الجمهورية الجزائرية الديموقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة امحمد بوقرة بومرداس
Université M'Hamed Bougara Boumerdes

Faculté des Sciences de L'ingénieur
Département : Ingénierie des Systèmes Electriques

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master

Spécialité : Electronique des Systèmes Embarqués

Thème :

Implémentation d'un détecteur de contours sur carte FPGA pour le contrôle qualité des cordons de soudure

Présenté par:

Niche Zineb

Encadreur:

Dr. Mazouz Nezhate M.R.B (C.R.T.I)

Co-encadreur:

Mlle. Mahdi Ismahane M.A.A (U.M.B.B)

Promotion 2017/2018

Remerciements

*Tout d'abord je remercie **ALLAH** le tout puissant de m'avoir donné le courage et la patience d'arriver à ce jour.*

*Je tiens à remercier **Mr YAHI Mostapha**, **Directeur du Centre de Recherche des Technologies Industrielles (C.R.T.I.)** de m'avoir permis d'effectuer mon stage au sein de son établissement, ainsi que **Mr Drai Redouane**, C**hef de la Division de Traitements du Signal et Imagerie** et tous les membres de son équipe de recherche pour m'avoir procuré de bonnes conditions de travail et de m'avoir orienté tout au long de mon stage.*

*J'adresse mes plus sincères remerciements à mon encadreur **Dr. MAZOUZ Nezhate** de m'avoir encadré et donné le maximum de ses connaissances, ainsi que ces précieux conseils tout au long de mon projet de fin d'études.*

*J'exprime ma profonde gratitude à ma **co-encadreur Mme Mahdi Ismahan** de m'avoir dirigé, conseillé durant mon travail.*

Enfin, j'adresse mes plus grands remerciements qui vont aussi à mes chers parents.

Niche Zineb

Dédicaces

Je dédie ce travail :

A mes très chers ***parents*** *qui n'ont pas cessé de m'encourager et de me soutenir durant toute la période de mes études, que Dieu les gardent pour moi.*

A mes chers frères et sœur :

Abdo, Rahim et Sara

A mes chers collègues : ***Adel, Amina, Cherifa, Mohamed*** et ***Walid***.

A toutes mes amies de la FSI et spécialement à : ***Amira, Bouchra, Meryem, Tissame, Rihana, Samira, Hassiba et Oussama***.

Et enfin,

A tous ceux qui m'aiment et que j'aime.

Niche Zineb

Résumé

ملخص:

الهدف الزئيسي من هذا العمل هو تحقيق نظام مدمج جزئيا أوتوماتيكي , يسمح بمراقبة جودة التماس لحام الأنابيب. لذلك قمنا بعملية التحكم في كاميرا صناعية باستخدام بطاقة مزودة بدارة مصفوفة ابواب القابلة للبرمجة ميدانيا , حيث تم تنفيذ عملية استحصال ومعالجة الصور المضمنة بواسطة مرشح سوبل) على نفس الدارة.

الكلمات المفتاحية: مراقبة الجودة ، التماس اللحام ، الأنابيب ، الكاميرا الصناعية ، مرشح سوبل.

Résumé :

L'objectif de ce travail est de réaliser un système embarqué permettant d'automatiser, en partie, le contrôle qualité du cordon de soudure. Pour cela, nous avons réalisé la commande d'une caméra industrielle à l'aide d'une carte FPGA où l'acquisition et le traitement d'image (filtre de Sobel) sont implémentés sur le même circuit FPGA.

Mots clés : Contrôle qualité, cordon de soudure, pipes, caméra industrielle, filtre de Sobel.

Abstract :

The objective of this work is to realize an embedded system allowing to automate, partly, the quality control of the weld bead. For that, we have carried out the command of an industrial camera using an FPGA card, where the acquisition and the image processing (Sobel filter) are implemented on the same FPGA circuit.

Keywords : Quality control, weld bead, pipes, industrial camera, Sobel filter.

Sommaire

Table des figures

Liste des tableaux

Table des abréviations

- ACP Accelerator Coherency Port.
- AMBA Advanced Microcontroller Bus Architecture.
- AMP Asymmetric Multi-Processing
- APU Application Processing Unit.
- ASIC Application Specific Intgreated Circuit.
- ASSP Application Specific Standard Circuit.
- AXI Advanced eXtensible Interface.
- BRAM Block Random Access Memory.
- BSP Board Support Package.
- CND Contrôle Non Destructif.
- CCD Charge Coupled Device.
- CD Contrôle Destructif.
- CLB Configurable Logic Bloc.
- CMOS Complementary Metal Oxide Semi conductor.
- DMA Direct Memory Access.
- DSP Digital System Processing/Processor.
- EMBV EMBedded Vision kit.
- FPGA Field Programmable Gate Arrays.
- GIC General Interrupt Controller.
- GPIO Général Purpose Input/output.
- HMI Human Machine Interface.
- HDMI High Definition Multimedia Interface.
- HDL Hardware Description Language.
- IPCore Intellectual Property Core.
- I/OB Input/Output Bloc.
- LUT Look-Up Table.
- MIO Multiplexed Input/Output.
- OCM On Chip Memory.
- PAL Programmable Array Logic.
- PLD Programmable Logic Device.
- PLC Programmable Logic Controller.

- PS Processing Logic.
- PL Programmable Logic.
- SCU Snoop Control Unit.
- SD Contrôle Semi-Destructif.
- SIMD Single-Instruction-Multiple-Data.
- SMP Symmetric Multi-Processing.
- SoC System on Chip.
- SOM System On Module.
- SXGA Super-EXtended Graphics Array.
- TCL Tool Command Language.

Introduction générale

De nos jours, le soudage joue un rôle très important dans l'industrie, puisqu'il est utilisé dans plusieurs secteurs tel que : l'automobile, le transport des fluides, la construction mécanique, navale et aéronautique.

Par définition, le soudage est un procédé d'assemblage permanent de deux ou plusieurs pièces ou structures par fusion localisée du métal. En général, il existe deux types de soudure: La soudure homogène comme dans les pipes de transport des hydrocarbures et la soudure hétérogène tel que : la plomberie. Il est clair que la qualité des joints soudés a un impact direct sur la fiabilité de la structure métallique ainsi que sur ces caractéristiques mécaniques.

Le soudage est aussi employé lors de la fabrication des tubes en acier (dits : « pipes »), dédiés au transport des hydrocarbures et au transfert d'eau des barrages.

Par ailleurs, afin d'évaluer la qualité du cordon de soudure des pipes et de respecter les mesures de sécurité strictes (basées sur des normes internationales, tel que API 5, il est nécessaire d'effectuer un contrôle qualité à deux niveaux :

- le premier étant lors de la fabrication et avant la mise en marché du pipe.
- Le deuxième contrôle qualité est effectué après une certaine durée d'exploitation, comme c'est le cas pour les pipes, destinés au transport des hydrocarbures.

En effet, la surveillance permanente de l'état des pipes, permet donc d'éviter les catastrophes écologiques et économiques (citons comme exemple : les catastrophes, liées au transport pétrolier). C'est pour cela, qu'on procède régulièrement à des périodes bien définie au contrôle des pipes par différentes méthodes qui sont :

- Le contrôle non destructive (C.N.D.)
- Le contrôle destructif (C.D.)
- Le contrôle Semi-Destructif (S.D.)

Soulignons que le premier contrôle qualité des pipes se fait au niveau des complexes industriels métallurgiques et sidérurgiques et il requiert l'utilisation de plusieurs méthodes et techniques, tel que le contrôle visuel et mesures dimensionnelles du cordon de soudure des pipes.

Par ailleurs, le deuxième contrôle se fait sur site (c'est à dire après exploitation des pipes), où les trois méthodes de contrôles (C.N.D., C.D. et S.D.), font parties des domaines de recherches de la division de Traitement du Signal et Imagerie (D.T.S.I.) du Centre de Recherche et Développement en Technologies Industrielles (C.R.T.I.) [1].

A travers le temps, les différents développements technologiques ont révolutionné un bon nombres d'industries, dans le but de simplifier le processus de fabrication et, en conséquence, d'accélérer le temps de mise sur le marché du produit fini. En effet, ces dernières années, on assiste à un intérêt croissant pour l'automatisation du processus de contrôle qualité des produits finis, car avec

l'augmentation de la vitesse de traitement de données et la disponibilité des caméras industrielles, le contrôle de qualité basé sur la vision par ordinateur[1] à 2D ou 3D (appelée aussi vision artificielle) est devenu possible.

Dans le cas des pipes, le contrôle qualité du cordon de soudure lors de leur fabrication, implique l'analyse de la morphologie des images métallographiques. En général, ce contrôle est accompli d'une manière rudimentaire : un contrôle visuel associé à un calcul manuel sur des images des échantillons de cordon de soudure, qui sont préparés dans un laboratoire. De nos jours, cette méthode de contrôle qualité est jugée « trop lente, coûteuse et moins précise ».

Pour parer à ce problème, on se propose, dans le cadre de ce mémoire, de réaliser un système embarqué permettant d'automatiser, en partie, le contrôle qualité du cordon de soudure. Pour ce faire, nous avons défini les tâches suivantes :

1. Acquisition d'image d'un échantillon du cordon de soudure : à l'aide d'une caméra industrielle, commandée par une carte à base de circuit FPGA.
2. Traitement de l'image obtenue par extraction des contours utiles.

Les résultats obtenus dans ce mémoire, sont utilisés dans les travaux de recherche de la division de « Traitement du Signal est Imagerie (D.T.S.I) » [1]**.**

Il est à noter que pour réaliser les tâches de ce mémoire, les outils de développement suivants, ont été utilisés: la suite de développement de Xilinx (VIVADO, SDK, IP Integrator, ISIM, XST) version 2016.3, MATLAB version *2013.a*, ainsi que les langages de programmation C/C++, TCL Scripting.

Afin d'atteindre les objectifs que nous nous sommes fixés, le présent mémoire est organisé comme suit:

Le premier chapitre présente succinctement les généralités sur la fabrication et le contrôle qualité du cordon de soudure des pipes. Le second chapitre présente des généralités sur les cameras industrielles dédiées au contrôle qualité. Le chapitre suivant est consacré à la commande d'une caméra industrielle de type CMOS par une carte FPGA à base de circuit Zynq 7000.

Le dernier chapitre liste les principaux algorithmes utilisés dans le domaine de traitement d'image (filtres de: Robert, Sobel, Prewitt, Canny). Aussi, une implémentation matérielle du filtre de Sobel sur circuit FPGA est présentée.

A la fin, nous terminons par une conclusion générale, qui met en évidence les résultats obtenus et ainsi que les perspectives futures de développement de ce projet.

1 - Vision par ordinateur : dite « Computer Vision ».

Chapitre I: Généralités sur le contrôle qualité du cordon de soudure des pipes

I.1. Introduction

Par définition, un pipe est une canalisation. Elle peut être enfuie sous terre ou sous mer, comme elle peut être entièrement, ou en parties, apparente.

En général, pour pouvoir transporter un liquide ou un gaz, une série de pipes sont soudés entre eux pour former des lignes de longues distances, appelés : « pipelines ». Par exemple, on trouve des pipelines de transport des hydrocarbures au Sahara Algérien (voir fig. I.1 et I.2) avec différentes longueurs et formes géométriques.

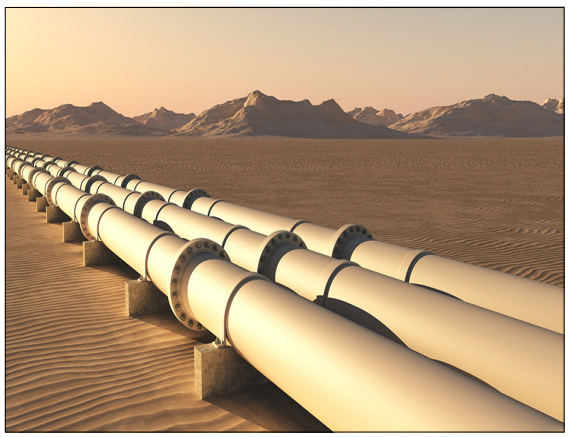


Figure I.1 : Pipeline apparent.

Figure I.2 : Pipeline en construction.

Grâce aux progrès de la soudure, la technologie des tuyauteries a été révolutionnée, ce qui a induit à la naissance des sites importants de pipelines (comme dans le Sud Algérien).

Pour construire un pipeline, les pipes sont acheminés sur place (ou sur site) et mis dans des tranchées ou sur des supports par des grues spéciales (voir fig. I.3). Des systèmes permettent d'emboîter les tuyaux avec précision. Les tubes soudés suivent le relief du terrain sans qu'il y est de fuites malgré que les pressions internes peuvent dépasser les 10 bars.

Les points de pressions causées par des coups de bélier ou autres peuvent être importants. La structure, les parois et les soudures doivent résister à des pressions très élevées, de l'ordre de 100 bars.

Il est à noter qu'avant d'exploiter les pipes sur site (comme dans les figures I.2 et I.3), les pipes doivent être vérifiés juste après leur fabrication. Cette opération est appelée: contrôle qualité. C'est à dire le pipe doit être conforme ou des spécifications ou exigences préétablies et incluant une décision d'acceptation ou de rejet

Figure I.3 : Soudage sur sites des pipes.

Rappelons qu'avant de voir ce qu'est le contrôle qualité des pipes après leur fabrication, il est nécessaire de connaître leur procédé de fabrication.

I.2. Evolution de l'utilisation des pipes

En 1865 et aux États-Unis, pour la première fois, une conduite a été utilisée pour évacuer la production d'un gisement [2] . Depuis, l'emploi de ce mode de transport des hydrocarbures liquides ou gazeux s'est généralisé.

C'est vers les années 1920 que sont apparus les premiers tubes en acier soudés à partir d'un feuillard chauffé, où le soudage se fait par rapprochement des bords en fusion. Par la suite, l'invention de la soudure électrique a permis, par effet de Joule, de fondre le métal, ce qui a mené au développement de l'industrie des petites soudures par résistance puis par induction.

A partir des années 1940, le développement de la sidérurgie s'est concentré sur l'industrie des tubes soudés dans le but de répondre au besoin croissant de la construction mécanique, ce qui a mené à l'apparition de nouvelles techniques et procédés de soudage automatique sous flux avec apport de métal. Ce flux est généralement constitué d'un gaz rare ou d'une poudre composée d'éléments favorables à l'homogénéité du métal déposé par un fil métallique, possédant un aspect mécanique et une composition chimique proche des tubes à souder.

En général, les tubes soudés sont classés en trois familles :

- les petits tubes soudés de diamètres inférieur à 219 mm (ou 8'').
- les moyens tubes soudés de diamètres entre 228.6 mm à 406.4 mm (ou de 9'' à 16'').
- les gros tubes soudés de diamètres supérieur à 406.4 mm (ou 16'').

Mis à part la multiplicité des diamètres des pipes soudés, les pipes de transport des

hydrocarbures et hydrauliques utilisés en Algérie, sont classés en 3 catégories :

- Les pipes non soudés.
- Les pipes soudés en spirale.
- Les pipes soudés longitudinalement.

Entre 1964 et 1965, étaient inaugurés les premiers transports réguliers de gaz naturel liquéfié entre l'usine de liquéfaction d'Arzew (Algérie) et les terminaux méthaniers de Canvey Island (Royaume-Uni) et du Havre (France) [2].

En 1993, les principaux pays exportateurs de gaz étaient l'ex-URSS, les Pays-Bas, l'Algérie, l'Indonésie et la Norvège. Les principaux pays importateurs de gaz étaient l'Allemagne, les Etats-Unis, le Japon, l'Italie, la France, la Tchécoslovaquie et la Belgique.

En Afrique, l'Algérie possède un important réseau de gazoducs destiné principalement à évacuer le gaz du gisement de Hassi R'Mel vers les usines de liquéfaction situées sur les côtes méditerranéennes ainsi que vers l'Italie via la Tunisie (système TransMed).

Notons qu'en raison de la forte demande du gaz et des produits pétroliers et qui risque de doubler d'ici l'année 2025, les pipelines sur de longues distances sont des moyens sûrs et économiques pour le transport du gaz entre les sites d'exploitation et les points de consommation, qui peuvent être parfois séparés par plus de 5000Km [2].

I.3. Fabrication des pipes en Algérie

Dans le cadre de ce mémoire, on se limitera à la présentation de la fabrication des pipes de gros diamètres, qui sont utilisés dans le transport des hydrocarbures et de l'hydraulique (voir fig. I.4).

Figure I.4 : Pipe après soudage en spirale.

En Algérie, le groupe Industriel IMETAL- ALFAPIPE, situé à Annaba [3], est le leader de fabrication des produits plats :

- Tubes en acier soudés, destinés au transport des hydrocarbures (gaz et pétrole) ainsi qu'au transfert d'eau des barrages.
- Tubes soudés en hélicoïdal destinés au transport d'hydrocarbures (liquides et gazeux) de diamètre de 16'' à 64'', d'épaisseur de 6 à 15 mm et de longueur de 7 à 13 m, conformément aux normes API 5L et API Q1.

La figure I.5 résume les grandes étapes du procédé de fabrication des pipes avec soudage en spirale.

En général, ces pipes sont fabriqués à partir d'un formage des tôles en acier laminées à chaud suivi d'une opération de soudage en spirale à double fil et à double face [4]. La soudure est réalisée à l'arc submergé.

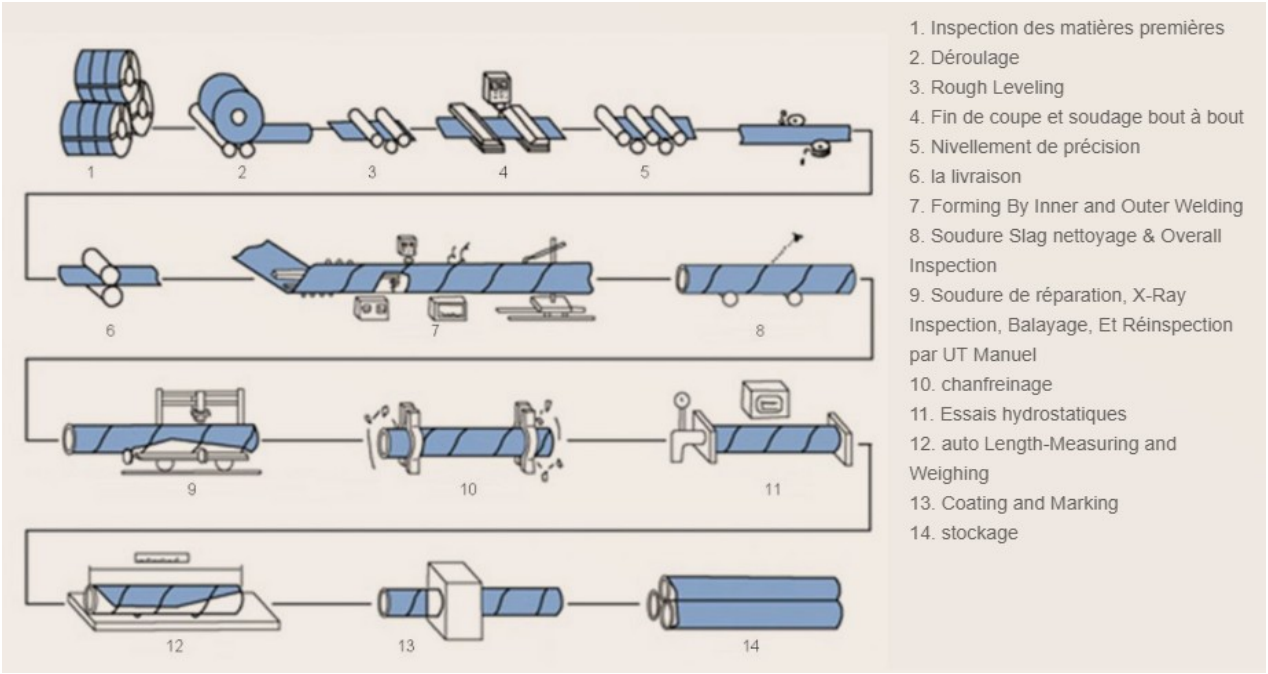


Figure I.5 : Procédé de fabrication des pipes avec soudage en spirale.

La figure ci-dessus détaille le procédé de fabrication des tubes soudés en spirale. Cette opération est réalisée en continu sur des machines à souder. Après déroulement de la bobine en bande par un train d'entraînement, la tôle est poussée dans la cage de formage où elle subit une déformation de cintrage sous l'action d'un vérin. L'obtention du formage en spirale est réalisée par l'inclinaison d'un angle α entre l'axe initial de la bande et l'axe de sortie du tube. L'enroulement des tubes en spirale permet d'obtenir des tubes calibrés.

La fabrication des tubes spirale peut être subdivisée en opérations principales suivantes :

1. La préparation de la bande.
2. La bande de raboutage.

3. Le cintrage.
4. Le soudage.
5. Le parachèvement.
6. La réception et inspection.

I.4. Définition du cordon de soudure

Le cordon de soudure représente la zone de liaison entre deux pièces soudées avec ou sans métal d'apport (voir fig. I.6).

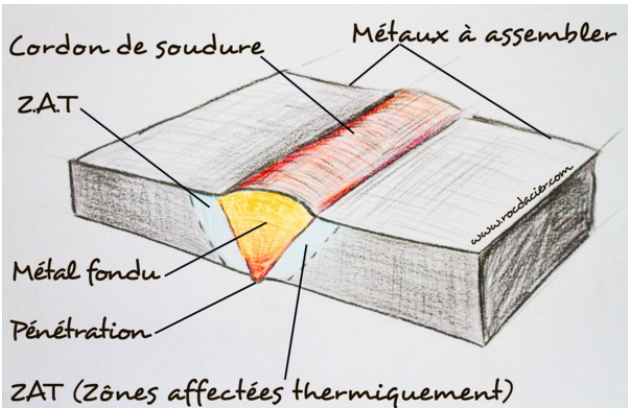


Figure I.6 : Cordon de soudure.

Les pièces à souder sont appelées métaux de base et le cordon de soudure n'est autre que le mélange (métal fondu) de ces métaux de base avec le métal d'apport. Les métaux de bases peuvent être homogènes (de même nature) ou hétérogènes [2].

Le cordon de soudure peut éventuellement présenter un léger excédent de métal à la racine qu'on appellera pénétration ou excès de métal.

Le cordon de soudure est constitué en grande partie du métal d'apport, puis d'une proportion des métaux de bases.

Les frontières théoriques entre le cordon et les métaux de bases sont appelées zones de contact ou zones de liaisons. Ensuite la zone qui aura subit l'échauffement du à la soudure s'appelle zone affectée thermiquement (Z.A.T).

