Implantation parallèle de la detection de visages : Méthodologie et implantation sur FPGA

Nicolas Farrugia[1], Franck Mamalet[1], Sébastien Roux[1], Michel Paindavoine[2], Fan Yang[2]

[1] Orange - France Telecom R&D, 28 Chemin du Vieux Chène, 38243 Meylan, France

[2]Laboratoire Electronique Informatique et Image, Université de Bourgogne, UFR Sciences et Techniques, Bat. Sciences de l'Ingénieur, Faculté Mirande, 21000 Dijon, France

[1] prenom.nom@orange-ftgroup.com, fanyang@u-bourgogne.fr, paindav@u-bourgogne.fr

Résumé − Nous présentons dans ce papier une méthodologie d'exploration du parallélisme d'un algorithme de detection de visages et son implantation sur FPGA . L'algorithme choisi est le Convolutional Face Finder (CFF), qui consiste en une cascade de convolutions 2D et de sous-échantillonages. Notre but est de définir une architecture parallèle implantant cet algorithme de manière efficace. Nous présentons une méthodologie d'adéquation algorithme architecture (AAA) utilisant l'outil SynDEx, afin de trouver un bon compromis entre la puissance de calcul, la fonctionalité de chaque Processeur Elémentaire (PE) et l'efficacité de parallelisation. Nous décrivons ensuite une première implantation d'un PE sur un FPGA Virtex 4, en utilisant les blocs de traitement de signal dédiés DSP48. Ce PE fonctionne à une fréquence maximale de 350 MHz et n'occupe que 2% d'un FPGA Virtex 4 SX 35.

Abstract − In this paper, we introduce a methodology for designing a system for face detection and its implementation on FPGA. The chosen face detection method is the well-known Convolutional Face Finder (CFF) algorithm, which consists in a pipeline of convolutions and subsampling operations. Our goal is to define a parallel architecture able to process efficiently this algorithm. We present a dataflow based Architecture Algorithm Adequation (AAA) methodology implemented using the SynDEx software, in order to find the best compromise between the processing power and functionality requirement of each processor element (PE), and the efficiency of algorithm parallelization. We describe a first implementation of a PE on a Virtex 4 FPGA using the DSP48 dedicated blocks. This PE is able to run at a maximum frequency of 350 MHz and occupies only 2% of a Virtex 4 SX35 device.

1 Introduction

La detection et l'analyse de caractéristiques de visages est un domaine de recherche important et qui a de nombreuses applications en sécurité, indexation de contenu, reconnaissance d'identité. De nouvelles applications sont envisagées sur des objets contraints, comme l'amélioration du codage pour la visioconférence mobile ainsi que des interfaces homme-machine intelligentes.

De nombreux algorithmes ont été proposés lors des vingt dernières années [1]. La méthode de détection de visages retenue dans ce papier est le Convolutional Face Finder (CFF), introduit par C. Garcia et M. Delakis dans [2] en 2004. Cet algorithme a les meilleures performances publiées à ce jour sur les bases de visages standards. Le CFF est une approche image basée sur les réseaux de neurones convolutionnels qui permet une détection robuste de plusieurs visages simultanément, de taille et d'apparence variables, tournées de ± 20 degrés par rapport au plan de l'image et de ± 60 degrés dans ce plan. Dans [3], les auteurs ont montré que le CFF peut-être implanté efficacement sur des plateformes logicielles embarquées, tout en conservant les mêmes performances de detection. Les optimizations algorithmiques d'utilisation mémoire menées ont permis de faire une detection de visages à 5 images QCIF (176×144) par seconde utilisant seulement 220 Koctets de mémoire, sur un téléphone mobile SPV M3000. Cependant, la detection de visages étant souvent la première étape d'une chaine complète d'analyse de visages, une accélération significative du système est donc souvent souhaitable. Il existe quelques implantations matérielles de detection de visages [4] mais qui obtiennent des performances de detection plus faibles et des cadences de traitement de 50 images par seconde au maximum. Notre objectif est de réaliser la première implantation matérielle rapide et robuste de detection de visages, en définissant une architecture parallèle pour l'algorithme du CFF.

