Ix}(\mathrm{Ix}\,\bar{\mathrm{U}} + \mathrm{Iy}\,\bar{\mathrm{V}} + \mathrm{It})}{(\alpha^2 + \mathrm{Ix}^2 + \mathrm{Iy}^2)}$$

$$\mathrm{V} = \bar{\mathrm{V}} - \frac{\mathrm{Iy}(\mathrm{Ix}\,\bar{\mathrm{U}} + \mathrm{Iy}\,\bar{\mathrm{V}} + \mathrm{It})}{(\alpha^2 + \mathrm{Ix}^2 + \mathrm{Iy}^2)}$$

– La condition précédente dépend d' A (approximation du Laplacien) et mais aussi des gradients de l'image : en pratique cela fonctionne.

1-5-2 Mise en correspondance BM et flot optique :

-Introduction :

On a vu la méthode de mise en correspondance BM dans les méthodes globales mais aussi il Ya aussi un lien entre cette méthode BM et le FO qui est incluse parmi les méthodes différentielles, cette liaison est étude en 1981 par Lucas Et Kanade.

-Approche de Lucas Et Kanade [1981] :

Vue comme une implémentation directe de la contrainte de flot optique, sans formulation variationnelle (mais les papiers sont sortis indépendamment la même année).

Initialement d´développée dans une application de recalage (non rigide) : mettre en correspondance une paire d'images.

-Pose le problème :

Trouver ω telque : $\mathbf{I_2(x+\omega_x)=I_1(x)}\ \forall x$

En minimise donc : $\mathbf{E(\omega)=\sum_x(I_2(x+\omega_x)-I_1(x))^2}$

ω Est un vecteur de très grandes tailles : une minimisation combinatoire est trop couteuse.

-Principe de la méthode :

Restriction a une fenêtre d'observation pour un pixel donné :

$\mathbf{E(\omega_x)=\sum_{y\epsilon w_x}(I_2(x+\omega_x)-I_1(x))^2}$

Ou ω_x est une fenêtre centrée autour du pixel X.

Linéarisation Du Cout Quadratique (Développement De Taylor)

$\mathbf{I_2(x+\omega_x)\sim I_2(y)+<\nabla I_2(y),\omega_x>}$

Donc l'équation de LUCAS et KANADE devient :

$$\mathbf{E(\omega_x)=\sum_{y\in\omega_x}(I_2(y)-I_1(y)+<\nabla I_2(y),\omega_x>)^2}$$

On pose $\mathbf{I_{21}(y)=I_2(y)-I_1(y)}$C'est une donnée d'entrée (la dérivée temporelle).

Le terme $\mathbf{I_{21}(y)+<\nabla I_2(y),\omega_x>}$est linéaire en $\boldsymbol{\omega_x}$ (vecteur à deux composantes).

On note que $\mathbf{<\nabla I_2(y),\omega_x>=\nabla I_2^T(y)\omega_x}$

$\mathbf{I_{21}=(I_2(y),y\in\omega_x)}$

La valeur de $\boldsymbol{\omega_x}$ qui minimise l'énergie (12) est donnée directement par la formule des moindres carrés :

$\mathbf{\omega_x=-(\nabla I_2\nabla I_2{}^T)^{-1}\nabla I_2\nabla I_{21}}$

En pose que : $\mathbf{A=\nabla I_2{}^T, B=I_{21}}$, **X =** $\mathbf{(u,v)^t}$

Il faut résoudre équation **AX=-B** (mais maitrise A n'est pas carré donc l'équation no inversible)

En l'écrite : $\mathbf{A^TAX=-BA^T}$

Maintenant $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ est une matrice carrée (maitrise multiple par leur transposer égale une matrice carrée) donc peut être inversible.

$$\mathbf{X} = -(\mathbf{A} * \mathbf{A}^{\mathrm{T}})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{B}$$

1-6 Conclusion

Dans ce chapitre en a essayer de donner quelques notions fondamentales sur l'estimation de mouvement, aussi le problème d'ouverture qui peut apparaitre dans cette opération et les 2 grandes catégories des méthodes pour éliminer ce problème : des méthodes globales et différentielles.

En a étude de ces méthodes de coté : principe, mathématiques, avantages/inconvénient aussi ont passé obligatoirement par la définition de certaines notions : L'Objet, Un plan, détection de mouvement…etc

2.1 Introduction:

Un système embarqué est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Ses ressources sont généralement limitées**.**

- "Embedded system": tout système conçu pour résoudre un problème ou une tâche spécifique mais n'est pas un ordinateur d'usage général.„
- Utilisent généralement un microprocesseur combiné avec d'autres matériels et logiciels pour résoudre un problème de calcul spécifique.„
- Système électronique et informatique autonome ne possédant pas des entrées-sorties standards.„
- Le système matériel et l'application sont intimement liés et noyés dans le matériel et ne sont pas discernables comme dans un environnement de travail classique de type PC.

Grace à cette définition, il est possible de comprendre les différentes caractéristiques de base typiquement :

- Dédié à une application spécifique
- Coût réduit, maximisation rapport performance/prix
- Volume restreint (compact, pas modulaire)
- Capacité mémoire adaptée
- Capacité de calcul appropriée à l'application
- Exécution temps réel (souvent)
- Fiabilité et sécurité de fonctionnement
- Consommation d'énergie maitrisée
- voir très faible en cas d'utilisation sur batterie

2.2 Conception des systèmes numériques:

Le domaine des systèmes embarqués est très vaste, qui peut inclure différentes architectures basées sur µp comme : DSP, ASIC, ASIP, GPP, FPGA….etc. Dans ce chapitre on ne peut pas détailler chaque architecture en détail, mais de définir seulement quelques aspects importants qui ont un lien avec notre travail (FPGA, circuits logiques programmables) et masquer les détails les autres aspects, et à la fin de ce chapitre, on donne un aperçu sur l'implémentation de quelques outils de traitement d'image dans les circuits FPGA.

2.2.1 *Architectures à base d'un processeur*

Les GPP (General Purpose Processor) sont des processeurs dont l'image est souvent associée à celle de l'ordinateur personnel (PC) voir figure 2.1. Leur popularité est liée à leur flexibilité, leur simplicité d'utilisation et leur puissance de calcul. L'association processeur-PC n'est toujours vraie, puisque les processeurs destinés aux systèmes embraqués occupent la plus grande partie du marché des processeurs. Par contre, les processeurs destinés aux systèmes embarqués sont différents des processeurs destinés aux ordinateurs grand public. D'abord le concept CISC (Complex Instruction Set Computer) a subi au cours des années une « cure d'amincissement » afin d'arriver à des architectures RISC (Reduced Instruction Set Computer) dans lesquelles les instructions et les modes d'adressage complexes étaient bannis. Des principes de simplification ont été mis en œuvre (comme par exemple le codage uniforme des instructions) afin de permettre notamment d'avoir des compilateurs fournissant de bonnes performances.

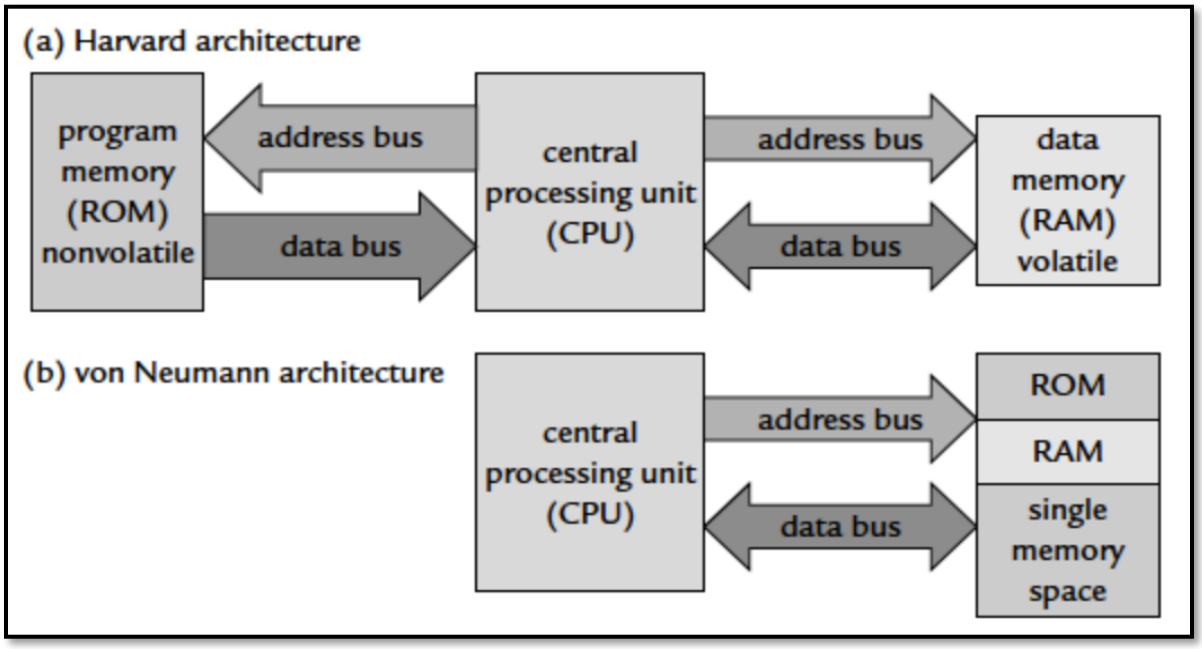


Figure 2.1 : Deux grands types d'architectures

2.2.2 Les processeurs d'usage général (GPP)

La puissance de calcul et la flexibilité sont les objectifs premiers d'un GPP. Ils sont atteints grâce à une fréquence de fonctionnement élevée et des architectures matérielles et logicielles adaptées : super scalaire, fort pipeline, prédiction de branchement, caches multi-niveaux, instructions spécialisés (par exemple l'instruction multimédia : MMX pour le Pentium et AltiVec pour le PowerPC) et extension SIMD (Single-Instruction Multi-Data). Cependant, un GPP n'est

pas bien adapté aux applications embarquées du traitement du signal et de l'image, et cela à cause de deux principales raisons :

- Dissipation d'énergie généralement élevée dans les GPP.
- Les GPP ne sont pas adaptés au temps réel pour les applications complexes.

On retrouve dans la plupart des systèmes sur puce des processeurs à usages généraux comme par exemple PowerPC, ARM, MIPS, ST-Microelectronics, etc. Ces processeurs ont des caractéristiques plus ou moins rapprochées. Ils sont destinés à effectuer des traitements ordinaires, sans qu'ils aient des performances optimales. La différence remarquable de ces processeurs utilisés dans les systèmes sur puce par rapport aux processeurs ordinaires, de la famille 80xx d'Intel et des 68xx de Motorola par exemple, réside dans le fait qu'ils utilisent du pipeline et de la mémoire cache afin de limiter le nombre d'accès à la mémoire centrale du système et de minimiser, par conséquent, l'utilisation du bus et de la consommation d'énergie du système.

2.2.3 Les processeurs de traitement du signal

Certains systèmes numériques intègrent également des processeurs à usages spécifiques pour des applications de traitement du signal et des images (DSP pour Digital Signal Processors). Ces processeurs spécialisés sont surtout destinés aux traitements d'images de type «bas niveau» pour lesquels les calculs sont réguliers (les traitements sont identiques pour toutes les données d'entrée) et doivent être réalisés à hautes fréquences.

Les DSP sont des processeurs simplifiés relativement semblables aux GPP. Leur particularité essentielle est qu'ils sont conçus pour effectuer des calculs en temps réel et intègrent donc de nombreux opérateurs. En effet leur architecture interne est dimensionnée pour le calcul intensif. Suivant les modèles, ils permettent de réaliser des opérations sur des nombres soit à virgule fixe, soit à virgule flottante. Ces processeurs permettent également un accès rapide aux données par des adressages particuliers. Leur jeu d'instructions est souvent plus réduit que celui d'un processeur traditionnel et peut être programmé soit en assembleur, soit avec un compilateur C dédié. Ces processeurs ont la possibilité de réaliser plusieurs instructions en parallèle et possèdent des opérateurs de calculs arithmétiques flottants très performants. Dans un DSP, les transferts entre les périphériques et la mémoire d'une part et entre la mémoire interne et la mémoire externe d'autre part, sont réalisés par le DMA (Direct Memory Access), qui permet de décharger le cœur de calcul, et de paralléliser les deux opérations (accès à la mémoire et le traitement). Des interfaces série, PCI, mémoire sont également intégrées au DSP.

Les DSP sont accompagnés d'outils de développement performants. Les débogueurs fournissent des informations de bas niveau pour valider rapidement une application. Les fabricants fournissent en général des compilateurs capables d'optimiser fortement un programme et d'informer le développeur sur le résultat en ajoutant des commentaires dans le programme (boucle optimisée, nombre de cycles, instructions utilisées). Des simulateurs permettent d'obtenir des informations supplémentaires sur l'exécution d'une application (charge de calcul, comportement des mémoires caches, temps d'exécution).

Un DSP est utilisé dans plusieurs domaines d'application qui nécessitent l'utilisation de filtres numériques ou adaptatifs, des FFTs, dans l'instrumentation (analyse transitoire, spectrale), dans le domaine médical (monitoring, échographie, imagerie médicale), dans les applications de contrôle (asservissement, robotique), ainsi que dans le multimédia, l'imagerie, le militaire (radar, guidage de missile), les télécommunications (modems radio, cryptage de données, répéteurs de ligne) et les applications grand public (automobile, électroménager), etc.

2.3 Les processeurs reconfigurables

En plus des GPP et des DSP, on retrouve également des processeurs reconfigurables qui peuvent intégrer des unités fonctionnelles selon le besoin. Ces processeurs peuvent être taillés sur mesure selon les besoins de l'application et de gagner par conséquent énormément de ressources. En effet, un processeur softcore (synthétisable) est un processeur implémenté sur un système reprogrammable comme un FPGA (Field Programmable Gate Array). Si on intègre plusieurs processeurs sur la même puce, on parle alors de système sur puce programmable (System on Programmable Chip ou SoPC).

Architecture très flexible de par sa nature, une implémentation softcore d'un processeur peut être reconfigurée en tout temps contrairement à un processeur dit hardcore dont le cœur dispose de sa propre puce qui ne peut être modifiée. Un processeur softcore s'adapte donc aux besoins de ses développeurs et aux contraintes matérielles (périphériques, performances, consommation, etc.). Toutefois, ces performances sont inférieures à celles d'un processeur hardcore. Un softcore est en général programmé dans un langage de description matérielle comme le VHDL (acronyme de VHSIC-HDL : Very High Speed Integrated Circuit Hardware Description Language) ou le Verilog. Parmi les processeurs softcore les plus connus, on peut citer : le Microblaze de Xilinx, le NIOS de la société Altera, Openrisc d'Opencore et le LEON de chez Gaisler Research.

2.4 Les circuits logiques programmables

2.4.1 Introduction :

Les circuits logiques programmables représentent une solution incontournable dans le domaine de l'électronique numérique. En effet, la possibilité de programmer un composant pour qu'il puisse fonctionner selon les besoins du concepteur est une aide précieuse pour pouvoir élaborer efficacement un circuit complexe avec des délais court et un coût de reviens faible. Contrairement aux circuits logiques prédéfini par les fabricants tels que les portes logiques (la famille 4000…) ou les fonctions logiques (les contrôleurs de bus …) qui ont des entrées sorties fixe une architecture figée un encombrement étendu (quelques portes logiques dans un boîtier volumineux) une consommation en énergie élevée réduit l'usage de ces composants pour de petites applications simple et des circuits à densités moyenne.

Les PLDs (Programmable Logic Device en Anglais) offrent la possibilité de programmer l'architecture interne du composant, pour réaliser une fonction souhaitée par l'utilisateur et de configurer les pins d'entrées/sorties de celui-ci. En distinction par rapport aux microprocesseurs ou microcontrôleurs qui sont, eux aussi programmables mais pour qui l'on peut seulement programmer le fonctionnement selon un programme existant dans une mémoire, l'architecture interne est proposée par le fabricant tout comme les entrées/sorties **[10]**

Les PLDs ont connu une évolution technologique au fil du temps depuis la parution du premier PAL (Programmable Array Logique ou réseau logique programmable) jusqu'à l'aboutissement aux FPGAs (Field Programmable Gate Array ou réseau de cellules logiques programmables) qui sont les circuits logiques programmables les plus performant qui existent en ce moment, une évolution dû à une concurrence industrielle et scientifique **[10]** Dans ce qui suit, il sera présenté les différentes familles de circuits logiques programmables.