On distingue plusieurs types de soudures qui sont présentées dans la figure suivante :

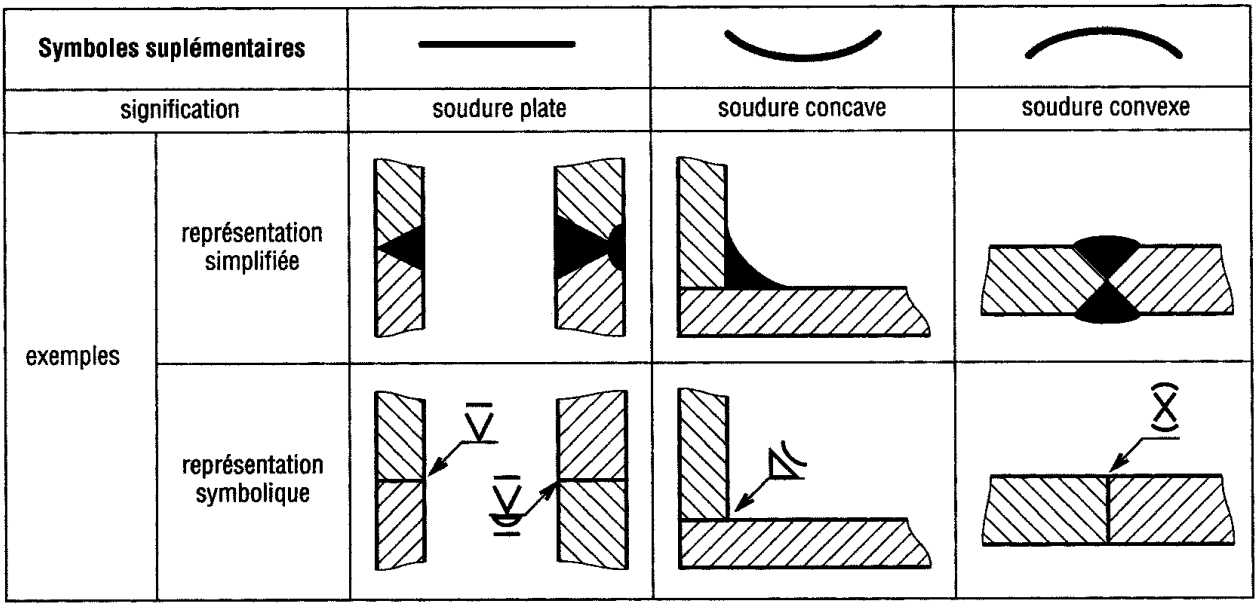


Figure I.7 : Quelques formes de soudures.

I.5. Défauts de soudure

En général, les cordons de soudures constituent des hétérogénéités géométriques et matériel par rapport à l'acier, ce qui peut donner naissance à des irrégularités (ou défauts) lors de l'opération de soudure. Cependant, il existe plusieurs types de défauts de soudure, qui sont des défauts: de soufflure, de fissure, d'inclusion, de manque de fusion (ou collage), de pénétration, d'effondrement, de morsure et caniveaux ainsi que des défauts géométriques du cordon (convexité, concavité, alignement).

Dans ce qui suit, on se limitera à donner quelques exemples illustrés.

I.5.1. Défauts de soufflure

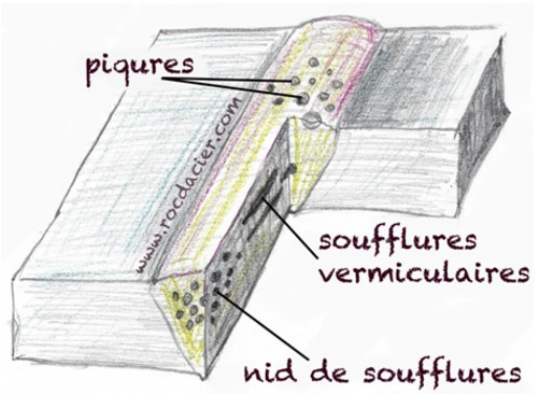


Figure I.8 : Défaut de soufflure.

Les soufflures (cavités) sont des défauts fréquents. Ce sont souvent des bulles de gaz enfermées dans le cordon de soudure. Les piqûres sont des soufflures débouchantes, donc visibles en surface (voir

fig. I.8).

I.5.2. Défauts de fissures

Les fissures sont des ruptures du matériau. Ces fissures peuvent se trouver aussi bien dans le métal de base que dans le cordon de soudure, dans la zone affectée thermiquement ou la zone de liaison (voir fig. I.9).

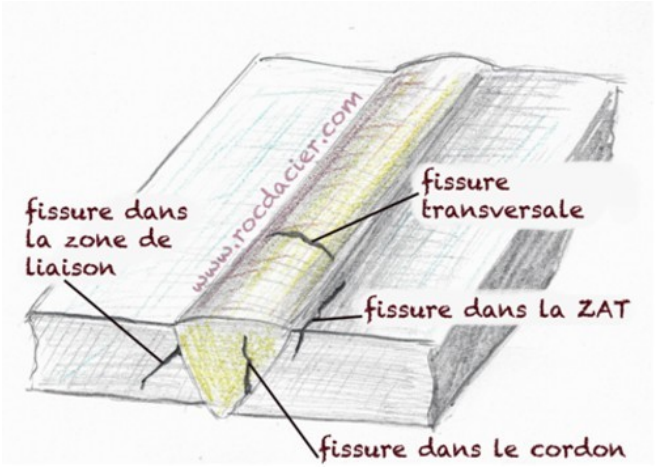


Figure I.9 : Défauts de fissures.

I.6. Contrôle qualité du cordon de soudure des pipes

Dans cette partie, on s'intéressera au contrôle qualité des pipes juste après leur fabrication. Rappelons qu'au niveau du groupe Industriel IMETAL- ALFAPIPE, le contrôle qualité des pipes est fait sur place, juste après la fin de l'opération de soudage, le pipe produit est soumis à un contrôle. Pour cela, un échantillon du cordon de soudure est prélevé directement du pipe (application du C.D.[2]). Cette technique a été utilisée pendant longtemps au niveau des labo d'ALFAPIPE et va bientôt être remplacée par d'autres techniques, qu'on citera plus bas dans ce mémoire. Entre autre, la partie pratique développée dans ce mémoire sera utilisée comme une nouvelle technique de contrôle qualité des pipes.

Il est à noter que le rôle principal de l'opération contrôle qualité des pipes est en fait, d'éliminer les pipes, présentant des défauts (ou des zones mal soudées), qui selon des normes internationales telles que API 5L, peuvent être rejetés. En d'autre termes, l'emploi d'un pipe défectueux dans un pipeline, diminue considérablement la durée d'exploitation du pipeline à long terme, qui est en moyenne de l'ordre de 20 à 40 ans.

2 - C.D. - Contrôle Destructif.

I.6.1. Critères d'acceptation d'une soudure

Comme toute autre opération, le contrôle qualité possède certaines caractéristiques (paramètres) qui ont une grande influence sur la sécurité et la fonctionnalité du produit :

- Fréquence de contrôle : systématique ou par prélèvement.
- Le type de contrôle : non destructif, destructif (parfois appelé « essai »).
- La méthode de contrôle : par mesure, par comparaison (défauthèque), par appréciation (contrôle visuel par exemple).
- Les moyens de contrôle à utiliser : appareil de mesure, référentiel.
- L'entité qui réalise le contrôle : personnel de fabrication (autocontrôle), personnel spécialisé, personnel d'encadrement, machine (automatisation du contrôle).

Dans le cas des pipes en Acier, les critères cités ci-dessous, sont explicitement détaillés dans la norme internationale.

I.6.2. Méthodes de contrôle qualité

Dans l'industrie de soudage il existe plusieurs techniques de contrôle qualité. Dans le cas de soudure sur tubes en acier inoxydable (bout à bout). On pourra faire un examen visuel et un examen macrographique, pour détecter les défauts internes. Dans certains cas l'éprouvette réalisée pourra faire l'objet d'une radiographie. Parmi les méthodes les plus utilisés, on citera :

I.6.2.1. Contrôle visuel et dimensionnel

Ce type de contrôle est dit « Non Destructif ». Il consiste à vérifier visuellement à l'œil nu, où à l'aide d'une caméra, un échantillon du cordon de soudure pour détecter la présence d'éventuels caniveaux, surépaisseurs des soudures bout à bout, valeur de gorge des soudures d'angles et défauts de surface.

En plus du contrôle visuel du cordon de soudure [5], un contrôle dimensionnel est effectué sur le pipe sous test comme dans la figure I.10.

Figure I.10 : Contrôle dimensionnel d'un pipe: mesure d'épaisseur, de diamètre et de longueur.

I.6.2.2. Contrôle par radiographie (rayons X et ɣ)

Figure I.11: Contrôle par radiographie de pipe.

Cette technique de contrôle non destructif est utilisée dans l'industrie lourde, notamment dans les centrales nucléaires et les chantiers navales et pétroliers. L'avantage de cette technique est de fournir des informations directement exploitables sur l'intérieur des matériaux. Son inconvénient majeur est l'exposition dangereuse aux rayons ɣ [5]. Elle est aussi exploitée lors du contrôle qualité des pipes après leur fabrication comme dans la figure I.11.

I.6.2.3. Contrôle par Ultrasons

Le contrôle par ultrasons (voir fig. I.12) est basé sur la transmission, la réflexion et l'absorption d'une onde ultrasonore se propageant dans l'échantillon du cordon de soudure. Cette méthode présente une résolution spatiale élevée et la possibilité de trouver des défauts aussi bien dans le volume de métal qu'en sa surface. L'avantage de ce type de contrôle est qu'il donne le résultat en temps réel pour des pièces de forte épaisseur. Par contre, cette méthode présente une faible sensibilité à la détection de porosité [5].

Figure I.12 : Inspection par Ultrasons.

I.7. Contrôle qualité

Vu l'importance du processus de contrôle qualité en industrie, plusieurs solutions ont été proposées, à travers divers travaux de recherche, afin d'améliorer les résultats. Dans ce sens nous pouvons citer les travaux suivants :

Dans le mémoire [6], l'auteur s'est particulièrement intéressé à l'état des aciers formant les pipelines dédiés au transport des hydrocarbures. Ces pipelines sont sollicités par des phénomènes de corrosion par le sol où ils sont enfouis. Les méthodes de détection des défaillances des aciers se font à l'aide d'un contrôle non destructive par Ultrason. Il permettent ainsi d'évaluer la durée de vie restante des pipes corrodés en utilisant le modèle de Weibull.

Dans le mémoire [7], l'auteur a étudié l'endommagement des aciers pour pipelines par fatigue et à la limite de son élasticité qui dépend de plusieurs paramètres de nature métallurgiques, des propriétés mécaniques et des conditions de service (pression, contraintes au niveau des parois des tubes).

Plusieurs méthodes ont été appliquées pour mettre en évidence les différents grades d'aciers API 5L. Parmi ces méthodes : Harter, modèle de propagation de Walker afin de prédire la durée de vie résiduelle en matière de fatigue et la vitesse de fissuration. A travers les joints soudés, la fissuration dépend de l'état de contrainte générée lors du soudage et de l'état d'expansion des tubes.

Dans l'article [8], une étude de caractérisation de l'acier API-5L-X70 est réalisée par des essais mécaniques de contrôle destructifs, de traction, d'analyse de la microstructure, de dureté, de résilience et enfin, une étude de validation par éléments finis est faite en utilisant un code de calcul ANSYS, qui est présentée dans le cas d'un pliage afin de valider les lois de comportements.

I.8. Conclusion

Dans ce premier chapitre, nous avons présenté brièvement le procédé de fabrication des pipes, les notions de bases de la soudure des pipes avec les différentes méthodes de détection des défauts de soudure (en utilisant les méthodes de contrôle CND : contrôle visuel, par ultrason ou par radiographie). Nous avons aussi présenté quelques notions et critères sur le contrôle qualité du cordon de soudure des pipes.

Par ailleurs, vu la forte demande sur les ressources énergétiques dans le monde, les pipelines, installés sur site, sont vite devenus un moyen sûr pour transporter le gaz et les produits pétroliers, ce qui a mené à une augmentation fulgurante du volume de production des pipes. Et pour répondre, en partie, à ce besoin, nous avons constater qu'il était nécessaire d'automatiser l'opération de contrôle qualité du cordon de soudure des pipes juste après leur fabrication. Pour ce faire et dans le deuxième chapitre nous allons nous intéresser au contrôle visuel des pipes. Cette méthode de contrôle qualité est

beaucoup plus proche de notre formation (Master en Électronique des Systèmes Embarqués). Pour cette raison , nous allons voir comment réaliser un système embarqué intelligent et, plus précisément, voir l'intérêt et la manière d’utilisation des caméras industrielles pour automatiser l’opération de contrôle qualité des pipes.

Chapitre II: Généralités sur les caméras industrielles dédiées au contrôle qualité

II.1. Introduction

Une des méthodes de contrôle qualité du cordon de soudure des pipes, est accomplie à l'aide d'une inspection visuelle, qui est appliquée à plusieurs niveaux lors du processus de fabrication. Ceci dit, le contrôle qualité peut être en temps réel au moment de la soudure des tôles d'acier (voir fig.II.1), ou en différé, c'est à dire après soudage (voir fig.II.2).

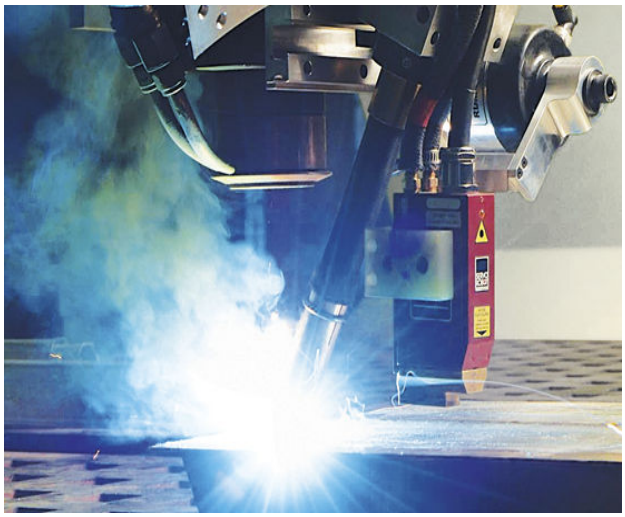


Figure II.1 : Inspection visuelle en temps réel.

Figure II.2 : Inspection visuelle d'un échantillon après fabrication.

Depuis quelques années, grâce au développement des caméras laser, il est devenu possible de déterminer les profils des cordons de soudure soit en cours de fabrication, soit durant l'opération de contrôle qualité après soudage. Dans le premier cas les informations, traitées en temps réel, permettent d'agir « en temps réel » pour corriger les paramètres commandant la machine à souder. Dans le second cas, l'inspection visuelle permet de détecter les différents défauts de soudure, tel qu'ils sont présentés dans le premier chapitre et au final, la décision d'accepter, de rejeter ou de retoucher le pipe sous contrôle, est prise par l'opérateur humain.

Dans ce contexte, les techniques de vision et d'imagerie jouent un rôle important. L'amélioration permanente de ces techniques assure une performance de haute qualité du processus de fabrication grâce à un positionnement précis et à un contrôle continue.

Des systèmes avancés incorporant de multiples méthodes de vision et de traitement d'images, fournissent des solutions robustes à des situations et problèmes complexes dans des applications industrielles. Un large éventail d'industries, notamment aérospatiale, automobile, électronique, pharmaceutique, biomédical, semi-conducteur et agroalimentaire, ont bénéficié de ces avancées technologiques récentes.

En effet, plusieurs techniques de vision et de traitements d'images ont permis la simplification de

l'inspection industrielle pour augmenter la qualité des produits. Parmi ces méthodes de vision et de traitements d'images, citons les:

- Techniques numériques d'acquisition et de stockage d'images .
- Techniques numériques de segmentation et de détection de contours dans une image acquise.
- Techniques de mesures de morphologie.
- Techniques d'apprentissage et de reconnaissance des formes et d'objets
- Techniques de création, de stockage et de gestion de bases d'images originales et traitées.

La plupart des industries utilisent aujourd'hui la vision artificielle pour automatiser les opérations de contrôle qualité des produits. En général on considère que la vision artificielle se prête bien au contrôle qualité lorsque :

- Les tâches sont répétitives : telles que la recherche de défauts, mesure de dimensions, comparaison avec des valeurs de référence...etc.
- Les tâches exécutées sont simples et bien définies **:** les scènes à observer sont connues à l'avance pour extraire des informations.
- Le capteur de vision peut changer radicalement de type d'inspection par simple reprogrammation, ce qui permet une très grande variété d'utilisation.
- On peut faire des mesures non destructives sans contact, avec une vitesse et une précision appréciable.
- On peut garantir une objectivité de certaines mesures et de la répétabilité de l'apparence des objets (constance de la couleur, de la forme, … etc).
- Un système de vision artificielle automatique permet d'aider les opérateurs humains dans des tâches fastidieuses et répétitives, qui requièrent une grande attention et concentration[8].

II.2. Application de la vision artificielle en industrie

Les applications de vision artificielle (par exemple : voir fig. II.3) sont aujourd'hui nombreuses et sont ouvertes à tous les secteurs de l'industrie. En effet, les progrès technologiques ont permis un élargissement considérable du champ d'application de la vision artificielle et de ce fait, on entend souvent parler de la vision industrielle, qui n'est autre qu'une application de la vision artificielle (ou vision par ordinateur), conçue spécialement pour apporter des solutions d'automatisation des lignes de production dans les usines [9].

Il est à noter que la vision Industrielle est aussi une application de contrôle non destructif de processus de fabrication.

Figure II.3 : Exemples d'application de la vision artificielle en industrie.

par ailleurs, la principale mission de la vision industrielle dans le cas d'un contrôle qualité, est de doter les machines de la capacité de « VOIR » afin d'automatiser les tâches de contrôle qualité ou de processus dans le but d'optimiser le temps et les coûts de production. Autrement dit, cette automatisation vise principalement à augmenter la cadence de la production, de la rendre plus fiable, d'assurer la traçabilité des produits finis et de minimiser les accidents dans les usines, qui sont, généralement, dus au non respect des normes sécuritaires par les ouvriers.

II.3. Éléments d'un système de vision industrielle

Un système de vision industrielle est constitué typique de plusieurs éléments, qui sont :

1. La caméra est l'élément principal de la vision industrielle et qui sert à prendre des images des produits sur les lignes de production.
2. L'éclairage : il est important de bien choisir l'intensité d'éclairage, pour mettre en évidence les

défauts du produit sous contrôle.

3. Le logiciel de traitement d'image prises : installé sur un ordinateur de visualisation ou embarqué dans le système relié avec un automatisme d'éjection en cas de refus du produit sous contrôle.
4. Machines industrielles.

II.4. Avantages d'un système de vision industrielle

L'avantage d'avoir un tel système de vison industrielle est qu'il est possible avec le même système de réaliser un contrôle en continu sur tous les produits de même type, ou d'effectuer plusieurs contrôles, qui sans ce système, nécessiteraient l'emploi de plusieurs machines.

Pour résumer, l'emploi d'un système de vison industrielle permet de faire:

- un contrôle de conformité d'assemblage : qui a pour but de vérifier l'absence ou la présence d'éléments constituant le produit à fabriquer, ainsi que leur positionnement et leur orientation.
- un contrôle d'aspect, c'est à dire examiner les états de surface afin de détecter des défauts d'aspect comme : les rayures, les griffures, les trous, les taches et les défauts de nuances de couleurs ou de texture.
- un contrôle dimensionnel, qui consiste à mesurer les dimensions d'une pièce (longueur, diamètre, profondeur, angle ou une géométrie particulière).
- le comptage et le tri de pièces peuvent également être effectués par un système de vision industrielle.

II.5. Fonctionnement d'un système de vision industrielle

En plus des facteurs standards tel que: le coût, la flexibilité, la rapidité et l'interaction de l'application avec différents systèmes, le système de vison industrielle permet de matérialiser les quatre principes essentiels au bon fonctionnement de tout processus de vision industrielle:

1. Acquisition d'images.
2. Traitement des images.
3. Sauvegarde d' images.
4. Décision et action.

Les possibilités d'acquisition, de stockage et de traitement des données permettent d'envisager une meilleure connaissance des paramètres de soudage et, en final, une meilleure maîtrise de la qualité des produits.

Dans ce qui suit, on détaillera les points cités ci-dessous.

II.5.1. Acquisition d'images

Par définition, un système d'acquisition d'images permet d'obtenir une image numérique, représentant l'objet à contrôler. En outre, l'acquisition d'images du cordon de soudure se fait à l'aide d'une caméra. l'image obtenue, peut être par la suite, sauvegardée dans une mémoire et, éventuellement, accessible via un programme applicatif en vue d'un traitement d'image.

En général, le système d'acquisition d'images peut être décomposé en plusieurs parties :

- La partie optique.
- Le capteur.
- Le transport des données images (format et support) vers un ordinateur de bord d'une machine industrielle ou, par exemple, vers une tablette ou PC.

II.5.2. Traitement d'images

Par définition, un traitement d'images numériques est l'ensemble des techniques permettant de modifier une image numérique afin d'améliorer ou d'en extraire des informations jugées « pertinentes ». Ces techniques sont principalement utilisés dans différents domaines tel que : la médecine, l'aérospatiale, la sécurité et les processus de fabrications.

Le traitement d'images peut être fait en plusieurs étapes, qui sont:

- Élimination par filtrage le bruit produit par l'acquisition ou par les conditions de prise de vue.
- La détection de contours qui permet de réduire de manière significative la quantité de données existantes dans une image, tout en conservant les informations utiles.

Autres types de traitement d'image : la binarisation, l'égalisation d'histogramme, la correction d'exposition….etc. Dans le cas d'une inspection des pipes, une reconnaissance de forme du cordon de soudure est nécessaire. Pour cela, nous utiliserons en plus du filtrage du bruit, une opération de détection de contours, qui sera décrite plus en détail dans le chapitre IV.

II.5.3. Sauvegarde d'images

Cette opération est nécessaire, lorsqu'on veut faire un traitement d'image après acquisition, puisque l'emploi de caméra sous-entend l'acquisition d'une vidéo, qui n'est autre qu'un ensemble de scènes ou images consécutives pour former le mouvement d'un objet.

Sauvegarder une images consiste à réserver un espace mémoire à l'image acquise pour pouvoir y effectuer des opérations de traitement d'image.

Le support mémoire sur lequel, l'image est sauvegardée peut être une carte mémoire de type SD, un disque dur HDD/SSD, ou autre support de stockage de données. Le choix de ce support dépend du systèmes d'acquisition.

II.5.4. Prise de décision

Dans le cas de contrôle qualité du cordon de soudure, la prise de décision sur l'état du pipe doit être basée sur des mesures et calculs, qui définiront, jusqu'à quel point on peut tolérer certains défauts de fabrication. En général, les mesures et calculs obtenues, sont comparés à ceux, définis dans les normes internationales. Par exemple, dans le cadre du contrôle du cordon de soudure, la norme API-5L est utilisée.

II.6. Types de caméras

Il existe sur le marché plusieurs types de caméras [9]. Elles sont présentées dans le tableau II.1.