La conception d'une architecture parallèle permet notamment d'obtenir des gains importants en termes de cadence de traitement et de réduction de la consommation. Pour arriver à ces objectifs, il faut extraire et exploiter efficacement le parallélisme de l'algorithme. Nous avons suivi une méthodologie d'Adéquation Algorithme Architecture (A^3) basée sur des modélisations flot de données de l'algorithme, afin d'explorer les différents types de parallélisme, et de définir la topologie de l'architecture ainsi que les spécifications des processeurs élémentaires qui la composent. L'objectif de cette étude est de prototyper notre architecture parallèle sur une plateforme FPGA.

Ce papier est organisé de la manière suivante : nous présentons l'algorithme du CFF et les travaux antérieurs sur une version optimisée du CFF dans la section 2. Dans la section 3, nous décrivons une méthodologie A^3 appliquée à l'implantation du CFF, ainsi que les résultats de cette

exploration du parallélisme. Nous montrerons en section 4 une première implantation d'un processeur élémentaire sur FPGA Virtex 4. Nous conclurons et donnerons les perspectives de ces travaux en section 5.

2 L'algorithme CFF

Le Convolutional Face Finder a été introduit dans [2] et est basé sur les Convolutional Neural Networks (CNN) introduits par LeCun et al [5].

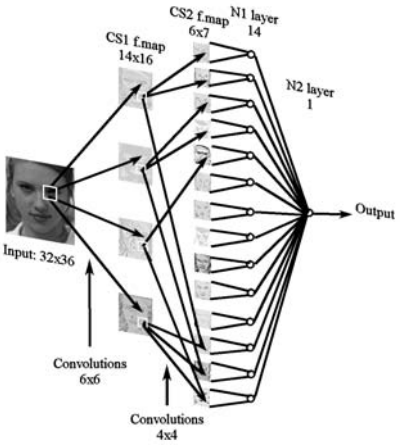


FIG. 1 – Convolutional Face Finder

Dans ce papier nous ne considérons que le coeur du processus de localisation de visages (figure 1). Le CNN utilisé pour implanter le detecteur de visages a été optimisé dans [3], et consiste en une cascade de deux types de couches distinctes.

- Les couches CSi sont appelées couches convolutives, et contiennent un certain nombre de cartes. Chaque élément d'une carte reçoit des entrées d'un petit voisinage des cartes de la couche précédente. Chaque carte peut etre considéré comme une carte de caractéristique, dont le détecteur est fixe et correspond à une simple convolution par un masque appliqué sur les cartes de la couche précédente. Un biais est ajouté aux résultats de chaque convolution. Plusieurs cartes sont utilisées dans chaque couche pour detecter différentes caractéristiques. Une fois détectée, la position exacte de cette caractérisitique importe peu. Ainsi, chaque convolution d'une couche CSi est effectuée avec des pas de déplacement horizontaux et verticaux de deux pixels, ce qui correspond à réaliser des moyennages et des sous échantillonages locaux, correspondant à une réduction par deux des dimensions de son entrée. Ensuite, une tangente hyperbolique est utilisée comme fonction d'activation.
- Les couches Ni sont des couches de classification appliquées après l'extraction de caractéristiques et la réduction de dimenstion des couches CSi. Ces couches correspondent à un perceptron multicouches.

Tous les paramètres (coefficients des convolutions, biais, poids des neurones) ont été appris automatiquement en utilisant une version modifée de l'algorithme de retropropagation du gradient avec momentum [2].

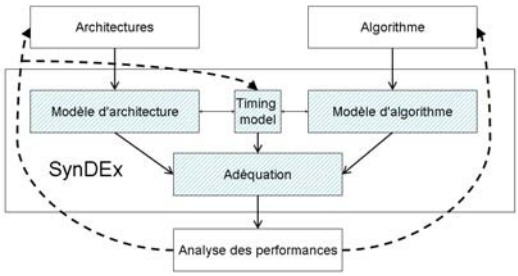


FIG. 2 – Méthodologie d'exploration à l'aide de SynDEx.

Le détail des couches utilisées dans l'algorithme du CFF est donné dans la figure 1 :

- La couche CS1 effectue des convolutions 6×6. Elle contient quatre cartes de caractéristiques.
- La couche CS2 effectue des convolutions 4×4 et contient 14 cartes. Chacune des cartes de CS1 est convoluée par deux masques 4×4 différents, ce qui donne les huit premières cartes de CS2. Les six autres sont obtenues par addition des résultats de deux convolutions 4×4 sur chaque paire possible de cartes de caractéristiques de CS1.
- La couche N1 contient 14 neurones. Chaque neurone est complètement connecté à une carte de CS2.
- La couche N2 contient un seul neurone complètement connecté aux neurones de la couche N1. La sortie de ce neurone est utilisée pour classifier l'image d'entrée comme visage (valeur positive) ou non visage (valeur négative).

