Adéquation Algorithme Architecture : Modélisations, Implémentations, Optimisations pour Applications Temps Réel Embarquées

Thierry Grandpierre

UNIVERSITÉ GUSTAVE EIFFEL

MÉMOIRE EN VUE DE L'OBTENTION DE L'
HABILITATION À DIRIGER DES RECHERCHES

Spécialité : Mathématiques et STIC - Traitement du Signal et des Images

---

présenté par Thierry Grandpierre

Adéquation Algorithme Architecture : Modélisations, Implémentations, Optimisations pour Applications Temps Réel Embarquées

---

Présenté et soutenu publiquement le 29 janvier 2024 devant le jury composé de :

Rapporteurs	Mme Virginie Fresse	Maîtresse de Conférence HDR	Université Jean Monnet
	Mr Nicolas Gac	Professeur	Université de Paris Saclay
	Mme Fan Yang	Professeure	Université de Bourgogne
Examinateurs	Mr Mohamed Akil	Professeur Emérite	Université Gustave Eiffel
	Mr Venceslas Biri	Professeur	Université Gustave Eiffel
	Mr Matthieu Gautier	Maître de Conférence HDR	Université de Rennes

Remerciements

Je tiens tout d'abord à remercier chaleureusement les membres du jury de me faire l'honneur de leur présence et de pouvoir ainsi me consacrer de leur temps précieux.

Les occasions ne sont pas si nombreuses de pouvoir remercier les personnes qui partagent notre quotidien professionnel ou personnel.

Je profite donc de cette page pour remercier l'ensemble des collègues de l'ESIEE, du LIGM, de la communauté GDR ISIS et SOC2, qui ont pu contribuer à ces travaux, chacun à leur façon.

Je tiens à exprimer ma gratitude envers Mohamed Akil qui m'a offert un environnement de travail idéal dès mon arrivée à l'ESIEE, et m'a permis d'encadrer mes premiers doctorants, en complète autonomie et en toute confiance.

J'aurais aimé que Laurent George puisse également lire mes sincères remerciements. Mais il nous a quitté bien trop vite, en laissant un grand vide mais aussi une riche empreinte de son passage. Je le remercie pour son soutien, la confiance qu'il m'a accordée, la motivation qu'il m'a transmise : le manuscrit est là, promesse tenue !

Merci à Laurent Perroton pour son entrain et sa bonne humeur quotidienne, son soutien, ses nombreux conseils et nos discussions si enrichissantes. Je souhaite aussi remercier Eric Llorens pour son aide précieuse et permanente depuis mon arrivée à l'ESIEE, notamment dans le maintien de toutes les plateformes présentées dans ce manuscrit. Je remercie également Eva Dokladalova pour nos nombreuses discussions tant scientifiques que pédagogiques, et pour m'avoir également offert de co-encadrer une thèse en adéquation avec mes travaux. Merci également à Jean Cousty pour son soutien et l'opportunité qu'il me donne de travailler avec lui en co-encadrant une nouvelle thèse passionnante. Je n'oublie pas non plus Christine Auger qui trouve toujours des solutions à nos tracasseries administratives NP-complètes.

Enfin, last but not least, je crois qu'il n'existe pas de mots suffisamment forts pour remercier mes proches. Je pense à ma dévouée maman, partie trop vite cette année, et mon très courageux papa. Mes trois enfants qui me donnent tellement de force à leur façon. Et bien sûr, leur maman, mon épouse, mon ange, toujours à mes côtés pour m'encourager, me soutenir, m'aider, et ce depuis bien avant les premiers travaux de recherche présentés dans ce manuscrit. Merci pour ton soutien sans faille.

Table des matières

Introduction

Il y a 22 ans j'ai rejoint l'ESIEE en tant qu'enseignant-chercheur après un DEUG[1] SSM (Science et Structure de la Matière), une licence et maîtrise[2] EEA[3] , un DEA[4] en systèmes électroniques pour le traitement de l'information (SETI), une thèse de doctorat au sein du projet SOSSO à l'INRIA Rocquencourt, pendant laquelle j'étais également moniteur (CIES de Versailles) à l'université de Paris Sud Orsay, suivie d'une année en tant qu'ATER (Attaché Temporaire d'Enseignement et de Recherche) où j'ai pu donner des cours à l'école d'ingénieurs (FIUPSO) et en DEA de l'université Paris Sud Orsay et enfin une année en postdoc à l'INRIA Rocquencourt dans le cadre de l'action de développement nationale AEE (Architectures Électroniques Embarquées) où j'ai pu travailler sur des voitures électriques (Cycab) partiellement autonomes.

L'ESIEE est une école d'ingénieurs née en 1904 à Paris sous le nom d'"Ecole Breguet". Elle a changé de nom en 1960 quand elle a été rattachée à la Chambre de Commerce et de l'Industrie de Paris (CCIP). Depuis 2020, l'ESIEE fait partie de la nouvelle université Gustave Eiffel dont elle est membre fondatrice avec l'Université Paris Est Marne La Vallée (UPEM), l'Ecole d'Ingénieurs de la Ville de Paris (EIVP), l'Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux (IFSTTAR), l'Ecole Nationale des Sciences Géographiques (ENSG) et l'École d'Architecture de la Ville et des territoires Paris-Est (EAVP).

L'ESIEE est organisée en 4 départements d'enseignements et de recherche : le département "Informatique" (dont je fais partie depuis mon arrivée), le département "Ingénierie des Systèmes", le département "Santé, Energie et Environnement", et le département "Management de la Technologie et Langues".

La plupart des enseignants-chercheurs de l'école sont rattachés à un laboratoire de recherche de l'université. Je suis ainsi membre du Laboratoire d'Informatique Gaspard Monge (LIGM), Unité Mixte de Recherche CNRS (UMR 8049), dans l'équipe A3SI (Algorithme Architecture Analyse et Synthèse d'Images). Le LIGM dépend de l'école doctorale MSTIC de la COMUE expérimentale Paris-Est Sup.

Depuis mon arrivée à l'ESIEE je mène en parallèle mon activité de recherche et d'enseignement que je retrace donc dans ce manuscrit organisé en 2 grandes parties. La première est une synthèse de toutes mes activités de recherche et d'enseignement. La seconde partie fournit une description détaillée de mes travaux de recherche et s'achève par un bilan et les perspectives à court et moyen termes.

[1] Equivalent L1, L2
[2] Equivalent M1
[3] Electronique Electrotechnique et Automatique
[4] Diplôme d'Etudes Approfondies, équivalent M2

Première partie

Notice personnelle

Chapitre 1

Curriculum Vitae

Thierry Grandpierre

Professeur associé

ESIEE Paris - Université Gustave Eiffel
Département Informatique et Télécom
2 boulevard Blaise Pascal
Cité Descartes, BP 99
93162 Noisy-le-Grand Cedex
Tel. : +33 (0) 1 45 92 65 53
E-mail : thierry.grandpierre@esiee.fr

Formation

- 2001 : Qualification aux fonctions de Maître de conférences, section 61
- 1997 - 2000 : Thèse de doctorat « Modélisation d'architectures hétérogènes distribuées pour la génération automatique d'exécutifs temps réel optimisés » à l'INRIA Rocquencourt, projet SOSSO sous la direction d'Yves Sorel
- 1997 – 1999 : Moniteur d'initiation à l'enseignement supérieur au CIES[1] de Versailles
- 1996 : DEA SETI (Systèmes Electronique pour le Traitement de l'Information) IEF/INSTN (Institut d'Électronique Fondamentale / Institut National des Sciences et Techniques Nucléaires)
- 1994 : Service militaire, base aérienne 128 - Metz
- 1993 : Maîtrise EEA (Electronique Electrotechnique et Automatique)
- 1992 : Licence EEA (Electronique Electrotechnique et Automatique)
- 1991 : DEUG SSM (Science et Structure de la matière)
- 1989 : Baccalauréat série C

[1] Centre d'Initiation à l'Enseignement Supérieur

Postes enseignements et responsabilités administratives

à l'ESIEE

- 2001 - aujourd'hui : enseignant-chercheur ESIEE (professeur associé)
- 2004 - 2010 : responsable de la filière informatique
- 2007 - aujourd'hui : responsable de la salle de réalité virtuelle de l'ESIEE
- 2013 - aujourd'hui : responsable des plate-formes pédagogiques et des relations industrielles du département informatique
- 2017 - 2019 : membre élu du Conseil d'Administration de la COMUE Paris Est (UPE)
- 2018 - 2022 : membre élu de la commission d'évaluation de l'ESIEE
- 2019 - aujourd'hui : responsable de la filière plein temps en cyber-sécurité
- 2023 - aujourd'hui : membre du conseil de perfectionnement de l'ESIEE

hors ESIEE

- 1996 - 1999 : Moniteur rattaché au CIES[2] de Versailles, enseignements de C en M1 à l'université Paris Sud Orsay
- 1999 – 2000 : ATER Université Paris Saclay, cours et TPs de VHDL pour le DEA SETI et TPs de C et VHDL à l'école d'ingénieurs (FIUPSO) de l'université Paris Sud Orsay
- 2000 - 2001 : Postdoc à l'INRIA Rocquencourt pour le développement du véhicule autonome "Cycab"
- 2007 - aujourd'hui : membre du Laboratoire d'Informatique Gaspard Monge (LIGM) de l'Université Gustave Eiffel, UMR CNRS 8049
- 2007 - aujourd'hui : relecteur pour plusieurs revues internationales (IEEE, Springer,...) et conférences françaises (Gretsi, ...)
- 2008 - 2012 : enseignant en formation continue à l'ENSG (école Nationale des Sciences Géographiques)
- 2008 - aujourd'hui : enseignant en Master 2 de l'UGE
- 2016 - 2021 : enseignant en Master 2 à l'IMAC (formation Image Multimédia Audiovisuel Communication) de l'ESIPE (École Supérieure d'Ingénieurs de Paris-Est)

Distinction

- 2014 - "Educator Award for Excellence in Embedded Processing" par Texas Instruments, remis pendant la conférence EDERC (6th European Embedded Design in Education and Research Conference)

[2] Centre d'Initiation à l'Enseignement Scientifique

Chapitre 2

Activités de recherche

2.1 Adéquation Algorithmes et Architectures

2.1.1 Modélisation et Méthodologie

Depuis mon arrivée à l'ESIEE, après ma thèse à l'INRIA Rocquencourt au sein du projet SOSSO et une année d'ATER à Paris Saclay, j'effectue des recherches dans le prolongement de mon doctorat [73]. Ce dernier définit un cadre complet de développement permettant l'implémentation optimisée temps réel d'algorithmes de contrôle, de traitement du signal et des images sur des architectures multi-processeurs. Dans ces travaux je définis un modèle formel d'architectures multi-composants programmables, puis expose des algorithmes d'optimisation basés sur des heuristiques gloutonnes permettant finalement la génération automatique de code distribué sur ces architectures. Durant cette thèse j'ai implanté toute la méthodologie dans un outil graphique : SynDex version 5.

A l'ESIEE, j'ai intégré le département Informatique de l'école, et suis rattaché à l'équipe A3SI (Algorithme, Architecture, Analyse et Synthèse d'Images) du Laboratoire d'Informatique Gaspard Monge (LIGM, UMR CNRS 8049). Mon collègue, Mohamed Akil, m'a immédiatement donné la chance de pouvoir co-encadrer ma première thèse, celle de Linda Kaouane (Cf. p. 42). Ces travaux mettent en oeuvre une extension de mon modèle d'architecture pour des architectures mono-FPGA. Nous y développons également des algorithmes d'optimisation du temps d'exécution d'une application mais aussi du nombre de ressources FPGA mobilisées en jouant sur le taux de déroulage de chaque boucle implémentée dans le FPGA. Ces travaux ont ensuite été étendus aux architectures mixtes (i.e. composées de composants programmables et reconfigurables) par le co-encadrement de la thèse de Oussama Feki (Cf. p. 49), en co-tutelle avec monsieur Masmoudi, responsable du laboratoire LETI de SFax, en Tunisie. Les résultats ont été implantés dans une extension du logiciel SynDEx de l'INRIA : SynDEx-Mixte. Plus récemment, avec ma collègue Eva Dokladalova j'ai pu co-encadrer la thèse de Elias Barbudo (Cf. p. 52) dont le but était de construire une méthodologie de développement générique pour les architectures reconfigurables à gros grain.

2.1.2 Applications temps réel critiques

Au cours des travaux de recherches effectués jusqu'ici, l'optimisation de l'application est effectuée par un ordonnancement hors-ligne des calculs et des communications. Pour couvrir davantage d'applications, j'ai pu étendre mes travaux à l'ordonnancement temps réel dynamique. Ainsi, mon regretté collègue et responsable de département, Laurent George, m'a permis de co-encadrer la thèse de Tristan Fautrel (Cf. p. 63) dont le but était d'optimiser l'ordonnancement de machines virtuelles

temps réel dans un contexte avionique certifié. J'ai pu appliquer les résultats de ces travaux dans le cadre du projet CEOS (Cf. p. 73), un projet FUI (Fonds Unique Interministériel) associant 4 industriels et 3 laboratoires de recherche pour la construction d'un drone volant autonome capable de surveiller l'état de lignes haute tension, de conduites d'eau forcées et de clôtures d'aéroports.

Actuellement, toujours dans le domaine temps réel critique, je co-encadre la thèse CIFRE de Florient Champenois (Cf. p. 81) qui s'effectue chez SAFRAN. Il s'agit d'optimiser les communications temps réel d'un réseau avionique embarqué, un brevet est en cours.

2.1.3 Applications temps réel souples

En parallèle avec les travaux formels de modélisation d'architectures, je mène des recherches plus applicatives dont le but est d'obtenir une implantation optimisée d'algorithmes sur architectures parallèles homogènes et hétérogènes. J'ai ainsi co-encadré la thèse de Nejmeddine Bahri (Cf. p. 85) dont le but était l'impléméntation temps réel embarquée de l'encodeur vidéo haute définition sur architecture multi-DSP. Par le biais de projets industriels, académiques ou d'encadrement de stages, j'ai également mené des recherches sur l'implantation d'algorithmes de traitement d'images sur architectures massivement parallèles : par exemple l'implantation d'algorithmes de "Tone mapping" sur GPU (Graphic Processor Unit), l'implantation d'algorithme de calcul de BRDF (Bi-directional Reflectance Distribution Function) sur architecture GPU ou encore un algorithme de morphologie mathématique. Aujourd'hui, je continue ces travaux en image, et je mène également des travaux dans le cadre de la réalité augmentée appliquée à la radiologie interventionnelle (Cf. p. 99) avec une équipe de radiologues interventionnelles de l'APHP (Henri Mondor).

2.2 Encadrements-Publications

- 5 co-encadrements de thèses de doctorat, et deux thèses en cours
- 4 jurys de thèses de doctorat
- Plus de 50 publications (Cf. chapitre 5) dont :

  16 journaux internationaux

  17 articles de conférences internationales avec comités de lecture

  6 articles dans des revues françaises et colloques nationaux

  3 brevets

  5 manuscrits de thèse
- 4 séminaires invités,
- 749 citations, h-index = 12, index i10 = 13, (selon Google Scholar)

2.3 Collaborations - Montage de projets

2.3.1 Académiques

- 2023 : projet CITYFAB LOOS-EN-GOHELLE (CITYFAB LeG) du projet d'établissement CITYFAB, lauréat du programme ExcellenCES dans le cadre de «

France 2030 », sous la responsabilité de Kristin Speck (Ex-IFSTTAR / UGE, porteuse)

- 2019-2022 : URBANVISION, projet ISITE FUTURE, sous la responsabilité de Roland Bremond (Ex-IFSTTAR / UGE)
- depuis 2019 : collaboration avec Vania Tacher, Radiologue Inteventionnelle de l'APHP Henri Mondor, pour l'aide à la radiologie interventionnelle par réalité augmentée (Cf. 99), participation au montage de la demande de financement Erganéo,
- depuis 2014 : collaboration avec Laurent Stubbe de l'ESO (Ecole Supérieure d'Ostéopathie) et Nicolas Houel, professeur à l'université de Reims, pour le développement de systèmes d'acquisitions. Une proposition de projet ANR Générique (projet ETUGIT) vient d'être déposée ensemble (novembre 2023, porteur N. Houel),
- 2011 - 2014 : montage (porteur T. Grandpierre) et renouvellement du partenariat Hubert Curien (PHC-Utique) avec Monsieur Nouri Masmoudi de l'ENIS (université de Sfax, Tunisie) pour le développement d'une plateforme embarquée de compression vidéo haute définition temps réel (Cf. 85, 49),
- 2011 - 2012 : montage du Partenariat Hubert Curien (PHC Barrande, porteur T. Grandpierre) avec Monsieur Adama Herout de l'université de Brno, République Tchèque, par le biais de ma collègue Eva Dokladalova. L'objectif du partenariat était de travailler une méthodologie globale d'implantation optimisée des applications sur des architectures émergeantes de type MPSoC : embarquées, multi-coeurs, reconfigurables ou adaptatives. Nous avons développé ces travaux par l'échange de stagiaires de Master 2 dont Pavel Karas (Cf. 97).
- 2005 - 2009 : participation au montage et à l'animation du "GIS AV" (Groupement d'intérêt Scientifique Aménagement Virtuel) sous la responsabilité de Mohamed Akil (porteur). Ceci a permis d'équiper notre laboratoire de sa première salle de réalité virtuelle.

2.3.2 Université d'été, séjours recherche

- 2019 : séjour au F'SATIE (French South African Institute of Technology) dans le cadre du programme ERASMUS+, Prétoria, Afrique du Sud : 4h de cours et 4h TP sur le traitement des images embarqué. Ce séjour a permis de rencontrer des chercheurs en robotique du F'SATIE, ainsi que des étudiants en Master pour préparer le co-encadrement d'une thèse commune. Cependant, le COVID survenu l'année suivante, a bloqué le processus.
- 2014 : learning expedition à Boston (Harvard University, MIT, MIT Media Lab, Wyss Institute Harvard, Babson College, NERD Microsoft, IGEM , Cambridge Innovation Center). Grâce aux rencontres durant ce séjour de 8 jours, j'ai pu, entre autres, accueillir des stagiaires du MIT dans le cadre du programme MIT-France, développer de nouvelles pratiques pédagogiques, découvrir de nouveaux laboratoires et thématiques de recherche.
- 2011 : séjour recherche dans le cadre d'un Partenariat Hubert Curien (PHC-Utique), travail avec doctorants et chercheurs.

- 2010 : invitation pour donner une formation théorique et pratique sur les DSP dans le cadre des journées DSP organisées par le LETI de Sfax, Tunisie. Cette formation de 16h m'a permis de préparer un Partenariat Hubert Curien-Utique (PHC-Utique) avec Monsieur Nouri Masmoudi et de rencontrer nos futurs doctorants en co-tutelle, Nejmeddine Bahri et Oussama Feki.
- 2007 : invitation pour donner 8h de cours et 8h de TP sur les DSP [1] dans le cadre de l'université d'été "Carte à puce, architectures et protocoles", INPT, Rabat, Maroc,
- 2002 : séjour à l'université de Belo horizonte (UFMG, Brésil) dans le cadre du programme CAPES-COFECUB (responsable ESIEE : M. Akil, responsable UFMG : Arnaldo de A Araujo) 8h de cours et 8h de TP sur la méthodologie AAA. J'ai pu nouer des contacts avec les responsables de filières ce qui a ensuite permis des échanges d'étudiants entre l'ESIEE et l'université UFMG.

2.3.3 Industriels

- 2019 - 2021 : projet FUI CEOS, "Drone autonome pour l'inspection d'ouvrages d'art", avec L. George, porté par Stéphane Menoret (Thales), Thales, INRIA, Université de Nancy, Alérion, EDF, ENEDIS, Aéroport de Caen (Cf. p. 73),
- 2022 : porteur ESIEE de l'ANR iCARE (Artificial Intelligence of things and connected digital twins for optimal guiding and resilience), non retenu pour passer l'étape 2,
- 2013 : participation au montage du projet POLEIAS (Prediction Of Lesion Evolution In Acute Stroke) pour appel à projet européen FP7-ICT-2013-10, non sélectionné,
- 2007 : porteur du projet ANR Architecture du futur MAXISS (Methdology for Architecture eXploration and optimIzation od diStributed System on a chip), non sélectionné.

2.4 Encadrements

2.4.1 Thèses co-encadrées soutenues

1. Elias Barbudo, "Towards Efficient Reuse of Software Programmable Streaming Coarse Grained Reconfigurable Architectures", taux d'encadrement 50%, co-dirigée par Eva Dokladalova, soutenue le 29 juin 2021 (Cf. p. 52),
2. Tristan Fautrel, "Analyse et ordonnancement d'un système hiérarchique virtualisé composé d'applications temps réel strictes", taux d'encadrement 50%, co-dirigée par Laurent George, soutenue le 6 mai 2021 (Cf. p. 63),
3. Oussama Feki, "Contribution à l'implantation optimisée de l'estimateur de mouvement de la norme H.264 sur plates-formes multi composants par extension de la méthode AAA", taux d'encadrement 80%, co-dirigée par Mohamed Akil, soutenue le 13 mai 2015 (Cf. p. 49),

[1] Digital Signal Processor - Processeur de traitement du signal

4. Nejmeddine Bahri, "Etude et conception d'un encodeur vidéo H264/AVC de résolution HD sur une plateforme multi cœur", taux d'encadrement 80%, co-dirigée par Mohamed Akil, soutenue le 9 Novembre 2015 (Cf. p. 85),

5. Linda Kaouane, "Formalisation et optimisation d'applications s'exécutant sur architecture reconfigurable", taux d'encadrement 40%, co-dirigée par Mohamed Akil, soutenue le 14 décembre 2004 (Cf. p. 42).

2.4.2 Thèses co-encadrées en cours

- Florient Champenois, "Analyse et ordonnancement d'un système hiérarchique virtualisé composé d'applications temps réel strictes", taux d'encadrement 32%, co-dirigée par L. George puis E. Borde, puis J.L. Scharbag, échéance prévue : fin 2023 (Cf. p. 81).

2.4.3 Encadrements de stages

Principaux stages de niveau Master 2 que j'ai encadré :

- 2022 : Alexandre Sanchez, "Implémentation temps réel de la BRDF sur GPU", projet ISITE Urban Vision, avec R. Brémond, F.Vienne (PICS-L, Ex-IFSTTAR/UGE), 3^e année cycle ingénieur ENSISA, Université de Haute Alsace,

- 2021 : Huiying Dai, "Immersive virtual visit of a district with multimodal transportation model and panoramic street views", projet ISITE Future / Eiffage E3S avec R. Belaroussi du COSYS-GRETTIA, 3^e année cycle ingénieur ESIEE/UGE,

- 2020 : Mathias Hudelot, "Etude en vue de l'implémentation d'une BRDF mesurée", Master 2 Informatique, UGE,

- 2019 : Christopher Pico, "Réalité Virtuelle pour la radiologie Interventionnelle", projet avec l'APHP et l'université de Créteil, projet soutenu par Erganéo, Master 2 Informatique, UGE,

- 2019 : Taniya Das, "Real time accurate localization using Ultra Wide Band", research internship, avec L. George,4th Year Undergraduate B.Tech, Electronics and Communication Engineering, Maulana Azad National Institute of Technology, Bhopal, India

- 2018 : Alvaro Baquedano, "Detection and tracking of target for automatic landing drones", Université de Navarre, Espagne, avec E. Dokladalova, Master 2 en Ingénierie de Télécommunications, Universidad Pública de Navarra,

- 2015 : Varun Deshpande, "Real Time communication using Time Sensitive Network", avec L. George, Master 2 Informatique, UGE,

- 2012 : Pavel Karas ,"Implantation of morphological filters on GPU", avec E. Dokladalova, Master 2 Masaryk University Brno, République Tchèque,

- 2011 : Baptiste Delporte, "Virtual Gyroscope on a microcontroleur using an accelerometer", avec L. Perroton et M. Akil et la société Freescale, 3^e année cycle ingénieur ESIEE/UGE,

- 2010 : Romain Naour, "Développement et implémentation d'un algorithme de détection d'horizon sur DSP", projet CNES - PERSEUS[2] - AETNA, 3^e année cycle ingénieur ESIEE/UGE,
- 2008 : Yé Peisheng, "Image processing on Xilinx Virtex II Pro mixed architecture", 3^e année cycle ingénieur ESIEE/UGE, programme N+I,
- 2006 : Tony Knorr Mabassa,"FPGA fuzzy code implementation using SynDEx-IC", F'SATIE, avec M. Akil, 3^e année cycle ingénieur F-SATIE (French South African Institute of Technology Institute), Pretoria, Afrique du Sud.

J'encadre également chaque année deux à trois projets élèves de niveau Master 1 et Master 2. Ces 2 dernières années, trois de mes équipes ont remporté le prix de l'innovation du "Jour des projets"[3] ESIEE avec les projets "Holoslide" en 2023 et "VR pour soulager les douleurs fantômes" en 2022 ainsi que le prix coup de coeur du public pour le projet "Hololens2 : radiologie interventionnelle avec réalité augmentée" en 2022.

2.5 Autres activités

Je participe à 3 GDR (Groupement de Recherche) : le GDR ISIS (Information Signal Image Vision), le GDR SOC2 (System On Chip, Systèmes embarqués et Objets Connectés) et le GDR Sécurité Informatique.

2.5.1 Relecteur

- Journal of Real-Time Image Processing,
- IEEE Sensors Journal,
- IEE Transactions on Computers,
- Applied Sciences,
- Journal of Low Power Electronics and Applications.
- Gretsi.

2.5.2 Jury de doctorat

- Soutenance de thèse de Yareb Elloumi, Université Paris Est, le 7 décembre 2013,
- Soutenance de thèse de Imen Werda, Université de Sfax, le 14 Janvier 2010,
- Soutenance de thèse de Mu Pengcheng, INSA Rennes, le 7 Juillet 2009,
- Soutenance de thèse de Ghislain Roquier, INSA Rennes, le 14 Novembre 2008.

2.6 Distinction

Educator Award for Excellence in Embedded Processing par Texas Instruments, remis pendant la conférence EDERC (6th European Embedded Design in Education and Research Conference), 2014.

[2] https ://www.perseusproject.com/
[3] https ://jdp.esiee.fr/laureats-2023/

Chapitre 3

Activités, responsabilités et vie institutionnelles

3.1 Responsabilités de filières

3.1.1 Filière Informatique (2004 - 2010)

Je suis arrivé à l'ESIEE en septembre 2001. Après deux ans j'ai été nommé responsable de la filière Informatique, c'est à dire responsable des 2 dernières années du cycle ingénieur (E4 et E5 dans le vocable ESIEE), soit un total alors d'environ 80 élèves (le chiffre fluctuant selon les années). La section 3.1.3, donne un aperçu du rôle et des tâches d'un responsable de filière à l'ESIEE. Après 6 années passées seul à gérer cette filière, j'ai demandé fin 2010 à être déchargé de cette responsabilité pour pouvoir me consacrer d'avantage à la recherche et au développement des plateformes que j'ai monté. La charge de travail de ce poste était telle que j'ai été remplacé par deux collègues.

3.1.2 Filière Cybersécurité (2019 - aujourd'hui)

Après plusieurs sollicitations, j'ai accepté en septembre 2019, de prendre la responsabilité de la toute nouvelle filière Cybersécurité de l'ESIEE. J'ai rapidement été rejoint par Eric Renault, et nouveau collègue et professeur HDR à l'ESIEE. Depuis nous nous partageons la responsabilité de la filière, il se charge ainsi des E4 (Master 1) et moi des E5 (Master 2).

En plus des tâches classiques d'un responsable de filière (développées ci-dessous), nous avons dû monter le dossier de labellisation "SecNumEdu" auprès de l'ANSSI[1] pour notre filière : "l'objectif de cette labellisation est d'apporter une assurance aux étudiants et employeurs qu'une formation dans le domaine de la sécurité du numérique répond à une charte et des critères définis par l'ANSSI en collaboration avec les acteurs et professionnels du domaine (établissements d'enseignement supérieur, industriels...)[2]."

3.1.3 Rôle du responsable de filière

A l'ESIEE, le responsable de filière assure d'abord le lien entre les élèves et la direction des études. A ce titre il participe régulièrement aux comités pédagogiques, comités de filières, différents groupes de travail et bien sûr les jurys (jurys de fin de

[1] Agence Nationale de la Sécurité des Systèmes Informatiques
[2] https://www.ssi.gouv.fr/entreprise/formations/secnumedu/

semestre, jurys d'attribution de diplôme, jurys de départ puis de retour de l'étranger etc.). Le responsable de filière assure également la définition et le suivi des programmes pédagogiques (une vingtaine d'unités de 30 heures réparties sur les 2 années), la recherche et la sélection des intervenants, la planification des interventions en lien avec le service dédié, le suivi du dossier administratif en lien avec les deux assistantes de département.

Il participe également aux différentes actions menées par les partenaires industriel, en particulier dans le cas de chaires entreprise pour lesquels il faut proposer un accueil et une attention privilégiée, participer aux évènements académiques de l'entreprise etc. Il interagit également avec les autres formations du campus (les différents masters de l'Université Gustave Eiffel, les écoles du campus comme l'ENSG[3], l'ENPC[4], etc.).

Il gère la recherche de stages, qu'il valide avant signature des conventions, et dont il assure en partie le suivi et le processus d'évaluation (attributions de suiveurs, soutenances, etc.).

Enfin, il participe à la communication auprès des élèves de la filière (bilans), mais aussi vers ceux qui souhaitent rejoindre la filière. Cela consiste donc à participer à différentes présentations, forums, et autres journées portes ouvertes.

3.2 Chargé du développement et des plate-formes

Etant impliqué dans les relations régulières avec nos partenaires industriels et ayant élaboré plusieurs plate-formes pour la pédagogie et la recherche j'ai été nommé chargé du développement depuis 2014 pour le département informatique. A ce titre je participe régulièrement aux réunions "Infrastructures" (INFRA) de l'école et organise régulièrement des partenartiats industriels comme décrit ci-dessous.

3.2.1 Relations industrielles

Texas Instruments

Depuis le milieu des années 1980, par les travaux de Geneviève Baudoin, Férial Viroleau et Olivier Venard, l'ESIEE collabore avec Texas Instrument sur les architectures à base de processeurs DSP[5], aux niveaux recherche et pédagogique. Cette collaboration s'est entre autres traduite par la création de matériel pédagogique (cours, travaux pratique) offert par TI au monde académique. Lors de mon arrivée à l'ESIEE j'ai repris peu à peu la responsabilité de cette collaboration, qui s'est concrétisée par de nombreuses donations de matériels (notamment pour nos cours de première année en introduction à l'architecture des ordinateurs). Nous avons également organisé des challenges avec remises de prix, et surtout plus récemment par l'ouverture dans les locaux de l'ESIEE de la "Texas Instrument Innovation Gateway"[6], un espace unique, dans notre bibliothèque où les élèves peuvent voir des démonstrations de cartes d'évaluation, des réalisations d'élèves. L'une des principales originalité de cette bibliothèque dont je suis responsable, réside dans la possibilité pour les élèves de pouvoir emprunter des cartes TI comme des livres.

[3] Ecole nationale des sciences géographiques
[4] École des Ponts ParisTech
[5] Processeur de traitement du signal (Digital Signal Processor)
[6] https://perso.esiee.fr/~grandpit/TIIGW/TIIGW.html

NxP

Sur le même schéma que pour Texas Instrument, j'ai également repris la collaboration entre NXP (initialement Freescale) et l'ESIEE. Nous avons ainsi obtenu des donations de matériel, mais j'ai également pu organiser deux finales de la NxP Cup à l'ESIEE (finale Européenne en 2013 puis qualification Française en 2019) : une course mondiale de mini-voitures autonomes sur un circuit fermé en intérieur. J'encadre ainsi chaque année des équipes d'élèves pour participer à ce challenge.

IBM

Dans le cadre d'une unité d'enseignement autour du métier d'architecture IT, j'ai été responsable d'une chaire avec IBM. Cette dernière a permis l'acquisition d'un Blacecenter-H [7] dont je suis co-responsable et co-administrateur.

3.2.2 Plate-formes

Salle de réalité virtuelle

A des fins d'enseignements, de projets et de recherche j'ai conçu une salle de réalité virtuelle financée initialement par le GIS[8] "Aménagement Virtuel". Dans sa version actuelle elle est équipée de 3 grands écrans sur lesquels 3 vidéo projecteurs stéréoscopiques projettent des images générées en temps réel. La plate-forme comporte également des casques de réalité virtuelle (Occulus, Occulus Quest, HTC Vive ...), et des Hololens pour la réalité augmentée. J'encadre chaque année des élèves en projet (par exemple pour la modélisation 3D de l'ESIEE ainsi que des salles blanches), les développements s'effectuent aujourd'hui principalement avec Unity et Unreal Engine après avoir débuté directement en OpenGL puis OpenSceneGraph.

Bladecenter

Dans le cadre de la chaire informatique avec IBM, nous avons pu acquérir une équipée d'un Bladecenter-H, recevant 14 lames de calcul (HS21 à base Intel, QS22 à base de processeurs IBM CELL), et sa baie de stockage SAN DS4300. J'ai participé à son assemblage puis à son administration avec l'aide de mon collègue Eric Llorens, technicien de département.

Laboratoires

Par ma responsabilité des plate-formes du département informatique, je suis l'interlocuteur pédagogique privilégié pour trois laboratoires d'enseignements équipés de matériels de mesure, imprimantes 3D, graveuse de circuits imprimés etc.

3.3 Comités, conseils, membre élu

3.3.1 Commission d'évaluation de l'ESIEE

J'ai été élu à la commission d'évaluation de l'ESIEE sur la période 2018-2022. Cette commission a pour rôle de gérer le recrutement des enseignants de l'ESIEE (fiches

[7] une baie pouvant accueillir 14 lames serveurs basés sur des processeurs Intel, IBM Power, IBM CELL

[8] Groupement d'Intérêt Scientifique

de postes, diffusions, auditions) ainsi que leur promotion (changement de titres et d'échelon).

3.3.2 COMUE Université Paris Est

J'ai été membre élu (conseil B2) au conseil d'administration de la COMUE Paris-Est sur la période 2017-2019 [9].

3.3.3 Conseil de perfectionnement de l'ESIEE

Je viens d'être nommé pour représenter le département informatique au conseil de perfectionnement de notre école.

3.4 Actions diverses

3.4.1 Promotion de l'établissement

Je participe chaque année depuis mon arrivée à la plupart des Journées Portes Ouvertes (3 à 4 par an) que nous organisons, notamment pour présenter la salle de réalité virtuelle. Je participe également à d'autres évènements comme les cordées de la réussite, les visites de lycéens, des démonstrations lors de la fête de la science, stages de troisième et secondes, etc.

3.4.2 Journée des projets ESIEE

Chaque année, en juin, nous organisons la journée des projets : un évènement festif qui réuni tous les élèves de l'école pour qu'ils puissent exposer leurs réalisations de l'année dans le grand hall de l'école. Des prix sont remis aux élèves, je suis membre du jury tous les ans (sauf en 2022) ainsi que membre du comité de pilotage.

3.4.3 Innovation pédagogique

- Présentation, avec Jean-François Bercher, de la plate-forme AMC *AMC*[10] au journées de l'innovation pédagogique organisées par le CIPEN (Centre d'Innovation Pédagogique et Numérique) de l'Université Gustave Eiffel, le 7 Novembre 2019,
- Journées de la pédagogie de l'ISITE FUTURE[11],
- Présentation lors du séminaire Mathworks, le 26 mars 2009 (retransmis par Webex à ses clients), "Prototypage rapide d'une chaîne de segmentation d'image sur DSP avec Simulink".

[9] https://www.paris-est-sup.fr/fileadmin/Fichiers/UPE/Universite/Instances/CA/CR-RdD_CA/2017/1-_CA_CR_du_4_juillet_2017.pdf

[10] https ://www.auto-multiple-choice.net/index.fr

[11] http://www.future-isite.fr/accueil/les-actualites-du-projet-future/actualites-du-projet-future/?no_cache=1&L=0&tx_ttnews%5BbackPid%5D=29302&tx_ttnews%5Btt_news%5D=50574

3.4.4 Groupes de travail

Je participe régulièrement à plusieurs groupes de travail, notamment :

- évaluation par compétences,
- travaux et aménagements du Bâtiment 6 qui accueillait les anciennes salles blanches de l'ESIEE,
- gestion des vacataires,
- innovation pédagogique.

3.4.5 Vulgarisation scientifique

J'attache beaucoup d'importance à l'éveil de la curiosité pour les sciences auprès des jeunes que je concrétise par des actions ponctuelles ou régulières comme :

- participation au tournage de l'émission "E=M6" pour présenter le principe des scanners 3D, diffusé[12] le 4 décembre 2016,
- participation à un reportage France 3 pour le journal du "19-20" afin de présenter le principe de notre bibliothèque matérielle "Texas Innovation Gateway", diffusé[13] le 12 mars 2018,
- bénévole et membre du conseil d'administration de l'association Science Ouverte[14] depuis 2014,
- participation à la fête de la science (par exemple pour présenter la réalité virtuelle auprès de collégiens en classe de 3ème pendant un cours de physique-chimie),
- organisation de "cordées de la réussite" auprès de collégiens et lycéens.