Type de caméra	Fonctionnalité
Caméra analogique	Enregistre ou transmet un signal électronique analogique
Caméra numérique	Enregistre ou transmet un signal électronique numérique.
Caméra argentique	Utilise un film photographique comme support d'enregistrement.
Caméra d'astronomie	Conçue pour des temps de pose longs et présente une haute sensibilité dans le domaine du visible mais aussi dans les domaines des infrarouges et des ultraviolets.
Caméra de détection de mouvement	Fonctionne par analyse continue des images.
Caméra endoscopique	Est un système médical d'exploration indirecte de certaines parties internes du corps humain ou un système industriel d'exploration à distance de l'intérieur d'une machine.
Caméra intelligente	**Est un système de vision industrielle qui capture des images et les traite.**
Caméra linéaire	Présente un capteur doté d'une seule ligne de pixel, contrairement à une caméra matricielle. Elle est largement utilisée dans le domaine de la vision industrielle.
Caméra IP	Caméra de vidéosurveillance utilisant le protocole IP.
Caméra sans fil	transmet un signal électrique par l'intermédiaire d'ondes radioélectriques grâce à un émetteur et une antenne.
Caméra thermique	permet de capter les différents rayonnements infrarouges afin de détecter des sources de chaleur.
Dashcam	Est installée dans un véhicule pour enregistrer ce que le conducteur voit, une installation prisée par les assureurs en cas d'accident.

Webcam	Caméra conçue pour être utilisée comme un périphérique d'ordinateur afin de transmettre une vidéo au travers d'un réseau, typiquement Internet.

Tableau II.1 : Types de caméras et leurs fonctions.

Rappelons que dans le cadre de ce mémoire, notre attention sera focalisée sur les caméras intelligente industrielles. puisqu'elles sont majoritairement utilisés dans les domaines suivants: automatisation industrielle, fabrication de composants électroniques, inspection d'emballages et agroalimentaire, inspection de semi-conducteurs, trafic et sécurité routière, inspection optique automatique, … etc.

II.7. Critères de choix d'une caméra industrielle

Les caméras industrielles sont préconisées dans tout contexte de flux et de cadences élevés. Elles différent selon le secteur d'activité et d'application souhaitée.

Pour réaliser un système de vision industrielle, il est important de bien choisir la caméra d'un point de vue : type de capteur, résolution et d'interface de communication. Nous allons détaillé dans ce qui suit les points cités ci-dessus.

II.7.1. Capteurs CCD ou CMOS

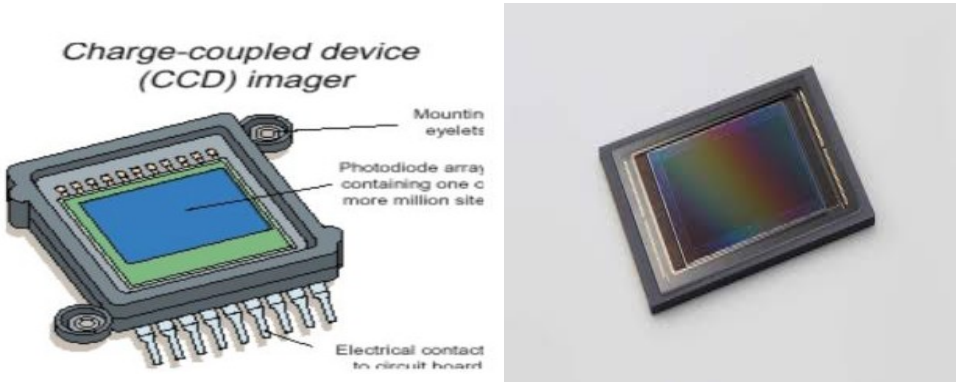


Figure II.4 : Capteur CCD et CMOS.

Qu'ils soient de type CCD ou CMOS (voir fig. II.4), ce sont tous deux des capteurs photographiques convertissant une image de départ, issue d'un rayonnement électromagnétique en un

signal analogique (ou numérique) à la fin.

Le principe de base d'un capteur repose sur l'effet photoélectrique, qui permet à un photon incident d'arracher des électrons à chaque élément sensible (appelé photo-site) d'une matrice de petits capteurs élémentaires.

Les capteurs CCD, sont dans la plupart des appareils photo numérique. Ce sont des dispositifs à couplage de charge, ils captent la lumière sur les petits photo-sites situés à leur surface. Ils tirent leur nom de la manière dont le nombre de charges est lu après une exposition à la lumière.

Les capteurs CDD sont utilisés dans applications : médicales, commerciales, scientifiques et militaires. Il existe deux types de capteurs CDD: linéaires et matricielles :

1. Capteurs CCD linéaires: sont essentiellement utilisés dans les systèmes de reconnaissance de caractères, de mesures sans contact, de reconnaissance de formes et de détection de défauts.
2. Capteurs CCD matriciels: ils ont une taille de 512x512 pixels jusqu'à 4096x4096 pixels. Ils sont dédiés aux applications vidéo instantanée ou vidéo rapide et de métrologie.

La technologie CMOS a été inventée bien avant les CCDs, mais très peut utilisée, puisque elle n'était pas aussi fiable que les capteurs CCD. Aujourd'hui, la technologie CMOS est au cœur de tous les circuits intégrés modernes. Elle a l'avantage d'être miniaturisée, ce qui engendre de faibles coûts et consommation en énergie [10].

Les tableaux II.2 et II.3 résument les caractéristiques des capteurs CCD et CMOS.

Caractéristique	CCD	CMOS
Signal issu du pixel	Change	Tension
Signal issu de la carte	Tension (analogique)	Bits (numérique)
Signal issu de la caméra	Bits (numérique)	Bits (numérique)
Facteur de remplissage	Élevé	Modéré
Niveau de bruit	Faible	Modéré à élevé
Complexité du système	Élevé	Faible
Complexité du capteur	Faible	Élevé
Composants de la caméra	Circuit intégré +cartes+lentilles	Carte + lentilles

Tableau II.2 : Comparaison des caractéristiques des capteurs CCD et CMOS.

Performances	CCD	CMOS
Dynamique	Élevé	Modéré
Uniformité	Élevé	Faible à modérée
Déclenchement uniforme	Rapide	Limité
Vitesse	Modéré à élevé	Élevé
Fenêtrage	Limité	Courant
Polarisation et synchronisation	Multiple, tension élevée	Simple, tension basse

Tableau II.3 : Comparaison des performances des capteurs CCD et CMOS.

II.7.2. Résolution de la caméra

La résolution est calculée à partir du nombre de pixels en longueur multiplié par le nombre de pixels en largeur. De cette résolution dépendra le format de sortie en visuel sur un écran ou en impression. La taille maximum d'une scène couverte à une résolution donnée dépend uniquement de la résolution de la caméra. Par conséquent, les caméras ayant des résolutions plus élevées peuvent couvrir de plus grandes zones. On choisira la résolution par rapport à ce que l'on doit contrôler. Si on doit contrôler une petite zone, on prendra une résolution moins élevée, tandis que si l'on veut contrôler une plus grande zone, on choisira une caméra avec une résolution élevée.

Le tableau suivant illustre Format de l'image en fonction de sa résolution.

Format	Résolution	Rapport	Taille de l'image
QVGA	320 x 240	4:3	76800
VGA	640 x 480	4:3	307200
SVGA	800 x 600	4:3	480000
XGA	1024 x 768	4:3	786432
SXGA	1280 x 1024	5:4	13107200
UXGA	1600 x 1200	4:3	1920000
QXGA	2048 x 1536	4:3	3145728
QSXGA	2560 x1920	4:3	4915200

Tableau II.4 : Dépendance de la résolution du format de sortie (Taille de l'image).

II.7.3. Interfaces de communication des caméras

Avec la multiplication des capteurs de vision sur les lignes de production, la gestion des capteurs devient une démarche essentielle. Les capteurs de vision doivent communiquer avec des PC distants (pour leur transmettre par exemple les informations d'acceptation ou de rejet, ou transférer les images des pièces défectueuses, … etc).

Pour cela, il faut s'assurer que le capteur de vision prend en charge une grande gamme de protocoles réseau standards. En effet, pour connecter les capteurs de vision à des automates programmables ou à des robots, il faut s'assurer qu'ils prennent en charge le protocole Ethernet/IP, qui permet aux capteurs de vision d'être connectés aux automates via un seul câble Ethernet (ce qui supprime les câblages complexes et les passerelles réseau trop chères) [9].

Si l'on choisit un capteur de vision intelligent, le fournisseur doit proposer un système autonome qui ne nécessite pas de PC, que ce soit durant la procédure de configuration ou de contrôle en ligne de production. Un capteur réellement autonome doit fournir un ensemble de fonctionnalités Plug-and-Play permettant de configurer ou de modifier rapidement les paramètres du système sans ajouter de module supplémentaire. Il doit aussi permettre, sans utiliser de PC, de brancher un moniteur pour visualiser les images en temps réel [10].

II.8. Caméras intelligentes en milieu industriel

Dans l'industrie, là où les PC sont souvent perçus comme des systèmes extérieurs, difficiles à incorporer aux équipements existants, les caméras intelligentes peuvent être vues globalement comme des capteurs s'intégrant facilement dans les processus de contrôle automatisés. En outre, celles-ci présentent l'avantage d'être compactes et de ne pas nécessiter la présence de multiples périphériques d'interface souvent redondants. Aussi, pour certaines applications nécessitant plusieurs captures d'images asynchrones, les caméras intelligentes sont la plupart du temps, mieux adaptées qu'un système informatique complet.

Par ailleurs, les caméras intelligentes combinent de puissants processeurs embarqués et des capteurs d'imagerie dans un système de vision tout-en-un. Les caméras intelligentes peuvent également inclure des Entrées/Sorties numériques intégrées et des options de communication industrielle lorsqu'il s'agit d'une communication dynamique en temps réel et une intégration avec les dispositifs d'automatisation industrielle, y compris les automates programmables, les HMI, la robotique et les capteurs [11].

Une caméra intelligente possède une partie électronique qui permet d'acquérir, de stocker des images, de traiter de l'information et de communiquer avec les systèmes environnants (réseau,

automates, opérateurs, … etc.).

De nos jours, la majorité des fabricants de caméras industrielles proposent aussi des caméras intelligentes dédiées à des applications variées, qui sont :

- Contrôle de conformité (nombres d'éléments et leurs positions).
- Mesures sans contact.
- Identification et tri de produits.
- Lecture et vérification de marquage (caractères, code barre).
- Commande robot (détection de position et d'orientation 2D et 3D).
- Contrôles de processus continu (exp. vérification de soudure).
- Contrôles de mouvements et de déplacements (vidéo surveillance).
- Contrôles biométriques (cartes, passeports biométriques, … etc).

II.9. Caméras intelligentes

Dans l'article [12] proposé par B.HEYRMAN, M.PAINDAVOINE, R.SCHMIT et L. LETELLIER, un nouveau concept de caméra intelligente a été proposé. La caméra est dotée de capacités de traitement vidéo en temps réel.

L'architecture de la caméra intelligente, est composée d'un capteur CMOS, qui autorise l'accès aléatoire aux pixels et est doté de sorties parallèles qui sont capables de fournir des régions d'une image. Le tout est relié à une mémoire RAM à double ports. Un réseau de processeurs à usage générale, combinés à plusieurs unités de traitement permettent l'exécution des opérations d'addition, de soustraction et d'autres opérations logiques. Ces unités de traitements peuvent réaliser des fonctions de traitements d'image comme : la convolution (filtrage) et le calcul d'histogramme. Les données sont transférées à l'aide d'un réseau de type « Cross-bar » qui sont peu extensibles en terme de coût mais deviennent bloquant lorsque le flux de données est très grand. Tout le système est commandé par un contrôleur qui génère les signaux de commandes et gère l'acquisition et le stockage des images en mémoire. L'architecture proposée a été testée sur trois applications de traitement d'image, à savoir : « détection de mouvements », « reconnaissance de visage » et « stabilisation d'image ». L'architecture proposée a été modélisée en Système-C et non pas en VHDL**.** Les résultats obtenus, restent liés aux performances et à la capacité de traitement des processeurs à usage générale.

II.10. Conclusion

Dans ce deuxième chapitre, nous avons exposé les différents domaines d'application de la vision artificielle dans l'industrie, nous avons donné une description succincte sur les différents types de caméras en fonction de la nature du capteur d'image (CCD ou CMOS), de la résolution de l'image

souhaitée et de l'interface de communication caméra-PC. En plus, nous avons donné les critères de choix de la caméra selon le domaine d'application. Dans le cas d'un contrôle qualité des pipes, nous avons choisi d'utiliser une caméra intelligente, qui peut embarquer un capteur CMOS (pour acquisition des images) et un circuit FPGA, permettant ainsi d'embarquer les algorithmes de traitement d'image pour traiter les images du cordon de soudure des pipes.

Notons que sur le marché, on trouve plusieurs fabricants de caméras intelligentes, proposant des solutions matérielles et logicielles de contrôle qualité, mais la nature de ces solutions reste spécifique à une application bien précise et, qu'il n'était pas possible d'effectuer diverses contrôles qualités avec le même matériel. C'est pour cela que nous voudrions arriver à réaliser un système où on peut facilement changer le type de caméras et d'algorithme de traitement d'image pour pouvoir faire différents types de contrôle qualité.

Après avoir fait une investigation sur les différents types de caméras intelligentes existantes sur le marché [8], nous avons trouvé que les caméras intelligentes, commandées par un circuit FPGA, sont très peu utilisées dans l'industrie, à cause de leur prix, qui n'est pas très compétitif (plutôt très élevé) par rapport à d'autres types de caméras industrielles intelligentes, ceci est en partie expliqué par la possibilité de reconfiguration du circuit FPGA, pour effectuer différents types de traitement d'image et par la technologie du circuit FPGA en lui même, comparé à un processeur à usage général, qu'on trouve dans la plus-part des caméras intelligentes [11].

Il est à noter que les dernières générations des circuits FPGA [12] ont, en plus de la logique combinatoire, un processeur à usage générale. C'est pour cela notre choix s'est porté sur le kit caméra d'Avnet, qui allie à la fois un capteur de type CMOS et qui peut être commandé par un circuit FPGA.

Chapitre III: Commande d'une caméra industrielle par carte FPGA pour l'acquisition d'images du cordon de soudure

III.1. Introduction

Le contrôle visuel de l'état de la soudure des pipes juste après leur fabrication, est accompli en plusieurs étapes. Pour cela dans ce troisième chapitre nous nous intéresserons à l'acquisition d'images à l'aide d'une caméra industrielle intelligente, où la commande de cette dernière ce fait par un circuit FPGA.

Il est à noter qu'en général, la commande de la caméra, d'un point de vue électronique, dépend essentiellement du matériel qui la compose et avec qui elle doit communiquer. Autrement dit, il existe plusieurs variantes de caméras sur le marché et il es donc difficile de définir, au préalable, un modèle universel de commande de caméra industrielle intelligente.

Nous allons commencé dans ce chapitre par une présentation des circuits FPGA, des outils de développements modernes, ainsi que la méthodologie de conception. On présentera aussi une caméra intelligente industrielle**,** où nous allons donné une description succincte du matériel, composant cette dernière. Pour ce faire, nous avons choisi d'utiliser le kit de développement de vision intégrée « EMBV-Python1300-C », fabriquée par Avnet, comme exemple de commande de caméra intelligente industrielle. Ce kit est spécialement conçu pour répondre aux besoins des applications vidéo industrielles personnalisées, en matière d'acquisition et de traitement d'images en temps réel ou en différé.

En plus de l'acquisition d'image, le kit « EMBV-Python1300-C » nous permet de vérifier l'implémentation des algorithmes de traitement d'image, pour le contrôle qualité des cordons de soudure des pipes. Cette étape sera explicitement décrite dans le chapitre IV.

III.2. Description des circuits FPGAs

Les circuits FPGAs de Xilinx connaissent une évolution technologique importante depuis leur apparition. D'une famille à l'autre, on voit que l'évolution concerne spécialement le taux d'intégration en termes de portes logiques, de mémoires, de processeurs et circuits DSP dans un seul circuit FPGA.

Les circuits FPGAs sont programmables et reconfigurables, c'est-à-dire : l'implémentation de nouvelles fonctions est possible une fois sa fabrication terminée.

Les circuits FPGAs sont généralement composés de plusieurs types de ressources matérielles dont les blocs logiques programmables, de blocs d'entrées/sorties, des BRAM et des DSP. Ces éléments sont disposés par groupe et par colonne comme le montre la figure III.1.

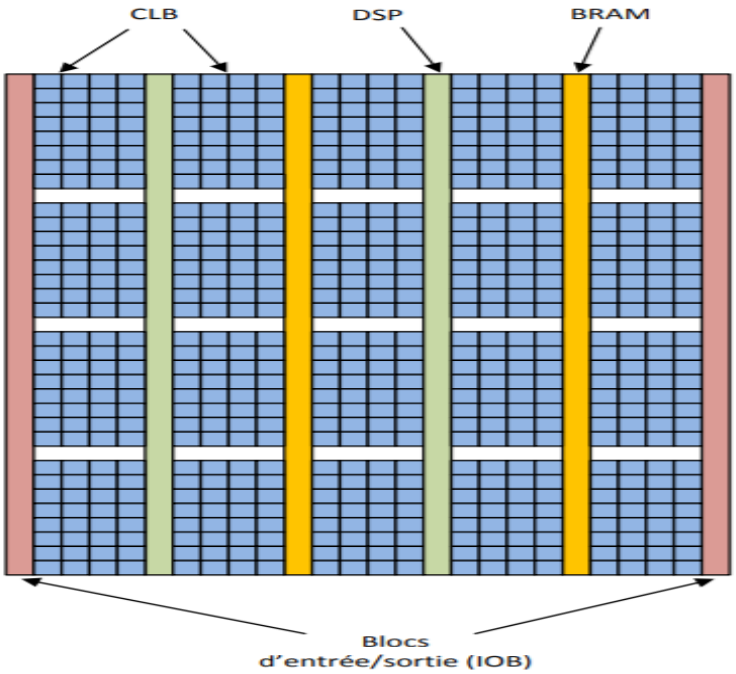


Figure III.1 : Architecture matérielle d'un circuit FPGA en général.

III.2.1. Blocs logiques configurables

Un bloc logique configurable (CLB) est un composant de base d'un FPGA. Ses ressources sont conçues pour la configuration et la programmation des fonctions logiques combinatoires et séquentielles. Les CLB contiennent de la logique pour la création de machine d'état et des LUT pour les fonctions logiques combinatoires. On retrouve aussi des bascules D et des multiplexeurs afin d'assurer l'acheminement et la synchronisation des données.

L'architecture des CLB dépend du circuit FPGA et de la technologie utilisée lors de sa fabrication. On peut citer comme exemple la famille Xilinx Virtex-4, ou les CLB se composent de deux LUT à quatre entrées comme on peut le voir sur la figure III.2.

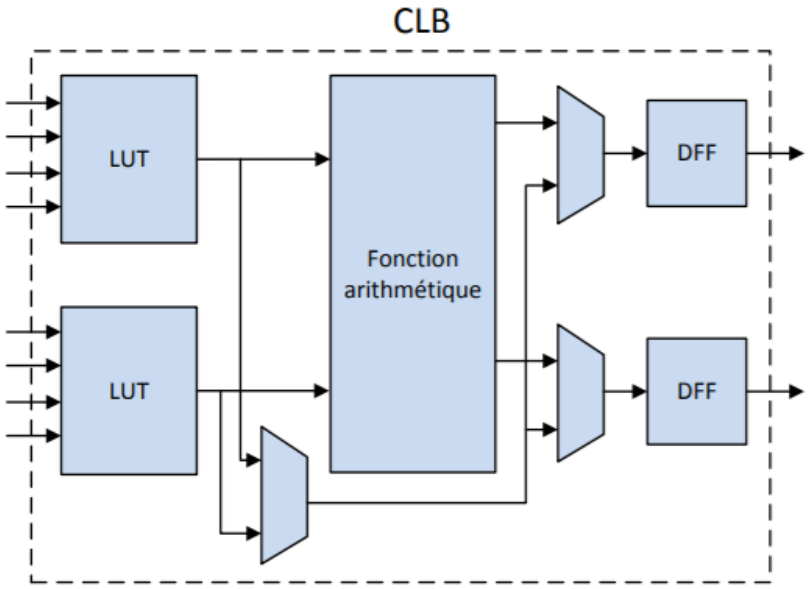


Figure III.2 : Elements d'un bloc CLB.

III.2.2. Les blocs d'entrées/sorties

L'interconnexion des éléments et fonctions internes du FPGA avec les ressources externes se fait à travers les I/OB. Chaque bloc d'entrée/sortie est dédié à une branche du composant et possède des caractéristiques propres à lui en mémoire, comme son niveau de tension et sens de communication. Les bus macro réalisent la communication interne entre les différentes fonctions du FPGA et la communication entre les modules reconfigurables et la partie statique du design.

III.2.3. Les blocs BRAM

Les blocs BRAM sont des mémoires internes du FPGA, elles permettent de stocker rapidement et proposent un temps d'accès plus rapide que les mémoires externes qui sont très avantageux pour le stockage des données entre les fonctions. Elles présentent plusieurs avantages dont la possibilité d'utilisation de deux horloges distinctes : une pour l'écriture et l'autre pour la lecture. Ceci est faisable grâce à la présence du double port (PORT A et PORT B).

III.2.4. Les blocs DSP

Les blocs DSP permettent la réalisation de fonctions telles que les multiplications, les additions, le multiplexage et le traitement de signaux. l'utilisation de ces blocs permet d'avoir un gain considérable en terme de temps d'exécution et ressources matérielles consommées.

III.3. Circuits FPGA de la famille Zynq-7000

La famille « Zynq-7000 » contient des circuits de type SoC développé par Xilinx. Chaque circuit de la famille Zynq-7000 est constitué d'une logique programmable PL et un système de traitement PS qui pilote la partie PL (voir fig. III.3). La partie logique programmable est développée par Xilinx, l'autre partie (PS) est un processeur RISC 32 bits, développé par la société ARM qui possède une mémoire intégrée et des interfaces d'interconnexions qui assures l'accès aux périphériques.

L'utilisation du Zynq-7000 comme solution pour certaines applications, offre une meilleure performance et présente une consommation d'énergie réduite par rapport à l'ASSP, ASIC ou même le PLD[13].