2.4.2 Définition des PLDs (Programmable Logic Device):

Un circuit programmable est un assemblage d'opérateurs combinatoires (les opérateurs combinatoires génériques qui interviennent dans les circuits programmables proviennent soit des mémoires (réseaux logiques) soit des fonctions standard (multiplexeurs et OU exclusif)) et de bascules dans lequel la fonction réalisée n'est pas fixée lors de la fabrication. Il contient potentiellement la possibilité de réaliser toute une classe de fonctions, plus ou moins large suivant son architecture. La programmation du circuit consiste à définir une fonction parmi toutes celles qui sont potentiellement réalisables. Comme dans toute réalisation en logique câblée, une fonction logique est définie par les interconnexions entre des opérateurs combinatoires et des bascules, et par les équations des opérateurs combinatoires. Ce qui est programmable dans un circuit concerne donc les interconnexions et les opérateurs combinatoires **[11]**.

2.4.3 Structure de base d'un PLD :

La plupart des circuits PLDs suivent la structure suivante **[11]** :

- Un bloc d'entrées qui permet de fournir au bloc combinatoire l'état de chaque entrée et de son complément.
- Un ensemble d'opérateur « ET » sur lesquels viennent se connecter les variables d'entrées et leurs compléments.
- Un ensemble d'opérateurs « OU » sur lesquels les sorties des opérateurs « ET » sont connectées.
- Un bloc de sorties.
- Un bloc d'entrées sorties.
- Un bloc d'entrées/sorties, qui comporte une porte 3 états et une broche d'entrée/sortie.

Le bloc combinatoire programmable est formé de matrices « ET » et de matrices « OU » que l'on appelle aussi somme de produits (toute fonction logique combinatoire peut être écrite comme somme de produit) [*STE05*], les interconnexions de ces matrices doivent être programmables, ceci est réalisé par des fusibles qui sont grillés lors de la programmation.

L'autre approche, radicalement opposée, « cellule universelles interconnectées » est de renoncer à la réduction en première forme normale des équations logiques. On divise le circuit en blocs logiques indépendants, interconnectés par des chemins de routage. Une fonction logique est récursivement décomposée en opérateurs simples, jusqu'à ce que les opérations élémentaires rentrent dans une cellule [*JAC00*].

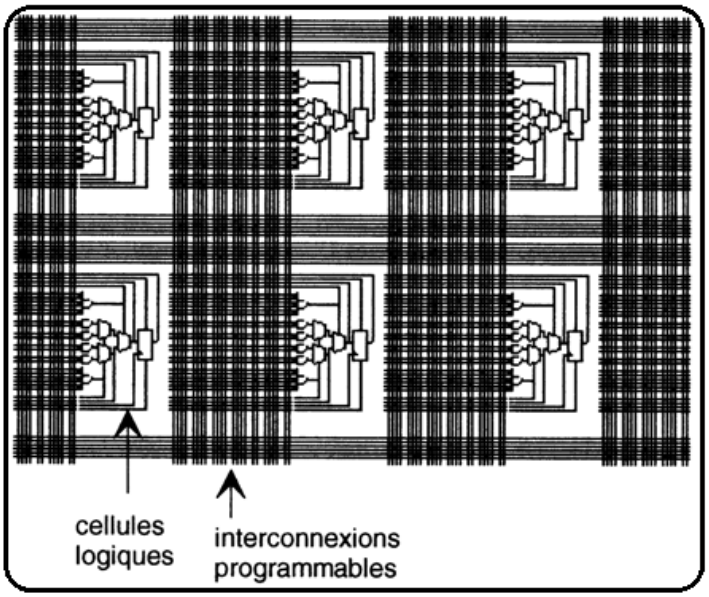


Figure 2.2 : cellules logiques interconnectées.

Le bloc de sortie est souvent appelé macro cellule, OLMC1 (Output logic macro cell, macro cellule logique de sortie). La macro cellule fut l'élément clé dans le développement des circuits

logiques programmables. En effet, la macro cellule procure au composant la flexibilité de configuration : entré, sortie, entré/sortie ou haute impédance. Plus le composant est performant plus celui-ci présente des options sur sa macro cellule [10] Celle-ci comporte :

_ Une porte « OU » exclusif, une bascule D.

_ Des multiplexeurs, qui permettent de définir différentes configurations et un dispositif de re-bouclage sur la matrice « ET ».

_ Des fusibles de configuration (pour les FPGAs, il est utilisé des cellules de commande des ponts de connexions).

Placement et routage : consiste à attacher des blocs de calcul aux opérateurs logiques d'une fonction et à choisir les broches d'entrées/sorties. Le routage consiste à créer les interconnexions nécessaires. Pour les PLDs simples, le placement est relativement trivial et le routage inexistant. Les compilateurs génériques (i.e. indépendants du fondeur) effectuent très bien ces deux opérations. Pour les CPLDs, et plus encore les FPGAs, ces deux opérations deviennent plus complexes et nécessitent un outil spécifique du fondeur **[11]**.

2.4.4 Les différentes familles de PLDs :

La classification des PLDs peut se révéler délicate et difficile, les différences de technologie se doublent de différences d'architectures. La classification suivante n'a que pour objectif de mettre en lumière de grands points de repère2. Néanmoins, on peut les classer suivant leurs structures internes à savoir : le nombre d'entrées, de sorties, de connexions programmables et le niveau d'intégration [10]

Type	Nombre de porte intégré	Matrice ET	Matrice OU	Effaçable
PAL	10 à 100	Programmable	Fixe	Non
GAL	10 à 100	Programmable	Fixe	Électriquement
EPLD	100 à 3000	Programmable	Fixe	Par UV
FPLA	2000 à 3000	Programmable	Programmable	Électriquement
FPGA	Plus de 50 000	Programmable	Programmable	

a) Les PALs (Programmable Array Logique):

Les PALs ont eu un grand succès dès leur première parution sur le marché, ce fut les premiers circuits logiques programmables3.

Un PAL est un composant relativement simple, dérivé des PROM (Programmable Read Only Mémoire, mémoire morte à lecture seule programmable une fois). Les ingénieurs de MMI ont combiné la technologie à fusibles (utilisée pour les mémoires PROM) avec des portes ET, OU pour réaliser des fonctions logiques. La compréhension de la cellule de base d'un PAL suffit car c'est la même qui se répète sur l'étendue de la capacité de celui-ci (voir la Figure I.4). La cellule de base se compose d'un buffer d'entrée qui dispose de l'information et de son complément

(tous les PALs sans exception disposent d'un certain nombre d'entrées qui aboutissent toutes, sous forme directe et inversé, sur la matrice de fusible de programmation), suivit de la matrice à fusibles puis de portes ET (considérées en entrée) puis suivit de portes OU en sortie **[10]**

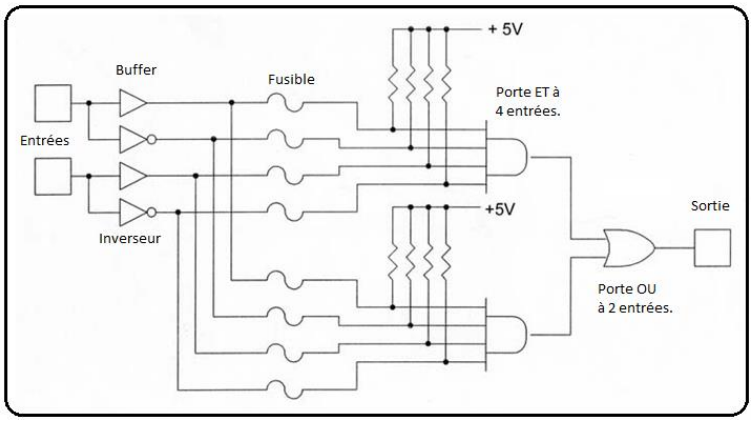


Figure 2.3 : Cellule De Base D'un PAL (Architecture Combinatoire).

La programmation d'un PAL s'effectue par la destruction de fusibles, à l'aide d'un programmateur dédié en appliquant des tensions de programmation requises.

L'inconvenant majeur des PALs, c'est qu'une fois programmée, ils ne sont plus effaçables (fusible détruit) c'est contraignant en cas d'erreur de programmation ou de mise à jour.

Les PALs existent sous 4 d'architectures (dans l'ordre de leur évolution) **[12]** :

1. Les PALs combinatoires : possède l'architecture la plus simple. Comme dans tout type de PALs, certaines broches sont dédiées uniquement aux entrées. D'autres bidirectionnelles, sont associés à un buffer de sortie trois états.

2. **Les PALs à registres :** ils disposent en sortie d'une bascule D. Tout signal de sortie passe obligatoirement par cette bascule. Pour cette raison, certains boîtiers sont proposés avec un certain nombre de sorties à registre (synchrone) et d'autres de type combinatoire.

3. **Les PALs asynchrones à registres :** Ce type de PALs constitue une variante du type évoqué précédemment. Une première différence réside dans la méthode de distribution du signal d'horloge des bascules. Contrairement aux PALs à registres, un multiplexeur permet de shunter le registre. Les deux entrées AP (PRESET Asynchrone) et AR (RESET Asynchrone) sont simultanément utilisées pour piloter ce multiplexeur. A noter la présence d'un XOR se comportant comme un inverseur programmable. Le signal de commande de ce XOR a une valeur statique, il ne s'agit ni d'un signal global ni d'un signal issu de la matrice. C'est à partir de là que le concept de la « macro cellule » programmable a fait son apparition. Cette ressource d'inversion permet de coder une fonction sans tenir compte de la polarité du signal de sortie.

4. **Les PALs versatiles (VPAL) :** ils constituent une évolution des PALs très significative. Ils proposent des macros cellules très évoluées, du type de cellules rencontrées au sein de composants plus complexes (CPLD, FPGA). Le mode de fonctionnement est obtenu par l'utilisation de deux multiplexeurs qui utilisent comme signal de sélection les points de programmation S0 et S1. En combinaison de ces deux points de programmation, on obtient différentes configurations de la sortie.

b) Les GALs (Generique Array Logic):

Les GALs ne sont rien d'autre (d'un point de vue architectural) que des PALs reprogrammables. D'un point de vu technologique au lieu d'utiliser des transistors bipolaires, ils ont utilisé des transistors MOS FET pouvant être régénérés. Cette possibilité de régénération des fusibles sans pour autant restreindre la durée de vie du composant.

Après étude des besoins du marché LATTICE SEMICONDUCTOR se fixe 4 objectifs pour la mise au point des GALs :

1. Offrir des produits ayant des vitesses de travail comparable a celle des PALs bipolaires, tout en étant testable a 100%.

2. Permettre un remplacement, au moins fonctionnel, mais idéalement broche pour broche, des PALs bipolaires dans n'importe quelle application.

3. Offrir une consommation beaucoup plus faible que les PALs bipolaires d'une complexité équivalente.

4. Proposer une plus grande souplesse de configuration des entrées/ sorties que les PALs bipolaires.

C'est ainsi que LATTICE a palier aux inconvenants majeurs des PALs pour donner naissance à un composant lui imposant une forte concurrence **[10]**

c) Les EPLDs (Erasable_Programmable_Logique_Device):

Les EPLDs sont des circuits réalisés en technologie CMOS, présentant l'avantage de faible consommation électrique, mais qui augmente en fonction de la fréquence. Ils disposent d'une macro cellule plus évolué que celles des PALs, en plus ils sont effaçables **[10]**

L'introduction des EPLDs telle que l'a voulu ALTERA visait deux buts distincts:

1. Permettre une densité d'intégration nettement supérieur à celle offerte par les PALs et aussi proche que possible que celle permise par les réseaux de portes programmable.

2. Fonctionner a une vitesse, si non égal, du moins comparable à celle des PAL bipolaires et en tout cas nettement supérieur à celle des portes traditionnels.

Les_différentes_familles_d'EPLDs:

1. **Les EPLDs classique de la série EP**. Ces circuits existent en différentes versions avec des boîtiers disposant de 20 à 68 pins. Effaçable à l'UV.

2. **Les EPLDs de la série MAX 5000**, qui ont la même fonction que leur homologue de la série EP, avec toutefois une densité d'intégration plus élevée et une architecture d'interconnexion différente, effaçable à l'UV.

3. **La série MAX 7000** plus dense que MAX 5000 en plus effaçable électriquement.

4. **La série MAX 9000** plus dense, effaçable électriquement en plus programmable sur circuit avec la tension unique de 5v.

d) Les CPLDs:

L'architecture typique d'un CPLD se présente comme un ensemble de fonctions de type PAL pouvant être interconnectées à l'aide d'une matrice. La physionomie est généralement très structurée. Un certain nombre de macros cellules de base sont regroupées pour former des blocs logiques. La complexité, le nombre de macros cellules dans un bloc ainsi le nombre de blocs varie d'un composant à l'autre. On peut considérer deux niveaux d'interconnexion : une matrice globale et un système de distribution des signaux intégrés à chaque bloc logique **[12]**.

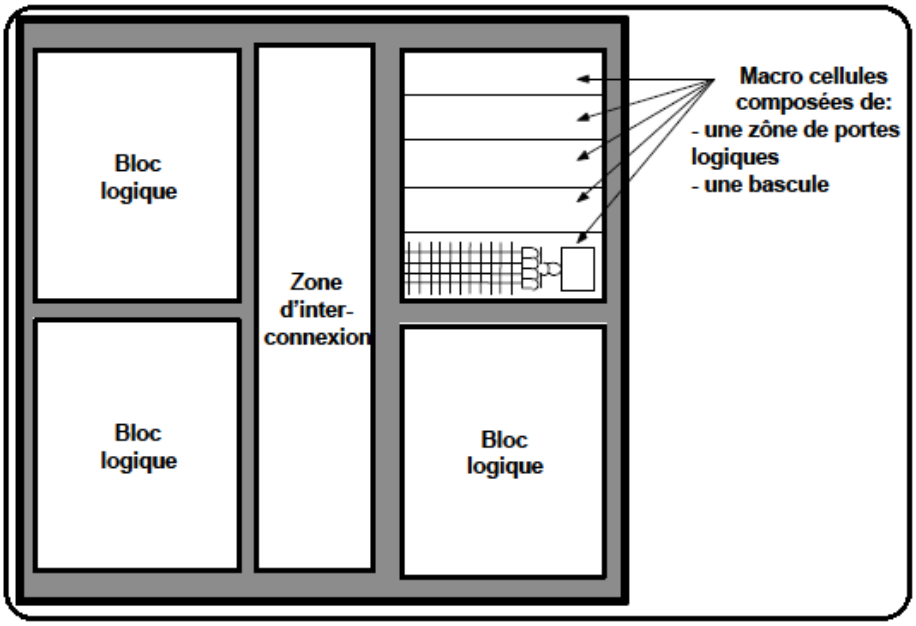


Figure 2.4 Architecture global d'un CPLD.

2.5_Les FPGA Field Programmable Gate Array_ou réseau logique programmable sur site_:

Un FPGA est un circuit logique reprogrammable. À l'aide de blocs logiques préconstruits et de ressources de routage programmables, c'est un circuit configurable afin de mettre en œuvre des fonctionnalités matérielles personnalisées, sans avoir jamais besoin d'utiliser une maquette

ou un fer à souder. Il suffit de développer des tâches de traitement numérique par logiciel et de les compiler sous forme de fichier de configuration ou de flux de bits (bit stream) contenant des informations sur la manière dont les composants doivent être reliés. En outre, les FPGAs sont totalement reconfigurables et peuvent adopter instantanément une nouvelle circuiterie si une nouvelle configuration du circuit est recompilée.

2.5.1 Étude des FPGAs :

a) Présentation_et_généralités_sur_les_FPGAs_:_

Un FPGA est un composant électronique constitué de millions voir milliers de transistors connectés ensembles pour réaliser des fonctions logiques. Des fonctions logiques simples peuvent être réalisées telles que des additions ou soustractions tout comme des fonctions complexes peuvent être réalisées telles que le filtrage numérique du signal ou détection d'erreurs et correction. Aérospatial, aviation, automobile, radar, missiles, ordinateur… ne sont que quelques exemples des domaines ayant recours aux FPGAs.