[12] https://tv-programme.com/e-m6_magazine/replay/e-m6-vetements-loisirs-sante-comment-la-technologie-3d-va-58447abca3c2a

[13] https://youtu.be/ocPevhukRcY

[14] https://scienceouverte.fr/

Chapitre 4

Activités d'enseignement

Chaque année j'effectue environ 135 heures équivalent TD de cours magistraux , 130 heures de travaux pratiques et j'encadre également plusieurs projets de premier et second cycles ingénieur (de 130 à 200 heures équivalent TD en fonction des années).

Ce chapitre résume l'ensemble des unités d'enseignement dans lesquelles j'interviens (ou intervenais car certaines ont été remplacées ou supprimées). Elles sont classées par filières plein temps puis apprentissage, puis extérieures. Pour chaque filière elles sont organisées en années d'enseignement (Tronc commun / Cycle préparatoire puis filières spécialisées / Cycle ingénieur ou Masters).

Bien sûr j'ai adapté mes cours et TPs au distanciel quand ce fut nécessaire lors des confinements de 2020 et 2021 (enregistrement vidéo des cours, mise en place de quizz interactifs pendant les cours, accès à distances aux cartes de développement de TP, examens à distances…).

4.1 Filières Temps plein ESIEE

Dans les lignes qui suivent, les années et les titres qui sont en gras indiquent que je suis responsable de l'unité : je conçois cours, sujets et corrections de TPs, sujets d'examens et leurs corrections. L'effectif moyen de chaque unité est aussi indiqué.

4.1.1 Tronc commun (L1 à L3)

Actuels

- **Depuis 2007, "Introduction à la programmation en C"** (code IGI-2203), 2e année ("E2", équivalent L2, 265 élèves) : 15h de cours et 15h TP,
- **Depuis 2022, "Introduction à la ligne de commande Linux"** (code IGI-2202), 2e année ("E2", équivalent L2, 265 élèves) : 4h de cours et 8h TP.

Passés

- 2018-2022, "Programmation des micro-contrôleurs" (code IGE 3016), 3e année ("E3", équivalent L3, 300 élèves) : 4h de TD et 18h de TP. Programmation en C bas niveau des ARM Cortex-M3 MSP432 (famille TI) et de leurs principaux périphériques (Timer, UART),
- **2014-2017 "Introduction aux sciences de l'ingénieur : robotique**, 1ère année ("E1", équivalent L1, 50 élèves) : 8h de Cours/TP/Projet. Cours optionnel de découverte de la robotique par la construction d'un robot mobile, programmation graphique Ardublock, logiciel Energia adapté de Arduino pour les cartes MSP432P401R de Texas Instrument.

- **2013-2014 , "Introduction à la programmation des microprocesseurs"**, 1ere année ("E1", équivalent L1) : Sélection d'une nouvelle architecture matérielle pour le cours E1 "Introduction à la programmation des microprocesseurs" (IGI-1203). Nous utilisions alors les architectures à base de 68000, j'ai eu pour mission d'explorer les solutions de remplacement (type de processeur, cartes d'évaluation). Après avoir exploré les gammes de microcontroleurs disponibles à l'époque (PIC de Microchip, Coldfire de NxP, ARM Cortex M3 de TI et STM), le choix s'est porté sur l'architecture Cortex-M3 Stelaris, et s'est concrétisé par un partenariat avec TI qui nous a fourni 200 kits Evalbot (carte d'évaluation robotique basée sur M3 Stelaris, comportant 2 moteurs, des capteurs, écrans OLED, micro, haut parleur, ethernet, etc.) permettant de prêter pour la première fois chaque kit à chaque binôme pendant la durée de l'unité. Il a été aussi nécessaire de former les collègues à cette nouvelle architecture et de créer les supports pédagogiques.
- **2002-2016, "Conception de systèmes embarqués multimédia"** (code IN213, optionnelle), 2e année ("E2", équivalent L3, 50 élèves) : unité orientée projet durant laquelle les élèves concevaient des prototypes associant des microcontrôleurs, capteurs, actionneurs et communications radio.
- **2012-2016, "Programmation des interfaces graphiques"** (code IGI 3604), 3e année ("E3", équivalent L3, 50 élèves) : co-responsable avec O. Venard du cours. 30h élèves répartis en 8h de cours et 22h de TP de programmation C++ sous visual Studio pour développer des IHM.

4.1.2 Filière Informatique (plein temps, M1 et M2)

Actuels

- **Depuis 2003, "Algorithmique et programmation des systèmes distribués"** (code INF-4201), 2e année cycle ingénieur ("E4", équivalent M1, 40 élèves en moyenne) : 30h d'enseignements réparties en 12h de cours et 18h de TPs, programmation système des communications par socket, algorithmique distribués (horloges logiques, exclusions mutuelles etc.).
- **Depuis 2016 "Architecture des processeurs RISC"** (code INF-4101), 2e année cycle ingénieur ("E4", équivalent M1, 40 élèves en moyenne). Co-responsable avec E. Dokladalova. Pour ma part 6h de cours et 6h de TPs. Etude de l'architecture interne, optimisations assembleur (pipeline), simulation.
- **Depuis 2010 "Architecture et programmation parallèle"** (code INF-4302), 2e année cycle ingénieur, ("E4", équivalent M1, 40 élèves en moyenne). 14h de cours, 16h de TP. Exploitation du parallélisme intra processeur et interprocesseur : étude SIMD/SSE/AVX, programmation multi-threads, programmation OpenMP.
- **Depuis 2019, "Programmation GPU"** (code INF-5201A), 3e année cycle ingénieur ("E5", équivalent M2, 40 élèves en moyenne). 18h de cours et 12h de TP. Étude de l'architecture des GPUs puis programmation CUDA, avec mise en place d'un accès distant aux maquettes NVIDIA Jetson-TX2 pendant le confinement dû au COVID.
- Depuis 2013, "Réalité virtuelle" (code OUAP-4317, ex-"Atelier 3D"), 2e année cycle ingénieur ("E4", équivalent M1, 28 élèves en moyenne), Conception des cours et TP ou gestion de vacataires selon les années.

- **Depuis 2001, "Encadrement projets E4"** : chaque année les élèves réalisent des projets par groupe de 4 sur un thème donné par les enseignants ou les partenaires industriels. J'encadre en moyenne 2 projets par an. Les sujets sont très variés, il peut s'agir de développements de systèmes embarqués mêlant robotique et vision (NxP Cup), d'applications médicales (echographie open source), de dispositifs originaux d'affichage 3D (projet Holoslide), d'applications sur smartphone pour musées (musée la Contemporaine à Nanterre), de visite en réalité virtuelle des salles blanches de l'ESIEE.

Passés

- **2002-2016, "Architecture et programmation des DSP"** (code IF4-ARCH), 2e année cycle ingénieur, ("E4", équivalent M1, 36 élèves) : co-responsable avec Mohamed Akil. Architecture Texas Instrument TMS320C6711) mise en place du cours et des TPs.

4.1.3 Filière Électronique (M2)

Passés

- **2007-2011, "Architectures embarquées"** (code AI4-SOC), 2e année cycle ingénieur ("E4", équivalent M1, 24 élèves) : co-responsable Farbod Ghassemi puis Ludovic Noury : mise en place d'un soft-core (NIOSII) dans un FPGA.

4.2 Filières Apprentissages (Alternances) ESIEE

4.2.1 Réseaux et sécurité de Noisy et Cergy

- **2003-2020, "Programmation systèmes distribués"**, 2e année cycle ingénieur, ("E4", équivalent M1, 40 élèves en moyenne) : 6h de cours, 6h de TP. Programmation système des communications par socket.

4.2.2 Informatique et Applications

- **2003-2020, "Architecture et Traitement des images"**, ("E4", équivalent M1, 40 élèves en moyenne) : 15h de cours et 15h de TP sur notre ancien site de Cergy Pontoise.

4.2.3 Electronique embarquée

- **Depuis 2022**, cours "IA pour l'embarqué" ("E5, équivalent M2, 40 élèves). 10h de cours, 8h de TP. Introduction à l'IA, implémentation sur architectures embarquées généralistes et sur GPU.

- **Suivi de projets "fil rouge"**[1], (élève de dernière année - M2) : exemples : développement d'un radar 66Ghz embarqué, conception d'un scanner 3D, portage d'un autopilote pour drone sur architecture STM32MP1, développement d'une carte embarquée de reconnaissance d'oiseaux par IA, etc.

[1] Pendant la dernière année du 3ème cycle

4.3 Université Gustave Eiffel

4.3.1 Master 2 SI (Science de l'Image) de l'UGE

- **Depuis 2008, "Architectures et programmation des microprocesseurs"**, co-responsable du cours avec M. Akil puis E. Dokladalova et R. Kachouri. Environ 15 étudiants, 6h de cours, 4h de TP.

4.3.2 Ingénieur IMAC, Master 2 SI, Master 2 ANCV de l'UGE

- **2016-2021, "Introduction à la réalité virtuelle et augmentée"**, 3e année du cycle ingénieur, 20h de cours. Cours partagé avec les filières IMAC3 (Ecole d'ingénieurs UGE "Informatique Multimédia, Art et Communication"), le Master 2 SI (Sciences de l'Image), ainsi que le Master 2 ANCV (Arts Numériques et Cultures Visuelles), soit un total d'une quarantaine d'étudiants.

4.4 Institut Supérieur de Bio Sciences - Créteil (ISBS)

- **2008-2012, "Systèmes d'exploitation"**, 15h de cours, 10h de TP,
- **2010-2012, "Systèmes embarqués sur DSP"**, 15h de cours, 10h de TP.

4.5 Formation continue

- **2008-2012, "Structure interne des ordinateurs"**, formation continue pour l'ENSG (École Nationale des Sciences Géographique), 8h de cours.

Chapitre 5

Liste des publications

5.1 Journaux internationaux

1. Nejmeddine BAHRI, Thierry GRANDPIERRE, Med Ali Ben AYED, Nouri MASMOUDI et Mohamed AKIL. « Embedded Real-Time H264/AVC High Definition Video Encoder on TI's KeyStone Multicore DSP ». In : *Journal of Signal Processing Systems* 86.1 (2017), p. 67-84. DOI : 10.1007/s11265-015-1098-x. URL : https://hal.archives-ouvertes.fr/hal-01286949 (Impact Factor, IF : 1.8)

2. Oussama FEKI, Thierry GRANDPIERRE, Nouri MASMOUDI et Mohamed AKIL. « Optimized Implementation of H.264/AVC Motion Estimation on a Mixed Architecture Using SynDEx-Mix ». In : *International Review on Computers and Software (IRECOS)* 11.5 (mai 2016). DOI : 10.15866/irecos.v11i5.8764. URL : https://hal.archives-ouvertes.fr/hal-01800584

3. Pavel KARAS, Vincent MORARD, Jan BARTOVSKY, Thierry GRANDPIERRE, Eva DOKLADALOVA, Petr MATULA et Petr DOKLÁDAL. « GPU Implementation of Linear Morphological Openings with Arbitrary Angle ». In : *Journal of Real-Time Image Processing* 10.1 (2015), p. 27-41. DOI : 10.1007/s11554-012-0248-7. URL : https://hal.archives-ouvertes.fr/hal-00680904 (IF : 3)

4. Nejmeddine BAHRI, Nidhameddine BELHADJ, Thierry GRANDPIERRE, Mohamed ALI BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « Real-time H264/AVC encoder based on enhanced frame level parallelism for smart multicore DSP camera ». In : *Journal of Real-Time Image Processing* 12 (nov. 2014), p. 791-812. DOI : 10.1007/s11554-014-0470-6. URL : https://hal.archives-ouvertes.fr/hal-01192770 (IF : 3)

5. Nejmeddine BAHRI, Imen WERDA, Thierry GRANDPIERRE, Mohamed Ali BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « Optimizations for real-time implementation of H264 AVC video encoder on DSP processor ». In : *International Review on Computers and Software (IRECOS)* 8.9 (sept. 2013), p. 2025-2035. URL : https://hal.archives-ouvertes.fr/hal-01192792

6. Mohamed AKIL, Thierry GRANDPIERRE et Laurent PERROTON. « Real Time Dynamic Tone-Mapping Operator on GPU ». In : *Journal of Real-Time Image Processing* 7.3 (2012), p. 165-172. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00826330 (IF : 3)

7. Imen WERDA, Taheni DAMMAK, Thierry GRANDPIERRE, Mohamed Ali BEN AYED et Nouri MASMOUDI. « Real-time H.264/AVC baseline decoder implementation on TMS320C6416 ». In : *Journal of Real-Time Image Processing* 7.4 (déc. 2012), p. 215-232. DOI : 10.1007/s11554-010-0181-6. URL : https://hal.archives-ouvertes.fr/hal-01192810(IF : 3)

8. Baptiste DELPORTE, Laurent PERROTON, Thierry GRANDPIERRE et Jacques TRICHET. « Accelerometer and Magnetometer Based Gyroscope Emulation on Smart Sensor for a Virtual Reality Application ». In : *Sensor and Transducers Journal* 14-1.Special Issue ISSN 1726-5479 (mars 2012), p32-p47. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00826243 (Index Copernicus : 6.13 en 2011)

9. Rachida SAOULI, Mohamed AKIL et Thierry GRANDPIERRE. « Learning System for Defactorization Factor Classification of Factorized Data Dependence Graph ». In : *IJACT : International Journal of Advancements in Computing Technology* 3.4 (2011), p. 1-13. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00826253 (SJR : 0.55 en 2012)

10. Mohamed BOUBAKER, Mohamed AKIL, Khaled BEN KHALIFA, Thierry GRANDPIERRE et Mohamed HEDI BEDOUI. « Implementation of an LVQ neural network with a variable size : algorithmic specification, architectural exploration and optimized implementation on FPGA devices ». In : *Neural Computing and Applications* ISSN 0941-0643 (sept. 2009), 10pp. URL : https://hal.archives-ouvertes.fr/hal-00622461 (IF :6)

11. R. SAOULI, Mohamed AKIL et Thierry GRANDPIERRE. « Load Balancing and Static Placement/Scheduling Heuristic on Distributed Heterogeneous Architecture ». In : *International Review on Computers and Software (IRECOS)* 1 (2007), p. 227-234. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622384

12. Mohamed AKIL, Laurent PERROTON et Thierry GRANDPIERRE. « FPGA-based architecture for hardware compression/decompression of wide format images ». In : *Journal of Real-Time Image Processing* 1.2 (2006), p. 163-170. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00621972 (IF : 3)

13. Linda KAOUANE, Mohamed AKIL, Thierry GRANDPIERRE et Yves SOREL. « A methodology to implement real-time applications on reconfigurable circuits ». In : *Special issue on Engineering of Configurable Systems of the Journal of Supercomputing* 30.1 (2004), p. 283-301. URL : https://hal.archives-ouvertes.fr/hal-00622081 (IF : 3.3)

14. Linda KAOUANE, Mohamed AKIL, Thierry GRANDPIERRE et Yves SOREL. « A methodology to implement real-time applications on reconfigurable circuits ». In : *Journal of Supercomputing* 30.3 (2004). Special Issue Engineering of Configurable Systems, p. 283-301. DOI : 10.1023/B:SUPE.0000045213.82276.8e. URL : https://hal.archives-ouvertes.fr/hal-00826258

15. K. MABASA, Mohamed AKIL, Thierry GRANDPIERRE, B.J. van WYK et M.A. van WYK. « Automatic VHDL Code Generation for Fuzzy Logic Systems ». In : *African Journal of Science and Technology* 1 (2008), 10pp. URL : https://hal.archives-ouvertes.fr/hal-00622125

16. Michel KOENIG, Jean Michel BOUDENNE, P. LEGRIEL, A. BENUZZI, T. GRANDPIERRE, Dimitri BATANI, Simone BOSSI, Sonia NICOLELLA et René BENATTAR. « A computer driven crystal spectrometer with charge coupled device detectors for x-ray spectroscopy of laser plasmas ». In : *Review of Scientific Instruments* 68.6 (mars 1997), p. 2387-2392. DOI : 10.1063/1.1148122. URL : https://hal.science/hal-03948342 (SJR : Q2, Scopus : 3.1, IF :1.6)

5.2 Chapitre de livre

17. Pierre NIANG, Thierry GRANDPIERRE et Mohamed AKIL. « Implementing Real-Time Algorithms by using the AAA Prototyping Methodology ». In : *Embedded System Design, Techniques and Trends*. IFIP. Springer, 2007, p. 27-36. URL : https://hal.archives-ouvertes.fr/hal-00622231

5.3 Brevets

18. 2023 (en cours de dépôt) : Florient Champenois and Abraham Suissa (Safran) and Florian Bradner and Marc Kaufman (Safran) and Thierry Grandpierre, dans le cadre de la thèse CIFRE de Florient Champenois (Cf. page 7.3).

19. 2021 : Vania TACHER, Christopher PICO, Hicham KOBEITER, Thierry GRANDPIERRE et Laetitia SACCENTI. « Interventional radiology medical device for real-time guidance of a medical operating needle in a volume ». 01. 2021, (Cf. page 8.4),

20. 2014 : Mireille JACQUESON, Andre MOGOUTOV et Thierry GRANDPIERRE. « Device for projecting images on e.g. soap bubble in smoke, has optical sensor i.e. video camera, for detecting position of bubble screens, and projector for projecting images on screens, where sensor includes infra-red ray emitter ». 01. 2014

5.4 Conférences internationales avec comité de lecture

21. Elias BARBUDO, Eva DOKLADALOVA et Thierry GRANDPIERRE. « A New Modelling Framework for Coarse-Grained Programmable Architectures ». In : *Compas 2020*. accepted for 2020, presentation delayd to 2021. Lyon, France, juin 2021. URL : https://hal.archives-ouvertes.fr/hal-03108479

22. Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A New Mapping Methodology for Coarse-Grained Programmable Systolic Architectures ». In : *22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES 2019)*. St Goar, Germany, mai 2019. DOI : 10.1145/3323439.3323982. URL : https://hal.archives-ouvertes.fr/hal-02013560

23. Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A Mapping Methodology for Coarse-Grained Pipelined Configurable Architectures ». In : *14th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2019)*. Renesse, Netherlands, juin 2019. URL : https://hal.archives-ouvertes.fr/hal-02104162

24. Tristan FAUTREL, Laurent GEORGE, Frederic FAUBERTEAU et Thierry GRANDPIERRE. « An hypervisor approach for mixed critical real-time UAV applications ». In : *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. Kyoto, France : IEEE, mars 2019, p. 985-991. DOI : 10.1109/PERCOMW.2019.8730705. URL : https://hal.science/hal-03578489

25. Nejmeddine BAHRI, Nidhameddine BELHADJ, Med Ali BEN AYED, Nouri MASMOUDI, Thierry GRANDPIERRE et Mohamed AKIL. « Real-time H264/AVC high definition video encoder on a multicore DSP TMS320C6678 ». In : *2015 International*

Conference on Computer Vision and Image Analysis Applications (ICCVIA). Computer Vision and Image Analysis Applications (ICCVIA), 2015 International Conference on. Sousse, Tunisia : IEEE, jan. 2015. DOI : 10.1109/ICCVIA.2015.7351893. URL : https://hal.archives-ouvertes.fr/hal-01797192

26. Nejmeddine BAHRI, Thierry GRANDPIERRE, Mohamed Ali BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « GOP level parallelism implementation for real-time H264/AVC video encoder on multicore DSP TMS320C6472 ». In : *EDERC 2014*. Sous la dir. d'IEEE. Proceedings of the 6th European Embedded Design Education and Research Conference. IEEE, EURASIP. Milan, Italy, nov. 2014, p. 152-156. DOI : 10.1109/EDERC.2014.6924378. URL : https://hal.archives-ouvertes.fr/hal-01192779

27. Nejmeddine BAHRI, Thierry GRANDPIERRE, Med Ali BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « GOP level parallelism implementation for real-time H264/AVC video encoder on multicore DSP TMS320C6472 ». In : *2014 6th European Embedded Design in Education and Research Conference (EDERC)*. Education and Research Conference (EDERC), 2014 6th European Embedded Design in. Milano, France : IEEE, sept. 2014. DOI : 10.1109/EDERC.2014.6924378. URL : https://hal.archives-ouvertes.fr/hal-01797197

28. Feki OUSSAMA, Thierry GRANDPIERRE, Mohamed AKIL et Nouri MASMOUDI. « Automatic Hardware/Software interface generation for SynDEx-mixte ». In : *ATSIP 2014*. International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). Sousse, Tunisia : IEEE, mars 2014, p. 512-516. DOI : 10.1109/ATSIP.2014.6834668. URL : https://hal.archives-ouvertes.fr/hal-01192824

29. Yareb ELOUMI, Mohamed AKIL, Thierry GRANDPIERRE et Mohamed Hédi BEDOUI. « Latency and power optimization in AAA methodology for integrated circuits ». In : *ICECS 2010*. Sous la dir. d'IEEE. IEEE. Athens, Greece, déc. 2010, p. 639-642. DOI : 10.1109/ICECS.2010.5724593. URL : https://hal.archives-ouvertes.fr/hal-01192801

30. Petr MATAS, Eva DOKLADALOVA, Mohamed AKIL, Thierry GRANDPIERRE, Laurent NAJMAN, M. POUPA et V. GEORGIEV. « Parallel Algorithm for Concurrent Computation of Connected Component Tree ». In : *Advanced Concepts for Intelligent Vision Systems (ACIVS'08)*. T. 5259/2008. Lecture Notes in Computer Science 1. France : Springer-Verlag, oct. 2008, p. 230-241. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622406

31. Pierre NIANG, Thierry GRANDPIERRE, Mohamed AKIL et Yves SOREL. « AAA and SynDEx-Ic : A Methodology and a Software Framework for the Implementation of Real-Time Applications onto Reconfigurable Circuits ». In : *International Embedded Systems Symposium*. 1. France, 2007, 10pp. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622183

32. Mohamed AKIL, Pierre NIANG et Thierry GRANDPIERRE. « A rapid prototyping methodology to implement and optimizing image processing algorithms for FPGAs ». In : *IS&T / SPIE, Symposium on Electronic Imaging Science and Technologies, Real-Time Imaging IX Conference*. 1. electronic version (10 pp.) San Jose, USA, France, 2006, 10pp. URL : https://hal.archives-ouvertes.fr/hal-00622356

33. Thierry GRANDPIERRE et Yves SOREL. « From Algorithm and Architecture Specifications to Automatic Generation of Distributed Real-Time Executives : a Seamless Flow of Graphs Transformations ». In : *2003 1st IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2003)*. Mont Saint Michel, France : IEEE, juin 2003. DOI : 10.1109/MEMCOD.2003.1210097. URL : https://hal.archives-ouvertes.fr/hal-01800622

34. Linda KAOUANE, Mohamed AKIL, Yves SOREL et Thierry GRANDPIERRE. « From Algorithm Graph Specification to Automatic Synthesis of FPGA Circuit : A Seamless Flow of Graphs Transformations ». In : *Field Programmable Logic and Application, 13th International Conference*. Lisbon, Portugal, sept. 2003, p. 934-943. URL : https://hal.archives-ouvertes.fr/hal-01800628

35. T. GRANDPIERRE, G. FLEUTOT, N. PARENT et Mooncheol WON. *A joystick driving algorithm with a collision stop feature on an electric vehicle (Cycab)*. Mars 2003. DOI : 10.1109/IVS.2002.1188000. URL : https://hal.science/hal-03949756

36. Michel BARRETEAU, Juliette MATTIOLI, Thierry GRANDPIERRE, Christophe LAVARENNE, Yves SOREL, Philippe BONNOT et Philippe KAJFASZ. « PROMPT : A mapping environment for telecom applications on System on a Chip ». In : *International conference on Compilers, architecture, and synthesis for embedded systems*. San Jose, United States : ACM Press, nov. 2000, p. 41-47. DOI : 10.1145/354880.354887. URL : https://hal.science/hal-03949779

37. Thierry GRANDPIERRE, Christophe LAVARENNE et Yves SOREL. « Optimized Rapid Prototyping for Real-Time Embedded Heterogeneous Multiprocessors ». In : *the seventh international workshop*. Rome, France : ACM Press, 1999. DOI : 10.1145/301177.301489. URL : https://hal.archives-ouvertes.fr/hal-01800625

5.5 Revue française

38. T. GRANDPIERRE. « Visite virtuelle des salles blanches ESIEE ». In : *Journal sur l'enseignement des sciences et technologies de l'information et des systèmes* 18 (2019), p. 1002. DOI : 10.1051/j3ea/20191002. URL : https://hal.science/hal-03948432

5.6 Conférences et colloques nationaux

39. Elias BARBUDO, Eva DOKLADALOVA et Thierry GRANDPIERRE. « Modelling and Mapping Framework for Coarse-GrainedProgrammable Architectures ». In : *14th Summer School on Modelling and Verification of Parallel Processes*. (online), France, juin 2020. URL : https://hal.archives-ouvertes.fr/hal-03108451

40. Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A mapping tool for configurable pipeline co-processors ». In : *Colloque National du GDR SoC-SiP*. GDR-SOC-SIP. Paris, France, juin 2018. URL : https://hal-enpc.archives-ouvertes.fr/hal-01801018

41. Nejmeddine BAHRI, Thierry GRANDPIERRE, Nouri MASMOUDI et Mohamed AKIL. « Optimisations structurelles et matérielles de l'encodeur vidéo H264/AVC

sur un seul coeur d'un DSP multicoeurs TMS320C6472 ». In : *GRETSI 2013 (Symposium on Signal and Image Processing)*. BREST, France, 2013. URL : https://hal.archives-ouvertes.fr/hal-01797228

42. T. GRANDPIERRE et Y. SOREL. « Un nouveau modèle générique d'architecture hététogène pour la méthodologie AAA ». In : *Actes des Journées Francophones sur l'Adéquation Algorithme Architecture, JFAAA'02*. Monastir, Tunisia, déc. 2002. URL : http://www-rocq.inria.fr/syndex/publications/pubs/jfaaa02/jfaaa02.pdf

43. Christine ANCOURT, Michel BARRETEAU, Bernard DION, T. GRANDPIERRE, Irigoin FRANÇOIS, Jean JOURDAN, Philippe KAJFASZ, C. LAVARENNE et Y. SOREL. « PROMPT : Placement rapide optimisé sur machines parallèles pour applications de télécommunications ». In : *Actes du 5ème Worshop sur l'Adéquation Algorithme Architecture*. Paris, France, jan. 2000. URL : http://www-rocq.inria.fr/syndex/publications/pubs/aaa100/aaa100.pdf

5.7 Séminaires scientifiques

44. T. Grandpierre, "Traitement d'images en temps réel", Erasmus+, F'SATI (French South African Institute of Technology), Prétoriat, Afrique du Sud, 2019

45. T. Grandpierre, "Compréssion vidéo haute définition sur architecture multi-composants", projet Campus France - PHC Utique, Université de Sfax, ENIS, Tunisie, 2013

46. T. Grandpierre, "Architecture interne du processeur DSP multi-coeurs TMS320DM6437 et programmation", Formation organisée par le LETI de Sfax, Sousse, 2010,

47. T. Grandpierre, E. Dokladalova et Mohamed Akil. "Carte à puce, architectures et protocoles". Séminaire d'école d'été, INPT, Rabat, Maroc 2007

48. T. Grandpierre, "Méthodologie Adéquation Algorithme Architecture", projet Campus France CAPES-COFECUB, Universidade Federal De Minas Gerais, Belo Horizonte, Brésil, 2002

5.8 Rapports et manuscrit

49. Livrables du projet FUI CEOS : D2.4, D3.2b, D5.3 et rapport de fin de programme BPI pour l'ESIEE,

50. Livrables WP4.2 du projet ISITE Urbanvision,

51. Rapport de recherche INRIA [43],

52. Grandpierre THIERRY. « Modélisation d'architectures parallèles hétérogènes pour la génération automatique d'exécutifs distribuées temps réel optimisés ». Theses. Université Paris Sud Orsay, nov. 2000. URL : https://theses.hal.science/tel-03545962

5.9 En préparation

- Article de conférence sur la génération de configurations réseaux avioniques (Cf. 7.3), soumis à la conférence "32nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2024)",
- Article de journal sur l'implémentation de la BRDF sur GPU dans Unity (Cf. 8.3.3), en finalisation,
- Article de journal sur la modélisation d'architecture (Cf. 6.14), en finalisation,
- Article de conférence sur un système de réalité augmentée breveté (Cf. 8.4), en finalisation.

Deuxième partie

Présentation des travaux de recherche

Préambule

Contexte et objectifs

Les travaux présentés dans ce mémoire couvrent principalement des applications du domaine temps réel, que ce soit le temps réel strict (hard real time) ou le temps réel souple (soft real time). Dans le cadre du temps réel strict, le non respect des contraintes de l'application peut conduire à une situation critique, voire catastrophique (perte humaine ou financière) : on ne peut se le permettre, il faut concevoir l'application pour que cette situation n'arrive pas. Dans le cas du temps réel souple, le non respect des contraintes ne conduit pas à une catastrophe mais généralement à une baisse de qualité ou par exemple un inconfort (par exemple visuel) temporaire, on souhaite donc principalement minimiser ces situations.

Ces applications réalisent de nombreuses transformations. Ainsi pour commencer, elles doivent transformer des grandeurs physiques issues de capteurs (températures, positions, vitesses, etc.) en données numériques. Puis ces données sont transformées par des algorithmes afin de produire de nouvelles données, généralement à destination d'actionneurs (moteurs, servo-moteurs, écrans, haut parleur etc.).

Pour réaliser ces transformations, il existe une très grande variété de composants électroniques. Par exemple les convertisseurs analogiques numériques (CAN, ou en anglais ADC pour Analog to Digital Converter) permettent de transformer des tensions analogiques issues des capteurs (température, luminosité, champs magnétique, etc.) en valeurs numériques. Certains capteurs intègrent ces convertisseurs, d'autres fournissent directement des valeurs numériques. Symétriquement, les convertisseurs numériques analogiques (CNA, ou en anglais DAC pour Digital to Analog Converter) transforment des données numériques en tensions exploitables par des actionneurs.

Entre capteurs et actionneurs, les algorithmes transforment les données numériques d'entrée en les combinant entre elles de façon plus ou moins complexe pour produire de nouvelles données de sortie. Ces algorithmes peuvent se décrire par des enchaînements d'opérations élémentaires (opérations arithmétiques et logiques) qui accèdent en lecture/écriture à des mémoires, et sont associés à des tests (comparaisons) dont les résultats conditionnent l'exécution de sous-ensembles d'opérations.

Il existe deux grands types d'implémentation de ces algorithmes : les implémentions logicielles et les implémentions matérielles qui sont toutes deux abordées dans ce manuscrit.

Les implémentions logicielles sont les plus connues. Dans ce cas les opérations des algorithmes sont transformées en instructions de calcul, de contrôle et d'accès à la mémoire. L'ensemble forme alors un programme qui est exécuté par un ou plusieurs processeurs. Comme nous le verrons en détail plus loin, chaque processeur repose au minimum sur un séquenceur d'instructions, une unité arithmétique et logique et de la mémoire. Il existe une très grande diversité de processeurs (GPP scalaires monocoeur et multi-coeurs superscalaires ou non, processeurs SIMD, GPU, DSP, MCU) que nous présenterons également dans les chapitres suivants.

Les implémentations matérielles sont plus complexes et donc moins connues. Les opérations des algorithmes sont implantées à l'aide de portes logiques élémentaires (portes ET, OR, etc), de bascules, et de mémoires, câblées ensemble de façon définitive dans les ASIC (Application Specific Integrated Circuit) ou de façon reconfigurable dans les FPGA (Field Programmable Gate Array). Les ASICS sont très coûteux à produire et sont donc réservés aux productions en très grandes séries alors que les FPGA sont bien adaptés au prototypage et aux petites séries.

Chacune des deux implémentations, logicielles ou matérielles, comporte des avantages et des inconvénients : du côté des implémentations logicielles nous pouvons par exemple citer leur grande flexibilité et leur coût plus faible. De plus le système peut aisément être amélioré, corrigé ou adapté pour supporter par exemple de nouveaux standards. Les architectures matérielles offrent généralement de meilleures performances en terme de consommation et de vitesse d'exécution, mais au détriment de temps de développement bien plus élevés.

Cependant, comme nous allons le découvrir dans les chapitres suivants, la frontière entre le monde de l'implémentation matérielle et celui de l'implémentation logicielle n'est pas si stricte. En effet, pour obtenir le meilleur de chacun des mondes il est possible de construire des architectures mixtes qui comprennent à la fois des composants programmables (processeurs), et des composants câblés (ASIC) ou reconfigurables (FPGA). Il faut alors découper judicieusement les algorithmes pour utiliser au mieux les ressources matérielles sous-jacentes.

Ces architectures peuvent être obtenues en connectant les cartes électroniques qui les supportent, soit en les plaçant sur une même carte électronique ou un même module ("System on Module" - SoM - en anglais), soit comme on le voit de plus en plus, en associant ces différents composants dans un même composant. Ce dernier est alors appelé "Système sur Puce" ("System On a Chip" - SoC - en anglais).

Verrous et objectifs des travaux

Il existe donc une très grande diversité d'architectures matérielles susceptibles d'exécuter les algorithmes d'une application temps réel, qu'elle soit critique ou non.

Le choix de l'architecture dépend essentiellement des contraintes de l'application : contraintes temps réel ou non, embarquabilité (donc taille / nombre de composants), consommation et bien sûr coût (coût de l'architecture bien sûr, mais aussi coût de développement).

Lors de la conception de ces applications, on peut identifier trois grandes phases. Après avoir défini les contraintes de l'application, le concepteur doit tout d'abord choisir ou, le cas échéant, concevoir l'architecture qui répond à ses contraintes applicatives. Ensuite, il faut utiliser au mieux l'architecture cible, c'est à dire utiliser efficacement tous ses éléments. Enfin, il s'agit d'évaluer les performances pour les comparer aux contraintes initiales.

Nous verrons qu'il existe peu d'outils qui couvrent tous ces aspects dans un même paradigme. L'objectif des travaux présentés dans ce manuscrit est de proposer un cadre méthodologique complet et de le valider par l'implantation de différentes applications sur un large panel d'architectures matérielles.

Nous allons donc aborder la conception d'architectures dédiées, leur modélisation, leur caractérisation, l'exploration d'architectures, l'optimisation de l'implantation d'algorithmes sur ces architectures. Il s'agit alors de problèmes de distribution, ordonnancement d'algorithmes sur les architectures en utilisant par exemple des heuristiques d'optimisation.

Organisation du manuscrit

La suite du manuscrit est donc organisée en 4 chapitres. Le premier chapitre présente mes contributions en méthodologie de conception d'application temps réel sur diverses architectures telles-que les architectures reconfigurables à base de FPGA, puis d'architectures mixtes programmables et reconfigurables. Enfin je présenterai mes plus récents travaux sur les architectures reconfigurables.

Le second chapitre est dédié aux apports de mes travaux dans le domaine spécifique des applications temps réel critique. Après une partie théorique, nous y aborderons un cas réel d'utilisation dans le cadre d'un projet industriel.

Le troisième chapitre présente mes travaux dans le domaine des applications temps réel souple. Ces applications sont assez variées puisqu'elles iront de la compression vidéo à la réalité virtuelle.

Le dernier chapitre présente une synthèse de ce manuscrit et les perspectives envisagées à court et moyen termes.

Chapitre 6

Méthodologie et outils de conception d'applications temps réels - Modélisation d'architectures

Les travaux présentés dans ce chapitre sont en continuité de ceux débutés pendant ma thèse. La première section expose donc brièvement le sujet et les apports de cette dernière. Puis, au fil des trois sections suivantes, je résume les travaux d'extension que j'ai pu mener en co-encadrant trois thèses successives.

6.1 Méthodologie Adéquation Algorihme Architecture (AAA)

Durant ma thèse [43, 46, 45] au sein du projet SOSSO de l'INRIA Rocquencourt, j'ai participé au développement de la méthodologie Adéquation Algorithme Architecture (AAA). Elle a pour but d'aider au prototypage rapide d'applications temps réel de contrôle-commande et de traitement du signal et des images sur des architectures dédiées distribuées hétérogènes. AAA prend en compte toutes les étapes du développement d'une application temps réel embarquée, de sa spécification haut niveau jusqu'à l'exécution du code dans les composants. Cette méthodologie, que j'ai implanté dans la version 5 du logiciel SynDEx, a pour but d'aider les concepteurs à développer rapidement des applications temps réel distribuées et optimisées (i.e. qui respectent les contraintes temps réel et minimisent la taille des architectures) sur des architectures multi-processeurs programmables communiquant par passages de messages ou par mémoires partagées.

AAA est basée sur un formalisme d'hypergraphes autant pour spécifier l'algorithme (graphe flot de données conditionné factorisé mettant en évidence le parallélisme potentiel) et l'architecture matérielle (graphe orienté mettant en évidence le parallélisme disponible) que pour décrire l'implantation. L'adéquation consiste à choisir l'implantation qui respecte les contraintes temps réel et minimise les ressources matérielles utilisées. Une implantation optimisée est obtenue par transformation de graphes, elle correspond à une distribution et un ordonnancement hors-lignes des éléments de l'algorithme sur l'architecture. La construction de l'implémentation est effectuée par une heuristique gloutonne qui se base sur une caractérisation préalable de chaque élément des composants de l'algorithme et de l'architecture. Le graphe d'implantation ainsi obtenu est ensuite transformé en un exécutif temps réel distribué et optimisé puisque taillé sur mesure à l'application.