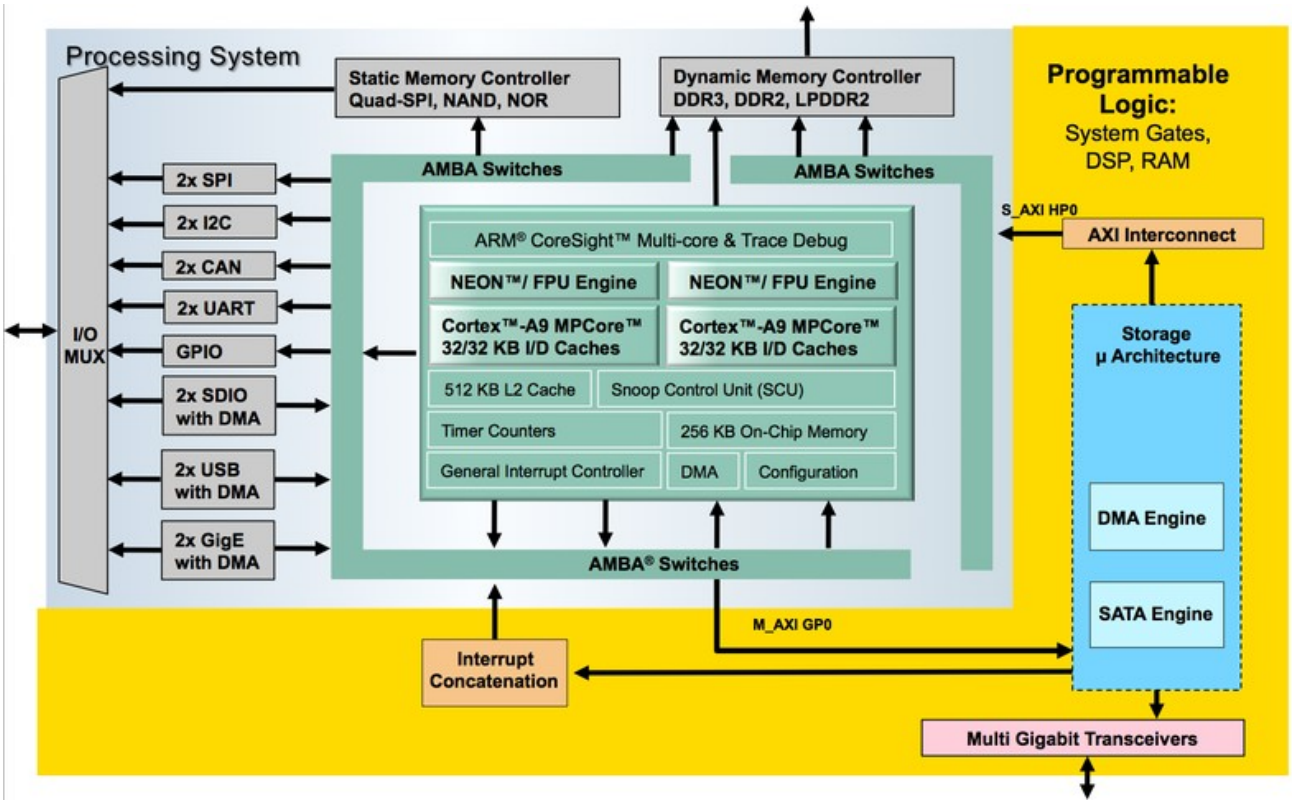


Figure III.3 : Architecture interne du circuit FPGA Zynq-7000.

La famille Zynq 7000 contient six circuits : Le Z-7030, Z-7045, 7-100, Z-7010, Z-7015 et Z-7020. Les trois premières utilisent comme PL le Kintex-7 de Xilinx, ces derniers sont destinés aux applications qui exigent des performances et un débit d'entrées/sorties relativement élevées. Les autres circuits contiennent l'Artix-7 comme PL, leur utilisation apparaît dans des applications limitées en termes de consommation électrique et de coût. On remarque très bien que la partie PL et ses ressources d'entrées/ sorties varient en fonction du circuit de la famille Zynq-7000. La partie PL est reconfigurable pour implémenter les IPCores et les modules matériels (Hardaware modules). En outre, la partie PS est divisée en plusieurs parties, à savoir :

- L'APU : c'est un processeur Dual-Core ARM Cortex 9 (double cœurs, voir fig. III.3), où

chaque cœur est appelé CPU0 et CPU1, il a la possibilité d'exécuter un système d'exploitation sur chaque cœur en mode AMP, ou sur les deux cœurs en mode SMP. Chaque cœur comprend un coprocesseur NEON 128 bits SIMD, qui est destiné aux applications intensives de traitement multimédia et du traitement du signal. On trouve ainsi en chaque cœur deux mémoires cache L1 de 32 kilo octets, une mémoire cache L2 associative de 512 kilo octets à 8 voies partagées par les deux cœurs, des horloges privées, SCU pour assurer la cohérence entre les deux niveaux de mémoires caches, 256 kilo octets de mémoire OCM SRAM à double port, un contrôleur d'accès directe à la mémoire DMA et d'un contrôleur d'interruption général (GIC) avec priorité et d'autres fonctionnalités.

- La mémoire : elle contient divers interfaces DDR, dotées de plusieurs fonctionnalités telle que la mise hors tension autonome durant les périodes d'inactivité, équipée aussi d'un planificateur de transaction qui a pour but d'assurer l'optimisation des données de la bande passante.
- Les périphériques et interfaces : Il existe en général 246 signaux d'entrées/sorties à usage générale - GPIO. Parmi ces 246 on distingue 54 signaux connus sous le nom d'entrées/sorties multiplexées - MIO qui assurent la connexion entre le Zynq-7000 et les périphériques extérieurs, et 192 signaux connues sous le nom Extended MIO (EMIO). Ces derniers, établissent une connexion entre la PS et la PL, appelés aussi interfaces PL-PS qui sont les interfaces AXI (maîtres/esclaves). Les esclaves AXI sont constituées d'un port ACP, 4 interfaces HP (High Performance) et de 2 interfaces GP (General Puropose). Les maîtres AXI sont deux interfaces maîtres GP. Il existe aussi des contrôleurs pour Ethernet Gigabit, USB, cartes MicroSD, SPI, UART.
- Les interconnexions. Le responsable de toutes les communications de données au sein de la partie PS est l'interconnexion AXI qui est composée de commutateurs qui transmettent des données entre les interfaces et ressources interne du PS. Les interconnexions suivantes : « OCM interconnect, Master interconnect et Central interconnect » sont considérées comme des sous-interconnections d'interconnexion sur puce, elles sont construites sur la base de la norme NIC-301 (Corelink Network interconnected), développée par ARM, elle définit un réseau AMBA qui connecte de nombreux maitres AXI avec des esclaves AXI, ou qui relie des interfaces AXI avec un bus AMBA en utilisant plusieurs commutateurs [13].

III.4. Outils de développements de Xilinx

L'environnement VIVADO de Xilinx (voir fig. III.4) a été conçus dans le but d'assister l'utilisateur lors de la conception de tout le système. VIVADO est un nouvel outil de développement plus performant que son prédécesseur ISE. VIVADO permet de réaliser toutes les étapes de

développement d'un projet; de la conception jusqu'à vérification sur carte FPGA.

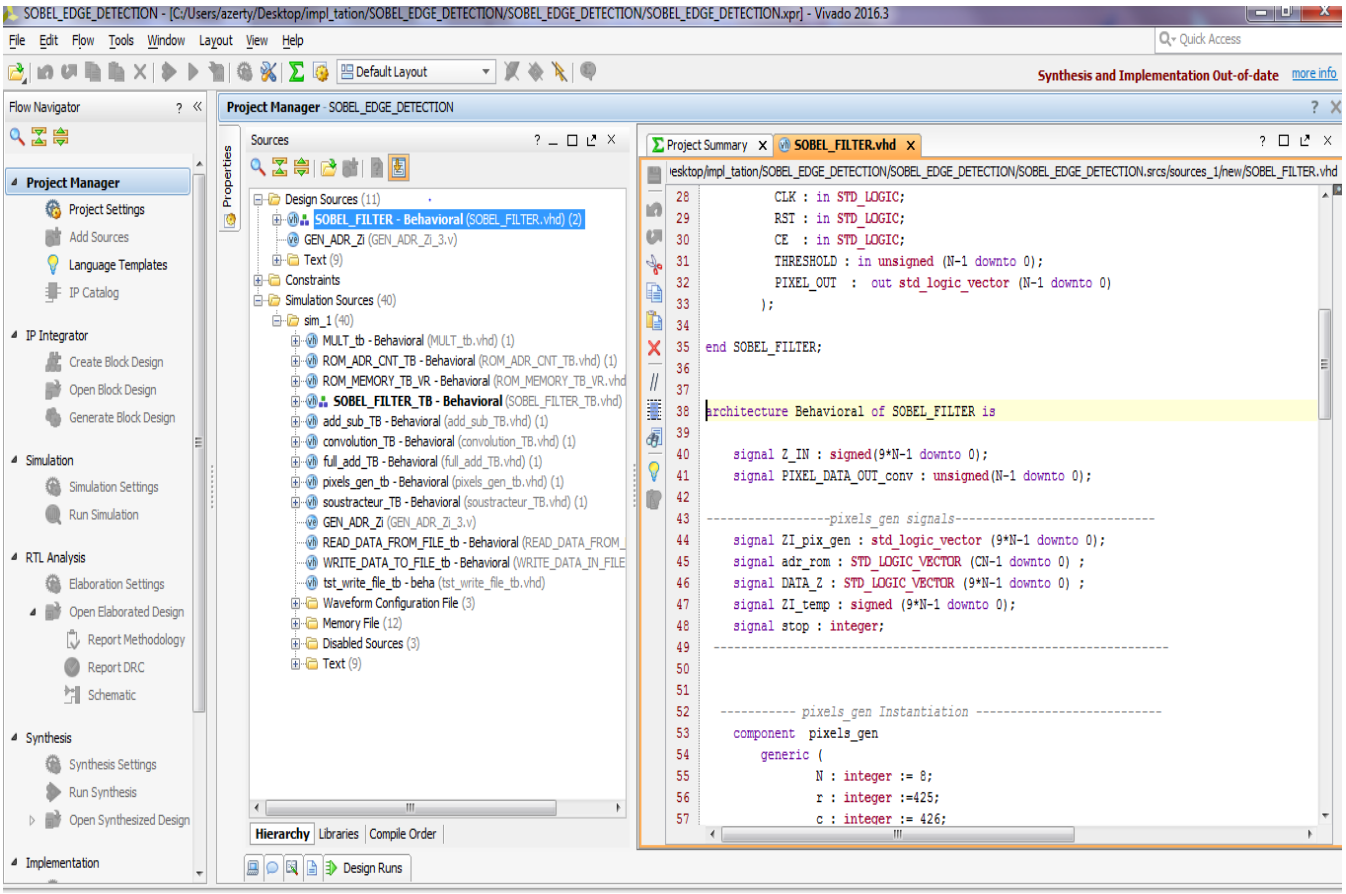


Figure III.4 : Environnement de développement VIVADO.

Chaque année, Xilinx propose 4 versions de VIVADO et chaque version supporte un certain nombre de familles de circuits FPGA (exp: dans VIVADO v.2016.3 on trouve les familles: SPARTAN-6, Virtex-7, Kintex, Artix7, Zynq-7000 ...etc).

VIVADO permet l'utilisation des langages de description matérielle Verilog et VHDL. En plus, VIVADO permet d'effectuer plusieurs types de simulations (fonctionnelle, post-synthèse et post-implémentation), synthèse, implémentation et configuration du circuit FPGA.

Le flot de conception de VIVADO (voir fig. III.5) utilise de nombreux blocs de construction système, qui sont des éléments d'IP (également nommé IPCore), peuvent être intégrés dans un projet. Contrairement aux anciennes méthodes de conception, VIVADO se concentre plutôt sur l'exploitation des IPcores, développés et vérifiés par Xilinx ou par d'autres compagnies (Third party). Ces IPCores sont disponibles dans la bibliothèque « IP catalog ».

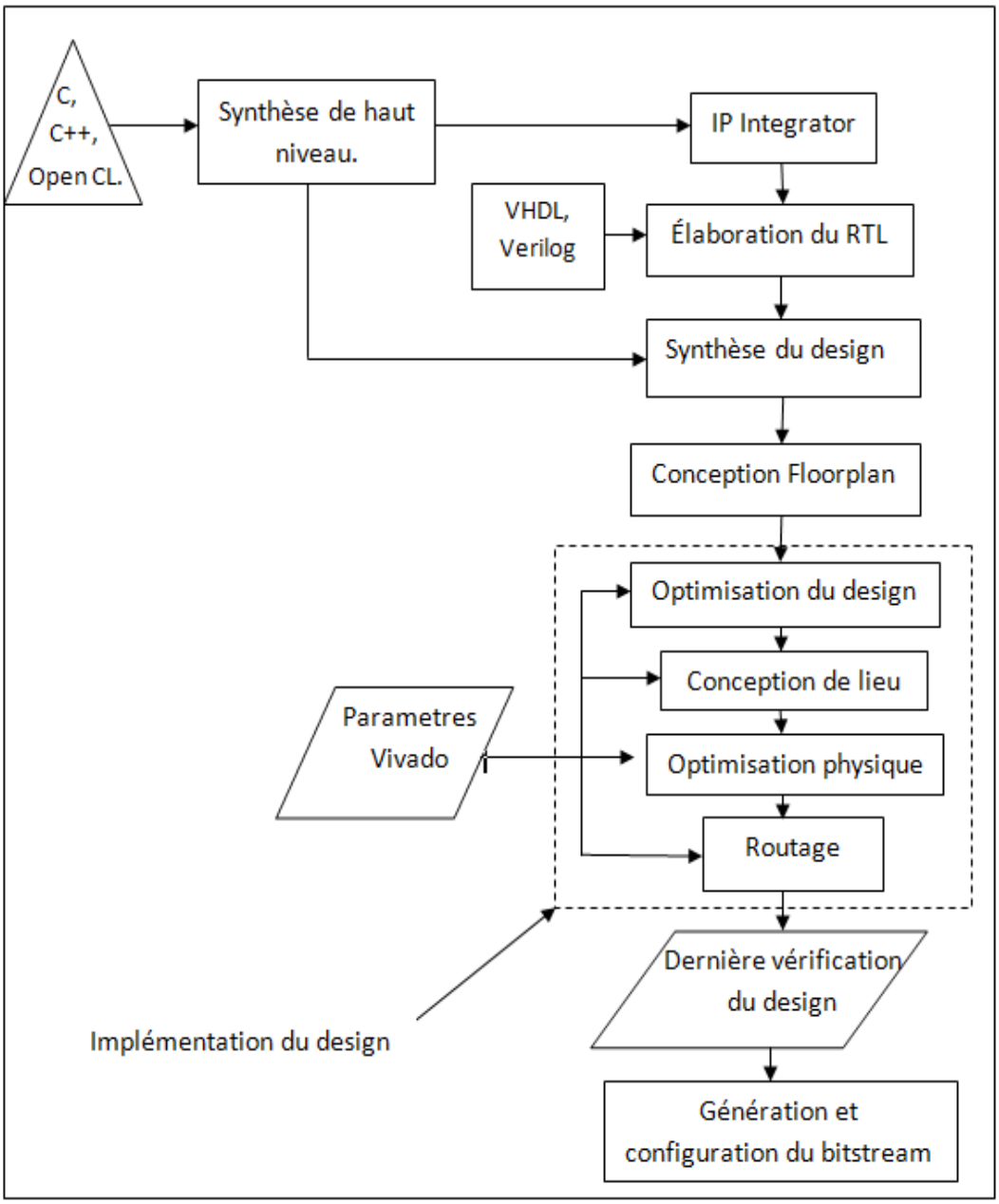


Figure III.5 : Flot de conception de VIVADO.

L'outil « IP Integrator » est le principal mode de réalisation de cette méthode de conception orientée IPCore. IP Integrator est une fonctionnalité intégré dans VIVADO, grâce à laquelle le concepteur peut adopter la même approche «descendante ou TOP-DOWN» qu'il prendrait naturellement pour concevoir la hiérarchie du système. Les instances IPCore peuvent être introduites à partir du catalogue existant, le cas échéant, ou créées sous forme de boîtes noires pour une population ultérieure avec des sous-systèmes fonctionnels et les interfaces entre ces différents éléments établies. Cette approche se prête au développement rapide du système matériel.

III.5. Description du kit « EMBV-Python1300-C »

Figure III.6 : Kit Caméra PYTHON-1300C-MicroZed Embedded Vision Carrier Card .

Le kit « EMBV-Python1300-C » (voir fig. III.6) est composé:

- d'un capteur d'images couleur de type CMOS.
- d'une carte « Embedded Vision carrier card», qui sert de support pour transférer l'image ou la vidéo, prise par le capteur CMOS vers la carte MicroZed.
- d'une carte de développement « MicroZed 7020 SOM ».

III.5.1. Capteur d'images « Python Receiver 1300-C »

Figure III.7 : PYTHON-1300-C Camera Module .

Le capteur « Python Receiver 1300-C » (voir fig. III.7) est composé d'un capteur d'images couleur, de type CMOS-SXGA, de 1/2 pouce et doté d'une **résolution de 1280 (H) x 1024 (V) pixels** et il peut capturer jusqu'à **210 images/seconde** [14].

III.5.2. Carte « MicroZed 7020 SOM »

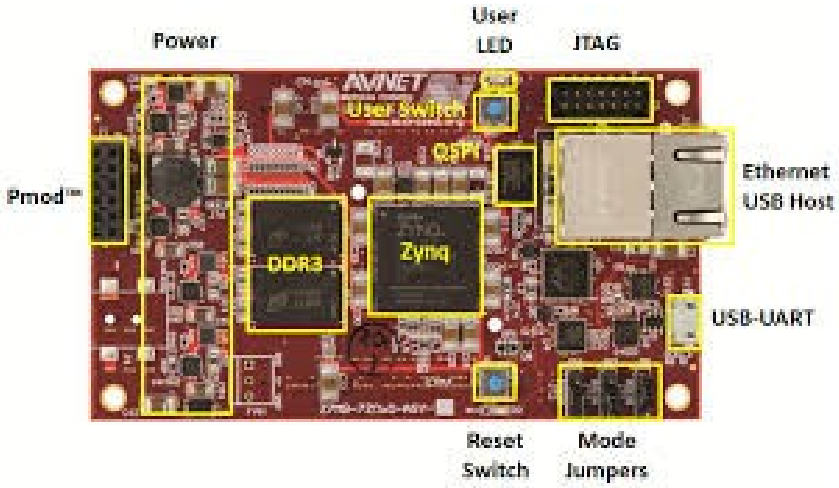


Figure III.8 : Carte de développement « MicroZed 7020 SOM ».

La carte « MicroZed 7020 SOM » est une carte de développement économique, basée sur le plus petit circuit FPGA de type Zynq-7000. Ce dernier est basé sur la logique programmable d'Artix-7, dotée d'une capacité de 133000 portes logiques, de 200 DSP48E1 et de 140 blocs RAM, le Z7020 contient également un bloc d'adresse DS190. Le circuit Zynq s'interface avec deux mémoires qui se trouvent sur la carte, l'une est une mémoire flash de 256 Mo et l'autre est une mémoire DDR3 de 512 Mo. La carte MicroZed est dotée de deux oscillateurs donc deux sources d'horloge, une est à 100 MHZ et l'autre est à 33.333 MHZ [15].

La conception unique de la carte MicroZed lui permet d'être utilisée à la fois comme une carte de développement, autonome pour l'expérimentation des SoC de base, ou combinée avec une carte porteuse (Carrier card), pour former un Système Embarqué sur Module (SOM). La carte MicroZed contient deux connecteurs d'E/S qui lui permettent, dans le circuit FPGA, la connexion à deux banques d'E/S du côté de la logique programmable (PL). En mode autonome, ces E/S PL sont inactives. Lorsqu'ils sont branchés sur une carte porteuse (Carrier card), les E/S deviennent accessibles d'une manière définie par la conception de la carte porteuse.

Les différents périphériques et interfaces de la carte MicroZed sont entourés d'un carré jaune (voir fig. III.8), parmi ce périphériques et interfaces, on site:

- Interface GPIO : englobe au totale 2 LED, 4 DIP-switches et 2 boutons poussoirs.
- Interface JTAG.
- Interface Pmod.
- Interface Ethernet.
- Interface USB-UART (pour la programmation et communication).
- SD card Slot située sur le dessous de la carte.

La carte MicroZed peut être programmée par différentes manières via la configuration des jumpers (voir fig. III.8):

1. En utilisant l'interface USB-JTAG : c'est la méthode la plus simple, pour l'adopter, on connecte directement le câble USB du PC vers carte MicroZed.
2. En utilisant JITAG: la carte est dotée d'un connecteur JTAG Xilinx qui peut être utilisé à la place de la connexion USB-JTAG, mais cela nécessite un câble USB Xilinx plateforme ou un câble de programmation Digilent USB-JTAG.
3. En utilisant Mémoire flash Quad-SPI, contenue sur la carte MicroZed. c'est une mémoire flash non-volatile. Elle permet de stocker la configuration du FPGA. Cette configuration persiste lorsque la carte est mise hors tension.
4. En utilisant Carte MicroSD : elle a la meme fonctionalité que la mémoire flash Quad-SPI [15].

III.5.3. Carte « EMBedded Vision carrier card»

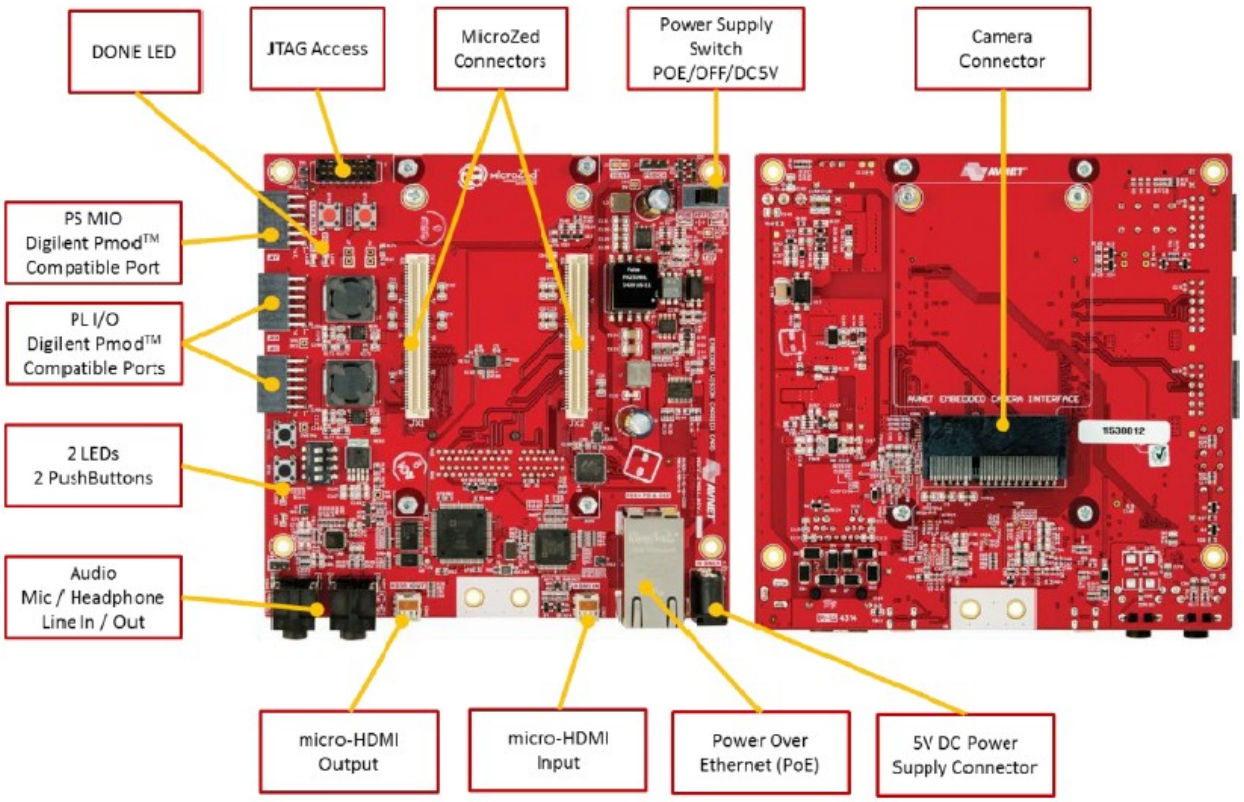


Figure III.9 : Carte porteuse « Embedded Vision carrier card ».