L'évolution des circuits logiques programmable, depuis la création des PALs qui présentaient l'avantage de réduire l'encombrement et de créer des fonctions logiques personnalisé. Puis vient l'étape des EPLDs qui présentent l'avantage de l'écriture électrique mais en ayant recours à un programmateur (un appareil qui permet d'injecter le routage du circuit via un fichier de programmation) mais effaçable à l'UV (ultraviolet) pour évoluer vers les CPLDs qui sont effaçable électriquement. Puis l'avènement des FPGAs qui représentent une technologie qui permet de reprogrammer le circuit in situ (c'est-à-dire sur circuit ou sur site).

L'avantage majeur que présentent les FPGAs est leur grande flexibilité. En effet, la structure interne du circuit FPGA peut être changée sans avoir à modifier la structure globale de la maquette. Cet avantage est très apprécié par les concepteurs de cartes électroniques vu que ça leurs permet de faire des prototypage rapide et de moindre coût en comparaison aux ASICs pour les quels il faut des mois pour réaliser un prototype sans avoir de certitude qu'il puisse être opérationnel en plus de ça la moindre erreur nécessite de refaire le travail depuis le début. Un coût de reviens important et une durée de développement étendu ce qui doit être minimisé dans les milieux industriels. Il y aussi l'avantage de la mise à jour du circuit face à des bugs ou l'ajout de nouvelles fonctionnalités [13].

Xilinx, Altera et Quicklogic sont les pionniers dans le domaine des FPGAs, et plusieurs autres compagnies produisent les FPGAs. Toutes ces compagnies se partagent le même concept architectural4. [13]. Il se divise en trois parties : Interfaces d'entrées/Sorties (I/O interface), les blocs logiques de base (Basic Logic Building Blocks) et les interconnexions (voir figure 2.5).

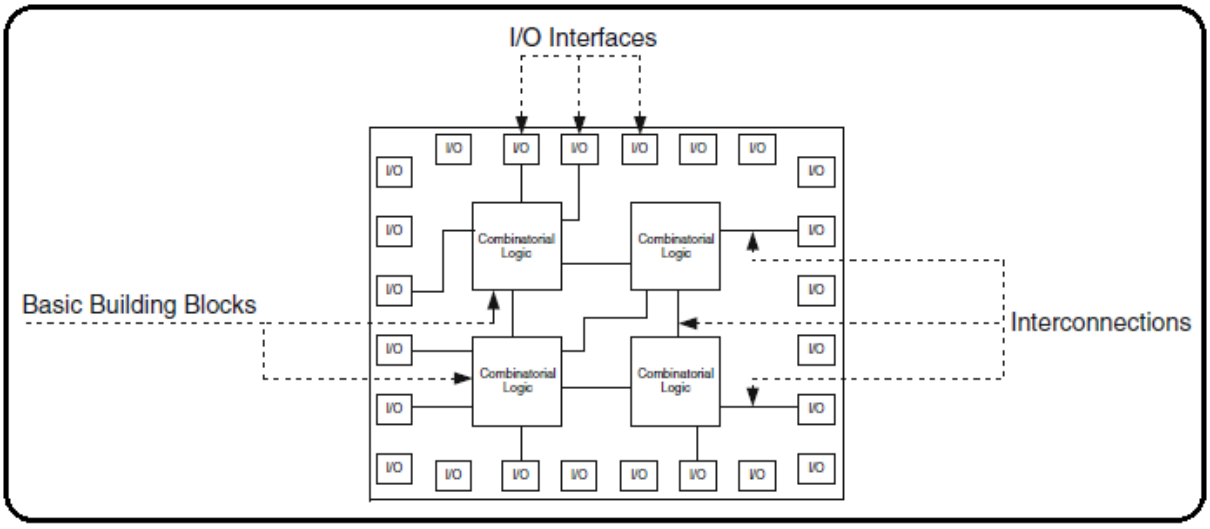


Figure 2.5 : Concept architectural de base des FPGAS .

b) Interfaces_d'entrés/Sorties (I/O_interfaces):

Les interfaces d'entrées/Sorties se présentent comme les intermédiaires par lesquelles les données transitent depuis les blocs logiques internes jusqu'aux ressources externes et vice versa. L'interfaçage des signaux peut être : unidirectionnel, bidirectionnel, à deux états ou trois états (0, 1 ou haute impédance) voir même obéir à un standard d'entrées/sorties. Voici quelques-uns de ces standards :

_ GTL (gunning transceiver logic).

_ HSTL (high-speed transceiver logic).

_ LVCMOS (low-voltage CMOS).

_ LVTTL (low-voltage transistor-transistor logic).

_ PCI (peripheral component interconnect)…

Le rôle principal des interfaces d'entrées/sorties est de transmettre et de recevoir des données.

Néanmoins l'interface d'entré/sorties peut être dotée d'options telles que des registres, impédances et buffers [GIN10].

Chaque fabricant a sa propre appellation pour désigner l'interface d'entrées/sorties mais la fonction reste toujours la même.

Altera, les nomme IOE Input Output Element. L'IOE remplie toujours son rôle d'interface d'entrées/sorties, elle dispose d'une résistance de rappel pull-up et un temporisateur du signal.

Chez Xilinx, les interfaces d'entrées/sorties sont nommées IOB pour Input Output Blocks.

L'IOB est constitué de registres, de diviseurs de tensions, des résistances de rappel pull up et autres ressources spécifiques. La Figure 1.6 est un exemple d'IOB simplifié de Xilinx.

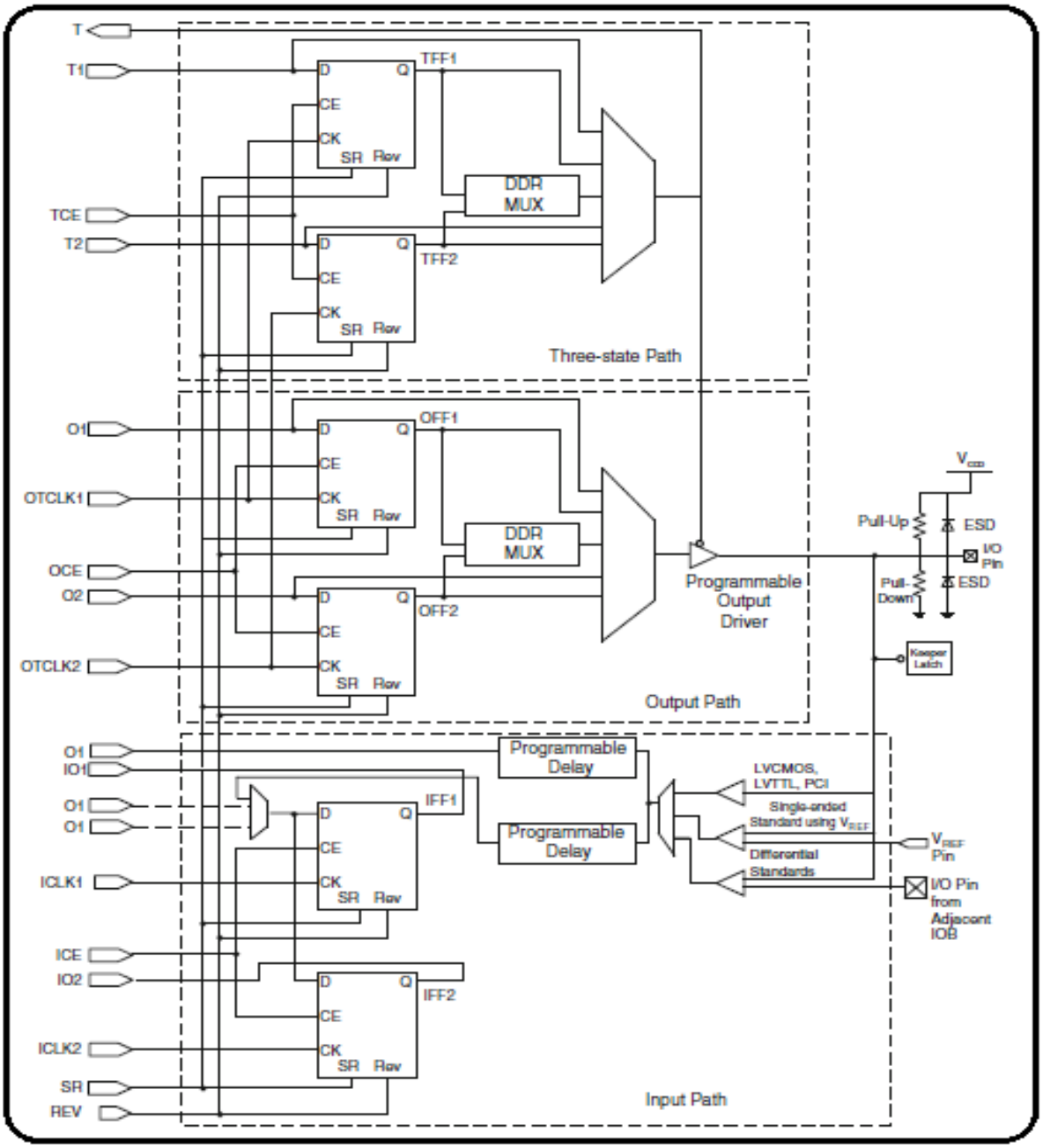


Figure 2.7 : structure simplifiée d'un IOB.

***c)* Les_blocs_logiques_de_base_(basic_logic_building_blocks):**

Les blocs logiques de base, sont des blocs logiques configurés ou optimisés de sorte à réaliser des fonctions logiques.

Altera, nomme les blocs logiques de base ALM Adaptative Logique Module. L'ALM est constitué d'éléments de logique combinatoire, des registres et des additionneurs. L'élément de la logique combinatoire est constitué de 8 entrés et d'un LUT Look Up Table **[13].** Voir la figure 2.8.

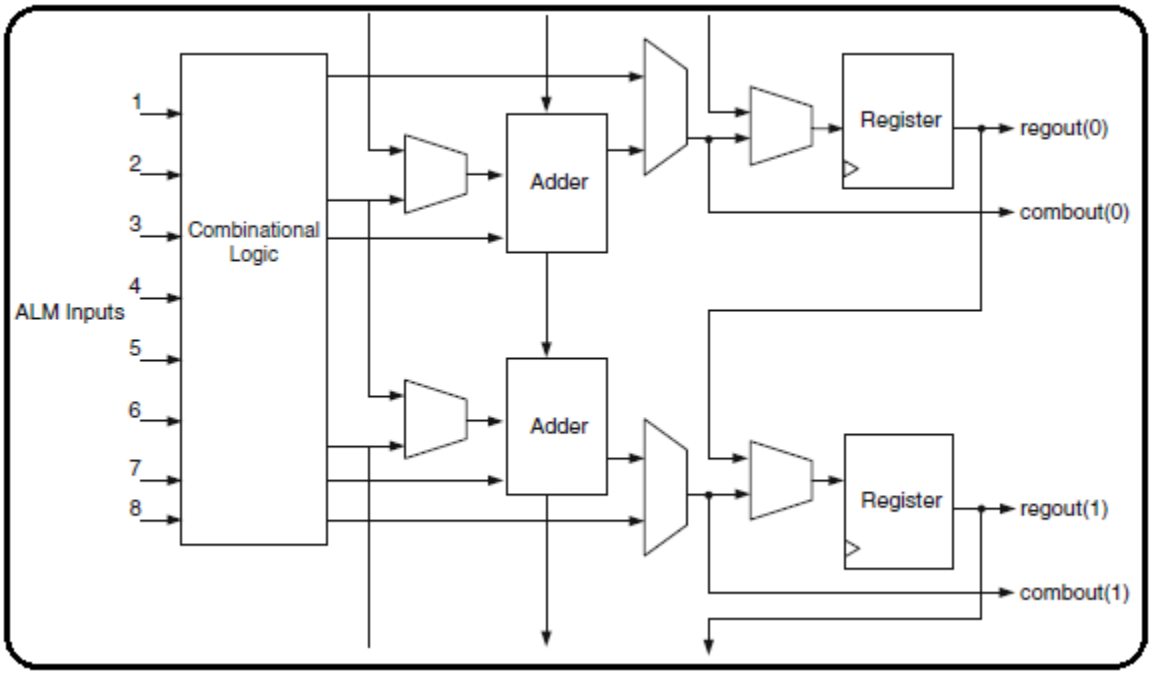


Figure 2.8 : Constitution interne d'un ALM Adaptative Logic Module.

Xilinx, nomme les blocs logiques de base CLB Configurable Logic Blocks. Chaque CLB contient des slices et chaque slice a des LUTs Lookup Tables. Voir la figure 2.9.

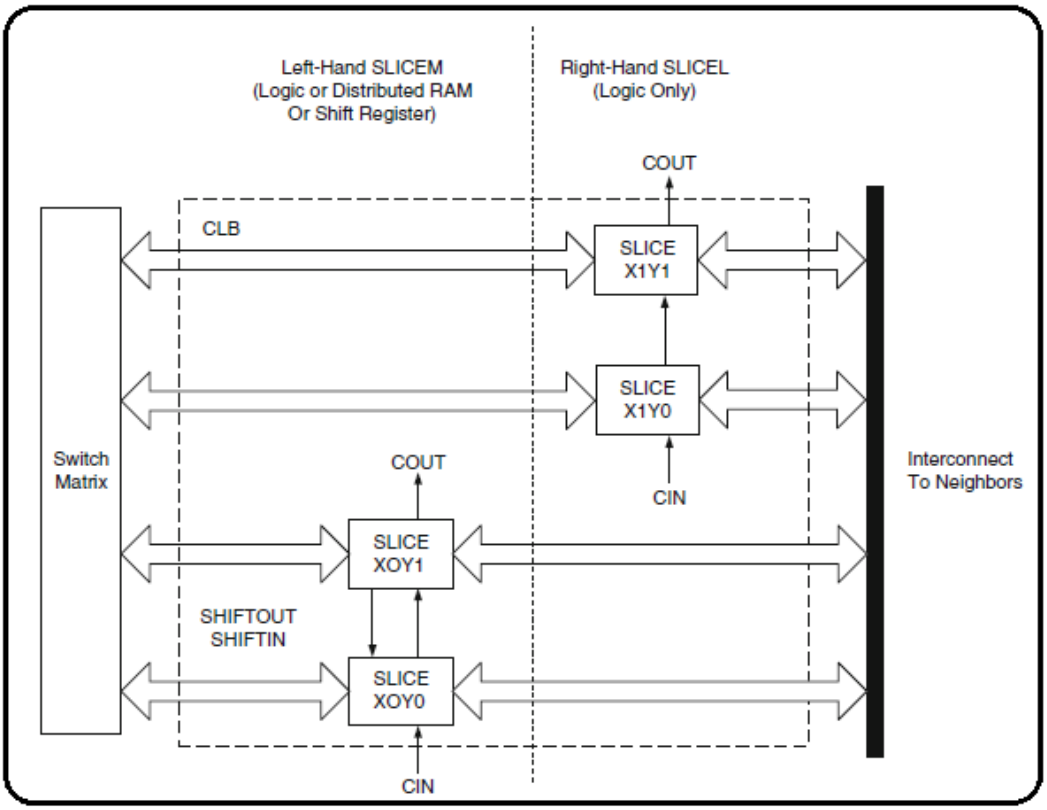


Figure 2.9 : Constitution d'un CLB configurable logic block et la disposition des slices (famille Virtex).

Chaque fabricant défini ses propres structure des blocs logiques de base et le nombre de blocs disponibles. Ceci est dû au fait que les FPGAs sont utilisés dans des applications variées et il n'y a pas un seul composant FPGA pour toutes les applications. Ceux qui sont utilisés en aérospatial ne sont pas prédestinés à réaliser les mêmes fonctions ni à opérer dans les mêmes conditions que ceux utilisés en automobile ou bien en téléphonie mobile **[13]**.

Le choix du bon FPGA est simplifié par la classification adoptée par les fabricants, Xilinx s'est organisé en familles au moment où Altéra s'est organisé en séries. Chaque famille ou série est constituée de membres ou sous familles. Les membres d'une même famille se partagent les mêmes caractéristiques de base et chaque membre se distingue des autres par ses propres caractéristiques à savoir la mémoire, les ressources disponibles ou le nombre d'entrées/sorties.