L'implantation temps réel de ces applications est donc une tâche complexe puisque de nombreux problèmes doivent être résolus :

- Cibler une architectures distribuée implique de distribuer et ordonnancer chaque partie de l'algorithme sur chaque composant de l'architecture. Cette distribution amène à ajouter des opérations de communication dont la durée est rarement négligeable.

- Il est fondamental de garantir les performances temps réel quelles que soient les conditions d'exécution (charge des processeurs, des média de communication). Dans ce type d'architecture, nous avons montré [45] que les communications étaient souvent négligées bien que pouvant compromettre sérieusement les performances de ces applications. De même le problème de tolérance aux pannes ne doit pas être négligé pour ces systèmes critiques.

- La nature embarquée de ces applications rend nécessaire d'optimiser l'utilisation de chaque ressource de l'architecture matérielle pour en minimiser la taille.

- Il existe un problème de génération de code délicat à traiter dans le cas d'architectures intrinsèquement hétérogènes puisque chaque composant nécessite très souvent des langages de programmation et des protocoles de communications différents. De plus, il est fondamental de veiller à ce que le code implanté reflète exactement la distribution et l'ordonnancement choisi, sans quoi les performances ne peuvent être garanties.

- Le coût du déverminage des applications distribuées représente une part importante dans le cycle de développement étant donné la complexité engendrée par ce type d'architecture. Des transformations automatiques permettent de minimiser cette partie.

- Enfin il faut prendre en compte les critères économiques tel que le délai de mise sur le marché et dans ce but essayer de ne pas rompre le flot de conception de la spécification à l'industrialisation.

Durant ma thèse j'ai proposé [42], [46] et [73] un modèle d'architecture original, basé sur les graphes orientés, afin de couvrir la plus grande variété possible de machines programmables communiquant par mémoires partagées ou par passages de messages (lien de communications, bus, réseaux). Le niveau de modélisation choisi est suffisamment fin pour décrire précisément le coût des communications (au contraire de la plupart des modèles existants), mais pas trop fin pour encapsuler la complexité engendrée par les détails architecturaux. Chaque sommet correspond à une machine à états finis qui modélise les séquenceurs d'opérations de calcul (opérateurs), les séquenceurs de communications (communicateurs), les mémoires (privées, partagées), les arbitres et les bus. Ceci permet de modéliser des architectures composées de processeurs de différents types, connectés également par des médias de différents types (mémoires partagées, communications par passages de messages sur bus point-à-point, multi-points supportant ou non la diffusion).

A titre d'exemple, la figure 6.2 présente la modélisation du processeur DSP TMS 320C40 de Texas Instruments dont l'architecture interne est présentée par la figure 6.1. Ce processeur de traitement du signal à la particularité de pouvoir être connecté directement à 6 autres processeurs DSP de même type grâce à des liens de communications rapides modélisés dans ce graphe par les sommets $S1$ à $S6$, chaque canal DMA étant modélisé par les sommets communicateurs $C1$ à $C6$.

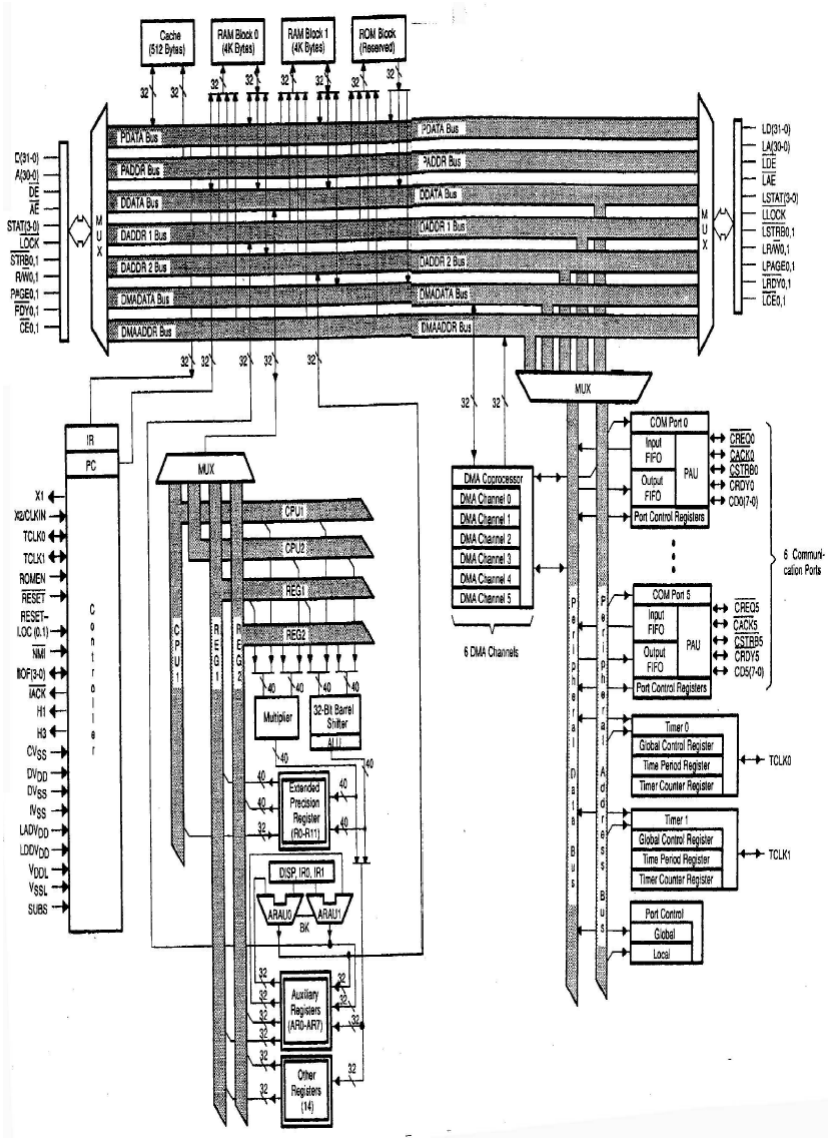


FIGURE 6.1 : Architecture interne du processeur DSP TMS320C40

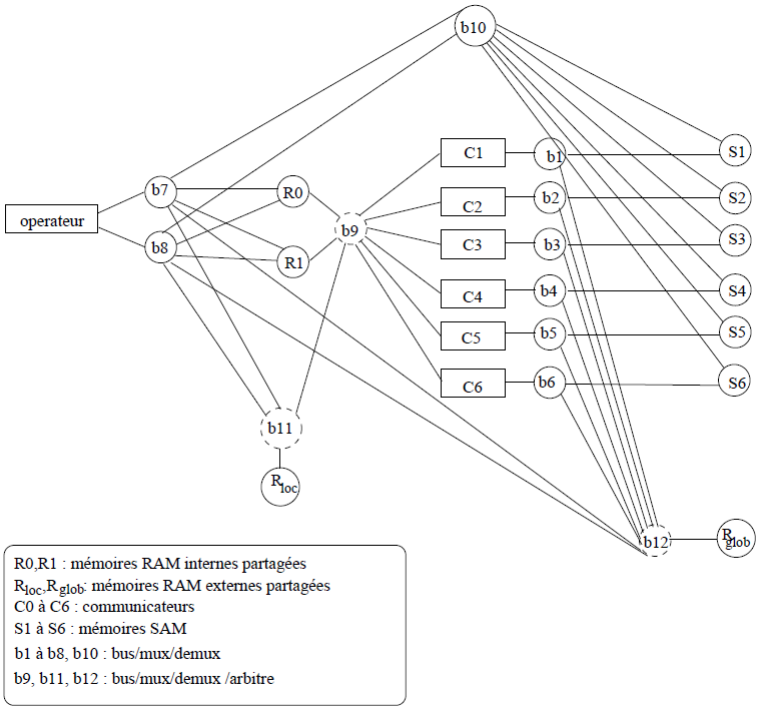


FIGURE 6.2 : Modélisation du processeur DSP TMS320C40

Le modèle d'algorithme est un graphe de données orienté, factorisé et conditionné. Il comprend des sommets capteurs et actionneurs pour réaliser les opérations d'interface avec le monde extérieur, des opérations de transformation de données (opérations de calcul) et des opérations de mémorisation. Il comprend également des sommets "répétitions" qui permettent de spécifier des répétitions de sous-graphes (boucle for) et des sommets de conditionnement pour spécifier les tests. J'ai enrichi ce modèle de graphes avec des sommets et des hyper-arcs spécifiques de façon à pouvoir modéliser l'allocation des différents sommets de l'architecture, en particulier les sommets mémoire.

J'ai ensuite formalisé toutes les transformations de graphes qui permettent de construire un graphe d'implantation par la composition de trois relations appliquées à un couple (graphe d'algorithme, graphe d'architecture). Grâce à ce modèle, il est possible de construire le diagramme temporel d'allocation de la mémoire ce qui permet des optimisations de l'utilisation de chaque mémoire.

A partir d'une application composée d'un graphe d'algorithme et d'un graphe d'architecture il est possible de construire un très grand nombre de graphes d'implantation. Parmi ceux-ci, nous recherchons ceux qui permettent de minimiser l'utilisation des ressources (sommets du graphe de l'architecture) et dont la durée d'exécution (chemin critique dans le graphe d'implantation étiqueté par les durées d'exécution des calculs et des communications) respecte les contraintes temps réel de l'application. Ce problème étant reconnu NP-difficile, nous utilisons une heuristique pour construire le graphe d'implantation optimisé. Parmi les différentes classes d'heuristiques existantes nous avons adopté l'approche gloutonne qui a l'avantage de donner une solution de bonne qualité sans avoir un coût d'exécution rédhibitoire vis-à-vis du prototypage rapide visé par AAA.

Dans une première étape nous avons développé un modèle de caractérisation permettant de prédire les performances d'exécution d'une implantation en tenant compte, entre autres, du rôle des arbitres (partage de la bande passante) et des

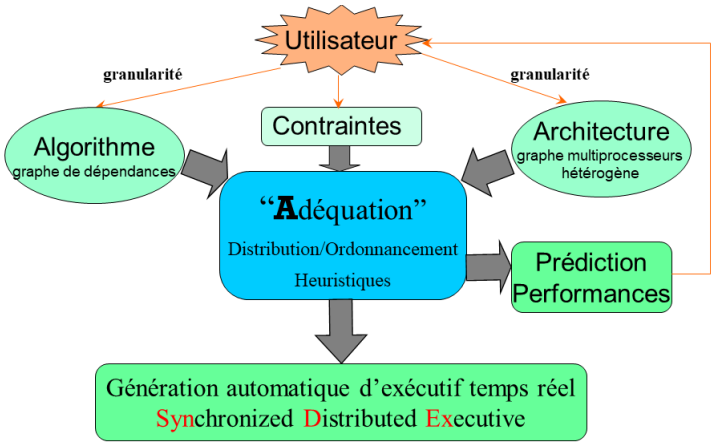


FIGURE 6.3 : Méthodologie AAA pour les architectures programmables

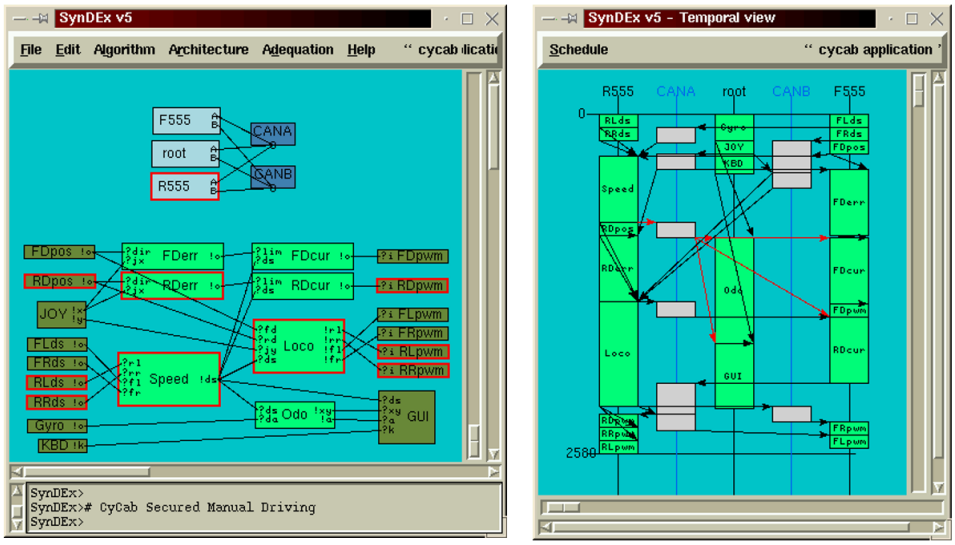


FIGURE 6.4 : Capture écran du logiciel SynDEx implémentant la méthodologie AAA

bandes passantes d'accès aux mémoires de notre nouveau modèle d'architecture. Ensuite nous avons développé une heuristique gloutonne dont la fonction de coût (*pression d'ordonnancement*, notée $sigma(o, j)$ pour une opération o exécutée sur un opérateur j) tient compte à la fois des durées d'exécution des opérations (pour calculer des dates de début et de fin au plus tôt et au plus tard), de la durée des communications qui dépend du type des données et des média de communication empruntés, et enfin des mémoires (programmes, données locales et données communiquées). Cette heuristique est basée sur un algorithme de liste. Lors de chaque itération on distribue et ordonnance une opération (choisie parmi une liste d'opérations dont tous les prédécesseurs sont placés) sur l'opérateur qui minimise la fonction de coût. Pour optimiser davantage l'utilisation de la mémoire des architectures embarquées nous avons enrichi le modèle d'implantation de façon à permettre la "réallocation statique" de chaque mémoire.

La figure 6.3 récapitule l'ensemble des étapes de la méthodologie que j'ai implantée dans le logiciel SynDEx dont des captures écrans sont données figure 6.4.

6.2 Architectures mono-FPGA

- *Thèse de Linda Kaouane : Formalisation et optimisation d'applications s'exécutant sur architecture reconfigurable [61]*
- *Direction : Mohamed Akil (Pr. Université Gustave Eiffel/UGE)*
- *Co-encadrement : Thierry Grandpierre (40%)*
- *Soutenue le 17/12/2004 - Durée : 40 mois*
- *Financement MESR*
- *École Doctorale : MSTIC - Mathématiques et Sciences et Technologies de l'Information et de la Communication n°532*
- *Publications : [54, 55, 57, 50, 51]*

6.2.1 Contexte et objectifs

Comme brièvement présenté en chapitre II, les architectures à base de FPGA présentent de grands atouts pour les applications temps réel étant donné le parallélisme massif qu'elles offrent. Ils permettent de réduire la vitesse d'exécution des algorithmes qui y sont câblés et offrent une faible consommation relative. En revanche, le développement est généralement beaucoup plus long et complexe que pour les algorithmes programmés sur processeurs. En effet, l'espace de solution est immense, il faut gérer le matériel à un niveau très bas (en particulier gérer les synchronisations) et le nombre de spécialistes est très faible en proportion du monde de la programmation des processeurs. Cette section présente donc le travail d'extension de la méthodologie AAA aux architectures reconfigurables qui doit permettre d'utiliser plus facilement ce type d'architecture en conservant le même cadre méthodologique AAA.

Nous commencerons par une brève présentation des caractéristiques des FPGA puis nous présenterons les résultats obtenus pour ce type d'architecture.

6.2.2 Architectures FPGA

La plupart des FPGA sont constitués au minimum :

- d'un ensemble de blocs logiques configurables (appelés par exemple Logic Cells, Logic Element, ou Configurable Logic Blocks selon les constructeurs) chacun capable de réaliser des fonctions combinatoires plus ou moins complexes (selon la technologie du FPGA : des portes logiques de base à plusieurs entrées, des multiplexeurs, des Look Up tables), et des fonctions séquentielles (grâce à des bascules et des mémoires),
- d'un ensemble de blocs arithmétiques câblés pour le traitement du signal (DSP slice, DSP Cells),
- d'un ensemble de blocs mémoire configurables,
- d'un ensemble de blocs d'entrées-sorties configurables pour permettre les communications depuis et vers l'extérieur du FPGA,
- d'un réseau configurable permettant d'interconnecter librement entre eux les différents blocs précédents.

La configuration de l'ensemble de ces éléments s'effectue par des langages de description du matériel comme VHDL ou Verilog. Ces langages permettent de décrire le circuit électronique soit directement par assemblage de blocs et sous-blocs (description dite hiérarchique) soit par description du comportement souhaité, soit bien sûr un mélange des deux. Cette description alimente ensuite un logiciel conçu par le constructeur du FPGA cible. Ce logiciel va produire (synthétiser) la configuration de l'ensemble des composants et du réseau d'interconnexion sous la forme d'un fichier de configuration à télécharger ensuite dans le FPGA.

Selon la famille FPGA le nombre de blocs configurables peut aller de quelques milliers à plusieurs centaines de milliers. Un FPGA offre donc potentiellement un parallélisme de calcul massif qu'il s'agit donc d'exploiter efficacement.

Notons qu'il est bien sûr possible d'implanter un processeur dans un FPGA puisqu'un processeur n'est qu'un assemblage de portes, de bascules, de multiplexeurs et de mémoires. Ce type de processeur est appelé "soft-core" (par opposition au processeurs classiques dits "hard-core"), mais ce n'est pas nécessairement l'implémentation la plus avantageuse d'un algorithme dans un FPGA puisque l'exécution des opérations de calcul de l'algorithme risque alors d'être séquencée sans tirer partie de tout le parallélisme disponible. L'algorithme qui est implanté dans un FPGA prend donc rarement la forme d'un processeur et d'un programme. Pour éviter toute ambiguïté et confusion, on emploiera par la suite le terme "configuration" pour le résultat de l'implantation d'un algorithme sur un FPGA (composant reconfigurable) et de "programme" pour l'implantation d'un algorithme sur un processeur (composant programmable). De même le verbe "configurer" sera utilisé pour l'action de configuration d'un composant reconfigurable et le verbe "programmer" sera uniquement utilisé pour l'action de programmation d'un composant programmable[1].

En résumé, implémenter un algorithme de calcul intensif sous la forme d'un processeur dans un FPGA est rarement la solution la plus intéressante puisqu'elle ne permet pas de tirer profit du parallélisme massif qu'il offre. Cependant, un algorithme peut comporter des parties plus adaptées à une implantation sur processeur que sur FPGA, nous verrons donc dans la prochaine section que nous avons le choix entre implanter un processeur dans une partie du FPGA (soft-core), soit d'utiliser de véritables coeurs de processeurs (hard-core) intégrés au FPGA par certains constructeurs, soit encore de construire des cartes électroniques avec côte à côte un processeur et un FPGA. Bien sûr les performances en terme de vitesse et de consommation d'un processeur hard-core sont meilleures que celles d'un processeur soft-core puisque la finesse de gravure du silicium est bien moindre, comme le nombre de composants et de pistes electriques à traverser.

6.2.3 Approche proposée

L'extension de la méthodologie AAA que nous présentons (figure 6.5) permet de générer des circuits synthétisables optimisés à partir de la même spécification algorithmique que celle utilisée dans AAA : un graphe flots données conditionné factorisé présenté en détail dans la section suivante (6.2.4). Ainsi nous donnons les transformations de graphes qui permettent d'une part de synthétiser le contrôle d'une spécification flot de données et les règles qui permettent de générer le VHDL synthétisable et téléchargeable dans le FPGA.

Nous avons aussi développé des heuristiques qui permettent de rechercher automatiquement une solution qui permette d'implanter l'algorithme sous des contraintes

[1] Parfois le verbe programmer est utilisé pour la configuration d'un FPGA ce qui engendre une certain confusion.

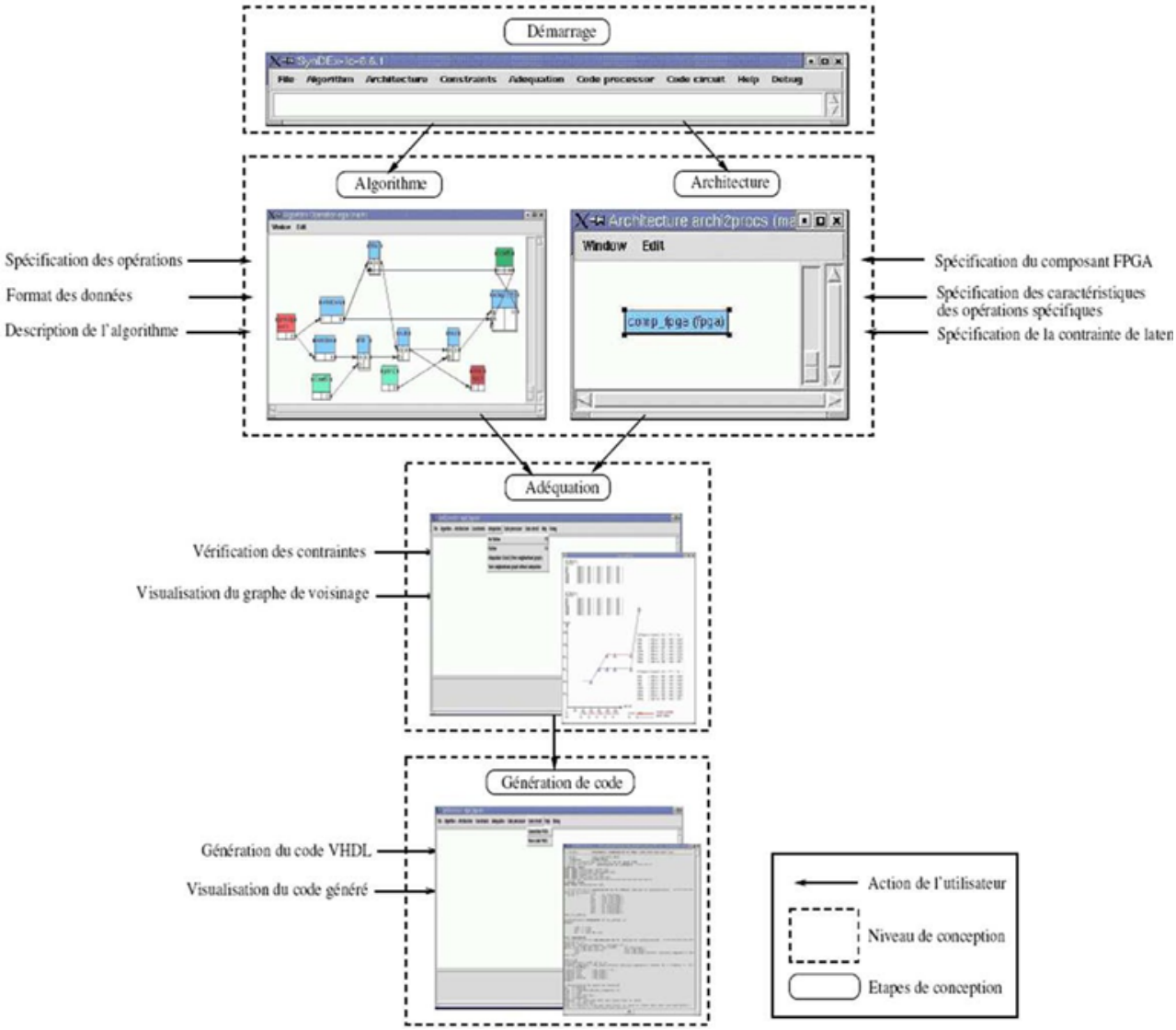


FIGURE 6.5 : Méthodologie proposée et capture écran de l'outil SynDEx-IC

de latences et de surfaces. Nous avons implanté ces résultats dans le logiciel "SynDEx-Ic"[2] que nous avons développé à l'ESIEE à partir du noyau SynDEx de l'INRIA. Des captures d'écrans sont données figure 6.5.

6.2.4 Transformations du graphe d'algorithme

Comme nous l'avons vu la spécification de l'algorithmique de l'application est faite sous la forme d'un Graphe Factorisé Dépendances de Données (GFDD), elle nécessite la mise en œuvre d'un processus de décomposition de l'algorithme en opérations implantables devant être indépendantes de toute contrainte liée à l'implantation matérielle.

En effet, la spécification sous la forme d'un Graphe de Dépendances de Données (GDD) comprend des parties régulières (motifs répétitifs périodiques, c'est-à-dire

[2]http ://www.esiee.fr/a2si//web-ca/syndex-ic/index.htm

des opérations identiques mais opérant sur des données différentes) et non régulières. Pour réduire la taille de la spécification et mettre en évidence ces parties régulières, on applique un processus de factorisation basé sur des sommets spéciaux (sommets frontières de factorisation) : un sommet F (comme Fork) réalise la partition d'un tableau en autant d'éléments que de répétitions dans le motif, un sommet J (comme Join) compose un tableau à partir des résultats de chaque répétition de motif, D diffuse la donnée à toutes les répétitions du motif et I réalise la dépendance de donnée inter-itération du motif. Une opération, située du côté d'une frontière qui présente un groupe d'opérations identiques opérant sur un groupe factorisé de données différentes, sera réalisée au moyen d'un seul opérateur (ou de plusieurs opérateurs en parallèle) utilisé itérativement autant de fois qu'il existe d'opérations dans le groupe factorisé. La factorisation réduit la taille de la spécification (la sémantique opératoire n'est pas modifiée) et permet de décrire en intention plusieurs implantations, chacune avec des caractéristiques (surface, temps de réponse) différentes.

A titre d'exemple la figure 6.6 présente le graphe d'algorithme du produit terme à terme de 2 vecteurs générés par les sommets capteurs A et B. Le vecteur résultant est envoyé au capteur D.

6.2.5 Synthèse de circuit

Nous avons ainsi développé des règles permettant de synthétiser les chemins de données et de contrôle du circuit correspondant à l'algorithme spécifié à l'aide d'un modèle de graphe de dépendances factorisé. Nous avons montré qu'il est possible de synthétiser de manière simple et systématique le chemin de contrôle à l'aide d'une technique de synchronisation de transferts de données entre registres. Cette approche a permis de réaliser un générateur automatique de VHDL structurel synthétisable, réalisant ainsi la synthèse automatique de circuits. Cette dernière, présentée dans [29], consiste à construire automatiquement une structure physique à partir d'une représentation comportementale de plus haut niveau.

Cependant, pour chaque partie factorisée du graphe, il existe plusieurs implémentations possibles dans un FPGA (un exemple complet est donné ensuite) :

1. soit une implémentation totalement séquentielle : on implémente chaque sous-graphe (identifié par les sommet F, J et I) par une "boucle" matérielle (les détails sont donnés dans l'exemple ci-dessous). Cette implémentation a l'avantage d'utiliser un nombre minimal de cellules du FPGA, mais induit un temps maximal d'exécution puisqu'elle n'exploite aucun parallélisme de calcul,

2. soit une implémentation totalement parallèle : on réplique le sous-graphe autant de fois qu'il est répété : cela conduit au temps d'exécution le plus court possible, mais avec l'inconvénient d'utiliser le plus grand nombre de cellules du FPGA,

3. soit un mélange conjuguant implémentations partiellement parallèles et implémentations partiellement séquentielles. Ce type d'implantation sera étudié plus loin dans la partie optimisation.

Exemple

La figure 6.6 présente un graphe d'algorithme factorisé : les opérations A et B produisent chacun des vecteurs de 4 données. Ces données sont traitées 2 à 2 par l'opération C répétée 4 fois. Cette opération produit à chaque fois une donnée en sortie.

L'ensemble des 4 données produites seront regroupées en un vecteur de 4 éléments pour alimenter l'opération D.

Pour ce graphe d'algorithme, il existe :

1. une implantation totalement factorisée en construisant l'équivalent de boucles "for" dépeint figure 6.7 : les 4 instances C_1, C_2, C_3, C_4 de l'opération C sont exécutées successivement. Cela conduit à un temps d'exécution maximale mais à l'utilisation minimale des ressources (et donc de la surface) du FPGA.

2. une implantation totalement défactorisée où toutes les instances de la boucle sont exécutées en parallèle, figure 6.8 : les 4 instances C_1, C_2, C_3, C_4 de l'opération C sont exécutées en parallèle. Cela conduit au temps d'exécution minimal, mais à l'utilisation maximale des ressources du FPGA.

3. un mélange de plusieurs implémentations partiellement factorisées ; comme par exemple celle de la figure 6.9 où nous avons 2 boucles parallèles de 2 itérations chacune.

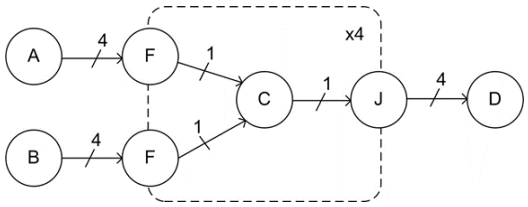


FIGURE 6.6 : Graphe factorisé d'un produit (sommet C de vecteurs générés par les capteurs A et B pour alimenter l'actionneur D). Les nombres sur les arcs représentent la taille du vecteur

L'implantation matérielle des opérations consiste à faire correspondre un opérateur (un ensemble de cellules configurées) à chaque sommet du graphe factorisé d'opérations. Cet opérateur est combinatoire pour un sommet opération, ou est composé d'un multiplexeur et/ou de registres pour un sommet frontière que l'on souhaite implémenter sous forme d'une boucle. L'implantation matérielle des dépendances de données entre opérations consiste à faire correspondre à chaque arc du graphe une connexion physique entre les opérateurs correspondant aux opérations. Les opérateurs et leurs interconnexions constituent le chemin de données du circuit. Chaque frontière de factorisation a des relations amont et aval des deux côtés, lent et rapide. Les relations entre les signaux de requête et d'acquittement amont et aval des deux côtés, constituent l'unité de contrôle de la frontière de factorisation. Ces relations sont déduites des relations de voisinage d'un graphe algorithmique et permettent de construire le graphe de voisinage de frontière (GFD) dont un exemple est donné figure 6.10. Chaque frontière y est modélisée par une rectangle connecté

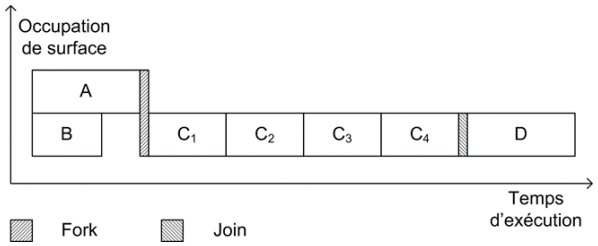


FIGURE 6.7 : Implantation totalement séquentielle

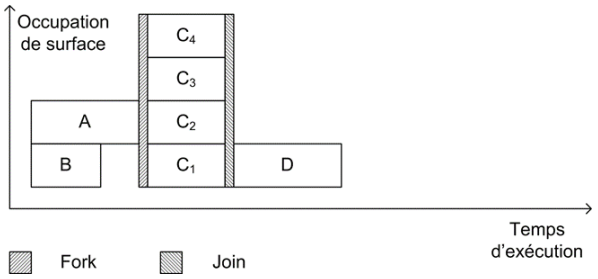


FIGURE 6.8 : Implantation totalement parallèle

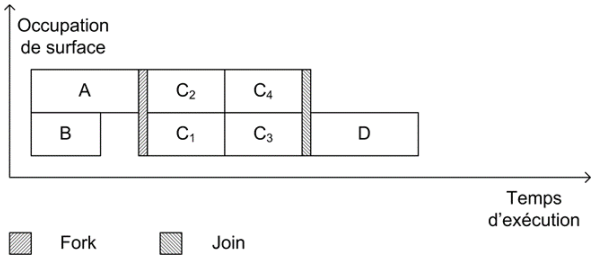


FIGURE 6.9 : Implantation partiellement séquentielle

par des arcs bleus, rouges et noirs aux frontières amont et aval. Chaque rectangle est aussi connecté par des arcs verts aux sommets répétés (ovales) qui appartiennent à cette frontière. L'unité de contrôle de chaque frontière est composée d'un compteur à états et d'une logique supplémentaire afin de générer le protocole de communication entre frontières (signaux de requête et d'acquittement lent et rapide, en amont et en aval). La logique de contrôle ainsi délocalisée permet aux outils de CAO utilisés pour la synthèse de placer les unités de contrôle plus près des opérateurs à contrôler, ce que ne permet pas une logique de contrôle centralisée.

6.2.6 Optimisation

Nous avons vu les règles qui permettent de transformer un graphe d'algorithme en un circuit équivalent dans lequel les parties factorisées peuvent être implantées soit sous forme totalement parallèle, soit sous forme séquentielle ou un mélange des deux. Il faut donc explorer un immense espace de solution pour trouver l'implémentation qui répond aux contraintes de l'application : un temps maximal d'exécution, et un nombre maximal de cellules reconfigurables (appelé surface du FPGA).

Cette phase d'exploration, que nous appelons également optimisation, est fondée sur une heuristique de défactorisation/factorisation cherchant à respecter la contrainte temps réel tout en minimisant la surface. Nous avons proposé plusieurs heuristiques qui déterminent les facteurs de défactorisation optimaux de chacune des frontières composant le chemin critique. Pour cela le GFDD est étiqueté par des éléments tenant compte des relations de production et de consommation entre frontières de factorisation décrites par le graphe de voisinage des frontières de factorisation (GVF). Ensuite pour comparer chaque impléméntation, nous avons proposé une fonction de coût : $f = \frac{\Delta S}{l - max(l', C_t)}$ où $\Delta S = S' - S$ est la variation de surface, S et l représente respectivement la surface et la latence avant défactorisation, S' et l' représentent la surface et la latence après défactorisation et enfin C_t représente la contrainte temporelle. Cette fonction permet de choisir une frontière et son facteur de défactorisation associé. Deux grandes classes d'heuristiques ont été étudiées : les heuristiques gloutonnes qui ont l'avantage de fournir rapidement une solution,

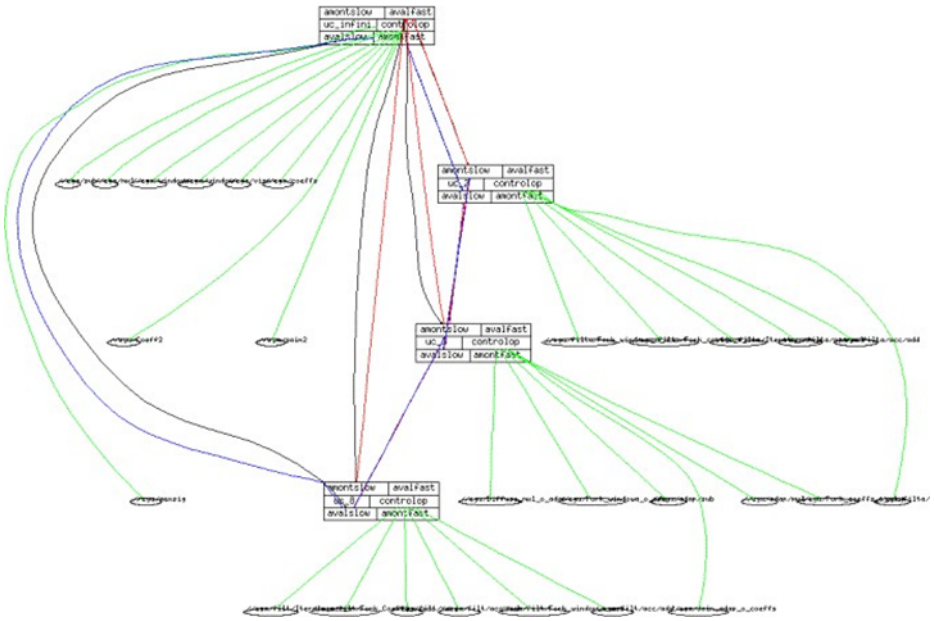


FIGURE 6.10 : Graphe de voisinage de frontières

et les heuristiques de recuit simulés (simulated annealing, algorithmes d'optimisation stochastiques permettant de sortir d'un minimul local). Ces dernières, basées sur des tirages aléatoires, donnent de meilleures solutions mais ont l'inconvénient d'être plus lentes que les gloutonnes.

6.2.7 Implémentation des résultats : SynDEx-Ic

Nous avons implanté les résultats de ces travaux en développant de nouveaux modules logiciels qui transforment SynDEx V6.6.x (logiciel de l'INRIA qui implémente la méthodologie AAA pour architectures multi-composants) en SynDEx-Ic (logiciel qui couvre donc les architectures à base de mono-FPGA). Ces modules, de même que la version 6 de SynDEx, sont écrits en CAML (langage de type fonctionnel développé à l'INRIA). Les modules développés viennent s'interfacer avec une partie commune de SynDEx en reprenant notamment la structure de données représentant le GFDD de l'algorithme à implanter. A partir de cette structure, notre premier module construit un nouveau graphe interne auquel on peut appliquer aussi bien le processus de défactorisation que la caractérisation en temps et surface. Dans ce graphe chaque sommet est étiqueté par un couple représentant respectivement sa surface et sa latence. Cette structure comprend le graphe de voisinage de frontière et le GFDD. A partir de cette structure il est possible de générer automatiquement le code VHDL synthétisable grâce au module Genexec qui transforme les graphes GVF et GFDD en code macro code m4[3]. Chaque macro de ce code est remplacée par sa définition issue de librairies m4 que nous avons développées pour chaque architecture FPGA. Après la compilation des fichiers m4, nous obtenons le code VHDL synthétisable qui peut ensuite alimenter la chaîne de CAO classique du FPGA pour effectuer la simulation ou la synthèse. Ce logiciel est mis librement à disposition sur le site internet de notre laboratoire (http ://www.esiee.fr/a2si/web-ca/syndex-ic).

