Détection d'obstacles en contexte routier par stéréo vision sur un système intégré sur puce

Mathieu Carrier
Laboratoire LIP6/ASIM
4, place Jussieu
75252 Paris Cedex
mathieu.carrier@lip6.fr

Alain Greiner
Laboratoire LIP6/ASIM
4, place Jussieu
75252 Paris Cedex
alain.greiner@lip6.fr

Résumé

Dans cet article, nous présentons l'implémentation d'une application multi-thread pour la détection d'obstacles en situation de pré-crash. Cette application utilise la stéréo vision et l'algorithme "V-disparité"[1] qui requiert un grand nombre de calculs. L'application est implémentée sur un système intégré sur puce à la fois générique, peu cher, massivement parallèle et multi-processeurs. L'architecture matérielle est particulièrement pertinente pour le domaine de l'automobile vis-à-vis des contraintes de performance, de coût et de flexibilité. Cette application matérielle/logicielle est capable de traiter 40 paires stéréoscopiques par seconde, chaque image étant composée de 256 lignes de 512 pixels (8 bits par pixel), et une disparité jusqu'à 256. Notre architecture se compose de 8 clusters, 30 processeurs généralistes 32 bits et 750ko de mémoire on-chip.

1. Introduction

Les constructeurs automobiles recherchent des systèmes permettant de prévenir et/ou d'éviter les accidents. Plusieurs systèmes ont déjà été développés, notamment par Mercedes-Benz en 2002, Denso et Honda en 2003. Ces systèmes permettent de déclencher différents dispositifs de sécurité. Dans les laboratoires universitaires, différentes plates-formes ont été étudiées comme le véhicule autonome ARGO basé sur GOLD (*Generic Obstacle and Lane Detection*)[2], développé par l'université de Parme. Le challenge DARPA exposent les travaux de recherche d'universités américaines dans le domaine de la détection d'obstacles par le biais d'une compétition. Traditionnellement, les équipementiers et constructeurs automobiles utilisaient des ASIC (*Application Specific Integrated Circuit*), aujourd'hui, remplacés par des systèmes intégrés sur puce (SoC) qui intègrent à la fois des composants logiciels et matériels sur une seule puce. Cette approche permet de réduire significativement les coûts de fabrication et d'augmenter les performances. C'est pourquoi les SoC seront largement utilisés en automobile. Cet article présente une architecture de SoC multi-processeurs (MP-SoC), implémentant une application de détection d'obstacles en situation de pré-crash. Cette architecture matérielle générique permet d'effectuer un grand nombre de calculs grâce à plusieurs coeurs de microprocesseurs utilisés en parallèle, et d'être facilement reprogrammable. Dans cet article, nous présentons les résultats obtenus par mapping d'une application parallèle multi-thread, implémentant une méthode de stéréo vision, basée sur l'algorithme "V-disparité" sur l'architecture mentionnée précédemment.

2. Travaux précédemment menés

Depuis plusieurs années, différentes méthodes ont été évaluées pour la détection d'obstacles en contexte routier. Celles-ci sont basées sur des capteurs extéroceptifs comme les laser, les radar, les lidar ou sur des caméras. Dans notre étude, nous utilisons deux caméras filmant la même scène sous deux angles différents, permettant de reconstruire la scène filmée en 3 dimensions, d'obtenir la distance des objets présents sur la scène et d'en déduire la présence ou non d'obstacles sur la route. La distance d'un obstacle est inversement proportionnelle à sa disparité. La disparité correspond à la différence d'abscisse d'un même point sur l'image gauche et l'image droite, cette valeur est toujours positive, et dans notre cas, fixée à une valeur importante afin de détecter les objets proches. Utilisant la stéréo vision, un certain nombre de systèmes sont basés sur des FPGA en raison de la reprogrammabilité aisée comme la plate-forme de Tyzx [3] capable de traiter 200 images par seconde mais avec une valeur de disparité peu élevée. Han et al.[4] ont récemment développé une autre plate-forme basée sur un FPGA avec une disparité plus importante, permettant la détection d'obstacles proches. Les performances atteintes sont de l'ordre de 60 images par seconde. Ces approches obtiennent donc de bons résultats mais les cartes à base de FPGA ont un coût élevé et ne sont donc pas pertinentes pour le domaine automobile qui requiert un bas coût. D'autres projets sont basés sur des ASIC, avec des matrices d'éléments effectuant des traitements de type SIMD (*Single-Instruction Multiple-Data*). Ces architectures sont très efficaces mais très peu flexibles. Pour le domaine automobile, plusieurs critères sont nécessaires : vitesse de traitement élevé, flexibilité et bas coût. C'est pourquoi, comme décrit en section 4, notre approche est basée sur un parallélisme MIMD (*Multiple-Instructions Multiple-Data*), et l'architecture matérielle choisie est une seule puce, de type SMP (*Symmetric Multi Processor*) à mémoire partagée.