Xilinx présente plusieurs familles de FPGAs dont voici quelques-unes : Extended SpartanW3A, Spartan-3E, Spartan 6, Virtex, Virtex-E, Virtex-Pro, Virtex 5, et Virtex 6. Par exemple Spartan-3^E est une famille qui comporte 5 membres (voir le tableau suivant) le numéro qui suit le S dans la référence, représente 1 pour 1000 portes. Ce qui est pratique, d'un simple regard sur la référence le nombre de portes que le composant comporte est déduit [GIN10].

2.5.2. Les langages de description matérielle HDL (Hardware Description Langages)

2.5.2.1 Évolution des HDL :

La description matérielle a connu une évolution au fur et à mesure que les moyens informatiques et les technologies d'intégrations des transistors (de plus en plus denses) évoluaient. Les langages de description Matérielle ont connu une évolution en quatre étapes pour arriver à maturité :

Première étape : dessin au micron.

Au début, la conception de circuits intégrés était manuelle. On dessinait les composants (transistors) à la main, sur un papier spécial (Mylar) avec des crayons de couleur. C'est le dessin au micron. Une telle technique limitait la complexité des dispositifs conçus.

Deuxième étape : les langages de description.

Plus la technologie microélectronique évoluait vers l'intégration de nombres plus important de transistors, plus la nécessité de nouveaux outils de conception s'imposait c'est ainsi que les langages de description matériel (HDL) ont fait leurs apparitions. Ils avaient pour but de modéliser, donc de simuler, mais aussi de concevoir. Des outils informatiques (placeurs routeurs) permettant de traduire automatiquement une description textuelle en dessins. Le concepteur aborde les problèmes à un niveau d'abstraction plus élevé. Il manipule des objets élémentaires de l'ordre de la dizaine de transistors : les portes logiques.

Troisième étape : les schémas.

L'apparition des interfaces graphiques et donc des éditeurs de schémas simplifie le travail de conception de circuits intégrés. En effet : il est en général beaucoup plus facile de lire et de comprendre un schéma qu'une description textuelle. La description textuelle est remplacée par une description schématique**.**

Quatrième étape : l'abstraction fonctionnelle

Les langages fonctionnels (ou comportementaux) de description de matériel, grâce aux nouvelles possibilités de description à un niveau d'abstraction plus élevé, ont répondu à des besoins fondamentaux des concepteurs de circuits intégrés :

_ La réduction des temps de conception.

_ L'accélération des simulations qui devenaient prohibitives avec l'accroissement de la complexité des dispositifs.

_ La normalisation des échanges : des langages normalisés et universellement reconnus servent aux échanges entre partenaires industriels, entre fournisseurs et clients. Verilog, VHDL et SystemC en sont des exemples.

_ L'anticipation : grâce aux modèles HDL il est possible de concevoir un système alors que ses composants ne sont pas encore disponibles.

_ La fiabilité : les langages HDL sont conçus pour limiter en principe les risques d'erreur.

_ La portabilité : les langages normalisés sont très largement portables.

_ La maintenabilité et la réutilisabilité : les modifications et adaptations sont rendues plus simples donc moins risquées et moins coûteuses.

L'augmentation du nombre de portes intégrées (plusieurs millions de portes) sur une même puce fait apparaître la nécessité de développer des outils encore plus puissants : les synthétiseurs logiques. Le dispositif à modéliser ou à concevoir sera représenté par sa fonction et non plus par sa structure. C'est le synthétiseur logique qui déterminera la structure automatiquement à partir de la fonction [].

2.5.2.2 Utilité des HDL :

Les langages HDL offrent deux avantages majeurs en plus de la simplicité lors de la conception :

1. **Simulation :** Le but de la modélisation, c'est la simulation. Il existe deux points importants pour la simulation :

_ La fidélité : un modèle se doit d'être aussi précis que possible dans son champ d'application.

_ L'efficacité : le modèle doit pouvoir être simulé le plus rapidement possible et doit être portable, réutilisable et facile à maintenir.

Pour simuler un modèle, il faut disposer du modèle, bien sûr, mais aussi de stimuli, c'est-à-dire de la description des signaux d'entrées du modèle au cours du temps. Ces stimuli sont appelés les TESTSBENCHES en Anglais.

Les langages fonctionnels de description de matériel permettent de simplifier la conception en analysant automatiquement les résultats en cours de simulation.

Un environnement de simulation complet comprend : un générateur de stimuli, un modèle de l'objet à simuler et un vérificateur automatique des résultats. Ces trois composantes sont modélisées à l'aide du même langage.

2._Synthèse : Les langages fonctionnels de description matérielle servent aussi à concevoir. Il ne s'agit plus de modéliser en vue de la simulation, mais de décrire les objets qui seront véritablement fabriqués. Si les considérations de vitesse d'exécution en simulation existent toujours (la description sera simulée avant d'alimenter le synthétiseur, afin de vérifier que la fonction décrite est bien la fonction désirée) elles ne sont plus prioritaires. Ce qui compte le plus, c'est l'efficacité du code au sens du synthétiseur. En effet, pour que ce dernier produise la description structurelle la plus économique possible (et donc la surface de silicium la plus petite possible). **[14]**

2.5.2.3 Exemples de langages de description matérielle HDL :

Il existe un bon nombre de langages HDL, malgré leurs diversités, ils répondent tous aux objectifs fixés par les grandes lignes des langages HDL. Parmi eux: VHDL, Verilog, SysremC, SysremVerilog…etc.

a) VHDL (Very High Speed Integrated Circuits Hardware Description Language):

À l'origine, avant même la mise en place du VHDL, le programme VHSIC (Very High Speed Integrated Circuits), impulsé par le département de la défense des états unis dans les années 1970-1980, a donné naissance à un langage : VHSIC-HDL ou VHDL. Deux normes successives (IEEE-1976-1987 et 1993) en ont stabilisé la définition, complété par des extensions plus récentes ou a venir. L'évolution prévue est la création d'un langage commun analogique – numérique, dont le VHDL, sous sa forme actuelle, constituerait la facette numérique6 [13].

b) Verilog :

Verilog a été inventé par Gateway Design Automation Inc. aux alentours de 1984. C'était un langage propriétaire. Conçu à l'origine pour être utilisé dans leurs simulateurs logiques, mais le succès grandissant du VHDL a incité ses concepteurs de faire de Verilog un standard ouvert ; c'est le standard IEEE 1364 dont il existe plusieurs versions, qui ont été enrichies pour offrir des

fonctions équivalentes à celles de VHDL. La syntaxe de Verilog est réputée largement inspirée du langage de programmation C, bien que la ressemblance se limite aux expressions. Ceci explique en partie son succès et sa diffusion rapide dans la communauté des ingénieurs qui ont déjà appris le langage C **[13]**.

c) Comparaison entre VHDL et VERILOG et choix :

En comparaison entre VHDL et VERILOG, ils sont conçus pour répondre aux mêmes exigences des descriptions HDL, mais il se trouve que ces deux langages ont été conçus chacun en donnant plus d'importance à un aspect plus tôt qu'un autre d'une description matériel. En effet, le langage VHDL se base plus sur le fait d'obtenir le circuit le plus optimisé que possible donc le moins encombrant dans le souci d'embarquer le plus de fonctionnalités dans un circuit logique programmable. Ce qui n'est pas sans conséquence car le code se voit plus long que celui rédigé en langage VERILOG pour une même fonction. Par contre le langage VERILOG cherche à réduire le temps de mise en place d'un code opérationnel donc le plus court possible mais en réduisant les dimensions du code la conséquence, le circuit synthétisé consomme plus de ressources. Ce qui peut apparaitre comme une contrainte, mais en réalité un circuit logique programmable remplie à 50% donne satisfaction pour les concepteurs et celui rempli à 80% **[12]**est très optimisé .

d) Le langage de description matériel VHDL (Very High Speed Integrated Circuits Hardware Description Language):

Le VHDL est un langage de description matériel HDL. Il répond à tous les critères établis pour un langage HDL (voir paragraphe 2.1 Quatrième étape).

Les plus importants points forts de ce langage sont **[12]** :

Simulation,_ synthèse: le VHDL permet d'avoir des lignes de code pouvant être simulé (simulation fonctionnelle) dans le but de vérifier si le code obéit correctement à la fonction souhaitée mais pour parvenir à une maquette réalisable ce n'est pas suffisant. D'où la nécessité de pouvoir synthétiser le même code. Lors de la synthèse, le compilateur traduit le premier code en un autre équivalent mais à un niveau d'abstraction plus bas. À ce niveau de synthèse, le compilateur traduit le code de haut niveau en portes logiques ceci en fonction du circuit logique programmable qui est ciblé. Le fichier synthétisé lui aussi, se doit d'être simulé (simulation événementielle) et vérifier si le compilateur (synthétiseur) est resté fidèle aux exigences de départ. Puis le fichier est compilé une deuxième fois pour aboutir au circuit logique qui sera implémenté. Cette fois encore, une troisième simulation (facultative) (simulation temporelle) pour être certain que le circuit est resté inchangé.

Portabilité : avec deux objectifs, portable vis-à-vis du circuit logique programmable, c'est-à-dire peut être implémenté sur l'importe quel circuit logique programmable à condition d'avoir la

capacité logique requise. Portable vis-à-vis du compilateur, pouvoir passer d'un compilateur à un autre et obtenir un même circuit en fin de processus.

Une construction hiérarchique : cette approche permet de simplifier la conception puis ce que les taches peuvent être divisées pour aboutir au point le plus simple que possible puis faire un assemblage des blocs.

Une description fonctionnelle : complémentaire de la précédente, la vision fonctionnelle apporte la puissance des langages de programmation. Tout algorithme est la description interne d'un bloc situé quelque part dans la hiérarchie du schéma complet. La vision structurelle est concurrente, la vision algorithmique est séquentielle, au sens informatique du terme [JAC00]. Un programma VHDL doit être compris comme l'assemblage en parallèle des taches indépendantes qui s'exécutent concurremment.

2.5.2.4 Structure d'un programme VHDL le_couple entity , architecture:

Un opérateur élémentaire, un circuit intégré, une carte électronique ou un système complet, est complètement défini par des signaux d'entrées et de sorties et par la fonction réalisée de façon interne **[12]**. Les concepteurs du VHDL ont adopté l'approche suivante : l'importe quel système est considéré comme une boite noire. Cette boite noire a des entrées et des sorties. Ils ont appelé la boîte noire « ENTITY ». L'importe quel système électronique effectue des opérations sur le signal d'entré pour donner le résultat du traitement en sortie.

Ces opérations sont le contenu de la boîte noire ce contenu est appelé « ARCHITECTURE ».

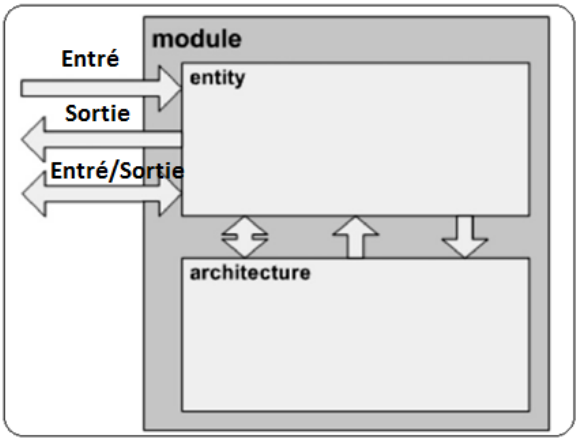


Figure 2.10 : structure de base d'un module sous VHDL .

Entity ou Entité: Dans la partie déclarative de l'entité le circuit est décrit comment il est vu par l'extérieur ceci à travers les entrées, sorties et entrées/sorties.

```
Entity <entity name> Is Port(
   <signal name   : <signal direction> <data type>);
End <entity name>;
```

Figure 2.11 : Syntaxe déclarative de l'entité.

Architecture : l'architecture décrit le comportement que doit avoir le circuit ou les opérations qu'il doit effectuer. Une architecture se doit toujours d'être attachée à une entité (ils vont de pair).

```
Architecture <architecture name> Of <entity name> Is
<Define signals and constants>
Begin
End <architecture name>;
```

Figure 2.12 : Syntaxe déclarative de l'architecture.

C'est dans cette section que le programme est rédigé. Un programme comporte essentiellement les éléments suivants : les signaux internes, opérateurs logiques (synchrone ou concurrent), les process [*JAC11*]. La description d'une architecture peut prendre trois formes :

Description comportementale : Ce type de description spécifie le comportement du composant ou du circuit à réaliser au moyen d'instructions séquentielles ou sous forme de flot de données (constituant un process).

Description structurelle : Dans ce type de description, les interconnexions des composants préalablement décrits sont énoncées. Cette description est la transcription directe d'un schéma.

Description mixte : Elle regroupe les deux descriptions décrites précédemment. À chaque entité peut être associée à une ou plusieurs architectures mais au moment de l'exécution (simulation, synthèse…) seulement une architecture et une seule est utilisée.

Synthèse

La description HDL du circuit est transformé en un assemblage (Netlist) de primitives de base (portes logiques, bascules …). Après la synthèse on peut visualiser le niveau RTL comme le montre la figure 2.13

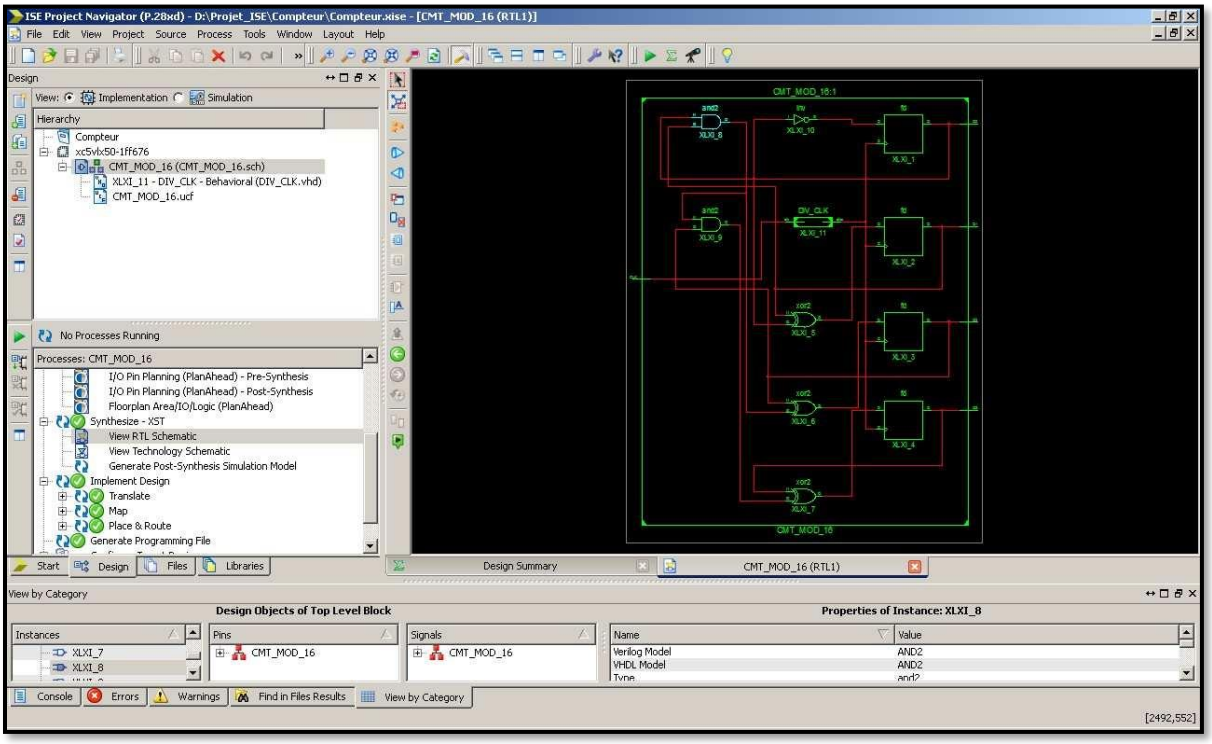


Figure 2.13 : schéma RTL

Simulation fonctionnelle avec Isim

Une fois l'étape de conception terminée, on passe à l'étape de simulation fonctionnelle du projet à l'aide d'un testbench et du simulateur Isim. Pour cela on génère:

_ Un fichier testbench VHDL à l'aide du navigateur ISE.

_ Une fois appelé par ISE, Isim prendra en charge la simulation et la présentation des résultats.