[3] m4 est le macroprocesseur du monde Unix

6.3 Modélisation d'architectures mixtes - FPGA - GPP/DSP

- *Thèse de Oussama Feki : Contribution à l'Implantation Optimisée d'Estimateurs de Mouvements de la Norme H264 sur Plates-Formes Multi-Composants par Extension de la Methodologie AAA [36]*
- *Direction co-tutelle : Mohamed Akil (Pr. UGE) et Nouri Masmoudi (Pr. université de Sfax - Tunisie)*
- *Co-encadrement : Thierry Grandpierre (80%)*
- *Soutenue le 13/05/2015 - Durée : 65 mois*[a]
- *Financement PHC-Utique, Francophonie et université de Sfax*
- *École Doctorale : MSTIC - Mathématiques et Sciences et Technologies de l'Information et de la Communication n°532*
- *Publications : [38, 39]*

[a] suspension de 12 mois

6.3.1 Contexte et objectifs

Cette thèse fait suite à celle de Linda Kaouane présentée dans la section précédente (6.2). Nous avons vu que les architectures à base de FPGA sont très performantes pour implanter des algorithmes à fort parallélisme de calcul de type flot de données. En revanche, elles ne sont pas adaptées aux algorithmes basés sur des structures de données complexes et qui effectuent des accès aléatoires à la mémoire. Ces propriétés sont en effet plus adaptées à des implémentations programmées (i.e. sur processeurs). Pour bénéficier du meilleur des deux, nous nous intéressons maintenant à l'implémentation temps réel sur des architectures dites "mixtes" : à base de composants programmables (processeurs d'usage général mais aussi DSP) et de composants reconfigurables (FPGA).

Etant donné l'ampleur du travail d'implantation et d'optimisation d'une telle application temps réel sur ce type d'architecture, il est important de proposer une méthodologie rigoureuse. Nous basons donc nos travaux sur une nouvelle extension de la méthodologie de prototypage rapide Adéquation Algorithme Architecture (AAA).

6.3.2 Approche proposée

Dans l'approche que nous proposons (figure 6.11), l'utilisateur décrit d'une part son algorithme par un graphe flot de données factorisé, et d'autre part son architecture mixte par un second graphe. Nous avons développé une heuristique (présentée plus loin), capable de distribuer et ordonnancer chaque élément du graphe d'algorithme sur chaque élément du graphe d'architecture en fonction des contraintes temporelles données.

Nous avons vu précédemment dans ma thèse que AAA et l'outil SynDEx couvrent les architectures programmables. Puis, nous avons vu que les travaux de thèse de Linda Kaouane, implantés dans le logiciel SynDEx-IC, permettent de couvrir les architectures reconfigurables à base de mono-FPGA.

Pour couvrir le champs des architectures mixtes (i.e. composées de processeurs et de FPGA), nous avons associé ces deux outils, en passant par quatre étapes :

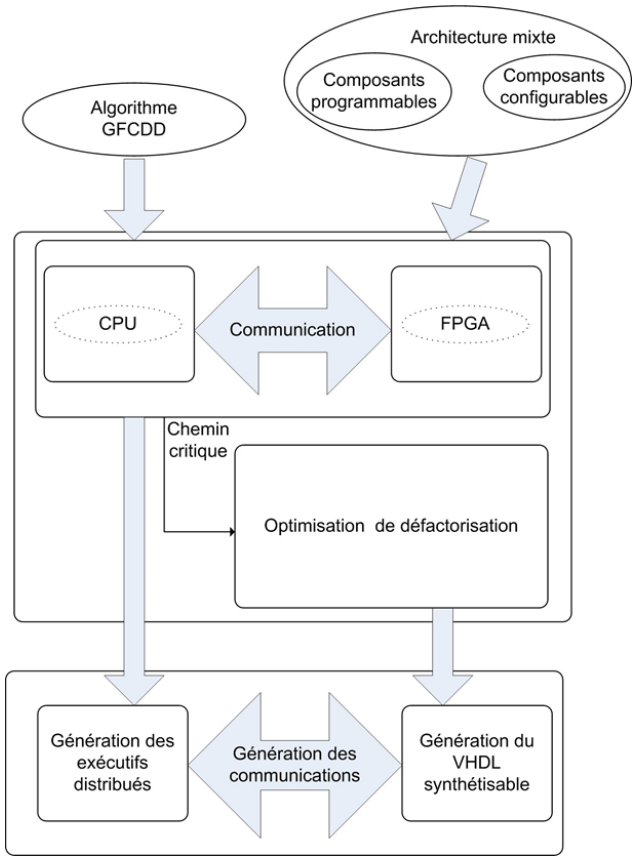


FIGURE 6.11 : Extension aux architectures mixtes

1. Etendre le modèle formel d'architectures afin de pouvoir modéliser l'ensemble des composants programmables et des composants reconfigurables dans un même formalisme, tout en tenant compte des spécificités de chaque type.
2. Coupler les heuristiques d'optimisation des deux outils.
3. Générer automatiquement les communications entre les composants programmables et les composants reconfigurables. Générer les codes correspondants à l'implémentation optimisée de l'application sur l'architecture mixte.
4. Créer un nouvel outil utilisant partiellement SynDEx et SynDEx-IC pour pouvoir optimiser l'implantation des applications sur les architectures mixtes.

6.3.3 Extension des modèles d'architecture et d'implantation AAA

Dans l'extension de AAA aux architectures programmables, seul le graphe d'algorithme était pris en compte car la cible est une architecture mono-FPGA modélisée finalement par un nombre maximum de cellules reconfigurables.

Afin de pouvoir décrire des architectures mixtes composées de circuits programmables et reconfigurables dans un même formalisme, la modélisation d'un FPGA a été étendue pour prendre en compte le nombre de ressources offertes, les communications mais également modéliser les phases de configuration.

Ainsi, pour modéliser les architectures programmables, le modèle d'architecture comporte des sommets *opérateurs*, *communicateurs*, *mémoires* et *bus, multiplexeurs et démultiplexeurs*. Pour pouvoir modéliser les architectures reconfigurables dans un même formalisme, le modèle d'architecture comporte maintenant de nouveaux sommets *opérateur élémentaire* et *opérateur configuré*.

Notons que la configuration d'un circuit reconfigurable (transformation d'une architecture non configurée en une architecture configurée) est également décrite formellement en terme de transformation de graphes.

6.3.4 Couplage des heuristiques

Avant de présenter la solution retenue, nous rappelons maintenant les objectifs de l'heuristique utilisée pour la partie programmable et celle utilisée pour la partie reconfigurable.

Dans le cas des architectures programmables, l'objectif de l'heuristique d'optimisation est de distribuer chaque opération du graphe d'algorithme sur chaque élément du graphe d'architecture en générant les opérations de communication nécessaires pour transférer les données. La fonction de coût implantée cherche à minimiser la longueur du chemin critique tout en utilisant un minimum de ressources matérielles.

Dans le cas des architectures reconfigurables, l'objectif de l'heuristique d'optimisation est de trouver le meilleur taux de défactorisation de chaque frontière de factorisation (i.e. le nombre de répétitions pour chaque boucle) de façon à atteindre une durée d'exécution maximale donnée tout en ne dépensant pas le nombre de ressources disponibles.

Plutôt que de re-developper une nouvelle heuristique, nous avons donc couplé ces deux heuristiques, préalablement éprouvées dans leurs domaines respectifs. Le principe est le suivant : le parcours du graphe d'algorithme s'effectue classiquement selon l'ordre des dépendances de données. Chaque fois qu'une frontière de factorisation est rencontrée l'heuristique de défactorisation est appelée avec une contrainte calculée de façon à ne pas allonger le chemin critique global de l'application.

6.3.5 Synthèse des communications

Dans cette approche, les communications sont synthétisées sur mesure en fonction des décisions d'allocation des heuristiques. Plus précisément, l'interface de communication entre un composant programmable et un composant reconfigurable est modélisée au choix par une mémoire à accès séquentiel (type FIFO) ou aléatoire (RAM), partagée entre ces deux éléments. Côté programmable, un opérateur ou un communicateur peut y lire et/ou écrire des données. Côté reconfigurable, c'est un communicateur spécifique qui est synthétisé, basé sur une machine à état générée en fonction des décisions de distribution/allocation de l'étape précédente.

6.3.6 Développement logiciel

L'ensemble des résultats a été implémenté sous la forme d'un script python qui appelle tour à tour les logiciels SynDEx et SynDEx-IC en ligne de commande et récupère leurs résultats intermédiaires sous forme de fichiers xml. Au final notre outil fournit un exécutif pour chacun des composants programmables (i.e. un fichier C compilable par processeur, incluant le code des communications), et un code VHDL synthétisable pour chaque composant programmable connecté.

6.3.7 Validation

En guise d'exemple de validation, nous avons implanté un algorithme de compression vidéo numérique. En effet, la taille des vidéos doit être minimisée pour résoudre les problèmes de stockage, mais aussi pour faciliter leurs transferts par des médias à faible bande passante. La compression vidéo est basée sur l'exploitation des redondances spatiales et temporelles à l'intérieur des images et entre les images. Une fonction incontournable de la compression vidéo repose sur la recherche de sous-blocs d'images identiques entre des images successives. Cette recherche est basée sur l'estimation de mouvement, une opération clé en compression vidéo, mais qui implique une complexité de calcul conséquente étant donné le nombre de déplacements possibles à explorer entre images successives. Sur un encodeur vidéo, jusqu'à 60% de la charge de calcul est dédiée à cette opération. L'évolution des besoins, en parallèle avec l'évolution des capacités de calcul nous amène vers l'évolution de standards de compression vidéo (ex. MPEG-4 H.264/AVC au moment de la thèse) permettant la haute définition. Les contraintes temps réel sont donc accrues, et des architectures mixtes peuvent répondre à ces besoins. C'est pourquoi nous avons validé notre méthodologie par l'implantation optimisée d'un estimateur de mouvement sur une architecture mixte composée d'un processeur NIOS II connecté à un FPGA Xilinx "Stratix EP3SL150F1152C2".

6.4 Modélisation d'architectures reconfigurables à gros grain (SPS-CGRA)

- *Thèse de Elias Barbudo : Towards Efficient Reuse of Software Programmable Streaming Coarse Grained Reconfigurable Architectures [23]*
- *Direction : Eva Dokladalova (Pr. UGE)*
- *Co-encadrement : Thierry Grandpierre (50%)*
- *Soutenue le 29/06/2021 - Financement Conacyt - Durée 43 mois*
- *École Doctorale : MSTIC - Mathématiques et Sciences et Technologies de l'Information et de la Communication n°532*
- *Publications : [18, 22, 15, 20]*

6.4.1 Contexte et objectifs

Dans cette thèse nous continuons à explorer les possibilités offertes par les architectures à base de composants reconfigurables (FPGA), mais avec une approche assez différente de ce qui a été fait précédemment. En effet, nous venons de présenter une méthodologie qui, à partir d'un algorithme décrit sous la forme d'un graphe flot de données, permet de générer la configuration d'un FPGA et de générer également le code des éventuels processeurs connectés à ce FPGA. Dans l'approche présentée ici, il s'agit d'exploiter plus efficacement un FPGA préalablement configuré à gros grains.

Les architectures reconfigurables à gros grains (Coarse Grained Reconfigurable Architectures, CGRA) sont conçues pour offrir des performances élevées tout en réduisant considérablement la latence de calcul. Elles sont en général utilisées sous forme de co-processeurs pour accélérer la partie calcul intensif des algorithmes. Il

existe plusieurs types de CGRA en fonction de la structure, de l'application, du type de ressources et de l'organisation de la mémoire. Nous concentrons notre travail sur un sous-ensemble de CGRA que nous nommons "architectures gros grain, flot de données, reconfigurables et programmables" (Software Programmable Streaming - Coarse Grained Reconfigurable Architectures, SPS-CGRA). Un SPS-CGRA est une grille plus ou moins complexe de ressources matérielles hétérogènes à gros grains avec une granularité plus grande que les architectures CGRA classiques. Un SPS-CGRA peut effectuer de grandes quantités de calculs avec une faible latence. Son principe de traitement orienté flot de données offre des performances élevées tout en maintenant un niveau élevé de flexibilité. Bien que les SPS-CGRA soient souvent hautement optimisés pour un domaine spécifique, ils conservent plusieurs niveaux de paramétrage après la phase de configuration. Ces paramétrages sont presque comparables à la programmation d'un processeur dont les instructions seraient des fonctions complexes, qui fonctionneraient essentiellement de façon séquentielle (pas de structure de contrôle élaborée). Ce paramétrage rend ainsi théoriquement possible la réutilisation de ces architectures. Cependant, cette réutilisation est généralement limitée en raison de la complexité de l'identification de la meilleure allocation des tâches de traitement sur les ressources matérielles. Un autre facteur limitant la réutilisation réside dans la complexité à produire une analyse de performance fiable pour chaque nouvelle implémentation car généralement il n'existe aucun outil spécifique pour explorer et exploiter pleinement le potentiel des architectures ainsi produites.

Pour illustrer notre propos, la figure 6.12 présente le "Morphological Co-Processing Unit (MCPU)", un SPS-CGRA développée dans notre laboratoire [25]. Entièrement implanté dans un FPGA, il s'interface avec le bus PCI d'un PC pour accélérer des opérateurs morphologiques en traitement d'images. Il possède ainsi 2 pipelines configurables capables de lire-écrire des données dans des FIFO d'entrées-sorties. Le FPGA est configuré une fois pour toute, mais les paramètres que l'on peut lui envoyer permettent de choisir les fonctionnalités (opérations) et chemin de données dynamiquement (sans reconfiguration) en fonction de l'algorithme à implanter.

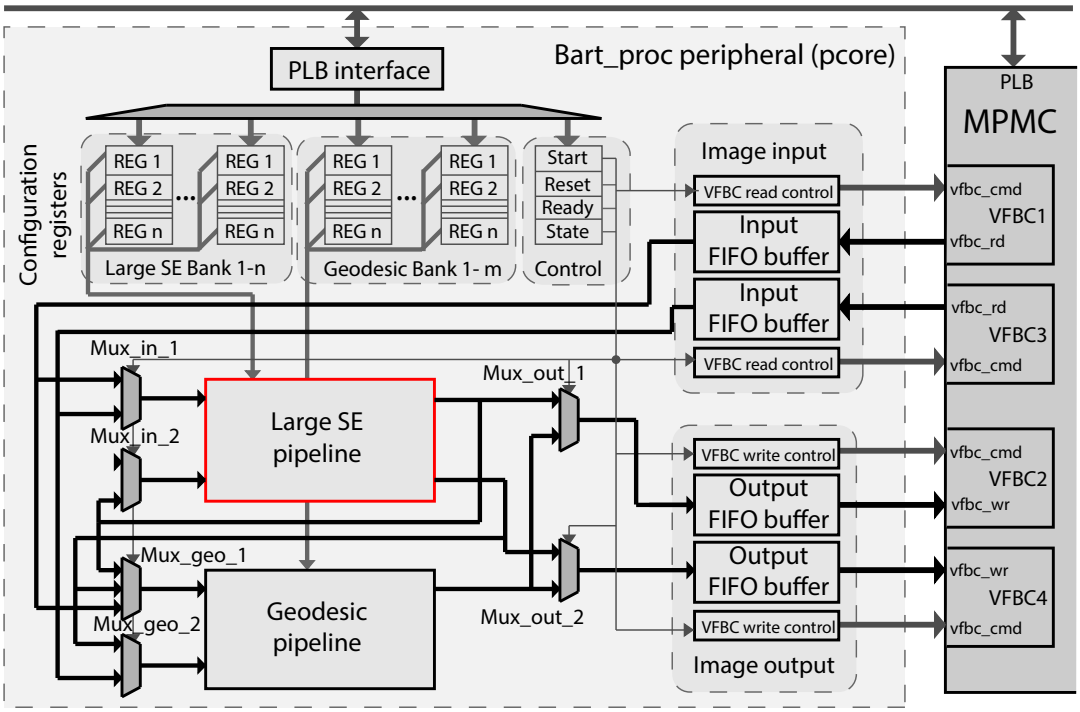


FIGURE 6.12 : Morphological Co-Processing Unit

La figure 6.13 présente une vue détaillée de l'unité "LargeSE pipeline" de la figure précédente. Ce sous-ensemble est composé de blocs paramétrables capables de réaliser des opérations morphologiques précâblées au choix parmi une liste prédéfinie (par exemple : "Large Structuring Element", "Erosion", "Dilatation"). Il est aussi possible de choisir des paramètres pour l'opérateur sélectionné (taille de l'élément structurant, etc.). En fonction de leurs paramètres de configuration chacun de ces opérateurs peut lire ou écrire dans les mémoires ou bien communiquer avec un autre directement.

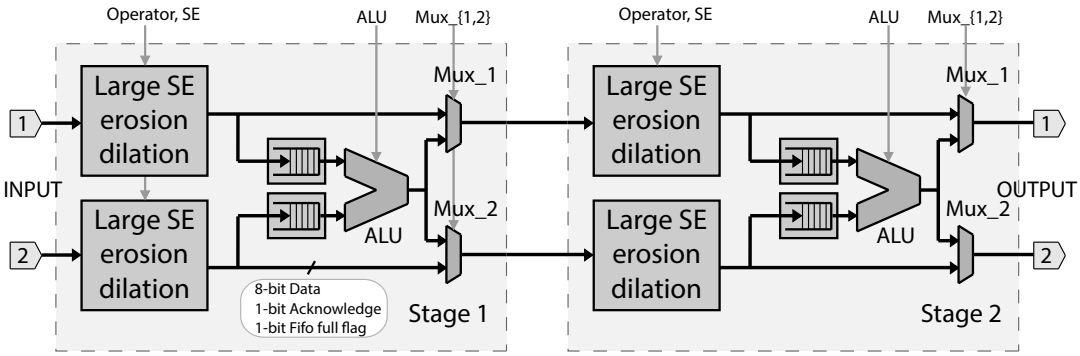


FIGURE 6.13 : Pipeline configurable du MCPU

6.4.2 Approche proposée

Pour aider à l'optimisation sur ces architectures et pour favoriser leur réutilisation, nous proposons un cadre complet de spécification, distribution et d'ordonnancement qui cible les SPS-CGRA (figure 6.14). Calqué sur la méthodologie AAA présentée plus haut, ce cadre repose sur un formalisme de graphe pour modéliser l'algorithme applicatif, l'architecture matérielle, puis l'optimisation de l'implantation

(distribution, ordonnancement) du graphe d'algorithme sur l'architecture en se basant sur une estimation des performances.

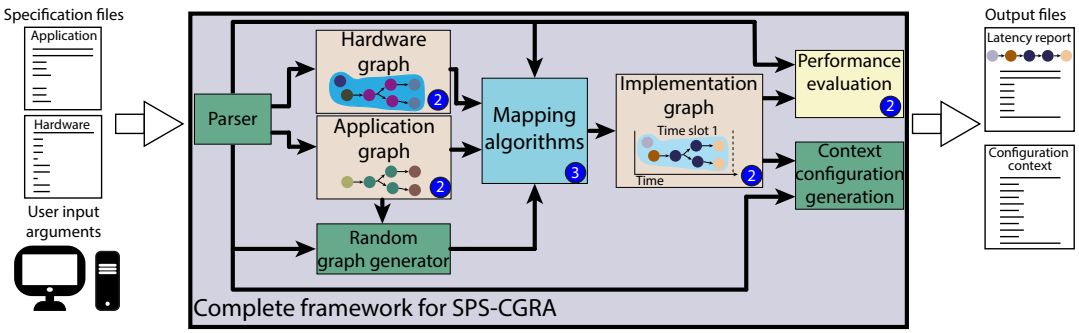


FIGURE 6.14 : Méthodologie proposée

Nous introduisons donc un modèle théorique et générique de l'architecture matérielle permettant d'exprimer ses niveaux de flexibilité intrinsèquement personnalisés. Ce modèle permet aussi de modéliser finement l'accès aux données dans les mémoires. Enfin, il permet également de modéliser le contrôle de la configuration du système, souvent négligé dans les travaux existants. Nous proposons également une analyse d'estimation des performances, basée sur la latence des ressources. Pour l'optimisation de l'implantation, nous présentons quatre solutions différentes de distribution et d'ordonnancement : un algorithme avec retour en arrière basé sur des listes, une heuristique basée sur les algorithmes de type "lookahead", une heuristique basée sur un algorithme Bayésien et un algorithme d'ordonnancement basé sur le Q-learning. Pour finir, nous évaluons et comparons nos solutions sur des ensembles d'architectures et d'applications dont les paramètres sont générés aléatoirement, ainsi que sur deux applications réelles.

6.4.3 Modèle d'architecture

En cohérence avec les travaux précédents, l'architecture matérielle SPS-CGRA est décrite par un hypergraphe orienté dont les sommets représentent les ressources matérielles disponibles et les arcs les connexions entre ces ressources. Plusieurs types de ressources ont été identifiés et donnent lieu à plusieurs catégories de sommets : les sommets ressources de calcul, ressources de mémorisation, ressources de communication et enfin les ressources de configuration (pour modéliser la "paramètrisation" de l'architecture SPS-CGRA).

La figure 6.15 présente le graphe d'architecture correspondant au "LargeSE pipeline" du MCPU présenté précédemment (figure 6.13).

Le sommet séquenceur de configuration s^{cfg} controle les paramètres de tous les autres sommets avec lesquels il est connecté par l'hyperarc bleu. Les opérations de calculs sont modélisées par les sommets "processing" r^{P_5} à $r_2^P 0$. La lecture et l'écriture dans les modules "Image input" et "Image output" sont respectivement modélisés par les sommets "sensors" $r_{0,1}^{SNSR}$ et "actuators" $r_{21,22}^{ACTR}$. Ceux-ci écrivent et lisent dans la mémoire partagée $r_2^{M_S RC}$ qui permet 2 accès simultanés en lecture par les sommets r^{RD_3} et r^{RD_4} et en écriture par $r^{WR_1}8$ et $r^{WR_1}9$. Les sommets r^{MUX} correspondent aux connexions physiques et aux multiplexeurs.

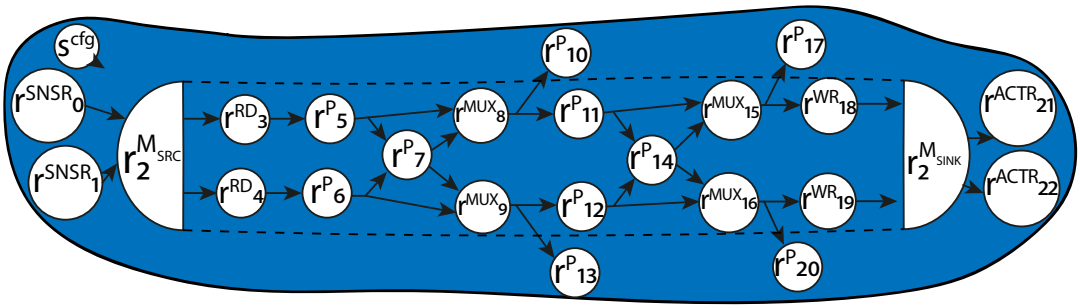


FIGURE 6.15 : Graphe d'architecture de la figure 6.13

6.4.4 Caractérisation et estimation de performances

L'objectif étant de minimiser la durée d'exécution de l'implantation, il est nécessaire d'estimer les performances de chaque implantation au fur et à mesure de la phase d'optimisation qui sera présentée dans la section 6.4.7. Il est donc indispensable de disposer d'un modèle de caractérisation et de prédiction des performances.

C'est pourquoi, nous devons au préalable spécifier la latence (en nombre de cycles horloge) de chaque sommet du graphe d'algorithme. La latence d'une opération dépend de l'opérateur qui va l'exécuter (il peut y avoir plusieurs implémentations pour chaque opération). Plus précisément, pour chaque couple opérateur/opération, nous définissons la latence d'entrée et la latence de calcul.

6.4.5 Modèle d'implantation

L'implantation d'un graphe d'algorithme sur une architecture est modélisée par un graphe d'implantation. Il est basé sur un ensemble de sous-graphes qui peut être disjoint ou non. Chaque sous-graphe s'exécute sur une instance du graphe d'architecture. Chaque sous-graphe sera appelé "slot temporel". Ainsi, un nouveau slot temporel sera créé chaque fois qu'il y aura pénurie de ressources matérielles par rapport aux besoins du graphe d'algorithme. Ceci permet de modéliser la ré-utilisation des ressources après re-paramétrage. En effet, lors d'un slot temporel, les paramètres de configuration ne doivent pas être modifiés.

La figure 6.16 présente un graphe d'algorithme (appelé graphe d'application sur la figure et dans la thèse), un graphe d'architecture, puis un exemple de graphe d'implantation correspondant à un seul slot temporel.

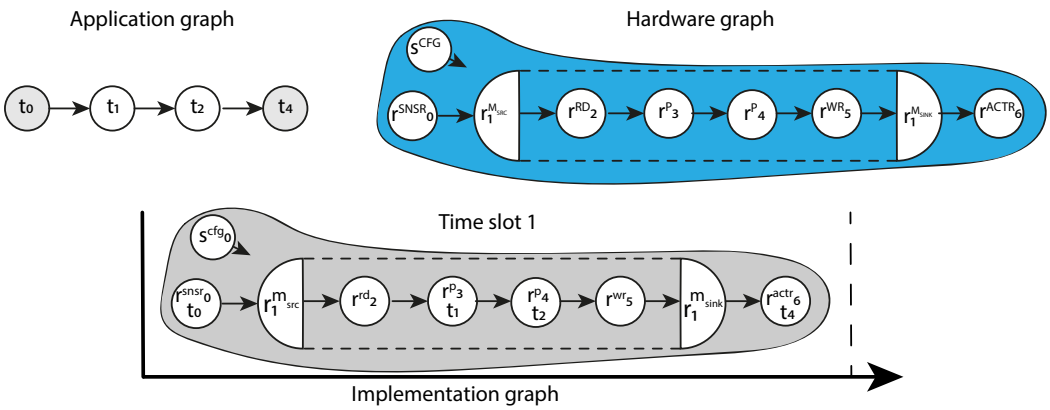


FIGURE 6.16 : Graphe d'implantation comportant 1 slot temporel

La figure 6.17 présente un graphe d'algorithme plus grand (une opération supplémentaire), qui ne peut s'exécuter en une seule fois sur le graphe d'architecture. Il faut donc créer 2 slots temporels pour l'exécuter, ce qui apparaît sur le graphe d'implantation sous le nom "Time slot 2". A noter que pour chaque slot temporel, le temps de configuration des paramètres est modélisé par les sommets S^{cfg_0} et S^{cfg_1}. Ils sont connectés aux autres opérations par les hyperarcs symbolisés par la couleur grises sous l'ensemble des sommets connectés.

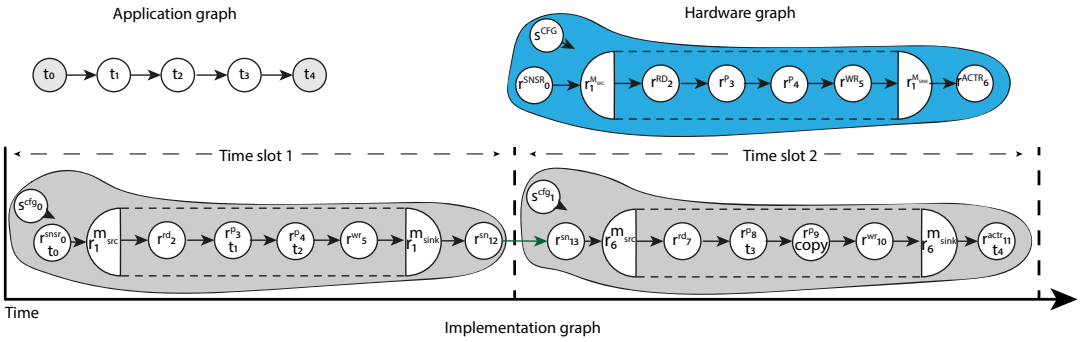


FIGURE 6.17 : Graphe d'implantation comportant 2 slots temporels

6.4.6 Évaluation de performances

Pour calculer le pire temps d'exécution d'une implantation nous nous basons sur la longueur du chemin critique CC du graphe d'implantation :

$$CC = \sum_{i=1}^{N} (T\text{IN}_i + T\text{EX}_i + T\text{C}_i) \tag{6.1}$$

- N est le nombre de slots temporels du graphe,
- $T\text{IN}_i$ la latence d'entrée du slot temporel i. Elle correspond au nombre de cycles d'horloge entre l'arrivée du premier échantillon dans le premier slot temporel et le premier échantillon produit par ce même slot. On calcul $T\text{IN}_i$ à partir de la latence d'entrée et de la latence de calcul de chaque ressource :

$$T\text{IN}_i = \sum_{j=1}^{|CP_i|-1} (\mathcal{L}_j^{IN})(\omega_j) + \mathcal{L}_j^{CL} \tag{6.2}$$

  - CP_i est l'ensemble des ressources du chemin critique du slot i,
  - $\mathcal{L}_j^{IN}$ la latence d'entrée de chaque ressource,
  - $\mathcal{L}_j^{CL}$ la latence de calcul de chaque ressource,
  - ω_j est une variable qui permet d'exprimer la propagation de l'impact de la latence de calcul de prédécesseurs en successeurs, c'est le paramètre de propagation de la latence :

$$\omega_j = max(\omega_{j-1}, \mathcal{L}_{j-1}^{CL}) \tag{6.3}$$

- La durée d'exécution $T\text{EX}_i$ d'un slot temporel i. C'est le temps d'exécution nécessaire pour traiter tous les échantillons à partir du moment où le pipeline est plein et peut produire un flot continu d'échantillons de sortie.

$$T\text{EX}_i = (CL_i)(TS) \tag{6.4}$$

  - CL_i est le pire temps du chemin critique (valeur finale de ω),
  - TS est le nombre total d'échantillons d'entrées du slot i.

- Le temps de configuration TC_i d'un slot temporel i est égal à la durée associée au sommet s_i^{cfg}

A partir de ce modèle, nous pouvons donc calculer la durée d'exécution pire cas de l'implantation d'un algorithme sur une architecture donnée. La figure suivante (6.18), tirée de la thèse, est un exemple de graphe obtenu à partir de ce modèle.

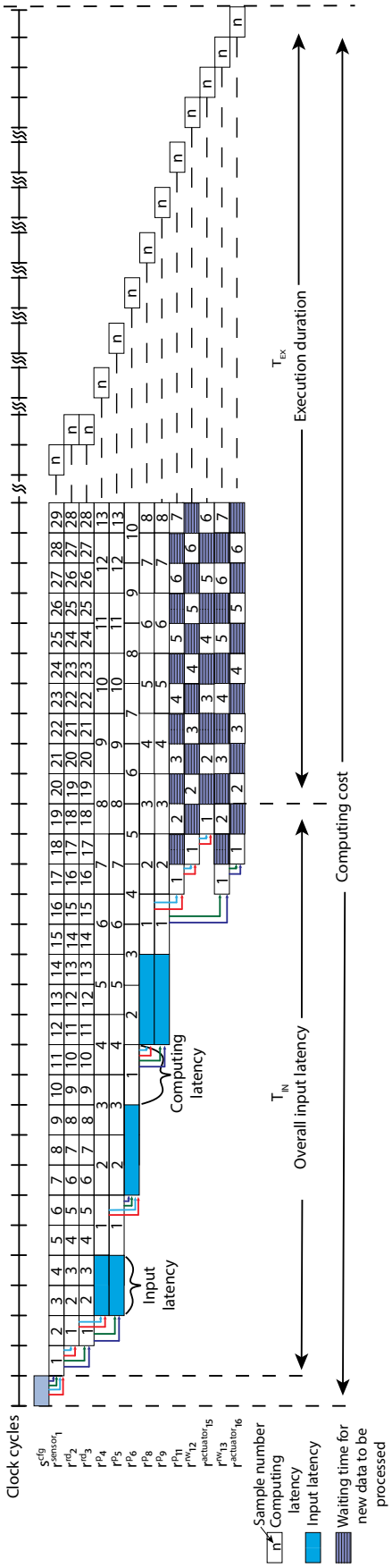


FIGURE 6.18 : Execution temporelle d'une application : latence d'entrée, latence de calcul

6.4.7 Optimisation

Disposant maintenant des modèles d'architecture, d'implantation et d'un modèle d'estimation de performances, nous proposons des techniques d'optimisations permettant d'explorer l'espace des implantations afin d'y identifier et sélectionner celles qui respecteront les contraintes temps réels données. Pour répondre à ce problème de distribution-ordonnancement, nous proposons 4 approches : l'utilisation d'une heuristique rapide basée sur de l'ordonnancement par liste (list scheduling) associée à du retour-arrière (back-tracking), une variante qui recherche plus avant ("Look-ahead based heuristic") car basée sur la connaissance de la topologie de l'architecture, une heuristique basée sur une approche Baysienne, et enfin un algorithme basé sur du Q-Learning.

La première heuristique (Single Shot Mapping Algorithm, SSMAP) est classiquement basée sur une approche par liste de candidats parmi laquelle l'algorithme sélectionne l'opération de calcul à implanter à l'aide d'une fonction de coût basée sur la latence de calcul, la distance topologique et le nombre de ressources compatibles (c'est à dire les ressources capables d'exécuter l'opération). Ceci permet de construire rapidement une solution valide mais pas nécessairement la meilleure. Elle est capable de créer des slots temporels, mais a pour inconvénient de donner de mauvais résultats dans le cas de plusieurs capteurs en entrée. Elle sert essentiellement de benchmark pour comparer les résultats avec les heuristiques suivantes.

Afin de produire de meilleures solutions, notre seconde heuristique (Topology Aware Mapping Algorithm, TT-MAP) reprend les principes de la première, mais calcul l'impact de l'ordonnancement de chaque opération sur chacun de ses successeurs. Pour cela elle prend aussi en compte la distance topologique, la latence de calcul et le nombre de ressources compatibles. Elle ajoute aussi une nouvelle fonction de coût qui correspond à la probabilité de réussir à ordonnancer les successeurs d'une opération. Cette heuristique permet de construire de meilleures implémentations que la première mais reste cantonnée dans des minimas locaux.

Pour palier à cela, notre troisième approche est basée sur une approche Bayséienne, c'est à dire l'observation et la probabilité. Au cours de l'allocation et l'ordonnancement cette méthode analyse dynamiquement les conséquences du placement de chaque opération et de ses successeurs afin d'augmenter la probabilité d'obtenir un ordonnancement optimal ou sous-optimal. Concrètement cela revient à introduire dans la fonction de coût un calcul de probabilité de réussite pour chaque couple (opération, ressource) en prenant en compte également les probabilités passées. Cet algorithme est bien adapté aux applications complexes pour lesquelles les deux premiers algorithmes donnent de moins bon résultats.

Cependant, toujours à la recherche de meilleurs résultats notre quatrième approche repose sur du Q-Learning (apprentissage par renforcement). Cette technique utilise des agents capables d'identifier une ressource de traitement appropriée pour une tâche donnée après avoir "appris" sur des exemples précédents. Le processus d'apprentissage de cet agent est basé, entre autres, sur le type et les paramètres de la ressource de traitement de l'opération, la connectivité entre les ressources et les dépendances des données du graphe d'application. Ce type d'algorithme repose donc sur une phase d'entraînement qui commence hors ligne et se poursuit "en ligne". Comme le nombre d'applications associées à une architecture donnée est souvent limité, nous générons aléatoirement des graphes sur lesquels l'algorithme peut s'entrainer hors-ligne. Cette approche permet de répondre aux cas d'applications complexes où le nombre d'entrées-sorties de chaque opération est élevé.

6.4.8 Validation

L'ensemble de ces travaux est implanté dans un logiciel écrit en langage Python (environ 10000 lignes sans les commentaires, disponible sur github[4]) utilisant les bibliothèques Networkx (pour la manipulation de graphe), Graphviz et Matplotlib pour les affichages. Les temps d'exécution de chaque heuristique ont été comparés sur différents exemples pour lesquels nous avons utilisé une méthode exhaustive (ou "brut force") pour obtenir la solution optimale. Ceci nous a permis de constater que l'algorithme basé sur le Q-Learning donne les meilleurs résultats quelque soit le type de graphe d'application donné alors que les 3 autres donnent surtout de bons résultats quand les graphes d'applications contiennent à la fois peu d'hétérogénéité dans le type des opérations et peu de branches parallèles, disjointes ou non. Notre outil a également été validé avec succès sur deux applications réelles dans le domaine du traitement des images. Il a été utilisé sur notre architecture MCPU présentée plus haut (Cf. 6.12) pour implanter deux algorithmes : un "filtre séquentiel alterné" issu de la morphologie mathématique qui a pour but de flouter les images en préservant les caractéristiques topologiques, et un algorithme de détection de lignes pour la conduite automatique de véhicules.

[4]https ://github.com/ebarbudo/ MappingSPSCGRA

Chapitre 7

Implémentation et optimisation d'applications temps réel critiques

Le chapitre précédent traitait de mes contributions dans le domaine des méthodologies de conception d'application temps réel embarquées. Ce chapitre présente celles dans le cadre du domaine du temps réel critique (strict). Elles prennent la forme du co-encadrement d'une thèse soutenue, de la participation active dans un projet FUI et se poursuit actuellement par le biais du co-encadrement d'une thèse CIFRE tous trois présentés dans les sections suivantes.