L'algorithme que nous utilisons pour la détection d'obstacles utilise l'approche "V-disparité". Cet algorithme fonctionne quel que soit la surface de la route (il n'y a pas de supposition d'une route plane contrairement à d'autres algorithmes). De plus, la détection est générique même pour le cas d'une occlusion partielle. Enfin, cet algorithme

est robuste vis-à-vis des conditions météorologiques et de luminosité.

3. Nécessité de l'application et contraintes physiques

La résolution des images est de 256 lignes de 512 pixels, 8 bits par pixel soit 128ko par image. La disparité est calculée jusqu'à 256 pixels. Le temps alloué pour le traitement d'une paire stéréoscopique résulte du temps pour parcourir 6m à 130 km/h, soit 150ms, comprenant le temps nécessaire au déclenchement des dispositifs de sécurité, 125ms, soit un temps pour le traitement de 25 ms. Comme le *process* de fabrication pour le MP-SoC est en technologie CMOS 90nm, il est raisonnable d'utiliser une fréquence d'horloge de 300MHz. De ce fait les 25ms correspondent à 7 500 000 cycles. D'autre part, l'application parallèle multi-thread nécessite de la mémoire embarquée pour les communications inter-thread et le stockage des images. Avec ce *process* CMOS 90nm, nous avons décidé d'allouer un budget d'1Mo de mémoire SRAM embarquée. Cette capacité de stockage sera distribuée sur plusieurs bancs mémoire. Nous visons une puce dont l'aire soit au plus de $60mm^2$.

4. Architecture matérielle/logicielle

L'architecture choisie est un système intégré sur puce multi-processeurs clusterisé. Un cluster est un sous-système synchrone contenant un nombre variable de processeurs généralistes 32 bits (jusqu'à 4 par cluster). Tous les microprocesseurs sont des MIPS R3000 RISC (incluant les caches instructions et données). Chaque cluster contient également 2 bancs mémoires locaux (SRAM), et plusieurs périphériques système (timer, mémoire de sémaphores, contrôleur d'interruption, etc). Un "bus système" local interconnecte les processeurs à la mémoire locale et aux périphériques. Les communications entre les clusters sont réalisées à travers un micro-réseau intégré, DSPIN [5]. Cette architecture matérielle est vraiment générique puisque ne comportant que des microprocesseurs généralistes. Pour cette application de détection d'obstacles utilisant la stéréo vision, nous utilisons 8 clusters, chacun d'entre eux possédant 4 processeurs, excepté les 2 clusters permettant l'acquisition des images dans lesquels un processeur est remplacé par un coprocesseur dédié (figure 1).

L'application logicielle initiale a été développée par le LIVIC (Laboratoire d'Interactions Véhicule Infrastructure Conducteurs) sous la forme d'un programme en langage C destiné à un PC mono-processeur. Le temps d'exécution de cette application sur une telle architecture mono-processeur est de 40ms sur des images plus petites et une disparité plus faible. Notre première tâche a été de transformer ce programme séquentiel écrit en langage C en une application multi-thread afin d'exécuter efficacement cette application sur notre architecture multi-processeurs. L'application multi-thread utilise les canaux de communication inter-tâches (fifos mwmr, *multi-writer multi-reader*) fournis par l'environnement de co-design hardware/software, DISYDENT et est entièrement décrite comme un graphe de tâches, dans lequel toutes les tâches s'exécutent en parallèle et communiquent via des canaux de communication point à point, en utilisant des primitives de lecture et d'écriture bloquantes. Conformément à la sémantique des *Kahn Process Networks* (KPN), la synchronisation entre tâches est entièrement réalisée par les données échangées, tant que la contrainte de temps réel est respectée. A la fois les canaux de communication et la resynchronisation, en cas de violation de la contrainte de temps réel, sont supportés par le système d'exploitation temps réel MUTEK.