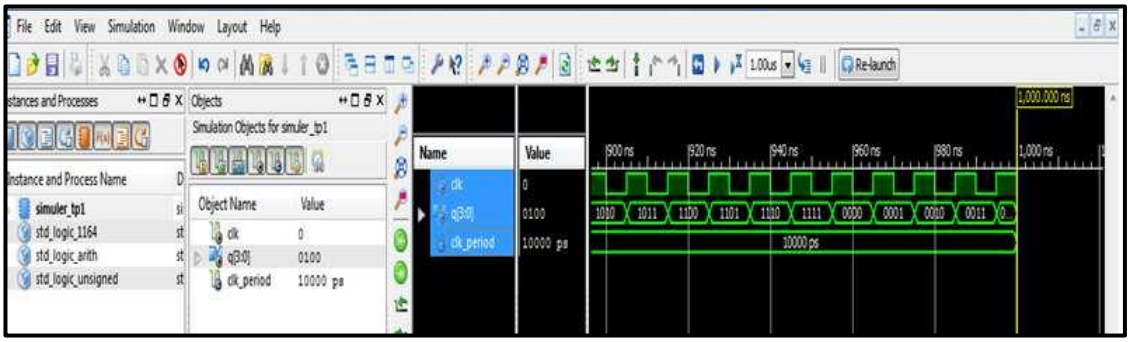


Figure (2.14) : Résultat de simulation

2-5 Conclusion:

En a découvrir dans ce chapitre les systèmes électroniques embarqués de côté hard et soft, en voir aussi les caractéristiques, l'utilité, les points faible–fort de différentes type de ces systèmes, aussi en a étude les circuits logique et l'FPGA et définir quelque notion fondamentaux comme : PLD, CPLD.

3-1 Introduction:

L'FPGA est une technique très performante utilisée par les développeurs des systèmes électroniques pour concevoir , implémenter des solutions matérielles uniques sans avoir à développer des dispositifs silicium personnalisés , Xilinx est l'une des grands fabricants des FPGA standard qui vendent vierges ou non programmées aux clients, ensuite ces appareils programmés aux clients pour mettre en œuvre leurs propres systèmes ,Si une fonctionnalité change ou si un bogue est découvert , tout simplement l'utilisateur capable de charger un nouveau programme sur l'FPGA ou mettre à jour l'ancien.
Cette technique révolutionnaire a une incidence sur le cycle de développement des produits électroniques pour presque tous les appareils électroniques depuis son introduction à la fin des années 1980.

Dans ce chapitre en doit commencer par découvrir les différents outils de développement d'une application sous Xilinx puis en doit utiliser l'un de ces outils pour implémenter l'algorithme de Horn et Schunck, en va passer étape par étape jusqu'en obtenir la résultat final.

3-2 Outil de développement Xilinx :

- **ISE** (Integrated System Environment) **Deign Suite :**

Est un outil logiciel produit par Xilinx pour la synthèse et l'analyse des HDL conceptions, ce qui permet au développeur de compiler leurs conceptions effectuées l'analyse temporelle, configurer la périphérique cible avec le programme.

- **Vivado Design Suite :**

Contient des services qui prennent en charge tous les phases de la conception FPGA : à partir de la saisie de la conception, la simulation, la synthèse, de la localisation et de routage, de la génération de flux binaires, du débogage et de la vérification, ainsi que du développement de logiciels ciblés pour ces FPGA.

- **Vivado Hls** (High-Level Synthesis):

Le compilateur Vivado High-Level Synthesis permet aux programmes C, C ++ et SystemC d'être directement ciblés sur les périphériques Xilinx sans qu'il soit nécessaire de créer manuellement une RTL. Vivado HLS a été largement évalué pour augmenter la productivité des développeurs et il est confirmé qu'il prend en charge les classes, les modèles, les fonctions et la surcharge d'opérateur C ++.Vivado 2014.1 a introduit la prise en charge de la conversion automatique des noyaux OpenCL en IP pour les périphériques Xilinx. Les noyaux OpenCL sont des programmes qui s'exécutent sur diverses plates-formes de processeur, **de GPU et de FPGA.**

- **System Generator :**

Différentes de tout l'autre outil Xilinx en ce qui il intègre dans un environnement privé appelé SIMULINK. L'accès à l'outil est fourni via un catalogue de blocs disponibles dans le navigateur de la bibliothèque Simulink.

Tous les blocs système générateur peuvent être distingués des blocs Simulink grâce à la présence du logo Xilinx.

- **Vivado Ip Integrator :**

Composé d'une interface graphique appelée bloc design pour sélectionner les adresses IP de périphérique configurer les paramètres matériels et assembler les blocs IP pour créer le système numérique.

3-3 Implémentation d'algorithme de Horn Et Shunk dans une FPGA :

3-3-1 Hypothèse de travail :

Tout d abord en va faire une simulation Simulink (mathWorks) et sauvegarder les résultats ensuite en utiliser system generator (Xilinx) qui relie auparavant avec Simulink, la dernière étape est comparée entre les résultats obtenir de Simulink et l'autre de XSG.

Les étapes à suivre sont :

- **1 ère étape :**

Prends 2 images successives (on suppose qui on a obtenu ces 2 images à partir d'une séquence d'images qui passe par un retardateur) et fait un traitement primaire pour obtenir 2 vecteur prés pour la étape suivante.

- **2 émé étape :**

Regroupe l'ensemble des blocs qui fait une partie de l'opération (gardian comme exemple) en sous-système puis étudier ce sous-système d'une manière indépendant comme montrer dans la figure 3-1 après en va faire la simulation avec Simulink et XSG.

- **3 émé étape :**

Compare entre les résultats de Simulink et de XSG.

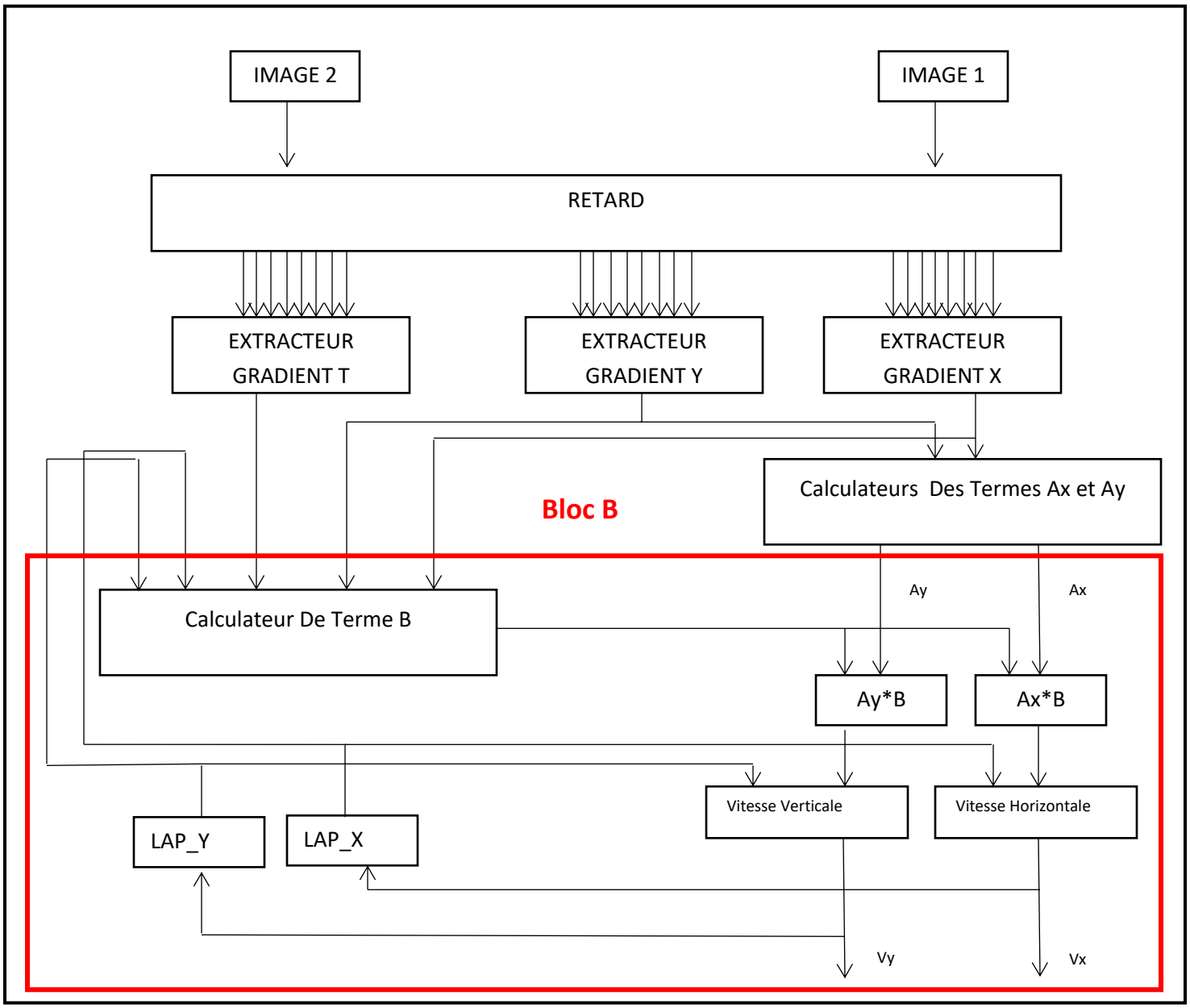


Figure 3-1 Schéma fonctionnel de l'estimateur de flot optique

Pour éviter la boucle du nombre d'itérations, une solution a été proposée (Pipeline) , qui est de reprendre en cascade le bloc en rouge un certain nombre de fois (par exemple 5 fois) en prenant comme condition initiale des deux vecteurs vitesse nulles (Vx=0, Vy = 0).

3-3-2 L'Implémentation :

- **Traitements primaires des images :**

  - **Redimensionnement (resize) ou la mise à l'échelle :**

Est une transformation applicable à une image numérique qui consiste à modifier la taille, qui ce soit pour l'agrandir ou pour la rétrécir.

En fait cette opération pour obtenir une image sous forme d'un carré 128*128.

Computer Vision System Toolbox ⟶ Geometric Transformations ⟶ Resize

- **Convertir l'image en vecteur (reshape 2D-1D) :**

DSP system toolbox ⟶ Signal Management ⟶ Signal Attribues ⟶ Convert 2-D To 1d

- **Le Bloc frame conversion :**

Transforme l'entrée à la sortie et définit le mode d'échantillonnage du signal de sortie qui peut être basée sur une image ou sur un échantillon.

DSP system toolbox ⟶ signal management ⟶ signal attribues ⟶ Frame conversation

- **Le Bloc Unbuffer :**

Les entrées ne sont pas rangées dans mémoire-tampon de sorte que chaque ligne de la matrice devienne un échantillon temporel indépendant dans la sortie.

DSP system toolbox ⟶ signal management ⟶ buffer

La figure 3-2 montre le traitement primaire

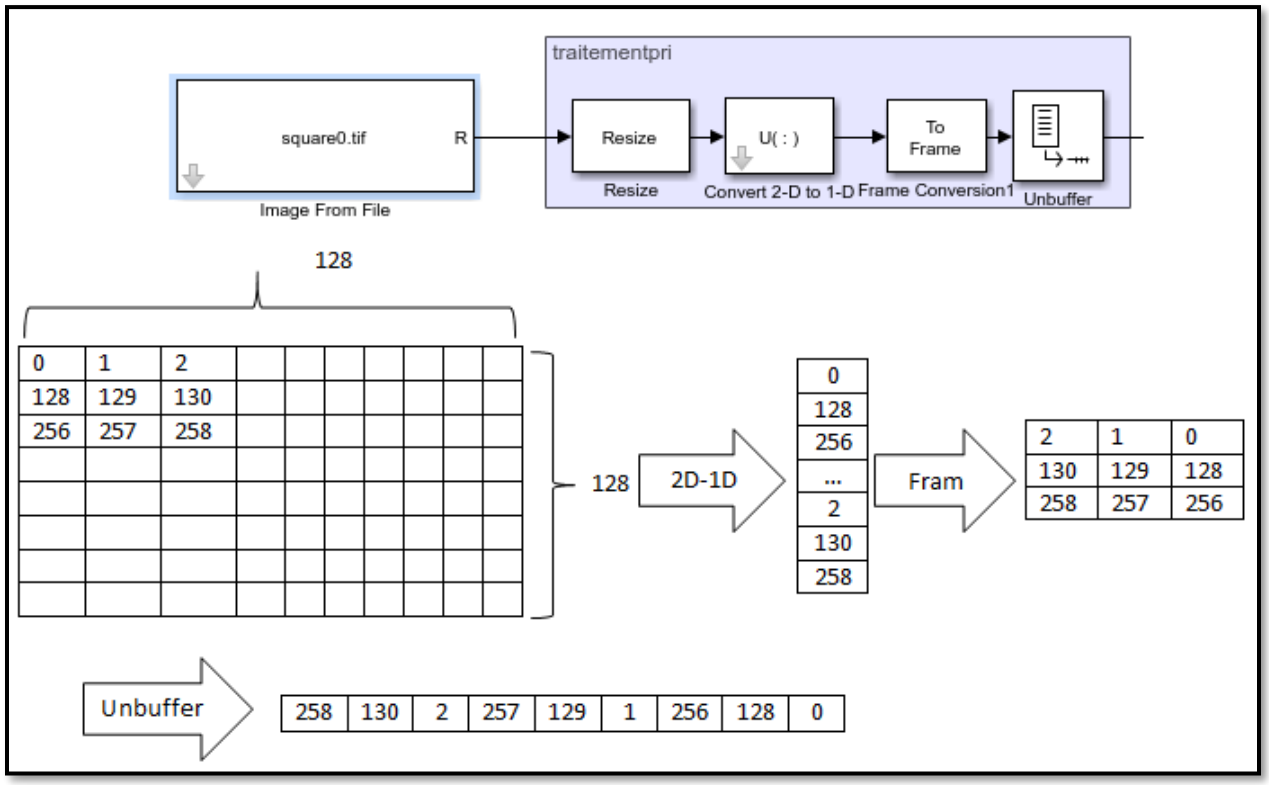


Figure 3-2 traitements primaires des images

- **Block retard :**

à la fin du traitement primaire des images on obtient vecteur de 16384 éléments qui est le nombre de pixels pour chaque image et pour calculer le gradient horizontal, vertical, temporelle on a besoin de 4 pixels d'image actuelle et précédente.