7.1 Ordonnancement d'applications temps réel sur architectures multi-processeurs à l'aide de la virtualisation

- *Thèse de Tristan Fautrel : Analyse et ordonnancement d'un système hiérarchique virtualisé composé d'applications temps réel strictes [34]*
- *Direction : Laurent George (Pr. UGE)*
- *Co-encadrement : Thierry Grandpierre (50%), Christophe Fauberteau (MCF ESILV)*
- *Soutenue le 06/05/2021 - Durée : 53 mois*[a]
- *Financement MESR*
- *École Doctorale : MSTIC - Mathématiques et Sciences et Technologies de l'Information et de la Communication n°532*
- *Publications : [33, 34, 35]*

[a] Suspension 6 mois

7.1.1 Contextes et objectifs

Lorsque plusieurs applications de criticités différentes s'exécutent sur la même plateforme matérielle, l'application la plus critique impose, par contagion aux applications les moins critiques, une certification du même niveau d'exigence que celui de l'application la plus critique. Ceci n'est pas toujours possible, compte tenu de la complexité des applications, et peut conduire à un coût de certification très élevé puisqu'il faut faire une analyse globale. Une solution à ce problème consiste à utiliser un système matériel basé sur la virtualisation. Ainsi les ressources matérielles ne sont pas exposées directement aux applications mais passent par l'intermédiaire de cette couche de virtualisation. Cette dernière est elle même contrôlée par un hyperviseur

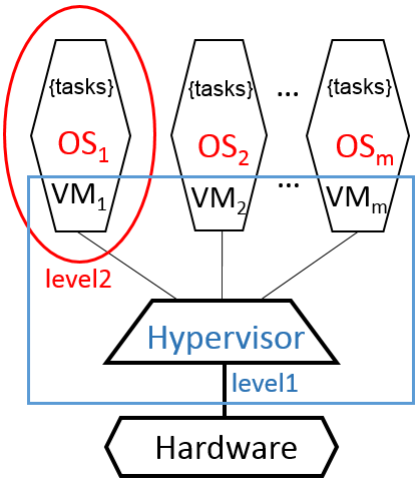


FIGURE 7.1 : Approche à deux niveaux : niveau 1 - hyperviseur controlant les VMs, niveau 2 - systèmes d'exploitations exécutant des tâches

capable d'assurer une isolation spatiale et temporelle des applications. Il existe aujourd'hui des hyperviseurs certifiés, garantissant ces isolations ainsi que le respect de contraintes temps réels. Dans ce type de système virtualisé et contrôlé par un hyperviseur certifié, chaque application s'exécute dans une machine virtuelle (VM) dédiée. L'activation des VMs est alors gérée par l'hyperviseur en charge de cette isolation.

Du point de vue de l'ordonnancement, cela forme un système hiérarchique à deux niveaux. Au premier niveau les VMs sont ordonnancées par l'hyperviseur. Au second niveau, les tâches de chaque VM sont ordonnancées par un algorithme d'ordonnancement spécifique. Dans nos travaux, nous considérons un ordonnancement classique à priorité fixe au sein du second niveau. L'objectif est en effet d'optimiser l'ordonnancement des VMs et non des tâches dans les VMs.

Comme nous le verrons dans la section suivante (Cf. 7.2), ces travaux s'inscrivent dans le cadre d'un projet FUI qui a pour but de développer un drone autonome certifiable de surveillance d'infrastructures civiles (pylones, conduites d'eau forcées, clotures ...).

Préalablement au projet, il a été décidé d'utiliser PikeOS, un hyperviseur de type 1 développé par la société Sysgo. En effet, au demarrage du projet, PikeOs etait le seul hyperviseur certifié DAL-A pour divers standards avioniques tels que ARINC 653[1]. Une de ses spécificités et atout pour ce projet, est qu'il est possible de maîtriser l'ordonnancement de niveau 1 des VMs. Ainsi, l'ordonnancement des machines virtuelles s'effectue à l'aide d'une table d'ordonnancement statique qui est construite hors ligne. En revanche, l'ordonnancement des tâches au sein des VMs est déterminé par le système d'exploitation temps réel associé à chaque machine virtuelle. Il est également possible d'introduire des tâches directement dans PikeOS qui seront alors ordonnancées par un algorithme d'ordonnancement à priorité fixe.

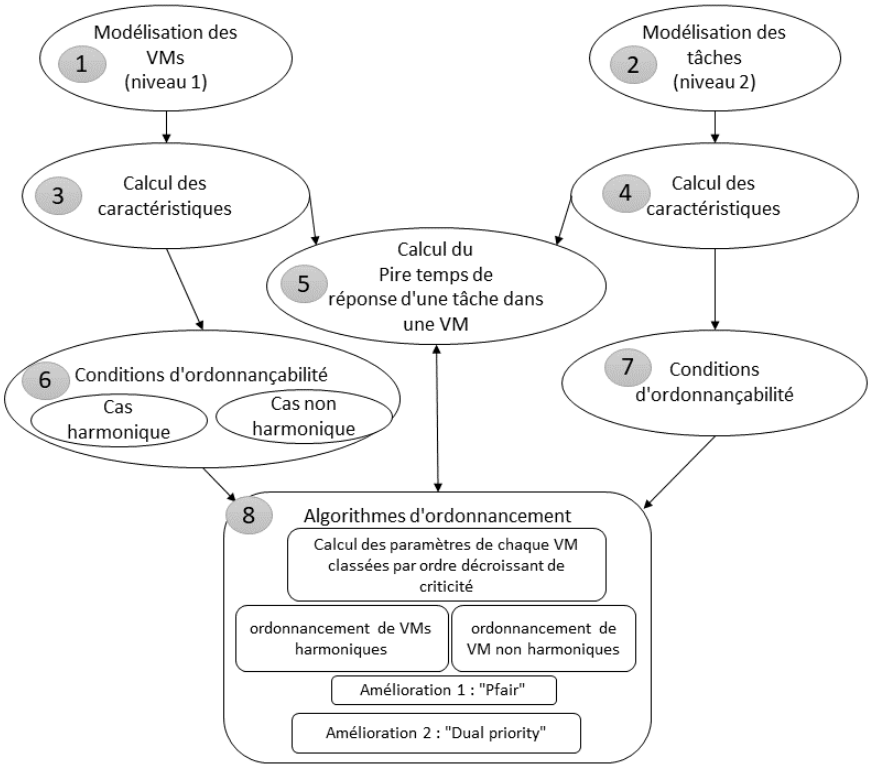


FIGURE 7.2 : Approche proposée dans la thèse

7.1.2 Approche proposée

Nous commençons par formaliser le modèle d'ordonnancement hiérarchique à deux niveaux (blocs 1 et 2 de la figure 7.2). Puis nous adaptons les travaux de la littérature pour notre approche hiérarchique (blocs 3 et 4). Nous en déduisons ensuite des conditions d'ordonnançabilité des VMs (bloc 5) et des tâches (bloc 6) ce qui nous permet d'élaborer le calcul du pire temps d'exécution d'une tâche sur un VM (bloc 7). Grâce à ces éléments nous pouvons appliquer nos algorithmes d'ordonnancement originaux afin de construire des ordonnancements compatibles avec les contraintes temporelles (bloc 8).

Nous proposons un premier ordonnancement de machines virtuelles garantissant un taux d'utilisation pour la VM selon une approche à périodicité stricte et pour des périodes d'activation harmoniques[2] des VMs. Cette approche conduit à activer les VMs selon un motif d'activation constitué d'un seul segment, activé périodiquement. Cependant, ce seul algorithme n'est pas toujours capable d'ordonnancer toutes les machines virtuelles. C'est pourquoi nous proposons un second algorithme, appelé algorithme proportionnel itératif, qui s'inspire de l'ordonnancement proportionnel (P-fair, Cf. 7.1.6). Il nous permet de compléter la table d'ordonnancement des machines virtuelles que la première approche n'a pas réussi à ordonnancer.

Ensuite, nous proposons des améliorations. Ainsi, concernant l'ordonnancement de niveau 2, nous considérons un ordonnancement à priorité fixe au niveau des tâches. Celles-ci sont modélisées par des tâches sporadiques à échéances arbitraires.

[1]C'est un standard aeronautique de partitionnement temporel et spatial de ressources informatiques

[2]multiples deux à deux entre elles

Nous établissons les conditions d'ordonnançabilité des tâches pour un motif arbitraire d'activation périodique de la VM en charge de l'exécution de ces tâches. Nous nous intéressons ensuite à l'ordonnancement de niveau 2 Dual Priority pour obtenir un meilleur taux de succès pour des tâches ordonnancées dans un modèle hiérarchique à deux niveaux. Cet ordonnancement bien que non optimal a un très bon taux de succès même pour des charges élevées. Cet ordonnancement nécessite d'affecter deux priorités fixes à la tâche : la première à la date de son activation, la seconde à une de ses échéances de promotion. Nous proposons deux algorithmes permettant de choisir de manière efficace les deux priorités fixes et l'échéance de promotion. Pour évaluer les différents taux de succès des algorithmes proposés, nous avons développé un simulateur d'ordonnancement temps réel, générique et extensible. Enfin, pour valider nos algorithmes, nous avons également développé un outil pour le projet FUI CEOS (Cf. 7.2) qui permet de déterminer la table d'ordonnancement des machines virtuelles à utiliser pour PikeOS.

Exemple

La Figure 7.3 présente un exemple de système extrait de la thèse, composé de deux VMs : VM_1 (dont l'ordonnancement est donné ligne n°1) et VM_2 (ligne n°5) sur lesquelles sont ordonnancées arbitrairement des tâches. Selon ce schéma, la VM_1 est activée de la date 0 à 3, puis de 5 à 6, de 8 à 12, de 14 à 15 et enfin de 18 à 20. Les lignes 1 et 5 représentent donc l'ordonnancement de niveau 1.

Dans nos travaux, le niveau 2, c'est à dire l'ordonnancement des tâches au sein des VMs repose sur Rate Monotonic (RM) choisi pour sa simplicité (d'autant qu'il ne s'agit pas ici d'optimiser l'ordonnancement des tâches, mais des VMs). Dans cet exemple, nous avons un jeu de six tâches pour deux VMs. Les tâches[3] τ_1 (1, 5, 5), τ_2 (3, 10, 10) , τ_3 (1, 20, 20) sont associées à la VM_1 et τ_4 (1, 5, 5), τ_5 (2, 10, 10) et τ_6 (1, 20, 20) sont associées à la VM_2.

En appliquant un ordonnancement RM sur la VM_1, c'est τ_1 qui a la plus grande priorité, suivi de τ_2 puis de τ_3. Sur VM_2 c'est τ_4, suivi de τ_5 puis τ_6. Bien évidemment, τ_1, τ_2 et τ_3 ne peuvent s'exécuter que quand la VM_1 est active et donc ordonnancée. Il en est de même pour τ_4, τ_5 et τ_6 avec VM_2. Dans un premier temps, nous montrons les ordonnancements obtenus par RM pour chaque tâche isolée (Lignes 2, 3, 4 et 6, 7, 8), puis nous regroupons ces ordonnancements pour chaque VM (lignes 9 et 10) pour montrer l'ordonnancement final du système (Ligne 11). C'est ce type d'ordonnancement que nos travaux visent à construire efficacement.

[3]Pour mémoire une tâche périodique $\tau_i(C_i, T_i, D_i)$ est une tâche qui génère des instances de tâches dont la durée d'exécution pire cas est notée C_i, dont l'inter-arrivée entre deux instances successives est égale à T_i et où D_i est l'échéance relative associée à toute instance de la tâche τ_i.

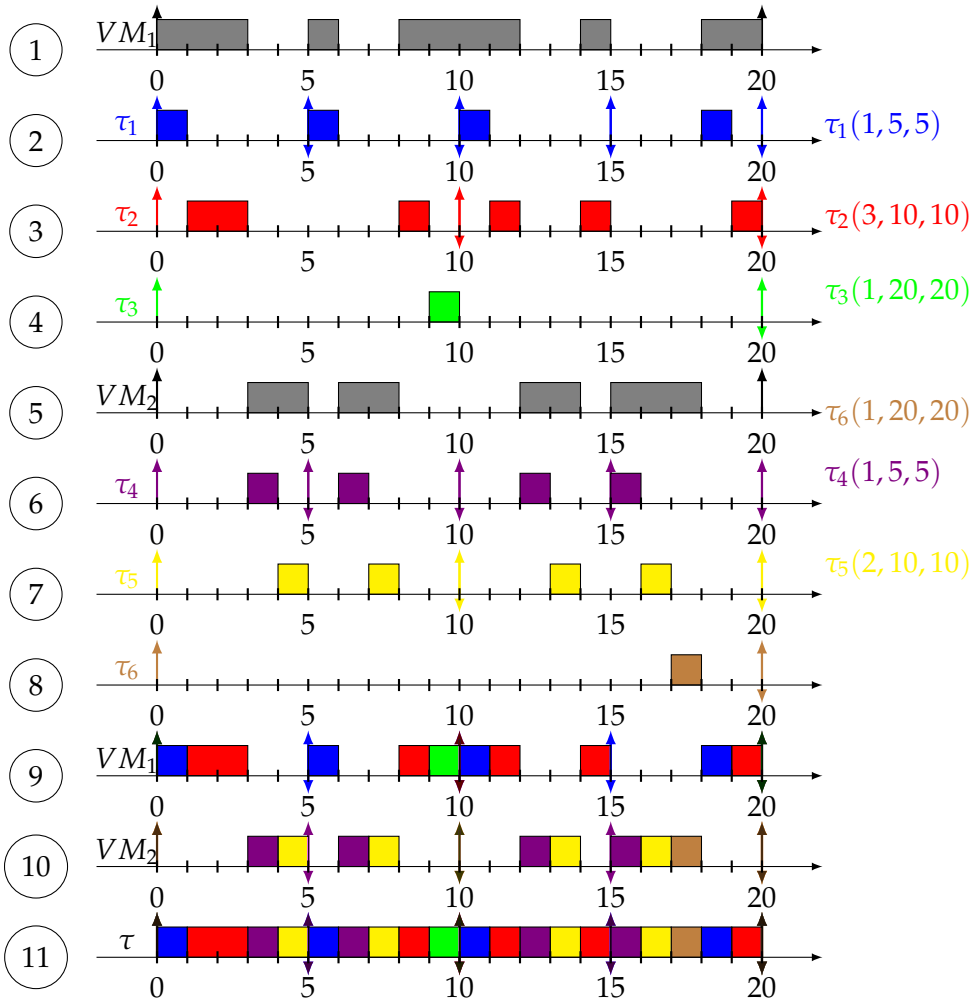


FIGURE 7.3 : Exemple d'ordonnancement arbitraire de deux VMs pour 5 jeux de tâches

7.1.3 Formalisation

Comme indiqué précédemment, notre modèle d'ordonnancement est basé sur un modèle hiérarchique à deux niveaux.

Niveau 1 : les machines virtuelles

Au premier niveau, on ordonnance un ensemble ρ de m machines virtuelles. Cet ensemble ρ est trié par ordre de criticité décroissante. Ainsi ρ^1 est la VM la plus critique et ρ^m la moins critique. Notre système est un processeur unique de notre système multiprocesseurs contrôlé par un hyperviseur en charge d'exécuter un ensemble d'applications de criticités différentes. Si aucun ordonnancement n'est trouvé pour assigner toutes les VMs sur un processeur unique, cette approche peut être itérée sur les autres processeurs de notre système. Dans nos travaux nous nous sommes limités au cas mono processeur.

Les algorithmes d'ordonnancement définis plus loin, reposent sur la notion de *candidat d'instant critique* que nous définissons donc maintenant à partir du motif de base (ou motif racine) d'une VM. Celui-ci permet de caractériser les périodes d'activité d'une VM exécutée périodiquement. Le segment de temps (ou durée) pendant laquelle une VM est exécutée correspond à un "slot d'activation". Au contraire,

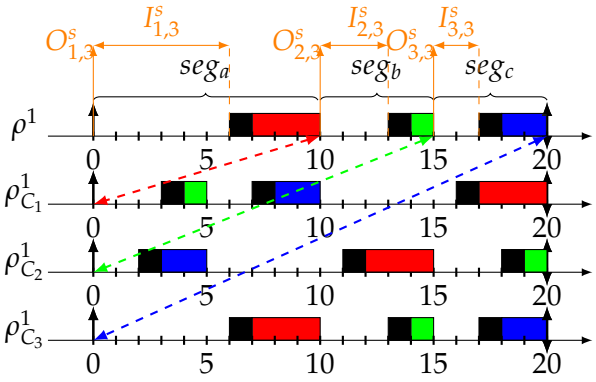


FIGURE 7.4 : Motif de base et 3 candidats d'instant critique

quand la VM n'est plus exécutée (c'est à dire suspendue) on parle de "slot inactif". Enfin, nous modélisons également la durée qui correspond à un changement de contexte (temps mis pour passer de l'état actif à inactif et inversement). Ce motif de base peut être défini comme une liste composée d'une succession de n_s segments. Soit $k \in [1, n_s]$, le segment σ^s_k de ρ^s est défini par :

- O^s_k l'offset du segment σ^s_k de ρ^s,
- I^s_k le segment d'inactivité qui débute à O^s_k durant lequel ρ^s est inactif,
- C^s_k est la durée du segment pendant lequel la VM est active (précédée d'un changement de contexte si la durée d'exécution de la VM est non nulle).

Pour illustrer ces définitions, la première ligne de la figure 7.4 présente un exemple de motif de base composé de trois segments (seg_a, seg_b and seg_c). Les rectangles noirs correspondent aux coûts des changements de contexte.

A partir d'un motif de base, nous avons étendu la notion d'instant critique d'une tâche pour définir l'instant critique d'une VM et ainsi définir les différents candidats d'instant critique pour une VM donnée. Ils correspondent au différentes dates possibles pour la fin de la VM. Ces candidats sont obtenus en introduisant un offset sur le démarrage du motif de la VM. La figure 7.4 présente 3 candidats d'instant critique : $\rho^1_{C_1}$, $\rho^1_{C_2}$ et $\rho^1_{C_3}$ obtenus par décalage du démarrage (flèches pointillées).

Niveau 2 : les jeux de tâches

Au sein du niveau 2 de notre modèle hiérarchique, chaque VM ρ^s accueille une application basée sur un sous-ensemble $\tau(\rho^s)$ constitué de n tâches sporadiques telles que $\tau = \{\tau_1, \ldots, \tau_n\}$ (chaque tâche ne peut être exécutée que par une seule VM). Une tâche sporadique task τ_i est alors définie par :

- son pire temps d'exécution (wcet[4]) c_i,
- son temps d'inter-arrivé minimal it_i,
- sa date de fin relative d_i.

Une tâche sporadique τ_i exécute donc au plus un job de durée maximale c_i tous les it_i unités de temps. Ainsi un job τ_i réalisé au temps t_0 doit être complété avant la date $t_0 + d_i$.

[4]Worst case execution time

L'utilisation des taches $\tau(\rho^s)$ est définie par $U^{\tau(\rho^s)} = \sum_{\tau_i \in \tau(\rho^s)} \frac{c_i}{it_i}$. Nos travaux ont montré qu'une condition nécessaire quand à l'utilisation des tâches de ρ^s est :

$$U^{\tau(\rho^s)} \leq U^s + n_s \frac{C_o^s}{T^s} \leq 1$$

Nous utiliserons ultérieurement cette condition pour ordonnancer nos VMs.

Harmonicité

L'objectif de nos travaux est de trouver les paramètres des VMs du niveau 1 qui permettent de respecter les contraintes des tâches qui leur sont assignées dans le niveau 2. Rappelons que les paramètres des tâches doivent être connus.

Dans un premier temps, et pour simplifier le problème, nos travaux traitent de VMs dites "harmoniques" (nous avons étendu la propriété de tâches harmoniques définies dans [31] et [75]). Des VMs sont dites harmoniques lorsque toutes leurs périodes sont multiples deux à deux. Plus formellement, les VMs d'un ensemble ρ de VMs sont dites harmoniques si et seulement si :

$$\forall \rho^i, \rho^j \in \{\rho\}^2, \left(T^i \bmod T^j = 0\right) \vee \left(T^j \bmod T^i = 0\right)$$

Calcul Pire temps de réponse d'une tâche

Nous définissons ici le calcul pire temps d'exécution d'une tâche dans notre modèle hiérarchique à niveaux. En effet, ce calcul sera utilisé pour vérifier que nos systèmes sont ordonnançables par nos algorithmes d'ordonnancement présentés plus loin. Nous considérons ici un ordonnancement arbitraire des VMs ainsi que des tâches sporadiques exécutées dans chaque VM.

Nous avons montré que le temps de réponse pire cas d'une tâche τ_i dans la VM ρ^s avec comme ensemble de scénarios de candidats d'instant critique ρ_S^s est ainsi donné par :

$$r_i = \max_{\forall \rho_{S_k}^s \in \rho_S^s} (r_{i,k})$$

Où $r_{i,k}$ correspond au temps de réponse pire cas de la tâche τ_i pour le candidat d'instant critique $\rho_{S_k}^s$ obtenu par :

$$r_{i,k} = \max_{\forall q \in \{0,\ldots,Q\}} (w_{i,q,k} - q \times T_i)$$

$w_{i,q,k}$ est le temps de réponse de la tâche τ_i de période T_i, k est le $k^{ème}$ scénario de candidat d'instant critique de la VM, q correspond au numéro de l'instance de tâche activée en $q \times T_i$, Q est le nombre maximal d'instances de tâches à considérer tel que :

$$w_{i,Q,k} \leq (Q+1) \times T_i$$

Le calcul de $w_{i,q}$ est basé sur le calcul de la suite $w_{i,q}^{m+1}$ dont nous avons prouvé la convergence dans nos travaux et donné les conditions d'initialisation. Cette valeur

est obtenue quand $w^{m+1}_{i,q,k} = w^m_{i,q,k}$ avec :

$$w^{m+1}_{i,q,k} = (q+1) \times C_i + \sum_{j \in hp(i)} \left(\left\lceil \frac{w^m_{i,q,k}}{T_j} \right\rceil \times C_j \right) + \sum_{\psi_l \in \Psi^s_{S_k}} \left(\left\lceil \frac{w^m_{i,q,k} - S_l}{T^s} \right\rceil \times (I_l + O^S) \right)$$

- C_i, C_j et T_j correspondent aux durées pire cas des tâches et à l'inter-arrivée minimale,
- C^S, T^s et O^S correspondent à la durée, la période et l'overhead de la VM, I_l est le temps d'inactivité de l'intervalle d'exécution observé c'est à dire $T^S - C^S$,
- $hp(i)$ est une fonction qui permet de trouver toutes les tâches de priorité supérieure ou égale à la tâche τ_i dans le jeu de tâches τ.
- $\Psi^S_{S_n}$ est l'ensemble des différents intervalles d'exécution de la VM $\rho^S(C^S, T^S)$ pour un scénario de candidat d'instant critique donné $\rho^S_{S_k}$.
- S_l correspond au début de l'intervalle d'exécution (qui peut être vu comme un offset).

Le pire temps de réponse pour la tâche τ_i est ainsi obtenu en prenant le maximum des temps de réponse obtenu pour toutes les instances de τ_i activées en $q \times T_i$ avec $q = 0, \ldots, Q$ tel que $w_{i,Q}$ corresponde à la fin de la période active de niveau i.

La Figure 7.5 illustre le calcul du pire temps en fonction de chaque candidat d'instant critique. Dans cette figure, nous avons d'abord défini un ordonnancement arbitraire pour la VM ρ^1 (ligne 1). À partir de cet ordonnancement, nous avons d'abord ordonnancé deux tâches $\tau_1(4, 20, 20)$ et $\tau_2(2, 20, 20)$ selon RM. La Ligne 2, indique l'ordonnancement des tâches sans prendre en compte les candidats d'instant critique, on a alors le temps de réponse de $\tau_2 = 12$ unités de temps. Les trois lignes suivantes montrent les trois candidats d'instant critique $\rho^1_{C_1}$, $\rho^1_{C_2}$ et $\rho^1_{C_3}$ pour lesquels nous obtenons respectivement les pires temps de réponse 15, 17 et 14.

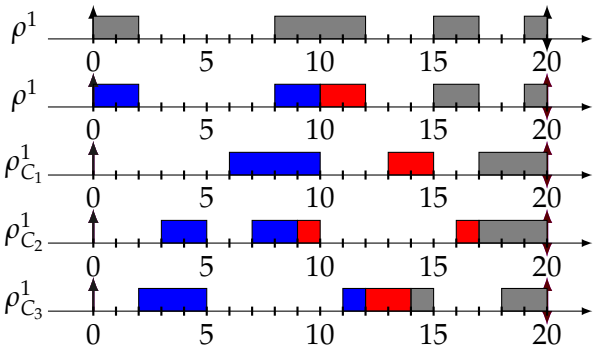


FIGURE 7.5 : 3 candidats d'instant critique pour une VM ρ^1 (gris) exécutant 2 tâches : $\tau 1$ (bleu) et $\tau 2$ (rouge)

7.1.4 Conditions d'ordonnançabilité

Les algorithmes d'ordonnancement que nous proposons plus loin (7.1.5) reposent sur des tests d'ordonnançabilité que nous avons développés et que nous présentons maintenant.

Une première condition nécessaire consiste à vérifier que l'utilisation totale des VM est inférieure à 1 (la charge processeur doit être inférieure à 100%) :

$$\sum_{\forall \rho^s \in \rho} U^s \leq 1$$

avec ρ l'ensemble des VM du système et U^s correspondant à l'utilisation totale des tâches de la VM ρ^s. Cette utilisation est obtenue par :

$$U^s = \sum_{\forall \tau_i \in \tau} \frac{C_i}{T_i} \tag{7.1}$$

où τ définit l'ensemble des tâches de la VM ρ^s étudiée.

Pour chaque VM, nous définissons des bornes d'exécutions minimale et maximale sur la période des VM. La borne minimale de la VM ρ^s correspond au maximum parmi 3 bornes :

$$T^s_{min} \geq \max(\lceil \frac{O^s}{1-U^s} \rceil, \lceil \frac{O^s}{\alpha} \rceil, v) \tag{7.2}$$

avec U^S, présenté plus haut, qui correspond au temps d'utilisation de la VM, O^s l'overhead dû au changement de contexte de la VM, v est le nombre de VM du système, $0 \leq \alpha \leq 1$ permet de borner le pourcentage de temps que l'overhead peut utiliser par rapport à la période de la VM.

Nous avons également montré que la borne maximale sur la période pouvait s'écrire : $\forall \tau_i \in \tau, T^s_{max} \leq \left\lfloor \frac{D_i - C_i}{(1-U^s)} \right\rfloor$ avec D_i l'échéance relative des tâches.

Conditions pour des VMs harmoniques

Nous avons montré que l'ensemble ρ^s des VMs harmoniques est ordonnançable si et seulement si :

$$\forall \rho^s \in \{\rho \setminus \{\rho^1\}\}, U^s \times T^s + O^s \leq T^1 - C^1$$

avec C_i est le WCET d'une VM ρ^s, $U^s \times T^s$ étant la période d'activation de la VM ρ^s pendant lequel une VM doit être exécutée pour ordonnancer les tâches de la VM avec $T^i > T^{i-1}$. Enfin, O^s est l'overhead induit par le changement de VM.

7.1.5 Ordonnancement hiérarchique de VMs harmoniques

Connaissant les paramètres des tâches et à partir des bornes minimales et maximales des VMs que nous avons pu établir dans la section 7.1.4, il est possible de calculer le temps de réponse des tâches dans les VMs grâce aux calculs de pire temps exposés dans la section 7.1.3. A partir de ces résultats, il est ensuite possible de calculer les paramètres de chaque VM en effectuant une recherche dichotomique pour trouver la plus petite période possible pour la première VM. Le calcul des paramètres des autres VMs s'effectue de manière itérative en testant exhaustivement pour chaque VM les différents paramètres possibles jusqu'à satisfaire les conditions d'ordonnançabilité. Une fois que l'ensemble des paramètres des VMs a été déterminé, nous avons développé un algorithme qui construit un ordonnancement de ces VMs.

Voici un exemple basé sur 3 VMs exécutant chacune un jeu de tâches données dans le tableau 7.1. Nous fixons l'utilisation des VM à $U^1 = \frac{1}{20} = 0.05$ pour ρ^1,

VMs	Tâche τ_i	C_i	T_i	D_i
ρ^1	τ_1	1	20	15
ρ^2	τ_2	1	15	20
ρ^2	τ_3	2	15	30
ρ^3	τ_4	2	20	40
ρ^3	τ_5	1	20	40
ρ^3	τ_6	1	40	40

TABLE 7.1 : 3 VMs et 6 tâches

$U^2 = \frac{1}{15} + \frac{2}{15} = 0.2$ pour ρ^2 et $U^3 = \frac{2}{20} + \frac{1}{20} + \frac{1}{40} = 0.175$ pour ρ^3. Nous fixons également l'overhead des VMs à 1 et le facteur α à 0.1. Pour chaque VM, nos algorithmes produisent les paramètres : $\rho^1(2, 10, 1)$, $\rho^2(5, 20, 1)$ et $\rho^3(8, 40, 1)$. L'ordonnancement produit pour ce système est donné figure 7.6.

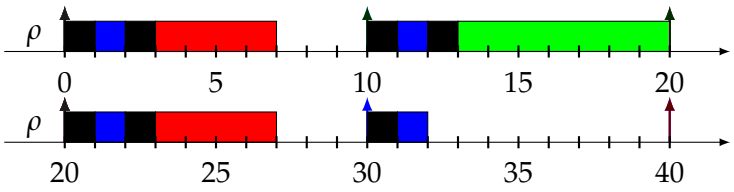


FIGURE 7.6 : Ordonnancement obtenu pour les trois VMs

7.1.6 Améliorations

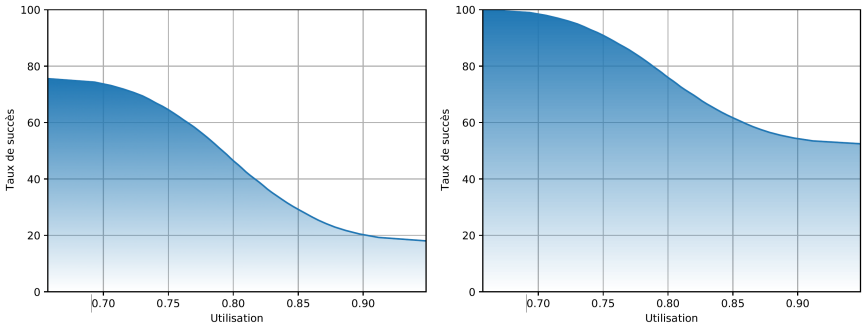
Ordonnancement niveau 1 : P-fair

L'algorithme harmonique présenté précédemment fonctionne bien pour des systèmes à 3 VMs mais pas au delà (son taux de succès s'effondre), la contrainte harmonique est trop restrictive. C'est pourquoi nous proposons d'autres algorithmes d'ordonnancement basés sur les algorithmes d'ordonnancement proportionnel (ou P-fair) dont la particularité est de répartir équitablement les ressources processeur entre les instances actives des tâches à ordonnancer.

Ordonnancement niveau 2 : Dual priority

Dans notre modèle hiérarchique à deux niveaux présenté jusqu'ici, nous nous basons sur un ordonnancement RM des taches du niveau 2. Pour améliorer encore le taux de succès de nos algorithmes, nous avons remplacé l'algorithme RM par un algorithme "Dual Priority" que nous avons adapté à notre modèle hiérarchique. En effet, dual priority permet d'aller plus loin que la simple priorité fixe de RM et possède des propriétés proches du très classique Earliest Deadline First (EDF). Ainsi, deux priorités sont utilisées pour chaque tâche, et une échéance intermédiaire est définie pour savoir quand passer d'une bande de priorité à l'autre. L'ordonnancement dual priority est ainsi utilisé comme alternative à RM lorsque RM n'a pas pu aboutir à un résultat satisfaisant.

La Figure 7.7 présente un exemple de résultats obtenus par simulation en générant des systèmes à 3 VMs et plusieurs jeux de tâches générés aléatoirement (nous expliquons ce processus de génération dans la section suivante). Nous observons un taux de succès bien plus élevé avec dual priority (à droite) par rapport à un ordonnancement à priorité fixe (à gauche).



(A) Niveau 2 : Fixed Priority (B) Niveau 2 : Dual Priority

FIGURE 7.7 : Comparaison du taux de succès en fonction de l'utilisation entre un ordonnancement niveau 2 à priorité fixe (A) et dual priority (B), avec un algorithme harmonique seul pour le niveau 1. Exemple pour 3 VMs

7.1.7 Validation, expérimentations

Pour valider nos travaux nous avons implanté tous ces algorithmes en Java. Pour disposer d'un nombre de tests significatif (4300 systèmes) nous avons également développé un générateur aléatoire de tâches basé sur une extension de UUnifast[26] (un algorithme de génération d'ordonnancements uniformes). Nous avons également couplé notre outil à une base de données pour sauvegarder les résultats et produire les courbes présentées précédemment.

Dans le cadre du projet CEOS (présenté dans la section suivante) auquel nous participions en parallèle, nous avons également développé un outil de configuration automatique de l'hyperviseur PikeOS.

7.2 Criticité mixte pour les systèmes embarqués critiques communicants

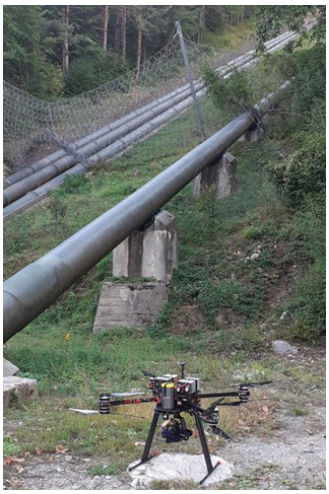
- Projet FUI[a] CEOS : Comportement et Evaluation des Ouvrages Spéciaux
- Coordinateur projet : Stéphane Menoret THALES COMMUNICATIONS & SECURITY S.A.S
- Coordinateur ESIEE : Laurent George puis Thierry Grandpierre
- Partenaires : ADCIS, Aeroport de Caen, EDF R&D Chatou, ERDF, ESIEE, INRIA, Real-Time At Work, , THALES Communications & Security, Université de Lorraine
- Publications : livrables D2.4, D3.2b, D5.3 et rapport de fin de programme BPI[b] pour l'ESIEE.
- *Mai 2017 à mai 2021*

[a] Fonds Unique Interministériel
[b] Banque Publique d'Investissement, financeur du projet

7.2.1 Description

L'objectif de ce projet était de concevoir un drone volant autonome capable d'inspecter trois types d'ouvrages civils pour y détecter automatiquement d'éventuelles

(A) Isolateur (ENEDIS)



(B) Conduite d'eau (EDF)

(C) Clôture (Aéroport de Caen)

anomalies. Pour EDF il s'agit d'inspecter l'état des conduites d'eau, c'est à dire de gros tuyaux parcourant plusieurs kilomètres dans des montagnes, des vallées etc. Pour ENEDIS il s'agit d'inspecter les lignes électriques haute tension et en particulier l'état des isolateurs en porcelaine. Enfin pour Aéroport de Caen, il s'agit de détecter d'éventuels trous dans les grillages de clôture de l'aéroport. Pour cela le projet a été découpé en 7 parties :

1. Châssis du drone, motorisation : Alerion,
2. Communications air-sol : ESIEE, Thales,
3. Développement et Intégration des systèmes sol et volant : ESIEE, Thales,
4. Sûreté de fonctionnement : ESIEE, INRIA, RTaW, Thales
5. Algorithme et interface de planification de trajectoire : université de Lorraine,
6. Détection d'anomalie par vision : ADCIS
7. Réglementation : Thales, RTaW.

Le cadre général du projet concerne le problème de la sûreté de fonctionnement des systèmes embarqués temps réels communicants, aussi appelés systèmes cyber-physiques, qui sont au cœur de l'Internet des Objets. Le défi est de garantir le fonctionnement et la réactivité des fonctions applicatives les plus critiques sur une plate-forme COTS[5] dont les ressources sont partagées et la performance en terme de temps d'exécution des fonctions peu prédictible. Notre approche s'appuie sur

[5] "Commercial Off-The-Shelf", c'est à dire vendu sur étagère, disponible sur le marché.

une solution hybride innovante multi-critique et criticité mixte réalisant une adaptation dynamique du fonctionnement par limitation des accès aux ressources et changement d'algorithmes sur un socle COTS et un RTOS[6] certifié supportant les trois fonctions principales développées dans le projet. La solution CEOS proposée s'appuie sur l'état de l'art le plus récent et correspond à un compromis raisonnable par rapport à des solutions très coûteuses nécessitant de certifier toute fonction critique (e.g. avionique) ou de sur-dimensionner les plate-formes COTS.

J'ai personnellement été impliqué dans les trois missions transversales (2,3 et 4) relevant en partie de l'ESIEE, accompagné de notre regretté collègue Laurent George (item n°3 et 4) et de notre doctorant Tristan Fautrel (item n°4). J'ai ainsi participé à la rédaction de nombreux livrables du projet ainsi qu'à la rédaction du rapport de fin de programme pour la BPI.

7.2.2 Architecture fonctionnelle

Le système CEOS se compose d'une partie volante, le drone, et d'une station sol pour contrôler ce drone à distance sur plusieurs kilomètres.

L'architecture de calcul embarquée doit donc assurer plusieurs fonctions : asservissement et stabilisation du drone, contrôle de trajectoires, communication bidirectionnelle avec le sol, envoi d'images vers le sol, traitement d'images embarquées pour la détection d'anomalies et enfin contrôle de la sécurité du vol pour déclencher automatiquement des procédures d'urgence (atterrissage du drone, déclenchement du parachute de secours).