5. Modélisation de l'architecture et mapping de l'application

La plate-forme SoCLib permet la modélisation et la simulation de systèmes intégrés sur puce multi-clusters multi-processeurs, qui peut être utilisée via le framework DISYDENT. Elle contient un jeu de modèles de simulation, paramétrables, précis au cycle, pour des composants matériels (*IP cores*) réutilisables, écrits en SystemC. Les *IP cores* respectent tous le standard "Virtual Component Interface". Pour notre application, chaque cluster contient 4 microprocesseurs MIPS et 2 bancs mémoire de 64ko chacun. Pour tous les processeurs, la taille du cache a été fixée à 4 ko pour, à la fois, les caches instructions et données (128 lignes de 32 octets). Pour le *process* de fabrication CMOS 90nm, la taille d'un cluster est de l'ordre de $8mm^2$. Pour la partie logicielle, la première étape a été de valider l'application multi-thread complète écrite en langage C. Celle-ci a été compilée pour une station LINUX et liée avec la bibliothèque POSIX. Comme les primitives de communication inter-thread fournies par DISYDENT ont été développées au-dessus du système d'exploitation respectant la norme POSIX, il est possible d'exécuter directement l'application logicielle sur une station LINUX. Dans cette approche complètement logicielle, les contrôleurs d'entrée-sortie s'interfaçant avec les caméras ont été remplacés par 2 threads spécifiques permettant l'acquisition des images par lecture sur le disque dur de la station LINUX. L'étape suivante consistait à implémenter l'application multi-thread sur le SoC multi-processeurs, en remplaçant les contrôleurs d'entrée-sortie. Dans une architecture clusterisée comme la nôtre, le placement des tâches sur les processeurs aussi bien que le placement des buffers de communication sur les bancs mémoire est très important. En effet, une transaction au sein d'un même cluster nécessite environ 10 cycles alors qu'une transaction entre clusters nécessite environ 50 cycles. L'application multi-thread est compilée grâce au cross-compilateur GCC pour le processeur MIPS R3000. L'application complète requiert 750ko de mémoire embarquée, pour, à la fois, les données (FIFOs logicielles) et le code. Le binaire résultant est ensuite chargé dans la mémoire embarquée par le simulateur. On peut alors procéder à la simulation, précise au cycle, matérielle/logicielle.

6. Evaluation des performances

Pour la simulation SystemC, nous avons utilisé le simulateur SystemCass [6], partie intégrante de l'environnement DISYDENT, permettant d'améliorer la vitesse de simulation comparativement au simulateur SystemC. Pour cette architecture à 30 processeurs, la vitesse de simulation est de 4100 cycles par seconde sur un Pentium 4 à 3.2 GHz.

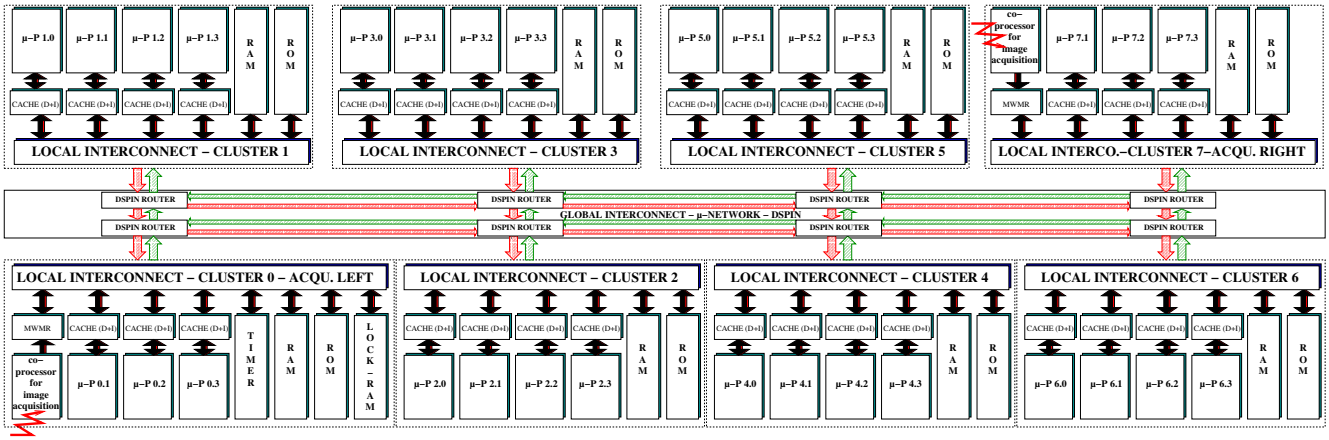
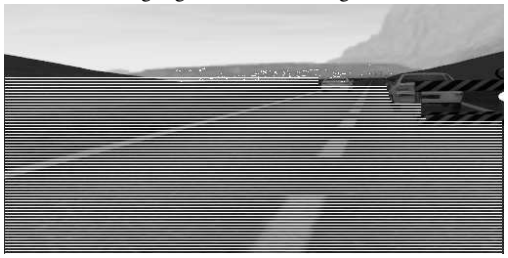