Pour extraire ces 4 pixels en utilise bloc retard composé de plusieurs durées, la première durée est le temps de traitement d 1 seul pixel, la 2 émé durée est le temps nécessaire pour traitement d'un 1 line (128 éléments), le schéma au-dessous montre comment ce bloc ça fonctionne

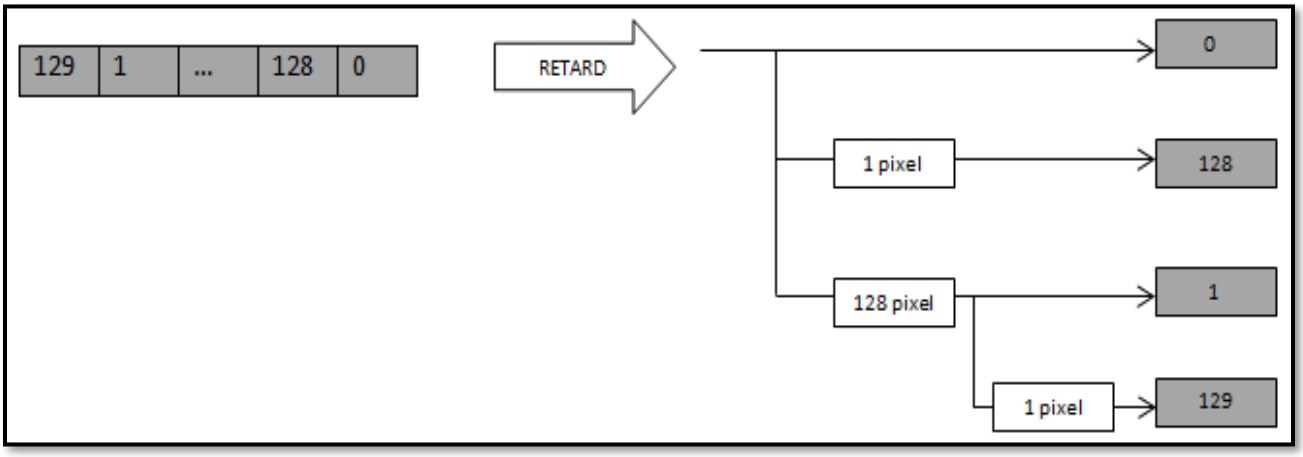


Figure 3-3 Fonctionnement de bloc retard

- **Bloc retard avec simulink:**

La figure 3-4 montre la réalisation de bloc retard avec simulink :

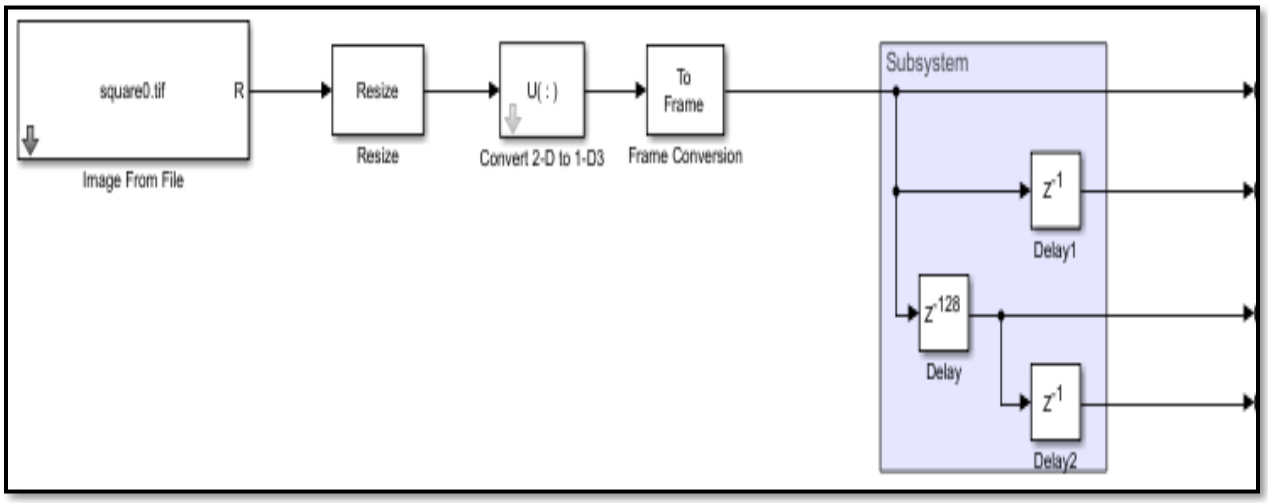


Figure 3-4 Bloc retard avec simulink

- **Block retard avec XSG :**

Pour réaliser ce bloc avec XSG en utilise la bibliothèque Xilinx Blockset

Générateur du système : Xilinx Blockset ⟶Basic Eléments ⟶System Generator.

Port entrée: Xilinx Blockset ⟶Basic elements ⟶Gateway in.

Durée: Xilinx Blockset ⟶Basic elements ⟶Delay.

La figure 3-5 montre le bloc retard avec XSG

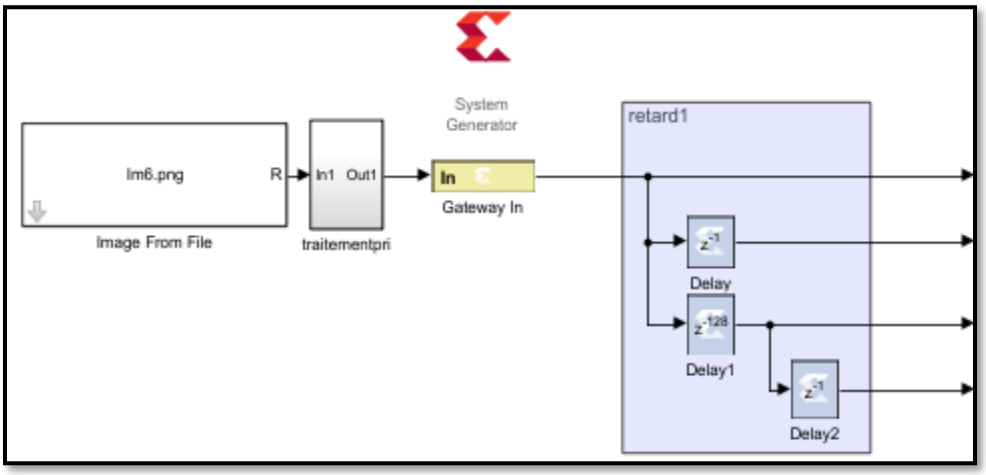


Figure 3-5 bloc retard avec XSG

➢ **Gradient spatio-temporel :**

- **Gradient Horizontal :**

Le gradient horizontal **Ix** donne par les relations suivantes :

Ix = conv2 (im1, kx, 'same') + conv2 (im2, kx, 'same')

La Figure 3-6 donne la convolution d'une image B avec un noyau Y

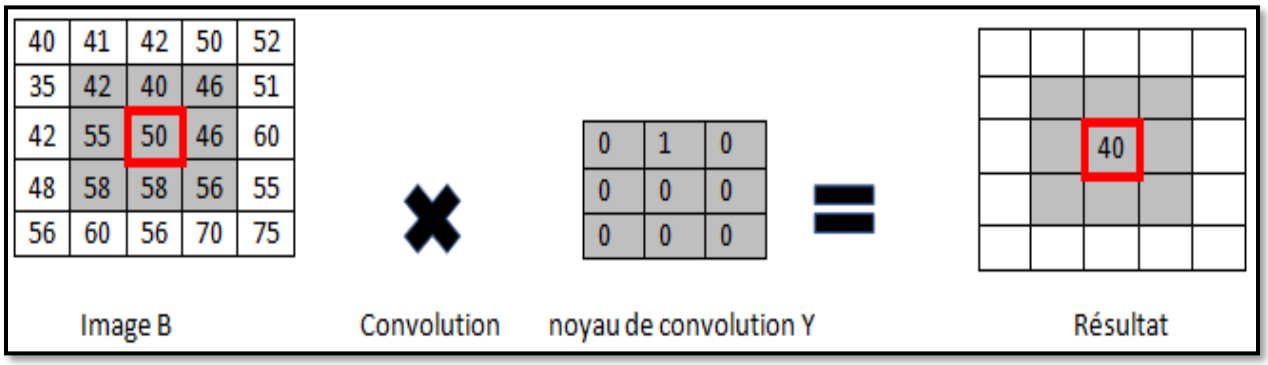


Figure 3-6 Convolution d'une image B avec un noyau Y

Pour gradient Horizontal en doit utiliser noyau de convolution : **Kx** =

-1/4	**1/4**
-1/4	**1/4**

- **Gradient Horizontal avec simulink:**

La figure 3-7 montre la réalisation de gradient horizontal avec simulink

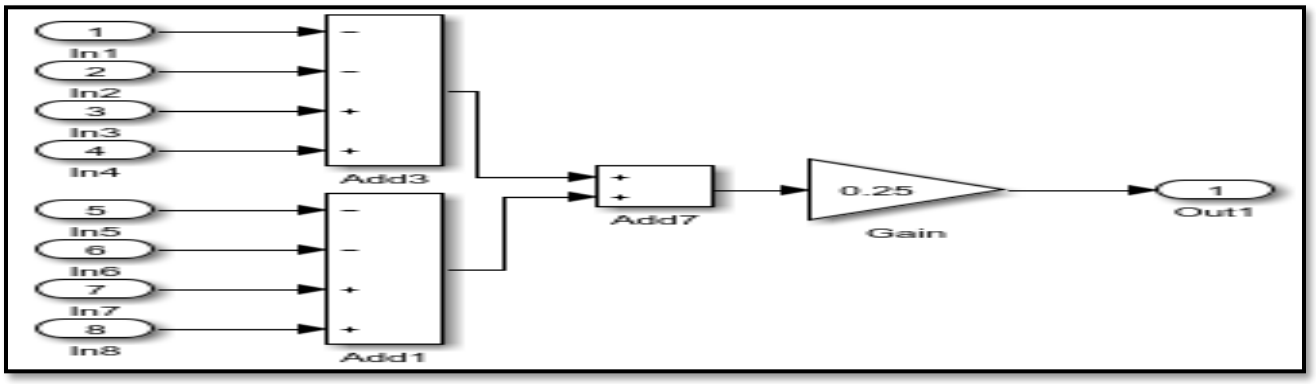


Figure 3-7 Gradient Horizontal avec simulink

- **Gradient Horizontal avec XSG :**

Pour réaliser ce bloc en utilise :

Addition /Soustraction: Xilinx Blockset ⟶ Math ⟶ Addsub.

Multiplication avec constant : Xilinx Blockset ⟶ Math ⟶ CMult.

La figure 3-8 montre le Gradient Horizontal avec XSG

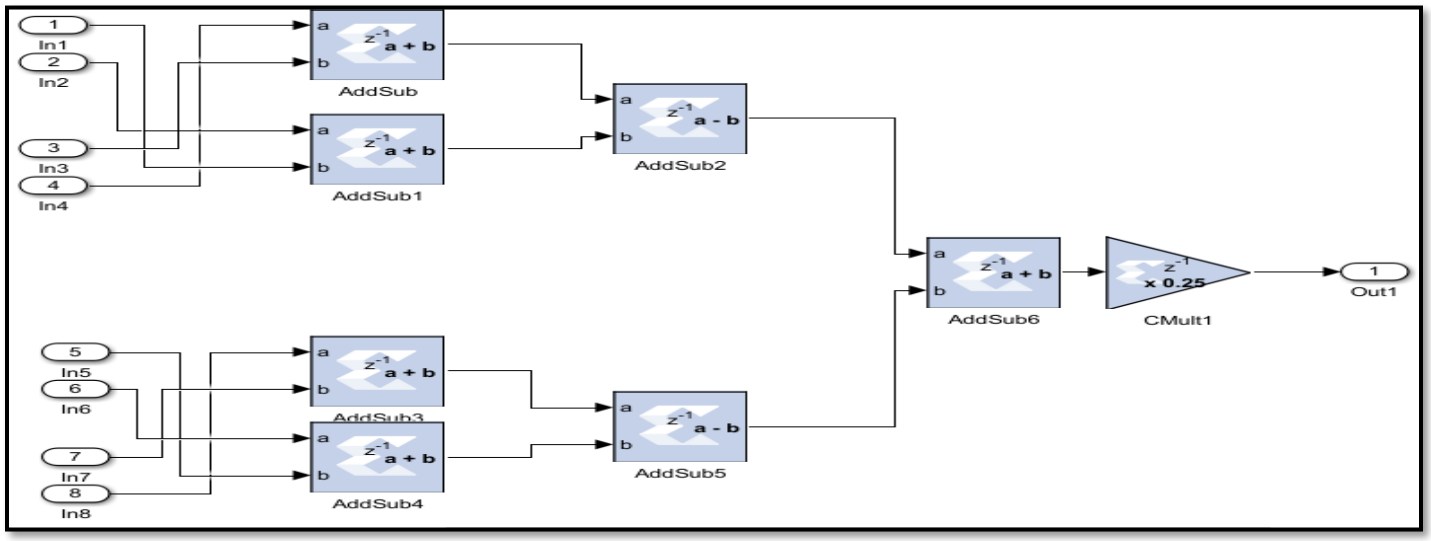


Figure 3-8 Gradient Horizontal avec XSG

- **Gradient vertical :**

Le gradient vertical **Iy** donné par la relation :

Iy = conv2 (im1, ky, 'same') + conv2 (im2, ky, 'same')

Noyau ky:

-1/4	**-1/4**
1/4	**1/4**

- **Gradient vertical avec Simulink:**

La figure 3-9 montre la réalisation de gradient vertical avec Simulink :

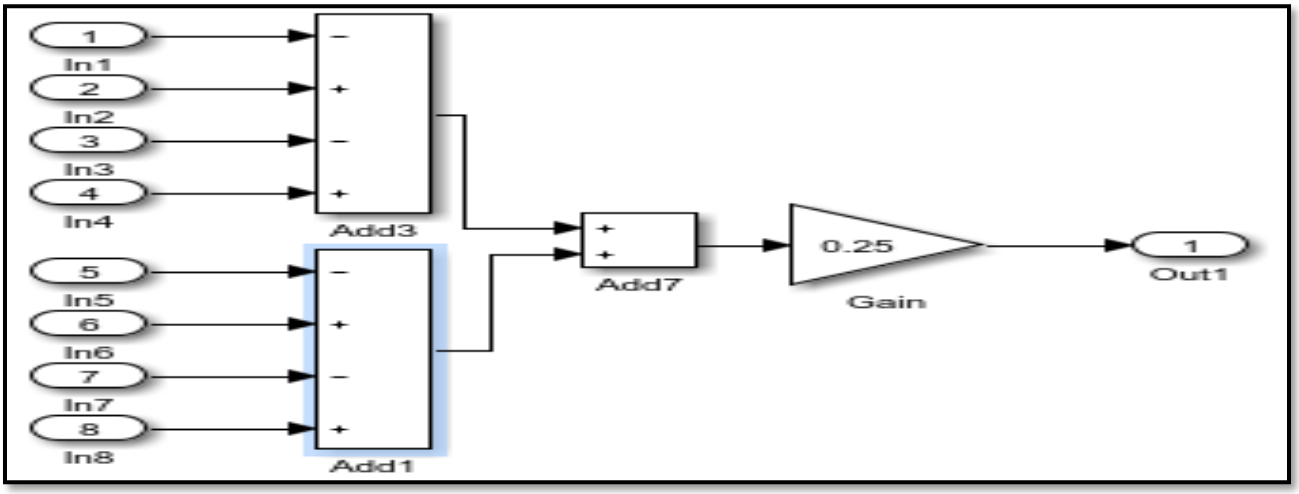


Figure 3-9 Gradient Vertical avec Simulink

- **Gradient vertical avec XSG :**

La figure 3-10 montré le Gradient vertical avec XSG :

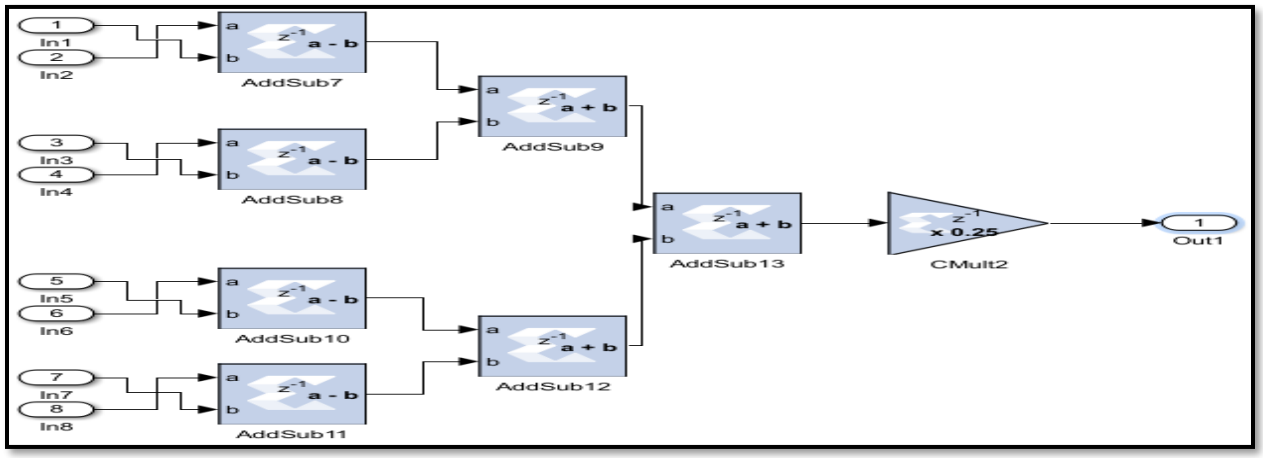


Figure 3-10 Gradient vertical avec XSG

- **Gradient temporel:**

Le gradient vertical **It** donné par la relation :

It = conv2 (im1, kt, 'same') + conv2 (im2, -kt, 'same')

Noyau kt**:**

-1/4	**-1/4**
-1/4	**-1/4**

- **Gradient temporel avec Simulink:**

La figure 3-11 montre la réalisation de gradient temporel avec simulink :