Les communications entre le drone et le sol sont assurées par deux modes de communication distincts afin de les rendre plus tolérantes aux éventuelles perturbations : une communication par LTE[7] Thales, et une communication par radio logicielle reconfigurable (ESIEE).

La figure 7.9 présente un schéma récapitulatif de ces fonctionnalités et de leurs interactions que nous avons défini au cours de ce projet, elle fait partie du livrable D2.4 "Spécification Système". Nous pouvons y voir les fonctionnalités à implémenter dans le drone pour répondre au cahier des charges, c'est à dire principalement :

- Le Contrôle-commande du drone (asservissement bas niveau, stabilité, commandes de vols, capteur attitude du drone),
- La navigation, en lien avec la plateforme de contrôle-commande par bus MAVLink[8]. La plate-forme de contrôle-commande assure : l'autopilotage (c'est à dire la gestion, planification de la trajectoire passage par des waypoints GPS), le recalage de position par vision et le Géofencing (limitation de l'espace d'évolution).
- Les communications avec la station sol par deux moyens radio : SDR (Soft Defined Radio) reparamétrable et LTE par opérateur classique. Les données à transmettre étant :
  - la télémétrie (paramètres vitaux du drone : altitude, état des batteries, etc.),
  - la planification de missions,

[6] Real-Time Operating System - Système d'Exploitation Temps Réel

[7] LTE (Long Term Evolution) est une évolution des normes de téléphonie mobile qui correspond à la 3G et la 4G

[8] un protocole léger de communication pour les drones

  - le retour vidéo des caméras,
  - le contrôle de trajectoire en cas de nécessité.

- L'analyse d'images par IA en vol pour pilotage/asservissement trajectoire du drone sur la cible à suivre.

- Le diagnostic temps réel et la gestion des défaillances.

7.2.3 Architecture Logiciel - Criticité

Le développement des applications s'effectue par des équipes distinctes avec des contraintes spécifiques (processeurs, mémoire, communication, latence, etc.) en utilisant potentiellement des systèmes d'exploitation différents. De plus dans un soucis d'embarquabilité il faut minimiser le poids, la consommation et donc mutualiser les ressources matérielles.

Le recourt à la virtualisation est une bonne solution car elle permet d'exécuter plusieurs systèmes d'exploitation virtuels (invités) sur une seule machine réelle (physique) dite hôte. Le logiciel qui permet et contrôle la virtualisation est appelé hyperviseur. Il existe deux types de virtualisation ; la virtualisation logicielle et la virtualisation matérielle. La virtualisation logicielle ne repose pas sur un jeu d'instruction particulier du processeur, mais pour fonctionner il faut que le système d'exploitation invité soit modifié. Chaque appel au matériel est alors intercepté et transmis à la machine virtuelle. Ce système est le plus ancien et le moins performant. De plus son utilisation est complexe car il demande à modifier le système d'exploitation qui est accueilli. Au contraire, la virtualisation matérielle repose sur une unité et un jeu d'instructions spécifiques du processeur. Grâce à cela, le système invité n'a pas à être modifié, il ne « sait » pas qu'il est exécuté par une machine virtuelle. Dans ce cas l'impact sur les performances d'exécution sont minimisées. Dans le cadre de CEOS, étant donné que nous avons besoin de performances, et que nous ne souhaitons pas modifier les systèmes d'exploitations invités nous avons uniquement visé la virtualisation matérielle. Pour cela, nous avons identifié plusieurs solutions :

- QEMU, VirtualBox, VMware - Ce sont des machines virtuelles complètes destinées aux puissants PCs de bureau et aux serveurs plutôt qu'aux systèmes embarqués. Ils émulent en effet la plupart des périphériques de ces machines.

- XEN - Un hyperviseur de machines virtuelles, libre et intégré dans de nombreuses distributions Linux (RedHat, Fedora, SuSE, Mandriva, Ubuntu etc.). Il est plutôt destiné aux environnement serveurs.

- KVM - C'est l'hyperviseur implanté dans les noyaux Linux, il est basé sur le code de QEMU.

- PikeOS - PikeOS est développé par la société Sysgo, c'est un système d'exploitation temps réel qui intègre un hyperviseur capable d'exécuter des systèmes d'exploitation invités dans des partitions séparées. C'est un des seuls hyperviseurs certifiés automobile, ferroviaire et au plus haut niveau (norme ferroviaire EN 50657 SIL 4). Il permet de garantir une séparation sécurisée des machines virtuelles tout en garantissant des performances temporelles strictes. Il permet donc d'adresser le problème de criticité mixte qui nous intéresse.

Dans le cadre de CEOS, c'est donc l'hyperviseur PikeOS qui a été retenu puisqu'il est certifié d'une part et disponible sur les architectures qui nous intéressent

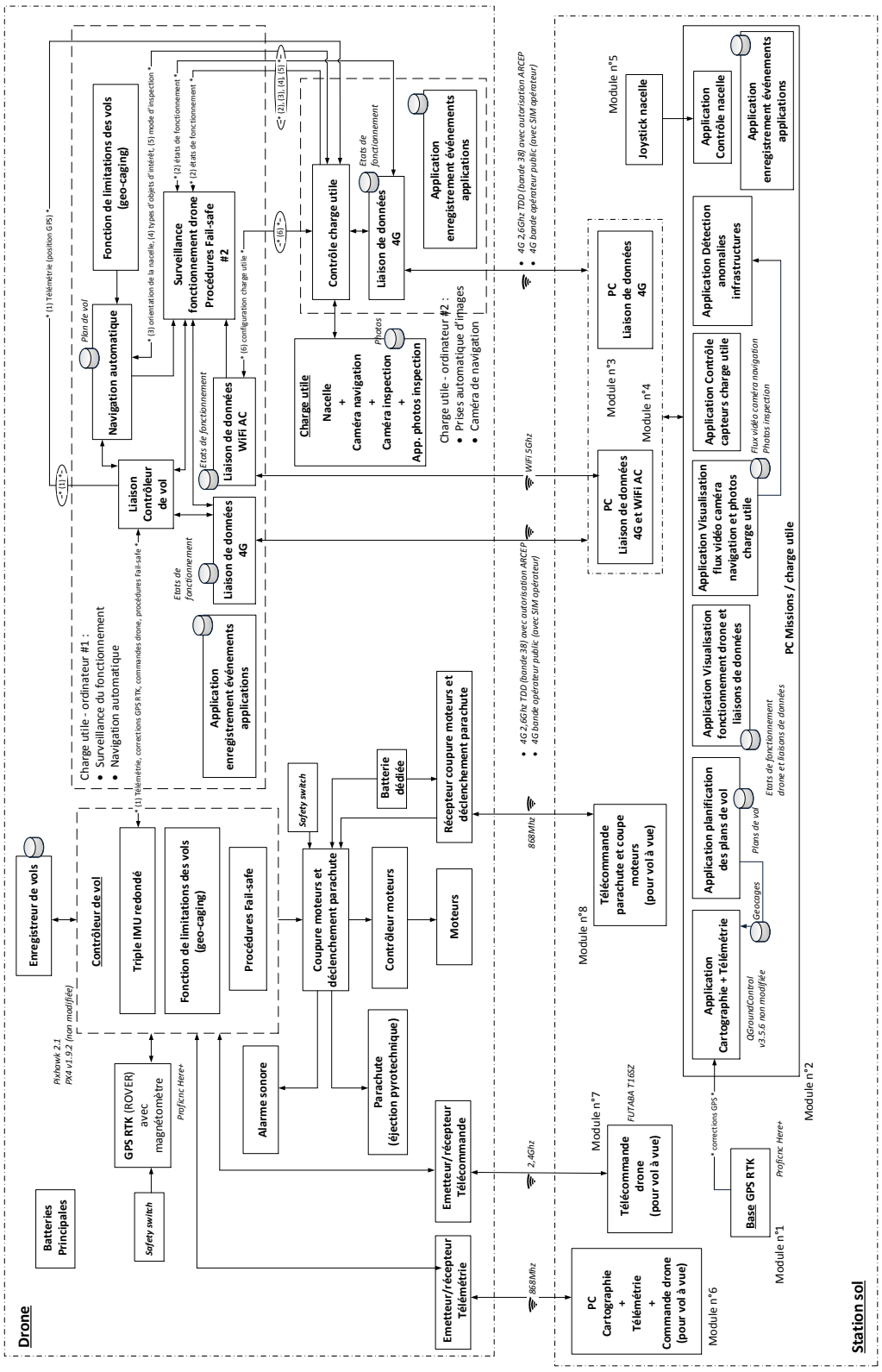


FIGURE 7.9 : Architecture fonctionnelle du système CEOS

d'autre part. PikeOS permet une isolation à la fois spatiale et temporelle des machines virtuelles qu'il héberge. L'isolation spatiale consiste à affecter une machine virtuelle à un sous-ensemble de coeurs d'une architecture multi-coeurs. L'isolation temporelle repose sur un motif temporel pré-calculé que l'on fournit à PikeOS. Ce motif permet d'indiquer des périodes d'activation et de désactivation des machines virtuelles. Cette double isolation repose sur un fichier de configuration qu'il faut donc créer en fonction des contraintes applicatives.

Notre système virtualisé est donc composé d'applications hébergées par des machines virtuelles. Cela peut être considéré comme un système hiérarchique à deux niveaux comme nous l'avons étudié dans le cadre de la thèse de Tristan Fautrel (Cf. page 7.1). Dans ce contexte, le niveau 1 ordonnance les VM gérées par l'hyperviseur. Au niveau 2, chaque VM exécute un ensemble de tâches qui composent l'application. Les tâches sont ordonnancées par le système d'exploitation spécifique de chaque VM. Une application est exécutée dans des slots temporels affectés à sa VM dont les périodes d'activités sont gérées par l'hyperviseur. Les VM sont exécutées selon un motif fixe de slots temporels. Ce motif est répété périodiquement et détermine les périodes d'activités d'une VM. Le dimensionnement des slots temporels d'une VM peut se faire soit à partir du taux d'utilisation de la VM soit à partir de la liste des tâches à exécuter dans la VM caractérisant l'utilisation minimale de la VM. Dans le second cas, nous traitons les configurations CEOS où les tâches composant une application sont exécutées par un système d'exploitation préemptif selon un algorithme à priorité fixe de type Fixed-Task-Priority (FTP). Notre hypothèse est alors que seuls les paramètres des tâches au niveau 2 sont connus. Au niveau 1, les paramètres des VM (motif des slots et période d'activation du motif de la VM) ne sont pas connus au départ et doivent être calculés pour configurer PikeOS.

Le problème à résoudre correspond exactement au problème traité dans la thèse de T. Fautrel. Pendant celle-ci, Tristan à participé au projet CEOS et développé un outil de configuration pour PikeOS. Cet outil codé en Java repose sur les algorithmes mis au point dans la thèse et permet de produire le fichier de configuration de PikeOS à partir des spécifications des applications qui s'exécutent dans les machines virtuelles. Cet outil non graphique reçoit un fichier de description Json[9] en entrée, et produit un fichier de configuration PikeOS en sortie. La documentation détaillée et l'outil ont été fournis dans le livrable "D3.2b - Ordonnancement de systèmes hypervisés" du projet CEOS.

7.2.4 Architecture Matérielle

A partir des spécifications effectuées en collaboration avec tous les partenaires, il s'agit ensuite de définir une architecture matérielle compatible. Les principales difficultés étant de répondre au besoin fort en terme de puissance de calcul, mais aussi construire cette architecture à partir d'éléments immédiatement disponibles sur le marché. Nous avons ainsi exploré plusieurs possibilités d'architectures car il s'est avéré que beaucoup de composants aux performances élevées ne sont disponibles qu'en très petite quantité ou ne peuvent pas être utilisées hors des champs d'applications visées par les constructeurs.

L'ensemble de ce travail est restitué dans le livrable D5.3 "Rapport de spécification d'architecture des applications : plates-formes embarquées et sol, déploiement des applications CEOS" que j'ai rédigé. Il contient une synthèse des spécifications pour chaque applicatif qui doit être exécuté sur les systèmes embarqués en vol et au

[9] Le format JSON est un méta format de fichier java, qui permet de décrire facilement des structures de données

Cartes	CPU modele	CPU Type	CPU Nb. Coeurs	CPU famille	Support KVM	Supp. Sysgo[16]	Virt. hardware	OS, outils dispo.
SABRE LITE SDB	i.MX6Q (NXP)	Cortex A9	4	ARM v7		100%	non	Linaro
TEGRA TX1	TX1 Nvidia	Cortex A57 +Denver	4	ARM v8	Oui	En cours ?	Oui	Ubuntu
TEGRA TX2	TX2 Nvidia	Cortex A57 +Denver	4 (+2)	ARM v8	oui	Non ?	Oui	Ubuntu
R-Car H3 Starter Kit	R-Car H3 Renesas	A57 A53	4+4	ARM v8	oui	oui	oui	Yocto
Newport GW6304	Cavium Octeon TX	CN8xxx **custom**	4 (1 – 24)	ARM v8		non	oui	
Miriac SBC LS1046A	QorIQ LS1046A NxP	Cortex A72	4	ARM v8	oui	oui	non	Microware OS9
MCIMX8M-EVK	i.MX8MQuad	Cortex A53 M4F	4 1	ARM v8	?	non	oui	

FIGURE 7.10 : Extrait du livrable D3.2b CEOS : comparatif processeurs compatibles CEOS

sol, mais également l'étude des différentes architectures matérielles disponibles, suivie de la définition du système proposé et retenu. Différents matrices comparatives ont ainsi été créées, les figures 7.10 et 7.11 en sont des exemples.

A noter que nous recherchions une architecture matérielle supportée par Sysgo (c'est à dire avec un BSP[10] PikeOs disponible) et supportant la virtualisation matérielle. Pendant le projet, les seules architectures avec support matériel de la virtualisation disponibles étaient les architectures à base de cœurs ARM v8 et les processeurs Intel.

Suite à cette étude, la solution proposée puis retenue repose sur 3 cartes interconnectées par trois différentes interfaces :

- une carte PX4[11] qui prend en charge la gestion de l'autopilote (asservissement bas niveau du drone, gestion des moteurs, accéléromètres et gyroscopes),
- une carte Miriac LS1046 pour exécuter :
  - l'application de contrôle - diagnostic et gestion des défaillances (c'est elle qui peut par exemple déclencher l'ouverture d'un parachute de secours),
  - l'application de communication air-sol par SDR (Software Defined Radio),
  - l'application de navigation autonome,
  - la gestion des échanges entre les cartes, du routages des paquets entre les cartes et le sol, et la gestion de la qualité de service (pour pouvoir passer en mode dégradé en cas de défaillance d'un des moyens de communication sol).
- une carte NVIDIA Tegra TX2 incluant un GPU pour exécuter :
  - les captures d'images issues de l'appareil photo numérique embarqué,
  - la commande du berceau de l'appareil photo numérique (pour viser),

[10] Board Support Package
[11] dont le code a été rendu plus robuste par l'INRIA

Cartes	CPU modele	USB 2	USB 3	PCIe	GPIO	I2C	UART	eth	SATA	CAN	Carte SD	SIM
SABRE LITE SDB	i.MX6Q (NXP)		0	1		oui	oui	<500Mb/s	1	1	2	1
TEGRA TX1	TX1 Nvidia		1	1				1 Gb/s	1	1	1	
TEGRA TX2	TX2 Nvidia		1	1				1 Gb/s	1	0	1	
R-Car H3 Starter Kit	R-Car H3 Renesas			1					1	1	2 (dont 1 sur carte KF)	1
Newport GW6304	Cavium Octeon TX	2 ou PCIe	2 + (1 ou PCIe)	1 à 3	oui	oui	2	2	1 à 6	1	1	
Miriac SBC LS1046A	Qoriq LS1046A NxP	Meme cnx que USB3	2	2+ [1]	12	2	1	4 (1 Gb/s)	1		1	
WB10-AT iMX8M-AT SOM	i.MX8MQuad	1	1	1	oui	3	3	1 Gb/s			1	

Table 3 - 1.1.1. Matrice de comparaison de la connectivité des cartes

FIGURE 7.11 : Extrait du livrable D3.2b CEOS : comparatif connectivité cartes disponibles CEOS

- le streaming vidéo d'une caméra de navigation,
- l'analyse temps réel par IA des images de pylônes, grillages ou conduites d'eau forcées,
- la gestion des communications LTE 4G opérateur.

Un schéma simplifié de cette architecture est disponible figure 7.12. Nous y trouvons respectivement les 3 cartes (rectangles bleus), les interfaces de chacune de ces cartes vers différents capteurs (cercles oranges) et enfin le placement des différentes applications (cercles verts) sur chacune de ces cartes.

7.2.5 Communications Air-Sol par SDR

Les communications avec le sol sont effectuées par deux liens radio indépendants et gérés par deux partenaires différents. Thales est en charge de la communication par sa LTE, l'ESIEE se charge d'une des deux communication par SDR (Software Redefined Radio). C'est Dereje Molla, doctorant co-encadré par Laurent George et Hakim Badis, qui était en charge de cette partie. J'ai également contribué à ces travaux, notamment dans l'optimisation de l'implémentation temps réel sur architecture ARMv8 et plus précisément dans l'utilisation des instructions SIMD "Neon" pour accélérer les calculs.

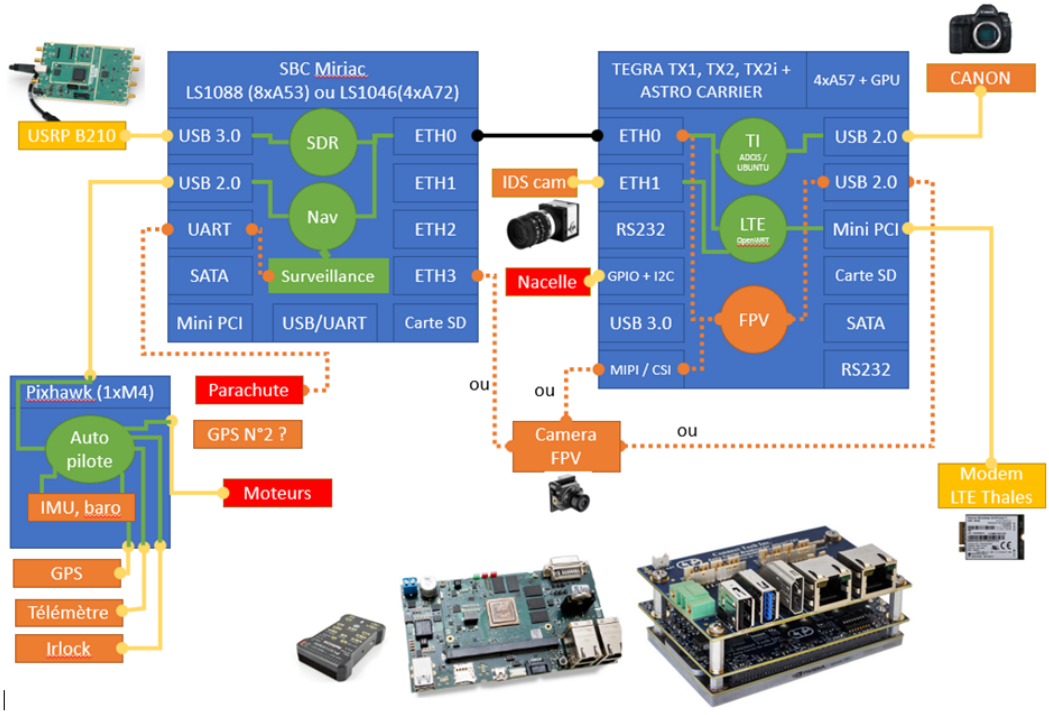


FIGURE 7.12 : Architecture matérielle embarquée CEOS

7.3 Optimisations des communications temps réel, configuration d'un réseau avionique

- *Thèse de Florient Champenois : Configuration et analyse temporelle de réseaux avioniques à sauts multiples*
- *Direction : Laurent George (Pr. UGE) puis Etienne Borde (Pr. Télécom Paris), puis Jean-Louis Rougier (Pr. Télécom Paris),*
- *Co-encadrement : Thierry Grandpierre (32%) et Florian Brandner (33%, McF Télécom Paris)*
- *Thèse en cours depuis le 02/11/2020*
- *Financement CIFRE, société SAFRAN*
- *École Doctorale : IP Paris - Ecole Doctorale de l'Institut Polytechnique de Paris n°626*
- *Publications : un brevet (soumis), un article (en attente d'autorisation de soumission), une communication GDR SOC2,*

Cette thèse CIFRE entre SAFRAN, Télécom Paris et ESIEE a été initiée par Laurent George (directeur de thèse initial). Suite au décès de Laurent George, Etienne Borde en a repris la direction, mais suite à son départ de Télécom Paris, c'est Jean-Louis Rougier qui le remplace maintenant. De mon côté je co-encadre cette thèse depuis son commencement avec Florian Brandner.

7.3.1 Contextes et objectifs

En raison de la bande passante qu'ils offrent, les réseaux ethernet ont été adoptés dans la plupart des systèmes embarqués temps réel complexes, tels que les avions, les trains et même les voitures modernes. L'architecture réseau de ces systèmes repose sur la configuration de commutateurs (switchs) afin de rendre déterministe les communications entre les unités de calcul, ce qui est indispensable dans le domaine avionique. Dans ce domaine, de tels réseaux ont été normalisés dans la partie 7 du standard ARINC664, également appelée AFDX (Full-Duplex Switched Ethernet). Les réseaux Ethernet commutés sont aujourd'hui déployés dans de nombreux systèmes embarqués temps réel et ont été étudiés sous l'angle de l'analyse temporelle dans de nombreux travaux de recherche. Ces travaux reposent sur des méthodes d'analyse telles que l'approche du calcul de réseau, l'approche de la trajectoire ou l'analyse en avant du délai de bout en bout. La plupart des applications ethernet commutées reposent sur une architecture de réseau centralisée, qui offre souvent une réplication active répondant aux normes de sûreté des systèmes critiques. Ces architectures sont réputées fiables et plus faciles à analyser en termes de temps de transmission des paquets. Cependant, ce type d'architecture n'est pas suffisamment flexible pour répondre aux exigences de volume, de poids et de consommation énergétique des systèmes aéroportés légers. Cela est particulièrement vrai pour les petits systèmes, où l'équipement matériel du commutateur peut représenter une partie considérable du volume et du poids du système. Dans ce contexte, nous proposons d'étudier les réseaux commutés décentralisés à sauts multiples avec les protocoles de routage simple (par exemple, par flooding[12]) afin de réduire le volume, le poids, et la consommation énergétique des systèmes aéroportés légers.

Le passage d'une architecture centralisée à une architecture à sauts multiples pose plusieurs défis. Tout d'abord, les techniques d'analyse de synchronisation doivent être adaptées afin de prendre en compte les caractéristiques d'une architecture à sauts multiples (par exemple, lorsque des trames redondantes sont ignorées avec un protocole de commutation basé sur le flooding). Ensuite, les techniques de commutation doivent être adaptées pour maintenir la prédictibilité des temps de transmission des paquets, améliorer l'efficacité du réseau et bien sûr garantir la fiabilité du réseau.

Ces caractéristiques peuvent être par exemple obtenues, en reconfigurant de manière statique, ou même dynamique, la stratégie de commutation de paquets en cas de défaillance d'un sous-système, d'un commutateur ou d'un lien de communication.

7.3.2 Approche proposée

Nous partons d'un réseau Ethernet décentralisé simple, dans lequel les sous-systèmes terminaux fonctionnent eux-mêmes comme des commutateurs. Ceci est combiné à une stratégie de commutation simple, dans laquelle les trames entrantes d'un commutateur sont diffusées vers tous ses liens sortants (principe du flooding). Un réseau décentralisé de cette façon permet d'éliminer les équipements dédiés de commutation avec une mise en oeuvre simple offrant un niveau élevé de redondance en termes de transmission de trames. A partir de cette mise en oeuvre, nous étudions, évaluons, et adaptons les méthodes d'analyse temporelle existantes dans l'état de l'art pour de telles architectures. Ceci nous permet de mieux comprendre leurs avantages et leurs limitations.

[12] chaque paquet entrant est envoyé via chaque lien sortant sauf celui sur lequel il est arrivé.

Une telle architecture entraîne une importante utilisation de la bande passante du réseau, cela impacte les délais de transmission et devient un problème lorsque la taille du réseau est augmentée. Cependant, cette architecture de base simple est un point de départ intéressant pour les réseaux à petite échelle.

Afin de résoudre le problème de la bande passante, nous explorons d'autres protocoles de commutation, par exemple nous mesurons l'impact du remplacement de la stratégie de transmission de trame basée sur le broadcast par du multicast. Cela nous permettrait d'améliorer la précision des analyses de temps de transmission en séparant spatialement les transmissions indépendantes les unes des autres.

Nous étudions également l'impact des définitions de groupes multicast sur les performances des transmissions des trames (délais de transmission de trames de bout en bout, taux de pertes de messages acceptable). Afin de préserver la fiabilité obtenue par redondance active tout en diminuant le nombre de nœuds dans le réseau, nous envisageons une approche basée sur la criticité mixte dans laquelle les paquets seront caractérisés par leur niveau de criticité. En utilisant ces niveaux de criticité, plusieurs itinéraires seront calculés hors ligne afin de garantir les délais de transmission des trames critiques en cas d'occurrence de pannes (en nombre borné) des commutateurs et/ou des liens de communication (crash de commutateur, omission de paquets, rupture de liaisons).

7.3.3 Développements en cours

Dans cette thèse, nous explorerons des méthodologies de conception dédiées à l'exploration de topologies de réseau multi-sauts en fonction d'applications spécifiques. L'objectif principal est d'améliorer les techniques d'analyse de temps de transmission et de les coupler avec des méthodes de configuration automatisées ou semi-automatisées. L'objectif est à la fois de réduire le volume, la consommation énergétique, et le poids des systèmes aéroportés légers, mais aussi de réduire leur temps de conception. De tels procédés de configuration pourraient être utilisés pour adapter de manière statique, voire dynamique, la stratégie de commutation des paquets, par exemple en définissant des mécanismes de retransmission/récupération en présence de défaillances de composants de réseau.

Afin d'analyser le temps de transmission des paquets sur un réseau Ethernet commuté, plusieurs techniques ont été proposées dans la littérature. Les plus célèbres reposent sur le calcul de réseau et l'approche de trajectoire. Plus récemment, l'approche d'analyse en avant a été proposée. Des travaux récents dans ce domaine ont également porté sur l'intégration de protocoles de communication plus sophistiqués tels que AVB (Audio Vidéo Bridging). Cependant, peu de travaux ont été réalisés sur le sujet de l'analyse d'architectures décentralisées à sauts multiples. En effet, la plupart des systèmes existants utilisant AFDX supposent un ensemble de commutateurs centralisés avec des tables de routage prédéfinies. Tout en facilitant l'analyse et la configuration d'architectures de réseaux de grandes tailles, les architectures centralisées présentent également des inconvénients pour les architectures plus petites en termes de volume, de poids et de consommation énergétique. En effet les commutateurs de réseaux centralisés sont plutôt lourds car ils doivent fournir suffisamment de capacités de stockage et de calcul pour transmettre des quantités importantes de messages transmis par de très nombreux sous-systèmes.

Les résultats obtenus sont encourageants, nous arrivons aujourd'hui à prédire le temps de transmission pire cas, et nous avons développé un outil de génération automatique de topologies optimisées. La validation sur le simulateur hardware de l'architecture est en cours. Cependant, à ce stade, des accords de confidentialité ne

me permettent pas de donner d'avantage d'indications sur les travaux et outils développés dans le cadre de cette thèse. De plus, un brevet doit être déposé ce qui empêche toute description tant que la déclaration d'invention n'a pas été effectuée. Il en est actuellement de même pour les publications qui sont prêtes.

Chapitre 8

Implémentation et optimisation d'applications temps réel souples

Ce chapitre présente mes contributions dans le domaine temps réel souple, pour lequel il s'agit essentiellement d'accélérer les performances d'applications, de minimiser les ressources matérielles (pour l'embarquabilité) et de développer des systèmes innovants.

8.1 Application vidéo sur architecture multi-DSP

- *Thèse de Nejmeddine Bahri : Étude et conception d'un encodeur vidéo H264/AVC de résolution HD sur une plateforme multicœur [6]*
- *Direction : Mohamed Akil (Pr. UGE) et Nouri Masmoudi (Co-tutelle université de SFAX - Tunisie)*
- *Co-encadrement : Thierry Grandpierre (80%)*
- *Soutenue le 09/11/2015 - Durée : 49 mois*
- *Financement PHC-Utique et Université de Sfax*
- *École Doctorale : MSTIC - Mathématiques et Sciences et Technologies de l'Information et de la Communication n°532*
- *Publications : [7, 8, 9, 10, 11, 12]*

8.1.1 Contextes et objectifs

Les besoins en termes de qualité vidéo, résolution, débit ne cessent de croître. Ils conduisent à la création régulière de nouveaux standards de compression vidéo temps réel pour permettre de minimiser la quantité de données à transmettre et stocker. Cette compression repose sur une complexité de calcul croissante. L'implantation de ces encodeurs s'effectue généralement en plusieurs étapes : le prototype est souvent développé de façon purement logicielle sur des PCs puissants mais atteint rarement une exécution temps réel. Quand la preuve de concept est validée, les parties les plus gourmandes en calcul peuvent être portées sur FPGA et le reste sur un microprocesseur embarqué, éventuellement multi-coeurs. Enfin, quand un standard s'impose sur le marché, les grands constructeurs s'emparent de l'encodeur et l'intègrent dans des circuits intégrés dédiés (ASICs) dont le coût élevé de développement doit être amorti ensuite par la production en très grande quantité. Les

constructeurs de microprocesseurs intègrent également ces encodeurs (et décodeurs) dans leurs composants, sous forme de périphériques intégrés.

Au cours de cette thèse nous nous sommes intéressés à l'implémentation de l'encodeur vidéo "Baseline profile" de la norme H264/AVC[1] afin d'une part qu'elle supporte la compression temps réel d'images de qualité HD (1280x720p), et d'autre part qu'elle soit embarquable dans un système portable.

Étant donné la forte puissance de calcul nécessaire, nous avons abordé une implémentation sur des multi-DSP, processeurs embarqués parmi les plus puissants de l'époque. Notons que quand cette thèse a débuté, le standard H264 était en plein essor, la norme H265/HEVC est devenue accessible en fin de thèse mais à néanmoins été étudiée brièvement. Cette dernière ne s'est pas fortement implantée (pour des problèmes de licence très coûteuse), elle est aujourd'hui en voie d'être remplacée à son tour par la norme H266 ou peut être l'AV1 qui a l'énorme avantage d'être libre de droit.

8.1.2 Encodage H264/AVC HD et parallélisme

En vidéo, les images sont rarement codées en utilisant directement[2] les composantes RVB (rouge, verte et bleue) issues des capteurs d'image, mais préférablement en YUV. Y étant la luminance (une somme pondérée de composantes RVB), U la chrominance rouge et V la chrominance bleue (toutes deux obtenues également par une somme pondérée des composantes RVB). Ceci permet de séparer la luminance, pour laquelle l'oeil humain est très sensible, de la chrominance, pour laquelle l'oeil humain est moins sensible. On tire partie de cette caractéristique visuelle pour transmettre moins d'informations de chrominance que de luminance sans dégrader le confort visuel. Par exemple en YUV 4 :2 :0, très souvent utilisé, on transmet 2 pixels de chrominance (rouge et bleu) pour 4 pixels de luminance, ce qui permet déjà une première réduction de données. L'encodage vidéo H264 repose sur la détection des redondances d'information qui peuvent exister dans une image (ou "image intra"), et entre des images successives ("images inter"). Ceci permet de ne transmettre qu'une fois ces informations spatialement redondantes (pour les images intra), ou temporellement redondantes (pour les images inter) et finalement réduire le débit du flux de données binaires (bitstream) transmis. Pour effectuer cette détection, chaque image (appelée aussi "frame") est décomposée en petits blocs carrés, appelés macroblocs (MB) de 16x16 pixels maximum pour la luminance et 8x8 pour les chrominances rouge (Cr) et bleue (Br).

Pour simplifier, l'idée directrice d'une transmission vidéo compressée est de construire des séries de groupes d'images (Group of Pictures, GOP) qui commencent chacune par une image intra complète compressée (appelée aussi I-frame), puis des images appelées "P-frame" qui sont obtenues en calculant les nouvelles positions (déplacement) de chaque macrobloc de l'I-frame initiale. Chaque image de ce GOP, hormis la première est appelée P-frame (Predicted frame). Il existe d'autres types d'images en fonction des profils H264 choisis, mais uniquement ces 2 types dans le baseline profile qui nous intéresse.

Un encodeur vidéo est donc composé de 2 grandes parties : d'une part un détecteur de redondances intra-image (rectangle "intra prédiction" sur la figure 8.1) et d'autre part de détecteurs de redondance temporelle entre les images : on cherche

[1] https ://www.itu.int/rec/T-REC-H.264. Il existe en effet différents profils destinés à différentes applications : "base line" profile pour les applications mobiles et vidéoconférences, "main profile" pour les applications grand public, "high profile" pour les applications TV HD, etc.

[2] si on néglige l'étape de dématriçage (ou demosaicing).

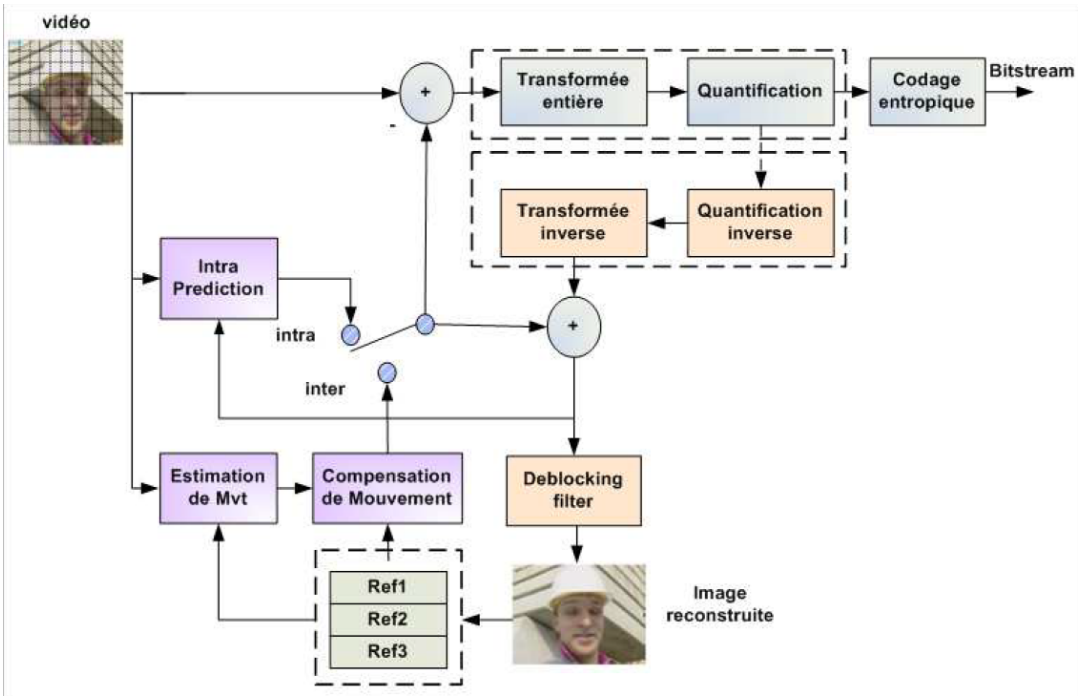


FIGURE 8.1 : Chaîne d'encodage H264/AVC

la nouvelle position de chaque macrobloc (rectangles "Estimation de mvt" et "Compensation de mouvement" sur la figure 8.1). Notons qu'on utilise le terme prédicteur pour désigner l'ensemble des fonctions qui, à partir d'une image et éventuellement de la suivante (cas inter), recherche la meilleure image compressée équivalente, en comparant ses encodages et décodages complets avec l'original. Pour cela les prédicteurs produisent différentes images en faisant varier un ensemble de paramètres (appelés modes) parmi lesquels est sélectionnée le mode qui minimise la différence avec l'image réelle. Cette différence est calculée par la somme des différences absolues (SAD) des luminances ou chrominances des pixels.

Prédiction intra

Il n'est pas question ici de décrire toutes les possibilités de prédiction intra (qui sont décrites en détail dans le paragraphe 2.2.2 de la thèse) mais de donner une idée générale des techniques d'encodage possibles pour comprendre la suite. Ainsi pour chaque macrobloc de luminance 16x16 à coder on distingue le cas où le bloc est constitué d'une texture relativement uniforme ou non. Dans le premier cas, le prédicteur essaie de construire le MB à partir de la rangée de pixels haut et gauche voisins que l'on peut propager dans le MB courant selon 4 modes de prédiction possibles. Dans le second cas, des textures non uniformes, les MB sont découpés en blocs 4x4 sur lesquels on applique 9 modes de prédictions (voir figure 8.2) pour trouver celui qui minimise la SAD. Comme pour les MB 16x16, le principe est ainsi de propager la valeur des pixels voisins hauts (pixels M,A,B,C,D) et gauches (pixels M,I,J,K,L) selon cette fois 9 possibilités (modes).