La carte « EMBdded Vision carrier card » (EMBV) est une carte dite « porteuse », puisque on peut facilement lui connecter le capteur d'images, ainsi que la carte MicroZed via les deux slots en blanc (voir fig. III.9). Aussi, elle possède une interface d'entrées/sorties de type HDMI, ce qui nous

permet de visualiser l'image ou video sur un écran.

La fonctionnalité de la carte EMBV n'est pas fixée au préalable, mais configurée à partir de la carte MicroZed, qui dépend de l'application chargée et démarrée par l'utilisateur, à partir d'une mémoire Non-volatile, se trouvant sur la carte MicroZed elle même [15].

III.6. Commande de la caméra

La commande du capteur CMOS lors de l'acquisition d'images, se fait par le circuit FPGA, contenu dans la carte MicroZed.

Après avoir défini le matériel de la caméra intelligente, nous allons définir dans ce qui suit, les composantes logicielles, nécessaires au développement d'applications vidéo personnalisées. Il est à noter que les fabricants Xilinx et Avnet, en plus du matériel qu'ils proposent, ils fournissent aussi des composantes logicielles, appelés « IPCore ». Ces derniers peuvent être des blocs de logique ou de données, conçus spécialement pour réaliser des fonctions bien spécifiques. En général, ils sont décrits en utilisant les Langages de Description Matérielle (HDL): VHDL ou Verilog.

Aussi, vu la complexité du matériel (kits de développement, cartes FPGA spécialisées dans différents domaines...etc), les fabricants (Xilinx, Avnet et bien d'autres) proposent aussi, en plus des IPCores, ce qu'on appelle des « Reference Design », qui ne sont autres que des portions de projets, développés spécialement dans le but de faciliter l'utilisation de leur matériel. Autrement dit, ces « reference design » servent comme un point de départ pour les utilisateurs pour créer des projets personnalisés en un minimum de temps.

De ce fait et en ce qui concerne le matériel qu'on utilisé (Kit EMBV Python1300-C), Avnet a proposé un design de départ, qui dans un premier temps, nous a considérablement faciliter le choix de ces IPCores, puisqu'ils sont étroitement liés au matériel à contrôler sur le kit, mais le revers de la médaille s'est avéré que le script fourni dans le referecne design, n'était pas fonctionnel pour multiples raisons; incompatibilité des versions des IPCore avec la version de VIVADO, incompatibilité entre les différents ports de connexion des IPCores,… etc. C'est pour cela qu'on était obligé de refaire tout le design à notre manière pour faire fonctionner la caméra. Pour commencer, nous nous sommes servi de ce design pour l'acquisition d'images, puis nous lui avons ajouté d'autres composantes, que nous avons nous même développés en VHDL pour le traitement d'image (voir chapitre IV).

III.6.1. Description des IPCore utilisés

Le tableau III.1 résume l'ensemble des blocs IPCore, qu'on a utilisé pour la commande de la caméra. Ces IPCores sont réalisés en VHDL et sont dans la majorité des cas des IPCore payants. Dans le cadre de ce mémoire, on a utilisé une version d'évaluation, qui est limitée dans le temps (4 mois).

IPCore	Version	Fabricant
Color Filter Array Interpolation (CFA)	7.0	Xilinx
Chroma Resampler	4.0	Xilinx
Video On Screen Display (OSD)	6.0	Xilinx
RGB to YcrCb Color-Space Converter	7.1	Xilinx
Video Timing Controller (VTC)	6.1	Xilinx
Video Test Pattern Generator (TPG)	7.0	Xilinx
Avnet hdmi in	1.1	Avnet
Avnet hdmi out	1.1	Avnet
onsemi_vita_spi	3.1	Avnet
onsemi_vita_cam	3.1	Avnet

Tableau III.1: Liste des IPCores utilisés dans la commande de la caméra.

La figure III.10 illustre le bloc diagramme de commande de la caméra. La chaîne de commande de la caméra, commence par le bloc « Python Receiver », qui représente le capteur d'images. L'image ou la vidéo capturée est donc transférée à l'IPCore « Video To AXI4S».

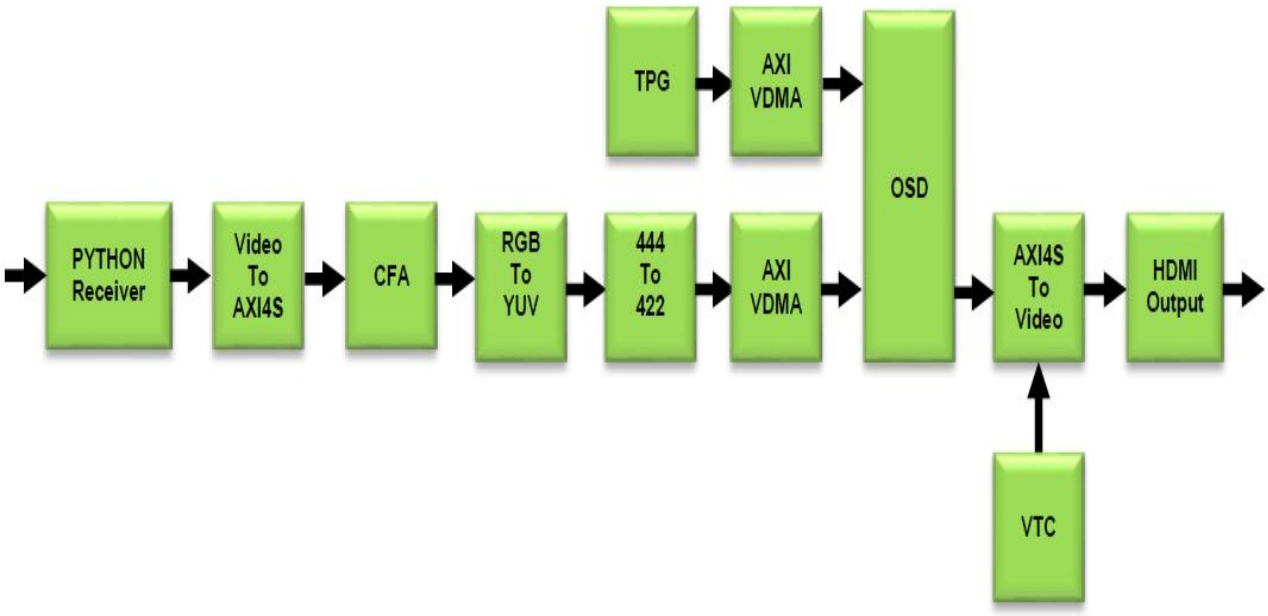


Figure III.10 : Chaîne de commande de la caméra EMBV-Python1300-C.

- Le bloc « PYTHON Receiver » décrit le fonctionnement du capteur CMOS.

- Le bloc « Vidéo ToAXI4-Stream » convertit une entrée vidéo (composée de données vidéo parallèles et de signaux de synchronisations vidéo) en un bus maître de type AXI4-Stream selon le protocole AXI4-Stream Video [16].
- Le bloc « Color Filter Array Interpolation» (CFA), reconstruit une image couleur à partir d'un capteur filtré (RVB ou CMY) en utilisant une ouverture d'interpolation 5x5 avec une résolution maximale de 7680 colonnes par 7680 lignes avec 8, 10 ou 12 bits par pixel et prend en charge la bande passante nécessaire pour les résolutions haute définition (1080 par 60) [17].
- Le bloc « RGB To YUV » est un convertisseur couleur-espace. Il transforme les données vidéo de couleur rouge, vert, bleu (RVB), en données vidéo de type: YCrCb 4: 4: 4 ou YUV 4: 4: 4. Le bloc RGB To YUV prend en charge quatre conversions de format courantes ainsi qu'un mode personnalisé permettant une transformation définie par l'utilisateur. Ce bloc est capable de traiter une résolution maximale de 7680 colonnes par 7680 lignes avec 8, 10, 12 ou 16 bits par pixel et prend en charge la bande passante nécessaire pour les résolutions haute définition (1080p60) [18].
- Le bloc « Video Direct Memory Access » (VDMA), fournit un accès direct à une large bande passante entre la mémoire et la cible de type vidéo AXI4-Stream [19].
- Le bloc « Video on screen Display » (OSD), produit des sorties vidéo à partir de plusieurs sources vidéo externes et de plusieurs contrôleurs graphiques internes. Chaque contrôleur graphique génère des superpositions de texte et de graphiques simples. Chacun d'entre eux est associée à une couche d'image. Jusqu'à huit couches d'image peuvent être positionnées dynamiquement, redimensionnées, avancées ou reculées et combinées à l'aide de l'alpha-blending, qui par définition est la combinaison convexe de deux couches d'image permettant la transparence. Chaque couche de l'OSD a un ordre de plan Z défini; ou conceptuellement, chaque couche est plus proche ou plus éloignée de l'observateur ayant une profondeur différente. Ainsi, l'image et l'image directement "sur" sont mélangés. L'ordre et la quantité de mélange sont programmables en temps réel [20].
- Le bloc « Video Test Patern Generator » (VTPG), génère des modèles de test vidéo, qui peuvent être utilisés lors du développement des IPCores de traitement vidéo ou de l'installation d'un système vidéo. Les modèles de test peuvent être utilisés pour évaluer et déboguer la couleur, la qualité, les contours et les performances d'un système vidéo, ou pour optimiser le fonctionnement de la vidéo [21].
- Le bloc « HDMI OUT » est une norme et une interface audio/vidéo totalement numérique pour transmettre des flux chiffrés constitués de données vidéo non compressées et des données audio pouvant être compressées. Elle est destinée au marché grand public.

- Le bloc « Video Timing Controller » (VTC), est un générateur et un détecteur de synchronisation vidéo à usage général. Ce bloc est hautement programmable grâce à un ensemble de registres complet permettant le contrôle de divers paramètres de génération, de synchronisation et d'interruption qui permettent une intégration facile dans un système de processeur pour un contrôle en temps réel. Le VTC est fourni avec une interface compatible AXI4-Lite [22].
- Le bloc « Avnet hdmi out» permet de contrôler l'interface HDMI via le circuit intégré ADV7611, ainsi qu'un code de synchronisation intégré dans ce bloc. l'entrée de ce bloc est sur 16 bits en mode YCbCr 4:2:2 [15].

III.6.2. Implémentation de la commande sur MicroZed

Le principe de fonctionnement de la commande est comme suit: sachant que le circuit FPGA Zynq-7000 de la carte MicroZed possède deux parties PS et PL (voir partie III.3, p. 45), alors la partie PS est utilisée pour initialiser le capteur d'images PYTHON-1300-C et la sortie de l'interface HDMI (sortie vers un écran reliée par câble HDMI). Aussi, dans la partie PL, un pipeline du capteur d'image est implémenté avec une mémoire tampon pour sauvegarder les trames vidéo.

Pour implémenter la commande de la caméra, nous avons utilisé VIVADO en mode « **script** », où nous avons:

1. Crée et configurer le projet VIVADO pour la carte MicroZed.
2. Crée le design (commande de la caméra) en y ajoutant le Processeur ARM avec sa configuration.
3. Ajouté et connecté l'ensemble des IPCores, listés dans le tableau III.1.

Il est à noter que par rapport au mode de conception graphique, le mode de conception en script TCL est plus pratique lorsqu'il est archivé pour être utilisé dans d'autre projets.

Pour implémenter ce design sur carte FPGA, nous avons exécuté les étapes suivantes:

1. Implémentation du design, tel qu'il est représenté dans VIVADO (voir fig. IV.3).
2. Création du BSP, de l'application logicielle et et du FSBL dans SDK.
3. Création de l'image du design (BOOT.bin) et sauvegarde sur carte MicroSD.
4. Insertion de la carte MicroSD dans le kit et allumage du système.

Après avoir lancé la caméra, il est possible de voir la communication entre le camera et le PC via un cable USB-UART (communication série), en utilisant l'émulateur **« TeraTerm »** (voir fig. III.11). la configuration de l'interface série est décrite dans la figure III.11.

Figure III.11 : Émulateur TeraTerm.

III.7. Conclusion

Dans le troisième chapitre nous avons vu en détail l'architecture des circuits FPGAs et en particulier la famille Zynq-7000, nous avons vu les outils de développement, ainsi que les différentes étapes de conception d'une application.

Par ailleurs, nous avons réalisé la commande d'une caméra industrielle sous VIVADO (en mode TCL-Script), puis on a testé le résultat sur le Kit EMBV et on voit bien que la caméra est fonctionnelle.

Dans le chapitre suivant nous allons présenter le travail effectué pour le traitement d'image sous l'environnement MATLAB (développement d'interface graphique) et VIVADO (Implémentation d'un filtre Sobel sur MicroZed).

Chapitre IV: Exemple de traitement d'image : Implémentation d'un filtre Sobel en VHDL

IV.1. Introduction

Le traitement d'image est un domaine très vaste qui a connu, et qui connaît encore, un développement important depuis quelques dizaines d'années. On désigne par « traitement d'images numériques» l'ensemble des méthodes et techniques permettant de modifier une image numérique afin d'améliorer l'aspect visuel ou d'en extraire des informations.

Dans ce chapitre on exposera les différents algorithmes de détection de contours, puis on appliquera ces algorithmes sur une image échantillon du cordon de soudure. Une comparaison entre ces algorithmes permettra de définir le meilleur algorithme à implémenter sur circuit FPGA d'un point de vue clarté des contours extraits pour pouvoir procéder à d'autres opérations de contrôle qualités du cordon de soudure.

Dans ce qui suit, on donnera quelques notions de base sur le domaine de traitement d'image

IV.2. Définition de l'image

L'image est un signal fourni par un capteur photosensible. Une image numérique est un signal multidimensionnel fini « D » (soit 2D, 3D, 4D…, etc.), composée d'unités élémentaires appelés 'Pixels' qui représente chacun une portion de l'image et est caractérisé par la valeur d'une fonction positive bornée nommée Niveau de Gris.

IV.3. Définition de la convolution

En mathématiques, le produit de convolution est un opérateur bilinéaire et un produit commutatif, généralement noté «*», qui, à deux fonctions f et g sur un même domaine infini, fait correspondre une autre fonction « $f*g$ » sur ce domaine, qui en tout point de celui-ci est égale à l'intégrale sur l'entièreté du domaine (ou la somme si celui-ci est discret) d'une des deux fonctions autour de ce point, pondérée par l'autre fonction autour de l'origine; les deux fonctions étant parcourues en sens contraire l'une de l'autre (nécessaire pour garantir la commutativité). La Convolution d'une fonction porte par elle-même.

Le produit de convolution généralise l'idée de moyenne glissante et est la représentation mathématique de la notion de filtre linéaire. Il s'applique aussi bien à des données temporelles (en traitement du signal par exemple) qu'à des données spatiales (en traitement d'image) [23].

En traitement d'image l'opération de convolution s'effectue sur une image en niveau de gris nommée « f » par un masque ou noyau de convolution « k » suivant la formule suivante :

$$f(x,y)*k(x,y)=\sum_{i=x1}^{i=x2}\sum_{j=y1}^{y2} f(x-i,y-j)*k(i,j) \qquad \text{(Eq. IV.1)}$$

IV.4. Détection de contours

Dans le traitement d'image, on définit un « contour » comme étant la frontière qui sépare deux objets, ou un objet du fond de l'image, par des discontinuités de l'intensité de lumière.

L'extraction de contours joue un rôle primordial dans tout système de vision par ordinateur. Beaucoup d'efforts ont été faits pour extraire les contours d'une image et plusieurs techniques ont été proposées à ce jour. Les contours correspondent généralement à des changements brusques de propriétés physiques ou géométriques de l'image perçue et forment ainsi des attributs très importants pour l'analyse [24].

Les contours dans une image proviennent des :

- discontinuités de la fonction de réflectance (texture, ombre),
- discontinuités de profondeur (bords de l'objet),

Le principe de la détection de contours consiste essentiellement à effacer tous les motifs à faible variation des niveaux de gris (ou de couleurs) pour ne conserver que les lignes de séparation entre régions homogènes. Il existe pour cela plusieurs opérations classiques de filtrage par **convolution**. Il y a aussi les filtres différentiels (**Gradient**, **Laplacien**) qui donnent des valeurs élevées aux points où la variation des niveaux est rapide. Les manières les plus répandues en détection de contours reposent sur la dérivation, en effet, celle-ci permet de détecter les variations d'une fonction. Il suffit donc de considérer une image *I* comme étant une fonction de deux variables *x* et *y* qui représentent les coordonnées dans l'espace dans le cas de l'image à deux dimensions. On peut calculer la dérivation sous deux formes: La première consiste en l'application d'un gradient sur l'image, et la deuxième utilise le Laplacien. Cependant, il n'existe pas à l'heure actuelle de processus complet et général qui pourrait extraire tous les types de contours [25].

IV.5. Méthodes de détection de contours

Il existe un grand nombre de méthodes de détection de contours dans une image, mais la plupart d'entre elles peuvent être regroupées en deux catégories :

- Recherche des extremums de la dérivée première : c'est-à-dire les maximums locaux de l'intensité du gradient. Les filtres de Sobel , Prewitt et Robert sont utilisés.
- Recherche des annulations de la dérivée seconde : en général les annulations du Laplacien ou d'une expression différentielle non linéaire : le filtre de Canny est utilisé .

Par définition, le gradient est un vecteur représentant la variation d'une fonction par rapport à la variation de ces différents paramètres. En physique et en analyse vectorielle, le gradient est une grandeur vectorielle indiquant la façon dont une grandeur physique varie dans l'espace, Aussi, le Laplacien est un opérateur différentiel défini par l'application de l'opérateur gradient, suivi de l'application de l'opérateur divergence [24].

IV.5.1. Filtres obtenus à partir de la dérivée première

Les filtres de dérivée première, appelés aussi « filtres étroits ». On cherche de maximiser leur réponse. Leur prototype est le filtre de gradient mais la dérivation accentuant le bruit (pixels parasites de répartition aléatoire). Le gradient est une dérivation au premier ordre et il est donné par la formule suivante :

$$\nabla I = \frac{\partial I}{\partial x}\vec{I}x + \frac{\partial I}{\partial y}\vec{I}y \qquad \text{(Eq.IV.2)}$$

Où $\vec{I}x$ est un vecteur unitaire suivant x et $\vec{I}y$ suivant y.

Dans la partie suivante, nous allons voir trois filtres, utilisant le gradient pour le filtrage de contours [23].

IV.5.1.1. Filtre de Robert

Le filtre de Roberts permet de calculer le gradient bidimensionnel d'une image de manière simple et rapide. Il amplifie les zones où la norme du gradient spatial est importante qui correspondent souvent aux contours. L'opérateur cherche les dérivées selon des directions diagonales et il est constitué de **deux masques 2×2 de convolution**.

$$Rx = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad Ry = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Rx et *Ry* représentent les masques de Robert.

Le module, ou force de contour, est calculé par la norme du vecteur composé par les deux composantes de la dérivée [24].

IV.5.1.2. Filtre de Sobel et Prewitt

Les opérateurs de Sobel et de Prewitt permettent d'estimer localement la norme du gradient spatial bidimensionnel d'une image en niveau de gris. Ils amplifient les régions de fortes variations locales d'intensité correspondant aux contours. Ces opérateurs consistent en une paire de masques de **convolution 3×3** dont une rotation de **90°** permet de passer d'un masque de convolution à l'autre [24]. Ces masques sont conçus pour répondre maximalement aux contours horizontaux et verticaux.

$$Px=\begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix} \qquad Py=\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix}$$

Px et ***Py*** représentent les masques de **Prewitt**.

$$Sx=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \qquad Sy=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}$$

Sx et ***Sy*** représentent les masques de **Sobel**.

L'application séparée de chacun des masques donne une estimation des composantes horizontales et verticales du gradient par un simple filtrage linéaire avec un masque **3×3**. Il est ensuite possible de calculer la norme et la direction du gradient en chaque point à partir des composantes du gradient:

$$H_s(0)=\int_{-w}^{+w} S(-x)f^2(x).d(x) \qquad \text{(Eq.IV.3)}$$

$$\nabla I=\frac{\partial^2 xI}{\partial x^2}+\frac{\partial^2 yI}{\partial y^2} \qquad \text{(Eq.IV.4)}$$

par la norme euclidienne:

$$\|\nabla\|=\sqrt{\nabla^2 x+\nabla^2 y} \qquad \text{(Eq.IV.5)}$$

La norme du gradient ainsi estimée correspond à l'intensité attribuée au pixel courant. C'est donc l'image de la norme du gradient que l'on visualise généralement. Les opérateurs de Sobel et de Prewitt appartiennent à la famille des filtres de contours directionnels car la norme du gradient en chaque point est approchée en ne considérant que les changements d'intensité dans deux directions spécifiques (horizontales et verticales)[24].

IV.5.2. Filtres obtenus à partir de la deuxième dérivée

L'utilisation des dérivées secondes est plus facile que celle des dérivées premières, au lieu de chercher les maximums de l'intensité du gradient, on cherche les zéros de la dérivée seconde, plus précisément le « Laplacien » qui est une dérivation au deuxième ordre :

$$\nabla I=\frac{\partial^2 xI}{\partial x^2}+\frac{\partial^2 yI}{\partial y^2} \qquad \text{(Eq.IV.6)}$$

Dans le cas d'une image, il n'existe pas une dérivée seconde unique mais quatre dérivées partielles. En pratique, on lève cette ambiguïté en ayant recours à l'opérateur Laplacien qui fait la somme des deux dérivées partielles principales seulement [23].

IV.5.3. Filtre optimal - Canny

Une autre approche plus récentes repose sur la définition de critères d'optimalité de la détection de contours; ces critères débouchent sur des filtres de lissage optimaux.

Le filtre de détection de contours est un filtre de réponse impulsionnelle finie *h(x)*, défini sur l'intervalle [-M ; M], ce qui signifie que *h* est nul en dehors de cet intervalle. La résolution du système est assez complexe [23] .

Canny a formalisé trois critères qui doivent valider un bon détecteur de contours, et à chacun d'entre eux est associé une formule mathématique comme suit :

1. Garantir une bonne détection, c'est-a-dire une réponse forte même à de faibles contours :

$$H_s(0)=\int_{-w}^{+w} S(-x)f^2(x).d(x) \qquad \text{(Eq.IV.7)}$$

2. Garantir une bonne localisation: $H_c=n_0.\sqrt{\int_{-w}^{+w} f^2(x).dx}$ (Eq.IV.8)

3. Assurer que pour un contour il n'y aura qu'une seule détection (éviter les effets de rebonds dus, par exemple, à la troncature des filtres):

$$RSB_{con}=\frac{|H_S(0)|}{H_c}=\frac{\left|\int_{-w}^{+w} f^2(x).dx\right|}{n_0.\sqrt{\int_{-w}^{+w} f^2(x).dx}} \qquad \text{(Eq.IV.9)}$$

IV.5.4. Comparaison des filtres « Roberts, Sobel et Prewitt»

Opérateur	Bruit	Orientation	contours	Localisation	Matrice
Sobel	**Peu sensible**	**90°**	**Moyens**	**Centré**	**3x3**
Prewitt	Assez sensible	90°	Épais	Centré	3x3
Robert	Sensible	45°	Fins	Décalage	2x2

Tableau IV.1 : Comparaison des filtres en utilisant le gradient.