FIG. **1. Architecture à 8 clusters et interconnect DSPIN**

Dans le processus de conception, différentes architectures ont été évaluées, avec un nombre de processeurs différent, permettant d'obtenir des premiers résultats de l'ordre de 12 000 000 de cycles et d'aboutir à des résultats d'environ 6 000 000 de cycles pour le traitement d'une paire stéréoscopique, c'est à dire le temps nécessaire pour obtenir l'espace roulable (sans obstacle) de la route à partir de 2 images d'une même scène prise sous deux angles différents (figure 2). Les résultats détaillés pour l'architecture finale comprenant 8 clusters, 30 processeurs et 2 contrôleurs d'entrée-sortie pour l'acquisition des images sont les suivants. L'initialisation du système nécessite 680 000 cycles, le temps de traitement pour une paire d'images de synthèse sans obstacle est de 5 490 000 cycles, le temps de traitement pour une paire d'images de synthèse avec plusieurs obstacles est de 6 460 000 cycles, le temps de traitement à partir d'images prises sur autoroute est de 7 020 000 cycles. Ces temps d'exécution diffèrent en raison du nombre d'éléments présents sur la scène et donc à traiter. Le temps moyen est de 6 800 000 cycles, soit 22,6 ms pour une stéréoscopique et une horloge à 300 MHz, ce qui est inférieur au temps fixé de 25 ms.

Images gauche et droite originales



Espace roulable extrait de la route

FIG. **2. Entrée et sortie de l'application**

7. Conclusion

Nous avons montré dans ce papier qu'il était possible d'implémenter une application de détection d'obstacles en situation de pré-crash sur un système intégré sur puce multi-processeurs. L'architecture comporte 8 clusters, 30 processeurs 32 bits généralistes et 750ko de mémoire embarquée. L'algorithme utilisé, "V-disparité", a été parallélisé en 30 threads. L'architecture multi-processeurs est capable de traiter les paires stéréoscopiques en moins de 25 ms avec des images de 256 lignes de 512 pixels. Le système intégré sur puce obtenu requiert une surface de silicium de $60mm^2$ en technologie CMOS 90nm. Enfin, l'environnement de conception matériel et logiciel DISYDENT a été vraiment utile dans le processus d'exploration architecturale du système intégré sur puce.

Références

[1] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real time obstacle detection on non flat road geometry through V-disparity representation", in IEEE Intelligent Vehicles Symposium, Versailles, pages 646-651, June 2002

[2] M. Bertozzi and A. Broggi, "GOLD : a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection," IEEE Transactions on Image Processing 7, pp. 62-81, January 1998

[3] J. Woodfill, G. Gordon, R. Buck, "Tyzx DeepSea High Speed Stereo Vision System", in Proceedings of the IEEE Computer Society Workshop on Real Time 3-D Sensors and Their Use, Conference on Computer Vision and Pattern Recognition, June 2004.

[4] D. Han, D.-H. Hwang, "A Novel Stereo Matching Method for Wide Disparity Range Detection", ICIAR 2005

[5] H. Charlery, A. Greiner, E. Encrenaz, L. Mortiez, A. Andriahantenaina, "Using VCI in a on-chip system around SPIN network", Proceedings of the 11th International Conference Mixed Design of Integrated Circuits and Systems, June 2004

[6] R. Buchmann, F. Pétrot, A. Greiner, "Fast cycle accurate simulator to simulate event-driven behavior", ICEEC 2004, September 2004