Le procédé est similaire pour le traitement des chrominances, sachant que le même mode sera appliqué pour la chrominance bleue et la chrominance rouge (le prédicteur n'est donc exécuté qu'une seule fois).

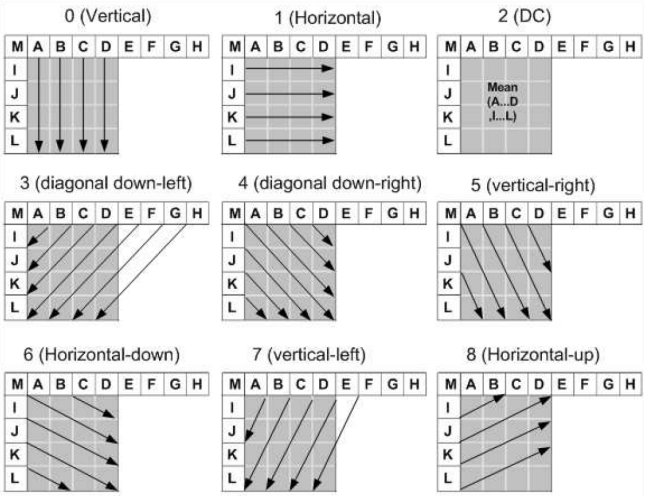


FIGURE 8.2 : 9 modes de prédiction intra pour blocs 4x4

Prédiction inter

L'inter prédiction recherche les redondances temporelles entre les images d'une séquence vidéo. Elle est donc basée sur la compensation de mouvements entre les images précédentes et l'image courante : cela consiste à rechercher des portions d'images identiques entre des images successives pour finalement ne coder que les vecteurs de mouvement de ces portions, ainsi que les différences. Ceci permet une réduction drastique de la quantité de données à transmettre.

L'image de luminance est découpée en MB dont on recherche la différence de position entre les images. Le vecteur obtenu est ensuite affecté aux MB de chrominance. Il existe plusieurs algorithmes pour réaliser ce calcul car la méthode "force brute" qui recherche exhaustivement tous les déplacements possibles est trop coûteuse en temps de calcul. De plus, pour couvrir les cas où un MB n'est pas uniforme, la norme H264 prévoit de le découper en sous-blocs correspondant à 7 modes de prédiction.

Choix de mode et encodage

Pour choisir le meilleur mode d'encodage il faut exécuter l'ensemble des prédicteurs vu précédemment et choisir celui qui induit le moins de distorsion ("Cost") calculée également par une SAD sur l'ensemble des MB. La figure 8.3 récapitule la chaîne de prédiction et présente son intégration dans la chaîne de décision.

Une fois le prédicteur choisi il reste l'encodage des MB et des déplacements. Cela passe par l'étape classique de transformation en espace fréquentiel pour séparer les basses fréquences (utiles) et hautes fréquences (moins importantes). Avec la norme H264 c'est une transformée en cosinus inverse (ICT) qui est recommandée. Elle permet de minimiser la puissance matérielle requise car elle est uniquement basée sur des additions et décalages entiers. La norme H264 repose également sur une transformée Hadamard [30] pour les coefficients constants de chaque bloc résiduel. La transformée entière est ensuite suivie d'une classique quantification qui consiste à diviser les coefficients par un pas de quantification qui permet d'éliminer les hautes fréquences, améliorant ainsi le taux de compression mais peut induire une perte de qualité. Enfin, les données à transmettre sont encodées avec un codeur entropique qui utilise des codes à longueur variables (Variable Length Coding - VLC) basés sur

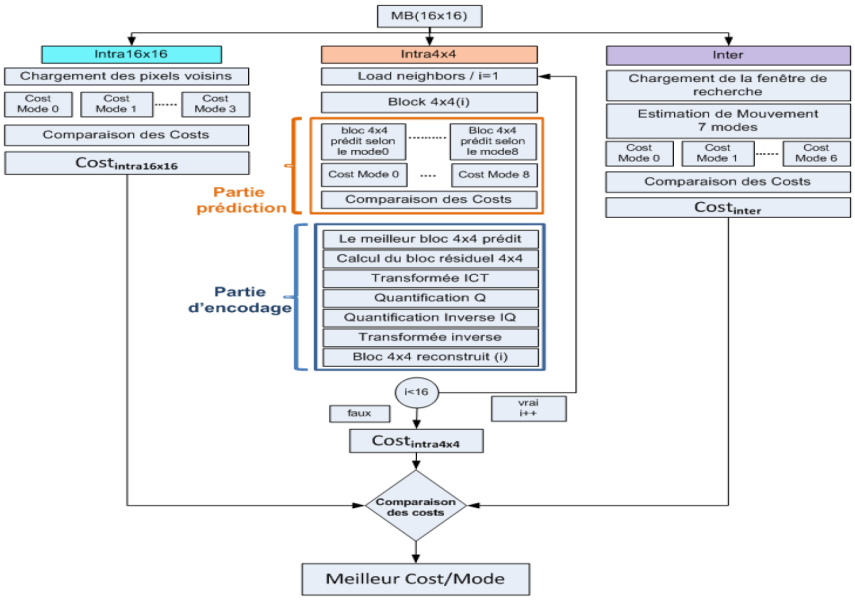


FIGURE 8.3 : Chaîne de prédiction - décision H264

le très classique codage de Huffman pour réduire la quantité de données binaires transmises.

8.1.3 Architecture DSP et Multi-DSP

L'objectif étant d'exécuter un encodeur H264 de qualité HD en temps réel sur une architecture embarquée, il est nécessaire de disposer d'une capacité de calcul élevée offrant plusieurs niveaux de parallélisme. Pendant la thèse le nombre d'architectures embarquées multi-coeurs disponibles était très faible. Après analyse des caractéristiques, nous avons alors choisi les plus puissantes (et disponibles) du moment, c'est à dire les versions 6 coeurs DSP[3] implantés dans le circuit TMS320C6472 et 8 coeurs DSP implémentés dans le circuit TMS320C6678, tout deux issus de la célèbre famille de processeurs "C64x" de Texas Instruments.

Les processeurs DSP sont en effet optimisés pour effectuer les calculs récurrents en traitement du signal (typiquement les sommes de produits) et pour effectuer rapidement des transferts de données entre ces entrées-sorties et la mémoire. Grâce à leur architecture VLIW (Very Large Instruction Word de 256 bits, présentée dans la partie gauche de la figure 8.4), ils sont capables d'exécuter jusqu'à 8 instructions 32 bits par cycle d'horloge et par coeur. La puissance crête de calcul pour le TMS320C6472 basée sur 6 coeurs à 700MHz (présenté dans la partie droite de la figure 8.4) est donc de 700x8x6=33600 MIPS pour une consommation d'environ 5 Watts (soit 0,15 mW/MIPS). Le TMS320C6678 peut quant à lui fonctionner à une fréquence comprise entre 1 et 1.25GHz pour offrir alors une puissance de calcul de 64000 MIPS.

Le TMS320C6672 dispose d'une mémoire interne (donc très rapide) partagée de 768Ko entre les coeurs (4Mo pour le C6678). Chaque coeur dispose de 608Ko de mémoire RAM (configurable en cache L2) et 32Ko de mémoire cache L1. Ces processeurs sont également dotés d'un contrôleur de transfert DMA intégré avancé (Enhanced Direct Memory Access) permettant un recouvrement presque total entre les calculs et les communications. En effet, en traitement du signal et des images temps

[3]Digital Signal Processor

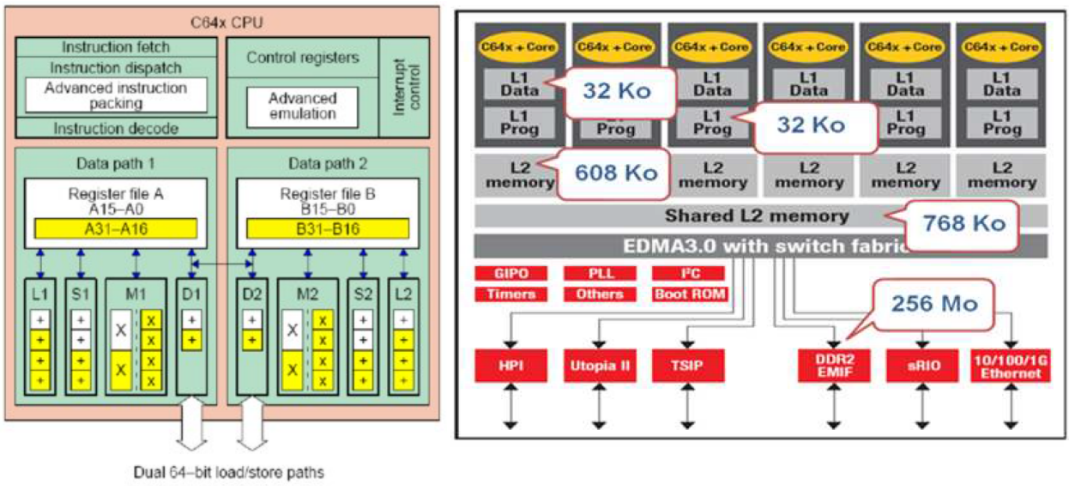


FIGURE 8.4 : Architecture VLIW d'un coeur C64x (à gauche), Architecture 6 coeurs du TMS320C6472 (à droite)

réel, il n'est pas suffisant de calculer rapidement, il faut également être capable de transférer les données tout aussi vite.

8.1.4 Implémentation optimisée

L'implantation s'est effectuée en 2 étapes : une implémentation mono-coeur, puis l'implémentation multi-coeurs.

Mono-coeur

Nous sommes partis des travaux menés au laboratoire LETI (Laboratoire d'Électronique et Technologies de l'Information de Sfax) et en particulier de la thèse de Imen Werda [47] que j'ai co-encadré de façon non officielle (elle ne s'est pas effectuée en co-tutelle). Cette thèse avait pour but l'implémentation de l'encodeur H264, basse résolution et a abouti à une implantation complète sur architecture PC x86, ainsi qu'à une implantation partielle sur mono-coeur H264. Diverses optimisations ont été appliquées pour les différents modules d'encodage tels que le filtre anti-block, la métrique de distorsion (SAD), le module d'inter-prédiction, le codeur entropique. Cependant, la version développée ne supportait pas l'ensemble de la norme et surtout est limitée à la résolution CIF (352x288).

Partant de cette implantation et pour exploiter le parallélisme potentiel de ce codeur, nous avons identifié deux types de partitionnement, le partitionnement fonctionnel (Task Level Parallelism) et le partitionnement de données (Data Level Parallélisme). Bien qu'un certain nombre d'optimisations ai été effectué sur le codage des images inter, les travaux se sont surtout focalisés sur la compression des images intra car elle représentait plus de 33% du temps total d'exécution lors du profiling du codeur sur l'architecture mono-coeur.

L'objectif étant d'exploiter efficacement la mémoire interne (rapide) du DSP, nous avons exploré différentes implémentations. Nous sommes partis d'une implémentation relativement naive, MB par MB de l'encodeur intra qui a pour inconvénient de nécessiter de nombreux et coûteux accès en mémoire externe (lente). Puis nous sommes passés à une implémentation "ligne de MB" permettant un gain de 36% en temps d'exécution puisque les accès mémoire externe y sont minimisés. Ensuite,

grâce à l'utilisation de l'EDMA pour masquer le temps d'accès à la mémoire externe, et un découpage en lignes de MB, nous avons pu gagner 12% de plus.

Pour gagner encore en performance nous avons également exploré les optimisations algorithmiques. Celles-ci consistent à réduire le nombre de modes de prédictions testés en s'appuyant sur des algorithmes existant à partir desquels nous avons proposé notre propre implémentation qui permet un gain de 30% à 54% (en fonction des images) pour la complexité de l'intra 16x16 et 4x4 au prix d'une dégradation jugée acceptable de 0.03% du PSNR (mesure de distorsion). Le débit a pu être réduit jusqu'à 1.8% .

L'ensemble des optimisations effectuées sur l'implémentation H264 mono-coeur DSP ont permis d'atteindre une vitesse d'encodage de 25 images par seconde pour des images CIF (352x288). Cette implantation supporte également l'encodage d'images SD (720x480) et HD (1280x720) mais sans atteindre le temps réel (respectivement 7 images par seconde et 2,7 images par seconde en moyenne) comme on pouvait s'y attendre.

Multi-coeur

Après l'étude de plusieurs approches de parallélisation pour atteindre le temps réel, nous utilisons le parallélisme au niveau image ("Frame Level Parallelism") avant d'explorer l'approche par groupes d'images ("GOP Level Parallelism").

La première implémentation de l'approche "Frame Level Parallelism" est basée sur un fonctionnement pipeliné où le flux d'images est distribué sur chaque coeur pour être encodé ligne par ligne de MB, avec un synchronisation inter-core de 3 lignes de MB. Cette stratégie permet d'atteindre une vitesse d'encodage de 70 images QCIF par seconde et 10 images HD par seconde sur 6 coeurs (en réalité 5 coeurs car un coeur doit être dédié au contrôle et à l'interface avec les périphériques de réception des images et de d'émission du flux compressé sur le réseau TCP/IP). Cette approche a été ré-écrite pour être améliorée en utilisant des doubles buffers afin d'éliminer des temps d'attente inutiles, permettant d'atteindre une vitesse de 99 images par seconde en QCIF, mais encore limitée à 11,7 images par seconde en HD sur l'architecture à 6 coeurs. Les mesures sur une architecture TMS320C6678 à 8 coeurs indiquent une vitesse d'encodage de tout juste 26 images HD par seconde en moyenne. Le temps réel est atteint mais la marge est insuffisante alors que cette implémentation est déjà fortement optimisée.

L'alternative est donc d'explorer une autre approche dite "GOP level Parallelism" en ré-utilisant toutes les techniques d'optimisation acquises dans les premières implémentations. Cette approche consiste à assigner à chaque CPU un GOP (de 8 images) à traiter ce qui est rendu possible par l'absence de dépendances entre les GOP. La figure 8.5 donne un aperçu macroscopique de cette implémentation. Chaque colonne représente un thread qui s'exécute soit sur le PC (qui effectue l'acquisition vidéo et le transfert vers la carte multi-DSP), soit sur l'un des 6 coeurs DSP. La vitesse moyenne d'encodage atteint alors 12,8 images par seconde sur l'architecture 6 coeurs, et 28,8 images par seconde sur la version 8 coeurs. En effet, la version 8 coeurs offre également une bande passante mémoire plus élevée vers la mémoire externe qui a doublé entre les 2 architectures.

Côté codage, il faut programmer les DMA, les contrôleurs d'interruption, les sémaphores matériels (ces DSP possèdent 32 sémaphores spécifiques), mais aussi gérer finement la cohérence des caches en les invalidant manuellement ou en forçant le rafraîchissement avant certaines lectures. Tout cela s'effectue grâce à des instructions

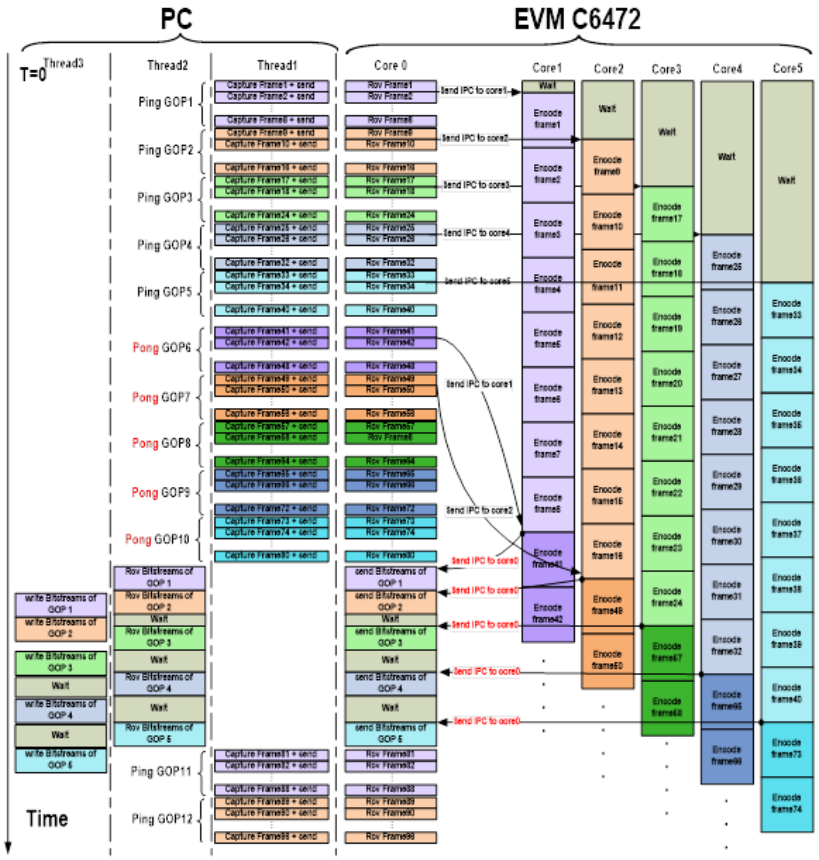


FIGURE 8.5 : Implémentation pipelinée de l'approche "GOP Level Parallelism"

disponibles dans la CSL (Chip Support Library de Texas Instrument). Il faut également configurer l'affectation des mémoires et leurs tailles au niveau de l'éditeur de lien de l'environnement graphique Code Composer Studio (CCS) de TI.

Lors de sa publication c'était l'encodeur le plus rapide en regard de sa consommation et de son embarquabilité puisque les rivaux reposaient alors sur une implémentation GPU de PC, qui sont des composants réputés gourmands en énergie.

8.2 Traitement d'images sur architecture multi-coeurs

- *Stages M2 (Erasmus) : Petr Matas*
- *Co-encadrement avec E. Dokladalova*
- *Publication : [64]*

Dans le cadre d'un stage Erasmus de Master 2, co-encadré avec Eva Dokladalova et Laurent Najman, nous avons parallélisé un algorithme de construction d'arbre des composantes connexes pour l'exécuter sur une machine multi-coeurs. L'arbre des composantes connexes est au coeur du traitement d'image. L'intensité et les relations de connectivité entre les pixels sont utilisées pour construire un arbre (graphe) permettant ensuite d'effectuer des opérations classiques de traitement d'images comme le filtrage, la segmentation ou encore la reconnaissance d'objets en coupant certaines parties de l'arbre judicieusement choisies.

Nos travaux considèrent des images en niveaux de gris à 2 dimensions et se basent sur les relations de 4-connectivité entre les pixels (donc leurs 4 voisins). La construction de l'arbre des composantes connexes de telles images consiste à :

- identifier des ensembles de pixels connectés ayant une même intensité (i.e. des "composantes connexes") : chaque ensemble de pixels correspond alors à un noeud de l'arbre des composantes connexes,
- identifier les relations d'inclusion entre ces composantes connexes : une composante est incluse dans une autre si l'intensité de ses pixels est plus petite (dans le cas des arbres "max-tree") que celle de l'autre composante, dans ce cas la seconde est dite parent de la première. Ces relations d'inclusion correspondent aux arcs de l'arbre des composantes connexes.

Dans les chaînes de traitement d'images basées sur ces arbres, la construction de l'arbre est très coûteuse en temps (près de 80% du temps total d'exécution), il est donc important de le réduire. Pour cela nous parallélisons la construction de l'arbre d'une image en 2 étapes :

1. nous construisons en parallèle les arbres de chaque ligne de l'image,
2. puis nous fusionnons deux à deux les arbres partiels obtenus.

Pour nos algorithmes nous utilisons une variante de l'arbre des composantes connexes appelée arbre de points qui a l'avantage de minimiser l'occupation mémoire. Dans celui-ci chaque sommet correspond à un point de l'image, les arcs représentent les liens de parentés entre les pixels. L'implémentation parallèle reste similaire à celle présentée précédemment, c'est à dire une étape de construction 1D des arbres partiels, suivie d'une étape de fusion d'arbres. Différentes implémentations multi-thread (en c++) sur architectures multi-coeurs ont été effectuées pour explorer différentes stratégies de distribution des fusions : par exemple une fusion par

thread, regroupement de fusions pour minimiser les transferts, etc. Sur une machine à 4 coeurs nous avons obtenu un speedup de 3.57 en utilisant autant de threads que de coeurs et une stratégie de minimisation des communications par regroupement de fusions.

Nous verrons dans le prochain chapitre, dédié aux perspectives de mes travaux, que ces résultats vont être repris dans un cadre plus large pour une implémentation GPU.

8.3 Application et optimisations sur architectures GPU

Cette section présente mes principales contributions dans le cadre d'optimisations d'applications sur architectures GPU (Graphic Processor Unit). Les processeurs de type GPU sont des processeurs initialement dédiés à l'exécution de la chaîne de rendu graphique 3D pour les jeux vidéos, la réalité virtuelle et la CAO. Dans ces chaînes de traitement, les mêmes calculs sont effectués sur de grandes quantités de données (les millions de polygones qui représentent les objets et les scènes 3D). En effet, un objet 3D est modélisé par un polygone complexe, c'est à dire un maillage fait d'une liste de points 3D (3 coordonnées x,y et z). Pour appliquer des transformations (rotations, translations, homothéties) à cet objet, il faut appliquer les mêmes matrices de transformation (des matrices 4x4) à tous les points de cet objet : il y a un parallélisme massif de données. Ensuite, la projection 3D vers les deux dimensions de l'écran d'affichage repose également sur des matrices de transformation. La coloration et le texturage des objets (applications d'images sur des objets 3D) relèvent de techniques similaires qui reposent également sur un parallélisme massif de données. Pour exploiter ce parallélisme, l'architecture matérielle des GPU est construite à partir de dizaines, centaines voir milliers de processeurs relativement simples, capables d'effectuer des fonctions de calcul (ou "kernels") quasiment identiques sur des millions, voir milliards de données différentes.

Peu à peu, la flexibilité et la programmabilité des cartes GPU a été augmentée, permettant d'utiliser ces processeurs pour des applications plus générales que la chaîne 3D mais au prix d'une certaine complexité de développement. Ainsi est né le GPGPU : "general-purpose computing on graphics processing units". Les premiers GPU programmables n'étaient pas accompagnés d'un langage et d'un environnement de programmation dédié au calcul généraliste, il fallait donc programmer les GPU en codant les calculs dans des shaders (fonctions de calcul dédiées aux traitements géométriques et aux textures) en utilisant des languages comme Cg (C for Graphics). Puis les constructeurs de GPU ont développé des environnements plus généralistes comme CUDA[4] (par NVIDIA), ou OpenCl[5] (par un consortium d'industriels).

Cependant, pour tirer parti des performances offertes par ces architectures, il est indispensable de bien comprendre l'architecture interne de ces processeurs, et en particulier leur organisation mémoire. En effet, afin de résoudre les problèmes de bande passante dûs à des accès concurrents à la mémoire par tous les processeurs qui composent le GPU, son architecture repose sur des mémoires organisées hiérarchiquement. Cette organisation mémoire est parfaitement adaptée à la chaîne de rendu graphique 3D mais elle est contraignante pour la programmation d'applications non 3D car c'est au programmeur de placer les données dans les mémoires les

[4]https ://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html
[5]https ://www.khronos.org/opencl/

plus adaptées en fonction des types d'accès requis par l'application. Ici encore, il faut trouver la meilleure adéquation entre l'algorithme et l'architecture.

8.3.1 Tone Mapping HDR sur GPU

- *Stages Master 2 : Clément Chesnel puis Alexis Fischer*
- *Publication : [2]*

Au cours de deux stages successifs de Master 2 et d'une collaboration avec l'INRETS et le LCPC [6], nous avons réalisé l'implémentation d'un algorithme de Tone Mapping HDR sur PC (le premier stage M2) puis une implantation temps réel sur GPU (le second stage M2).

Le Tone Mapping a pour but de réduire la dynamique d'une image tout en préservant les détails et les nuances de couleurs. C'est une technique utilisée pour compenser la faible dynamique des dispositifs d'affichage actuels (écrans, vidéoprojecteurs, imprimantes etc.) qui permet donc de passer d'une image HDR (High Dynamic Range) à une image LDR (Low Dynamic Range). Dans notre cas, il s'agit de réduire la dynamique d'images de synthèse HDR pour les afficher dans les simulateurs de conduite de véhicules (automobile, vélo) de l'INRETS. Le tone mapping doit donc s'effectuer en temps réel.

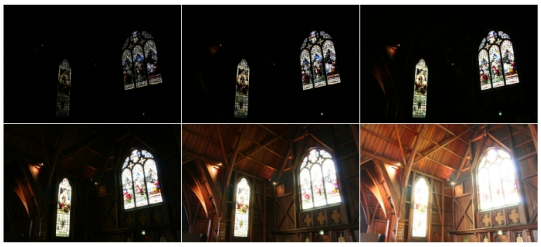


FIGURE 8.6 : 6 images d'une même scène capturées avec différents temps d'exposition (source Wikipedia)

FIGURE 8.7 : Image obtenue par algorithme de Tone Mapping des images de la figure 8.6 (source Wikipedia)

Pour donner un exemple visuel de tone mapping, la figure 8.7 présente une image résultant de l'application d'un algorithme de tone mapping sur un ensemble de 6 images d'une même scène capturée avec différentes sensibilités (figure 8.6). En

[6] qui ont fusionné successivement au sein de l'IFSTTAR (Institut Français pour le Transport l'Aménagement et les Réseaux), puis maintenant de l'UGE)

effet, comme il existe peu de capteurs HDR, les images HDR de scènes réelles sont souvent obtenues en fusionnant différents clichés d'une même scène capturée avec des temps d'exposition et des ouvertures (diaphragmes) différents.

Concrètement, le tone mapping revient à réduire le nombre de couleurs d'une image en réduisant l'amplitude de chacune des composantes couleur. Dans les formats courant d'images LDR (jpeg, tiff, etc.), chaque pixel est associé à 3 octets encodant chacun une couleur primaire (un octet pour le rouge, un pour le vert et un pour le bleu) ce qui permet donc 256 valeurs possibles pour chacune d'elles. Dans le cas des images HDR qui nous intéresse, chaque composante est codée par un nombre flottant de 32bits ce qui permet une très grande dynamique.

Pour réduire cette dynamique sans perdre les détails, nous avons implémenté sur PC et mesuré les performances de trois algorithmes réputés : l'opérateur de Ward [74], celui de Pattanaik [68] et celui de d'Irawan [48]. L'opérateur de Ward consiste à réduire l'amplitude de la luminance en modifiant l'histogramme de l'image tout en garantissant la reproduction des détails. Cependant, dans le cas de vidéos ou d'images générées informatiquement par nos simulateurs, cette méthode génère des artefacts gênants car elle traite chaque image indépendemment les unes des autres. C'est une méthode dite statique. Au contraire, l'algorithme de Pattanaik est dynamique, il repose sur une modélisation de l'adaptation de l'oeil face aux variations d'intensité lumineuse dans un flot d'images. Il a cependant l'inconvénient de compresser la dynamique de l'image et ainsi engendrer une perte de détails. L'algorithme de Irawan combine les deux précédents en gardant leurs points forts : préservation des détails et prise en compte d'un flux d'images.

Concrètement l'opérateur de Ward repose sur une fonction d'ajustement de l'histogramme nommée "TVI" (pour Threshold Versus Intensity). Irawan introduit une nouvelle fonction "TVIA" (pour Threshold Versus Intensity and Adaptation) qui ajoute l'adaptation de l'oeil à la fonction TVI. Nous avons donc intégré cette dernière dans le pipeline temps réel de rendu d'images de nos simulateurs. Ainsi, l'image HDR d'entrée est convertie dans l'espace XYZ[7] Ces données sont ensuite traitées en plusieurs étapes : ajustement de l'histogramme, seuillage de l'histogramme, interpolation de la luminance de l'histogramme seuillé puis transformation en RGB pour affichage. L'implantation sur Pentium 4 en utilisant son unité SIMD "SSE2" a seulement permis d'atteindre 1 à 2 images par seconde pour des images 1000x666 pixels, très en deça du temps réel que nous visions (25 images/seconde). Le profiling de l'exécution CPU a permis d'identifier les étapes les plus coûteuses : les changements d'espace de couleur (30% du temps pour RGB vers XYZ, 37% pour XYZ vers RGB) ainsi que l'interpolation de l'histogramme seuillé afin de créer la luminance de sortie. Dans ces 3 transformations, il n'y a pas de dépendances entre les données, elles ont donc été implémentées sur GPU en utilisant OpenGL et le langage Cg[8]. Cette implémentation a permis d'atteindre un débit très satisfaisant de 37 images par seconde.

[7] C'est un espace de couleur spécifique développé par la CIE (Commission Internationale de l'Eclairage) où la composante Y correspond quasiment à la luminance et les composantes X et Z à la chrominance d'un pixel de l'image. La conversion de l'espace RGB vers l'espace XYZ est similaire à celle de RGB vers YCbCr.

[8] Les langages Cuda et OpenCl n'étaient pas encore mûrs lors de ces travaux

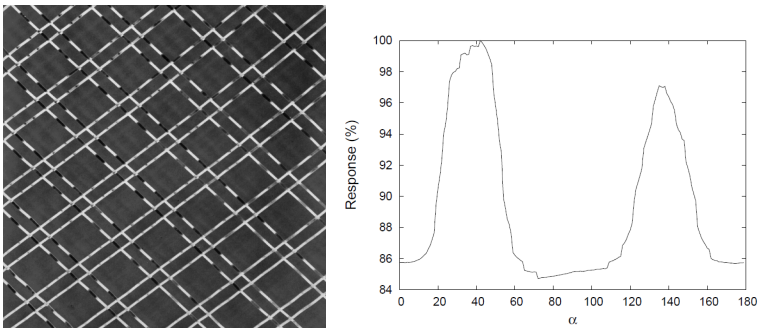


FIGURE 8.8 : Image d'entrée (à gauche) et scores en fonction des angles (à droite)

8.3.2 Analyse d'images avec opérateurs morphologiques sur GPU

- *Stage Master 2 (Erasmus) : Pavel Karas*
- *Co-encadrement avec E. Dokladalova*
- *Publication : "GPU implementation of linear morphological openings with arbitrary angle" [59]*

Il s'agit d'implanter un opérateur de filtrage morphologique d'images sur une architecture GPU afin d'accélérer son exécution. Le filtrage morphologique repose sur la morphologie mathématique qui est basée sur une description ensembliste des images. Plutôt que de s'intéresser à l'amplitude des signaux, on s'intéresse à la notion de forme. Les deux opérateurs de base sont l'opérateur d'*érosion* et l'opérateur de *dilation*. Ils utilisent une forme de référence (dite "élément structurant") pour faire des comparaisons locales. En les combinant il est possible de créer des opérateurs dits "d'ouverture" et de "fermeture". Il existe des versions à une dimension de ses opérateurs (1D, dit "opérateurs linéaires"), qui peuvent alors être utilisées sur des lignes parallèles de l'image, traitées potentiellement de façon concurrente.

Nous nous intéressons ici à l'implémentation GPU du filtre d'ouverture morphologique linéaire et plus particulièrement à son utilisation pour la mesure d'angles de segments de droite dans une image.

Pour cela, l'opérateur d'ouverture 1D est appliqué plusieurs fois à toutes les lignes d'une même image selon un angle donné. L'image 2D résultante est ensuite analysée pour fournir un score qui peut par exemple être la somme des intensités de tous les pixels résultants. Puis l'angle d'application de l'opérateur est modifié avant d'être appliqué pour obtenir une seconde image dont le score, lié à ce nouvel angle, est obtenu. Cet algorithme est appliqué de la même façon pour tous les angles et finalement, la comparaison des scores des images obtenues permet de déterminer les orientations majoritaires des segments voulus de l'image. La figure 8.8 présente un exemple d'image d'entrée et le spectre obtenu après application de l'algorithme pour des angles variant de 0 à 180 degrés. On constate bien les 2 pics pour les 2 angles visibles sur cette image.

Nous avons comparé plusieurs opérateurs d'ouverture linéaires existant en les implantant sur architecture GPU selon le schéma : un opérateur 1D par thread. La rotation s'effectuant à chaque fois en appliquant des accès mémoire selon des adresses calculées par l'algorithme de Bresenham. Nous avons obtenu une exécution 35 fois plus rapide sur une architecture GPU (nVIDIA Tesla C2050, 3Gb RAM, 1.15 GHz)

par rapport à une architecture CPU (Intel Core i7-870 2.93 GHz), pour des images 640x640 pixels et une latence de 60ms pour traiter les 180 angles.

8.3.3 Réalité virtuelle - Projet Urbanvision

- *Projet ISITE FUTURE - Université Paris Est*
- *Responsabilité du workpackage 4 - "Implémentation temps réel de la BRDF (Bidirectional Reflectance Distribution Function) sur GPU*
- *Deux stages Master 2 : Matthias Hudelot (2020) et Alexandre Sanchez (2022)*
- *Publication : Article sur l'implémentation temps réel de la BRDF sur GPU (en cours)*

Le projet Urbanvision (2019-2022) associait les laboratoires de recherche PICS-L et LIGM de l'Université Gustave Eiffel, de l'école des Ponts Paritech et de l'IGN. L'objectif était d'utiliser les techniques de reconstruction 3D basées sur la vision 3D pour générer automatiquement des environnements virtuels représentant un quartier de la ville dans lequel des agents réels et virtuels peuvent interagir. Nous voulions également simuler les performances visuelles des habitants, notamment en milieu difficile : la nuit, dans le brouillard, en présence d'éblouissement. Une base de données des propriétés photométriques des matériaux (BRDF) utilisés en milieu urbain a été créée et mise à disposition afin d'améliorer la qualité visuelle des environnements virtuels. Ce projet était organisé en 4 workpackages (WP) : le premier dédié au management du projet, le WP2 abordait les problèmes de vision par ordinateur (reconstruction 3D par photogrammétrie) pour la réalité virtuelle, le WP3 abordait la vision humaine. Enfin le WP4, sous ma responsabilité, avait pour but de modifier les étapes de calcul de rendu graphique des images produites par le logiciel Unity afin de prendre en compte les caractéristiques optiques de matériaux réels. Ces caractéristiques, appelées BRDF sont préalablement mesurées à l'aide d'un banc de mesure spécial développé par l'Université Gustave Eiffel[69].

L'objectif était donc de modifier le pipeline de rendu graphique pour qu'il puisse tenir compte des valeurs de BRDF mesurés. Pour cela les principaux verrous levés lors de ce projet étaient :

- d'analyser le pipeline de rendu graphique d'Unity : peu documenté, nous avons développé des outils d'analyse de code pour faire de la rétro-ingénierie,
- de stocker les valeurs des textures BRDF dans le GPU,
- modifier le pipeline de rendu pour la prise en compte de ses valeurs en fonction de la position des sources lumineuses et de l'observateur,
- implémenter ces modification dans le GPU afin de garantir une exécution temps réel,
- valider l'implémentation.

Nous disposons donc aujourd'hui d'une technique innovante de rendu graphique, testée sur GPU sous Unity, qu'il nous reste à décrire dans un article en cours de rédaction.

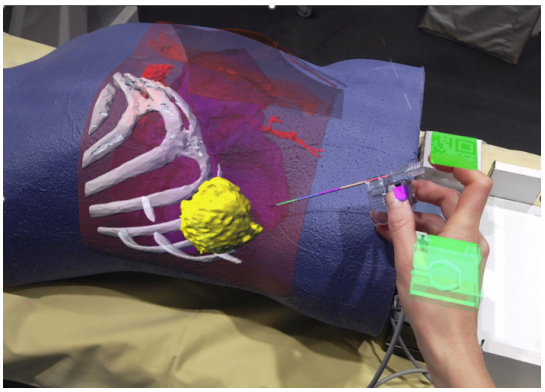


FIGURE 8.9 : Capture écran de ce que voit le praticien à travers le casque lorsqu'il réalise une biopsie de l'organe jaune grâce à l'aiguille qu'il tient dans sa main. La reconstruction virtuelle est affichée sur une reconstruction en polystyrène d'un buste humain (appelé "fantôme", en bleu) car les essais cliniques n'ont pas encore été autorisés.

8.4 Système de réalité augmentée pour Radiologie interventionnelle

- *Collaboration ESIEE - Hôpital Henri Mondor / Université Créteil - soutenue par Erganéo -*
- *Dispositif de réalité augmentée pour la radiologie interventionnelle*
- *Stage Master 2 : Christopher Picot (2018)*
- *Publications : un brevet [72], un article en cours*

Dans le cadre des mes enseignements j'ai pu créer une salle de réalité virtuelle puis encadrer de nombreux projets en réalité virtuelle et augmentée. Certaines réalisations sont présentées lors d'évènements internes et externes comme notamment les journées FUTURES de l'ISITE[9]. A la suite de ces présentations j'ai été contacté par Erganéo (société de transfert de technologie) pour aider à concrétiser les idées originales de deux radiologues interventionnelle de l'APHP Henri Mondor : Vania Tacher et Laetitia Saccenti, également rattachées à l'université de Créteil (UPEC).

La radiologie interventionnelle consiste à effectuer des gestes médicaux (ablation, biopsies, etc.) sous l'assistance d'un scanner à rayon X pour obtenir une précision suffisante et éviter des organes vitaux. Cependant, ce type d'opération induit une dose élevée de rayon X pour les patients mais aussi pour les radiologues qui doivent porter de lourdes protections en plomb pour se protéger.