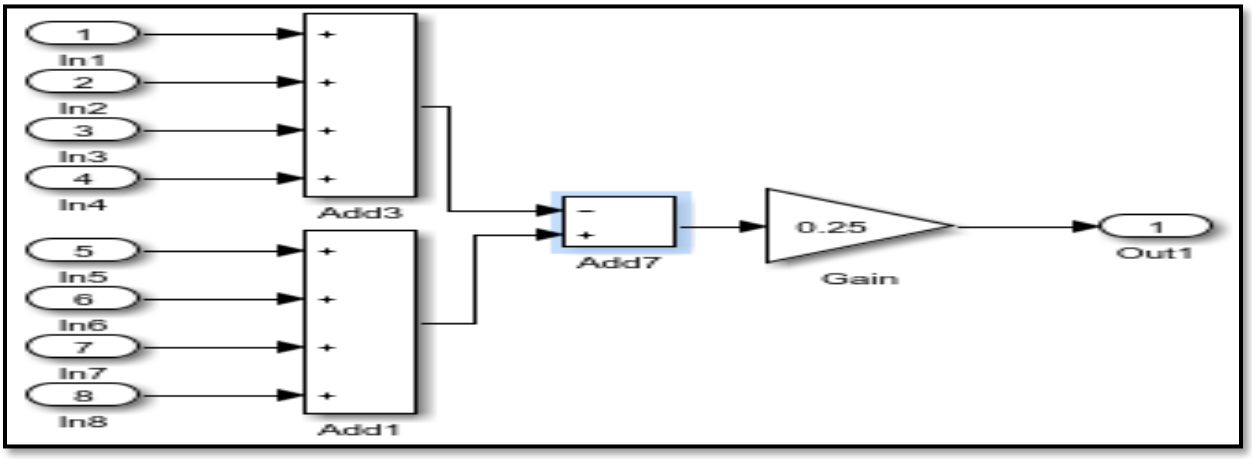


Figure 3-11 Gradient temporel avec Simulink

- **Gradient temporel avec XSG :**
  La figure 3-12 montré le Gradient temporel avec XSG

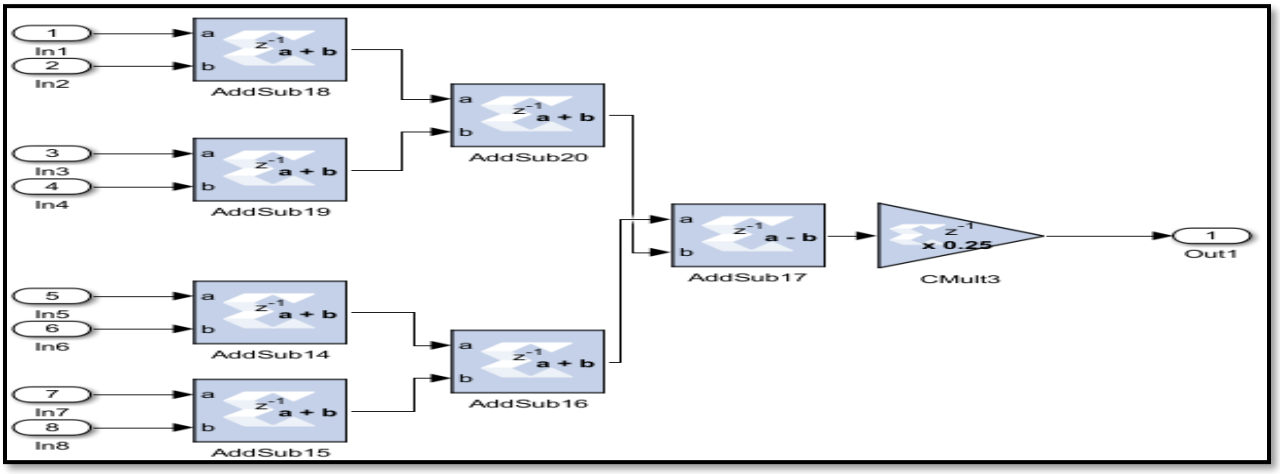


Figure 3-12 Gradient temporel avec XSG

➢ **Calculateurs de termes Axy :**

Les deux termes **Ax** et **Ay** donne par les relations suivantes :

$$\mathrm{Ax} = \frac{\mathrm{Ix}}{(\alpha^2 + \mathrm{Ix}^2 + \mathrm{Iy}^2)}$$

$$\mathrm{Ay} = \frac{\mathrm{Iy}}{(\alpha^2 + \mathrm{Ix}^2 + \mathrm{Iy}^2)}$$

- **Bloc Axy avec Simulink:**
  - La figure 3-13 montre la réalisation de bloc **Axy** avec Simulink :

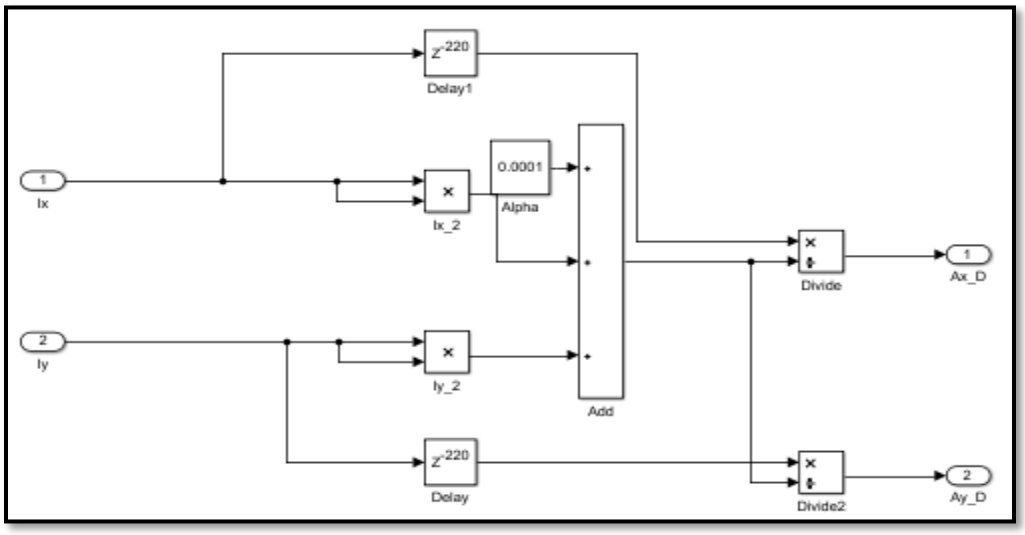


Figure 3-13 bloc Axy avec Simulink

- **Bloc Axy avec XSG:**

La figure 3-14 montre la réalisation de bloc Axy avec XSG

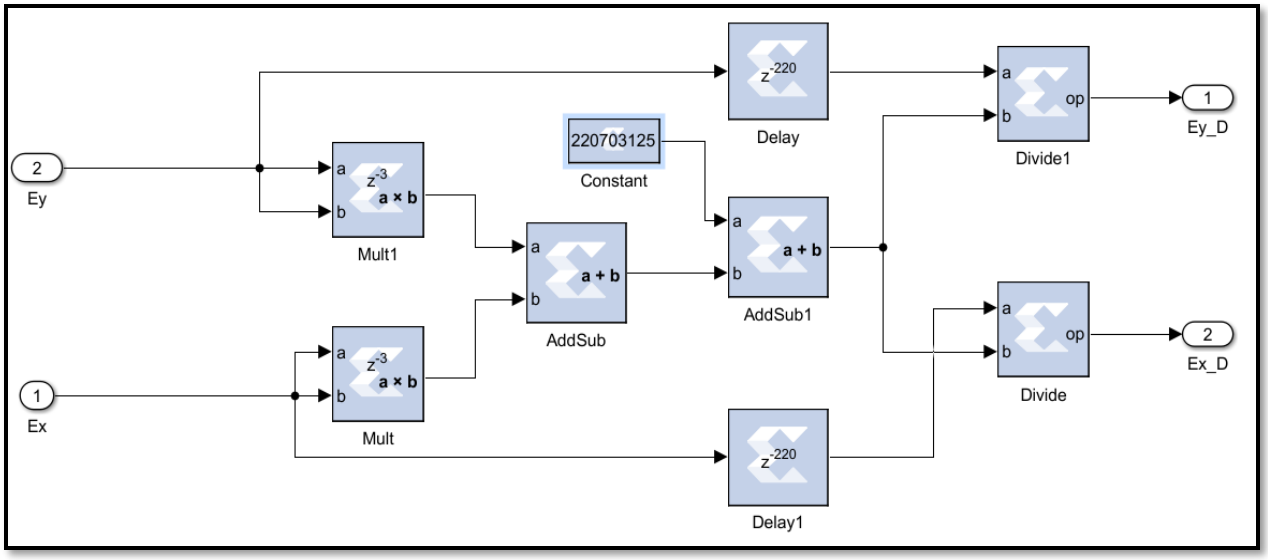


Figure 3-14 bloc Axy avec XSG

- **Laplacien X et Y :**

Laplacien X et Y est donné par les relations suivantes :

U_avg = conv2 (U, k, 'same')

V_avg = conv2 (V, k, 'same')

Le noyau de convolution k est la suivante :

1/12	1/6	1/12
1/6	-1	1/6
1/12	1/6	1/12

- **Block Laplacien avec simulink :**

La figure 3-15 montre la réalisation de bloc Laplacien avec Simulink :

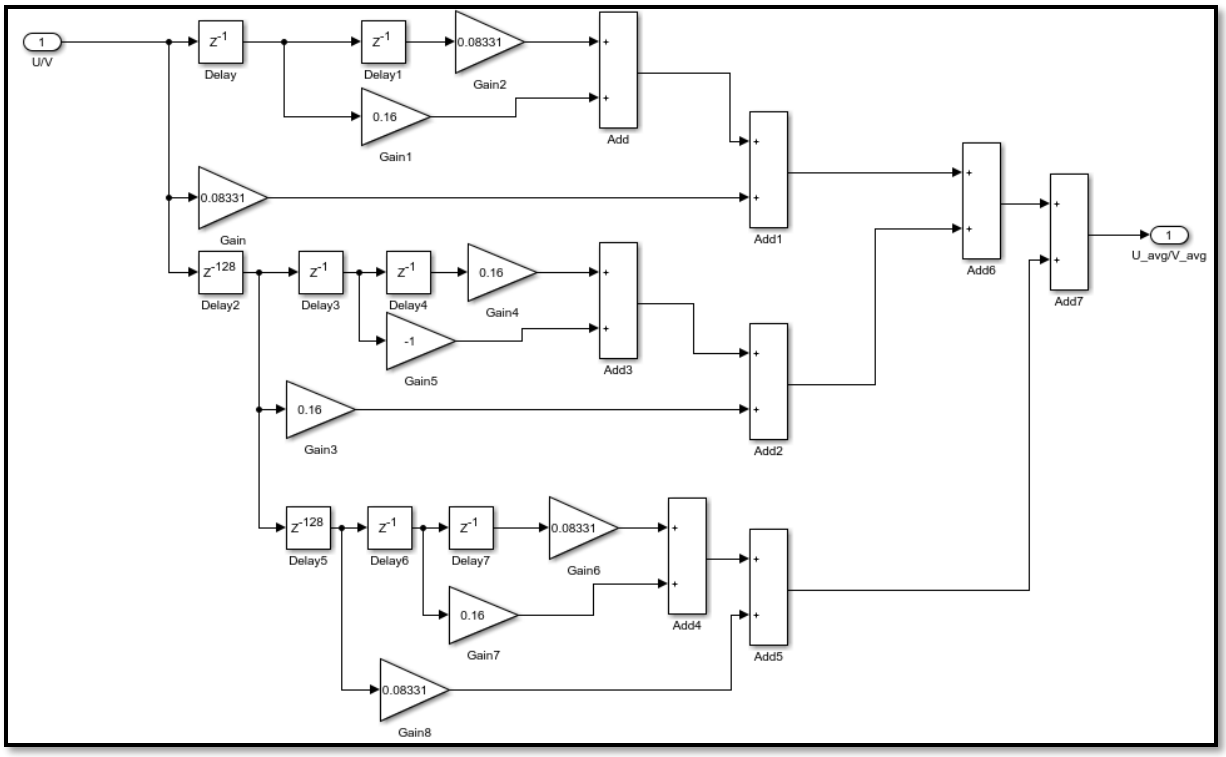


Figure 3-15 bloc Laplacien avec Simulink

- **Bloc Laplacien avec XSG :**

La figure 3-16 montre la réalisation de bloc Laplacien avec XSG

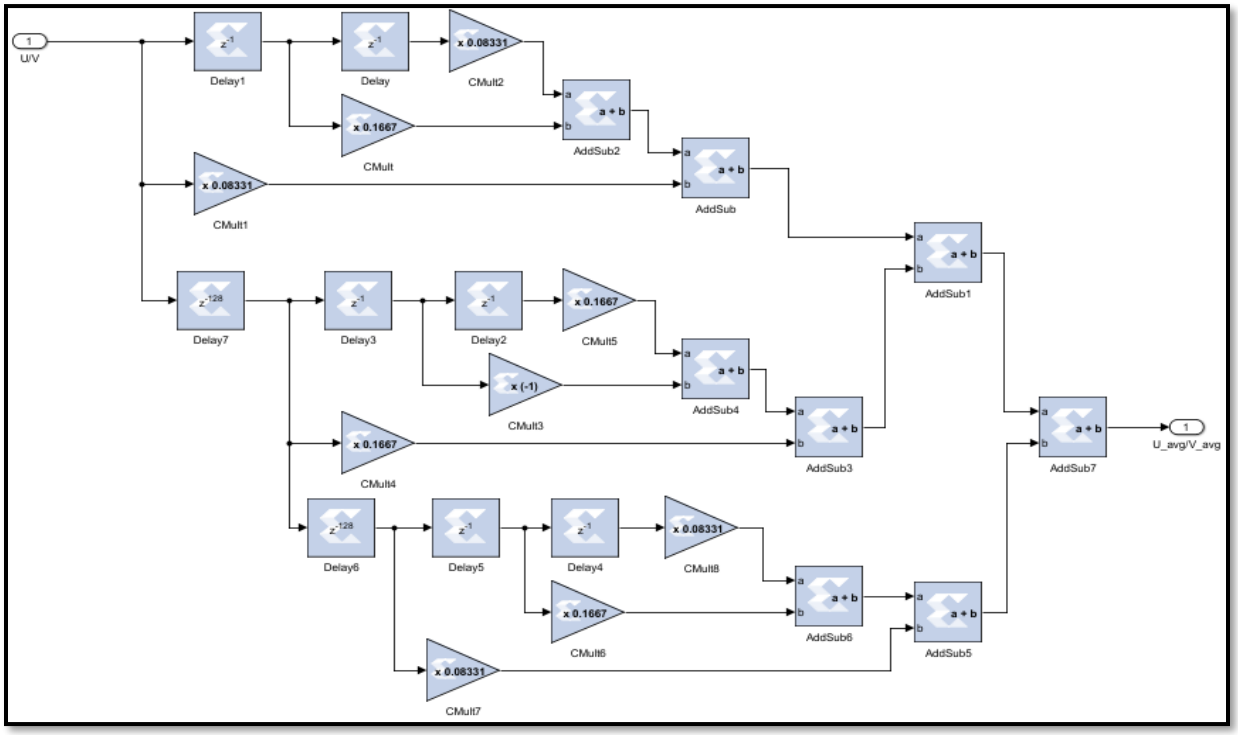


Figure 3-16 bloc Laplacien avec XSG

- **Bloc B :**

En réalise le bloc B qui inclut :

- Calculateurs de terme B :
- B= $Ix\,\overline{U} + Iy\,\overline{V} + It$ tel que : $\overline{U}$: Laplacien X, $\overline{V}$: Laplacien y
- Les deux produits Ax*B et Ay*B.