Dans la section suivante, nous présentons une méthodologie pour explorer les implantations parallèles possibles en étudiant le parallèlisme de données et le parallèlisme de tâches de l'algorithme du CFF.

3 Méthodologie d'exploration du parallélisme

Nous présentons ici une méthodologie A^3 pour l'exploration du parallélisme de l'algorithme du CFF, utilisant l'outil SynDEx.

3.1 Le logiciel SynDEx

Le logiciel SynDEx (Synchronous Distributed Executive) [6] est un outil de CAO qui met en oeuvre une méthodologie A^3. Il utilise des graphes pour décrire un algorithme et une architecture parallèle, ainsi qu'un modèle temporel. SynDEx implante une heursistique d'optimisation pour un ordonnancement optimal de l'algorithme sur cette architecture. Nous présentons figure 2 la méthodologie suivie pour guider notre implantation du CFF. Sur une architecture parallèle et régulière, nous utilisons SynDEx dans une méthodologie itérative afin d'explorer l'espace de parallélisation algorithmique (où varient le degré de parallélisme de tâche et la granularité des données traitées) et de modèle architectural. Dans la figure 2 les parties hachurées sont celles effectuées sous SynDEx. Chaque itération consiste en :

- un modèle flot de données décrivant l'algorithme,

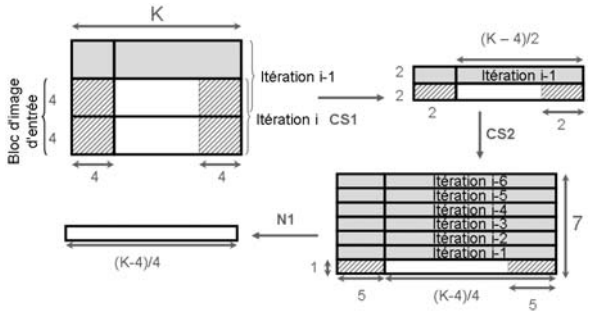


FIG. 3 – Modèle flot de données pour le calcul d'un bloc de données de la couche $N2$ paramétré par K.

- un modèle d'architecture et un modèle temporel,
- une adéquation par l'heuristique de SynDEx qui nous fournit un ordonnancement des tâches,
- une extraction automatique des performances de cet ordonnancement.

Dans cette dernière étape, nous avons défini des métriques nous permettant d'extraire automatiquement un ensemble de mesures de performances à partir des résultats de SynDEx (accélération, efficacité de parallélisation et nombre de transferts) en utilisant des scripts d'analyse syntaxique. Ces mesures nous donnent des indications sur les modifications à effectuer soit sur l'algorithme, soit sur l'architecture, afin d'améliorer les performances et de converger vers des implantations efficaces.

3.2 Modélisation flot de données du CFF

Dans cette partie nous montrons comment nous avons modélisé l'algorithme du CFF pour exploiter différents types de parallélisme.

3.2.1 Parallélisme de données

Les travaux antérieurs [3] ont présenté une analyse des dépendances de données de l'algorihtme pour optimiser l'utilisation mémoire sur les processeurs embarqués, conduisant à un traitement ligne par ligne. Nous généralisons cette approche en divisant l'image d'entrée en blocs de largeur K et présentons dans la figure 3 le modèle correspondant. Ce modèle commence avec un bloc de taille $K \times 8$ dans l'image d'entrée.

Les convolutions CS1 (6×6), CS2 (4×4) et les neurones N1 (6×7) sont appliquées pour produire un bloc d'une seule ligne de la couche N1 de largeur $\frac{K-4}{4}$. Sur la figure 3, les parties hachurées correspondent aux dépendances de données entre blocs successifs. Ce modèle, paramétré par K, permet d'exhiber et de contrôler le parallélisme de données.

3.2.2 Analyse de compléxité

Selon le modèle flot de données précédent, la quantité de calcul de chaque couche peut-etre exprimée en fonction de K. Nous considérons dans cette étude le nombre de multiplications-accumulations (MAC) à effectuer. La table 1 présente pour chaque couche la compléxité d'une tâche pour le calcul d'un bloc d'entrée de largeur K. Par exemple pour $K = 8$, une tâche CS1 calcule une carte de taille $2\times\frac{K-4}{2}$ en utilisant une convolution 6×6, ce qui nécessite au total 36×4 MAC.