Remarque:

Après avoir vu le principe de calcul de chaque filtre de détection de contours, on déduit que : le filtre de Robert est un algorithme complexe à réaliser, puisque il est composé de petits masques de convolution, ce qui implique une forte consommation de temps de calcul et d'espace mémoire. Par

ailleurs, les filtres de Sobel et Prewitt sont des algorithmes simples à réaliser, puisque la taille de leur masque est plus grande que celle de Robert. En plus, ces filtres sont moins sensibles au bruit, contrairement au filtre de Robert.

On peut toutefois noter que, les filtres de Sobel et de Prewitt produisent des contours plus épais et donc moins bien localisés que l'opérateur de Roberts mais sont plus résistants au bruit . L'opérateur de Sobel peut se décomposer en un lissage et une dérivation, le lissage supprime une partie du bruit mais en atténuant les transitions, il augmente l'épaisseur des contours. L'utilisation de ces filtres est recommandée dans le cas d'images bruitées [24].

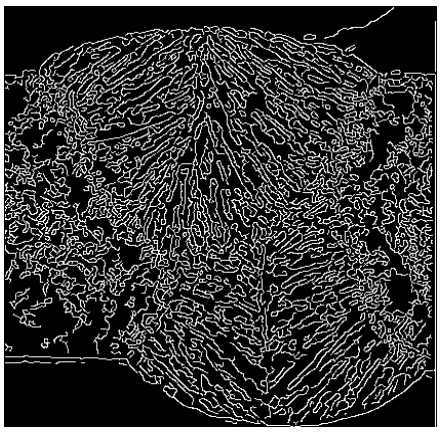


Figure IV.1 : Résultat de filtrage Canny dans Matlab.

En ce qui concerne le filtre de Canny, nous avons remarqué qu'il ne donne pas de contours utiles pour ce type d'images.

IV.6. Exemple d'application des 3 filtres sous MATLAB

Afin de voir l'effet de chaque filtre sur des images échantillon du cordon de soudure, les trois algorithmes : Robert, Prewitt et Sobel ont été implémentés sous MATLAB (version 2013.a). Le code MATLAB correspondant est donné en annexe « B ».

La figure IV.2 représente le résultat d'application des trois filtres sur une même image échantillon réelle du cordon de soudure sous MATLAB.

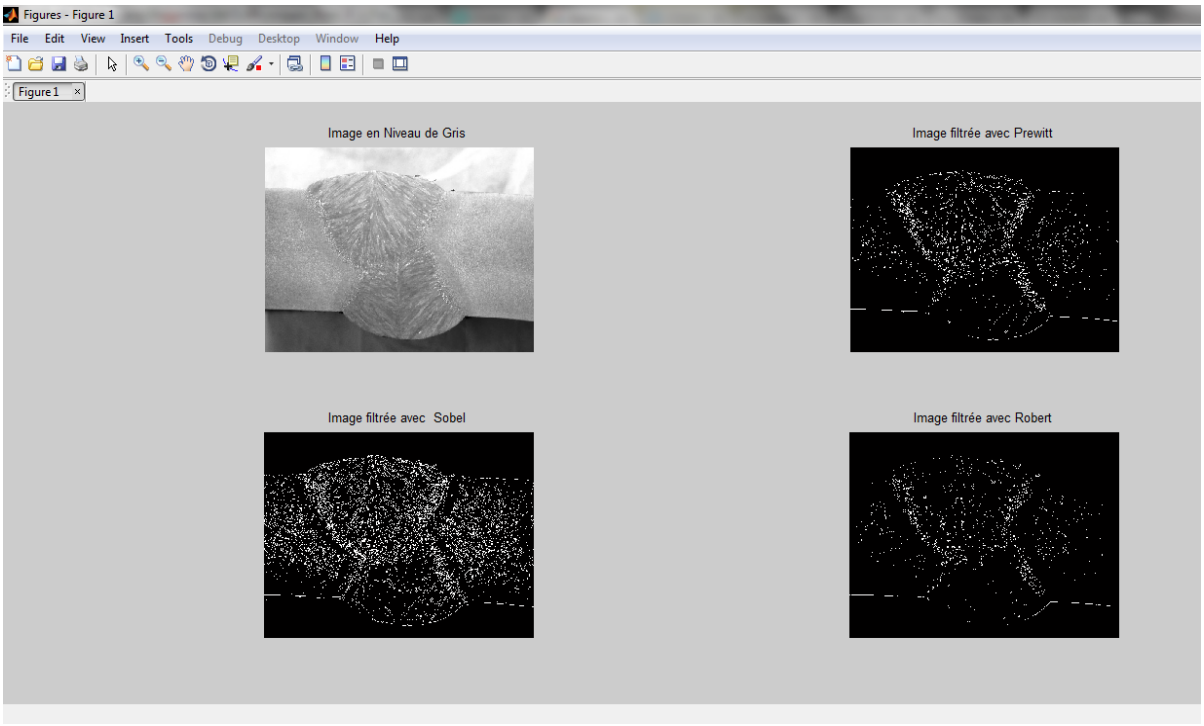


Figure IV.2 : Résultat de l'application des filtres sur image échantillon réelle du cordon de soudure.

Interprétation :

Les images, représentées dans la figure ci-dessus, nous permettent de voir l'effet de chaque filtre sur la même image échantillon de cordon de soudure. On peut clairement voir que les filtres de Robert et Prewitt ne donnent pas de contours visibles (contours faibles), par contre, le filtre de Sobel donne des contours plus précis (contours forts).

Le choix de la valeur du seuil pour le filtre Sobel a été pris suivant une intervalle, comprise entre 0,01 et 0,09, où on a constaté que pour unvaleur égale à 0,04, le contours est plus visible. Cette valeur a été fixée expérimentalement. c'est à dire, on a appliqué à la même image plusieurs valeurs de cette intervalle avec un pas de 0.01.

En conclusion, on déduit que pour ce type d'images, le filtre Sobel est le filtre le plus efficace.

IV.7. Interface graphique MATLAB

D'après ce qu'on a vu plus haut dans la figure **IV.2**, on a constaté que pour chaque type d'images (exp: images radiographiques, échographiques ou numériques ...etc.), il faut appliquer un certain type de filtre pour avoir un meilleur résultat lors de l'opération de détection de contours. C'est pour cela qu'on a décidé de réaliser une interface graphique sous MATLAB, qui nous permettra de voir rapidement l'effet de chaque filtre sur l'image qu'on souhaite traiter avant même de passer à

l'implémentation de ce filtre sur circuit FPGA en VHDL. Cela nous permet donc de gagner un temps considérable, lors du choix filtre à implémenter sur circuit FPGA.

Les détails de la réalisation de l'interface graphique ainsi que le code MATLAB, sont dans l'annexe B. La fenêtre principale de cette interface est la suivante:

Figure IV.3 : Interface graphique - 1 ere fenetre

Dans la deuxième fenêtre on peut choisir l'image à traiter et appliquer de filtre souhaité pour voir son effet comme montre la figure **IV .4**

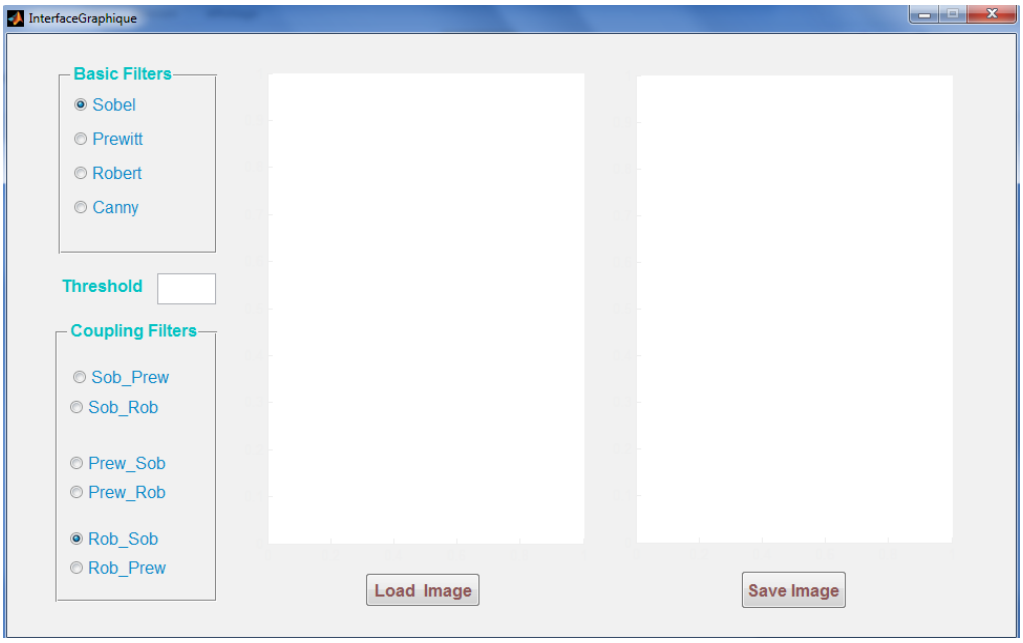


Figure IV.4 : Chargement de l'image et choix du filtre.

Après exécution, on obtient le résultat suivant:

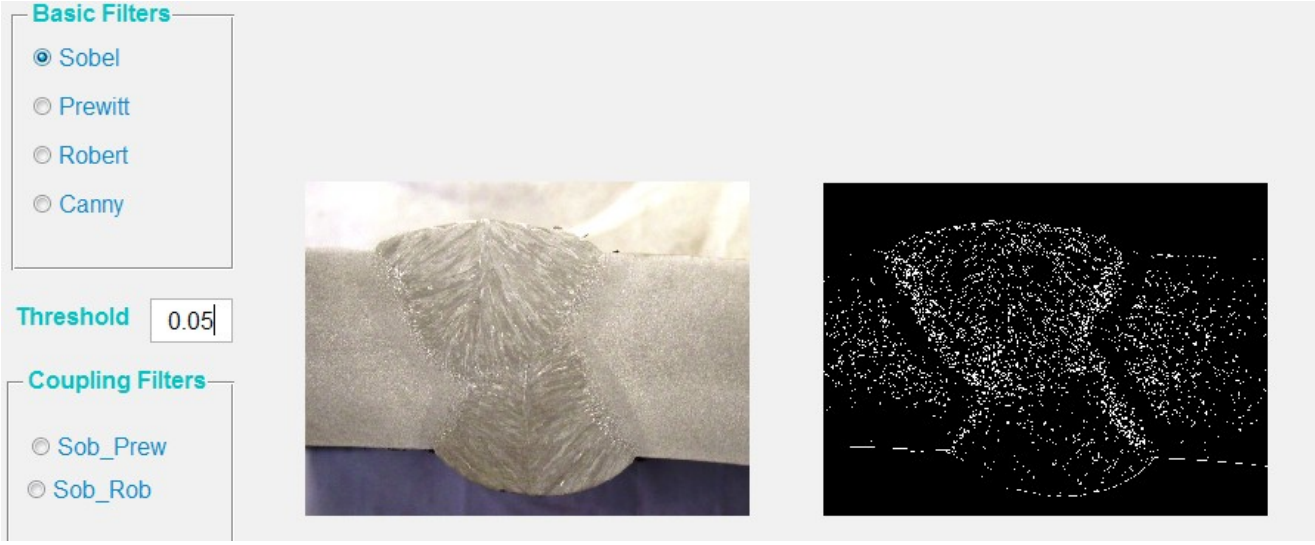


Figure IV.5 : Résultat de l'application du filtre de Sobel sur une image du cordon de soudure.

IV.8. Extraction de contours à l'aide du filtre de Sobel

Le filtre de Sobel, est l'un des filtres les plus utilisé dans le filtrage des image vu sa composition d'un masque de dérivation (3x3) et d'un opérateur de lissage élimant le bruit présent dans l'image.

Dans cette partie nous allons voir l'efficacité de ce filtre dans diverses applications.

L'article[26], présente une architecture du filtre Sobel implémentée sur circuit FPGA, selon une conception matérielle basique et simple qui se compose :

- Un bloc de génération d'adresses : composé essentiellement de 8 additionneurs complets et

d’un registre à décalage.

- Un bloc de multiplexage :qui multiplexe les les 8 bits composant le pixels et qui sont générer par le bloc de génération d’adresses.
- Un circuit de contrôle : qui commande et gère le fonctionnement du système, en générant les signaux de commande pour le multiplexeur ainsi que circuit de génération d’adresses.

Un autre article [27] se basant sur le filtre Sobel, à été proposé par Jérôme Dubois, Dominique Ginhac et Michel Paindavoine pour la conception et la création d’un capteur d'images reconfigurable dédié à l'imagerie rapide, aux traitements d'images linéaires et réseaux convolutifs testé.

Les principaux objectifs de ce capteur sont:

- évaluer la vitesse du capteur, en particulier à la cadence très élevée de 10 000 images par seconde.
- démontrer les possibilités de reconfigurations dynamiques de l'unité de traitement au sein du pixel.
- proposer un système de vision embarqué.
- Et cela en suivant une architecture simple pour des post traitements de bas niveaux comportant :
- Unité Arithmétique Analogique UA : capable de réaliser une combinaison linéaire de quatre pixels connexes.
- Mémoire Analogique, Amplificateur et Multiplexeur(MAM):permettant d’accélérer le processus d’acquisition et de lecture d’image(4096 pixels à balayer en 100µs soit » 40Mpixels/ s).

Pour conclure, les chercheurs ont testé leur architecture et obtenus des résultats assez satisfaisants, en vue de perspectives plus poussées (conception d'un capteur CIF (352×258) en technologie CMOS 130nm, intégrant des traitements basés sur la morphologie mathématique à haute cadence).

Travaux de Raiser et al [26] Ont étudié l'impact de diverses approches d'extraction d'objets pour la classification d'images dans l'inspection de surface. Diverses méthodes pour regrouper les pixels représentant les zones de défauts potentiels ont été étudiées, y compris les techniques classiques de traitement d'image et les approches de regroupement de données. Le défi consiste à regrouper des objets non connectés ou généralisés. Les classificateurs d'images ont été formés avec les fonctions dérivées des objets extraits. Un algorithme de cluster basé sur la densité nommé BDSCAN a été montré pour surpasser les autres méthodes dans les tests. Cette approche de regroupement est optimisée pour traiter un grand nombre de points de données.

IV.9. Implémentation du filtre Sobel en VHDL

Dans cette section, on présente le modèle d'architecture pour la mise en œuvre du filtre Sobel sur carte FPGA. Cette architecture proposée concerne des images échantillons en niveau de gris de cordons de soudure où chaque pixel est représenté par 8 bits dans un fichier txt générer par MATLAB, donné en entrée du bloc Pixels_gen qui va généré les Zi qui seront pris comme entrée du bloc de Convolution donnant en sortie le gradient du filtre Sobel. Il le comparera par la suite à une valeur de seuil bien précise donnée en entrée du bloc Sobel_Filter. En sortie un fichier txt sera généré avec les valeurs du gradient qui sera lu dans MATLAB afin d'afficher et de visualiser l'image filtrée.

l'architecture proposé est illustré dans la figure **IV.7** et comprend plusieurs blocs opérationnels décrits un plus loin.

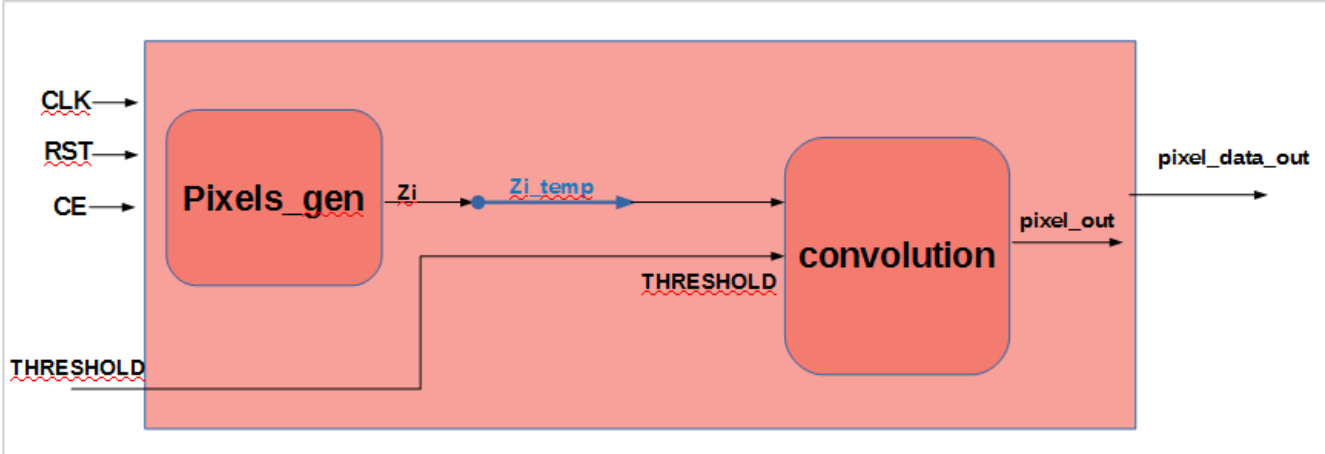


Figure IV.7 : Architecture proposée pour l'implémentation du filtre Sobel sur carte FPGA

IV.9.1. Bloc pixels-gen

Le bloc Pixels_gen nous permet la génération en parallèle des Zi qui sont des portions de l'image originale de taille 3x3 à partir de 9 ROM , où chaque ROM contient les valeurs de chaque pixel de l'image sous 8 bits .Les mémoires ROM ont été initialiser avec ces valeurs à l'aide d'un fichier (.txt) générer sous MATLAB pour être ajouter a notre projet en VHDL sous forme de fichier (.mem).

Les Zi générer en sotie seront transférer par un bus de données de taille 72 bits à l'aide d'un registre intermédiaire Zi_temps au bloc de convolution, où on va effectuer nos traitements.

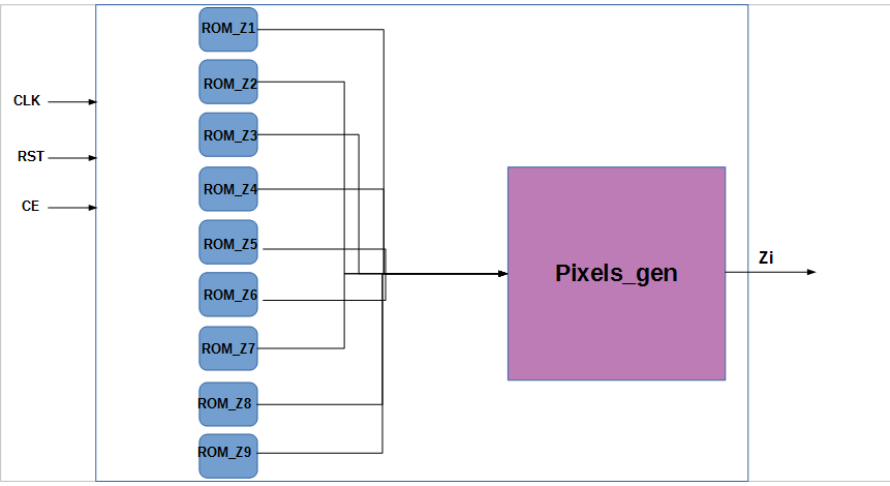


Figure IV.8 : Architecture proposée pour le bloc Pixels_gen

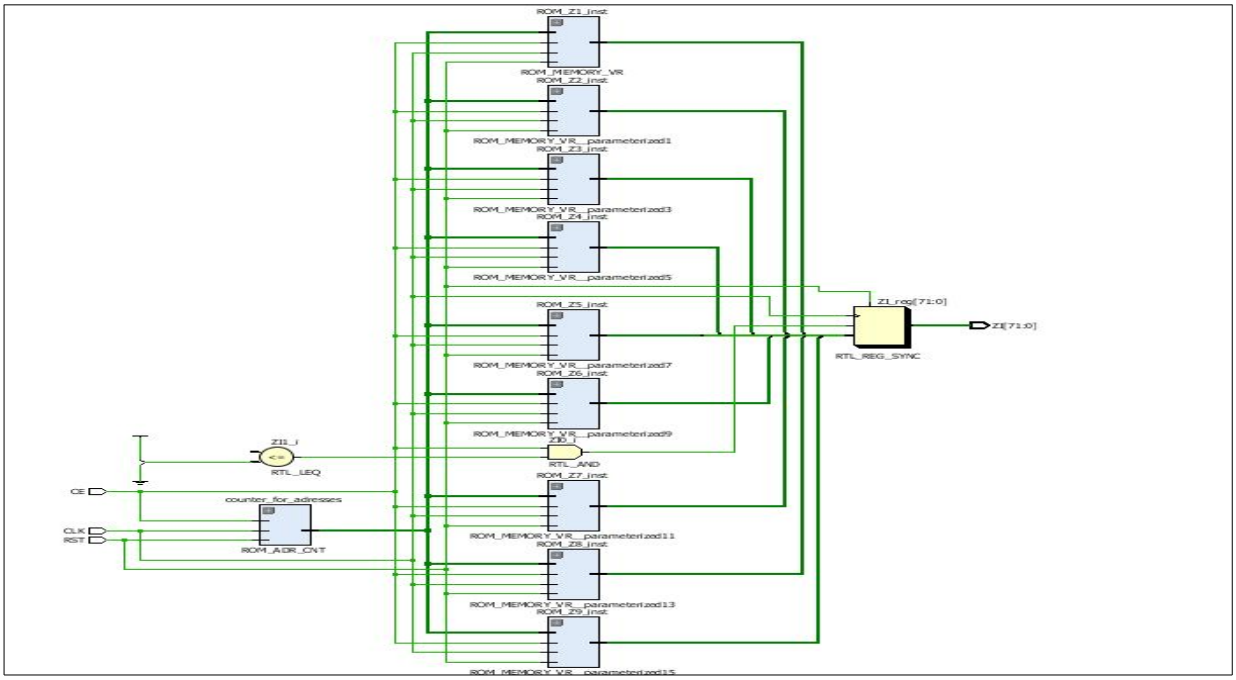


Figure IV.9 : Architecture du bloc Pixels_gen sous .

IV.9.2. Bloc de convolution

Ce bloc effectue l'opération de convolution des Zi générer par le bloc Pixels_gen avec les masques (3x3) du filtre Sobel. Le calcul du module est fait à partir du calcul des composantes (horizontal et vertical) Gx et Gy suivant les relation suivantes :

$$Gx=(Z_7+2*Z_8+Z_9)-(Z_1+2*Z_2+Z_3). \qquad \text{(Eq.IV.10)}$$

$$Gy=(Z_3+2*Z_6+Z_9)-(Z_1+2*Z_4+Z_7). \qquad \text{(Eq.IV.11)}$$

Le bloc Convolution se compose des sous blocs décrits ci-dessous afin de calculer les deux composantes Gx et Gy du filtre Sobel comme le montre les figures IV.10 et IV.11.