L'objectif de notre dispositif est de minimiser les doses et réduire la nécessité du scanner pendant l'opération grâce à l'utilisation de la réalité augmentée. Pour cela, avant l'intervention, la partie ciblée du corps du patient est scannée par un scanner à rayon X. Les images obtenues sont ensuite segmentées puis assemblées de façon à pouvoir réaliser une reconstruction 3D des organes que l'on souhaite visualiser pendant l'opération. Pendant cette dernière, le radiologue interventionnelle porte un casque de réalité augmentée (type Hololens) qui permet d'afficher, en superposition au corps du patient (i.e. sous forme d'hologramme), la reconstruction 3D de ses organes. En parallèle, la position et l'orientation de l'instrument qu'utilise le praticien (aiguille en générale) sont mesurés en temps réel afin d'afficher en temps réel

[9] http ://www.future-isite.fr/

une version virtuelle de cet instrument dans le casque. Ainsi, même quand l'instrument n'est plus visible à l'oeil nu (parce qu'il est inséré dans le corps du patient), le praticien continue à le voir (grâce à son casque, Cf. figure 8.9) tout en pouvant aussi visualiser les organes du patient : cela évite au praticien de faire de nombreux scans pendant l'opération pour localiser son instrument et la proximité d'organes.

Ce dispositif a fait l'objet d'un brevet récent8.4, des publications sont maintenant possibles et donc en cours de soumission. Enfin, la startup Medisyst vient d'être créée pour exploiter ces travaux.

Chapitre 9

Conclusion

9.1 Synthèse des travaux effectués

9.1.1 Apports méthodologiques et outils

Grâce aux travaux effectués et présentés lors de la thèse de Linda Kaoune nous avons pu proposer un nouveau modèle pour les architectures à base de FPGA. Nous y avons également présenté des heuristiques qui utilisent ce modèle pour implanter efficacement des algorithmes applicatifs sur FPGA. Enfin nous avons présenté des règles de génération automatique de code à partir de graphes factorisés, conditionnés de dépendances de données.

La thèse d'Oussama Feki a permis d'étendre les travaux de Linda Kaouane aux architectures mixtes. Nous y proposons un cadre formel à base de graphes pour spécifier des architectures constituées de composants programmables et reconfigurables. Ce modèle permet l'optimisation et la génération automatique de code.

La thèse d'Elias Barbudo porte sur l'utilisation optimisée des architectures à gros grains (CGRA). Elle s'est effectuée en restant dans le cadre formel établi précédemment. Ces travaux nous ont permis de proposer un nouveau modèle permettant la spécification de ces CGRA sous la forme de graphes, et surtout leur programmation optimisée. Notre modèle générique permet la réutilisation de ce type d'architectures souvent développées par les industriels ou les chercheurs, pour des applications et des contextes très spécifiques, sans soucis initial de ré-utilisation.

Au final l'ensemble de ces travaux apportent un modèle et une méthodologie pour décrire assez précisément mais de façon générique, un grand nombre d'architectures hétérogènes. Cette approche permet ensuite d'optimiser l'utilisation de ces architectures en proposant chaque fois une approche Adéquation Algorithme Architecture.

9.1.2 Applications

Temps réel critique

A travers la thèse de Tristan Fautrel, nous avons pu proposer une approche qui permet l'implantation d'algorithmes temps réels sur des architectures virtualisées utilisant des hyperviseurs temps réel. Nous avons modélisé un hyperviseur ainsi que les machines virtuelles qu'il contrôle, et pour chaque machine virtuelle nous avons modélisé les tâches exécutées. Grâce à cette modélisation globale à deux niveaux, nous sommes capables de déterminer les paramètres de configuration de l'hyperviseur qui maximisent l'utilisation des processeurs (i.e. pour permettre de les utiliser au maximum).

Ces travaux de thèse ont été validés par leur mise en application au sein du projet FUI CEOS visant la construction d'un drone de surveillance autonome.

Les travaux effectués lors de la thèse, toujours en cours, de Florient Champenois, permettent de proposer un modèle de réseau distribué embarqué. Nous utilisons ce modèle basé sur les graphes pour construire la configuration la plus adaptée à nos contraintes. Pour cela nous avons proposé une approche de calcul de pire temps de communication, développé un algorithme génétique et proposé de nouvelles fonctions de coût. Ce dernier a été évalué et comparé à une approche exhaustive que nous avons également développée. Un dépôt de brevet associé à cette approche est en cours, levant bientôt l'interdiction de publier.

Temps réel souple

Lors de la thèse de Nejmeddine Bari nous avons utilisé la méthodologie Adéquation Algorithme Architecture pour réaliser l'implantation temps réel d'un encodeur vidéo sur une architecture multi-DSP.

L'encadrement de plusieurs stagiaires de Master et ma participation à plusieurs projets colloboratifs m'ont permis de réaliser l'implantation temps réel d'algorithmes de traitement d'images (analyse par morphologie mathématique, tone mapping, calcul de BRDF) sur architectures multi-coeurs et sur architecture GPU. Cela m'a aussi donné l'occasion de développer un système d'aide à la chirurgie par réalité augmentée, aujourd'hui breveté.

9.1.3 Bilan

L'objectif commun des différents travaux présentés dans ce manuscrit réside en l'implantation et l'optimisation d'applications temps réel sur des architectures matérielles adaptées. Ces applications appartiennent au temps réel critique (avionique notamment), ou souple (traitement des images). En fonction des algorithmes que contiennent ces applications il faut choisir l'architecture matérielle la plus adaptée, celle qui permettra de respecter les contraintes de l'application. Dans ces travaux se sont les contraintes de temps d'exécution et d'embarquabilité (minimisation des ressources) qui ont guidé les choix. Tout au long de ce document et de mes recherches, nous avons exploré différentes familles d'architectures matérielles : cela va des architectures les plus petites à base de micro-contrôleurs, jusqu'aux architectures GPU les plus puissantes, en passant par des architectures SIMD, multi-DSP, multi-coeurs ou bien plus spécialisées comme les architectures à base de FPGA.

Les communications entre les ressources matérielles ont également un rôle primordial, c'est pourquoi elles ont également été explorées pour les principaux protocoles.

L'implantation de ces applications, qu'elles soient temps réel strictes ou souples, a été chaque fois effectuée par une approche méthodologique Adéquation Algorithme Architecture en modélisant dans un même cadre formel l'algorithme, l'architecture, l'implémentation mais aussi la prédiction de performances pour choisir l'implémentation qui répond aux contraintes temps réel et d'embarquabilité de l'application.

9.2 Projet de recherche, perspectives

9.2.1 A court terme

Les résultats des travaux précédents n'ont pas encore tous été publiés, ainsi quatre articles sont en cours de finalisation. Ils portent sur l'implantation de la BRDF sur

GPU (projet UrbanVision), la modélisation d'architecture CGRA (thèse de Elia Barbudo), l'algorithme génétique de construction de réseaux embarqués (thèse de Florient Champenois) et le système d'aide à la chirurgie par réalité augmentée.

En parallèle, par le co-encadrement avec Jean Cousty (professeur ESIEE) et Benjamin Perret (professeur ESIEE) de la thèse de Quentin Lebon qui vient de débuter, je développe mon expertise en intelligence artificielle, tout en continuant à explorer l'implantation d'algorithmes sur architecture GPU. Celle thèse porte en effet sur la construction d'arbres des composantes connexes sur architecture GPU dans le but d'accélérer les calculs en IA.

9.2.2 A moyen terme

A moyen terme, un nouvel axe applicatif de recherche se dégage, celui de la cybersécurité. En parallèle, il me semble qu'à travers mon large champs applicatif et mon expertise en architecture, je pourrai également intervenir davantage dans l'animation scientifique et développer mes collaboration nationales et internationales.

Axe Cybersécurité

De part ma responsabilité de la filière cybersécurité de l'ESIEE, je suis au contact de nombreux industriels dont je peux recueillir les besoins actuels et futurs : un axe se dessine peu à peu, celui de la sécurité des systèmes temps réel embarqués encore insuffisamment étudié. En effet, l'urgence de la sécurisation des applications internet est telle que les moyens n'ont pas encore pu être alloués au même niveau pour la sécurisation des systèmes temps réel, qu'ils soient embarqués ou fixes (dans l'industrie). Jusqu'à récemment la sécurité était principalement garantie par la non accessibilité aux interfaces physiques de communication de ces systèmes. Cependant avec les besoins d'interopérabilité pour les échanges de données, d'interconnexion entre les réseaux, la sécurisation vis à vis des intrusions, des attaques exterieurs devient indispensable mais très complexe. Il s'agit donc de sécuriser les systèmes pour les rendre plus robustes en analysant par exemple le traffic en temps réel, pour détecter puis reconfigurer les systèmes immédiatement.

Cela requiert de la puissance de traitement et de surcroît une exécution temps réel dans la plupart des cas. Cela passe donc par d'identification et/ou le développement d' architectures spécifiques, taillées sur mesures à ces nouveaux besoins. Nous retrouvons donc dans une approche algorithme architecture, thème central de mes travaux présentés ici. Il s'agit également de prendre en compte les temps de traitement des unités de sécurisation spécialisées qui commencent à être ajoutés dans les processeurs, leur impact sur les performances et sur le déterminisme de l'exécution. Un large champs d'investigation doit être exploré.

D'ailleurs, pour ouvrir cette nouvelle extension de mes travaux, je travaille aujourd'hui au montage d'une thèse dans la cybersécurité des réseaux avioniques avec l'industriel Safran pour concevoir une architecture adaptée à la sécurisation des communications intra et inter-véhicules.

Animation scientifique

Au niveau local, étant donné le large champs applicatif de mes travaux, et le grand nombre d'architectures matérielles expérimentées, je suis maintenant fortement impliqué dans deux équipes du laboratoire LIGM : l'équipe A3SI (Algorithme, Architecture, Analyse et Synthèse d'Images) dont je fais partie, mais aussi l'équipe

LRT (Logiciels, Réseaux et Temps réel). En effet, je collabore régulièrement avec ses membres qui travaillent sur l'ordonnancement temps réel d'une part et ceux qui travaillent sur les réseaux et objets connectés d'autre part. Ainsi, n'étant pas le seul membre du laboratoire dont la recherche porte sur les architectures matérielles, il me semble intéressant de voir comment former une équipe transversale au laboratoire pour offrir notre expertise en architecture et en adéquation algorithme architecture.

Au niveau national, je prévois de renforcer mon implication dans la communauté Adéquation Algorithme Architecture, notamment en participant d'avantage au développement et l'animation de l'axe 6 "Adéquation Algorithme-Architecture, Traitements embarqués" du GDR ISIS. Cela pourrai par exemple s'effectuer par la rédaction d'un ouvrage commun dédié à cette thématique.

Au niveau international, travailler en synergie avec les équipes d'autres pays, permettrait de résoudre des problèmes d'envergure, en répondant par exemple ensemble à des appels à projet européens. Pour cela il me semble maintenant opportun de préparer un congé de perfectionnement que je n'ai encore jamais eu l'occasion d'effectuer et qui me permettrait de nouer de nouveaux contacts, identifier de nouveaux verrous et d'y répondre ensemble en étendant mes travaux sur la méthodologie AAA, leur implémentation dans des outils existants, développer de nouvelles applications en cybersécurité et IA.

Bibliographie

[1] Mohamed AKIL, Thierry GRANDPIERRE et Laurent PERROTON. « Real Time Dynamic Tone-Mapping Operator on GPU ». In : *Journal of Real-Time Image Processing* 7.3 (2012), p. 165-172. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00826330.

[2] Mohamed AKIL, Thierry GRANDPIERRE et Laurent PERROTON. « Real Time Dynamic Tone-Mapping Operator on GPU ». In : *Journal of Real-Time Image Processing* 7.3 (2012), p. 165-172. URL : https://hal.science/hal-00826330.

[3] Mohamed AKIL, Pierre NIANG et Thierry GRANDPIERRE. « A rapid prototyping methodology to implement and optimizing image processing algorithms for FPGAs ». In : *IS&T / SPIE, Symposium on Electronic Imaging Science and Technologies, Real-Time Imaging IX Conference*. 1. electronic version (10 pp.) San Jose, USA, France, 2006, 10pp. URL : https://hal.archives-ouvertes.fr/hal-00622356.

[4] Mohamed AKIL, Laurent PERROTON et Thierry GRANDPIERRE. « FPGA-based architecture for hardware compression/decompression of wide format images ». In : *Journal of Real-Time Image Processing* 1.2 (2006), p. 163-170. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00621972.

[5] Christine ANCOURT, Michel BARRETEAU, Bernard DION, T. GRANDPIERRE, Irigoin FRANÇOIS, Jean JOURDAN, Philippe KAJFASZ, C. LAVARENNE et Y. SOREL. « PROMPT : Placement rapide optimisé sur machines parallèles pour applications de télécommunications ». In : *Actes du 5ème Worshop sur l'Adéquation Algorithme Architecture*. Paris, France, jan. 2000. URL : http://www-rocq.inria.fr/syndex/publications/pubs/aaa100/aaa100.pdf.

[6] Nejmeddine BAHRI. « Étude et conception d'un encodeur vidéo H264/AVC de résolution HD sur une plateforme multicœur ». Theses. Université Paris-Est ; University of Sfax, nov. 2015. URL : https://pastel.archives-ouvertes.fr/tel-01275746.

[7] Nejmeddine BAHRI, Nidhameddine BELHADJ, Med Ali BEN AYED, Nouri MASMOUDI, Thierry GRANDPIERRE et Mohamed AKIL. « Real-time H264/AVC high definition video encoder on a multicore DSP TMS320C6678 ». In : *2015 International Conference on Computer Vision and Image Analysis Applications (ICCVIA)*. Computer Vision and Image Analysis Applications (ICCVIA), 2015 International Conference on. Sousse, Tunisia : IEEE, jan. 2015. DOI : 10.1109/ICCVIA.2015.7351893. URL : https://hal.archives-ouvertes.fr/hal-01797192.

[8] Nejmeddine BAHRI, Nidhameddine BELHADJ, Thierry GRANDPIERRE, Mohamed ALI BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « Real-time H264/AVC encoder based on enhanced frame level parallelism for smart multicore DSP camera ». In : *Journal of Real-Time Image Processing* 12 (nov. 2014), p. 791-812. DOI : 10.1007/s11554-014-0470-6. URL : https://hal.archives-ouvertes.fr/hal-01192770.

[9] Nejmeddine BAHRI, Thierry GRANDPIERRE, Med Ali Ben AYED, Nouri MASMOUDI et Mohamed AKIL. « Embedded Real-Time H264/AVC High Definition Video Encoder on TI's KeyStone Multicore DSP ». In : *Journal of Signal Processing Systems* 86.1 (2017), p. 67-84. DOI : 10.1007/s11265-015-1098-x. URL : https://hal.archives-ouvertes.fr/hal-01286949.

[10] Nejmeddine BAHRI, Thierry GRANDPIERRE, Med Ali BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « GOP level parallelism implementation for real-time H264/AVC video encoder on multicore DSP TMS320C6472 ». In : *2014 6th European Embedded Design in Education and Research Conference (EDERC)*. Education and Research Conference (EDERC), 2014 6th European Embedded Design in. Milano, France : IEEE, sept. 2014. DOI : 10.1109/EDERC.2014.6924378. URL : https://hal.archives-ouvertes.fr/hal-01797197.

[11] Nejmeddine BAHRI, Thierry GRANDPIERRE, Mohamed Ali BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « GOP level parallelism implementation for real-time H264/AVC video encoder on multicore DSP TMS320C6472 ». In : *EDERC 2014*. Sous la dir. d'IEEE. Proceedings of the 6th European Embedded Design Education and Research Conference. IEEE, EURASIP. Milan, Italy, nov. 2014, p. 152-156. DOI : 10.1109/EDERC.2014.6924378. URL : https://hal.archives-ouvertes.fr/hal-01192779.

[12] Nejmeddine BAHRI, Thierry GRANDPIERRE, Nouri MASMOUDI et Mohamed AKIL. « Optimisations structurelles et matérielles de l'encodeur vidéo H264/AVC sur un seul coeur d'un DSP multicoeurs TMS320C6472 ». In : *GRETSI 2013 (Symposium on Signal and Image Processing)*. BREST, France, 2013. URL : https://hal.archives-ouvertes.fr/hal-01797228.

[13] Nejmeddine BAHRI, Imen WERDA, Thierry GRANDPIERRE, Mohamed Ali BEN AYED, Nouri MASMOUDI et Mohamed AKIL. « Optimizations for real-time implementation of H264 AVC video encoder on DSP processor ». In : *International Review on Computers and Software (IRECOS)* 8.9 (sept. 2013), p. 2025-2035. URL : https://hal.archives-ouvertes.fr/hal-01192792.

[14] Elias BARBUDO, Eva DOKLADALOVA et Thierry GRANDPIERRE. « A New Modelling Framework for Coarse-Grained Programmable Architectures ». In : *Compas 2020*. accepted for 2020, presentation delayd to 2021. Lyon, France, juin 2021. URL : https://hal.archives-ouvertes.fr/hal-03108479.

[15] Elias BARBUDO, Eva DOKLADALOVA et Thierry GRANDPIERRE. « A New Modelling Framework for Coarse-Grained Programmable Architectures ». In : *Compas 2020*. Lyon, France, juin 2021. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-03108479.

[16] Elias BARBUDO, Eva DOKLADALOVA et Thierry GRANDPIERRE. « Modelling and Mapping Framework for Coarse-GrainedProgrammable Architectures ». In : *14th Summer School on Modelling and Verification of Parallel Processes*. (online), France, juin 2020. URL : https://hal.archives-ouvertes.fr/hal-03108451.

[17] Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A Mapping Methodology for Coarse-Grained Pipelined Configurable Architectures ». In : *14th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2019)*. Renesse, Netherlands, juin 2019. URL : https://hal.archives-ouvertes.fr/hal-02104162.

[18] Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A Mapping Methodology for Coarse-Grained Pipelined Configurable Architectures ». In : *14th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2019)*. Renesse, Netherlands, juin 2019. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-02104162.

[19] Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A mapping tool for configurable pipeline co-processors ». In : *Colloque National du GDR SoC-SiP*. GDR-SOC-SIP. Paris, France, juin 2018. URL : https://hal-enpc.archives-ouvertes.fr/hal-01801018.

[20] Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A mapping tool for configurable pipeline co-processors ». In : *Colloque National du GDR SoC-SiP*. GDR-SOC-SIP. Paris, France, juin 2018. URL : https://hal-enpc.archives-ouvertes.fr/hal-01801018.

[21] Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A New Mapping Methodology for Coarse-Grained Programmable Systolic Architectures ». In : *22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES 2019)*. St Goar, Germany, mai 2019. DOI : 10.1145/3323439.3323982. URL : https://hal.archives-ouvertes.fr/hal-02013560.

[22] Elias BARBUDO, Eva DOKLADALOVA, Thierry GRANDPIERRE et Laurent GEORGE. « A New Mapping Methodology for Coarse-Grained Programmable Systolic Architectures ». In : *22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES 2019)*. St Goar, Germany, mai 2019. DOI : 10.1145/3323439.3323982. URL : https://hal.archives-ouvertes.fr/hal-02013560.

[23] Elias BARBUDO FRANCO. « Towards Efficient Reuse of Software Programmable Streaming Coarse Grained Reconfigurable Architectures ». Theses. Université Gustave Eiffel, juin 2021. URL : https://tel.archives-ouvertes.fr/tel-03551361.

[24] Michel BARRETEAU, Juliette MATTIOLI, Thierry GRANDPIERRE, Christophe LAVARENNE, Yves SOREL, Philippe BONNOT et Philippe KAJFASZ. « PROMPT : A mapping environment for telecom applications on System on a Chip ». In : *International conference on Compilers, architecture, and synthesis for embedded systems*. San Jose, United States : ACM Press, nov. 2000, p. 41-47. DOI : 10.1145/354880.354887. URL : https://hal.science/hal-03949779.

[25] Jan BARTOVSKY, Petr DOKLÁDAL, Matthieu FAESSEL, Eva DOKLADALOVA et Michel BILODEAU. « Morphological Co-Processing Unit for Embedded Devices ». In : *Journal of Real-Time Image Processing* (juill. 2015), pp. 1-12. DOI : 10.1007/s11554-015-0518-2. URL : https://hal-mines-paristech.archives-ouvertes.fr/hal-01117406.

[26] Enrico BINI et Giorgio C BUTTAZZO. « Biasing effects in schedulability measures ». In : *Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004.* IEEE. 2004, p. 196-203.

[27] Mohamed BOUBAKER, Mohamed AKIL, Khaled BEN KHALIFA, Thierry GRANDPIERRE et Mohamed HEDI BEDOUI. « Implementation of an LVQ neural network with a variable size : algorithmic specification, architectural exploration and optimized implementation on FPGA devices ». In : *Neural Computing and Applications* ISSN 0941-0643 (sept. 2009), 10pp. URL : https://hal.archives-ouvertes.fr/hal-00622461.

[28] Baptiste DELPORTE, Laurent PERROTON, Thierry GRANDPIERRE et Jacques TRICHET. « Accelerometer and Magnetometer Based Gyroscope Emulation on Smart Sensor for a Virtual Reality Application ». In : *Sensor and Transducers Journal* 14-1.Special Issue ISSN 1726-5479 (mars 2012), p32-p47. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00826243.

[29] Ailton DIAS. « Contribution 'a l'implantation optimis´ee d'algorithmes bas niveau de traitement du signal et des images sur des architectures mono-FPGA 'a l'aide d'une m´ethodologie d'Ad´equation Algorithme Architecture ». Thèse de doct. Université Paris Sud Orsay, juill. 2000.

[30] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC MPEG & ITU-T VCEG), —Advanced video coding for generic audiovisual services,|| Joint Video Team (JVT)*. Rapp. tech. http ://www.itu.int/rec/T-REC-H.264-201402-I/en, fév. 2014.

[31] Friedrich EISENBRAND, Nicolai HÄHNLE, Martin NIEMEIER, Martin SKUTELLA, José VERSCHAE et Andreas WIESE. « Scheduling periodic tasks in a hard real-time environment ». In : *Automata, languages and programming* (2010), p. 299-311.

[32] Yareb ELOUMI, Mohamed AKIL, Thierry GRANDPIERRE et Mohamed Hédi BEDOUI. « Latency and power optimization in AAA methodology for integrated circuits ». In : *ICECS 2010*. Sous la dir. d'IEEE. IEEE. Athens, Greece, déc. 2010, p. 639-642. DOI : 10.1109/ICECS.2010.5724593. URL : https://hal.archives-ouvertes.fr/hal-01192801.

[33] T FAUTREL, L GEORGE et F FAUBERTEAU. « A hypervisor schedulability analysis for safety and security critical applications scheduled in arbitrary patterns of slots ». In : *Proc. of JRWRTC* (2017).

[34] Tristan FAUTREL. « Analyse et ordonnancement d'un système hiérarchique virtualisé composé d'applications temps réel strictes ». Theses. Université Gustave Eiffel, mai 2021. URL : https://theses.hal.science/tel-03545962.

[35] Tristan FAUTREL, Laurent GEORGE, Frederic FAUBERTEAU et Thierry GRANDPIERRE. « An hypervisor approach for mixed critical real-time UAV applications ». In : *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. Kyoto, France : IEEE, mars 2019, p. 985-991. DOI : 10.1109/PERCOMW.2019.8730705. URL : https://hal.science/hal-03578489.

[36] Oussama FEKI. « Contribution à l'implantation optimisée de l'estimateur de mouvement de la norme H.264 sur plates-formes multi composants par extension de la méthode AAA ». Theses. Université Paris-Est ; University of Sfax, mai 2015. URL : https://pastel.archives-ouvertes.fr/tel-01271884.

[37] Oussama FEKI, Thierry GRANDPIERRE, Nouri MASMOUDI et Mohamed AKIL. « Optimized Implementation of H.264/AVC Motion Estimation on a Mixed Architecture Using SynDEx-Mix ». In : *International Review on Computers and Software (IRECOS)* 11.5 (mai 2016). DOI : 10.15866/irecos.v11i5.8764. URL : https://hal.archives-ouvertes.fr/hal-01800584.

[38] Oussama FEKI, Thierry GRANDPIERRE, Nouri MASMOUDI et Mohamed AKIL. « Optimized Implementation of H.264/AVC Motion Estimation on a Mixed Architecture Using SynDEx-Mix ». In : *International Review on Computers and Software (IRECOS)* 11.5 (mai 2016). DOI : 10.15866/irecos.v11i5.8764. URL : https://hal.archives-ouvertes.fr/hal-01800584.

[39] feki oussama feki, Thierry GRANDPIERRE, Mohamed AKIL et Nouri MASMOUDI. « Automatic Hardware/Software interface generation for SynDEx-mixte ». In : *ATSIP 2014*. International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). Sousse, Tunisia : IEEE, mars 2014, p. 512-516. DOI : 10.1109/ATSIP.2014.6834668. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-01192824.

[40] T. GRANDPIERRE. « Visite virtuelle des salles blanches ESIEE ». In : *Journal sur l'enseignement des sciences et technologies de l'information et des systèmes* 18 (2019), p. 1002. DOI : 10.1051/j3ea/20191002. URL : https://hal.science/hal-03948432.

[41] T. GRANDPIERRE, G. FLEUTOT, N. PARENT et Mooncheol WON. *A joystick driving algorithm with a collision stop feature on an electric vehicle (Cycab)*. Mars 2003. DOI : 10.1109/IVS.2002.1188000. URL : https://hal.science/hal-03949756.

[42] T. GRANDPIERRE et Y. SOREL. « Un nouveau modèle générique d'architecture hététogène pour la méthodologie AAA ». In : *Actes des Journées Francophones sur l'Adéquation Algorithme Architecture, JFAAA'02*. Monastir, Tunisia, déc. 2002. URL : http://www-rocq.inria.fr/syndex/publications/pubs/jfaaa02/jfaaa02.pdf.

[43] Thierry GRANDPIERRE, Christophe LAVARENNE et Yves SOREL. *Modèle d'exécutif distribué temps réel pour SynDEx*. Research Report RR-3476. Projet SOSSO. INRIA, 1998. URL : https://hal.inria.fr/inria-00073213.

[44] Thierry GRANDPIERRE, Christophe LAVARENNE et Yves SOREL. « Optimized Rapid Prototyping for Real-Time Embedded Heterogeneous Multiprocessors ». In : *the seventh international workshop*. Rome, France : ACM Press, 1999. DOI : 10.1145/301177.301489. URL : https://hal.archives-ouvertes.fr/hal-01800625.

[45] Thierry GRANDPIERRE, Christophe LAVARENNE et Yves SOREL. « Optimized Rapid Prototyping for Real-Time Embedded Heterogeneous Multiprocessors ». In : *the seventh international workshop*. Rome, France : ACM Press, 1999. DOI : 10.1145/301177.301489. URL : https://hal.archives-ouvertes.fr/hal-01800625.

[46] Thierry GRANDPIERRE et Yves SOREL. « From Algorithm and Architecture Specifications to Automatic Generation of Distributed Real-Time Executives : a Seamless Flow of Graphs Transformations ». In : *2003 1st IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2003)*. Mont Saint Michel, France : IEEE, juin 2003. DOI : 10.1109/MEMCOD.2003.1210097. URL : https://hal.archives-ouvertes.fr/hal-01800622.

[47] Werda IMEN. « Implémentation et Optimisation de la norme de codage vidéo H.264 sur une carte à base de DSP de la famille C6000 ». Thèse de doct. Université de Sfax, École Nationale d'Ingénieurs de Sfax, jan. 2010.

[48] Piti IRAWAN, James A. FERWERDA et Stephen R. MARSCHNER. « Perceptually based tone mapping of high dynamic range image streams ». In : *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. Eurographics Symposium on Rendering. 2005, p. 231-242.

[49] Mireille JACQUESON, Andre MOGOUTOV et Thierry GRANDPIERRE. « Device for projecting images on e.g. soap bubble in smoke, has optical sensor i.e. video camera, for detecting position of bubble screens, and projector for projecting images on screens, where sensor includes infra-red ray emitter ». 01. 2014.

[50] L. KAOUANE, M. AKIL, Y. SOREL et T. GRANDPIERRE. « A methodology to implement real-time applications on reconfigurable circuits ». In : *Proceedings of International Conference on Engineering of Reconfigurable Systems and Algorithms, ERSA'03*. Las Vegas, USA, juin 2003. URL : http://www-rocq.inria.fr/syndex/publications/pubs/ersa03/ersa03.pdf.

[51] L. KAOUANE, M. AKIL, Y. SOREL et T. GRANDPIERRE. « Implantation optimisée sur circuit dédié d'algorithmes spécifiés sous la forme d'un Graphe Factorisé de Dépendances de Données : application aux traitements d'images ». In : *Proceedings of 19th Synposium on Signal and Image Processing, GRETSI'03*. Paris, France, sept. 2003. URL : http://www-rocq.inria.fr/syndex/publications/pubs/gretsi03/gretsi03.pdf.

[52] Linda KAOUANE, Mohamed AKIL, Thierry GRANDPIERRE et Yves SOREL. « A methodology to implement real-time applications on reconfigurable circuits ». In : *Special issue on Engineering of Configurable Systems of the Journal of Supercomputing* 30.1 (2004), p. 283-301. URL : https://hal.archives-ouvertes.fr/hal-00622081.

[53] Linda KAOUANE, Mohamed AKIL, Thierry GRANDPIERRE et Yves SOREL. « A methodology to implement real-time applications on reconfigurable circuits ». In : *Journal of Supercomputing* 30.3 (2004). Special Issue Engineering of Configurable Systems, p. 283-301. DOI : 10.1023/B:SUPE.0000045213.82276.8e. URL : https://hal.archives-ouvertes.fr/hal-00826258.

[54] Linda KAOUANE, Mohamed AKIL, Thierry GRANDPIERRE et Yves SOREL. « A methodology to implement real-time applications on reconfigurable circuits ». In : *Special issue on Engineering of Configurable Systems of the Journal of Supercomputing* 30.1 (2004), p. 283-301. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622081.

[55] Linda KAOUANE, Mohamed AKIL et Yves SOREL. « Optimized implementation of application specific integrated circuits specified with data dependence graph ». In : *EDAA PhD Forum at DATE03, Design Automation and Test in Europe*. 1. Munich, Germany, France, mars 2003, 10pp. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622123.

[56] Linda KAOUANE, Mohamed AKIL, Yves SOREL et Thierry GRANDPIERRE. « From Algorithm Graph Specification to Automatic Synthesis of FPGA Circuit : A Seamless Flow of Graphs Transformations ». In : *Field Programmable Logic and Application, 13th International Conference*. Lisbon, Portugal, sept. 2003, p. 934-943. URL : https://hal.archives-ouvertes.fr/hal-01800628.

[57] Linda KAOUANE, Mohamed AKIL, Yves SOREL et Thierry GRANDPIERRE. « From Algorithm Graph Specification to Automatic Synthesis of FPGA Circuit : A Seamless Flow of Graphs Transformations ». In : *Field Programmable Logic and Application, 13th International Conference*. Lisbon, Portugal, sept. 2003, p. 934-943. URL : https://hal.archives-ouvertes.fr/hal-01800628.

[58] Pavel KARAS, Vincent MORARD, Jan BARTOVSKY, Thierry GRANDPIERRE, Eva DOKLADALOVA, Petr MATULA et Petr DOKLÁDAL. « GPU Implementation of Linear Morphological Openings with Arbitrary Angle ». In : *Journal of Real-Time Image Processing* 10.1 (2015), p. 27-41. DOI : 10.1007/s11554-012-0248-7. URL : https://hal.archives-ouvertes.fr/hal-00680904.

[59] Pavel KARAS, Vincent MORARD, Jan BARTOVSKY, Thierry GRANDPIERRE, Eva DOKLADALOVA, Petr MATULA et Petr DOKLÁDAL. « GPU Implementation of Linear Morphological Openings with Arbitrary Angle ». In : *Journal of Real-Time Image Processing* 10.1 (2015), p. 27-41. DOI : 10.1007/s11554-012-0248-7. URL : https://hal.science/hal-00680904.

[60] Michel KOENIG, Jean Michel BOUDENNE, P. LEGRIEL, A. BENUZZI, T. GRANDPIERRE, Dimitri BATANI, Simone BOSSI, Sonia NICOLELLA et René BENATTAR. « A computer driven crystal spectrometer with charge coupled device detectors for x-ray spectroscopy of laser plasmas ». In : *Review of Scientific Instruments* 68.6 (mars 1997), p. 2387-2392. DOI : 10.1063/1.1148122. URL : https://hal.science/hal-03948342.

[61] Kaouane LINDA. « Formalisation et optimisation d'applications s'exécutant sur architecture reconfigurable ». Thèse de doct. Université Paris Est, déc. 2004.

[62] K. MABASA, Mohamed AKIL, Thierry GRANDPIERRE, B.J. van WYK et M.A. van WYK. « Automatic VHDL Code Generation for Fuzzy Logic Systems ». In : *African Journal of Science and Technology* 1 (2008), 10pp. URL : https://hal.archives-ouvertes.fr/hal-00622125.

[63] Petr MATAS, Eva DOKLADALOVA, Mohamed AKIL, Thierry GRANDPIERRE, Laurent NAJMAN, M. POUPA et V. GEORGIEV. « Parallel Algorithm for Concurrent Computation of Connected Component Tree ». In : *Advanced Concepts for Intelligent Vision Systems (ACIVS'08)*. T. 5259/2008. Lecture Notes in Computer Science 1. France : Springer-Verlag, oct. 2008, p. 230-241. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622406.

[64] Petr MATAS, Eva DOKLADALOVA, Mohamed AKIL, Thierry GRANDPIERRE, Laurent NAJMAN, M. POUPA et V. GEORGIEV. « Parallel Algorithm for Concurrent Computation of Connected Component Tree ». In : *Advanced Concepts for Intelligent Vision Systems (ACIVS'08)*. T. 5259/2008. Lecture Notes in Computer Science 1. France : Springer-Verlag, oct. 2008, p. 230-241. URL : https://hal.science/hal-00622406.

[65] Pierre NIANG, Thierry GRANDPIERRE et Mohamed AKIL. « Implementing Real-Time Algorithms by using the AAA Prototyping Methodology ». In : *Embedded System Design, Techniques and Trends*. IFIP. Springer, 2007, p. 27-36. URL : https://hal.archives-ouvertes.fr/hal-00622231.

[66] Pierre NIANG, Thierry GRANDPIERRE, Mohamed AKIL et Yves SOREL. « AAA and SynDEx-Ic : A Methodology and a Software Framework for the Implementation of Real-Time Applications onto Reconfigurable Circuits ». In : *International Embedded Systems Symposium*. 1. France, 2007, 10pp. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622183.

[67] Feki OUSSAMA, Thierry GRANDPIERRE, Mohamed AKIL et Nouri MASMOUDI. « Automatic Hardware/Software interface generation for SynDEx-mixte ». In : *ATSIP 2014*. International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). Sousse, Tunisia : IEEE, mars 2014, p. 512-516.

DOI : 10.1109/ATSIP.2014.6834668. URL : https://hal.archives-ouvertes.fr/hal-01192824.

[68] Sumanta N. PATTANAIK, Hector Yee JACK TUMBLIN et Donald P. GREENBERG. « ATime-dependent visual adaptation for fast realistic image display ». In : *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. IEEE. 2000, p. 47-54.

[69] E. SAINT JACQUES, E. DUMONT et C. VILLA. « Characterisation of the reflection properties of road surfaces using an in-lab gonioreflectometer ». In : *in Proc. midterm sesion of the Commission Internationale de l'Éclairage*. Jeju, Korea : Commission Internationale de l'Éclairage, 2017.

[70] R. SAOULI, Mohamed AKIL et Thierry GRANDPIERRE. « Load Balancing and Static Placement/Scheduling Heuristic on Distributed Heterogeneous Architecture ». In : *International Review on Computers and Software (IRECOS)* 1 (2007), p. 227-234. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00622384.

[71] Rachida SAOULI, Mohamed AKIL et Thierry GRANDPIERRE. « Learning System for Defactorization Factor Classification of Factorized Data Dependence Graph ». In : *IJACT : International Journal of Advancements in Computing Technology* 3.4 (2011), p. 1-13. URL : https://hal-upec-upem.archives-ouvertes.fr/hal-00826253.

[72] Vania TACHER, Christopher PICO, Hicham KOBEITER, Thierry GRANDPIERRE et Laetitia SACCENTI. « Interventional radiology medical device for real-time guidance of a medical operating needle in a volume ». 01. 2021.

[73] Grandpierre THIERRY. « Modélisation d'architectures parallèles hétérogènes pour la génération automatique d'exécutifs distribuées temps réel optimisés ». Theses. Université Paris Sud Orsay, nov. 2000. URL : https://theses.hal.science/tel-03545962.

[74] G. W.WARD-LARSEN, H. RUSHMEIER et C. PIATKO. « A visibility matching tone reproduction operator for high dynamic range scenes ». In : *Visualization and Computer Graphics, IEEE Transactions on*. IEEE. 1997, p. 291-306.

[75] WD WEI et CL LIU. « On a periodic maintenance problem ». In : *Operations Research Letters* 2.2 (1983), p. 90-93.

[76] Imen WERDA, Taheni DAMMAK, Thierry GRANDPIERRE, Mohamed Ali BEN AYED et Nouri MASMOUDI. « Real-time H.264/AVC baseline decoder implementation on TMS320C6416 ». In : *Journal of Real-Time Image Processing* 7.4 (déc. 2012), p. 215-232. DOI : 10.1007/s11554-010-0181-6. URL : https://hal.archives-ouvertes.fr/hal-01192810.