TAB. 1 – Nombre de MACs pour chaque tâche, en fonction de K.

Granularité	K	8	12	68	260
CS1	$36 \times (K-4)$	144	288	2304	9216
CS2	$16 \times \frac{K-4}{4}$	16	32	256	1024
Fusion	$\frac{K-4}{4}$	1	2	16	64
N1	$42 \times \frac{K-4}{4}$	42	84	672	2688
N2	$14 \times \frac{K-4}{4}$	14	28	224	896

3.2.3 Parallélisme de tâches

Les couches successives du CFF sont composées de plusieurs convolutions (figure 1) qui peuvent être traitées en parallèle. Le degré maximal de parallélisme pour chaque couche est respectivement 4, 20, et 14 pour CS1, CS2 et N1. Les itérations successives de la méthodologie nous ont conduit à différents modèles de l'algorithme du CFF et nous présentons dans ce papier deux de ces modèles :

- Version 1 : le parallélisme de tâches est maximal (4 CS1, 20 CS2 et 14 N1),
- Version 2 : Aucun parallélisme de tâches, en regroupant tout les CS1, CS2 et N1 dans la même tâche.

3.3 Résultats de l'exploration

Nous avons modélisé sous SynDEx ces différents types de parallélisme ainsi que plusieurs modèles d'architecture en faisant varier la topologie, le nombre de PE et le nombre de liens entre ces PE. Nous nous focaliserons dans cet article sur une architecture en anneau qui a donné lieu aux meilleurs résultats : dans cette architecture, chaque PE est connecté à ses deux plus proches voisins par un lien direct et possède une mémoire locale. Le seul paramètre de ce modèle d'architecture est donc le nombre de PE, N_{PE}. Pour finir, nous avons utilisé un modèle temporel correspond aux formules données dans la table 1. L'exploration a été effectuée sur un bloc d'image d'entrée de taille 256×8, et le jeu de paramètres utilisé est le suivant :

- le degré de parallélisme de données K (ou nombre de blocs à traiter $P = \frac{256}{K-4}$)
- les deux degrés de parallélisme de tâches (versions 1 et 2)
- le nombre de PEs $1 \leq N_{PE} \leq 64$.

La figure 4 présente l'efficacité de parallélisation ($Eff = T_{seq}/(T_{par} \times N_{PE})$, où T_{seq} et T_{par} sont respectivement les temps de calcul sur 1 et N_{PE} PEs) de l'adéquation, en fonction de P, du type de parallélisme de tâches et pour différents nombres de PE. Pour la version 1, le maximum d'efficacité est obtenu pour $P = \frac{N_{PE}}{4}$ car SynDEx distribue les 4 tâches de la première couche sur 4 PE distincts. Cette efficacité maximale décroit rapidement lorsque N_{PE} augmente car cette parallélisation de tâches nécessite de nombreux transferts de données.

La version sans parallélisme de tâches nous permet d'atteindre une meilleure efficacité car les données à transmettre entre des blocs successifs sont peu nombreuses (par-

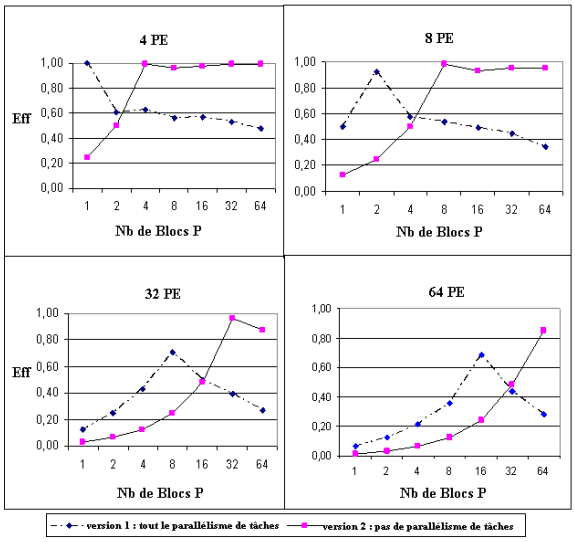
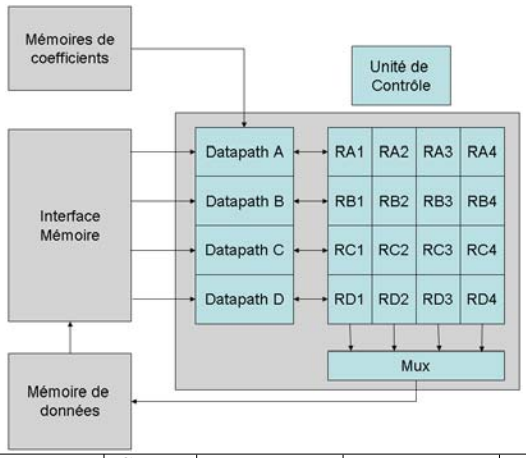