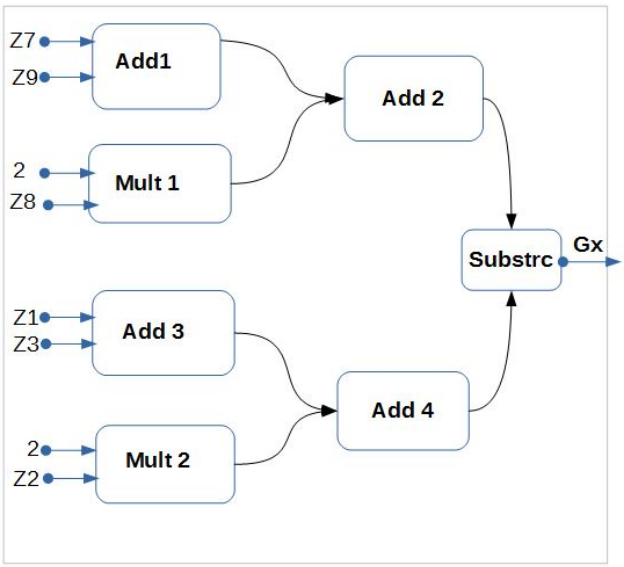


Figure IV.10 : Architecture de la composante Gx du filtre Sobel.

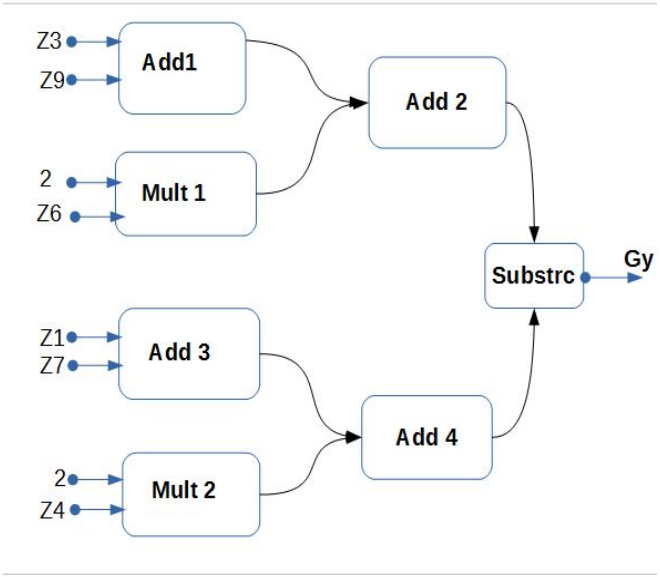


Figure IV.11 : Architecture de la composante Gx du filtre Sobel.

Pour implémenter le bloc de convolution en VHDL, on a proposé l'architecture présentée dans la figure **IV.3**

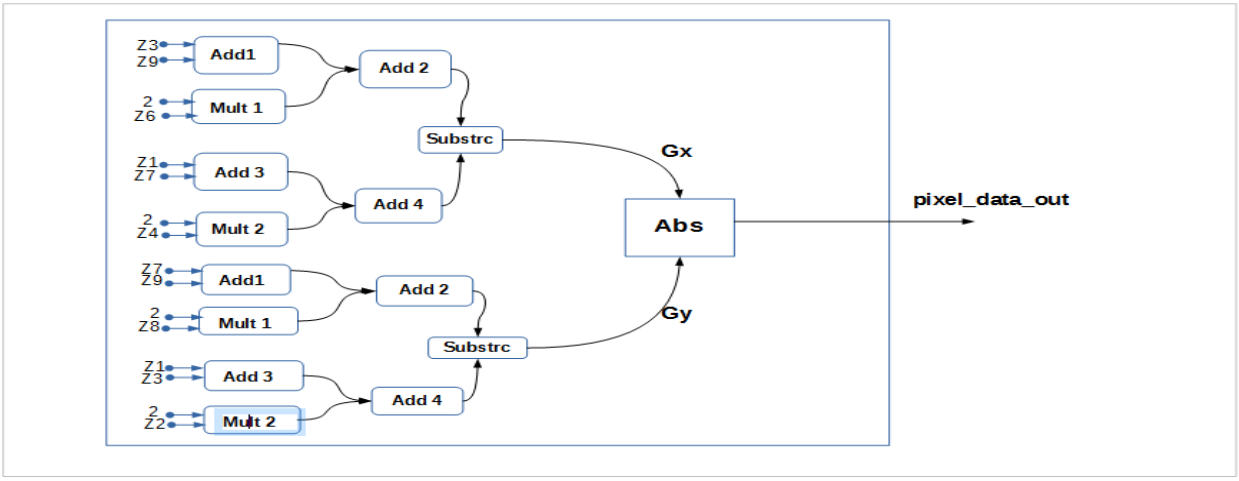


Figure IV.12 : Architecture proposée pour le bloc de Convolution

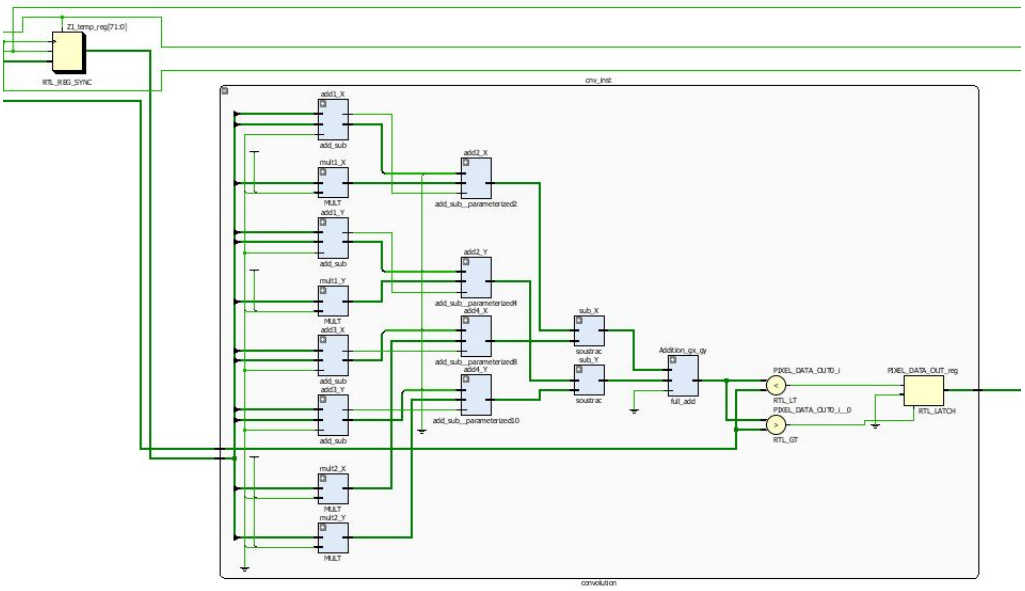


Figure IV.13 : Architecture du bloc de Convolution sous Vivado.

IV.9.2.1. Additionneur

Ce sous bloc a 3 entrées **a1, b1** et d'une retenue d'entée **cin** sur 8 bits et une sortie **s1** sur 8 bits avec une retenue de sortie **cout** sur 8 bits. Il effectue l'opération d'addition des différents Zi donnant les deux composantes du filtre Sobel Gx et Gy, ainsi que l'addition de ces deux composantes pour avoir la sortie du bloc de convolution qui est l'addition de la valeur absolue de chaque composante.

IV.9.2.2. Soustracteur

Ce sous bloc a été conçu pour effectuer l'opération de soustraction des différents Zi composants Gx et Gy. Il se compose de deux entrées **a, b** et d'une sortie **b** sur 8 bits.

IV.9.2.3. Multiplieur

On a conçu ce sous bloc pour effectuer les opérations de multiplication nécessaires à l'obtention des composantes horizontale et verticale du filtre suivant les équations **Eq.IV.10** et **Eq.IV.11**

Il est composé de deux entrées **A** et **B** sur 8 bits et une sortie **P** sur 8 bits.

IV.9.2.4. Valeur absolue

Le sous bloc Valeur absolue calcule les valeurs absolues des différentes valeurs de composantes Gx et Gy pour qu'elle soit comparer à une valeur seuil (dite THRESHOLD) à la sortie de ce sous bloc

IV.9.3. Filtrage d'une image avec le filtre Sobel en VHDL

Le résultat du filtrage Sobel en VHDL est donné sous forme d'un fichier texte, qui va être lu

dans MATALB pour pouvoir afficher et visualiser l'image traitée.

Image originale	Image traitée par filtre Sobel sous MATLAB	**Image traitée par filtre Sobel implémenté sur FPGA**

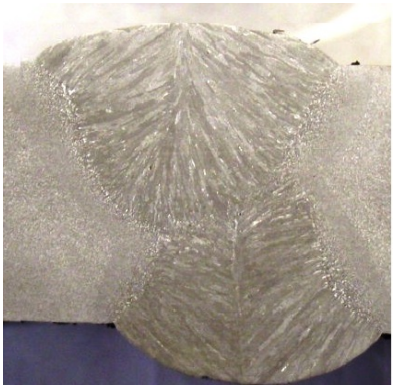
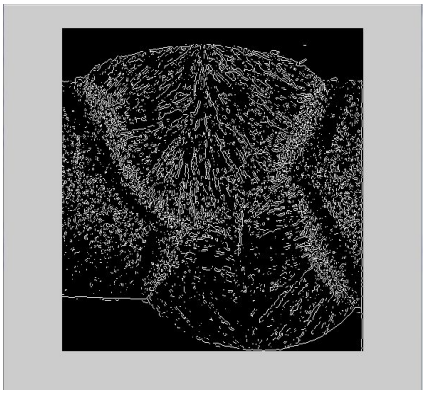
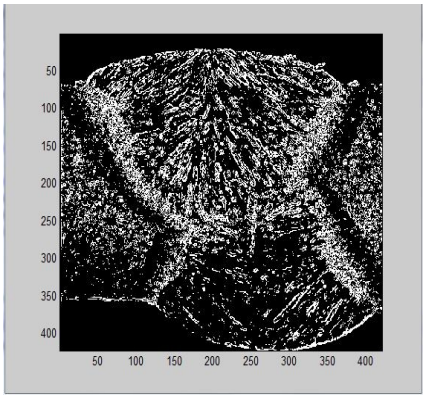
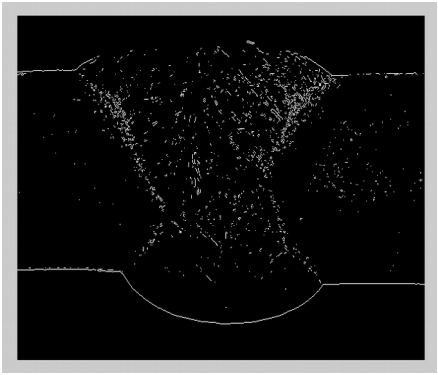
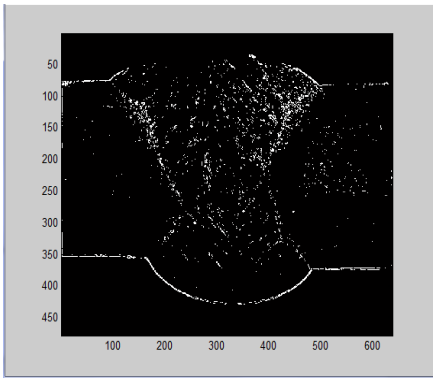
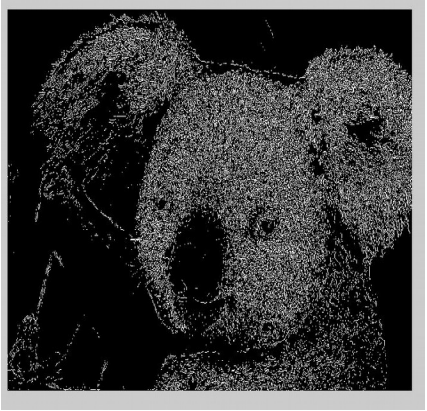
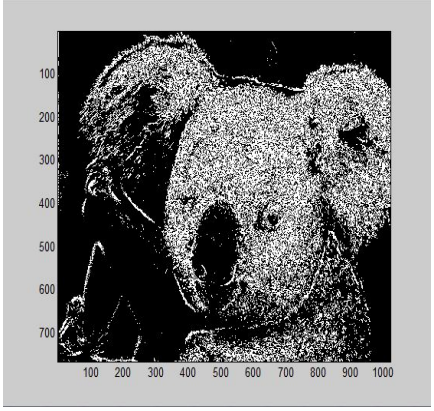

		

Figure IV.14	***Figure IV.15***	***Figure IV.16***
		
Figure IV.17	***Figure IV.18***	***Figure IV.19***
		
Figure IV.20:	***Figure IV.21***	***Figure IV.22***

Interprétation :

Les images **IV.14** et **IV.17** représentent des images originales de cordons de soudure, avec une résolution de **480x640** pixels auxquelles nous avons appliqué un filtrage Sobel par deux méthodes pour pouvoir comparer nos résultats. Dans une première étape nous avons appliqué aux images originales de cordons de soudure le filtre Sobel sous MATLAB où nous avons visualiser les résultats dans les figures **IV.15** et **IV.18**; dans la deuxième étape nous avons appliqué aux même images originales de cordons de soudure le même filtre de détection de contours mais implémenté en VHDL. Les résultats sont présentés dans les figures **IV.16** et **IV.19**. Nous avons remarqué que le traitement Sobel implémenté sur circuit FPGA donne le même résultats que le traitement Sobel sous MATLAB.

Afin de confirmer l'efficacité de notre filtre implémenté sur carte FPGA sur d'autres types d'images, nous avons pris une autre image échantillon représentée dans la figure **IV.20** (image d'un Koala), avec une plus grande résolution **1024x768 pixels** à laquelle nous avons appliqué le même traitement sous MATLAB et en VHDL. Les résultats sont identiques que les 2 premières images, ce qui confirme la validité des résultats obtenus lors de l'implémentation du filtre de SOBEL en VHDL, par la comparaison des valeurs des pixels de l'image traitée sous MATLAB et l'image traitée en VHDL, nous avons trouvé que les valeurs sont identiques. La figure **IV.23** représente les valeurs des pixels de l'image traitée en VHDL, et la figure **IV.24** représente les valeurs des pixels de l'image traitée sous MATLAB.

Figure IV.23:Pixels traités en VHDL

Figure IV.24:Pixels traités sous MATLAB

IV.10. Conclusion

Dans ce chapitre nous avons étudié la détection de contours pour choisir le bon filtre qui nous permettra d'obtenir le plus de contours précis sur les cordons de soudure pour pouvoir effectuer les calculs de distances; tout en considérant des algorithmes basés sur la convolution d'images par des masques d'extraction de contours (masque de Sobel, Prewitt et Robert). L'analyse de l'image du cordon de soudure nous a permis de dire que l'opérateur Robert fournit globalement, des contours fins et non précis. Cela est dû à la petite taille de ces masques et à la complexité de son algorithme. Alors que le filtre Prewitt donne des contours plus visibles que Prewitt. En revanche si on regarde les contours obtenus avec le filtre Sobel, on s'aperçoit que ce dernier a donné plus de contours (détails) que tous les autres filtres d'où notre choix d'utilisation de ce filtre dans ce mémoire.

Les résultats obtenus après implémentation du filtre Sobel en VHDLsont validé par ceux, délivrés par MATALB.

Conclusion générale

Cette étude nous a permis de réaliser un système embarqué, permettant l'automatisation de l'opération de contrôle qualité des cordons de soudure des pipes transportant divers produits(eau, gaz, pétrole).

Nous avons commencé le travail par une études sur les différentes méthodes de détection de défauts de soudures, ainsi que sur les critères de contrôle qualité des pipes; puis nous avons choisi le type de caméra (capteur CMOS) avec une haute résolution. Par la suite nous avons rélisé la commande de la caméra à l'aide du logiciel VIVADO 2016.3.

Par ailleurs, nous avons crée une interface graphique sous MATLAB 2013.a, nous permettant de visualiser le résultats de l'application de plusieurs filtres de détection de contours sur des images échantillons de cordons de soudure pour pouvoir choisir le filtre qui nous donnera le plus de contours utiles.

Après avoir choisi le filtre qui convient à nos besoins, et en vue de son implémentation sur carte FPGA, nous avons conçu un bloc de traitement Sobel à l'aide en VHDL et qu'on a appliqué à des images échantillons de cordons de soudure.

Le résultat de notre implémentation résout en partie le problème auquel est dédié cette étude. Et comme perspectives, l'amélioration de l'algorithme de détection de contours à implémenter sur carte FPGA, proposition d'une méthode automatique de mesure des distances facilitant ainsi la prise de décision sur la qualité du cordon de soudure .

Références bibliographiques

[1] Centre de Recherche en Technologies Industrielles (CRTI): https://www.crti.dz/presentation-csc.php?idlangue=2.(consulté le 25/10/2018)

[2] F. Aggoune, « Evaluation de l'endommagement des tubes dans leurs conditions d'exploitation » Thèse de magister, Université MENTOURI Constantine, Algérie 2010.

[3] Cite officiel de la société: Spa Algérienne de Fabrication de Pipe EPE ALFAPIPE : http://www.imetal.dz/imetal/entreprise/alfapipe-spa/. (consulté le 10/09/2018)

[4] Fabrication des pipes: http://www.reliance-foundry.com/blog/acier-lamine-chaud-contre-froid-fr#gref. (consulté le 10/08/2018)

[5] Méthodes de contrôle qualité des pipes : www.enst.dz/conferenceTCND/CONTROLE_NON_DESTRUCTIF.pptx.(consulté et télécharger le 23/07/2018)

[6] I. Djedid, « Etude sur les défaillances des aciers API-5XL60 » Mémoire de master, Université de Tlemcen, Algérie, 2013.

[7] F.Hadjoui, « Etude du comportement en fatigue des aciers pour pipelines à différents grades », Thèse de Doctorat, Université d'Abou Beker Belkaid, Telemcen, Algérie, 2013.

[8] O.Bouledroua, M. Ouled Mbereick, M.Hadj Meliani « Qualification d'un acier API5L Etude expérimentale et validation numérique ». Revue Nature et Technologie Volume 7, Numéro 2, Pages 34-39, 13 Avril 2015.

[9] M. Elalaili, S. Hamlat, « Comparaison de 12 caméras industrielles » Licence professionnelle, Université de Lille, France, 2013.

[10] Support cours, « Capteurs numériques CCD & CMOS», Master 1 EEA, 2013. 68p

[11] Groupe Industriel National Instruments, Des produits,[en ligne],Disponible sur : https://www.ni.com/en-lb/shop/select/machine-vision category#facet:&productBeginIndex:0&orderBy:&pageView:grid&pageSize:&. (Consulté le 20/09/2018)

[12] B. HEYRMAN, M. PAINDAVOINE, R. SCHMIT, L. LETELLIER, « Conception fonctionnelle d'une caméra intelligente pour le traitement vidéo temps réel, Traitement Du Signal Et Des Images », Vol 2: Actes Du 20e Colloque GRETSI, UCL presses universitaires de Louvain, Belgique, 2005.

[13] Fiche technique du Zynq 7000, Xilinx, version (1,11,1), 2 juillet 2018, version pdf, 25p. Disponible sur : https://www.xilinx.com/ .

[14] AVNET. Guide d'utilisateur : PYTHON-1300-C Frame Buffer Design Tutorial. Format pdf. 23 Février 2016. version 2015_4. 15p

[15] AVNETSILICA. MicroZed Industry 4.0 Ethernet Kit (I4EK)-Designed by Avnet. [En ligne]. Disponible sur :https://www.avnet.com/wps/portal/silica/products/product-highlights/2016 microzed-i4ek/. (consulté le 25/08/2018)

[16] Guide d'utilisateur. Xilinx.Video In toAXI4-Stream v4.0,LogiCORE IP Product Guide. format pdf. Version 4 Octobre 2017. 43p.

[17] Guide d'utilisateur. Xilinx. Color Filter Array Interpolation v7.0,LogiCORE IP Product Guide.format pdf. Version 18 Novembre 2015. 45p

[18] Guide d'utilisateur. Xilinx. RGB to YcrCb Color-Space Converter v7.1, LogiCORE IP Product Guide. Format pdf. 18 Novembre 2015. 41p

[19] Guide d'utilisateur. Xilinx. AXI Video Direct Memory Access v6.2, LogiCORE IP Product Guide. Format pdf. 30 Novembre 2016. 47p

[20] Guide d'utilisateur. Xilinx. Video On-Screen Display v6.0, LogiCORE IP Product Guide. Format pdf. 18 Novembre 2015. 38p

[21] Guide d'utilisateur. Xilinx. Video Test Pattern Generator v7.0, LogiCORE IP Product Guide. Format pdf . 5 Avril 2017. 103p

[22] Guide d'utilisateur. Xilinx. Video Timing Controller v6.1, LogiCORE IP Product Guide. Format pdf. 4 Octobre 2017. 16p

[23] R. Deriche, I. Sophia, « Techniques d'extraction de contour », Thèse De Doctorat, Université de Paris, 2004.

[24] C. Ferkous, « La méthode des contours actifs en traitement des images », Mémoire de magister, université de Guelma, Algérie, 2010.

[25] S. Rital « Hypergraphe de Voisinage Spatiocolorimétrique, Application en traitement d'image, Détection de contours et du bruit» Thèse De Doctorat, Université de Bourgogne Dijon, France, 2004.

[26] Eng. Dina Alghurair,Sefwan S. Al-Rawi.''Design of Sobel Operator Using Field Programmable Gate Arrays'';ISBN: 978-1-4673-5613-8©2013 IEEE.

[27] Jérôme Dubois, Dominique Ginhac, Michel Paindavoine « Conception en technologie CMOS d'un Système de Vision dédié à l'Imagerie Rapide et aux Traitements d'Images», Thèse de doctorat, Université de BOURGOGNE, France, le 27 Aout 2008.

Annexe A:

Pour l'implémentation filtre Sobel sur carte FPGA , nous avons conçu plusieurs sous blocs composant le bloc du filtre. Ci-dessous nous présentons les codes VHDL des sous blocs ainsi que leurs résultats de simulations.