le bloc B répéter 5 fois comme montre dans la figure 3-16

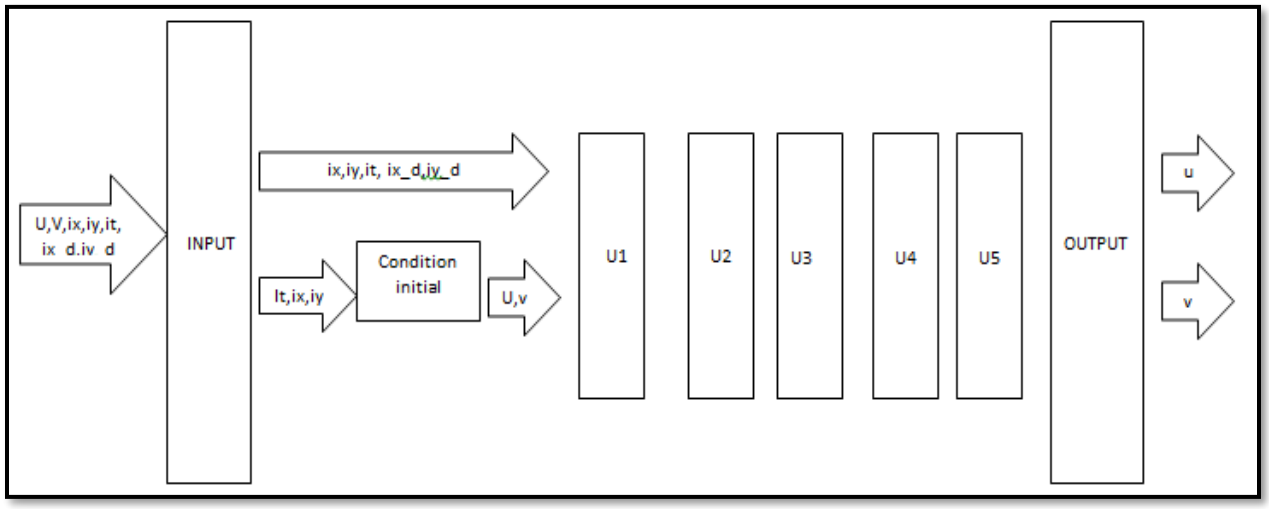


Figure 3-16 le fonctionnement de bloc B

- **bloc B avec Simulink :**
- la figure 3-17 montre la réalisation de bloc Laplacien avec Simulink

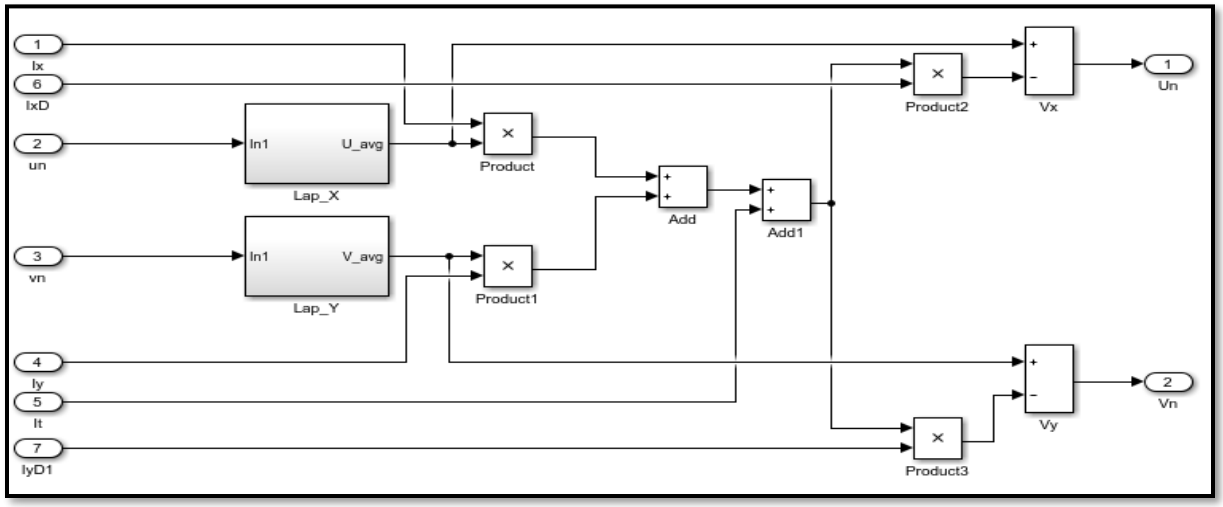


Figure 3-17 la réalisation de bloc B avec Simulink

- **bloc B avec XSG:**

La figure 3-18 montre la réalisation de bloc Laplacien avec XSG

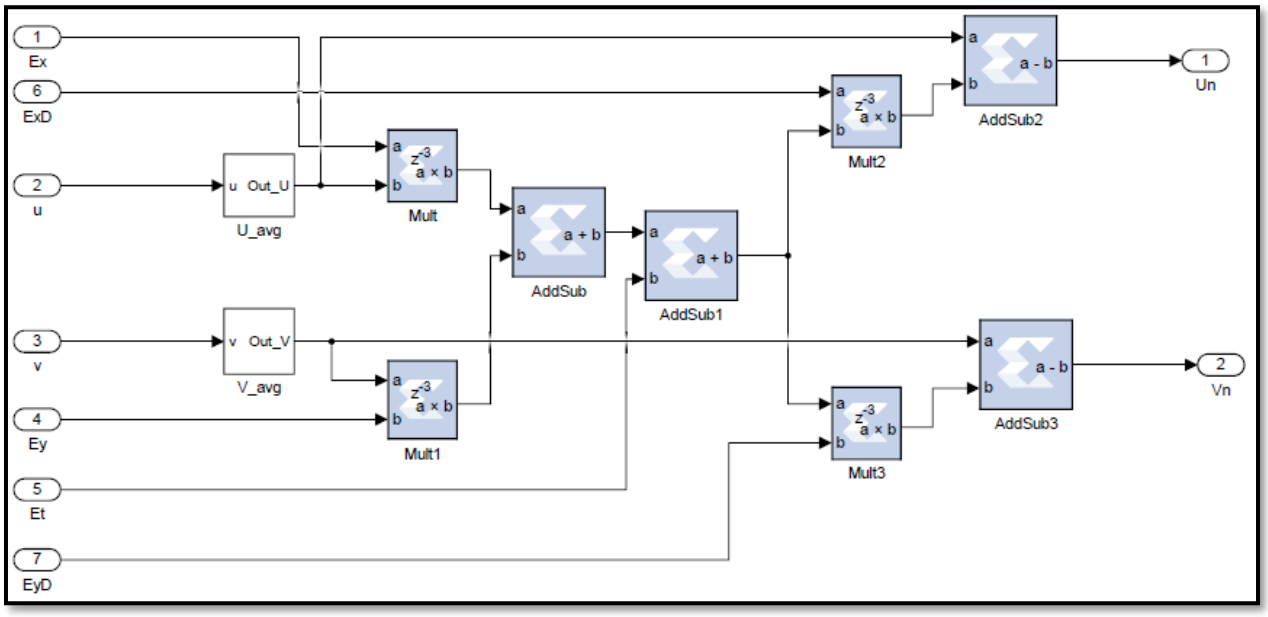


Figure 3-18 bloc B avec XSG

- **les conditions initiales :**

En suppose que la vitesse initiale d objet encore a étude est nul donc au début en a :

$U_0 = 0, V_0 = 0$

A l entrée de première itération en a :

$U_1 = U_0 - IxD * It$, $V_1 = V_0 - IyD * It$

La réalisation des conditions initiale avec simulink et XSG :

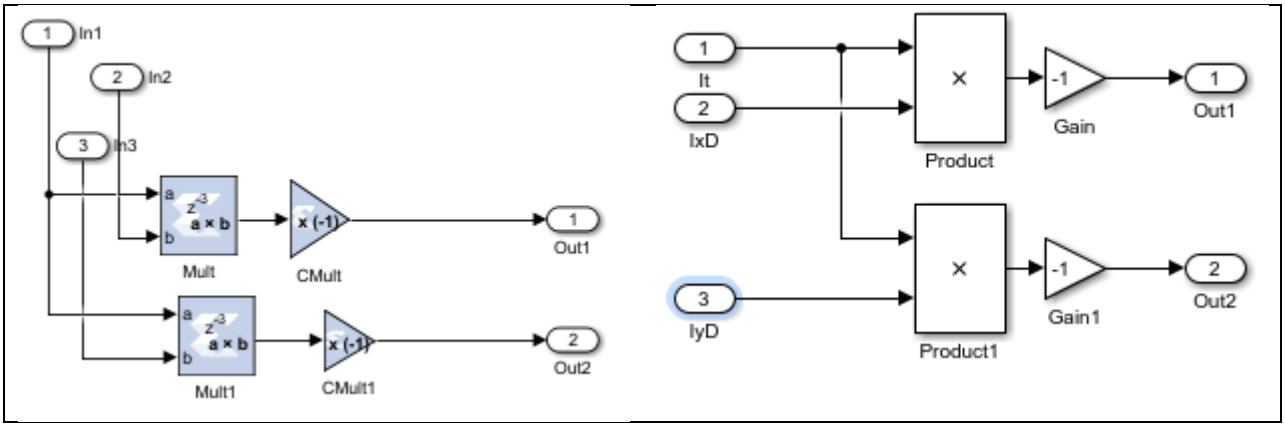


Figure 3-19 conditions initiales Avec simulink et XSG

3-4 Résultats de simulation et discussion :

Pour vérifier les résultats qui on a obtenu pour chaque gradient en prend 2 images successives de mouvement d'un carré :

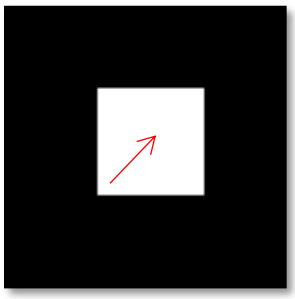
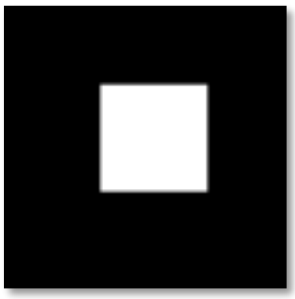
	

Image 1	Image 2

Tableau 3-1 les images utilisé pour vérification des gradients

Les résultats de simulation des gradients avec simulink et XSG montrent dans le tableau 3-2

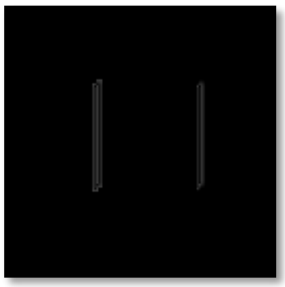
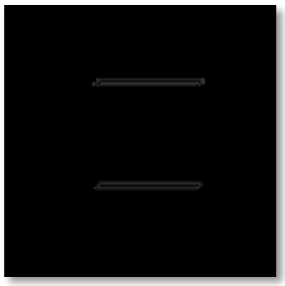
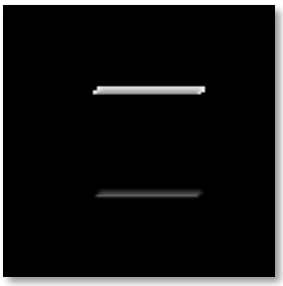
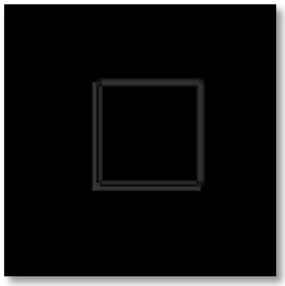
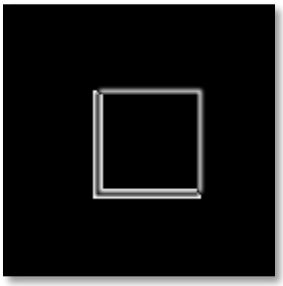
block	logiciel	simulink	XSG
Gradient X			
Gradient y			
Gradient T			

Tableau 3-2 Les résultats de simulation des gradients avec simulink et XSG

Pour vérifier les résultats qui ont obtenu pour les blocs **Ax** et **Ay** en prend les gradients des 2 images Successives de mouvement d'une voiture :

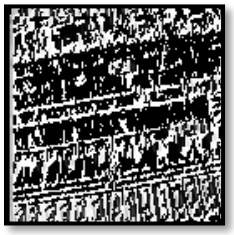
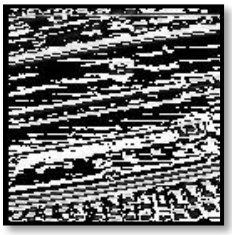
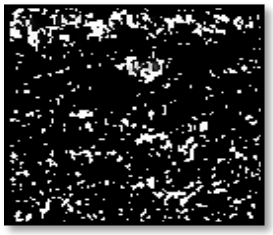
				
Im 1	Im2	Gradient X	Gradient y	Gradient T

Tableau 3-3 : les images utilisé pour vérification des 2 blocs Ax et Ay

Les résultats de simulation avec simulink et XSG montrent dans le tableau 3-4 :

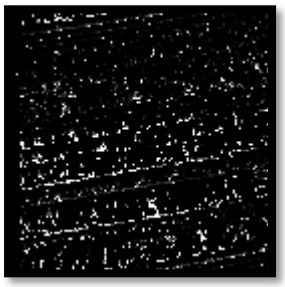
block / logiciel	simulink	XSG
Ax		
Ay		

Tableau 3-4 Les résultats de simulation des blocs Ax Ay avec simulink et XSG

- **discussion :**

En remarque qui il y a une différence entre les résultats simulink et XSG parce que avec Simulink en prend la précision complété (simple / double) des valeurs des pixels par contre avec XSG en utilisant l'entier et la fraction avec des représentations à virgule fixe.

L'entier et la fraction avec des représentations à virgule fixe :

- réduits les ressources matérielles.
- la vitesse de traitements peut être améliorée.

3-5 Conclusion :

Après avoir étudié plusieurs architecture possibles, le schéma fonctionnel sur figure 3-1 a été sélectionner cette architecture peut être divisé en 4 parties principales dans ce chapitre en avoir la réalisation de chaque partie à part avec simulink qui est un environnement informatique destiné pour la simulation et on a comparé le résultat obtenir avec l'autres d XSG qui permettre d'obtenir une programme près à implémenter.

Conclusion générale :

Dans ce mémoire nous sommes l'une des méthodes d'estimation de flot optique (Horn et Schunck) puis en fait une étude comparative entre 2 types de simulation de cette méthode.

La 1ére est une simulation à l'aide d'un environnement informatique Simulink.et l'autre avec générateur du système (xilinx) qui permettre après la vérification de résultat d'obtenu le programme HDL, ce programme est pré pour implémenter sur des plateformes reconfigurables qui à certaines caractéristiques déférentes des systèmes informatiques. Le programme obtenu prise en compte tout ses caractéristiques.

Perspective :

Il serait intéressant d'implémenter la méthode étudiée sur une Carte de développement FPGA pour montrer son efficacité.

Annexe

Des Sites utiles :

- www.Xilinx.com
- www.youtube.com
- www.mathworks.com

Bibliographie

[1] Détection et catégorisions d'Object en mouvement dans une vidéo thèse pour obtenir le titre de docteur en sciences -université de Caen basse Normandie-, France 2009 [page 2].

[2,3] vision par ordinateur-3D en mouvement-NICOLAS Loménie-master informatique-université de paris Descartes-[page 7,5]

[4] analyse de mouvement (estimation de flou optique) Antoine Manzanera -ENSTA-[page 5].

[5] Y. Weiss, et E. H. A, Adelson, uniéd mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'96) , San Francisco (CA), USA, 1996. [pages 321-326]

[6] J.-M. Odobez, et P. Bouthemy, Direct incremental model-based image motion segmentation for video analysis, Signal Processing, 1998.

[7], A.ROSENFELD - « Digital Picture Processing », Academic Press, 1982 J.P.COCQUEREZ, S.PHILIPP

[8], « Analyse d'images : filtrage et segmentation », Masson,1996.

[9] Traitement des Images médicales D´détection, estimation et analyse du mouvement Dominique.Bereziat

[10] Rima, H. (2009). Contribution à une plateforme d'aide à la conception de microsystème. LYON, FRANCE: Institut national des sciences appliquées de Lyon.

[11] Smith, G. R. (2010). FPGA 101. New York: ELSEVIER.

[12] TAVERNIER, C. (1996). Circuits logiques programmables. PARIS: DUNOD.

[13] Rima, H. (2009). Contribution à une plateforme d'aide à la conception de microsystème. LYON, FRANCE: Institut national des sciences appliquées de Lyon.

[14] TAVERNIER, C. (1996). Circuits logiques programmables. PARIS: DUNOD