FIG. 4 – Résultats de l'exploration avec SynDEx.



	Slices	Flip Flops	Block Ram	DSP48
Datapath	61	77	0	1
PE complet	478	638	7	4

FIG. 5 – Structure et synhèse FPGA d'un Processeur Elementaire

ties hachurées de la figure 3). Le meilleur compromis entre accéleration et nombre de PE (i.e surface) est atteint pour 32 PEs avec $K = 12$. Nous pouvons conclure de cette étude qu'une architecture efficace est celle formant un anneau de PEs, où chaque PE effectue l'ensemble des couches du CFF sur un bloc d'entrée de 12×8 et transmet seulement des données de recouvrement, en exploitant uniquement le parallélisme de données de l'algorithme.

4 Conception d'un Processeur Elementaire

Nous avons conçu une première version de PE, en nous basant sur les conclusions de l'exploration. Nous savons que chaque PE doit effectuer l'ensemble de l'algorithme. Nous présentons ici une première version de PE (figure 5) composé de 4 chemins de données (Datapaths) contenant chacun 4 registres de sauvegarde, un module de multiplication accumulation. Ces chemins de données sont reliés à des mémoires contenant les coefficients nécessaires aux calculs et les données d'image. Une interface mémoire à été conçue afin d'alimenter efficacement les chemins de données en générant les séquences d'adresses appropriées. Ce processeur élémentaire a été implanté sur un FPGA Xilinx Virtex 4 utilisant des blocs de traitement signal performants (DSP48[7]) pour les MAC. Ceci permet d'obtenir un coeur de processeur élémentaire fonctionnant à 350 MHz et utilisant seulement 3% des ressources (Slices) du FPGA (tableau de la figure 5). Ce PE a ensuite été simulé sur les deux premiers étages de l'algorithme, qui se parallélisent efficacement sur cette architecture. Les performances obtenues nous permettent d'envisager une implantation de 32 PE sur un FPGA Virtex 4 SX 35.

5 Conclusions et Perspectives

Nous avons mis en place une méthodologie appliquée à l'algorithme CFF de detection de visages, afin de définir et dimensionner une architecture parallèle implantant cet algorithme de manière optimale. Nous avons réalisé une première implantation d'un processeur élémentaire de cette architecture. Dans la suite de ces travaux, nous nous intéresserons à l'optimisation de la conception du processeur élémentaire en réalisant une implantation d'un PE complet basée sur des méthodes de synthèse de haut-niveau, puis à la réalisation d'une architecture parallèle. Nous étudierons ensuite la géneralisation de cette approche à d'autres algorithmes de la chaîne d'analyse de visages basées sur les CNN.

Références

[1] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in image : A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 1, Jan. 2002.

[2] C. Garcia and M. Delakis, "Convolutional face finder : A neural architecture for fast and robust face detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 11, Nov. 2004.

[3] S. Roux, F. Mamalet, and C. Garcia, "Embedded convolutional face finder," *ICME IEEE International Conference on Multimedia and Expo, Toronto (Canada)*, 2006, 09-12 July 2006.

[4] T.Theocharides, G.Link, N.Vijaykrishnan, M.J.Irwin, and W.Wolf, "Embedded hardware face detection," *Proceedings of the 17th International Conference on VLSI Design (VLSID 04)*, 2004.

[5] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," 1995.

[6] Y. Sorel, "Syndex : System-level cad software for optimizing distributed real-time embedded systems," *Journal ERCIM News*, vol. 59, 2004.

[7] "Virtex 4 user guide." [Online]. Available : www.xilinx.com