Les codes VHDL :

```
----------Code VHDL d'un Multiplieur----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity MULT is
  generic (N: integer :=8);
  Port ( A : in signed  (N-1 downto 0)  ;
  B : in signed(N-1 downto 0) ;
  P : out signed (2*N-1 downto 0)
);
end MULT;
architecture Behavioral of MULT is
begin
P <=  A  *  B ;
end Behavioral;
----------Fin du code----------
```

- Résultat de la simulation du code VHDL :

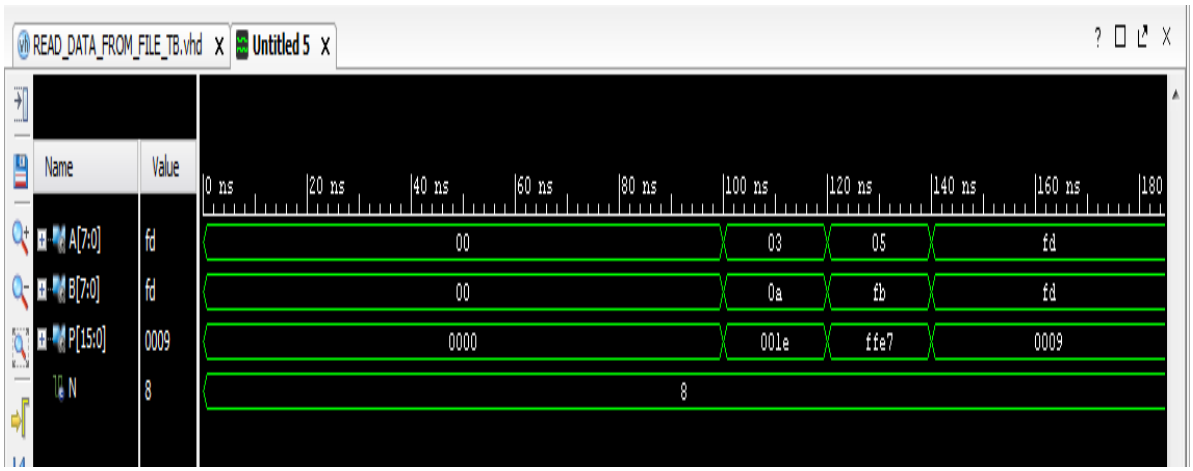


Figure 1 : Résultat de simulation du bloc 'Multiplieur'.

```
---------- code VHDL d'un Soustracteur----------
 library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use ieee.numeric_std.all;
  entity soustrac is
                generic (N: integer:=8);
                  port(   a : in  signed(N-1 downto 0);
                       b : in  signed(N-1 downto 0);
 s   : out signed(N-1 downto 0)  );
   end soustrac;
 architecture Behavioral of soustrac is
                begin
               s<= signed(signed(a)-signed(b)) ;  end Behavioral;
----------Fin du code----------
```

- Résultat de la simulation du code VHDL :

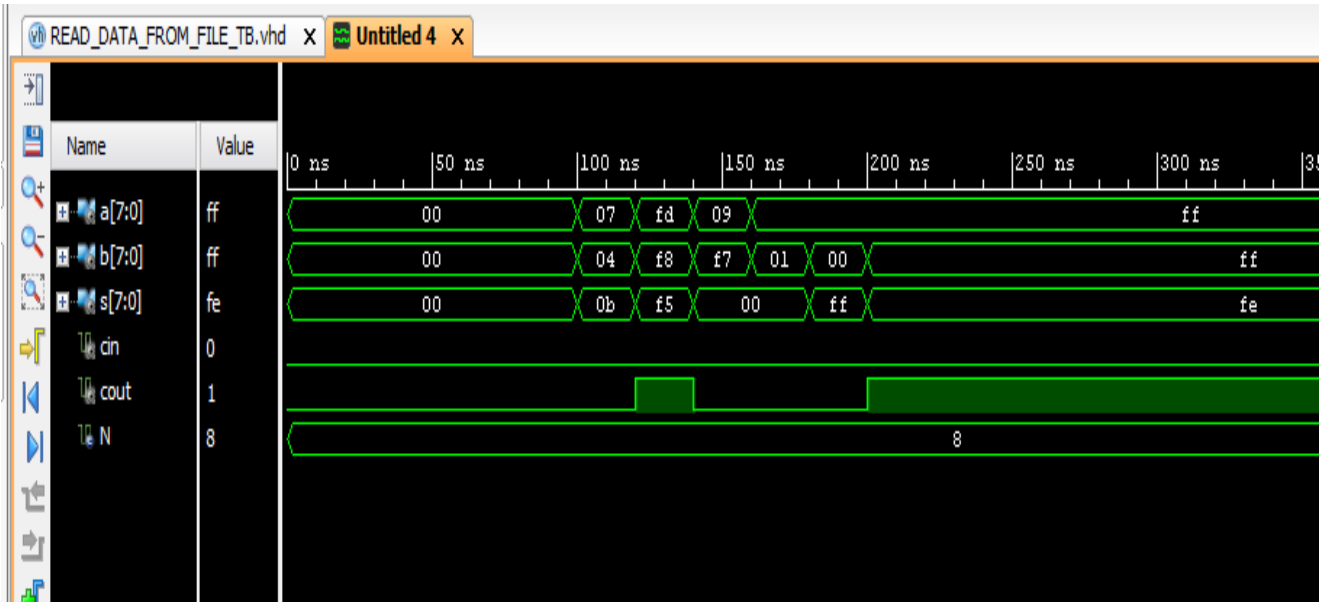


Figure 2 : Résultat de simulation du bloc 'Soustracteur'.

```
----------code VHDL d'un Additionneur complet----------

 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
entity full_add is
          generic (N: integer:=8);
            port( a1 : in signed(N-1 downto 0);
                        b1 : in signed(N-1 downto 0);
                         s1   : out unsigned(N-1 downto 0);
                       cin1: in std_logic;
                        cout1 : out std_logic
                       );
              end full_add;

   architecture Behavioral of full_add is
```

```
                signal P :signed(N-2 downto 0):= (others => '0');
                signal D :signed(N-1 downto 0):= (others => '0');
               signal y:signed(N-1 downto 0):= (others => '0');

                signal a2 : unsigned(N-1 downto 0);
              signal b2 : unsigned(N-1 downto 0);
              begin
   y <= (P&cin1) ;
                              D <=( a1 and b1 )or (y and (a1 xor b1)) ;
 cout1<= D(7);
                s1<=unsigned(abs(a1)+abs( b1)) ;
 end Behavioral;
```

----------Fin du code ----------

- Résultat de la simulation du code VHDL

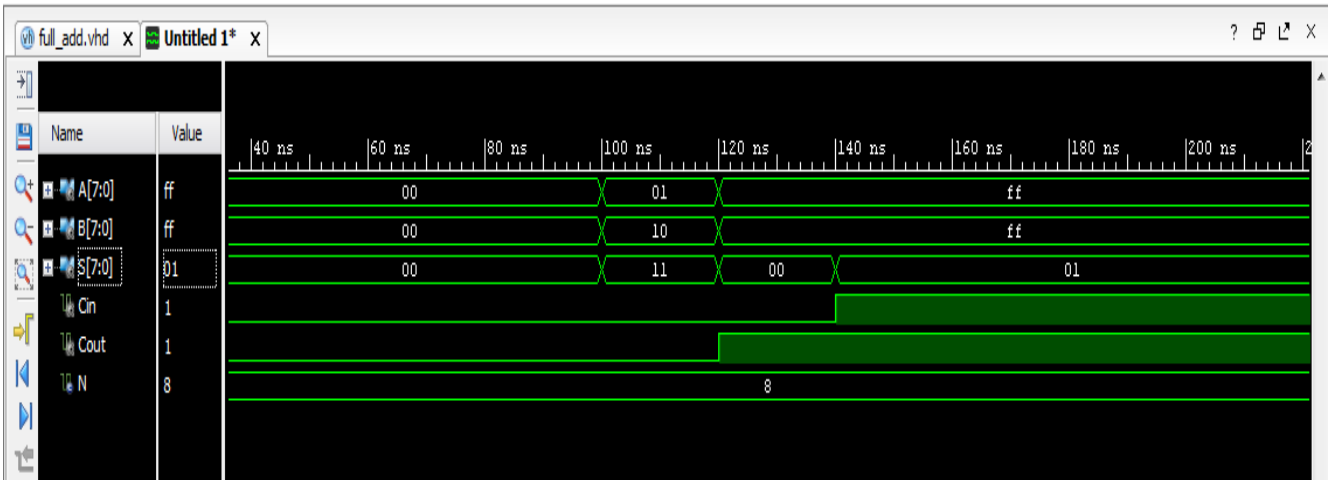


Figure 3 : Résultat de simulation du bloc 'Additionneur complet'

```
----------Code VHDL de la convolution (Gx et Gy)----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity convolution is
   generic ( N:integer:=8);
  Port (   Z1 : in signed(N-1 downto 0);
                  Z2 : in signed(N-1 downto 0);
                  Z3 : in signed(N-1 downto 0);
                  Z4:  in signed(N-1 downto 0);
                  Z6 : in signed(N-1 downto 0);
                        Z5: in signed(N-1 downto 0);
                  Z7 : in signed(N-1 downto 0);
                  Z8 : in signed(N-1 downto 0);
                       Z9 : in signed(N-1 downto 0);
                        treshold :in unsigned (N-1 downto 0);
                        Gx : inout signed(N-1 downto 0);
                        Gy  :inout signed(N-1 downto 0);
                  sum_gx_gy: inout unsigned (N-1 downto 0) ;
```

```
                    pixel_data_out : out unsigned (N-1 downto 0)
                    );
end convolution;

architecture Behavioral of convolution is

                     signal temp1 : signed(N-1 downto 0);
                    signal temp2 : signed(N-1 downto 0);
                    signal temp3 : signed(N-1 downto 0);
           signal temp4 : signed(N-1 downto 0);

          signal  P1 :  signed (2*N-1 downto 0);
          signal  P2 :  signed (2*N-1 downto 0);
          signal  P3 :  signed (2*N-1 downto 0);
          signal  P4 :  signed (2*N-1 downto 0);

          signal t1 : signed(2*N-1 downto 0);
          signal t2 : signed(2*N-1 downto 0);
          signal t3 : signed(2*N-1 downto 0);
          signal t4 : signed(2*N-1 downto 0);

          signal gx_tmp: signed (2*N-1 downto 0);
          signal gy_tmp: signed (2*N-1 downto 0);


          signal zero_tmp : signed(N-1 downto 0);
          signal zero_tmp1 : signed(2*N-1 downto 0);
          signal zero_tmp2 : signed(2*N-1 downto 0);
          signal zero_tmp3 : signed(2*N-1 downto 0);
          signal zero_tmp4 : signed(2*N-1 downto 0);

           signal Cin: std_logic;
          signal Cout : std_logic;
          signal cout1 : std_logic;

          signal X1: std_logic;
  signal X2: std_logic;
           signal X3: std_logic;
           signal X4: std_logic;

           signal Y1: std_logic;
          signal Y2: std_logic;
           signal Y3: std_logic;
          signal Y4: std_logic;

          signal gxx_abs : unsigned(N-1 downto 0);
          signal gyy_abs : unsigned(N-1 downto 0);

component add_sub

                    generic ( N:integer);
                     port(    a : in  signed(N-1 downto 0);
                              b : in  signed(N-1 downto 0);
                               cin : in std_logic;
                              cout : out std_logic;
                               s    : out signed(N-1 downto 0)
                                );
 end component;      --add signé


 component  MULT
          generic ( N:integer);
```

```
            Port (   A : in signed (N-1 downto 0)  ;
                              B : in signed (N-1 downto 0) ;
                             P : out signed (2*N-1 downto 0)
                         );
    end component;  -- MULT

   component soustrac
                    generic ( N:integer);
             port(    a : in  signed(N-1 downto 0);
                              b : in  signed(N-1 downto 0);
                             s   : out signed(N-1 downto 0));
    end component ; -- soustrac


    component full_add
            generic (N: integer:=8);
            port(    a1 : in signed(N-1 downto 0);
             b1 : in signed(N-1 downto 0);
                     s1   : out unsigned(N-1 downto 0);
             cin1: in std_logic;
                     cout1 : out std_logic
                              );
    end component;
begin
       zero_tmp <= (others => '0');
       zero_tmp1  <= (zero_tmp & temp1);
       zero_tmp2  <= (zero_tmp & temp2);
       zero_tmp3  <= (zero_tmp & temp3);
       zero_tmp4  <= (zero_tmp & temp4);

       add1_X : add_sub
                    generic map (N=>8)
                    port map (
                             a=>Z7,
                             b=>Z9,
                             cin=>'0',
                             s   =>temp1,
                             cout=>X1
                             );
        add1_Y : add_sub
                    generic map (N=>8)
                    port map (
                             a=>Z3,
                             b=>Z9,
                             cin=>'0',
                             s   =>temp3,
                             cout=>Y1 );

       mult1_X :  MULT
                   generic map (N=>8)
                    port map (
                             A=>Z8,
                             B=>x"02",
                             P=>P1
                             );
        mult1_Y :  MULT
              generic map (N=>8)
                    port map (
                             A=>Z6,
                             B=>x"02",

                           P=>P3
```

```
                        );

add2_X : add_sub
                generic map (N => 16)
                port map (
                        a=> zero_tmp1,
                        b=>P1,
                        cin=>X1,
                        s   =>T1,
                        cout=>X2
                        );

add2_Y : add_sub
                generic map (N => 16)
                port map (
                        a=> zero_tmp3,
                        b=>P3,
                        cin=>Y1,
                        s   =>T3,
                        cout=>Y2
                        );

add3_X : add_sub
                generic map (N=>8)
                 port map (
                        a=>Z1,
                        b=>Z3,
                        cin=>'0',
                        s   =>temp2,
                        cout=>X3
                        );
 add3_Y : add_sub
                generic map (N=>8)
                 port map (
                        a=>Z1,
                        b=>Z7,
                        cin=>'0',
                        s   =>temp4,
                        cout=>Y3
                        );
mult2_X :  MULT
                generic map (N=>8)
                 port map (
                        A=>Z2,
                        B=>x"02",
                        P=>P2
                        );
   mult2_Y :  MULT
                generic map (N=>8)
                 port map (
                        A=>Z4,
                        B=>x"02",
                        P=>P4
                        );
 add4_X : add_sub
                generic map (N=>16)
                  port map (
                        a=>zero_tmp2,
                        b=>P2,
                        cin=>X3,
                        s   =>T2,
                        cout=>X4
```

```
                        );

        add4_Y : add_sub
                generic map (N=>16)
                  port map (
                        a=>zero_tmp4,
                        b=>P4,
                        cin=>Y3,
                        s   =>T4,
                        cout=>Y4
                        );
          sub_X : soustrac
                  generic map (N=>16)
                  port map (
                        a=>T1,
                        b=>T2,
                        s=>gx_tmp
                        );
          sub_Y : soustrac
                  generic map (N=>16)
                  port map (
                        a=>T3,
                        b=>T4,
                        s=>gy_tmp
                        );
            absolute : full_add
                  generic map (N=>8)
                   port map (
                        a1=>Gx,
                        b1=>Gy,
                        s1=>sum_gx_gy,
                        cin1=>'0',
                        cout1=>cout1
                        );
         -- Outut ports
           Gx <= gx_tmp (N-1 downto 0);
           Gy <= gy_tmp (N-1 downto 0);

         -- absolute
         gxx_abs <= unsigned( abs(Gx));
        gyy_abs <= unsigned (abs(Gy));
        pixel_data_out <= x"FF" when sum_gx_gy > treshold else  ---- 255  en decimal  = BLANC
              x"00" when sum_gx_gy < treshold;                          ----- 0  en decimal  = NIOR
    end Behavioral;
  ----------------------------------------------Fin du code----------------------------------------------------------------------
```

- Résultat de la simulation du code VHDL :

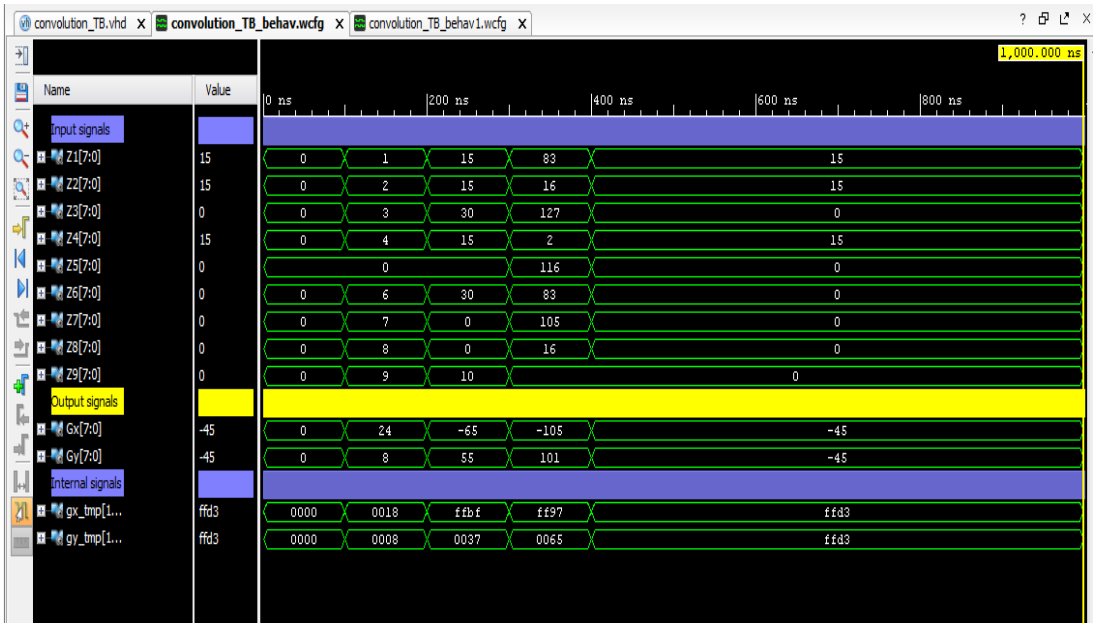


Figure 4:Résultat de simulation du bloc 'Convolution'

---Code VHDL pour lecture de données a partir d'un fichier mémoire -----

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 use STD.textio.all;
 use ieee.std_logic_textio.all;

        entity READ_DATA_FROM_FILE_tb is
                generic ( ligne : integer := 309444);
        end READ_DATA_FROM_FILE_tb;

  architecture Behavioral of READ_DATA_FROM_FILE_tb is
        constant c_WIDTH : natural := 8;
        file input_file : text;
        signal CLK : std_logic;
                signal read_data : std_logic_vector(c_WIDTH-1 downto 0) := (others =>
'0');
        signal cnt : integer := 0 ;
    begin
       clock_process: process
          begin
             CLK <= '0';
             wait for 10 ns;        --temps d'attente
             CLK <= '1';          -- debut du comptage
             wait for 10 ns;
          end process;        ---- CLK

     read_data_file : process
          variable v_ILINE : line;
          variable v_read_TERM : std_logic_vector(c_WIDTH-1 downto 0);
           begin
            file_open (input_file, "write_img.mem",read_mode);
            wait until CLK'event and (CLK= '1');
             while (not endfile(input_file) and (cnt <= ligne-1)) loop
                readline(input_file, v_ILINE);
                cnt <= cnt+1;
                read(v_ILINE, v_read_TERM);
```

```
                    read_data <= v_read_TERM;
                wait for 20 ns;
                end loop;
                 file_close(input_file);
            end process;                    -- read_data_file
stimulus : process
        begin
         -- wait for  100 ns;
           wait;
        end process ;   -- stimulus
  end Behavioral;
----------------------------------------Fin du code----------------------------------------------------------------------
```

- Résultat de la simulation du code VHDL :

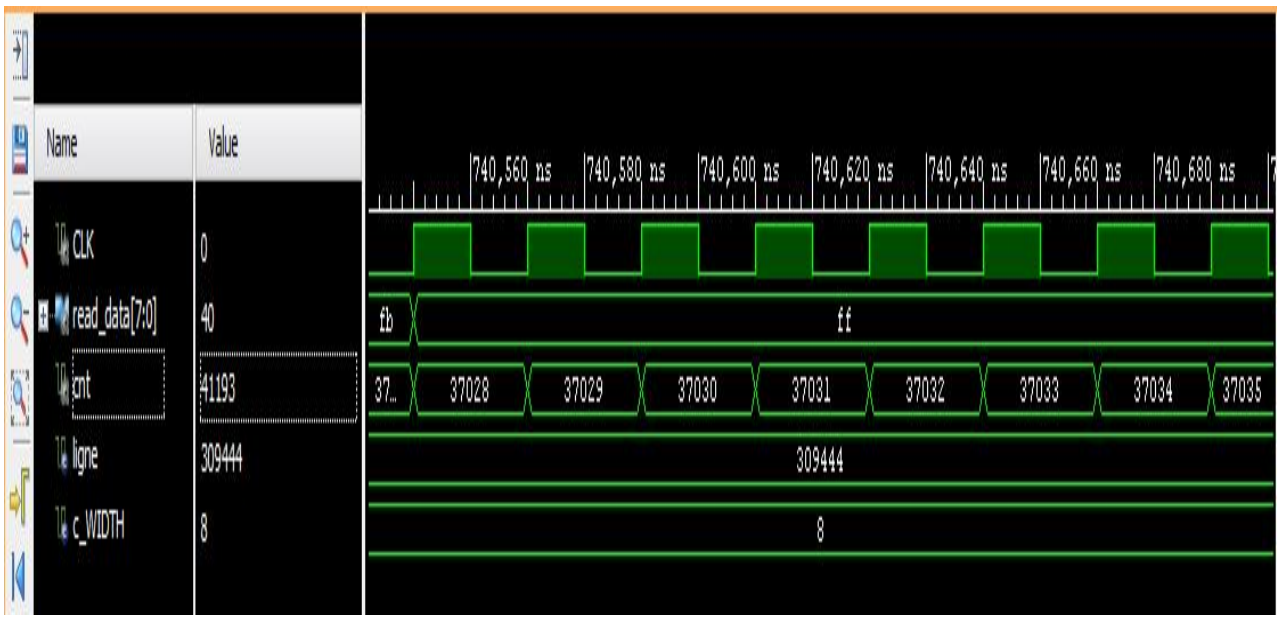


Figure 5 : Résultat de simulation du bloc 'Lecture de données à partir d'une mémoire'

Annexe B :

Interface graphique et code MATLAB

Dans cette partie on appliquera l'opération de filtrage sur des images échantillons des cordons de soudures Pour cela on développera une interface graphique qui nous permettra donc de voir l'effet de chaque algorithme sur des images des échantillons du cordon de soudure.

Nous présentons dans cette section les codes Matlab ainsi que les images de création de notre interface graphique.

Les codes MATLAB

1- Code Matlab du filtre Sobel

```
function[ img_R,img_S,img_R_S]= Sobel_edge(image,thresh)
        newImg = imread ( image );
        grayImage = rgb2gray ( newImg );
        img_S= edge(grayImage   ,'sobel',thresh)  ;
        imshow(img_S)
```

2-Code Matlab du filtre Prewitt

```
        function[ img_P]= Prewitt_edge(image)
                newImg = imread ( image )
                grayImage = rgb2gray ( newImg );
                img_P= edge(grayImage   ,'prewitt') ;
                imshow(img_P)
```

3-Code Matlab du filtre Robert

```
function[ img_R]= Robert_edge(image)
                newImg = imread ( image );
                grayImage = rgb2gray ( newImg );
                img_R= edge(grayImage   ,'roberts');
                imshow(img_R)
```

4-Code Matlab du filtre Canny

```
function[ img_C]= Canny_edge(image)
                newImg = imread ( image );
                grayImage = rgb2gray ( newImg );
                img_C= edge(grayImage   ,'canny') ;
                imshow(img_C)
```